Universitat de Vic
Escola Politècnica Superior

Treball Final de Màster

Application of image recognition for mobile phones with Android.

Anna Sobolewska

Màster en Tecnologies Aplicades de la Informació
Director/a: Ramon Reig i Albert Baucells

Vic, abril de 2011

I would like to dedicate this work to my father, who has always been my authority and guide in professional and personal life and always, independent of everything, believied in me.

TABLE OF CONTENTS

REGISTRE D'ENTRADA
Núm.:.....
DATA:.....

Universitat de Vic
Escola Politècnica Superior

Resum del Treball Final de Màster
Màster en Tecnologies Aplicades de la Informació

Títol: Application of image recognition for mobile phones with Android.
Paraules clau: Android, image processing, image recognition, colour recognition, Java, Euro banknote
Autor: Anna Sobolewska
Direcció: Ramon Reig and Albert Baucells
Avalador:
Data: 6.09.2010

Resume

Image processing algorithms nowadays are very popular area of research. This is the area that gives hide possibilities to discover new algorithms and ways of optimization. What is more, it is very important subject in medicine, real time application and normal life, areas, where everyday people use more and more computer vision technology and artificial intelligence.

The biggest problem of image processing algorithms is them complexity and image limitations. Some of them are more universal and some needs very specific and precise input image.

The main aid of this diploma project is to create an application for Euro banknote recognition with new algorithms developed and created by the author and other popular and well-known methods. The project shows that it is possible to implement sophisticated, greedy algorithms on mobile phone, with few memory and weak CPU and make complex tasks with simple tools and methods. The program is universal and does not need precised input image, in this way everybody with Android mobile phone or personal computer can try it.

The idea of application is simple: the user make a photo of Euro banknote. Banknote can be rotated, but it has to be on the plain background. The application analyses the image step-by-step, showing every step of algorithm, or directly returning the result of recognition.

The project has two parts: documentation and implementation part. Documentation part is divided into descriptive and technical documentation. Implementation also has two versions. One is and application for Euro banknote recognition for Android OS and another version of application is for all other operating systems like Windows, Linux and MacOS. Calculations code is exactly the same, the difference is just between graphical user interfaces.

1. INTRODUCTION

Image processing algorithms very often require a lot of memory and CPU. The easiest and the most common way to implement a sophisticated algorithm for mobile device is to exclude the calculations part of algorithm to some external server. In this way it is possible to save mobile resources, but it is always necessary to have access to the network and the server. Another way to do this, is to implement all in a mobile device, where resources are limited and does not have enough memory, CPU and battery to make greedy calculations. Therefore is very important to use the fastest algorithms inside of mobile device.

The main aim of the project is to create a program for Euro banknote recognition for mobile phones with Android operating system, that would be universal and could work with all images, even if their quality is not perfect. The general idea is that the user makes a photo of a Euro banknote by mobile phone and the program analyses the photo and as a result returns the denomination of the banknote. There should also be a possibility to execute a program in a debug mode, where the user can control all steps of used algorithms. In this way the project brings research values. As such, the project is also educational.

However, the biggest research value does not have an application, but algorithms used for the banknote recognition. They can be used in every program for image recognition because they are universal and quite fast. Unquestionable advantage of this work is its innovative and orginality. All parameters used in algorithms are effect of analysis of statistics for more than 200 banknotes. All algorithms are dinamic and can adapt to the image.

The program is completely written in Java with Android SDK and JJIL library for image processing All tests were made on Nexus One from Google with

Android 2.1 operating system, 512 MB of RAM memory and 1 GHz CPU. The application for other operating systems (Windows, Linux and MacOS) uses also JJIL library (in the same way that Android application), the only difference is user interface that in this case is made in Swing library (standard graphical library for JavaSE).

Android is one of the most popular operating systems for mobile devices like phones, tablets and smart-phones The system is based on Linux kernel and is released on the open source licence, so the source code is available for everyone to use[2]. Nowadays, Android programming every day is more popular and more widely used in other areas, not just in mobile phones. On the market are present tablets and personal computers with Android OS. The system is written mainly in Java and almost all main Java libraries are available to Android system too.

One of those libraries is JJIL (Jon's Java Imaging Library), wrote in different versions for Android, J2SE and J2ME. The library includes two sample applications of image recognition. One of them implements the bar code's reader, where bar codes from photo are recognised and read and another one, more sophisticated, for face recognition. The both are working only with mobile phone resources. The structure of Java classes in the library is very easy to extend. Some of the algorithms used in this project are implemented in JJIL library and some are author's implementation.

The project is very closely related to classes of Master De Tecnologies Aplicades en Informació, especially to two of them: "Processament de Dades Multidimensionals", where basis of algorithms for image processing ware detailed and "Sistemes Encastats en Temps Real", where students implemented an application in Android SDK environment.

The thesis is organised in nine parts. The first one explains a little Android environment, activities and JJIL library functions.

Parts from 3 to 6 describe steps of algorithm, where can be found details of algorithm of banknote localisation, rotation and denomination, hologram and colour recognition. Every of those chapters at the end has small resume, where every algorithm is explained step-by-step. The problem of recognition of the Euro banknote is divided into three smaller problems: locating the banknote, setting it in correct position and identifing the denomination of banknote by the colour, number or hologram. The application can detect just one banknote, so the photo has to be taken on plain background without any other big shape on the photo (however, there can be some noise though).

In chapter 7 are presented final results of the application for 200 Euro banknotes in different position and with different characteristics of light and background. There are both: global and individual statistics for every denomination.

In chapter 8 are presented possible ways for improving the efficiency and speed of the program and some methods of optimisation currently used in the application for Euro recognition like code optimisation, fast rotation, resizing.

The last chapter is a place for a final conclusions, where is author's self evaluation of the results and research values of this diploma project.

Objectives

The main objective of this diploma project is to discover something that put new values in science. For this reason in this work efficiency is not the most important, also because the problem of Euro recognition can be simplified and in easy way efficiency can arrive to near to 100%. The most important in this project are new, fresh ideas of fast algorithms, that can be used to

solve other problems and are enough fast for working with limited resources.

In the project is presented a sample application that uses only algorithms that can work just with mobile resources and some ways and ideas of their optimization. The main aim was to create a program, which will be unique and fresh. For this reason, in this project are used author's algorithms with a lot of them being faster than the standard solutions. In this way the project is gaining the nature of scientific research. The algorithms that are used in the main part are basic and fast transformations such as resizing, rotation and statistics methods like histograms. Thanks to this, the application can work on mobile phone without any problem.

The biggest issue of image processing is that every image is eternal universe, every one is different and even if for a man it can be very easy to recognise, for a machine it is a really difficult task. For this reason image recognition algorithms is a difficult task, especially when there is no available large computing power.

Current research

There are many documents and articles that describes similar problem of Euro banknote recognition. Anyone of them does not use the mobile phone resources. Usually they use sophisticated algorithms like neuronal networks and complex mathematical calculations and need large computing power.

In [20] authors use two kind of neuronal networks for recognise the pattern of Euro banknote. They used a three-layered perceptron for pattern recognition and Radial Basis Function network for data validation. The input data is a photo in two kind of light: green and IR (infra-red). They treat all banknote as a pattern so they do not check only small areas like in this

thesis, but all area of the banknote. The % of correctly recognised images is near to 100. Although the algorithm is very exact, the time of calculations is too big to implement in real time machines. The position of the banknote is exact and light parameters are always the same, so input images are always very similar, what simplifies the general problem.

Another paper where authors describes Euro banknote recognition algorithms is [23]. There are also used neuronal networks for looking for a pattern. In this case the algorithm uses just a part of the image of banknote as a pattern. This step decreases significantly time of execution. Them algorithm has very good rate of recognition between 95-100% depending on the denomination. Localisation of denomination is static too.

Some other solutions of Euro banknote recognition is recognition by size or by colour. This is especially good solution for counting machines, because is very fast and simple. However, it is assumed that inserted banknote has to be authentic.

For mobile phones Google created a program Goggle, that can recognise different objects like famous landmarks, store fronts, artwork, and popular images found online. The application also recognises Euro banknotes. After capturing the image the program sends it to one of servers of Google, where the image is processed and then the final result is returned to the user. In this way all sophisticated calculations are made in the server with large computational power. Google, as a main Internet solutions provider, has also very huge database of images that surly is used as a pattern database.

All of solutions presented above need high calculation time and resources. Some of them need specific input and no one is executed only on the mobile phone resources without the Internet connection. All of them need a help of external, big servers, that are dedicated to do sophisticated calculations.

2. ANDROID OPERATING SYSTEM

Android is a software stack for mobile devices that includes an operating system, middle-ware and key applications. The Android SDK (Android software development kit) provides tools and APIs necessary to begin developing applications on the Android platform using the Java programming language[3]. Android SDK is easy to integrate with Eclipse, that is one of the most popular programming environments for Java. Eclipse is a program, that allow to create, compile and debug Java applications and integrate a lot of useful tools for Java developer. All instructions on how to install Android SDK with Eclipse IDE can be found here: http://developer.android.com/sdk/installing.html

Android application is a program written in Java, that is separated from operating system and use only resources that are necessary to work and are declared before in special configuration file AndroidManifest.xml. This gives a better level of security and assure that application will not affect the core of the system. Thanks to this separation it is very difficult to write a virus or any other malicious program that affect Android system core. In case of Euro banknote recognition application is used just Camera (android.hardware.camera) and SDK (Software Development Kit).

Inside of the Android application can be distinguished other components, that are application building blocks. These are:

- **Activities** – that represents a single screen with a user interface. Euro recognition has nine activities. MainActivity that is the main window of application, MyGallery that opens Gallery and goes to look for a photo there, OpenCamera that take photo from Camera stream, MyView that shows final result and NextActivities that represent

states of image in Debug mode. Every NextActivity shows different part of algorithm and currently processing image.

- **Services** – useful in a long-running processes and in remote processes. It is a background process without user interface. No use in Euro recognition program.

- **Content providers** – share set of application data (ex. Contacts). No use in Euro recognition program.

- **Broadcast receivers** **-** responsible for global events. No use in Euro recognition program.

Activities, services and broadcast receivers can be activated by Intent object, that represents an asynchronous message. By Intent different Activities can communicate between them. The relation between activities in Euro recognition program is presented in Illustration 2.1:

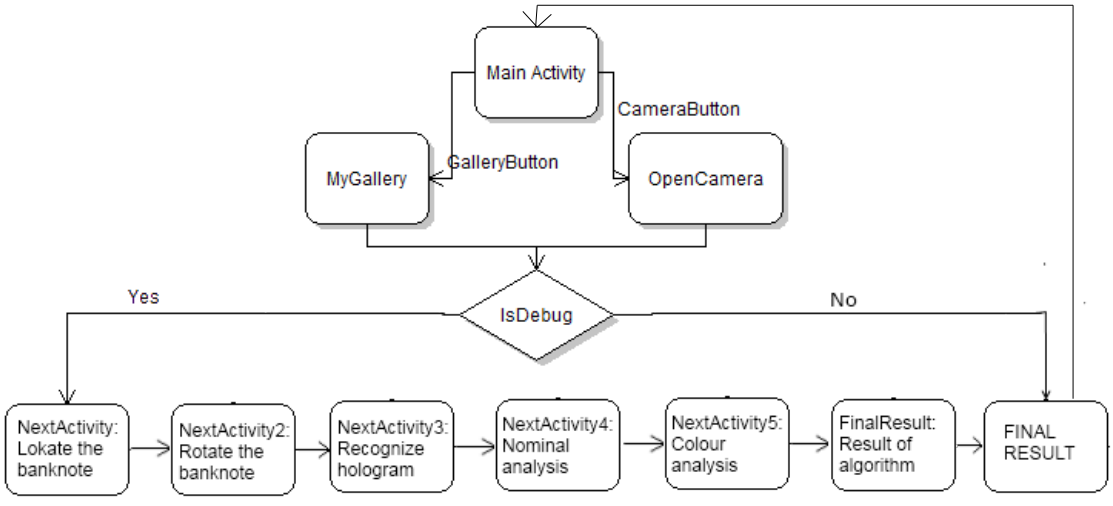


Illustration 2.1: Diagram of activities

As can be seen, the program structure is quite simple and intuitive. There are just four buttons with pictures: Gallery and Camera that describe possible sources of image, Exit that quit the program and About that contains the main information about the author and the application and check box "Debug Mode" that when is marked turn on debugging mode and make possible the execution of the algorithm step-by-step. As a result program returns a processed photo with Text RESULT: _, where can be seen the denomination of Euro banknote or error message, if the program could not recognise the denomination. Every activity has its class, that extends Activity class and control all events. The main screen of the program can be seen on Illustration 2.2:

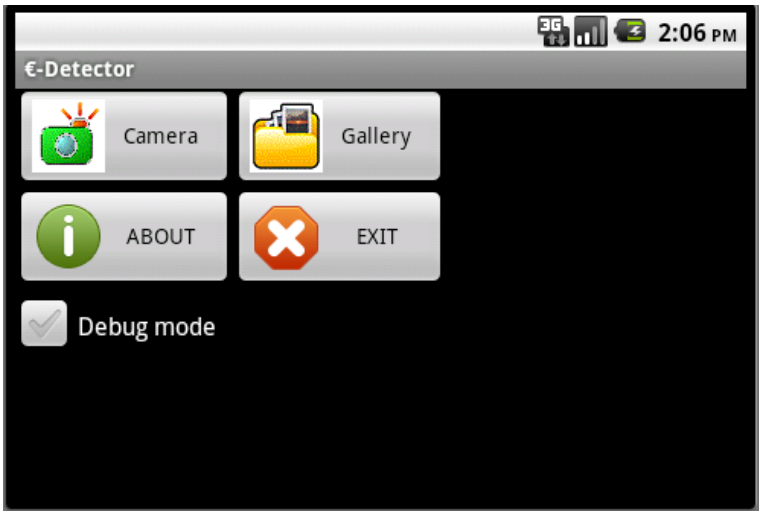


Illustration 2.2: Main Activity

Android enables to use almost all Java libraries. To implement image processing algorithms the presented program uses JJIL (Jon's Java Image Library), because it has a structure easy to extend, there exist some application written with this library that are working with a mobile phone and it is a native library written in Java.

There is also possibility to choose OpenCV, written in C (so faster), but it would be necessary to implement bridges between languages that is more difficult and less intuitive. Google supports native language usage in NDK, Native Development Kit. The Android NDK is a companion tool to the Android SDK that can build performance-critical portions of applications in native code. It provides headers and libraries that allow to build activities, handle user input, use hardware sensors, access application resources, and more, when programming in C or C++. Usage of Android NDK could be the next step of optimisation, but in this project Android SDK is enough efficient and it is not necessary to use a native code.

JJIL Library

JJIL library is particularly targeted towards mobile applications. It includes interfaces for image convertation to and from native formats for J2ME (Java for mobile phones), Android, and J2SE (Java Standard Edition for other operating systems like Windows, Linux and MacOS)[1].

In JJIL library are two essential objects: *Image* and *PipeLineStage*. *Image* is an interface that has methods: getWidth(), getHeight() etc. The examples of classes that implement this interface are Gray8Image, Gray16Image or RgbImage.

PipeLineStage is a class that represents a single image processing algorithm. It takes a single image as a parameter and produces another image as an output. *PipeLineStage* objects can grouped in *Sequence:*

```
Sequence seq = new Sequence();
seq.add(new Gray8HistEq());
seq.add(new Gray8CannyVert(50));
seq.add(new Gray8Threshold(threshold / 10, false));
seq.push(imgage);
this.image2 = (Gray8Image) seq.getFront();
```

By Sequence can be created a FIFO (First In First Out) of basic operations (*PipeLineStage* class instances). First step in Sequence usage is definition of algorithms. On Illustration 2.3 are defines three basic operations that are Gray8HistEq, that equalise the image, Gray8CannyVert that executes Canny edge detection algorithm in vertical direction and Gray8Threshold that binarize the image (changes image values to two for example 0 and 1). All of those basic operations extend *PipeLineStage* class. To defined Sequence the input image can be applied by call push method. The final result of sequence of defined algorithms inside of Sequence object can be return by getFront method. Resume of Sequence mechanism is presented on the Illustration below:

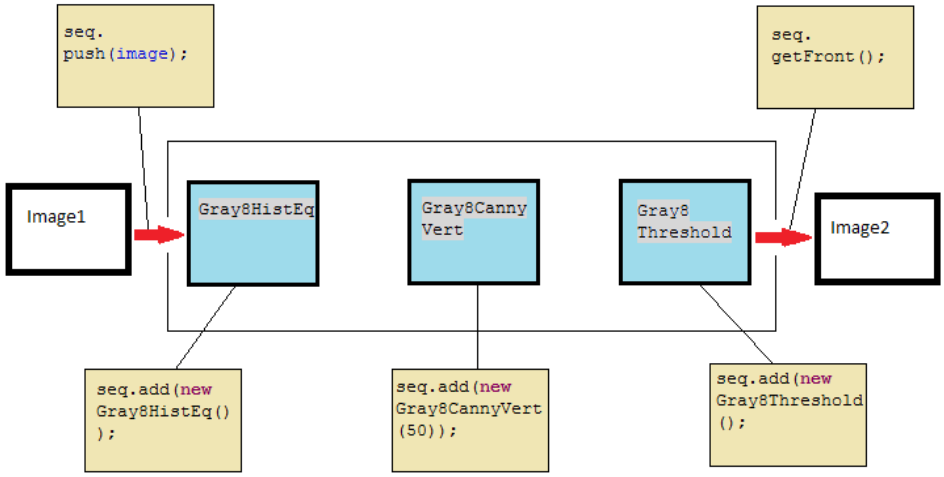


Illustration 2.3: JILL Sequence

In Euro recognition program are used the following classes that are build-in in JJIL library:

- **Gray8CannyHoriz** – calculates Canny operator to extract the horizontal borders of the image

- **Gray8CannyVert** – calculates Canny operator to extract the vertical borders of the image.

- **Gray16Gray8** - converts an 16-bit Gray image to an 8-bit Gray image.

- **Gray3Bands2Rgb** - converts three 8-bit Gray images to RGB by copying the three input Gray values into R, G, and B.

- **Gray8Hist** - computes the histogram of a Gray image.

- **Gray8HistEq** - equalize the histogram of a Gray image.

- **Gray8Rgb** - converts an 8-bit Gray image to RGB by replicating the Gray values into R, G, and B.

- **Gray8Threshold** - threshold Gray image. Output is Byte.MAX_VALUE over threshold, Byte.MIN_VALUE under.

- **RgbAvgGray** - averages the R, G, and B values to create a Gray image of the same size.

- **RgbHsv** - converts an RGB image to an HSV image.

- **RgbSelectGray** – transforms a RgbImage into a Gray8Image by selecting one of the three bands.

- There are also used structures classes for creating necessary objects (jjil.core package)

All other operations (except Hough algorithm that bases on other example and is a modification of existing code) are author's implementations. All of them extend *PipeLineStage* class and can be used in Sequence, exactly in the same way that build-in standard JJIL algorithms. Custom classes possible to use in Sequence push method are as follows:

- **Gray8Dilate** – dilate a Gray thresholded image
- **Gray8Erose** – erode a Gray thresholded image
- **Gray8Subimage** – cut defined rectangle region of a Gray image
- **GrayFastRotate** – rotate a Gray thresholded image
- **GrayRotate** – rotate a Gray image
- **Rgb2YCbCr** – pass RGB image to YcbCr colorspace
- **RgbHistEqualize** – equalize histograms of RGB image
- **RgbImageSubImage** – cut defined rectangle region of RGB image
- **RgbRotate** – rotate RGB image

Custom classes, that do not extends PipeLineStage class but are used in the program are:

- **FindRegion** – class responsible for looking for interesting region based on histograms (horizontal and vertical)
- **Gray8HistHorizontal** – horizontal histogram for Gray image.

- **Gray8HistVertical** – vertical histogram for Gray image.

- **HistRegion** – represents part of a histogram, that is zero or more that zero. Used in looking for number pattern.

- **HistRegions** – all objects HistResion of one histogram together.

- **HoughLine** – represents line in normalized form: x*cos(theta) + y*sin (theta) = r.

- **HoughTransform** – based on [35], calculates how many vertical lines cross every points of an image.

- **HsvHistogram** – colour histogram for an image in HSV colorspace.

- **OtsuThresholder** – calculate best threshold for an image.

- **Transformation** – the main class that is putting all operations together. It is responsible for localizing, checking position, rotating and recognizing of a banknote. Class implements Singleton programming pattern. It means that there is just one instance of the class in the program.

The list does not include classes of GUI (Graphical User Interface). There are presented just the most important classes necessary to understand the global algorithm.

All source code of Euro recognition application count 5273 lines of code, where 3680 lines are of methods, implements 36 custom new classes, where are 138 different methods. In Appendix 2 can be found one part of the source code of Euro recognition application for Android. Another part of application for Java SE is not included in this paper and will be provided as a additional software.

A detailed description of all classes used in this project and detailed description of its fields and methods can be found in Appendix 1: Java documentation, where is generated Javadoc (standard Java documentation).

For all tests and statistics generation is created another version of the Euro recognition program that is working with Java Standard Edition (Java SE). The code source of JavaSE version is near to the same that the source code of Android version. The unique difference is graphical user interface because of different characteristics of programming environments and graphical libraries. Java SE version is easier to present and faster than Android emulator provided with Android SDK. It can work with all popular operating systems like Windows, Linux and MacOS. The program is doing exactly the same that Android version but is a lot of easier to use in time of statistics generation and implementation, debugging and testing new algorithms. It is also easier to present for audience. Graphical user interface of JavaSE version is presented on the illustration below:

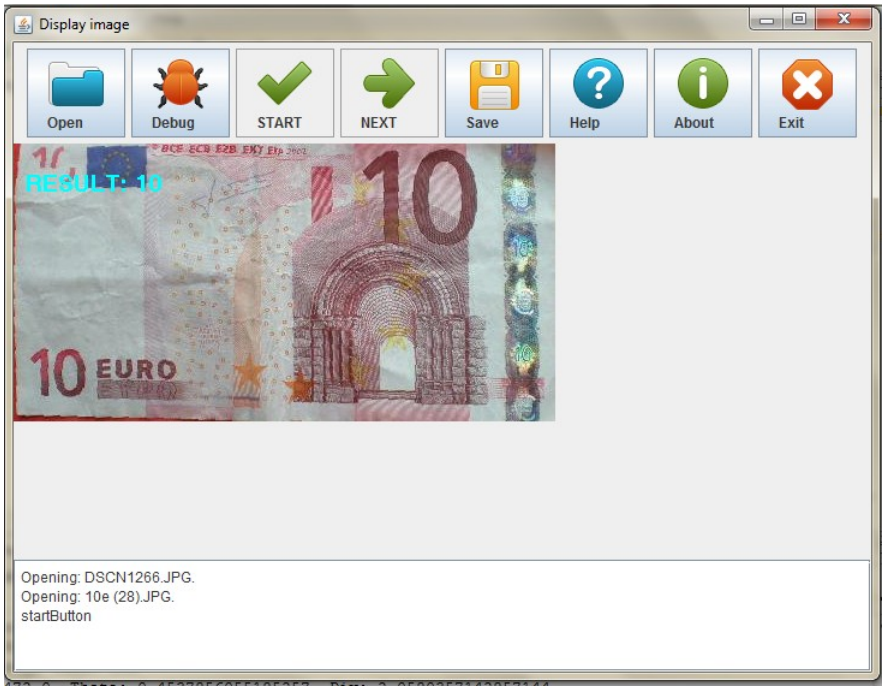


Illustration 2.4: Java Standard Edition version of Euro recognition program.

Java SE version has different graphical user interface (GUI) with more buttons. The GUI is created with Swing library, that is standard Java library for creating user interfaces and graphical layouts of applications. All buttons have small graphic what make using the program very easy and intuitive. Below are presented descriptions of function for every button:

- **Open** – open the window, where the user can choose an input file. Input file has to be image file. Accepted formats: jpg, png, bmp, gif
- **Debug** – it is a special button with two states. When it is pulled the program execution will be in direct mode. It means that program will return directly processed and recognised banknote. When the button Debug is pushed, program execution will be in debug mode and will executes step-by-step. Debug has to be pulled or pushed before use the Start button.
- **Start** – start execution of the program. Depending on Debug button state, the program execution can be direct or step-by-step.
- **Next** – if the program execution is step-by-step, button Next go to the next step of the algorithm.
- **Save** – saves current processed image.
- **Help** – shows the pdf document with all documentation.
- **About** – shows the main information about the program and author.
- **Exit** – exit the program.

Below, on the bottom part of the interface is a text area of Log. Every operation executed by the program return same text which is shown in this text area. In this way it is possible to debug the program and show a result of every step in easy way.

In the middle area is displayed an image. When the algorithm change the image, the result is visible in this area. Final result of recognition (denomination of banknote) also is described in this area.

Euro banknote characteristics used in the program.

Euro is a common European currency introduced in place of national currencies in 1995 in Madrid. It is a legal currency in 17 countries forming the Euro area in the European Union.

Euro has 7 denominations: 5€, 10€, 20€, 50€, 100€, 200€ and 500€. Because cash registers manage just with 5, 10, 20 and 50 €, the program presented in this paper recognises just those four denominations.

Every Euro banknote has different size, proportional to the denomination. If smaller denomination, smaller size. For the EUR 5 to EUR 100 banknotes, the size increases by five millimetres width wise and by six or seven millimetres lengthwise[8]. The quotient of width and height of the banknote is independent on denomination and is constant always equals two.

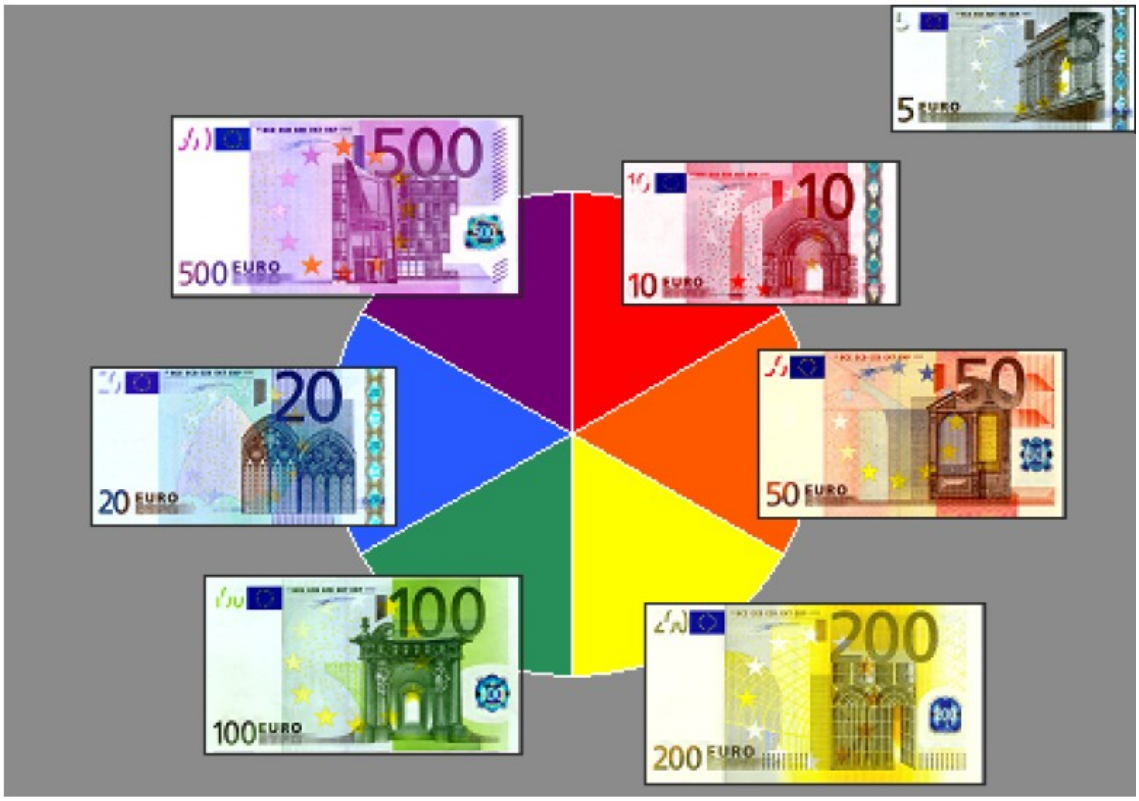


Illustration 2.5: Euro banknotes colors

Banknotes also have different colours. Primary colours (red, green and blue) were chosen for widely used banknotes (EUR 10, EUR 20 and EUR 100) because they are the most easily distinguishable. Secondary colours (orange, yellow and purple) were chosen for the EUR 50, EUR 200 and EUR 500 banknotes. The most widely used EUR 5 banknote is grey because it does not show the dirt as much[8]. In this paper is used colour recognition algorithm to recognise 20 and 50 Euros.

Every banknote in the top left corner has a blue European flag with 12 yellow stars. Number twelve in this case is a symbol of harmony and perfection. On the right side bridges and gateways of different European countries with the value of denomination can be seen. At the bottom left

side there is the number of denomination of the banknote. A part of denomination can be also seen on the top left side.

One of the security features is a hologram. On the front, the hologram is in the form of a stripe for the 5, 10 and 20 Euro banknotes and a patch for the 50, 100, 200 and 500 Euro banknotes. For the hologram stripe, depending on the banknote – the hologram image will change between the value and the (€) symbol and the stars on a rainbow-coloured background. At the edges, tiny letters show the value. For the hologram patch, the hologram image will change between the value and a window or doorway. In the background, you can see rainbow coloured concentric circles of tiny letters moving from the centre to the edges of the patch [8].

Almost all other security features are visible with white, infra-red and ultraviolet light and it is impossible to check it just with simple camera of mobile phone. For this reason this paper will not focus on those features.

Euro recognition program use especially the quotient of width and height to rotate the banknote to appropriate position. Recognition of denomination bases on patch hologram for denomination of 50€, colour recognition for denominations 20€ and 50€ and number analysis in bottom left corner for 5€ and 10€. Another characteristic that can be used to localise and recognise the banknote is European flag that could be found by the colour. However nowadays it is in experimental state.

3. BANKNOTE LOCALISATION

Banknote localization is the most important part of all banknote recognition process. All other algorithms depend on the result of this part. Near to all false or not recognized banknotes have bad detected localization.

Localisation problem can be solved very easy by forcing static, previously known position. In this case the user would have to put the banknote exactly in the marked area, where banknote image would be processed and recognised. In this way problem of Euro banknote recognition would be very easy and limited to compare simple patterns. However, this diploma project has research character and presented algorithm of localisation does not simplify the problem. Presented algorithm uses well-known algorithms like Canny edge detector, Otsu method and author's banknote detection algorithm based on histograms and can be used to detect all kind of objects.

Before starting to work with an image it is necessary to make preprocessing operations for optimization. First thing what is needed is an image resizing. Resizing improves speed of all calculations significantly. Nexus One and other modern mobile phones with Android operating system takes photos with high quality, which takes a lot of memory and them processing would be very expensive. The photo is resized to a photo 6 times smaller than the original one. It means that one pixel of the image ready to process is 6 pixels of the original image. In this way it is not necessary even to read all image and it saves a lot of memory and calculation time in next phases.

The next step is to change the RGB colour space to Gray colour space. In RGB every pixel has 3 values: R-red, G-green and B-blue (more about RGB colour space can be found in Chapter 6 of this paper). In Gray colorspace is just one value that makes easier all calculations. Gray colour value can be calculated as an average of those three elements (R, G and B). After those

two general operations the process is ready to start looking for the banknote.

The first step of the localisation algorithm is to find contours of the image and then try to decrease noise without losing important details. To find contours Canny algorithm, that is one of the most efficient algorithms for contour detection, is used.

Canny algorithm

For monochrome images, the edge is defined as physical, photometric and geometric features of image discontinuities.

The Canny edge detection operator was developed by John F. Canny in 1986 and uses a multi-stage algorithm to detect a wide range of edges in images[5]. Canny algorithm for edge detection is nowadays the most optimal method. What's more, Canny described what requirements should meet optimal edge detector:

1. minimal response - a concrete edge should be marked only once, and if possible, interference (noise) in the image should not create false edges.

2. good location – the marked edge should be the same or close to the actual edge of the image.

3. good detection - algorithm should detect as many edges as possible.

It uses not only the information about the intensity of changes (gradient) but also tries to decrease the impact of noise as much as possible, and to improve the quality of the result edge.

Canny edge detection algorithm has the following steps:

1. Smoothing an image using a Gaussian filter. It reduces noise of the image.

2. Finding the intensity gradient of the image. The edge of the image can have different directions. The algorithm uses four filters to detect horizontal, vertical and diagonal edges in the smoothed image. Edge detection angle is rounded to four cases representing the department, level and two diagonals (eg 0, 45, 90 and 135 degrees)

3. Removing non-maximum pixels. The third stage involves "thinning" of edges in a way that ensures their continuity. The result is a continuous line of individual pixels.

4. Thresholding. It removes unnecessary edges.

In Euro recognition program Canny algorithm is from JJIL library and it is a little bit modified. The sigma value for Gaussian filter is quite high: 3. It is because the image of Euro banknote is a low-frequency image and in this stage the program wants just the most important and the most significant edges. The algorithm is called for horizontal and vertical edges separately. This allows calculating horizontal and vertical histogram for object detection.

Canny algorithm is executed two times: in horizontal direction and in vertical direction. Both cases after Canny algorithm the image is thresholded (with the most optimal threshold found by Otsu algorithm) and noise is eliminated by opening the image. The result of all of those operations is a contour of the original image that after thresholding has just two values of pixels: -128 (Byte.MIN_VALUE) and 128 (Byte.MAX_VALUE).

In order to find the object is necessary to calculate histograms of sum of pixels in horizontal (Illustration 3.1) and vertical (Illustration 3.2) direction. In this paper horizontal histogram is the array of sums for columns (vertical rows), horizontal is the axe of arguments (arguments from 0 until width of the image). There is a similar definition for vertical histogram: it is the array of sums of horizontal rows for vertical axe of arguments (arguments from 0 until height of the image).

On every histogram can be found the maximum value (the position, where are more white pixels, maximal sum). Then, there is marked the region where can be found pixels that are greater than 10% of the maximum. First looking for object is done with small margin added to calculated area to minimize noises. Then, the image is cut just to the interested region. After it the algorithm starts the second time in previously found region and is called without margin, so returned just the most interested and significant object. In this way the second call can find the exact position of the bill. The result of the second call can be seen on Illustration 3.3.

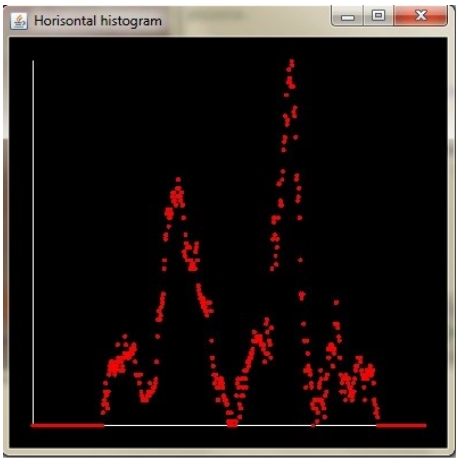


Illustration 3.1: Horizontal histogram

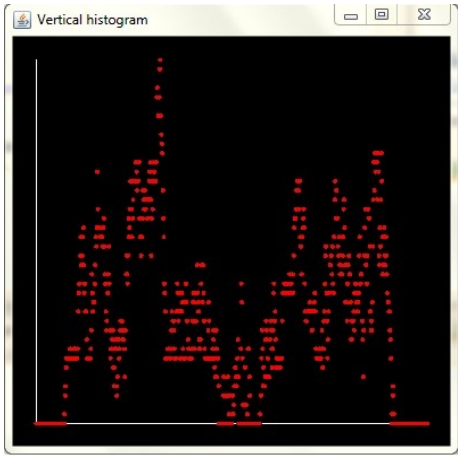


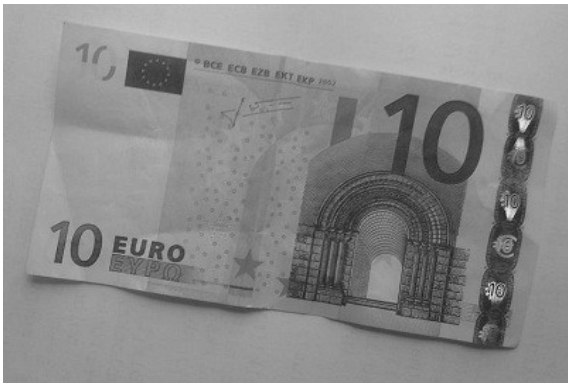
Illustration 3.2: Vertical histogram.

Illustration 3.3: Final result of banknote localization

Another option for banknote localization is to find the European flag, that has a deep blue colour and usually which is usually easy to extract. In Java SE version of the program there is made a small experiment and implemented a function markFlag in Transformation class. The problem is that when the image is small and is in a rotated position, colour detection is not precise enough to detect the image position. The method is working well for parts of images, but its efficiency was so much lower and its value is more experimental. On the other hand, the idea can be perfect for detection of more than one image, where the histograms method is not useful.

Resume of an algorithm for banknote localisation:

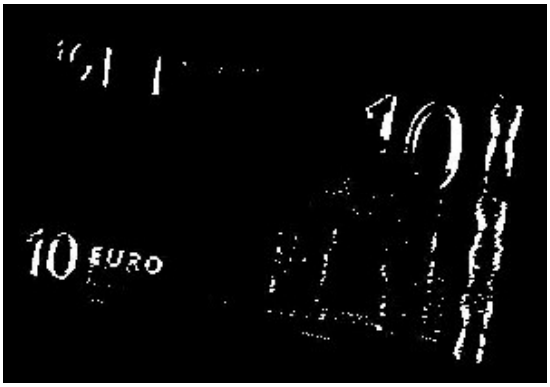
1. Preprocessing: resize the image and convert to Gray colorspace.



2. Execute Canny horizontal algorithm for contour detection

3. Calculate the most optimal threshold using Otsu algorithm and binarize the image.

4. After thresholding open the image (Erode and Dilate operation) to reduce a noise.



5. Do steps 2-5 with vertical Canny algorithm

6. Calculate two histograms: one for vertical contour and one for horizontal contour.

7. Based on calculated histograms find interesting region (region that in horizontal and vertical histogram is greater than some precent of maximum of histogram (the value depends of a step of algorithm and an image). In this way small noises can be ignored. Return region with small 10% margin.

8. Reduce the image to the found region with margin.

9. Repeat steps 2-7.

10. Calculate exact position of a bill based on image reduced to the area without margin.

Limitations:

The algorithm contains some limitations. One of them is that the photo of a banknote has to be without any other big object. It can contain some noise, it can be of a worse quality, with some light reflexes, but there should not be any other object. Strong noise or strong light reflexes can affect the efficiency of the algorithm. The algorithm can find just one bill per photo and can be used to localise any object, not only Euro banknote.

4. SETTING IN THE CORRECT POSITION

The application allows to take photos of banknotes in different positions (rotated). For this task are necessary algorithms that can detect and change the banknote position to horizontal.

To detect the position first a quotient of width and heigh of the image is calculated, than in Euro banknote always is two. If it is different that two algorithm of Hough is used. The Hough algorithm is modified in the way that detects only vertical lines. In order to minimize the number of lines the program is only looking for lines with more neighbours (for image 600 x 200 is 40 neighbours). Then, for every line the image is rotated, and program calculates horizontal and vertical histogram and checks if the quotient of width and height is near to 2. All of those operations are made using the thresholded horizontal image contour that was calculated before. Detailed Hough algorithm is described in the next part of this chapter.

Hough Transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform[7].

The definition looks very sophisticated, but the algorithm is quite simple. Hough Transform allows to look for various different shapes like

lines, circles, ellipses etc on an image. The lines are the most important for this project, therefore, this chapter will only focus on this issue.

In 2D space the line can be represented in a normal equation, which is an equation of the line situated at an angle Θ to the axis OY and remote r from point (0,0):

normalized line equation: $x \cdot \cos(\Theta) + y \cdot \sin(\Theta) - r = 0$

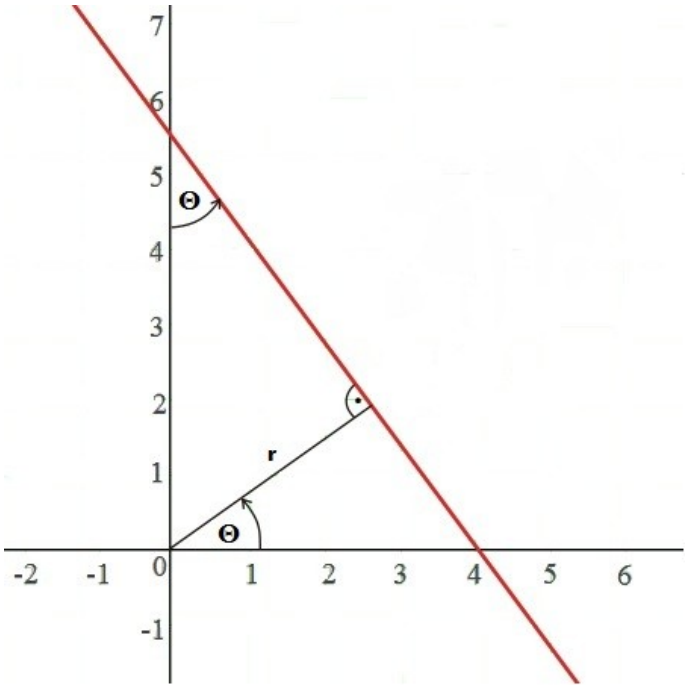


Illustration 4.1: Normalized line graphical representation

Assumption: Pixels with non-zero values are elements of the edge. If pixel (x, y) is in line then find a set of values (r,Θ) in parameters space of the line. (x,y) is known and (r,Θ) is for looking for.

It is necessary to define Hough array, where the results can be saved. Hough array has a form of a table of accumulators as is shown on the Illustration 4.2 below:

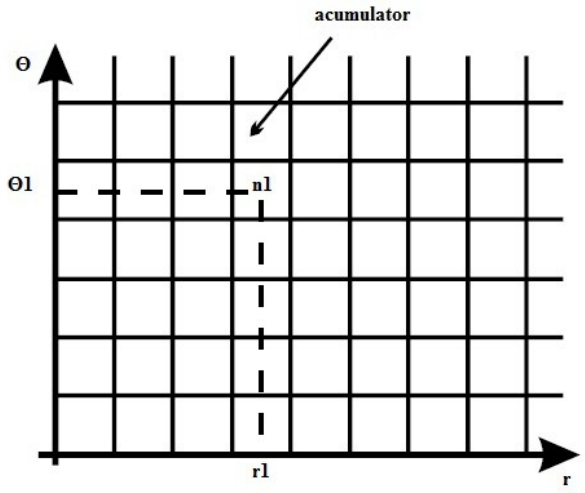


Illustration 4.2: Hough array, where n1 is the value of accumulator

Step 1: For every pixel of the image the program is looking for all lines that cross the pixel and then increased by 1 the number of accumulator for calculated values of (r,Θ). In this way the element value of the Hough array (r,Θ) is the number of lines that cross non-zero values pixels of the image and has those parameters in them normalized equation.

Step 2: Find elements with the highest values in Hough array. If an element (r_1,Θ_1) of the array has value *k,* it means that exactly *k* points of image form a part of the line with parameters (r_1,Θ_1).

Euro recognition program is looking for lines with high values of Hough array elements and high number of neighbours. In this way it is possible to decrease the number of lines from about 100 to 3-10. It is not necessary to look for all (horizontal and vertical) lines, it is enough to know just the vertical lines.

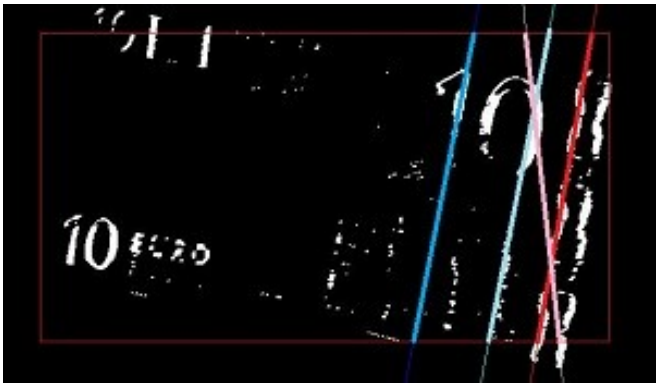


Illustration 4.3: Result of Hough algorithm for Euro recognition

Rotation

A rotation is a circular movement of an object around a centre (or point) of rotation[6]. In case of banknote recognition program rotation is always around the centre of the image.

Rotation is a basic operation for image processing, and the complexity of its computation is considered as the key problem of the implementation.

The rotation operator for arc Θ around point (x_0, y_0) performs a transformation of the form:

direct:
$$x' = x \cdot \cos(\Theta) - y \cdot \sin(\Theta) + x_0 \cdot (1 - \cos(\Theta)) + y_0 \cdot \sin(\Theta)$$
$$y' = x \cdot \sin(\Theta) + y \cdot \cos(\Theta) + y_0 \cdot (1 - \cos(\Theta)) - x_0 \cdot \sin(\Theta)$$

inverse:
$$x = x' \cdot \cos(\Theta) + y' \cdot \sin(\Theta) + x_0 \cdot (\cos(\Theta) - 1) + y_0 \cdot \sin(\Theta)$$
$$y = x' \cdot (-\sin(\Theta)) + y' \cdot \cos(\Theta) + y_0 \cdot (\cos(\Theta) - 1) - x_0 \cdot \sin(\Theta)$$

where (x_0, y_0) are the coordinates of the centre of rotation (in the input image) and Θ is the angle of rotation with clockwise rotations having positive angles. Pixel locations out of which an image has been rotated are filled in with black pixels. Usually rotation algorithms use inverse rotation, where for every pixel of the result image the algorithm is going to look for approximated pixel from input image. In this way it is sure that all interesting pixels of result image are filled by appropriate pixels from the input image.

For optimizations purposes the standard way is changed and the algorithm of rotation is a little bit modified. In case of banknotes after preprocessing, when there is ready binarized image, it is possible to simplify the rotation algorithm. There is not necessary to know a value of all pixels of the result. It is just necessary to know the new position of white points from the thresholded image. This is why a simplified algorithm applies direct rotation just for white points. Thanks to this operation the execution time is about ten times faster! Inverse rotation also is used, but only for final RGB image, when is known the exact value of arc of rotation.

Arcs for rotation are searched with Hough algorithm, that returns couple of lines with different value of Θ. For every Θ the program rotates thresholded image, calculates histograms (horizontal and vertical) for locating the new image and checks the quotient of width and height of the result. If the quotient is very close to 2, algorithm of rotation is finished and program rotates also RGB colour image (using inverse full rotation algorithm for every element R, G and B). If the algorithm is not able to find the position of the banknote, where quotient of width and height is two, then returns the value of arc, that is the closest to two.

Illustration 4.4: Rotated image

After rotation it is important to check the position of the banknote and if it is necessary rotate it 180°. This task is quite easy to solve because Euro banknotes have the left side lighter than the right side. There is just necessary to calculate the number of white pixels of part of the left side and compare with the number of white pixels of the right side.


Illustration 4.5: The region of interest for checking if the banknote is in correct position and its quality.

On the Illustration 4.5 the region where white pixels of the both sides are calculated is presented. There is also visible small black points in white regions. That is the result of the direct rotation of thresholded image.

At this point can be also checked the quality of the image, and if it is of low quality, banknote localisation algorithm is called again for equalized image. It is important to do this in this point, not before Hough Transform, because the program could lose efficiency. (Hough transform and rotation is slower and more sophisticated that banknote localisation algorithm). The image after equalisation of histogram is presented on the Illustration below:

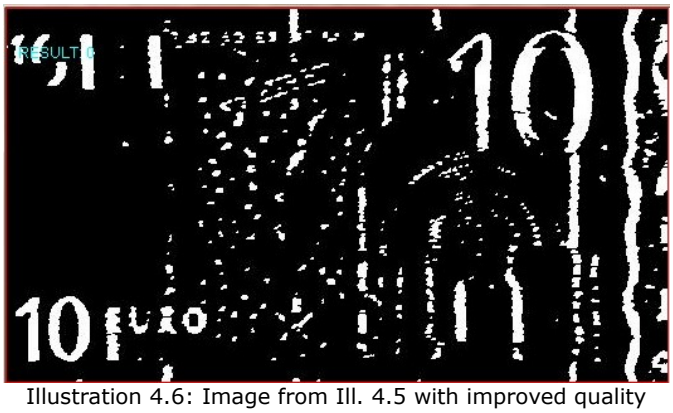


Illustration 4.6: Image from Ill. 4.5 with improved quality

At this point the image is ready for banknote recognition algorithms, that are presented in next chapters of this paper.

Resume of image rotation algorithm:

1. Check the quotient of width and height of the object found. If quotient is close to 2 then return 0 else go to step 2.

2. Execute Hough algorithm and find all significant vertical lines. Initiate the quotient value *m*. Also remember the value of Θ for this quotient.

3. Take a line from Hough lines and rotate the thresholded Gray image Θ degrees using direct, fast rotation algorithm. If lines vector is finish, return Θ value (it will be Θ value, for witch the quotient is the closest to 2).

4. Check the quotient of width and height of the object found on the rotated image. If |2 – *m*| (previous quotient) < |2-actual quotient| change *m*=actual quotient. If *m* is close to 2 return Θ else go to step 3.

Resume of checking position and increase quality algorithms

1. Take the horizontal region between height/3 and 2·height/3 from the rotated image.

2. Share the region in the middle of width.

3. Calculate the sum of white and black pixels in left and right side of the shared region.

4. If the sum of white pixels of left side is bigger than sum of the white pixels of the right side rotate the image 180^{O}. If the quotation of number of black pixels of whole region and number of white pixels of whole region is bigger than 25, execute localisation algorithm for rotated image with low quality set to true.

5. RECOGNITION OF THE NUMBER AND HOLOGRAM

In the image analysis process, one of the most important steps is the segmentation, where interesting objects in the image can be identified. Without a correct segmentation these objects can not be recognized. In this project, segmentation is quite easy, because after correct banknote localisation and rotation the position of the banknote is static and well known. In this way it is quite easy to separate the interesting region and process the image just in some small area. This makes calculations much faster and it is a perfect way of optimization.

To recognise the denomination of the banknote the program generates histogram patterns (vertical and horizontal) and looks for correct pattern option for each case. By using the histogram patterns of the denomination value it is easy to recognise that 5€ and 10€, 20€ and 50€ are very similar and for their distinction are used other algorithms like hologram analysis (for distinguish 50 Euros) and colour analysis. Applying this algorithm it is also easy to identify which banknote is corrupted. If the banknote is corrupted or has bad position, bad quality or is another object, then the calculations are finished and program returns zero result.

Another option for recognising the denomination is to analyse the hologram. For 5, 10 and 20 Euros the hologram on the right side is in the form of a line, for 50 Euros it is different and has the form of a small quadrant. This method is good to recognise the denomination of 50 Euros. The biggest fault of hologram recognition is that is very sensitive to the light and usually is distorted by the light reflexes. However, the method is quite easy and simple and it is used in the program as a part of recognition algorithms.

Hologram analysis

Every Euro banknote has a hologram. On the front, the hologram is in the form of a stripe for the 5, 10 and 20 Euro banknotes and a patch for the 50, 100, 200 and 500 Euro banknotes[8]. Because the program does not recognise 100, 200 and 500 Euros, the patch hologram would say with high probability that the processed banknote is of 50 €.

To perform this task, the program looks for an interesting area, and then with the help of histograms, examines if the histogram fits the pattern. The Illustration below presents the area where the program is looking for the hologram.

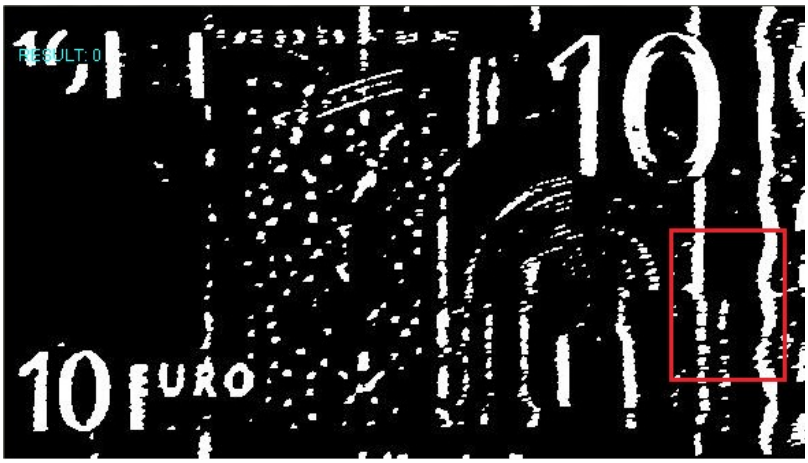


Illustration 5.1: The red area is the area where the hologram analysis is executed

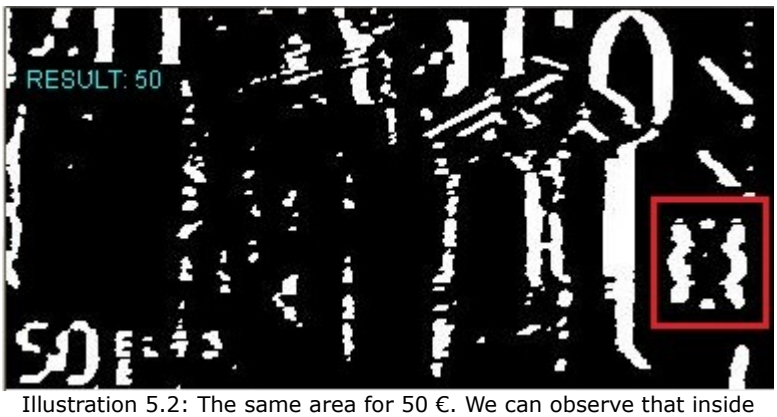


Illustration 5.2: The same area for 50 €. We can observe that inside the area is patch of hologram.

For this small area the program calculates horizontal histogram (sums of columns from 0 to width). A correct histogram of the hologram for 50 € is shown on the Illustration below. Its form is of a extended letter M as is shown on the Illustration 5.3

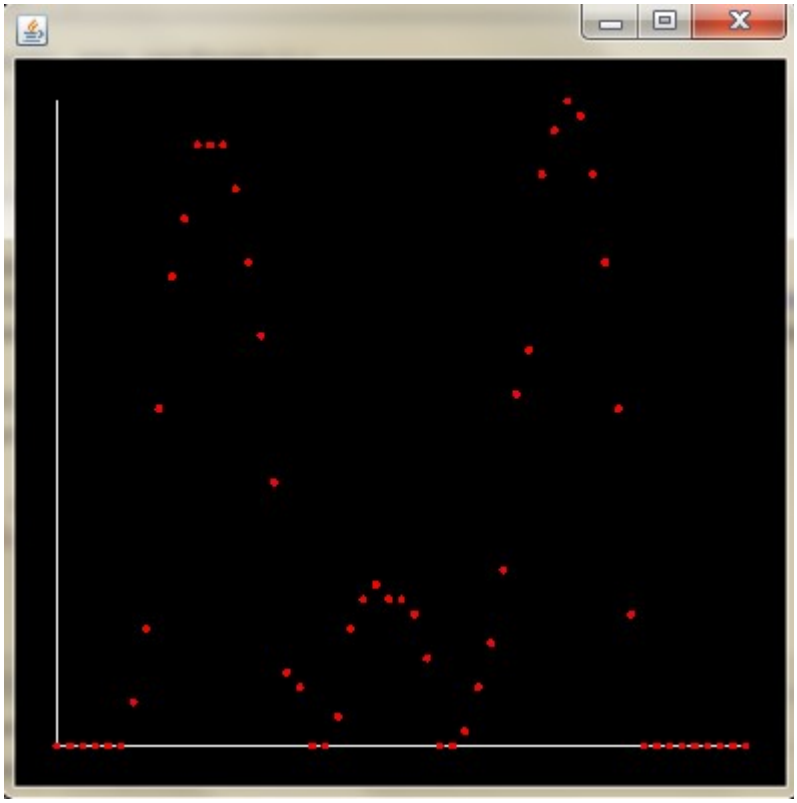


Illustration 5.3: Hologram histogram for 50€ patch.

Resume of hologram analysis

1. Find the region of the hologram

2. Calculate the horizontal histogram for the region of the hologram

3. If the maximum value of first and last non-zero area is greater than the middle non-zero area and the number of elements of zero

regions between non-zero areas are less than 15% of the total of all elements of the histogram then return 50 (it is 50 €) else return 0.

Patterns of Denominations

To recognise the denominations the program uses certain patterns. A pattern is a type of theme of recurring objects, in case of Euro recognition program a pattern is a part of image, where generated horizontal and/or vertical histograms have some determined properties. For example, the pattern for 5 Euros has the first non-zero area (number 5) bigger than the second non-zero area (small Euro letter) in the horizontal histogram. The pattern for 10 Euros has the first non-zero (that is number 1) area smaller than the second non-zero area (number 0). The pattern for 20 and 50 Euros are very similar and the first and the second non-zero areas have similar width. The area of the number is shown on the Illustration 5.4:

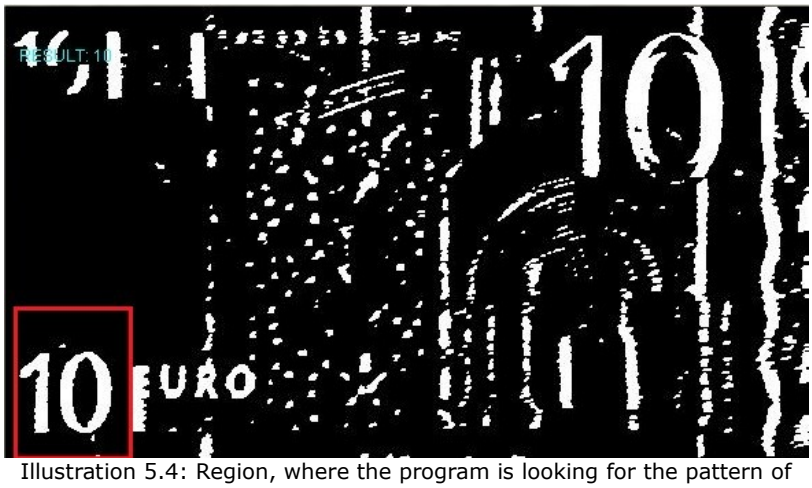


Illustration 5.4: Region, where the program is looking for the pattern of the denomination

For this small area from the illustration above the program generates histograms: horizontal and vertical as it is shown on the Illustration below:

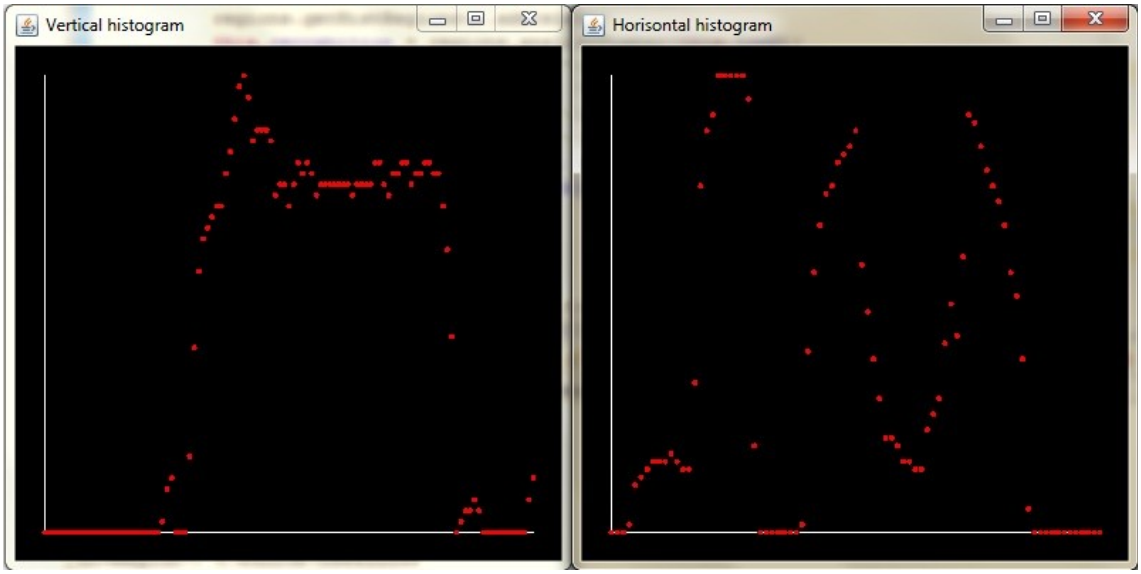


Illustration 5.5: Histogram patterns

The vertical pattern is used to detect and if it is necessary to eliminate the horizontal border line. In this way the horizontal histogram always has zero and non-zero areas. The pattern of Illustration 5.5 is of 10 Euros because the first non-zero area of the histogram is smaller than the second non-zero area. Patterns for other denominations are presented on the illustration 5.6:

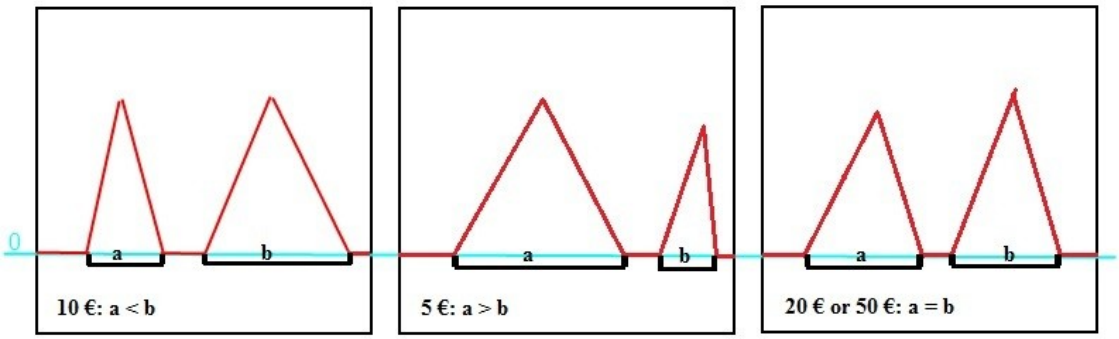


Illustration 5.6: Different patterns for banknote recognition

Schematic graphs from the Illustration 5.6 show general idea of number patterns based on histograms. For 10€ first triangle base, that represents number 1, is smaller than second triangle base, that represents number 0. For 5€ the first triangle represents number 5 and the second the small "E" letter from the Euro word next to the denomination of the banknote. For 20€ and 50€ hologram patterns are very similar and both triangle bases are near to be the same.

Resume of algorithm of number analysis

1. Find the area of denomination in right bottom side of the bill.

2. Calculate the horizontal histogram for the area

3. Analyse the histogram: if the histogram has two non-zero areas, shared by one zero area, continue else return 0 (program can not find the denomination or quality of denomination is bad)

4. Calculate the quotient of the first non-zero area per second non-zero area. If the result is between 0.3 and 0.7 return 10 (it is 10€). If it is between 1.5 and 3 return 5. If it is between 0.7 and 1 return 2050 (it can be 20€ or 50€).

6. RECOGNITION BY THE COLOUR

Colour analysis is a very difficult task, because colour can change on every picture depending on the light. Colour is the product of a visual environment, illumination and the human brain. The perceived colours of objects are determined by the spectrum reflectance of surfaces, illumination, and the visual angle; illumination and the visual angle are always in flux.[9]. In Euro recognition program colour recognition algorithm is used just for denominations of 20€ and 50€. It is because the algorithm of denomination recognition gives good results for 5€ and 10€ and it is not necessary to use other additional algorithms. For 20€ and 50€ denomination analysis does not give correct result and it is necessary implements additional method just for those two denominations. The colour recognition is simplified to the problem of deciding if the analysed colour is more blue or more red.

Colorspaces

From Wikipedia: "A colour model is an abstract mathematical model describing the way colours can be represented as tuples of numbers, typically as three or four values or colour components"[10]

The most popular colorspace is RGB colorspace. Its name comes from the first letters of English words red (R), green (G) and blue (B). RGB model was originally applied to analogue technology, now also digital. It is widely used in image devices (eg digital cameras, scanners) and image display devices (eg televisions, computer monitors). In this space every colour is the combination of three basic colours (see the Illus below). The model RGB we can be represented as a cube. In the Euro recognition program it is not used for colour analysis, because its values in wide area depend on the

light. Other colorspaces, like HSV and YcbCr colorspaces, can be used more effectively for colour analysis.

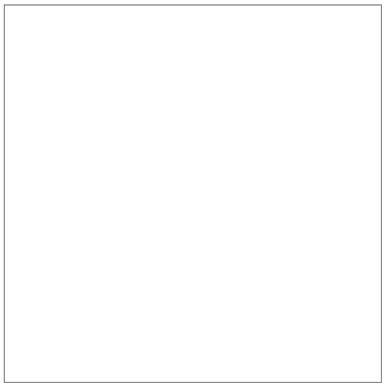


Illustration 6.1: RGB combination of colors.

Other colorspace that is more useful for colour recognition is HSV colorspace (Hue – Saturation - Value). HSV model refers to the way in which the human eye sees, where all colours are perceived as light. According to this model, all colours come from the white light, where part of the spectrum is absorbed and some reflected from the illuminated objects. The model is considered as a cone, which is based on the colour wheel. Hue is a light color, that can have values from 0 to 360° and represents an arc on the color wheel. Other symbols like S-saturation describes the radius of the base and V – value, represents the brightness and is described as the height of the cone. HSV model represents the Illustration below:

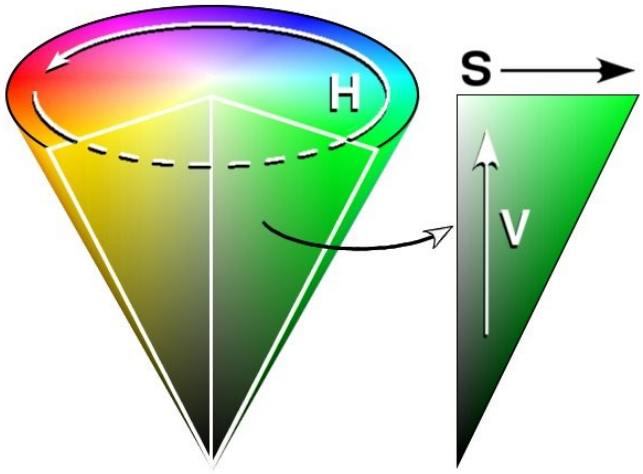


Illustration 6.2: HSV colorspace representation

There is an easy, direct way to convert RGB colorspace to HSV colorspace:

$$H = \cos^{-1}\left(\frac{\frac{1}{2}\cdot[(R-G)+(R-B)]}{[(R-G)^2+(R-B)\cdot(G-B)]^{(\frac{1}{2})}}\right)$$

$$S = 1-\frac{3}{(R+G+B)}\cdot[min(R,G,B)]$$

$$V = \frac{1}{3}\cdot(R+G+B)$$

Other colorspace used in the program is YCbCr colorspace. YCbCr is a model of colorspace, used for digital transmission and storage of images and video. It uses three types of data: Y - luminance component, Cb that is the difference between the luminance and blue and Cr - the difference between the luminance and red. The green colour is obtained on the basis of these three values.

The formula to convert RGB to YCbCr is direct:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

$$Cb = -0.168736 \cdot R - 0.331264 \cdot G + 0.5 \cdot B + 128$$

$$Cr = 0.5 \cdot R - 0.418688 \cdot G - 0.081312 \cdot B + 128$$

Based on [13], where YcbCr was used for face detection algorithm, YCbCr colorspace can be used also to recognise colour of 20 and 50€ with quite good result. In [13] authors made an experiment, and they discovered, that for human skin colour Cb value is similar to Cr value. The same characteristic can be used for recognition of 50€, that has similar colour to human skin. 20€ is recognised, when Cb and Cr values are quite different.

Colour recognition algorithm

In the presented algorithm first step is to extract a small region of interest. Sometimes the picture of a banknote can move a little bit depending on the quality and rotation of the photo, so the program just checks if the picture is more red than blue or more blue than red. The area of colour analysis is shown on the Illustration 6.3:

Illustration 6.3: Area of color analysis

Some face recognition algorithms uses the property of YCbCr colorspace that pixels belonging to skin region exhibit similar Cb and Cr values[13]. 50€

has a colour similar to human skin colour and the program for Euro recognition use this characteristic.

On the other hand very good results gives Hue value analysis from HSV colorspace. In the program pixel with H-value between 150 and 200 is classified as a Blue and pixel with H-value between 0 and 20 is classified as a Red. The banknote is recognise depend on which value is grater: if blue then recognise 20€ and if grater is red then program returns 50€. For 10€ and 5€ colour analysis is not necessary because in theory the program should recognise them in denomination analysis algorithm phase.

Resume of colour analysis

1. Find the region of interest, where colour should be analysed.
2. Convert the selected region to YCbCr colorspace.
3. For all pixels of interested region is Cb ≈ Cr then mark the pixel as a red, else mark a pixel as a blue.
4. Convert selected region to HSV colorspace.
5. For all blue pixels: if H value is between 150 and 200, add one to blue counter.
6. For all red pixels: if H value is between 0 and 20, add one to red counter
7. If red counter is bigger than blue counter return 50, else return 20.

7. RESULTS AND CONCLUSIONS

Final results depend on the quality of photos, background, light and denomination. Examples of incorrect pictures are presented on the Illustrations below:

Illustration 7.1: Incorrect non-plain background

Illustration 7.2: Strong light reflexes

Illustration 7.3: Another object

The colour of the background of the picture has secondary importance.

All presented statistics are taken for photos with more or less plain background, without strong light reflexes and with just one object per picture.

The total number of pictures used in statistics is 200. In The Error column null means no error, “Low quality” comment means that the program had to improve quality of the image and “can't find number” message refers to those algorithms of denomination recognition, which failed and did not find the correct pattern. “Can't find number” error usually is a result of bad image localisation.

Statistics for 5€

File name	Result	Error
5e (1).JPG	RESULT: 5	null

5e (10).jpg	RESULT: 5	Low quality
5e (11).jpg	RESULT: 5	null
5e (12).jpg	RESULT: 5	Low quality
5e (13).jpg	RESULT: 5	Low quality
5e (14).jpg	RESULT: 5	Low quality
5e (15).JPG	RESULT: 5	null
5e (16).jpg	RESULT: 5	Low quality
5e (17).JPG	RESULT: 5	null
5e (18).JPG	RESULT: 50	null
5e (19).JPG	RESULT: 5	null
5e (2).JPG	RESULT: 5	null
5e (20).jpg	RESULT: 5	Low quality
5e (21).JPG	RESULT: 50	null
5e (22).JPG	RESULT: 5	null
5e (23).jpg	RESULT: 5	Low quality
5e (24).jpg	RESULT: 10	Low quality
5e (25).jpg	RESULT: 5	null
5e (26).jpg	RESULT: 5	null
5e (27).jpg	RESULT: 10	null
5e (28).jpg	RESULT: 5	Low quality
5e (29).jpg	RESULT: 5	Low quality
5e (3).jpg	RESULT: 5	Low quality
5e (30).jpg	RESULT: 5	null
5e (31).jpg	RESULT: 5	Low quality
5e (32).jpg	RESULT: 5	Low quality
5e (33).jpg	RESULT: 5	Low quality
5e (34).JPG	RESULT: 5	null
5e (35).JPG	RESULT: 5	null
5e (36).JPG	RESULT: 5	null
5e (37).JPG	RESULT: 5	null
5e (38).JPG	RESULT: 5	null
5e (4).jpg	RESULT: 5	Low quality
5e (40).JPG	RESULT: 5	null
5e (41).JPG	RESULT: 5	null
5e (42).JPG	RESULT: 0	null
5e (43).JPG	RESULT: 0	null
5e (44).JPG	RESULT: 5	null
5e (45).JPG	RESULT: 0	null
5e (46).JPG	RESULT: 5	null
5e (47).JPG	RESULT: 0	null
5e (48).JPG	RESULT: 50	null

5e (49).JPG	RESULT: 5	Low quality
5e (5).jpg	RESULT: 10	Low quality
5e (50).JPG	RESULT: 50	null
5e (51).JPG	RESULT: 5	null
5e (52).JPG	RESULT: 5	Low quality
5e (53).JPG	RESULT: 0	Low quality
5e (54).JPG	RESULT: 5	Low quality
5e (55).JPG	RESULT: 5	Low quality
5e (56).JPG	RESULT: 0	Low quality
5e (57).JPG	RESULT: 5	Low quality
5e (58).JPG	RESULT: 50	Low quality
5e (59).JPG	RESULT: 5	Low quality
5e (6).JPG	RESULT: 5	null
5e (60).JPG	RESULT: 5	Low quality
5e (61).JPG	RESULT: 50	Low quality
5e (7).jpg	RESULT: 5	null
5e (8).jpg	RESULT: 10	null
5e (9).jpg	RESULT: 5	Low quality

Resume of statistics for 5€

Nominal	Total number of banknotes	Rotated	Horizontal position	Correct recognition	No recognition	Incorrect recognition
5 €	**60**	20	40	44	6	10
		% of Rotated	**% of horizontal**	**% of correct recognition**	**% of no recognition**	**% of incorrect recognition**
		33,33%	66,66%	**73,33%**	10,00%	16,66%

On a tables below can be observed that sometimes algorithm recognise false denomination for 5€. Sometimes the program did not detect the denomination of 5€ in denomination recognition phase, then the banknote is recognised as a 20€, because its grey colour is quite similar to 20€ blue colour. As a 10€ and 50€ the denomination of 5€ can be recognized when the first localisation algorithm does not return correct result. This situation can happen because the algorithm of localisation is sensitive for noise and light. Unfortunately, results of next algorithms depend on the result of

localisation. It means that the number of false results can be decreased by improvement of algorithm of localisation.

Statistics for 10 €

File name	Result	Error
10e (1).JPG	RESULT: 10	null
10e (10).JPG	RESULT: 10	null
10e (11).JPG	RESULT: 10	null
10e (12).JPG	RESULT: 10	null
10e (13).JPG	RESULT: 10	null
10e (14).jpg	RESULT: 10	null
10e (15).JPG	RESULT: 10	null
10e (16).JPG	RESULT: 10	null
10e (17).jpg	RESULT: 10	Low quality
10e (18).JPG	RESULT: 5	null
10e (19).jpg	RESULT: 10	null
10e (2).jpg	RESULT: 10	null
10e (20).jpg	RESULT: 10	null
10e (21).jpg	RESULT: 0	Low quality
10e (22).jpg	RESULT: 10	null
10e (23).jpg	RESULT: 10	Low quality
10e (24).jpg	RESULT: 10	null
10e (25).JPG	RESULT: 10	Low quality
10e (26).jpg	RESULT: 10	null
10e (27).jpg	RESULT: 10	Low quality
10e (28).JPG	RESULT: 10	null
10e (29).JPG	RESULT: 50	Can't find the number
10e (3).jpg	RESULT: 10	Low quality
10e (30).JPG	RESULT: 10	null
10e (31).JPG	RESULT: 10	null
10e (32).JPG	RESULT: 10	null
10e (36).jpg	RESULT: 10	null
10e (37).jpg	RESULT: 10	null
10e (38).jpg	RESULT: 0	Low quality
10e (39).jpg	RESULT: 10	null
10e (4).jpg	RESULT: 10	Low quality
10e (40).JPG	RESULT: 10	null
10e (41).JPG	RESULT: 10	null

10e (42).JPG	RESULT: 10	null
10e (43).JPG	RESULT: 10	null
10e (44).JPG	RESULT: 10	null
10e (45).JPG	RESULT: 10	null
10e (46).JPG	RESULT: 10	null
10e (47).JPG	RESULT: 10	null
10e (48).JPG	RESULT: 10	null
10e (5).jpg	RESULT: 10	null
10e (52).JPG	RESULT: 10	Low quality number
10e (53).JPG	RESULT: 10	Low quality number
10e (54).JPG	RESULT: 5	null
10e (56).JPG	RESULT: 10	null
10e (57).JPG	RESULT: 10	null
10e (58).JPG	RESULT: 10	null
10e (59).JPG	RESULT: 10	null
10e (6).JPG	RESULT: 50	Can't find the number
10e (60).JPG	RESULT: 10	null
10e (61).JPG	RESULT: 10	null
10e (7).jpg	RESULT: 10	null
10e (8).jpg	RESULT: 10	Low quality
10e (9).JPG	RESULT: 10	null

Resume of statistics for 10 €

Nominal	Total number of banknotes	Rotated	Horizontal position	Correct recognition	No recognition	Incorrect recognition
10 €	**54**	17	37	48	2	4
		% of Rotated	% of horizontal	% of correct recognition	% of no recognition	% of incorrect recognition
		31,48%	68,52%	**88,88%**	3,70%	7,40%

Results of 10€ are quite good. Usually the denomination is detected in denomination recognition phase. It means that algorithm of denomination recognition works quite good. Some false results in this case are effects of colour recognition that every time return some result. All validations of the image localisation are made before the coulour analisis but are not so much

restrict. It is because colour analisis does not need so well detected location like denomination and hologram recognition algorithms.

Statistics for 20 €

File name	Result	Error
20e (1).jpg	RESULT: 0	null
20e (10).jpg	RESULT: 20	Low quality
20e (11).jpg	RESULT: 20	Low quality
20e (12).jpg	RESULT: 20	Can't find the number
20e (13).jpg	RESULT: 20	Can't find the number
20e (14).jpg	RESULT: 20	Low quality
20e (15).jpg	RESULT: 20	Low quality
20e (16).jpg	RESULT: 20	null
20e (17).jpg	RESULT: 20	Low quality
20e (18).jpg	RESULT: 0	null
20e (19).jpg	RESULT: 20	null
20e (2).jpg	RESULT: 20	null
20e (20).jpg	RESULT: 20	null
20e (22).jpg	RESULT: 20	null
20e (23).jpg	RESULT: 20	Low quality
20e (24).jpg	RESULT: 20	Low quality
20e (25).JPG	RESULT: 20	Can't find the number
20e (26).JPG	RESULT: 20	Low quality
20e (27).JPG	RESULT: 20	null
20e (28).JPG	RESULT: 0	Low quality
20e (3).jpg	RESULT: 20	Low quality
20e (30).JPG	RESULT: 10	Low quality
20e (31).JPG	RESULT: 20	Low quality
20e (32).JPG	RESULT: 20	null
20e (33).JPG	RESULT: 20	Low quality
20e (34).JPG	RESULT: 20	null
20e (4).jpg	RESULT: 20	Low quality
20e (5).jpg	RESULT: 20	Low quality
20e (6).jpg	RESULT: 5	null
20e (7).jpg	RESULT: 20	null
20e (8).jpg	RESULT: 20	null

20e (9).jpg	RESULT: 20	null

Resume of statistics for 20 €

Nominal	Total number of banknotes	Rotated	Horizontal position	Correct recognition	No recognition	Incorrect recognition
20 €	**32**	11	21	27	3	2
		% of Rotated	**% of horizontal**	**% of correct recognition**	**% of no recognition**	**% of incorrect recognition**
		34,37%	65,63%	**84,38%**	9,37%	6,25%

20€ has one of the best rate of recognition that is about 84%. The reason of this situation is that colour recognition algorithm is less sensitive to the banknote in the correct position. The banknote can be moved or rotated, because algorithm just check if the colour of the object is more red or more blue. This can cause some false recognition of banknotes as is shown in statistics of 5€ and 10€.

Statistics for 50 €

File name	Result	Error
50e (1).jpg	RESULT: 50	Low quality
50e (10).jpg	RESULT: 50	null
50e (11).jpg	RESULT: 20	null
50e (12).jpg	RESULT: 50	null
50e (13).jpg	RESULT: 50	null
50e (14).jpg	RESULT: 50	null
50e (15).jpg	RESULT: 50	null
50e (16).jpg	RESULT: 50	null
50e (17).jpg	RESULT: 0	null
50e (18).jpg	RESULT: 50	null
50e (2).jpg	RESULT: 20	Can't find the banknote. (HoughTransform)
50e (22).jpg	RESULT: 0	null
50e (24).jpg	RESULT: 0	null

50e (25).jpg	RESULT: 0	null
50e (26).jpg	RESULT: 50	null
50e (27).jpg	RESULT: 50	null
50e (28).jpg	RESULT: 50	Can't find the number
50e (29).jpg	RESULT: 20	null
50e (3).jpg	RESULT: 50	null
50e (30).jpg	RESULT: 50	null
50e (31).jpg	RESULT: 50	null
50e (32).jpg	RESULT: 50	null
50e (33).jpg	RESULT: 20	null
50e (34).jpg	RESULT: 50	null
50e (35).jpg	RESULT: 50	Can't find the number
50e (36).jpg	RESULT: 50	null
50e (37).jpg	RESULT: 50	null
50e (38).jpg	RESULT: 50	null
50e (39).jpg	RESULT: 0	null
50e (4).jpg	RESULT: 50	null
50e (40).jpg	RESULT: 50	null
50e (44).jpg	RESULT: 0	null
50e (46).jpg	RESULT: 0	null
50e (49).JPG	RESULT: 50	null
50e (5).jpg	RESULT: 50	Can't find the number
50e (50).JPG	RESULT: 50	null
50e (51).JPG	RESULT: 10	null
50e (52).JPG	RESULT: 10	null
50e (53).JPG	RESULT: 50	null
50e (54).JPG	RESULT: 50	null
50e (55).JPG	RESULT: 50	null
50e (56).JPG	RESULT: 50	null
50e (57).JPG	RESULT: 50	Low quality
50e (58).JPG	RESULT: 50	null
50e (59).JPG	RESULT: 50	null
50e (6).jpg	RESULT: 20	Can't find the banknote. (HoughTransform)
50e (60).JPG	RESULT: 50	Can't find the number
50e (61).JPG	RESULT: 50	null
50e (62).JPG	RESULT: 50	null
50e (63).JPG	RESULT: 50	null
50e (64).JPG	RESULT: 50	null
50e (7).jpg	RESULT: 20	null
50e (8).jpg	RESULT: 50	null
50e (9).jpg	RESULT: 50	null

Resume of statistics for 50 €

Nominal	Total number of banknotes	Rotated	Horizontal position	Correct recognition	No recognition	Incorrect recognition
50 €	**54**	22	32	41	7	8
		% of Rotated	**% of horizontal**	**% of correct recognition**	**% of no recognition**	**% of incorrect recognition**
		40,74%	59,26%	**75,92%**	12,96%	14,81%

50€ can be recognized in two ways: in hologram analysis or in colour recognition. Hologram analysis is very precise and detect 50€ with high probability. The problem that the algorithm needs high image quality and perfect result of localisation algorithm. Because the localisation algorithm is so much universal, a lot of times happen that only a hologram analysis is not enough. More, that the hologram is often not in adequate quality because of light reflexes. For this reason program uses additional algorithm of coulour analysis for 50€ recognition. In this way 50€ even if can not be recognise by the hologram, can be perfectly recognised by colour, even is its location is not exact.

50€ sometimes is confused with 20€. It is because colour of the banknote can change significantly for every image. Colour depends on light, if there is any shadow and of localisation algorithm. It can happen that the localisation algorithm fails and colour recognition algorithm recognise colour of the background, not of the banknote. Unfortunately, colour is the most variable value in the image.

Global resume of statistics

Nominal	Total number of banknotes	Rotated	Horizontal position	Correct recognition	No recognition	Incorrect recognition
5 €	60	20	40	44	6	10
10 €	54	17	37	48	2	4
20 €	32	11	21	27	3	2
50 €	54	22	32	41	7	6
Totals:	**0**	**70**	**130**	**160**	**18**	**22**
		% of Rotated	**% of horizontal**	**% of correct recognition**	**% of no recognition**	**% of incorrect recognition**
		35,00%	**65,00%**	**77,50%**	**9,00%**	**11,00%**

Correct result of denomination recognition for 200 banknotes arrives to 77,5%. It could be improved by increasing efficiency of the localisation algorithm and use more specific characteristic of Euro banknote (like for example European flag) to detect where exactly the bill is. In this way it is possible to decrease the number of incorrect recognitions, that nowadays is quite high. On the other hand algorithms used in the program are quite simple and very fast, there is no sophisticated operation, everything is based on histograms, that are just basic sums, so the result of so simple operations in so hide area of kind of images is quite good.

Although correct recognition is not 100%, there are some denominations bad recognised, but what is important for this project is to show new algorithm for image processing in action. The algorithm very easy can arrive to near 100 % level by setting a static position of the banknote. The problem that if the main purpose is simplified by using the static position, the general algorithm would lost a lot of research value and it would be just another gadget for the mobile phone.

On the other hand localisation and rotation algorithm is something fresh and new and has strong research values and because of it is used in this project. In this way the project was difficult to develop and is a perfect example of well-done research work.

8. RESUME OF OPTIMIZATION ALGORITHMS

In the program are used some methods of optimisation that significantly decrease time of the execution and increase number of correctly recognised denominations.

The most significant reduction of the execution time is achieved in direct, partial rotation algorithm. The partial rotation is about ten times faster than the standard indirect rotation. The algorithm can be still improved by eliminating rotation of RGB image and making all calculations with the usage of a known angle.

Another optimisation is a reduction of Hough lines from nearly 100 to 5-10 by employing high threshold and the requirement of high number of neighbours. The execution time is greatly influenced by analysis of the position of a banknote just in vertical direction.

Otsu algorithm can also be used to improve the effectiveness in some way. It calculates the most optimal threshold for every image.

Slightly controversial optimisation is to execute the algorithm that increases the quality and details of the image after rotation. On the one hand less details means less Hough lines and faster rotation, on the other hand the localisation algorithm in case of low quality is executed two times. On the basis of the global statistics can be said that this method is faster and more effective than improving the quality of the image at the beginning, or complete abandonment of the improvement of image quality. Of course, every thing depends on the image. If the image needs rotation then improvement of quality after rotation is huge optimisation. If the image does not need rotation, localisation algorithm is executed two times in

place of just one time and better option would be to improve quality at the beginning. However, general time of execution is decreased.

For image manipulation *Transformation* class has 3 global attributes: thresholded image with horizontal contour, thresholded image with vertical contour and the RGB colour image. *Transformation* class is special class that is responsible for all algorithms execution. The class implements design pattern *Singleton* that ensures that the class *Transformation* has only one instance, and provides a global point of access to it. Thanks to this program can show every step of used algorithms, without making all calculations another time.

Other very significant improvement is the optimisation of the source code by avoiding Internal Getters/Setters and using Enhanced For Loop Syntax. The most significant is For Loop Syntax as presented in the example below:

```
public void zero() {
    int sum = 0;
         for (int i = 0; i < mArray.length; ++i) {
            sum += mArray[i].mSplat;
         }
}

public void one() {
         int sum = 0;
         Foo[] localArray = mArray;
         int len = localArray.length;
         for (int i = 0; i < len; ++i) {
            sum += localArray[i].mSplat;
          }
}
```

The method zero() call every iteration mArray.length, that makes it 3 times slower than method one(). In the project all iterations are in form of method one.

On the source code level is possible to use Android NDK that allow to program the most critical parts of code. It also should decrease time of execution of some algorithms.

On the other hand the project results can be significantly improved by using other, additional recognition algorithms and other ways of optimisation. Below are presented ideas, what nowadays are not completely implemented and which are scheduled in the next version of the program.

One of the ideas for the future is to improve the algorithm of detection of contours by using the flag (European flag in the left side of the bill), which unfortunately nowadays is not enough improved and its results are not quite significantly in order to exploit it. However, the algorithm of flag localisation gives a way to find the position of the banknote easier and more directly and has big potential. The algorithm just needs to check the position of very small region of European flag. In this way it would be also possible to recognise multiple banknotes or banknotes with complex background, that would be really huge improvement. The algorithm of European flag localisation has also strong research value and it is some additional idea for the future.

The easier idea for increment of efficiency of application is to set static, well-known position of the banknote. However, from the research point of view is not interesting idea, because simplifies so much the general problem.

Other idea of optimisation is to give up the Java Garbage collector and allocate the memory manually. Similar solution is to use the native languages C or C++, where memory allocation is always manual and are a lot of faster than Java. All of those solutions would improve internal execution time of calculations.

9. CONCLUSIONS

The main aim of this diploma project was to create an application, that is amusing, useful and gives a possibility to demonstrate a new image processing algorithms and gives a wide area for optimisation and for research. All of those aims are achieved in the presented project. There is also described various ways of optimisation at algorithms level and at the source code level as well. Some of those solutions are new and unconventional, for instance direct rotation or algorithm of banknote localisation, what makes this project a perfect example of the research work.

The project contains more that 5000 lines of source code, about 200 pages of documentation, available website and about 80% of global efficiency. But the biggest value of this thesis is its research value and very original implementation of new ideas and new algorithms, that works very well in mobile phone. The implemented algorithms can be used in other areas of image processing usage, because are quite universal. Additional application for JavaSE allows to tests and improve image algorithms in the future, allows to generate statistics and it is accessible for everybody because id working with all popular operating systems of personal computers. It makes the project easy to extend and personalise.

In general, algorithms and programs for image manipulation and image processing are kind of "Never ending story" and always there is something to improve. As it was told at the first chapter of this paper: every image is eternal world with different requirements and characteristics. The creation of the program that would be able to recognise the complex pattern in 100% in all possible images, that are possible to recognize by a human, is extremely difficult, but who knows, maybe one day the computer vision would arrive to the level of intelligence of human vision and brain.

Nowadays computer vision is one of the main research areas on the world and this project is one small, experimental step for image processing and mobile phone integration.

APPENDIX 1: Java documentation

Main, the most important and significant implemented packages:

Package cat.uvic.calculs – algorithms and calculations

Class Summary	
FindRegion	class responsible for looking for interesting region based on histograms (horizontal and vertical)
Gray8Dilate	dilate a Gray thresholded image
Gray8Erose	erode a Gray thresholded image
Gray8HistHorizontal	horizontal histogram for Gray image.
Gray8HistVertical	vertical histogram for Gray image.
Gray8Subimage	cut part of a Gray image
GrayFastRotate	rotate a Gray thresholded image
GrayRotate	rotate a Gray image
HistRegion	represents part of a histogram, that is zero or more that zero.
HistRegions	all objects HistResion of one histogram together.
HoughLine	Represents a linear line as detected by the hough transform.
HoughTransform	Note: This class is based on original code from: http://vase.essex.ac.uk/software/HoughTransform/HoughTransform.java.html If you represent a line as: x cos(theta) + y sin (theta) = r
HsvHistogram	colour histogram for an image in HSV colorspace.
OtsuThresholder	Calculates optimal Otsu threshold
Rgb2YCbCr	pass RGB image to YcbCr colorspace
RgbHistEqualize	equalize histograms of RGB image
RgbImageSubImage	cut defined part of RGB image
RgbRotate	rotate RGB image
Transformation	the main class that is putting all operations together.

Exception Summary	
ImageException	Exception in some image processing algorithm

Class FindRegion

```
java.lang.Object
  └ cat.uvic.calculs.FindRegion
```

```
public class FindRegion extends java.lang.Object
```

class responsible for looking for interesting region based on histograms (horizontal and vertical)

Author:
ANNA

Constructor Summary	
FindRegion(boolean marg)	
FindRegion(boolean marg, int pV, int pH)	
Method Summary	
Rect	findRegionOfInteres(int[] histH, int[] histV)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Constructor Detail	

```
public FindRegion(boolean marg)
```

Parameters:
marg - margin value

```
public FindRegion(boolean marg,
                  int pV,
                  int pH)
```

Parameters:
marg - global margin value
pV - border value for vertical histogram
pH - border value for horizontal histogram

Method Detail

```
public Rect findRegionOfInteres(int[] histH,
                                int[] histV)
```

Parameters:
histH - horizontal histogram
histV - vertical histogram
Returns:
rectangle of interesting region

Class Gray8Dilate

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.Gray8Dilate
```

```
public class Gray8Dilate extends PipelineStage
```

dilate a Gray thresholded image

Author:
ANNA

Constructor Summary	
Gray8Dilate() Creates a new instance of Gray8Dilate	
Gray8Dilate(int _many) Creates a new instance of Gray8Dilate	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public Gray8Dilate(int _many)
```

Creates a new instance of Gray8Dilate
Parameters:
_many - how many times Dilation should repeat

---

public Gray8Dilate()

Creates a new instance of Gray8Dilate

Method Detail

```
public void push(Image img)
          throws Error
```

Description copied from class: PipelineStage

Actual processing is done in the derived class here.

Specified by:

push in class PipelineStage

Parameters:

img - the input image

Throws:

Error - typically, when the image is not of the expected type.

Class Gray8Erose

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.Gray8Erose
```

```
public class Gray8Eroseextends PipelineStage
```

erode a Gray thresholded image

Author:

ANNA

Constructor Summary	
Gray8Erose() Creates a new instance of Gray8Erose	
Gray8Erose(int _many)	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public Gray8Erose(int _many)
```

Parameters:

_many - how many times Erosion should repeat

```
public Gray8Erose()
```

Creates a new instance of Gray8Erose

Method Detail

```
public void push(Image img)
          throws java.lang.Error
```

Description copied from class: PipelineStage

Actual processing is done in the derived class here.

Specified by:

push in class PipelineStage

Parameters:

img - the input image

Throws:

Error - typically, when the image is not of the expected type.

java.lang.Error

Class Gray8HistHorizontal

```
java.lang.Object
  └cat.uvic.calculs.Gray8HistHorizontal
```

```
public class Gray8HistHorizontalextends java.lang.Object
```

horizontal histogram for Gray image.

Author:

ANNA

Constructor Summary	
Gray8HistHorizontal() Creates a new instance of Gray8HistHorizontal *	
Method Summary	
static int[]	computeHistogram(Gray8Image image)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	

Constructor Detail

```
public Gray8HistHorizontal()
```

Creates a new instance of Gray8HistHorizontal *

Method Detail

```
public static int[] computeHistogram(Gray8Image image)
```

Parameters:
image -
Returns:
array of integer that represents sequence of histogram values

Class Gray8HistVertical

```
java.lang.Object
  └ cat.uvic.calculs.Gray8HistVertical
```

```
public class Gray8HistVerticalextends java.lang.Object
```

vertical histogram for Gray image.

Author:
ANNA

Constructor Summary	
Gray8HistVertical()	
Method Summary	
static int[]	computeHistogram(Gray8Image image)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	

Constructor Detail

```
public Gray8HistVertical()
```

Method Detail

```
public static int[] computeHistogram(Gray8Image image)
```

Parameters:
image -
Returns:
array of integer that represents sequence of histogram values

Class Gray8Subimage

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.Gray8Subimage
```

```
public class Gray8Subimageextends PipelineStage
```

cut part of a Gray image

Author:
ANNA

Constructor Summary	
Gray8Subimage(Rect _rect) Creates a new instance of Gray8Subimage.	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public Gray8Subimage(Rect _rect)
              throws Error
```

Creates a new instance of Gray8Subimage.

Parameters:
_rect -

Throws:
Error - if the target width or height is less than or equal to zero.

Method Detail

```
public void push(Image img)
          throws Error
```

Description copied from class: PipelineStage
Actual processing is done in the derived class here.

Specified by:
push in class PipelineStage

Parameters:
img - the input image

Throws:

Error - typically, when the image is not of the expected type.

Class GrayFastRotate

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.GrayFastRotate
```

```
public class GrayFastRotateextends PipelineStage
```

rotate a Gray thresholded image

Author:
ANNA

Constructor Summary	
GrayFastRotate(double theta)	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public GrayFastRotate(double theta)
```

Parameters:
theta - arc value

Method Detail

```
public void push(Image img)
         throws Error
```

Description copied from class: PipelineStage
Actual processing is done in the derived class here.

Specified by:
push in class PipelineStage

Parameters:
img - the input image

Throws:
Error - typically, when the image is not of the expected type.

Class GrayRotate

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.GrayRotate
```

```
public class GrayRotateextends PipelineStage
```

rotate a Gray image

Author:
ANNA

Constructor Summary	
GrayRotate(double theta)	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public GrayRotate(double theta)
           throws Error
```

Parameters:
theta - arc value
Throws:
Error

Method Detail

```
public void push(Image img)
           throws Error
```

Description copied from class: PipelineStage
Actual processing is done in the derived class here.
Specified by:
push in class PipelineStage
Parameters:
img - the input image
Throws:
Error - typically, when the image is not of the expected type.

Class HistRegion

```
java.lang.Object
  └ cat.uvic.calculs.HistRegion
```

```
public class HistRegionextends java.lang.Object
```

represents part of a histogram, that is zero or more that zero. Used in looking for number pattern.

Author:
ANNA

Field Summary	
static java.lang.String	Tipus_ONE
static java.lang.String	Tipus_ZERO
Constructor Summary	
HistRegion(java.lang.String type)	
Method Summary	
int	getCount()
double	getPercent()
double	getPercentLocal()
java.lang.String	getStrPercent()
java.lang.String	getStrPercentLocal()
java.lang.String	getType()
void	setCount(int count)
void	setPercent(double percent)
void	setPercentLocal(double percentLocal)
void	setType(java.lang.String type)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Field Detail	

```
public static java.lang.String Tipus_ZERO
```

```
public static java.lang.String Tipus_ONE
```

Constructor Detail

`public HistRegion(java.lang.String type)`

Parameters:

type -

Method Detail

`public double getPercentLocal()`

Returns:

local percent of region

`public void setPercentLocal(double percentLocal)`

Parameters:

percentLocal -

`public java.lang.String getStrPercent()`

Returns:

percent of region in String (#,##) format

`public java.lang.String getStrPercentLocal()`

Returns:

local percent of region in String (#,##) format

`public double getPercent()`

Returns:

percent of region

`public void setPercent(double percent)`

Parameters:

percent -

`public int getCount()`

Returns:

count of pixels of region

`public void setCount(int count)`

Parameters:

count -

`public java.lang.String getType()`

Returns:

type of HistRegion (ONE or ZERO)

```
public void setType(java.lang.String type)
```

Parameters:

type -

Class HistRegions

```
java.lang.Object
  └cat.uvic.calculs.HistRegions
```

```
public class HistRegions extends java.lang.Object
```

all objects HistResion of one histogram together.

Author:

ANNA

Constructor Summary	
HistRegions(long total)	

Method Summary	
boolean	analise50(boolean lowQ)
int	analiseNumber(boolean lowQ)
java.util.ArrayList<HistRegion>	getHistRegions()
java.lang.String	getResult()
long	getTotal()
long	getWidth()
void	setHistRegions(java.util.ArrayList<HistRegion> histRegions)
void	setResult(java.lang.String result)
void	setTotal(long total)
void	setWidth(long width)
java.lang.String	toString()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

```
public HistRegions(long total)
```

Parameters:

total -

Method Detail

```
public long getWidth()
```

Returns:

width of regions

```
public void setWidth(long width)
```

Parameters:

width -

```
public java.lang.String getResult()
```

Returns:

result

```
public void setResult(java.lang.String result)
```

Parameters:

result -

```
public long getTotal()
```

Returns:

total

```
public void setTotal(long total)
```

Parameters:

total -

```
public java.util.ArrayList<HistRegion> getHistRegions()
```

Returns:

array of regions

```
public void
setHistRegions(java.util.ArrayList<HistRegion> histRegions)
```

Parameters:

histRegions -

```
public int analiseNumber(boolean lowQ)
```

Parameters:

lowQ -

Returns:

10 if it is 10 €, 5 if it is 5€, 0 when not found, 2050 if can be 20€ or 50€

```
public java.lang.String toString()
```
Overrides:
toString in class java.lang.Object

```
public boolean analise50(boolean lowQ)
```
Parameters:
lowQ -
Returns:
true if is 50 €

Class HoughLine

```
java.lang.Object
  └cat.uvic.calculs.HoughLine
```

```
public class HoughLineextends java.lang.Object
```

Represents a linear line as detected by the hough transform. This line is represented by an angle theta and a radius from the centre.

Version:
1.0
Author:
Olly Oechsle, University of Essex, Date: 13-Mar-2008

Field Summary	
double	theta theta parameter (arc)

Constructor Summary	
HoughLine(double theta, double r) Initialises the hough line	

Method Summary	
void	draw(Bitmap image, int color) Draws the line on the image of your choice with the RGB colour of your choice.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

```
public double theta
```
theta parameter (arc)

Constructor Detail

```
public HoughLine(double theta,
                 double r)
```

Initialises the hough line

Parameters:

theta - arc value

r - distance value

Method Detail

```
public void draw(Bitmap image,
                 int color)
```

Draws the line on the image of your choice with the RGB colour of your choice.

Parameters:

image -

color -

Class HoughTransform

```
java.lang.Object
  └cat.uvic.calculs.HoughTransform
```

```
public class HoughTransform extends java.lang.Object
```

Note: This class is based on original code from:
http://vase.essex.ac.uk/software/HoughTransform/HoughTransform.java.html
If you represent a line as: x cos(theta) + y sin (theta) = r

Constructor Summary	
HoughTransform(int width, int height) Initialises the hough transform.	

Method Summary	
void	addPoint(int x, int y) Adds a single point to the hough transform.
void	addPoints(Gray8Image image) Adds points from an image.
int	getHighestValue()
java.util.Vector<HoughLine>	getLines(int threshold) Once points have been added in some way this

	method extracts the lines and returns them as a Vector of HoughLine objects, which can be used to draw on the
void	initialise() Initialises the hough array.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public HoughTransform(int width,
                      int height)
```

Initialises the hough transform. The dimensions of the input image are needed in order to initialise the hough array.

Parameters:

width - The width of the input image

height - The height of the input image

Method Detail

```
public void initialise()
```

Initialises the hough array. Called by the constructor so you don't need to call it yourself, however you can use it to reset the transform if you want to plug in another image (although that image must have the same width and height)

```
public void addPoints(Gray8Image image)
```

Adds points from an image. The image is assumed to be greyscale black and white, so all pixels that are not black are counted as edges. The image should have the same dimensions as the one passed to the constructor.

Parameters:

image – input image

```
public void addPoint(int x,
                     int y)
```

Adds a single point to the hough transform. You can use this method directly if your data isn't represented as a buffered image.

Parameters:

x – x parameter

y – y parameter

```
public java.util.Vector<HoughLine> getLines(int threshold)
```

Once points have been added in some way this method extracts the lines and returns them as a Vector of HoughLine objects, which can be used to draw on the

Parameters:

threshold - The percentage threshold above which lines are

determined from the hough array

Returns:

Result vector of Hough lines

```
public int getHighestValue()
```

Returns:

highest value in the hough array

Class HsvHistogram

```
java.lang.Object
  └ cat.uvic.calculs.HsvHistogram
```

```
public class HsvHistogram extends java.lang.Object
```

colour histogram for an image in HSV colorspace.

Author:

ANNA

Field Summary	
int[]	colors
static java.lang.String	H_HIST
static java.lang.String	S_HIST
static java.lang.String	V_HIST

Constructor Summary	
HsvHistogram()	
HsvHistogram(java.lang.String tipus)	

Method Summary	
int[]	computeHistogram(RgbImage image)
int[]	computeHistogram(RgbImage image, Gray8Image gray)
java.lang.String	getLog()
java.lang.String	getTipus()
void	setLog(java.lang.String line)
void	setTipus(java.lang.String tipus)
long	sumHistogram(int[] hist)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

```
public static java.lang.String H_HIST
```

```
public static java.lang.String V_HIST
```

```
public static java.lang.String S_HIST
```

```
public int[] colors
```

Constructor Detail

```
public HsvHistogram(java.lang.String tipus)
```

Parameters:

tipus -

```
public HsvHistogram()
```

Method Detail

```
public java.lang.String getLog()
```

Returns:

log (for debug)

```
public void setLog(java.lang.String line)
```

Parameters:

line -

```
public java.lang.String getTipus()
```

Returns:

tipus of histogram

```
public void setTipus(java.lang.String tipus)
```

Parameters:

tipus – set tipus of histogram

```
public int[] computeHistogram(RgbImage image)
```

Parameters:

image – input image

Returns:

histogram

```
public int[] computeHistogram(RgbImage image,
```

```
                                Gray8Image gray)
                         throws ImageException
```

Parameters:
image – input image
gray - mask
Returns:
histogram
Throws:
ImageException

```
public long sumHistogram(int[] hist)
```

Parameters:
hist – input histogram
Returns:
sum of histogram values

Class OtsuThresholder

```
java.lang.Object
  └cat.uvic.calculs.OtsuThresholder
```

```
public class OtsuThresholderextends java.lang.Object
```

Author:
jppr...@gmail.com (from project textdetection - http://code.google.com/p/textdetection/source/checkout)

Constructor Summary	
OtsuThresholder()	

Method Summary	
int	doThreshold(byte[] srcData, byte[] monoData)
int[]	getHistData()
int	getMaxLevelValue()
int	getThreshold()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public OtsuThresholder()
```

Method Detail

```
public int[] getHistData()
```
Returns:
histData

```
public int getMaxLevelValue()
```
Returns:
maxLevelValue

```
public int getThreshold()
```
Returns:
threshold

```
public int doThreshold(byte[] srcData,
                       byte[] monoData)
```
Parameters:
srcData -
monoData -
Returns:
threshold

Class Rgb2YCbCr

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.Rgb2YCbCr
```

```
public class Rgb2YCbCrextends PipelineStage
```

pass RGB image to YcbCr colorspace

Author:
ANNA

Constructor Summary	
Rgb2YCbCr()	

Method Summary	
void	push(Image imageInput) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public Rgb2YCbCr()
```

Method Detail

```
public void push(Image imageInput)
          throws Error
```

Description copied from class: PipelineStage
Actual processing is done in the derived class here.

Specified by:
push in class PipelineStage

Parameters:
imageInput - the input image

Throws:
Error - typically, when the image is not of the expected type.

Class RgbHistEqualize

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.RgbHistEqualize
```

```
public class RgbHistEqualizeextends PipelineStage
```

equalize histograms of RGB image

Author:
ANNA

Constructor Summary	
RgbHistEqualize()	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public RgbHistEqualize()
```

Method Detail

```
public void push(Image img)
          throws Error
```

Description copied from class: PipelineStage

Actual processing is done in the derived class here.

Specified by:

push in class PipelineStage

Parameters:

img - the input image

Throws:

Error - typically, when the image is not of the expected type.

Class RgbImageSubImage

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.RgbImageSubImage
```

```
public class RgbImageSubImageextends PipelineStage
```

cut defined part of RGB image

Author:

ANNA

Constructor Summary	
RgbImageSubImage(Rect _rect)	

Method Summary	
void	push(Image img) Reduces an RgbImage by a factor horizontally and vertically through averaging.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public RgbImageSubImage(Rect _rect)
                 throws Error
```

Parameters:
_rect -
Throws:
Error

Method Detail

```
public void push(Image img)
          throws Error
```

Reduces an RgbImage by a factor horizontally and vertically through averaging. The reduction factor must be an even multiple of the image size.
Specified by:
push in class PipelineStage
Parameters:
img - the input image.
Throws:
Error - if the input image is not gray, or the reduction factor does not evenly divide the image size.

Class RgbRotate

```
java.lang.Object
  └jjil.core.PipelineStage
      └cat.uvic.calculs.RgbRotate
```

```
public class RgbRotate extends PipelineStage
```

rotate RGB image

Author:
ANNA

Constructor Summary	
RgbRotate(double theta)	

Method Summary	
void	push(Image img) Actual processing is done in the derived class here.

Methods inherited from class jjil.core.PipelineStage
getFront, isEmpty

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public RgbRotate(double theta)
          throws Error
```

Parameters:
theta -
Throws:
Error

Method Detail

```
public void push(Image img)
          throws Error
```

Description copied from class: PipelineStage
Actual processing is done in the derived class here.
Specified by:
push in class PipelineStage
Parameters:
img - the input image
Throws:
Error - typically, when the image is not of the expected type.

Class Transformation

```
java.lang.Object
  └cat.uvic.calculs.Transformation
```

```
public class Transformation extends java.lang.Object
```

the main class that is putting all operations together. Is responsible for localisation, checking position, rotating and recognition of a banknote. Class implements Singleton pattern. It means that there is just one instance of the class in the program.

Author:
ANNA

Field Summary	
java.lang.String	error
static java.lang.String	EUR_10
static java.lang.String	EUR_20
static java.lang.String	EUR_5

static java.lang.String	EUR_50
Method Summary	
int	analiseColor()
int	analiseNumber()
void	checkPosition()
RgbImage	cutImage()
static void	destroy() destroy the instance of Transformation class
Rect	findBill(Image img, boolean margin)
Bitmap	getCurrentImg()
Image	getImg()
static Transformation	getInstance(Bitmap _img)
java.util.Vector<HoughLine>	getLines()
int	getRecognition()
Rect	getRegion()
boolean	isWrong()
boolean	recognise50()
void	setCurrentImg(Image currentImg)
void	setFinalResult() Set final result image
void	setImg(Image img)
void	setLines(java.util.Vector<HoughLine> lines)
void	setRecognition(int recognition)
void	setRegion(Rect region)
void	setWrong(boolean wrong)
ResultBitmap	transform()
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Field Detail	

```
public java.lang.String error
```

```
public static java.lang.String EUR_5
```

```
public static java.lang.String EUR_10
```

```
public static java.lang.String EUR_20
```

```
public static java.lang.String EUR_50
```

Method Detail

```
public boolean isWrong()
```

Returns:

true if the image is corrupted, false if everything is correct

---

```
public void setWrong(boolean wrong)
```

Parameters:

wrong -

```
public Bitmap getCurrentImg()
                    throws Error
```

Returns:

Bitmap of current processed image

Throws:

Error

```
public void setCurrentImg(Image currentImg)
```

Parameters:

currentImg -

```
public int getRecognition()
```

Returns:

number of recognition (5,10,20,50,0 or 2050)

```
public void setRecognition(int recognition)
```

Parameters:

recognition -

```
public java.util.Vector<HoughLine> getLines()
```

Returns:

vector of HoughLines (if any)

```
public void setLines(java.util.Vector<HoughLine> lines)
```

Parameters:

lines -

```
public Rect getRegion()
```

Returns:
rectangle of current processed region

```
public void setRegion(Rect region)
```
Parameters:
region -

```
public static Transformation getInstance(Bitmap _img)
                                  throws java.io.IOException
```
Parameters:
_img -
Returns:
instance of Transformation
Throws:
java.io.IOException

```
public static void destroy()
```
destroy the instance of Transformation class

```
public ResultBitmap transform()
                       throws Error,
                              java.lang.Exception
```
Returns:
Bitmap of final result of transformation
Throws:
Error
java.lang.Exception

```
public Image getImg()
```
Returns:
img

```
public void setImg(Image img)
```
Parameters:
img -

```
public void checkPosition()
                    throws Error,
                           java.lang.Exception
```
Throws:
Error
java.lang.Exception

```
public Rect findBill(Image img,
```

```
                    boolean margin)
             throws Error
```

Parameters:

img -

margin -

Returns:

rectangle where a bill is

Throws:

Error

```
public int analiseColor()
                 throws Error,
                        ImageException
```

Returns:

recognized denomination or 0

Throws:

Error

ImageException

```
public int analiseNumber()
                  throws Error
```

Returns:

result of number analysis (5, 10, 0 or 2050€)

Throws:

Error

```
public RgbImage cutImage()
                  throws Error
```

Returns:

cut of RgbImage

Throws:

Error

```
public boolean recognise50()
                    throws Error
```

Returns:

true if denomination is 50€ and false if not

Throws:

Error

```
public void setFinalResult()
```

Set final result image

Package cat.uvic.ui – graphical user interface

Class Summary	
FinalActivity	Result activity
MainActivity	Main class that represents the main menu with all Buttons listeners implemented
MyGallery	Take photo from the gallery
MyView	Customized View that show the image with drawn region of interes.
NextActivity	Activity responsible for locate the banknote
NextActivity 2	Activity responsible for the banknote rotation
NextActivity 3	Activity responsible for hologram recognition
NextActivity 4	Activity responsible for denomination analysis
NextActivity 5	Activity responsible for color analysis

Class FinalActivity

```
java.lang.Object
  └Activity
      └cat.uvic.ui.FinalActivity
```

```
public class FinalActivityextends Activity
```

Result activity

Author:
ANNA

Constructor Summary	
FinalActivity()	
Method Summary	
void	onCreate(Bundle savedInstanceState)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	

Constructor Detail

```
public FinalActivity()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class MainActivity

```
java.lang.Object
  └Activity
      └cat.uvic.ui.MainActivity
```

```
public class MainActivity extends Activity
```

Main class that represents the main menu with all Buttons listeners implemented

Author:
ANNA

Constructor Summary	
MainActivity()	
Method Summary	
void	onCreate(Bundle savedInstanceState)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Constructor Detail	

```
public MainActivity()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class MyGallery

```
java.lang.Object
  └Activity
      └cat.uvic.ui.MyGallery
```

```
public class MyGalleryextends Activity
```

Take photo from the gallery

Author:
ANNA

Constructor Summary	
MyGallery()	

Method Summary	
void	onCreate(Bundle savedInstanceState)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public MyGallery()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class MyView

```
java.lang.Object
  └View
      └cat.uvic.ui.MyView
```

```
public class MyViewextends View
```

Customized View that show the image with drawn region of interes.

Author:
ANNA

Constructor Summary	
MyView(Context context)	
MyView(Context context, AttributeSet attrs)	
MyView(Context context, Bitmap bitmap, Rect region, java.lang.String string)	

Method Summary	
Bitmap	getBtm()

Rect	getRect()
java.lang.String	getResult()
void	setBtm(Bitmap btm)
void	setRect(Rect rect)
void	setResult(java.lang.String result)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public MyView(Context context)
```

Parameters:
context -

```
public MyView(Context context,
              AttributeSet attrs)
```

Parameters:
context -
attrs -

```
public MyView(Context context,
              Bitmap bitmap,
              Rect region,
              java.lang.String string)
```

Parameters:
context - current Context
bitmap - Image bitmap
region - Rectangle region
string - result string Constructor of My View

Method Detail

```
public Bitmap getBtm()
```

Returns:
result image Bitmap

```
public void setBtm(Bitmap btm)
```

Parameters:
btm -

```
public Rect getRect()
```

Returns:
region of analysis

```
public void setRect(Rect rect)
```
Parameters:
rect -

```
public java.lang.String getResult()
```
Returns:
result of recognition

```
public void setResult(java.lang.String result)
```
Parameters:
result -

Class NextActivity

```
java.lang.Object
  └Activity
      └cat.uvic.ui.NextActivity
```

```
public class NextActivityextends Activity
```

Activity responsible for locate the banknote

Author:
ANNA

Constructor Summary	
NextActivity()	

Method Summary	
void	onCreate(Bundle savedInstanceState)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public NextActivity()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class NextActivity2

```
java.lang.Object
```

```
  └Activity
      └cat.uvic.ui.NextActivity2
```

```
public class NextActivity2extends Activity
```

Activity responsible for the banknote rotation

Author:
ANNA

Constructor Summary	
NextActivity2()	

Method Summary	
void	onCreate(Bundle savedInstanceState)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public NextActivity2()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class NextActivity3

```
java.lang.Object
  └Activity
      └cat.uvic.ui.NextActivity3
```

```
public class NextActivity3extends Activity
```

Activity responsible for hologram recognition

Author:
ANNA

Constructor Summary	
NextActivity3()	

Method Summary

void	onCreate(Bundle savedInstanceState)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Constructor Detail	

```
public NextActivity3()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class NextActivity4

```
java.lang.Object
  └Activity
      └cat.uvic.ui.NextActivity4
```

```
public class NextActivity4extends Activity
```

Activity responsible for denomination analysis

Author:
ANNA

Constructor Summary	
NextActivity4()	
Method Summary	
void	onCreate(Bundle savedInstanceState)
Methods inherited from class java.lang.Object	
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait	
Constructor Detail	

```
public NextActivity4()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

Class NextActivity5

```
java.lang.Object
  └Activity
      └cat.uvic.ui.NextActivity5
```

```
public class NextActivity5extends Activity
```

Activity responsible for color analysis

Author:
ANNA

Constructor Summary	
NextActivity5()	

Method Summary	
void	onCreate(Bundle savedInstanceState)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public NextActivity5()
```

Method Detail

```
public void onCreate(Bundle savedInstanceState)
```

APPENDIX 2: Java code

```
package cat.uvic.calculs;

import android.graphics.Rect;

/**
 * class responsible for looking for interesting region based on
histograms (horizontal and vertical)
 * @author ANNA
 */
public class FindRegion {

	private boolean margin;
	private int perV = 10;
	private int perH = 4;

	/**
	 * @param marg margin value
	 */
	public FindRegion(boolean marg) {
		super();
		this.margin = marg;
	}

	/**
	 * @param marg global margin value
	 * @param pV border value for vertical histogram
	 * @param pH border value for horizontal histogram
	 */
	public FindRegion(boolean marg, int pV, int pH) {
		super();
		this.perH = pH;
		this.perV = pV;
		this.margin = marg;
	}

	/*
	 * Rectangle: (x,y)|------------------| + Width, Height | | | |
| |
	 * |------------------|
	 */

	/**
	 * @param histH horizontal histogram
	 * @param histV vertical histogram
	 * @return rectangle of interesting region
	 */
	public Rect findRegionOfInteres(int[] histH, int[] histV) {
		int maxH = 0;
		int w = 0, h = 0;
		for (int i = 0; i < histH.length; i++) {
			maxH = Math.max(histH[i], maxH);
		}
```

```
            int maxV = 0;
            for (int i = 0; i < histV.length; i++) {
                  maxV = Math.max(histV[i], maxV);
            }

            int x = maxH;
            int y = maxV;

            for (int i = 0; i < histH.length; i++) {
                  if ((histH[i] > (maxH / perH))) {
                        y = Math.min(y, i);
                        w = Math.max(w, i);
                  }
            }
            for (int i = 0; i < histV.length; i++) {
                  if ((histV[i] > (maxV / perV))) {
                        x = Math.min(x, i);
                        h = Math.max(h, i);
                  }
            }
            if (this.margin) {
                  int marg = (int) (Math.max(histH.length,
histV.length) * 0.1);
                  // System.out.println("Margin:"+marg);
                  int xx = (x > marg ? x - marg : 0);
                  int yy = (y > marg ? y - marg : 0);
                  int hh = (h - xx + marg <= histV.length - xx ? h - xx
+ marg
                              : histV.length - xx);
                  int ww = (w - yy + marg <= histH.length - yy ? w - yy
+ marg
                              : histH.length - yy);
                  return new Rect(xx, yy, xx + hh, yy + ww);
            } else
                  return new Rect(x, y, h, w);
      }
}


package cat.uvic.calculs;

import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;

/**
 * dilate a Gray  thresholded image
 * <p>
 *
 * @author ANNA
 */
public class Gray8Dilate extends PipelineStage {
```

```
	int many;

	/**
	 * Creates a new instance of Gray8Dilate
	 *
	 * @param _many how many times Dilation should repeat
	 */
	public Gray8Dilate(int _many) {
		super();
		this.many = _many;
	}

	/** Creates a new instance of Gray8Dilate */
	public Gray8Dilate() {
		super();
		this.many = 0;
	}

	@Override
	public void push(Image img) throws Error {
		if (!(img instanceof Gray8Image)) {
			throw new IllegalArgumentException(img.toString()
					+ " should be a Gray8Image, but isn't");
		}
		// long start = System.currentTimeMillis();
		Gray8Image input = (Gray8Image) img;
		byte[] image = input.getData();
		int h = input.getHeight();
		int w = input.getWidth();
		int imgW = img.getWidth();
		for (int i = 0; i < h; i++) {
			for (int j = 0; j < w; j++) {
				if (image[i * imgW + j] == Byte.MAX_VALUE) {
					if (i > 0 && image[(i - 1) * imgW + j] ==
Byte.MIN_VALUE)
						image[(i - 1) * imgW + j] =
Byte.MAX_VALUE;
					if (j > 0 && image[i * imgW + j - 1] ==
Byte.MIN_VALUE)
						image[i * imgW + j - 1] =
Byte.MAX_VALUE;
				}
			}
		}
		// long stop = System.currentTimeMillis();
		// System.out.println("Dilate: "+(stop-start));
		super.setOutput(new Gray8Image(input.getWidth(),
input.getHeight(),
				image));
```

```
	}
}


package cat.uvic.calculs;

import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;

/**
 * erode a Gray thresholded image
 * @author ANNA
 *
 */
public class Gray8Erose extends PipelineStage {

	int many;

	/**
	 * @param _many how many times Erosion should repeat
	 */
	public Gray8Erose(int _many) {
		super();
		this.many = _many;
	}

	/** Creates a new instance of Gray8Erose */
	public Gray8Erose() {
		super();
		this.many = 0;
	}

	@Override
	public void push(Image img) throws Error {
		if (!(img instanceof Gray8Image)) {
			throw new IllegalArgumentException(img.toString()
					+ " should be a Gray8Image, but isn't");
		}
		// long start = System.currentTimeMillis();
		Gray8Image input = (Gray8Image) img;
		byte[] image = input.getData();
		int h = input.getHeight();
		int w = input.getWidth();
		int imgW = img.getWidth();
		for (int i = 0; i < h; i++) {
			for (int j = 0; j < w; j++) {
				if (image[i * imgW + j] == Byte.MIN_VALUE) {
					if (i > 0 && image[(i - 1) * imgW + j] ==
Byte.MAX_VALUE)
```

```
                                        image[(i - 1) * imgW + j] =
Byte.MIN_VALUE;
                                if (j > 0 && image[i * imgW + j - 1] ==
Byte.MAX_VALUE)
                                        image[i * imgW + j - 1] =
Byte.MIN_VALUE;
                        }
                }
        }

        // long stop = System.currentTimeMillis();
        // System.out.println("Erose: "+(stop-start));
        super.setOutput(new Gray8Image(input.getWidth(),
input.getHeight(),
                        image));

    }

}
```

```
package cat.uvic.calculs;

import jjil.core.Gray8Image;

/**
 * horizontal histogram for Gray image.
 * @author ANNA
 */
public class Gray8HistHorizontal {
    /**
     * Creates a new instance of Gray8HistHorizontal *
     */
    public Gray8HistHorizontal() {
        super();
    }

    /**
     * @param image
     * @return array of integer that represents sequence of
histogram values
     */
    public static int[] computeHistogram(Gray8Image image) {
        int[] result = new int[image.getWidth()];
        // long start = System.currentTimeMillis();
        int h = image.getHeight();
        int w = image.getWidth();

        for (int i = 0; i < w; i++) {
            result[i] = 0;
        }
        byte[] data = image.getData();

        for (int m = 0; m < w; m++) {
```

```
                  for (int j = 0; j < h; j++) {
                        result[m] = result[m] + data[j * w + m] -
Byte.MIN_VALUE;
                  }
            }
            // long stop = System.currentTimeMillis();
            // System.out.println("Gray8HistHorizontal: "+(stop-
start));
            return result;
      }

}
```

```
package cat.uvic.calculs;

import jjil.core.Gray8Image;

/**
 * vertical histogram for Gray image.
 * @author ANNA
 */
public class Gray8HistVertical {
      /**
       * @param image
       * @return array of integer that represents sequence of
histogram values
       */
      public static int[] computeHistogram(Gray8Image image) {
            int[] result = new int[image.getHeight()];
            int h = image.getHeight();
            int w = image.getWidth();

            for (int i = 0; i < h; i++) {
                  result[i] = 0;
            }
            byte[] data = image.getData();
            for (int m = 0; m < h; m++) {
                  for (int j = 0; j < w; j++) {
                        result[m] = result[m] + data[m * w + j] -
Byte.MIN_VALUE;
                  }
            }
            return result;
      }
}
```

```
package cat.uvic.calculs;

import android.graphics.Rect;
import jjil.algorithm.ErrorCodes;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
```

```
import jjil.core.PipelineStage;

/**
 * cut part of a Gray image
 * @author ANNA
 */
public class Gray8Subimage extends PipelineStage {
	Rect rect;

	/**
	 * Creates a new instance of Gray8Subimage.
	 *
	 * @param _rect
	 * @throws jjil.core.Error
	 *             if the target width or height is less than or
equal to zero.
	 */
	public Gray8Subimage(Rect _rect) throws jjil.core.Error {
		this.rect = _rect;
	}

	public void push(Image img) throws jjil.core.Error {
		if (!(img instanceof Gray8Image)) {
			throw new Error(Error.PACKAGE.ALGORITHM,
					ErrorCodes.IMAGE_NOT_RGBIMAGE,
img.toString(), null, null);
		}
		Gray8Image result = new Gray8Image(rect.width(),
rect.height());
		byte[] data = ((Gray8Image) img).getData();
		byte[] out = result.getData();
		for (int i = 0; i < rect.height(); i++) {
			System.arraycopy(data, (i + rect.top) *
img.getWidth() + rect.left,
					out, i * (rect.width()), rect.width());
		}
		super.setOutput(result);
	}

}


package cat.uvic.calculs;

import jjil.algorithm.ErrorCodes;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.Rect;

/**
 * rotate a Gray thresholded image
 *  @author ANNA
 */
public class GrayFastRotate extends PipelineStage {
```

```
	double theta;

	/**
	 * @param theta arc value
	 */
	public GrayFastRotate(double theta) {
		this.theta = theta;
	}

	@Override
	public void push(Image img) throws Error {
		if (!(img instanceof Gray8Image)) {
			throw new Error(Error.PACKAGE.ALGORITHM,
					ErrorCodes.IMAGE_NOT_GRAY8IMAGE,
img.toString(), null, null);
		}
		int x = 0;
		int y = 0;
		int x0 = img.getWidth() / 2;
		int y0 = img.getHeight() / 2;

		double sint = Math.sin(theta);
		double cost = Math.cos(theta);
		// Gray8Image result = new
		// Gray8Image((int)Math.sqrt(img.getWidth()*img.getWidth()
+img.getHeight()*img.getHeight())
		// ,
		// (int)Math.sqrt(img.getWidth()*img.getWidth()
+img.getHeight()*img.getHeight()));
		Gray8Image result = new Gray8Image((int)
Math.floor(img.getWidth()
				* Math.abs(cost) + img.getHeight() *
Math.abs(sint)),
				(int) Math.floor(img.getHeight() *
Math.abs(cost)
						+ img.getWidth() *
Math.abs(sint)));
		byte[] data = ((Gray8Image) img).getData();
		result.fill(new Rect(0, 0, result.getWidth(),
result.getHeight()),
				Byte.MIN_VALUE);
		byte[] out = result.getData();
		// int rotx=(result.getWidth()-img.getWidth())/2;
		// int roty=(result.getHeight()-img.getHeight())/2;
		int wi = img.getWidth();
		int hi = img.getHeight();
		int wr = result.getWidth();

		for (int i = 0; i < wi; i++) {
			for (int j = 0; j < hi; j++) {
				if (data[i + j * wi] == Byte.MAX_VALUE) {
					x = (int) (((cost * i) - (sint * j) + (x0
* (1 - cost) + y0
							* sint)));
					y = (int) (((sint * i) + (cost * j) + (y0
* (1 - cost) - x0
```

```
                                        * sint)));
                                if ((int) ((x) + (y) * wr) >= 0
                                        && (int) ((x) + (y) * wr) <
out.length)
                                    // out[(int)((x+rotx)+
(y+roty)*result.getWidth())]=data[i+j*wi];
                                    out[(int) ((x) + (y) * wr)] =
data[i + j * wi];
                        }
                    }
            }
            super.setOutput(result);
      }

}


package cat.uvic.calculs;

import jjil.algorithm.ErrorCodes;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.Rect;

/**
 * rotate a Gray image
* @author ANNA
 */
public class GrayRotate extends PipelineStage {
      double theta;

      /**
       * @param theta arc value
       * @throws Error
       */
      public GrayRotate(double theta) throws jjil.core.Error {
            this.theta = theta;
      }

      @Override
      public void push(Image img) throws Error {
            if (!(img instanceof Gray8Image)) {
                  throw new Error(Error.PACKAGE.ALGORITHM,
                              ErrorCodes.IMAGE_NOT_GRAY8IMAGE,
img.toString(), null, null);
            }
            int x = 0;
            int y = 0;
            int x0 = img.getWidth() / 2;
            int y0 = img.getHeight() / 2;
            if (theta > 2 * Math.PI)
                  theta -= 2 * Math.PI;
            Gray8Image result = (Gray8Image) img;
            double sint = Math.sin(theta);
```

```
            double cost = Math.cos(theta);
            byte[] data = ((Gray8Image) img).getData();

            if (theta == Math.PI) {
                  result = new Gray8Image(img.getWidth(),
img.getHeight());
                  byte[] out = result.getData();
                  for (int i = 0; i < result.getWidth(); i++) {
                        for (int j = 0; j < result.getHeight(); j++) {
                              x = result.getWidth() - i - 1;
                              y = result.getHeight() - j - 1;
                              out[(i) + (j) * result.getWidth()] =
data[x + y
                                          * img.getWidth()];
                        }
                  }
            } else {
                  result = new Gray8Image((int)
Math.floor(img.getWidth()
                              * Math.abs(cost) + img.getHeight() *
Math.abs(sint)),
                              (int) Math.floor(img.getHeight() *
Math.abs(cost)
                                          + img.getWidth() *
Math.abs(sint)));
                  result.fill(new Rect(0, 0, result.getWidth(),
result.getHeight()),
                              Byte.MIN_VALUE);
                  byte[] out = result.getData();

                  int wr = result.getWidth();
                  int hr = result.getHeight();
                  int wi = img.getWidth();
                  int hi = img.getHeight();
                  for (int i = 0; i < wr; i++) {
                        for (int j = 0; j < hr; j++) {

                              // (x,y,1) = (cos, sin ..) * (x', y', 1)
                              x = (int) (((cost * i) + (sint * j) + (-
x0 * (cost - 1) - y0
                                          * sint)));
                              y = (int) (((-sint * i) + (cost * j) + (-
y0 * (cost - 1) + x0
                                          * sint)));

                              if (x < wi && y < hi && x + y * wi <
data.length
                                          && x + y * wi >= 0 && x > 0)
                                    out[(int) ((i) + (j) * wr)] =
data[x + y * wi];

                        }
                  }
            }

            super.setOutput(result);
```

```
	}

}


package cat.uvic.calculs;

import java.text.DecimalFormat;

/**
 * represents part of a histogram, that is zero or more that zero.
Used in looking for number pattern.
  * @author ANNA
 */
public class HistRegion {
	/**
	 *
	 */
	public static String Tipus_ZERO = "0";
	/**
	 *
	 */
	public static String Tipus_ONE = "1";
	private String type;
	private int count;
	private double percent;
	private double percentLocal;

	/**
	 * @return local percent of region
	 */
	public double getPercentLocal() {
		return percentLocal;
	}

	/**
	 * @param percentLocal
	 */
	public void setPercentLocal(double percentLocal) {
		this.percentLocal = percentLocal;
	}

	/**
	 * @return percent of region in String (#,##) format
	 */
	public String getStrPercent() {
		DecimalFormat df = new DecimalFormat("#,##");
		return df.format(percent);
	}

	/**
	 * @return local percent of region in String (#,##) format
	 */
	public String getStrPercentLocal() {
		DecimalFormat df = new DecimalFormat("#,##");
		return df.format(percentLocal);
```

```
	}

	/**
	 * @return percent of region
	 */
	public double getPercent() {
		return percent;
	}

	/**
	 * @param percent
	 */
	public void setPercent(double percent) {
		this.percent = percent;
	}

	/**
	 * @return count of pixels of region
	 */
	public int getCount() {
		return count;
	}

	/**
	 * @param count
	 */
	public void setCount(int count) {
		this.count = count;
	}

	/**
	 * @return type of HistRegion (ONE or ZERO)
	 */
	public String getType() {
		return type;
	}

	/**
	 * @param type
	 */
	public void setType(String type) {
		this.type = type;
	}

	/**
	 * @param type
	 */
	public HistRegion(String type) {
		super();
		this.type = type;
	}

}


package cat.uvic.calculs;
```

```java
import java.util.ArrayList;

/**
 * all objects HistResion of one histogram together.
 * @author ANNA
 */
public class HistRegions {
	ArrayList<HistRegion> histRegions;
	private long total;
	private String result;
	private long width;

	/**
	 * @return width of regions
	 */
	public long getWidth() {
		return width;
	}

	/**
	 * @param width
	 */
	public void setWidth(long width) {
		this.width = width;
	}

	/**
	 * @return result
	 */
	public String getResult() {
		return result;
	}

	/**
	 * @param result
	 */
	public void setResult(String result) {
		this.result = result;
	}

	/**
	 * @return total
	 */
	public long getTotal() {
		return total;
	}

	/**
	 * @param total
	 */
	public void setTotal(long total) {
		this.total = total;
	}

	/**
```

```
	 * @param total
	 */
	public HistRegions(long total) {
		super();
		this.total = total;
	}

	/**
	 * @return array of regions
	 */
	public ArrayList<HistRegion> getHistRegions() {
		if (this.histRegions == null)
			this.histRegions = new ArrayList<HistRegion>();
		return histRegions;
	}

	/**
	 * @param histRegions
	 */
	public void setHistRegions(ArrayList<HistRegion> histRegions) {
		this.histRegions = histRegions;
	}

	/**
	 * @param lowQ
	 * @return 10 if it is 10 €, 5 if it is 5€, 0 when not found,
2050 if can be
	 *         20€ or 50€
	 */
	public int analiseNumber(boolean lowQ) {
		String pattern = "";
		int position = -1;
		int pos1 = -1;
		HistRegion elem;
		for (int i = 0; i < this.getHistRegions().size(); i++) {
			elem = this.getHistRegions().get(i);
			if (pattern.equals("")) {
				if (elem.getType().equals(HistRegion.Tipus_ONE)
						&& elem.getPercent() > 12) {
					pattern = "1";
					position = i;
				}
			} else if (pattern.equals("1")) {
				if
(elem.getType().equals(HistRegion.Tipus_ZERO)) {
					pattern = "10";
				} else {
					pattern = "";
					position = -1;
				}
			} else if (pattern.equals("10")) {
				pos1 = i;
				if
(elem.getType().equals(HistRegion.Tipus_ONE)) {
					pattern = "101";
				} else {
```

```
                                pattern = "";
                                position = -1;
                            }
                        }
                    }
                    if (pattern.equals("101")) {
                        int total =
this.getHistRegions().get(position).getCount()
                                + this.getHistRegions().get(position +
1).getCount()
                                + this.getHistRegions().get(position +
2).getCount();
                        this.getHistRegions()
                                .get(position)
                                .setPercentLocal(

        (this.getHistRegions().get(position).getCount() * 100)
                                                        / total);
                        this.getHistRegions()
                                .get(position + 1)
                                .setPercentLocal(

        (this.getHistRegions().get(position + 1).getCount() * 100)
                                                        / total);
                        this.getHistRegions()
                                .get(position + 2)
                                .setPercentLocal(

        (this.getHistRegions().get(position + 2).getCount() * 100)
                                                        / total);

                        double w1 = (double) this.width
                                / (double)
this.getHistRegions().get(position).getCount();
                        double w2 = (double) this.width
                                / (double)
this.getHistRegions().get(position + 1)
                                                .getCount();
                        double oneXone = (double)
this.getHistRegions().get(position)
                                .getCount()
                                / (double)
this.getHistRegions().get(position + 2)
                                                .getCount();
                        this.result = pattern + ";;WIDTH;" + this.width +
";;W1:;" + w1
                                + ";;W2:;" + w2 + ";;ONEXONE:;" +
oneXone;

                        if (!lowQ && (oneXone > 0.3 && oneXone < 0.7)) {
                            return 10;
                        } else if (lowQ && (oneXone > 0.3 && oneXone < 0.7))
{
                            return 10;
                        } else if (!lowQ && (oneXone > 1.5 && oneXone < 3)) {
                            return 5;
```

```
                } else if (lowQ && (oneXone > 1.0 && oneXone < 3)) {
                        return 5;
                } else if (w1 == 0 && w2 == 0)
                        return 11;
                else if ((w1 >= 0.7 && w1 <= 1) || (oneXone > 3)
                                || (oneXone > 0.7 && oneXone < 1.4))
                        return 2050;
                else
                        return 101;
            } else if (pos1 >= 0) {
                double w1 = (double) this.width
                                / (double)
this.getHistRegions().get(position).getCount();
                if (w1 >= 0.7 && w1 <= 1)
                        return 2050;
                else
                        return 0;
            } else
                return 0;
      }

      public String toString() {
            String result = "";
            for (int i = 0; i < this.getHistRegions().size(); i++) {
                result = result +
this.getHistRegions().get(i).getType() + ";"
                                + this.getHistRegions().get(i).getCount()
+ ";"
                                +
this.getHistRegions().get(i).getStrPercent() + ";;";
            }
            return result;
      }

      /**
       * @param lowQ
       * @return true if is 50 €
       */
      public boolean analise50(boolean lowQ) {
            if (this.getHistRegions().size() == 7) {
                if (this.getHistRegions().get(1).getPercent() > 15
                                && this.getHistRegions().get(1).getType()
                                        .equals(HistRegion.Tipus_ONE)
                                && this.getHistRegions().get(5).getType()
                                        .equals(HistRegion.Tipus_ONE)
                                &&
this.getHistRegions().get(5).getPercent() > 15
                                &&
this.getHistRegions().get(2).getPercent() < 15
                                &&
this.getHistRegions().get(4).getPercent() < 15
                                &&
this.getHistRegions().get(3).getPercent() < this
                                        .getHistRegions().get(1).getP
ercent()
                                &&
```

```
this.getHistRegions().get(3).getPercent() < this
                                                .getHistRegions().get(5).getP
ercent())
                        return true;
                  else
                        return false;
            } else
                  return false;
      }
}
```

```
package cat.uvic.calculs;

import android.graphics.Bitmap;

/**
 * Represents a linear line as detected by the hough transform. This
line is
 * represented by an angle theta and a radius from the centre.
 *
 * @author Olly Oechsle, University of Essex, Date: 13-Mar-2008
 * @version 1.0
 */
public class HoughLine {

      /**
       * theta parameter (arc)
       */
      public double theta;
      protected double r;

      /**
       * Initialises the hough line
       *
       * @param theta
       *            arc value
       * @param r
       *            distance value
       */
      public HoughLine(double theta, double r) {
            this.theta = theta;
            this.r = r;
      }

      /**
       * Draws the line on the image of your choice with the RGB
colour of your
       * choice.
       *
       * @param image
       * @param color
       */
      public void draw(Bitmap image, int color) {
```

```
            int height = image.getHeight();
            int width = image.getWidth();

            // During processing h_h is doubled so that -ve r values
            int houghHeight = (int) (Math.sqrt(2) * Math.max(height,
width)) / 2;

            // Find edge points and vote in array
            float centerX = width / 2;
            float centerY = height / 2;

            // Draw edges in output array
            double tsin = Math.sin(theta);
            double tcos = Math.cos(theta);

            if (theta < Math.PI * 0.25 || theta > Math.PI * 0.75) {
                  // Draw vertical-ish lines
                  for (int y = 0; y < height; y++) {
                        int x = (int) ((((r - houghHeight) - ((y -
centerY) * tsin)) / tcos) + centerX);
                        if (x < width && x >= 0) {
                              image.setPixel(x, y, color);
                        }
                  }
            }
      }
}
```

```
package cat.uvic.calculs;

import java.util.Vector;
import jjil.core.Gray8Image;

/**
 * <p/>
 * Note: This class is based on original code from:<br />
 * <a href=
 *
"http://vase.essex.ac.uk/software/HoughTransform/HoughTransform.java.h
tml"
 *
>http://vase.essex.ac.uk/software/HoughTransform/HoughTransform.java.h
tml</a>
 * </p>
 * <p/>
 * If you represent a line as:<br />
 * x cos(theta) + y sin (theta) = r
 * </p>
 */

public class HoughTransform {
```

```
	// The size of the neighbourhood in which to search for other
local maxima
	final int neighbourhoodSize = 40;

	// How many discrete values of theta shall we check?
	final int maxTheta = 180;

	// Using maxTheta, work out the step
	final double thetaStep = Math.PI / maxTheta;

	// the width and height of the image
	protected int width, height;

	// the hough array
	protected int[][] houghArray;

	// the coordinates of the centre of the image
	protected float centerX, centerY;

	// the height of the hough array
	protected int houghHeight;

	// double the hough height (allows for negative numbers)
	protected int doubleHeight;

	// the number of points that have been added
	protected int numPoints;

	// cache of values of sin and cos for different theta values.
Has a
	// significant performance improvement.
	private double[] sinCache;
	private double[] cosCache;

	/**
	 * Initialises the hough transform. The dimensions of the input
image are
	 * needed in order to initialise the hough array.
	 *
	 * @param width
	 *            The width of the input image
	 * @param height
	 *            The height of the input image
	 */
	public HoughTransform(int width, int height) {

		this.width = width;
		this.height = height;

		initialise();

	}

	/**
	 * Initialises the hough array. Called by the constructor so you
don't need
```

```
	 * to call it yourself, however you can use it to reset the
transform if you
	 * want to plug in another image (although that image must have
the same
	 * width and height)
	 */
	public void initialise() {

		// Calculate the maximum height the hough array needs to
have
		houghHeight = (int) (Math.sqrt(2) * Math.max(height,
width)) / 2;

		// Double the height of the hough array to cope with
negative r values
		doubleHeight = 2 * houghHeight;

		// Create the hough array
		houghArray = new int[maxTheta][doubleHeight];

		// Find edge points and vote in array
		centerX = width / 2;
		centerY = height / 2;

		// Count how many points there are
		numPoints = 0;

		// cache the values of sin and cos for faster processing
		sinCache = new double[maxTheta];
		cosCache = sinCache.clone();
		for (int t = 0; t < maxTheta; t++) {
			double realTheta = t * thetaStep;
			sinCache[t] = Math.sin(realTheta);
			cosCache[t] = Math.cos(realTheta);
		}
	}

	/**
	 * Adds points from an image. The image is assumed to be
greyscale black and
	 * white, so all pixels that are not black are counted as edges.
The image
	 * should have the same dimensions as the one passed to the
constructor.
	 *
	 * @param image
	 */
	public void addPoints(Gray8Image image) {
		int w = image.getWidth();
		int h = image.getHeight();
		// Now find edge points and update the hough array
		for (int x = 0; x < w; x++) {
			for (int y = 0; y < h; y++) {
				// Find non-black pixels
				if (((image.getData())[y * w + x]) !=
Byte.MIN_VALUE) {
```

```
                                addPoint(x, y);
                        }
                }
        }
    }

    /**
     * Adds a single point to the hough transform. You can use this
method
     * directly if your data isn't represented as a buffered image.
     *
     * @param x
     * @param y
     */
    public void addPoint(int x, int y) {

        // Go through each value of theta
        for (int t = 0; t < maxTheta; t++) {

            // Work out the r values for each theta step
            int r = (int) (((x - centerX) * cosCache[t]) + ((y -
centerY) * sinCache[t]));

            // this copes with negative values of r
            r += houghHeight;

            if (r < 0 || r >= doubleHeight)
                continue;

            // Increment the hough array
            houghArray[t][r]++;

        }

        numPoints++;
    }

    /**
     * Once points have been added in some way this method extracts
the lines
     * and returns them as a Vector of HoughLine objects, which can
be used to
     * draw on the
     *
     * @param threshold
     *            The percentage threshold above which lines are
determined from
     *            the hough array
     * @return Result vector of Hough lines
     */
    public Vector<HoughLine> getLines(int threshold) {

        // Initialise the vector of lines that we'll return
        Vector<HoughLine> lines = new Vector<HoughLine>(20);

        // Only proceed if the hough array is not empty
```

```
            if (numPoints == 0)
                  return lines;

            // Search for local peaks above threshold to draw
            for (int t = 0; t < maxTheta; t++) {
                  loop: for (int r = neighbourhoodSize; r <
doubleHeight
                              - neighbourhoodSize; r++) {

                        // Only consider points above threshold
                        if (houghArray[t][r] > threshold) {

                              int peak = houghArray[t][r];

                              // Check that this peak is indeed the
local maxima
                              for (int dx = -neighbourhoodSize; dx <=
neighbourhoodSize; dx++) {
                                    for (int dy = -neighbourhoodSize;
dy <= neighbourhoodSize; dy++) {
                                          int dt = t + dx;
                                          int dr = r + dy;
                                          if (dt < 0)
                                                dt = dt + maxTheta;
                                          else if (dt >= maxTheta)
                                                dt = dt - maxTheta;
                                          if (houghArray[dt][dr] >
peak) {
                                                // found a bigger point
nearby, skip
                                                continue loop;
                                          }
                                    }
                              }

                              // calculate the true value of theta
                              double theta = t * thetaStep;

                              if (theta != 0)
                                    // add the line to the vector
                                    lines.add(new HoughLine(theta, r));

                        }
                  }
            }

            return lines;
      }

      /**
       * @return highest value in the hough array
       */
      public int getHighestValue() {
            int max = 0;
            for (int t = 0; t < maxTheta; t++) {
                  for (int r = 0; r < doubleHeight; r++) {
```

```
                        if (houghArray[t][r] > max) {
                              max = houghArray[t][r];
                        }
                  }
            }
            return max;
      }
}
```

```
package cat.uvic.calculs;

import android.graphics.Color;

import jjil.core.Gray8Image;
import jjil.core.RgbImage;
import jjil.core.RgbVal;

/**
 * colour histogram for an image in HSV colorspace.
 *  @author ANNA
 */
public class HsvHistogram {

      /**
       *
       */
      public static String H_HIST = "H";
      /**
       *
       */
      public static String V_HIST = "V";
      /**
       *
       */
      public static String S_HIST = "S";

      /**
       *
       */
      public int[] colors = { Color.rgb(255, 0, 0), Color.rgb(255,
127, 0),
                  Color.rgb(255, 255, 0), Color.rgb(127, 255, 0),
                  Color.rgb(0, 255, 0), Color.rgb(0, 255, 127),
                  Color.rgb(0, 255, 255), Color.rgb(0, 127, 255),
                  Color.rgb(0, 0, 255), Color.rgb(127, 0, 255),
                  Color.rgb(255, 0, 255), Color.rgb(255, 0, 127) };
      private String tipus;
      private String log = "";

      /**
       * @return log (for debug)
       */
      public String getLog() {
            return log;
```

```
}

/**
 * @param line
 */
public void setLog(String line) {
	this.log = line;
}

/**
 * @return tipus
 */
public String getTipus() {
	return tipus;
}

/**
 * @param tipus
 */
public void setTipus(String tipus) {
	this.tipus = tipus;
}

/**
 * @param tipus
 */
public HsvHistogram(String tipus) {
	super();
	this.tipus = tipus;
}

/**
 *
 */
public HsvHistogram() {
	super();
	this.tipus = "H";
}

/**
 * @param image
 * @return histogram
 */
public int[] computeHistogram(RgbImage image) {
	int[] result = new int[12];
	int[] hsvData = ((RgbImage) image).getData();
	int step = (2 * Byte.MAX_VALUE + 12) / 12;
	int total = 0;
	int max = 0;
	this.log = "";
	int min = 256;
	for (int i = 0; i < 12; i++) {
		result[i] = 0;
	}
	int pp = 0;
	int h = image.getHeight();
```

```
            int w = image.getWidth();
            for (int m = 0; m < h * w; m++) {
                  if (this.tipus.equals("H")) {
                        pp = RgbVal.getR(hsvData[m]) - Byte.MIN_VALUE;
                  } else if (this.tipus.equals("S")) {
                        pp = RgbVal.getG(hsvData[m]) - Byte.MIN_VALUE;
                  } else if (this.tipus.equals("V")) {
                        pp = RgbVal.getB(hsvData[m]) - Byte.MIN_VALUE;
                  }
                  result[pp / step] = result[pp / step] + pp;
                  if (pp > max)
                        max = pp;
                  if (pp < min)
                        min = pp;
                  total = total + pp;
            }

            for (int i = 0; i < 12; i++) {
                  result[i] = result[i] * 100 / total;
                  this.log = this.log + result[i] + ";";
            }
            return result;
      }

      /**
       * @param image
       * @param gray
       * @return histogram
       * @throws ImageException
       */
      public int[] computeHistogram(RgbImage image, Gray8Image gray)
                  throws ImageException {
            if (gray.getWidth() * gray.getHeight() < image.getWidth()
                        * image.getHeight())
                  throw new ImageException("Wrong map");
            int[] result = new int[12];
            byte[] map = gray.getData();
            int[] hsvData = ((RgbImage) image).getData();
            int step = (2 * Byte.MAX_VALUE + 12) / 12;
            int total = 0;
            int max = 0;
            this.log = "";
            int min = 256;
            for (int i = 0; i < 12; i++) {
                  result[i] = 0;
            }
            int pp = 0;
            int h = image.getHeight();
            int w = image.getWidth();
            for (int m = 0; m < h * w; m++) {
                  if (this.tipus.equals("H") && map[m] ==
Byte.MIN_VALUE) {
                        pp = RgbVal.getR(hsvData[m]) - Byte.MIN_VALUE;
                  } else if (this.tipus.equals("S") && map[m] ==
Byte.MIN_VALUE) {
                        pp = RgbVal.getG(hsvData[m]) - Byte.MIN_VALUE;
```

```
                } else if (this.tipus.equals("V") && map[m] ==
Byte.MIN_VALUE) {
                        pp = RgbVal.getB(hsvData[m]) - Byte.MIN_VALUE;
                }
                result[pp / step] = result[pp / step] + pp;
                if (pp > max)
                        max = pp;
                if (pp < min)
                        min = pp;
                total = total + pp;
            }
            for (int i = 0; i < 12; i++) {
                if (total > 0) {
                        result[i] = result[i] * 100 / total;
                        this.log = this.log + result[i] + ";";
                }
            }
            return result;
      }

      /**
       * @param hist
       * @return sum of histogram values
       */
      public long sumHistogram(int[] hist) {
            long sum = 0;
            for (int i = 0; i < hist.length; i++) {
                sum = sum + hist[i];
            }
            return sum;
      }
}


package cat.uvic.calculs;

/**
 * @author jppr...@gmail.com (from project textdetection -
 *         http://code.google.com/p/textdetection/source/checkout)
 *
 */
public class OtsuThresholder {
      private int histData[];
      private int maxLevelValue;
      private int threshold;

      /**
       *
       */
      public OtsuThresholder() {
            histData = new int[256];
      }

      /**
       * @return histData
       */
```

```
	public int[] getHistData() {
		return histData;
	}

	/**
	 * @return maxLevelValue
	 */
	public int getMaxLevelValue() {
		return maxLevelValue;
	}

	/**
	 * @return threshold
	 */
	public int getThreshold() {
		return threshold;
	}

	/**
	 * @param srcData
	 * @param monoData
	 * @return threshold
	 */
	public int doThreshold(byte[] srcData, byte[] monoData) {
		int ptr;

		// Clear histogram data
		// Set all values to zero
		ptr = 0;
		while (ptr < histData.length)
			histData[ptr++] = 0;

		// Calculate histogram and find the level with the max
value
		// Note: the max level value isn't required by the Otsu
method
		ptr = 0;
		maxLevelValue = 0;
		int l = srcData.length;
		while (ptr < l) {
			int h = 0xFF & srcData[ptr];
			histData[h]++;
			if (histData[h] > maxLevelValue)
				maxLevelValue = histData[h];
			ptr++;
		}

		// Total number of pixels
		int total = srcData.length;

		float sum = 0;
		for (int t = 0; t < 256; t++)
			sum += t * histData[t];

		float sumB = 0;
		int wB = 0;
```

```
            int wF = 0;

            float varMax = 0;
            threshold = 0;

            for (int t = 0; t < 256; t++) {
                  wB += histData[t]; // Weight Background
                  if (wB == 0)
                        continue;

                  wF = total - wB; // Weight Foreground
                  if (wF == 0)
                        break;

                  sumB += (float) (t * histData[t]);

                  float mB = sumB / wB; // Mean Background
                  float mF = (sum - sumB) / wF; // Mean Foreground

                  // Calculate Between Class Variance
                  float varBetween = (float) wB * (float) wF * (mB -
mF) * (mB - mF);

                  // Check if new maximum found
                  if (varBetween > varMax) {
                        varMax = varBetween;
                        threshold = t;
                  }
            }

            // Apply threshold to create binary image
            if (monoData != null) {
                  ptr = 0;
                  while (ptr < srcData.length) {
                        monoData[ptr] = ((0xFF & srcData[ptr]) >=
threshold) ? (byte) 255
                                    : 0;
                        ptr++;
                  }
            }

            return threshold;
      }
}
```

```
package cat.uvic.calculs;

import jjil.algorithm.ErrorCodes;
import jjil.core.Error;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.RgbImage;
import jjil.core.RgbVal;
```

```
/**
 * pass RGB image to YcbCr colorspace
 *  @author ANNA
 */
public class Rgb2YCbCr extends PipelineStage {

      public void push(Image imageInput) throws Error {
            if (!(imageInput instanceof RgbImage)) {
                  throw new Error(Error.PACKAGE.ALGORITHM,
                              ErrorCodes.IMAGE_NOT_RGBIMAGE,
imageInput.toString(), null,
                              null);
            }
            RgbImage rgbInput = (RgbImage) imageInput;
            RgbImage result = new RgbImage(rgbInput.getWidth(),
                        rgbInput.getHeight());
            int[] rgbData = rgbInput.getData();
            int[] out = result.getData();
            int nR, nG, nB;
            double Y, Cb, Cr;
            for (int i = 0; i < rgbInput.getWidth() *
rgbInput.getHeight(); i++) {
                  nR = RgbVal.getR(rgbData[i]) - Byte.MIN_VALUE;
                  nG = RgbVal.getG(rgbData[i]) - Byte.MIN_VALUE;
                  nB = RgbVal.getB(rgbData[i]) - Byte.MIN_VALUE;

                  // Y = 0.299*R + 0.587*G + 0.114*B
                  // Cb = (B-Y)*0.564 + 0.5
                  // Cr = (R-Y)*0.713 + 0.5

                  Y = 0.299 * nR + 0.587 * nG + 0.114 * (nB);
                  Cb = (nB - Y) * 0.564 + 0.5;
                  Cr = (nR - Y) * 0.713 + 0.5;

                  out[i] = RgbVal.toRgb((byte) (Y + Byte.MIN_VALUE),
                              (byte) (Cb + Byte.MIN_VALUE), (byte) (Cr
+ Byte.MIN_VALUE));
            }
            super.setOutput(result);
      }
}
```

```
package cat.uvic.calculs;

import jjil.algorithm.Gray3Bands2Rgb;
import jjil.algorithm.Gray8HistEq;
import jjil.algorithm.RgbSelectGray;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.Sequence;

/**
```

```
 * equalize histograms of RGB image
 * @author ANNA
 */
public class RgbHistEqualize extends PipelineStage {

	/**
	 *
	 */
	public RgbHistEqualize() {
		super();
	}

	@Override
	public void push(Image img) throws Error {
		Sequence seqR = new Sequence(new
RgbSelectGray(RgbSelectGray.RED));
		seqR.add(new Gray8HistEq());
		Sequence seqG = new Sequence(new
RgbSelectGray(RgbSelectGray.GREEN));
		seqG.add(new Gray8HistEq());
		Sequence seqB = new Sequence(new
RgbSelectGray(RgbSelectGray.BLUE));
		seqB.add(new Gray8HistEq());

		seqR.push(img);
		seqG.push(img);
		seqB.push(img);

		super.setOutput(Gray3Bands2Rgb.push((Gray8Image)
seqR.getFront(),
				(Gray8Image) seqG.getFront(), (Gray8Image)
seqB.getFront()));
	}

}


package cat.uvic.calculs;

import android.graphics.Rect;
import jjil.algorithm.ErrorCodes;
import jjil.core.Error;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.RgbImage;

/**
 * cut defined part of RGB image
 * @author ANNA
 */
public class RgbImageSubImage extends PipelineStage {
	Rect rect;

	/**
	 * @param _rect
	 * @throws Error
```

```
         */
        public RgbImageSubImage(Rect _rect) throws jjil.core.Error {
              this.rect = _rect;
        }

        /**
         * Reduces an RgbImage by a factor horizontally and vertically
through
         * averaging. The reduction factor must be an even multiple of
the image
         * size.
         *
         * @param img
         *            the input image.
         * @throws jjil.core.Error
         *             if the input image is not gray, or the reduction
factor does
         *             not evenly divide the image size.
         */
        public void push(Image img) throws jjil.core.Error {
              if (!(img instanceof RgbImage)) {
                    throw new Error(Error.PACKAGE.ALGORITHM,
                                ErrorCodes.IMAGE_NOT_RGBIMAGE,
img.toString(), null, null);
              }
              RgbImage result = new RgbImage(rect.width(),
rect.height());
              int[] rgbData = ((RgbImage) img).getData();
              int[] out = result.getData();
              // System.out.println("RGBImageSubImageRect: L:
"+rect.left+" R: "+rect.right+" T: "+rect.top+" B: "+rect.bottom+"
Width: "+rect.width()+" Height: "+rect.height());
              for (int i = 0; i < rect.height(); i++) {
                    System.arraycopy(rgbData, (i + rect.top) *
img.getWidth()
                                + rect.left, out, i * (rect.width()),
rect.width());
              }
              super.setOutput(result);
        }
}


package cat.uvic.calculs;

import jjil.algorithm.ErrorCodes;
import jjil.algorithm.Gray3Bands2Rgb;
import jjil.algorithm.RgbSelectGray;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.PipelineStage;
import jjil.core.RgbImage;
import jjil.core.Sequence;

/**
```

```
 * rotate RGB image
 * @author ANNA
 */
public class RgbRotate extends PipelineStage {
	double theta;

	/**
	 * @param theta
	 * @throws Error
	 */
	public RgbRotate(double theta) throws jjil.core.Error {
		this.theta = theta;
	}

	@Override
	public void push(Image img) throws Error {
		if (!(img instanceof RgbImage)) {
			throw new Error(Error.PACKAGE.ALGORITHM,
					ErrorCodes.IMAGE_NOT_RGBIMAGE,
img.toString(), null, null);
		}
		Sequence seqR = new Sequence(new
RgbSelectGray(RgbSelectGray.RED));
		seqR.add(new GrayRotate(this.theta));
		Sequence seqG = new Sequence(new
RgbSelectGray(RgbSelectGray.GREEN));
		seqG.add(new GrayRotate(this.theta));
		Sequence seqB = new Sequence(new
RgbSelectGray(RgbSelectGray.BLUE));
		seqB.add(new GrayRotate(this.theta));

		seqR.push(img);
		seqG.push(img);
		seqB.push(img);
		super.setOutput(Gray3Bands2Rgb.push((Gray8Image)
seqR.getFront(),
				(Gray8Image) seqG.getFront(), (Gray8Image)
seqB.getFront()));
	}

}


package cat.uvic.calculs;

import java.io.File;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Vector;

import cat.uvic.android.ResultBitmap;
import cat.uvic.android.RgbImageAndroid;

import android.graphics.Bitmap;
import android.graphics.Rect;
```

```
import jjil.algorithm.*;
import jjil.core.Error;
import jjil.core.Gray8Image;
import jjil.core.Image;
import jjil.core.RgbImage;
import jjil.core.RgbVal;
import jjil.core.Sequence;

/**
 * the main class that is putting all operations together. Is
responsible for localisation, checking position, rotating and
recognition of a banknote. Class implements Singleton pattern. It
means that there is just one instance of the class in the program.
* @author ANNA
 */
public class Transformation {
	private Image img; // source image
	private String path;
	File result;
	Rect region;
	Vector<HoughLine> lines;
	private Gray8Image grayH;
	private Gray8Image grayV;
	private int threshold;
	private int recognition;
	private Image currentImg;
	private boolean wrong = false;

	/**
	 * @return true if the image is corrupted, false if everything
is correct
	 */
	public boolean isWrong() {
		return wrong;
	}

	/**
	 * @param wrong
	 */
	public void setWrong(boolean wrong) {
		this.wrong = wrong;
	}

	private boolean lowQ = false;
	/**
	*
	*/
	public String error;
	HoughLine arc = null;
	DecimalFormat df = new DecimalFormat("##,##");
	private ResultBitmap image;

	/**
	 * @return Bitmap of current processed image
	 * @throws Error
	 */
```

```
public Bitmap getCurrentImg() throws Error {
	if (this.currentImg instanceof Gray8Image) {
		Gray8Rgb g2r = new Gray8Rgb();
		g2r.push(this.currentImg);
		RgbImage rgb = (RgbImage) g2r.getFront();
		return RgbImageAndroid.toBitmap((RgbImage) rgb);
	} else
		return RgbImageAndroid.toBitmap((RgbImage) img);
}

/**
 * @param currentImg
 */
public void setCurrentImg(Image currentImg) {
	this.currentImg = currentImg;
}

/**
 *
 */
public static String EUR_5 = "5";
/**
 *
 */
public static String EUR_10 = "10";
/**
 *
 */
public static String EUR_20 = "20";
/**
 *
 */
public static String EUR_50 = "50";

/**
 * @return number of recognition (5,10,20,50,0 or 2050)
 */
public int getRecognition() {
	return recognition;
}

/**
 * @param recognition
 */
public void setRecognition(int recognition) {
	this.recognition = recognition;
}

String report;

/**
 * @return vector of HoughLines (if any)
 */
public Vector<HoughLine> getLines() {
	if (this.lines == null)
		this.lines = new Vector<HoughLine>();
```

```
		return lines;
	}

	/**
	 * @param lines
	 */
	public void setLines(Vector<HoughLine> lines) {
		this.lines = lines;
	}

	/**
	 * @return rectangle of current processed region
	 */
	public Rect getRegion() {
		return region;
	}

	/**
	 * @param region
	 */
	public void setRegion(Rect region) {
		this.region = region;
	}

	private Transformation() {
	}

	private Transformation(String _img) throws IOException {
		this.path = "resources/IMG_2.jpg";
		this.result = new File(this.path);
	}

	private Transformation(Bitmap _img) throws IOException {
		this.path = "resources/IMG_2.jpg";
		this.result = new File(this.path);
		this.image = new ResultBitmap(_img, region);
		RgbImageAndroid.toRgbImage(_img);
		this.img = RgbImageAndroid.toRgbImage(_img);
	}

	private static Transformation instance;

	/**
	 * @param _img
	 * @return instance of Transformation
	 * @throws IOException
	 */
	public static Transformation getInstance(Bitmap _img) throws
IOException {
		if (instance == null)
			instance = new Transformation(_img);
		return instance;
	}

	/**
	 * destroy the instance of Transformation class
```

```
	 */
	public static void destroy() {
		instance = null;
	}

	/**
	 * @return Bitmap of final result of transformation
	 * @throws Error
	 * @throws Exception
	 */
	public ResultBitmap transform() throws Error, Exception {
		this.region = this.findBill(this.img, true);
		this.cutImage();
		this.region = this.findBill(this.img, false);

		this.checkPosition();

		if (wrong)
			return image;
		if (!this.recognise50()) {
			analiseNumber();
			// if (wrong) return image;

			if (this.recognition != 5 && this.recognition != 10
					&& this.recognition != 1010 &&
this.recognition != 101) {
				this.recognition = this.analiseColor();
			}
			// this.report = this.report+";;Error:;"+this.error;
		} else
			this.recognition = 50;
		if (this.recognition == 2050)
			this.recognition = 20;
		if (this.recognition == 1010)
			this.recognition = 10;
		if (this.recognition != 5 && this.recognition != 10
				&& this.recognition != 20 && this.recognition !
= 50)
			this.recognition = 0;

		Gray8Rgb g2r = new Gray8Rgb();
		g2r.push(grayH);
		Bitmap result = RgbImageAndroid.toBitmap((RgbImage) img);
		return new ResultBitmap(result, region);
	}

	/**
	 * @return img
	 */
	public Image getImg() {
		return img;
	}

	/**
	 * @param img
	 */
```

```
	public void setImg(Image img) {
		this.img = img;
	}

	/**
	 * @throws Error
	 * @throws Exception
	 */
	public void checkPosition() throws Error, Exception {
		long start = System.currentTimeMillis();
		double divRegion = (double) (this.region.width())
				/ (double) (this.region.height());
		System.out.println("checkPositionDiv: " + divRegion);
		if (divRegion < 1.9 || divRegion > 2.1) {
			this.rotate(this.grayH);

			if (this.arc != null) {
				Sequence seq = new Sequence();
				if (this.arc.theta > 0.1) {
					seq.add(new RgbRotate(Math.PI -
this.arc.theta));
					seq.push(img);
					this.img = (RgbImage) seq.getFront();
				}
			} // else
				// throw new ImageException(
				// "Can't find the banknote.
(HoughTransform)");
		}
		this.cutImage();

		divRegion = (double) (this.grayH.getWidth())
				/ (double) (this.grayH.getHeight());
		if ((divRegion < 1.5 || divRegion > 2.4) && (this.arc !=
null)) {
			this.error = "Can't find the banknote.
(HoughTransform)";
			this.recognition = 0;
			this.wrong = true;
			return;
		}
		Sequence seq = new Sequence();
		this.region = new Rect(0, this.grayH.getHeight() / 3,
				(this.grayH.getWidth()), (2 *
this.grayH.getHeight()) / 3);
		seq.add(new Gray8Subimage(new Rect(0, this.img.getHeight()
/ 3,
				(this.img.getWidth()), (2 *
this.img.getHeight()) / 3)));
		seq.push(this.grayH);
		Gray8Image kk = (Gray8Image) seq.getFront();

		byte[] data = kk.getData();
		long sumWhite = 0;
		long sumBlack = 0;
		long sum1 = 0;
```

```
            long sum2 = 0;

            for (int i = 0; i < kk.getWidth(); i++)
                  for (int j = 0; j < kk.getHeight(); j++) {
                        if (data[j * kk.getWidth() + i] ==
Byte.MIN_VALUE)
                              sumBlack++;
                        else if (data[j * kk.getWidth() + i] ==
Byte.MAX_VALUE) {
                              sumWhite++;
                              if (i < kk.getWidth() / 2)
                                    sum1++;
                              else
                                    sum2++;
                        }

                  }

            if (sum1 > sum2) {
                  seq = new Sequence();
                  seq.add(new RgbRotate(Math.PI));
                  seq.push(img);
                  img = seq.getFront();
                  seq = new Sequence();
                  seq.add(new GrayRotate(Math.PI));
                  seq.push(this.grayH);
                  this.grayH = (Gray8Image) seq.getFront();
            }
            double res = (double) sumBlack / (double) sumWhite;
            if ((res > 25)) {
                  this.lowQ = true;
                  this.error = "Low quality";
                  this.recognition = 0;
                  this.findBill(this.img, true);
            }
            long stop = System.currentTimeMillis();
            this.currentImg = this.img;
            System.out.println("checkPosition: " + (stop - start));
      }

      private HoughLine rotate(Gray8Image im) throws Exception, Error
{
            long start = System.currentTimeMillis();
            Sequence seq = new Sequence();
            FindRegion ff = new FindRegion(false, 20, 10);
            double divRegion = (double) (this.region.width())
                        / (double) (this.region.height());
            double min = Math.abs(divRegion - 2);
            int idx = 0;
            int[] hist2;
            int[] hist;
            Gray8Image copyH = (Gray8Image) this.grayH.clone();
            Gray8Image copyV = (Gray8Image) this.grayV.clone();
            Gray8Image maxH = copyH;
            Gray8Image maxV = copyV;
```

```
            if (this.arc != null && this.arc.theta > 0.1) {
                  if (divRegion < 1.9 || divRegion > 2.1) {
                        seq = new Sequence();
                        System.out.println("R: " + this.arc.r + "
Theta: "
                                    + this.arc.theta);
                        seq.add(new GrayFastRotate(Math.PI -
this.arc.theta));
                        seq.push(this.grayH);
                        copyH = (Gray8Image) seq.getFront();
                        seq.push(this.grayV);
                        copyV = (Gray8Image) seq.getFront();
                        hist2 =
Gray8HistHorizontal.computeHistogram(copyV);
                        hist =
Gray8HistVertical.computeHistogram(copyH);
                        this.region = ff.findRegionOfInteres(hist,
hist2);
                        divRegion = (double) (this.region.width())
                                    / (double) (this.region.height());
                        if (min > Math.abs(divRegion - 2)) {
                              maxH = copyH;
                              maxV = copyV;
                              min = Math.abs(divRegion - 2);
                              idx = 0;
                        }
                  }
            }

            if (divRegion < 1.9 || divRegion > 2.1) {
                  HoughTransform h = new HoughTransform(im.getWidth(),
im.getHeight());
                  h.addPoints(grayH);
                  Vector<HoughLine> tmp = new Vector<HoughLine>();
                  tmp = h.getLines(50);
                  this.lines = tmp;
                  long stop = System.currentTimeMillis();
                  System.out.println("HoughLine: " + (stop - start));
                  for (int i = 0; i < tmp.size(); i++) {
                        if (tmp.get(i).theta > 0.1
                                    && (divRegion <= 1.9 || divRegion
>= 2.1)) {
                              seq = new Sequence();
                              System.out.println("R: " + tmp.get(i).r +
"  Theta: "
                                          + tmp.get(i).theta + " Div:"
+ divRegion);
                              start = System.currentTimeMillis();
                              seq.add(new GrayFastRotate(Math.PI -
tmp.get(i).theta));
                              seq.push(this.grayH);
                              copyH = (Gray8Image) seq.getFront();
                              seq.push(this.grayV);
                              copyV = (Gray8Image) seq.getFront();
                              stop = System.currentTimeMillis();
```

```
                              start = System.currentTimeMillis();
                              hist2 =
Gray8HistHorizontal.computeHistogram(copyV);
                              hist =
Gray8HistVertical.computeHistogram(copyH);
                              this.region =
ff.findRegionOfInteres(hist, hist2);
                              stop = System.currentTimeMillis();
                              divRegion = (double)
(this.region.width())
                                          / (double)
(this.region.height());

                              if (min > Math.abs(divRegion - 2)) {
                                    maxH = copyH;
                                    maxV = copyV;
                                    min = Math.abs(divRegion - 2);
                                    idx = i;
                              }
                        }
                  }
                  hist2 = Gray8HistHorizontal.computeHistogram(maxV);
                  hist = Gray8HistVertical.computeHistogram(maxH);

                  this.region = ff.findRegionOfInteres(hist, hist2);
                  divRegion = (double) (this.region.width())
                              / (double) (this.region.height());
                  if (divRegion >= 1.9 || divRegion <= 2.1)
                        this.arc = tmp.get(idx);
                  else
                        this.arc = null;
                  stop = System.currentTimeMillis();
            }
            this.grayH = maxH;
            this.grayV = maxV;
            return this.arc;
      }

      /**
       * @param img
       * @param margin
       * @return rectangle where a bill is
       * @throws Error
       */
      public Rect findBill(Image img, boolean margin) throws Error {
            // long start = System.currentTimeMillis();
            FindRegion ff = new FindRegion(margin);
            // long start = System.currentTimeMillis();
            Sequence seq = new Sequence();
            if (img instanceof RgbImage) {
                  seq.add(new RgbAvgGray());
                  seq.push(img);
                  this.grayH = (Gray8Image) seq.getFront();
            }
            // aqui l'algoritme Otsu
            OtsuThresholder thresholder = new OtsuThresholder();
```

```
            threshold = thresholder.doThreshold(((Gray8Image)
grayH).getData(),
                        null);
            seq = new Sequence();
            if (lowQ)
                  seq.add(new Gray8HistEq());
            seq.add(new Gray8CannyVert(50));
            seq.add(new Gray8Threshold(threshold / 10, false));
            seq.add(new Gray8Erose());
            seq.add(new Gray8Dilate());
            seq.push(grayH);
            this.grayV = (Gray8Image) seq.getFront();
            seq = new Sequence();
            if (lowQ)
                  seq.add(new Gray8HistEq());
            seq.add(new Gray8CannyHoriz(50));
            seq.add(new Gray8Threshold(threshold / 10, false));
            seq.add(new Gray8Erose());
            seq.add(new Gray8Dilate());
            seq.push(grayH);
            this.grayH = (Gray8Image) seq.getFront();

            int[] hist2 = Gray8HistHorizontal.computeHistogram(grayV);
            int[] hist = Gray8HistVertical.computeHistogram(grayH);

            this.region = ff.findRegionOfInteres(hist, hist2);
            this.currentImg = this.grayH;
            return region;
      }

      /**
       * @return recognized denomination or 0
       * @throws Error
       * @throws ImageException
       */
      public int analiseColor() throws Error, ImageException {
            // long start = System.currentTimeMillis();
            Sequence seq = new Sequence();
            long red = 0;
            long blue = 0;
            RgbImage hh = (RgbImage) this.img.clone();

            this.region = new Rect(5 * img.getWidth() / 7,
img.getHeight() / 15,
                        (img.getWidth() / 8) + (5 * img.getWidth() /
7),
                        (img.getHeight() / 5) + (img.getHeight() /
15));
            seq.add(new RgbImageSubImage(this.region));
            seq.push(img);
            // RGB
            hh = (RgbImage) seq.getFront();

            // YCbCr
            seq = new Sequence();
            seq.add(new Rgb2YCbCr());
```

```
            seq.push(hh);
            RgbImage hh2 = (RgbImage) seq.getFront();

            Gray8Image g = new Gray8Image(hh2.getWidth(),
hh2.getHeight());
            g.fill(new jjil.core.Rect(0, 0, g.getWidth(),
g.getHeight()),
                        Byte.MIN_VALUE);
            byte[] data = g.getData();
            int[] hdata = hh2.getData();

            int w = g.getWidth();
            int h = g.getHeight();
            int cb, cr;

            for (int i = 0; i < h; i++) {
                  for (int j = 0; j < w; j++) {
                        cb = (RgbVal.getG(hdata[i * w + j]) -
Byte.MIN_VALUE);
                        cr = (RgbVal.getB(hdata[i * w + j]) -
Byte.MIN_VALUE);

                        if (Math.abs(cb - cr) > 5 && (Math.abs(255 - cb
- cr) > 5))
                              data[i * w + j] = 0;
                        else if ((Math.abs(cb - cr) <= 5)
                                    || (Math.abs(255 - cb - cr) <= 5))
{
                              data[i * w + j] = 10;
                        }
                  }
            }

            seq = new Sequence();
            seq.add(new RgbHsv());
            seq.push(hh);
            hh2 = (RgbImage) seq.getFront();
            hdata = hh2.getData();
            w = g.getWidth();
            h = g.getHeight();
            for (int i = 0; i < h; i++) {
                  for (int j = 0; j < w; j++) {
                        cb = (RgbVal.getR(hdata[i * w + j]) -
Byte.MIN_VALUE);
                        if (cb >= 150 && cb <= 200 && data[i * w + j]
== 0) {
                              blue++;
                              data[i * w + j] = Byte.MAX_VALUE;
                        } else if (data[i * w + j] == 10 && cb >= 0 &&
cb <= 20) {
                              red++;
                              data[i * w + j] = Byte.MAX_VALUE;
                        } else if (data[i * w + j] == 0)
                              data[i * w + j] = Byte.MIN_VALUE;
                  }
            }
```

```
            System.out.println("TOTAL: " + w * h + ", Blue: " + blue +
", Red: "
                        + red);
            this.grayH = (Gray8Image) g.clone();
            this.currentImg = img;
            if (blue > 0 && blue > red
                        && ((g.getWidth() * g.getHeight()) / blue) <
230)
                  return 20;
            else if (red > 0 && red > blue
                        && ((g.getWidth() * g.getHeight()) / red) <
230)
                  return 50;
            else
                  return 0;
      }

      /**
       * @return result of number analysis (5, 10, 0 or 2050€)
       * @throws Error
       */
      public int analiseNumber() throws Error {
            Sequence seq = new Sequence();
            this.region = new Rect(img.getWidth() / 45, 2 *
img.getHeight() / 3,
                        img.getWidth() / 7 + img.getWidth() / 45,
img.getHeight() / 3
                                    + 2 * img.getHeight() / 3);
            seq.add(new Gray8Subimage(this.region));
            seq.push(this.grayH);
            Gray8Image ii = (Gray8Image) seq.getFront();

            new Gray8HistVertical();
            int[] histV = Gray8HistVertical.computeHistogram(ii);
            HistRegions regionsV = new HistRegions(histV.length);
            HistRegion elemV = null;
            int count = 0;
            int prev = -1;

            long sumWhite = 0;
            long sumBlack = 0;

            byte[] data = ii.getData();

            for (int i = 0; i < ii.getHeight() * ii.getWidth(); i++) {
                  if (data[i] == Byte.MIN_VALUE)
                        sumBlack++;
                  else if (data[i] == Byte.MAX_VALUE)
                        sumWhite++;
            }

            if (((double) sumBlack / (double) sumWhite > 20)) {
                  this.lowQ = true;
                  this.error = "Low quality number";
                  this.recognition = 1010;
                  return 0;
```

```
            }
            String type = null;

            int border = 1;

            for (int i = 0; i < histV.length; i++) {
                  if (elemV == null && histV[i] < border) {
                        elemV = new HistRegion(HistRegion.Tipus_ZERO);
                        prev = histV[i];
                        count++;
                  } else if (elemV == null && histV[i] > border) {
                        elemV = new HistRegion(HistRegion.Tipus_ONE);
                        prev = histV[i];
                        count++;
                  } else if (elemV != null && histV[i] < border && prev
< border) {
                        prev = histV[i];
                        count++;
                  } else if (elemV != null && histV[i] > border && prev
< border) {
                        elemV.setCount(count);
                        elemV.setPercent((count * 100) / histV.length);
                        if (elemV.getCount() > 2 && !
elemV.getType().equals(type)) {
                              regionsV.getHistRegions().add(elemV);
                              type = elemV.getType();
                              count = 0;
                              elemV = null;
                        } else {
                              prev = histV[i];
                              count++;
                        }

                  } else if (elemV != null && histV[i] > border && prev
> border) {
                        prev = histV[i];
                        count++;
                  } else if (elemV != null && histV[i] < border && prev
> border) {
                        elemV.setCount(count);
                        elemV.setPercent((count * 100) / histV.length);
                        if (elemV.getCount() > 2 && !
elemV.getType().equals(type)) {
                              regionsV.getHistRegions().add(elemV);
                              type = elemV.getType();
                              count = 0;
                              elemV = null;
                        } else {
                              prev = histV[i];
                              count++;
                        }
                  }
            }
            if (elemV == null || type == null) {
                  this.wrong = true;
                  this.error = "Can't find the number";
```

```
                this.recognition = 0;
                return 0;
            }
            elemV.setCount(count);
            elemV.setPercent((count * 100) / histV.length);
            regionsV.getHistRegions().add(elemV);

            if (regionsV.getHistRegions().get(0).getType()
                        .equals(HistRegion.Tipus_ONE)) {
                regionsV.getHistRegions().remove(0);
            }
            if (regionsV.getHistRegions().size() > 3) {
                if (regionsV.getHistRegions()
                            .get(regionsV.getHistRegions().size() -
1).getType()
                            .equals(HistRegion.Tipus_ZERO))
                    border = border
                                + regionsV.getHistRegions()
                                        .get(regionsV.getHistRe
gions().size() - 1)
                                        .getCount()
                                + regionsV.getHistRegions()
                                        .get(regionsV.getHistRe
gions().size() - 2)
                                        .getCount()
                                + +regionsV.getHistRegions()
                                        .get(regionsV.getHistRe
gions().size() - 3)
                                        .getCount();
                else
                    border = border
                                + regionsV.getHistRegions()
                                        .get(regionsV.getHistRe
gions().size() - 1)
                                        .getCount()
                                + regionsV.getHistRegions()
                                        .get(regionsV.getHistRe
gions().size() - 2)
                                        .getCount();
            } else
                border = 1;

            int width = 0;
            for (int i = 0; i < regionsV.getHistRegions().size(); i++)
{
                if (regionsV.getHistRegions().get(i).getCount() >
width) {
                    width =
regionsV.getHistRegions().get(i).getCount();
                }
            }

            seq = new Sequence();
            this.region = new Rect(img.getWidth() / 80, 2 *
img.getHeight() / 3,
                    img.getWidth() / 80 + img.getWidth() / 7, 2 *
```

```
img.getHeight()
                                    / 3 + img.getHeight() / 3 -
border);

            seq.add(new Gray8Subimage(this.region));
            seq.push(this.grayH);
            ii = (Gray8Image) seq.getFront();

            new Gray8HistHorizontal();
            int[] histH = Gray8HistHorizontal.computeHistogram(ii);
            HistRegions regions = new HistRegions(histH.length);
            regions.setWidth(width);
            HistRegion elem = null;
            count = 0;
            prev = -1;

            for (int i = 0; i < histH.length; i++) {
                  if (elem == null && histH[i] <= border) {
                        elem = new HistRegion(HistRegion.Tipus_ZERO);
                        prev = histH[i];
                        count++;
                  } else if (elem == null && histH[i] > border) {
                        elem = new HistRegion(HistRegion.Tipus_ONE);
                        prev = histH[i];
                        count++;
                  } else if (elem != null && histH[i] <= border && prev
<= border) {
                        prev = histH[i];
                        count++;
                  } else if (elem != null && histH[i] > border && prev
<= border) {
                        elem.setCount(count);
                        elem.setPercent((count * 100) / histH.length);
                        regions.getHistRegions().add(elem);
                        count = 0;
                        elem = new HistRegion(HistRegion.Tipus_ONE);
                        prev = histH[i];
                        count++;

                  } else if (elem != null && histH[i] > border && prev
> border) {
                        prev = histH[i];
                        count++;
                  } else if (elem != null && histH[i] <= border && prev
> border) {
                        elem.setCount(count);
                        elem.setPercent((count * 100) / histH.length);
                        regions.getHistRegions().add(elem);
                        count = 0;
                        elem = new HistRegion(HistRegion.Tipus_ZERO);
                        prev = histH[i];
                        count++;
                  }
            }
            elem.setCount(count);
            elem.setPercent((count * 100) / histH.length);
```

```
            regions.getHistRegions().add(elem);
            this.recognition = regions.analiseNumber(this.lowQ);
            this.currentImg = grayH;
            // long stop = System.currentTimeMillis();
            // System.out.println("analiseNumber: "+(stop-start));
            return 0;
      }

      /**
       * @return cut of RgbImage
       * @throws Error
       */
      public RgbImage cutImage() throws Error {
            // long start = System.currentTimeMillis();
            Sequence seq = new Sequence();
            seq.add(new RgbImageSubImage(region));
            seq.push(img);
            img = seq.getFront();
            seq = new Sequence();
            seq.add(new Gray8Subimage(region));
            seq.push(this.grayH);
            this.grayH = (Gray8Image) seq.getFront();
            // long stop = System.currentTimeMillis();
            // System.out.println("cutImage: "+(stop-start));
            return (RgbImage) img;
      }

      /**
       * @return true if denomination is 50€ and false if not
       * @throws Error
       */
      public boolean recognise50() throws Error {
            Sequence seq = new Sequence();
            this.region = new Rect((25 * img.getWidth()) / 30,
img.getHeight() / 2,
                        (img.getWidth() / 7) + ((25 * img.getWidth()) /
30),
                        (img.getHeight() / 2) + (img.getHeight() / 3));

            seq.add(new Gray8Subimage(this.region));
            seq.push(this.grayH);
            Gray8Image ii = (Gray8Image) seq.getFront();
            int[] hist2 = Gray8HistHorizontal.computeHistogram(ii);
            HistRegions regions = new HistRegions(hist2.length);
            HistRegion elem = null;
            int count = 0;
            int prev = -1;

            for (int i = 0; i < hist2.length; i++) {
                  if (elem == null && hist2[i] <= 0) {
                        elem = new HistRegion(HistRegion.Tipus_ZERO);
                        prev = hist2[i];
                        count++;
                  } else if (elem == null && hist2[i] > 0) {
                        elem = new HistRegion(HistRegion.Tipus_ONE);
                        prev = hist2[i];
```

```
                        count++;
                    } else if (elem != null && hist2[i] <= 0 && prev <=
0) {
                        prev = hist2[i];
                        count++;
                    } else if (elem != null && hist2[i] > 0 && prev <= 0)
{
                        elem.setCount(count);
                        elem.setPercent((count * 100) / hist2.length);
                        regions.getHistRegions().add(elem);
                        count = 0;
                        elem = new HistRegion(HistRegion.Tipus_ONE);
                        prev = hist2[i];
                        count++;

                    } else if (elem != null && hist2[i] > 0 && prev > 0)
{
                        prev = hist2[i];
                        count++;
                    } else if (elem != null && hist2[i] <= 0 && prev > 0)
{
                        elem.setCount(count);
                        elem.setPercent((count * 100) / hist2.length);
                        regions.getHistRegions().add(elem);
                        count = 0;
                        elem = new HistRegion(HistRegion.Tipus_ZERO);
                        prev = hist2[i];
                        count++;
                    }
                }
                elem.setCount(count);
                elem.setPercent((count * 100) / hist2.length);
                regions.getHistRegions().add(elem);
                this.currentImg = grayH;
                return regions.analise50(this.lowQ);
        }

        /**
         * Set final result image
         */
        public void setFinalResult() {
                this.currentImg = img;
                this.region = new Rect(0, 0, img.getWidth(),
img.getHeight());
        }
}


package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
```

```
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;

/**
 * Result activity
 * @author ANNA
 */
public class FinalActivity extends Activity {
	long state = 0;
	String result;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		requestWindowFeature(Window.FEATURE_NO_TITLE);

	getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.image);
		final MyView mm = (MyView) findViewById(R.id.myView);
		final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
		final TextView textViewLog = (TextView)
findViewById(R.id.TextViewLog);
		buttonNext.setText("  FINISH  ");
		textViewLog.setTextColor(Color.WHITE);
		buttonNext.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				try {
					Intent in = new
Intent(FinalActivity.this,

	cat.uvic.ui.MainActivity.class);
					startActivity(in);
					finish();
				} catch (Exception ex) {
					ex.printStackTrace();
				}
			}
		});

		Transformation trans;
		try {
			trans = Transformation.getInstance(null);
			ResultBitmap res = new ResultBitmap();
			trans.setFinalResult();
			res.setRegion(trans.getRegion());
			res.setBitmap(trans.getCurrentImg());
			mm.setRect(res.getRegion());
```

```
                    mm.setBtm(res.getBitmap());
                    mm.setResult(trans.getRecognition() + "");
                    if (trans.getRecognition() > 0)
                          result = "RESULT: " + trans.getRecognition() +
" €";
                    else
                          result = "Can't recognize the denomination.";
                    textViewLog.setText("  " + result);
                    mm.invalidate();
                    Transformation.destroy();
              } catch (IOException e) {
                    e.printStackTrace();
              } catch (Error e) {
                    e.printStackTrace();
              }
        }
}
```

```
      package cat.uvic.ui;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.InputStream;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;
```

```
/**
 * Main class that represents the main menu with all Buttons
 *         listeners implemented
 * @author ANNA
 */
public class MainActivity extends Activity {
 /** Called when the activity is first created. */
 CheckBox debug;
 static final String TMP_FILE = "/sdcard/TempPicture.jpg";

 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.home);
 final Button buttonGallery = (Button)
findViewById(R.id.ButtonGallery);
 final Button buttonCamera = (Button) findViewById(R.id.ButtonCamera);
 debug = (CheckBox) findViewById(R.id.CheckBoxDebug);
 final Button buttonExit = (Button) findViewById(R.id.ButtonExit);
 final Button buttonAbout = (Button) findViewById(R.id.ButtonAbout);

 Transformation.destroy();
 buttonGallery.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        try {
                              Intent in = new
Intent(MainActivity.this,
                                          cat.uvic.ui.MyGallery.class);
                              in.putExtra("DEBUG_MODE",
debug.isChecked());

                              startActivity(in);
                              finish();
                        } catch (Exception ex) {
                              ex.printStackTrace();
                        }
```

```
		}
	});

	buttonAbout.setOnClickListener(new View.OnClickListener()
{
		@Override
		public void onClick(View v) {
			AlertDialog alertDialog;
			alertDialog = new
AlertDialog.Builder(MainActivity.this)
					.create();
			alertDialog.setTitle("About...");
			alertDialog
					.setMessage("Program created as a
part of the diploma project \n"
							+ "of 'Master en
Tecnologies Aplicades a la Informacio' at the University of Vic. \n"
							+ "Author: Anna
Sobolewska");
			alertDialog.setButton("OK",
					new
DialogInterface.OnClickListener() {
						@Override
						public void
onClick(DialogInterface dialog,
								int which) {
							dialog.dismiss();
						}
					});
			alertDialog.show();
		}
	});

	buttonExit.setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View v) {
			finish();
			System.exit(0);
```

```
                }
            });

            buttonCamera.setOnClickListener(new View.OnClickListener()
{
                @Override
                public void onClick(View v) {
                      Intent intent = new
Intent("android.media.action.IMAGE_CAPTURE");
                      Uri uri = Uri.fromFile(new File(TMP_FILE));
                      intent.putExtra(MediaStore.EXTRA_OUTPUT, uri);
                      startActivityForResult(intent, 0);
                }
            });
      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
            if (requestCode == 0 && resultCode == Activity.RESULT_OK)
{
                Bitmap x = null;
                try {
                      File f = new File(TMP_FILE);
                      try {
                            Uri u = Uri.fromFile(f);
                            InputStream photoStream =
getContentResolver()
                                        .openInputStream(u);
                            BitmapFactory.Options opts = new
BitmapFactory.Options();
                            opts.inSampleSize = 4;
                            x =
BitmapFactory.decodeStream(photoStream, null, opts);
                            f.delete();
                      } catch (FileNotFoundException e) {
                            e.printStackTrace();
                      }
```

```
                        if (debug.isChecked()) {
                              Transformation trans =
Transformation.getInstance(x);
                              try {
                                    setContentView(R.layout.image);
                                    final MyView mm = (MyView)
findViewById(R.id.myView);

      mm.setRect(trans.findBill(trans.getImg(), true));
                                    mm.setBtm(trans.getCurrentImg());

      mm.setResult(trans.getRecognition() + "");

      mm.setHorizontalScrollBarEnabled(true);

      mm.setVerticalScrollBarEnabled(true);
                                    mm.setScrollContainer(true);
                                    final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
                                    buttonNext
                                                .setOnClickListener(new
View.OnClickListener() {
                                                      @Override
                                                      public void
onClick(View v) {
                                                            try {

      Intent in = new Intent(

            MainActivity.this,

            cat.uvic.ui.NextActivity.class);

      in.putExtra("STATE", (long) 1);

      startActivity(in);

      finish();
```

```
                                                                  } catch
(Exception ex) {

      ex.printStackTrace();
                                                                  }
                                                              }
                                                          });

                                } catch (Error e) {
                                      e.printStackTrace();
                                }
                          } else {
                                Transformation trans =
Transformation.getInstance(x);
                                ResultBitmap res = new ResultBitmap();
                                try {
                                      res = trans.transform();
                                      setContentView(R.layout.image);
                                      final MyView mm = (MyView)
findViewById(R.id.myView);
                                      mm.setRect(res.getRegion());
                                      mm.setBtm(res.getBitmap());

      mm.setResult(trans.getRecognition() + "");
                                      final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
                                      final TextView textViewLog =
(TextView) findViewById(R.id.TextViewLog);
                                      buttonNext.setText("  FINISH  ");

      textViewLog.setTextColor(Color.WHITE);
                                      String result;
                                      if (trans.getRecognition() > 0)
                                            result = "RESULT: " +
trans.getRecognition() + " €";
                                      else
                                            result = "Can't recognize the
denomination.";
```

```
                                    textViewLog.setText("   " +
result);
                                    buttonNext
                                                .setOnClickListener(new
View.OnClickListener() {
                                                      @Override
                                                      public void
onClick(View v) {
                                                            try {

      Intent in = new Intent(

            MainActivity.this,

            cat.uvic.ui.MainActivity.class);

      startActivity(in);

      finish();
                                                            } catch
(Exception ex) {

      ex.printStackTrace();
                                                            }
                                                      }
                                                });
                                    mm.invalidate();
                              } catch (Exception es) {
                              } catch (Error e) {
                                    e.printStackTrace();
                              }
                        }
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
      }
```

```
}
```

```
package cat.uvic.ui;

import java.io.InputStream;

import jjil.core.Error;
import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.*;

/**
 * Take photo from the gallery
 * @author ANNA
 */
public class MyGallery extends Activity {
	private boolean debug = false;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		debug = this.getIntent().getBooleanExtra("DEBUG_MODE",
false);
		requestWindowFeature(Window.FEATURE_NO_TITLE);

	getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);
		Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
		photoPickerIntent.setType("image/*");
		startActivityForResult(photoPickerIntent, 1);
	}

	protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
		super.onActivityResult(requestCode, resultCode, data);
		Transformation.destroy();
		if (resultCode == RESULT_OK) {
			Uri chosenImageUri = data.getData();
			if (chosenImageUri != null) {
				try {
					InputStream photoStream =
getContentResolver()
							.openInputStream(chosenImageU
```

```
ri);
                                BitmapFactory.Options opts = new
BitmapFactory.Options();
                                opts.inSampleSize = 6;
                                Bitmap mBitmap =
BitmapFactory.decodeStream(photoStream,
                                                null, opts);
                                // if (mBitmap.getWidth()>600 ||
mBitmap.getHeight()>600){
                                // int factor =
Math.max(mBitmap.getWidth()/300,
                                // mBitmap.getHeight()/300);
                                // mBitmap =
mBitmap.createScaledBitmap(mBitmap,
                                // mBitmap.getWidth()/factor,
mBitmap.getHeight()/factor,
                                // false);
                                // System.out.print("Factor: "+factor);
                                // }
                                if (debug) {
                                      Transformation trans =
Transformation
                                                      .getInstance(mBitmap);
                                      try {

      setContentView(R.layout.image);
                                                final MyView mm = (MyView)
findViewById(R.id.myView);

      mm.setRect(trans.findBill(trans.getImg(), true));

      mm.setBtm(trans.getCurrentImg());

      mm.setResult(trans.getRecognition() + "");
                                                mm.invalidate();

                                                final Button buttonNext =
(Button) findViewById(R.id.ButtonNext);
                                                buttonNext
                                                            .setOnClickListen
er(new View.OnClickListener() {
                                                                  @Override
                                                                  public void
onClick(View v) {
                                                                        try {

      Intent in = new Intent(

                  MyGallery.this,

                  cat.uvic.ui.NextActivity.class);

      in.putExtra("STATE", (long) 1);

      startActivity(in);
                                                                  }
```

```
catch (Exception ex) {

      ex.printStackTrace();
                                                                  }
                                                              }
                                                          });

                                    } catch (Error e) {
                                          e.printStackTrace();
                                    }
                              } else {
                                    Transformation trans =
Transformation
                                                .getInstance(mBitmap);
                                    ResultBitmap res = new
ResultBitmap();
                                    try {
                                          res = trans.transform();

      setContentView(R.layout.image);
                                          final MyView mm = (MyView)
findViewById(R.id.myView);
                                          mm.setRect(res.getRegion());
                                          mm.setBtm(res.getBitmap());

      mm.setResult(trans.getRecognition() + "");
                                          final Button buttonNext =
(Button) findViewById(R.id.ButtonNext);
                                          final TextView textViewLog =
(TextView) findViewById(R.id.TextViewLog);
                                          buttonNext.setText("  FINISH
");

      textViewLog.setTextColor(Color.WHITE);
                                          String result;
                                          if (trans.getRecognition() >
0)
                                                result = "RESULT: " +
trans.getRecognition()
                                                      + " €";
                                          else
                                                result = "Can't
recognize the denomination.";
                                          textViewLog.setText("  " +
result);
                                          buttonNext
                                                .setOnClickListen
er(new View.OnClickListener() {
                                                      @Override
                                                      public void
onClick(View v) {
                                                            try {

      Intent in = new Intent(

                  MyGallery.this,
```

```
                    cat.uvic.ui.MainActivity.class);

      startActivity(in);

      finish();
                                                                        }
catch (Exception ex) {

      ex.printStackTrace();
                                                                        }
                                                                  }
                                                            });
                                          mm.invalidate();
                                    } catch (Exception es) {
                                    } catch (Error e) {
                                          e.printStackTrace();
                                    }
                              }
                              photoStream.close();
                        } catch (Exception e) {
                              e.printStackTrace();
                        }
                  }
            }
      }
}
```

```
package cat.uvic.ui;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;

/**
 * Customized View that show the image with drawn region of
 *         interes.
 * @author ANNA
 */
public class MyView extends View {
      Bitmap btm;
      Rect rect;
      String result;

      /**
       * @return result image Bitmap
       */
      public Bitmap getBtm() {
            return btm;
```

```
}

/**
 * @param btm
 */
public void setBtm(Bitmap btm) {
	this.btm = btm;
}

/**
 * @return region of analysis
 */
public Rect getRect() {
	return rect;
}

/**
 * @param rect
 */
public void setRect(Rect rect) {
	this.rect = rect;
}

/**
 * @return result of recognition
 */
public String getResult() {
	return result;
}

/**
 * @param result
 */
public void setResult(String result) {
	this.result = result;
}

/**
 * @param context
 */
public MyView(Context context) {
	super(context);
}

/**
 * @param context
 * @param attrs
 */
public MyView(Context context, AttributeSet attrs) {
	super(context, attrs);
}

/**
 * @param context
 *            current Context
 * @param bitmap
```

```
         *            Image bitmap
         * @param region
         *            Rectangle region
         * @param string
         *            result string Constructor of My View
         */
        public MyView(Context context, Bitmap bitmap, Rect region,
String string) {
              super(context);
              this.btm = bitmap;
              this.rect = region;
              result = string;
              this.setHorizontalScrollBarEnabled(true);
              this.setVerticalScrollBarEnabled(true);
        }

        @Override
        protected void onDraw(Canvas canvas) {
              canvas.drawBitmap(btm, 0, 0, null);
              Paint p = new Paint();
              p.setColor(Color.MAGENTA);
              canvas.drawBitmap(btm, new Matrix(), p);
              if (rect != null && rect.width() > 0 && rect.height() > 0)
{
                    canvas.drawLine(rect.left, rect.top, rect.left +
rect.width(),
                                rect.top, p);
                    canvas.drawLine(rect.left, rect.top, rect.left,
                                rect.top + rect.height(), p);
                    canvas.drawLine(rect.left, rect.top + rect.height(),
rect.left
                                + rect.width(), rect.top + rect.height(),
p);
                    canvas.drawLine(rect.left + rect.width(), rect.top,
rect.left
                                + rect.width(), rect.top + rect.height(),
p);
              }
              canvas.save();
              invalidate();
        }
}


package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

/**
 * Activity responsible for locate the banknote
 * @author ANNA
 */
public class NextActivity extends Activity {
	long state = 0;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		state = this.getIntent().getLongExtra("STATE", 0);
		requestWindowFeature(Window.FEATURE_NO_TITLE);

	getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.image);
		final MyView mm = (MyView) findViewById(R.id.myView);
		final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
		buttonNext.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				try {
					Intent in = new Intent(NextActivity.this,

	cat.uvic.ui.NextActivity2.class);
					in.putExtra("STATE", state);
					startActivity(in);
					finish();
				} catch (Exception ex) {
					ex.printStackTrace();
				}
			}
		});
		if (state == 1) {
			Transformation trans;
			try {
				this.state = 2;
				trans = Transformation.getInstance(null);
				ResultBitmap res = new ResultBitmap();
				trans.cutImage();
				res.setRegion(trans.findBill(trans.getImg(),
false));
				res.setBitmap(trans.getCurrentImg());

				mm.setRect(res.getRegion());
				mm.setBtm(res.getBitmap());
				mm.setResult(trans.getRecognition() + "");
				mm.invalidate();
			} catch (IOException e) {
				state = 0;
				e.printStackTrace();
			} catch (Error e) {
```

```
                        state = 0;
                        e.printStackTrace();
                  }
            }
      }
}


package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

/**
 * Activity responsible for the banknote rotation
 * @author ANNA
 */
public class NextActivity2 extends Activity {
      long state = 0;

      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            state = this.getIntent().getLongExtra("STATE", 0);
            requestWindowFeature(Window.FEATURE_NO_TITLE);

      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);
            setContentView(R.layout.image);
            final MyView mm = (MyView) findViewById(R.id.myView);
            final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
            buttonNext.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        try {
                              Intent in = new
Intent(NextActivity2.this,

      cat.uvic.ui.NextActivity3.class);
                              in.putExtra("STATE", state);
                              startActivity(in);
                              finish();
                        } catch (Exception ex) {
                              ex.printStackTrace();
                        }
```

```
                }
            });
            if (state == 2) {
                this.state = 3;
                Transformation trans;
                try {
                    trans = Transformation.getInstance(null);

                    ResultBitmap res = new ResultBitmap();
                    trans.checkPosition();
                    res.setRegion(trans.getRegion());
                    res.setBitmap(trans.getCurrentImg());
                    mm.setRect(res.getRegion());
                    mm.setBtm(res.getBitmap());
                    mm.setResult(trans.getRecognition() + "");
                    mm.invalidate();
                } catch (IOException e) {
                    state = 0;
                    e.printStackTrace();
                } catch (Error e) {
                    state = 0;
                    e.printStackTrace();
                } catch (Exception e) {
                    state = 0;
                    e.printStackTrace();
                }
            }
        }
}


package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

/**
 * Activity responsible for hologram recognition
 * @author ANNA
 */
public class NextActivity3 extends Activity {
        long state = 0;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```

```
            state = this.getIntent().getLongExtra("STATE", 0);
            requestWindowFeature(Window.FEATURE_NO_TITLE);

      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);
            setContentView(R.layout.image);
            final MyView mm = (MyView) findViewById(R.id.myView);
            final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
            buttonNext.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        try {
                              if (state == 0) {
                                    Intent in = new
Intent(NextActivity3.this,

      cat.uvic.ui.FinalActivity.class);
                                    startActivity(in);
                                    finish();
                              } else {
                                    Intent in = new
Intent(NextActivity3.this,

      cat.uvic.ui.NextActivity4.class);
                                    in.putExtra("STATE", state);
                                    startActivity(in);
                                    finish();
                              }
                        } catch (Exception ex) {
                              ex.printStackTrace();
                        }
                  }
            });
            if (state == 3) {
                  this.state = 4;
                  Transformation trans;
                  try {
                        trans = Transformation.getInstance(null);
                        if (!trans.isWrong()) {
                              ResultBitmap res = new ResultBitmap();
                              if (trans.recognise50()) {
                                    state = 0;
                                    trans.setRecognition(50);
                              }
                              res.setRegion(trans.getRegion());
                              System.out.println("L: " +
trans.getRegion().left + " R: "
                                          + trans.getRegion().right + "
B: "
                                          + trans.getRegion().bottom +
" T: "
                                          + trans.getRegion().top + "
W: "
                                          + trans.getRegion().width() +
" H: "
```

```
                                                +
trans.getRegion().height());
                                    res.setBitmap(trans.getCurrentImg());
                                    mm.setRect(res.getRegion());
                                    mm.setBtm(res.getBitmap());
                                    mm.setResult(trans.getRecognition() +
"");
                              } else {
                                    this.state = 0;
                                    mm.setResult("0");
                              }
                              mm.invalidate();
                        } catch (IOException e) {
                              state = 0;
                              e.printStackTrace();
                        } catch (Error e) {
                              state = 0;
                              e.printStackTrace();
                        }
                  }
            }
}
```

```
package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

/**
 * Activity responsible for denomination analysis
 * @author ANNA
 */
public class NextActivity4 extends Activity {
      long state = 0;

      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            state = this.getIntent().getLongExtra("STATE", 0);
            requestWindowFeature(Window.FEATURE_NO_TITLE);

      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);
            setContentView(R.layout.image);
            final MyView mm = (MyView) findViewById(R.id.myView);
```

```
            final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
            buttonNext.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        try {
                              if (state == 0) {
                                    Intent in = new
Intent(NextActivity4.this,

      cat.uvic.ui.FinalActivity.class);
                                    startActivity(in);
                                    finish();
                              } else {
                                    Intent in = new
Intent(NextActivity4.this,

      cat.uvic.ui.NextActivity5.class);
                                    in.putExtra("STATE", state);
                                    startActivity(in);
                                    finish();
                              }

                        } catch (Exception ex) {
                              ex.printStackTrace();
                        }
                  }
            });
            if (state == 4) {
                  this.state = 5;
                  Transformation trans;
                  try {
                        trans = Transformation.getInstance(null);
                        ResultBitmap res = new ResultBitmap();
                        trans.analiseNumber();
                        res.setRegion(trans.getRegion());
                        res.setBitmap(trans.getCurrentImg());
                        mm.setRect(res.getRegion());
                        mm.setBtm(res.getBitmap());
                        mm.setResult(trans.getRecognition() + "");
                        mm.invalidate();
                  } catch (IOException e) {
                        state = 0;
                        e.printStackTrace();
                  } catch (Error e) {
                        state = 0;
                        e.printStackTrace();
                  }
            } else {
                  state = 0;
            }
      }
}
```

```
package cat.uvic.ui;

import java.io.IOException;

import jjil.core.Error;

import cat.uvic.android.ResultBitmap;
import cat.uvic.calculs.ImageException;
import cat.uvic.calculs.Transformation;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;

/**
 * Activity responsible for color analysis
 * @author ANNA
 */
public class NextActivity5 extends Activity {
	long state = 0;
	String result;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		state = this.getIntent().getLongExtra("STATE", 0);
		requestWindowFeature(Window.FEATURE_NO_TITLE);

	getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
				WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.image);
		final MyView mm = (MyView) findViewById(R.id.myView);
		final Button buttonNext = (Button)
findViewById(R.id.ButtonNext);
		final TextView textViewLog = (TextView)
findViewById(R.id.TextViewLog);
		// buttonNext.setText("  FINISH  ");
		textViewLog.setTextColor(Color.WHITE);
		mm.invalidate();
		buttonNext.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				try {
					Intent in = new
Intent(NextActivity5.this,

	cat.uvic.ui.FinalActivity.class);
					startActivity(in);
					finish();
				} catch (Exception ex) {
					ex.printStackTrace();
				}
```

```
                }
            });
            if (state == 5) {
                this.state = 6;
                Transformation trans;
                try {
                    trans = Transformation.getInstance(null);
                    ResultBitmap res = new ResultBitmap();
                    if (trans.getRecognition() != 5 &&
trans.getRecognition() != 10
                                    && trans.getRecognition() != 1010
                                    && trans.getRecognition() != 101) {

      trans.setRecognition(trans.analiseColor());
                    }
                    res.setRegion(trans.getRegion());
                    res.setBitmap(trans.getCurrentImg());
                    mm.setRect(res.getRegion());
                    mm.setBtm(res.getBitmap());
                    mm.setResult(trans.getRecognition() + "");
                    mm.invalidate();
                } catch (IOException e) {
                    state = 0;
                    e.printStackTrace();
                } catch (Error e) {
                    state = 0;
                    e.printStackTrace();
                } catch (ImageException e) {
                    state = 0;
                    e.printStackTrace();
                }
            }
      }
}
```

```
package cat.uvic.android;

/**
 * This is the class used to create Strings from Error objects when
running
 * under Android. The point is that the Android localization support
is used so
 * the same approach won't work with J2ME, etc.<br>
 * Usage:<br>
 * jjil.android.Error eAndroid = new jjil.android.Error(e);<br>
 * ... eAndroid.toString() ...
 *
 * @author webb
 *
 */
public class Error extends jjil.core.Error {
      /**
       *
       */
      private static final long serialVersionUID = 1L;
```

```
      private static final String szMessage[][] = new
String[jjil.core.Error.PACKAGE.COUNT][];

      {
            Error.szMessage[jjil.core.Error.PACKAGE.CORE] = new
String[jjil.core.ErrorCodes.COUNT];
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.BOUNDS_OUTSIDE_IMAGE] = Messages
                        .getString("BOUNDS_OUTSIDE_IMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.ILLEGAL_PARAMETER_VALUE] = Messages
                        .getString("ILLEGAL_PARAMETER_VALUE");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.IMAGE_MASK_SIZE_MISMATCH] = Messages
                        .getString("IMAGE_MASK_SIZE_MISMATCH");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.MATH_DIVISION_ZERO] = Messages
                        .getString("MATH_DIVISION_ZERO");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.MATH_NEGATIVE_SQRT] = Messages
                        .getString("MATH_NEGATIVE_SQRT");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.MATH_PRODUCT_TOO_LARGE] = Messages
                        .getString("MATH_PRODUCT_TOO_LARGE");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.MATH_SQUARE_TOO_LARGE] = Messages
                        .getString("MATH_SQUARE_TOO_LARGE");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.NO_RESULT_AVAILABLE] = Messages
                        .getString("PIPELINE_NO_RESULT");
            Error.szMessage[jjil.core.Error.PACKAGE.CORE]
[jjil.core.ErrorCodes.PIPELINE_EMPTY_PUSH] = Messages
                        .getString("PIPELINE_EMPTY_PUSH");

            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM] = new
String[jjil.algorithm.ErrorCodes.COUNT];
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.CONN_COMP_LABEL_COMPARETO_NULL] = Messages
                        .getString("CONN_COMP_LABEL_COMPARETO_NULL");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.CONN_COMP_LABEL_OUT_OF_BOUNDS] = Messages
                        .getString("CONN_COMP_LABEL_OUT_OF_BOUNDS");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.INPUT_TERMINATED_EARLY] = Messages
                        .getString("INPUT_TERMINATED_EARLY");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.FFT_SIZE_LARGER_THAN_MAX] = Messages
                        .getString("FFT_SIZE_LARGER_THAN_MAX");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.FFT_SIZE_NOT_POWER_OF_2] = Messages
                        .getString("FFT_SIZE_NOT_POWER_OF_2");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.HEAP_EMPTY] = Messages
                        .getString("HEAP_EMPTY");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.HISTOGRAM_LENGTH_NOT_256] = Messages
```

```
                        .getString("HISTOGRAM_LENGTH_NOT_256");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.ILLEGAL_COLOR_CHOICE] = Messages
                        .getString("ILLEGAL_COLOR_CHOICE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_COMPLEX32IMAGE] = Messages
                        .getString("IMAGE_NOT_COMPLEX32IMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_GRAY16IMAGE] = Messages
                        .getString("IMAGE_NOT_GRAY16IMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_GRAY32IMAGE] = Messages
                        .getString("IMAGE_NOT_GRAY32IMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_GRAY8IMAGE] = Messages
                        .getString("IMAGE_NOT_GRAY8IMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_RGBIMAGE] = Messages
                        .getString("IMAGE_NOT_RGBIMAGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_NOT_SQUARE] = Messages
                        .getString("IMAGE_NOT_SQUARE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_SIZES_DIFFER] = Messages
                        .getString("IMAGE_SIZES_DIFFER");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IMAGE_TOO_SMALL] = Messages
                        .getString("IMAGE_TOO_SMALL");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.INPUT_IMAGE_SIZE_NEGATIVE] = Messages
                        .getString("INPUT_IMAGE_SIZE_NEGATIVE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.INPUT_TERMINATED_EARLY] = Messages
                        .getString("INPUT_TERMINATED_EARLY");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.IO_EXCEPTION] = Messages
                        .getString("IO_EXCEPTION");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.LOOKUP_TABLE_LENGTH_NOT_256] = Messages
                        .getString("LOOKUP_TABLE_LENGTH_NOT_256");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.OBJECT_NOT_EXPECTED_TYPE] = Messages
                        .getString("OBJECT_NOT_EXPECTED_TYPE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.OUTPUT_IMAGE_SIZE_NEGATIVE] = Messages
                        .getString("OUTPUT_IMAGE_SIZE_NEGATIVE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.PARAMETER_OUT_OF_RANGE] = Messages
                        .getString("PARAMETER_OUT_OF_RANGE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.PARAMETER_RANGE_NULL_OR_NEGATIVE] =
Messages
                        .getString("PARAMETER_RANGE_NULL_OR_NEGATIVE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.PARSE_ERROR] = Messages
                        .getString("PARSE_ERROR");
```

```
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.REDUCE_INPUT_IMAGE_NOT_MULTIPLE_OF_OUTPUT_S
IZE] = Messages
                        .getString("REDUCE_INPUT_IMAGE_NOT_MULTIPLE_OF_
OUTPUT_SIZE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.SHRINK_OUTPUT_LARGER_THAN_INPUT] = Messages
                        .getString("SHRINK_OUTPUT_LARGER_THAN_INPUT");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.STATISTICS_VARIANCE_LESS_THAN_ZERO] =
Messages
                        .getString("STATISTICS_VARIANCE_LESS_THAN_ZERO"
);
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.STRETCH_OUTPUT_SMALLER_THAN_INPUT] =
Messages
                        .getString("STRETCH_OUTPUT_SMALLER_THAN_INPUT")
;
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.SUBIMAGE_NO_IMAGE_AVAILABLE] = Messages
                        .getString("SUBIMAGE_NO_IMAGE_AVAILABLE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.THRESHOLD_NEGATIVE] = Messages
                        .getString("THRESHOLD_NEGATIVE");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.WARP_END_LEFT_COL_GE_END_RIGHT_COL] =
Messages
                        .getString("WARP_END_LEFT_COL_GE_END_RIGHT_COL"
);
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.WARP_START_LEFT_COL_GE_START_RIGHT_COL] =
Messages
                        .getString("WARP_START_LEFT_COL_GE_START_RIGHT_
COL");
            Error.szMessage[jjil.core.Error.PACKAGE.ALGORITHM]
[jjil.algorithm.ErrorCodes.WARP_START_ROW_GE_END_ROW] = Messages
                        .getString("WARP_START_ROW_GE_END_ROW");
      }

      /**
       * @param e
       */
      public Error(jjil.core.Error e) {
            super(e);
      }

      public String getLocalizedMessage() {
            String szResult = null;
            switch (this.getPackage()) {
            case Error.PACKAGE.CORE:
                  if (this.getCode() < 0
                              || this.getCode() >=
jjil.core.ErrorCodes.COUNT) {
                        szResult =
Messages.getString("Illegal_error_code_core")
                                    + new
```

```
Integer(this.getCode()).toString();
                  } else {
                        szResult = szMessage[this.getPackage()]
[this.getCode()];
                  }
                  break;
            case Error.PACKAGE.ALGORITHM:
                  if (this.getCode() < 0
                              || this.getCode() >=
jjil.algorithm.ErrorCodes.COUNT) {
                        szResult =
Messages.getString("Illegal_error_code_algorithm")
                                    + new
Integer(this.getCode()).toString();
                  } else {
                        szResult = szMessage[this.getPackage()]
[this.getCode()];
                  }
                  break;
            case jjil.core.Error.PACKAGE.J2ME:
                  szResult =
Messages.getString("Illegal_error_code_j2me") + " "
                              + new Integer(this.getCode()).toString();
                  break;
            default:
                  szResult =
Messages.getString("Illegal_error_code_package") + " "
                              + new
Integer(this.getPackage()).toString() + " "
                              + new Integer(this.getCode()).toString();
                  break;
            }
            return szResult + ":  " + parameters();
      }

}
```

```
package cat.uvic.android;

/**
 * @author ANNA
 *
 */
public class ImageException extends Exception {
      /**
       *
       */
      private static final long serialVersionUID = 1L;

      /**
       *
       */
      public ImageException() {
            super();
      }
```

```
	/**
	 * @param arg0
	 * @param arg1
	 */
	public ImageException(String arg0, Throwable arg1) {
		super(arg0, arg1);
	}

	/**
	 * @param arg0
	 */
	public ImageException(String arg0) {
		super(arg0);
	}

	/**
	 * @param arg0
	 */
	public ImageException(Throwable arg0) {
		super(arg0);
	}
}
```

```
package cat.uvic.android;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

/**
 * @author ANNA
 *
 */
public class Messages {
	private static final String BUNDLE_NAME =
"jjil.android.messages"; //$NON-NLS-1$

	private static final ResourceBundle RESOURCE_BUNDLE =
ResourceBundle
			.getBundle(BUNDLE_NAME);

	private Messages() {
	}

	/**
	 * @param key
	 * @return name of Resource
	 */
	public static String getString(String key) {
		try {
			return RESOURCE_BUNDLE.getString(key);
		} catch (MissingResourceException e) {
			return '!' + key + '!';
		}
```

```
	}
}


package cat.uvic.android;

import android.graphics.Bitmap;
import android.graphics.Rect;

/**
 * @author ANNA
 *
 */
public class ResultBitmap {
	Bitmap bitmap;
	Rect region;

	/**
	 * @return image bitmap
	 */
	public Bitmap getBitmap() {
		return bitmap;
	}

	/**
	 * @param bitmap
	 */
	public void setBitmap(Bitmap bitmap) {
		this.bitmap = bitmap;
	}

	/**
	 * @return processed region
	 */
	public Rect getRegion() {
		return region;
	}

	/**
	 * @param region
	 */
	public void setRegion(Rect region) {
		this.region = region;
	}

	/**
	 * @param bitmap
	 * @param region
	 */
	public ResultBitmap(final Bitmap bitmap, final Rect region) {
		super();
		this.bitmap = bitmap;
		this.region = region;
	}

	/**
```

```
	 *
	 */
	public ResultBitmap() {
	}

}


package cat.uvic.android;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import jjil.core.RgbImage;
import android.content.Context;
import android.graphics.Bitmap;

/**
 * @author ANNA
 *
 */
public class RgbImageAndroid {
	/**
	 * The sole way to create an RgbImage from an image captured
from the
	 * camera. The parameters are the pointer to the byte data
passed to the
	 * JPEG picture callback and the width and height image you
want. You must
	 * reduce the image size because otherwise you will run out of
memory. Width
	 * and height reduction by a factor of 2 works on the GPhone.
	 * <p>
	 * Ex. usage
	 * <p>
	 * public void onPictureTaken(byte [] jpegData,
android.hardware.Camera
	 * camera) { RgbImage rgb = RgbImageAndroid.toRgbImage(jpegData,
	 * camera.getParameters().getPictureSize().width/2,
	 * camera.getParameters().getPictureSize().height/2); }
	 *
	 * @param bmp
	 * @return RgbImage initialized with the image from the camera.
	 */
	static public RgbImage toRgbImage(Bitmap bmp) {
		int nWidth = bmp.getWidth();
		int nHeight = bmp.getHeight();
		RgbImage rgb = new RgbImage(nWidth, nHeight);
		bmp.getPixels(rgb.getData(), 0, nWidth, 0, 0, nWidth,
nHeight);
		return rgb;
	}

	/**
	 * @param rgb
```

```
         * @return image bitmap
         */
        static public Bitmap toBitmap(RgbImage rgb) {
              return Bitmap.createBitmap(rgb.getData(), rgb.getWidth(),
                          rgb.getHeight(), Bitmap.Config.ARGB_8888);
        }

        /**
         * @param context
         * @param rgb
         */
        static public void toDisplay(Context context, RgbImage rgb) {
              // Bitmap bmp = toBitmap(rgb);

        }

        /**
         * @param context
         * @param rgb
         * @param nQuality
         * @param szPath
         * @throws IOException
         */
        static public void toFile(Context context, RgbImage rgb, int
nQuality,
                    String szPath) throws IOException {
              OutputStream os = new FileOutputStream(szPath);
              try {
                    Bitmap bmp = toBitmap(rgb);
                    Bitmap.CompressFormat format =
Bitmap.CompressFormat.JPEG;
                    szPath = szPath.toLowerCase();
                    if (szPath.endsWith("jpg") ||
szPath.endsWith("jpeg")) { //$NON-NLS-1$ //$NON-NLS-2$
                          format = Bitmap.CompressFormat.JPEG;
                    } else if (szPath.endsWith("png")) { //$NON-NLS-1$
                          format = Bitmap.CompressFormat.PNG;
                    }
                    bmp.compress(format, nQuality, os);
              } finally {
                    os.close();
              }
        }
}


package cat.uvic.android;

import java.util.HashMap;

import android.os.SystemClock;

/**
 * @author ANNA
 *
 */
```

```
public class TimeTracker implements jjil.core.TimeTracker {
	private class Times {
		long mlStart = 0, mlElapsed = 0, mlCumulative = 0;

		public Times(long lStart) {
			this.mlStart = lStart;
		}

		public long getCumulative() {
			return this.mlCumulative;
		}

		public void setEnd(long lEnd) {
			this.mlElapsed = lEnd - this.mlStart;
			this.mlCumulative += this.mlElapsed;
			this.mlStart = 0;
		}

		public void setStart(long lStart) {
			this.mlStart = lStart;
		}
	}

	private static TimeTracker sTimeTracker;
	private HashMap<String, Times> mhmTimes;

	static {
		sTimeTracker = new TimeTracker();
	}

	private TimeTracker() {
		this.mhmTimes = new HashMap<String, Times>();
	}

	@Override
	public String getCumulativeTimes() {
		String szCumulative = null;
		for (String szTask : this.mhmTimes.keySet()) {
			if (szCumulative != null) {
				szCumulative += ", ";
			} else {
				szCumulative = "";
			}
			Times t = this.mhmTimes.get(szTask);
			assert t != null;
			long l = t.getCumulative();
			if (l > 1000) {
				szCumulative += szTask + ": " +
(t.getCumulative() / 1000.0)
						+ "s";
			} else {
				szCumulative += szTask + ": " +
t.getCumulative() + "ms";
			}
		}
		return szCumulative;
```

```
	}

	/**
	 * @return single instance of TimeTracker
	 */
	public static TimeTracker getInstance() {
		return sTimeTracker;
	}

	public void startTask(String szTaskName) {
		long lTime = SystemClock.currentThreadTimeMillis();
		Times t = this.mhmTimes.get(szTaskName);
		if (t == null) {
			this.mhmTimes.put(szTaskName, new Times(lTime));
		} else {
			t.setStart(lTime);
		}
	}

	public void endTask(String szTaskName) {
		long lTime = SystemClock.currentThreadTimeMillis();
		Times t = this.mhmTimes.get(szTaskName);
		if (t != null) {
			t.setEnd(lTime);
		}
	}

	public void reset() {
		this.mhmTimes.clear();
	}

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
	android:versionCode="1" android:versionName="1.0"
package="cat.uvic.ui">
	<application android:icon="@drawable/euro2"
android:label="@string/app_name"
		android:debuggable="true" android:allowBackup="true"
android:enabled="true">
		<activity android:label="@string/app_name"
android:name=".MainActivity"
			android:screenOrientation="landscape">
			<intent-filter>
				<action
android:name="android.intent.action.MAIN" />
				<category
android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<activity android:label="@string/app_name"
android:name=".MyGallery"
			android:screenOrientation="landscape"></activity>
		<activity android:name=".NextActivity"
android:label="@string/app_name"
```

```
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
            <activity android:name=".NextActivity2"
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
            <activity android:name=".NextActivity3"
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
            <activity android:name=".NextActivity4"
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
            <activity android:name=".NextActivity5"
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
            <activity android:name=".FinalActivity"
                android:screenOrientation="landscape"
android:launchMode="singleInstance"
                android:noHistory="true"></activity>
      </application>

      <uses-permission
android:name="android.permission.CAMERA"></uses-permission>
      <uses-feature android:name="android.hardware.camera" />
      <uses-feature android:name="android.hardware.camera.autofocus"
/>
</manifest>


<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
<TableRow >
      <Button android:text="  Next  "
      android:layout_height="wrap_content"
      android:id="@+id/ButtonNext"
      android:layout_width="fill_parent"
      android:drawableLeft="@drawable/next"></Button>
      <TextView style="@style/CodeFont" android:text=""
android:id="@+id/TextViewLog" android:layout_width="fill_parent"
android:layout_height="wrap_content"></TextView>
</TableRow>
<TableRow >
      <cat.uvic.ui.MyView android:layout_span="2"
android:id="@+id/myView" android:layout_width="fill_parent"
android:layout_height="fill_parent"></cat.uvic.ui.MyView>
</TableRow>
</TableLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
      android:layout_width="fill_parent"
android:layout_height="fill_parent"
      xmlns:android="http://schemas.android.com/apk/res/android">

      <TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
android:layout_height="wrap_content"
            android:stretchColumns="2">
            <TableRow>
                  <Button android:text="   Camera   "
android:layout_height="wrap_content"
                        android:autoText="true"
android:id="@+id/ButtonCamera"
                        android:layout_width="wrap_content"
android:drawableLeft="@drawable/camera">
                  </Button>
                  <Button android:text="   Gallery   "
android:autoText="true"
                        android:layout_height="wrap_content"
android:id="@+id/ButtonGallery"
                        android:layout_width="wrap_content"
android:drawableLeft="@drawable/gallery">
                  </Button>
            </TableRow>
            <TableRow>
                  <Button android:text="   ABOUT   "
android:autoText="true"
                        android:layout_height="wrap_content"
android:id="@+id/ButtonAbout"
                        android:layout_width="wrap_content"
android:drawableLeft="@drawable/about">
                  </Button>
                  <Button android:text="   EXIT   "
android:autoText="true"
                              android:layout_height="wrap_content"
android:id="@+id/ButtonExit"
                              android:layout_width="wrap_content"
android:drawableLeft="@drawable/exit">
                  </Button>
            </TableRow>
            <TableRow >
                  <CheckBox android:layout_span="2"
android:layout_height="wrap_content" android:id="@+id/CheckBoxDebug"
                  android:text=" Debug mode "
android:layout_width="match_parent"></CheckBox>
            </TableRow>
      </TableLayout>
</LinearLayout>
```

BIBLIOGRAPHY

[1] Google code [online]: *JILL library*
http://code.google.com/p/jjil/
(access 01-2011)

[2] Wikipedia, free encyclopedia [online]: *Android (operating system)*
http://en.wikipedia.org/wiki/Android_(operating_system)
(access 06-2011)

[3] Android developer [online]: *What is Android?:*
http://developer.android.com/guide/basics/what-is-android.html
(access 06-2011)

[4] Wikipedia, free encyclopedia [online]: *Canny edge detector*
http://en.wikipedia.org/wiki/Canny_edge_detector
(access 08-2011)

[5] Wikipedia, free encyclopedia [online]: Rotation
http://en.wikipedia.org/wiki/Rotation
(access 04-2011)

[6] Wikipedia, free encyclopedia [online]: *Hough transform*
http://en.wikipedia.org/wiki/Hough_transform
(access 04-2011)

[7] Bank of France, *Fact sheet 138: Euro banknotes and coins: characteristics and use*, Bank of France, February 2008

[8] TAO Linmi & XU Guangyou, *Color in machine vision and its application*, Tsinghua University, September 2001

[9] Wikipedia, free encyclopedia [online]: *Color space*
http://en.wikipedia.org/wiki/Color_space
(access 08-2011)

[10] Wikipedia, free encyclopedia [online]: *HSL and HSV*
http://en.wikipedia.org/wiki/HSL_and_HSV
(access 08-2011)

[11] Wikipedia, free encyclopedia [online]: *RGB color model*
http://en.wikipedia.org/wiki/RGB_color_model
(access 08-2011)

[12] Sanjay Kr. Singh, D. S. Chauhan, Mayank Vatsa, Richa Singh, *A Robust Skin Color Based Face Detection Algorithm,* Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227-234 (2003)

[13] *Pautes per a la confecció de la memòria del Treball Final de Màster.* Vic: Universitat de Vic, 2011.

[14] Ramon Reig Bolaño, *Procesado de Datos Multidimencionales*, Vic: Universitat de Vic, 2010.

[15] TAO Linmi, XU Guangyou, *Color in machine vision and its application,* Chinese Science Bulletin Vol. 46 No. 17 September 2001

[16] Constantin Vertan, Nozha Boujemaa, *Color Texture Classification by Normalized Color Space Representation,* INRIA Rocquancourt – Projet IMEDIA, Domaine de Voluceau BP105 Rocquancourt, 78153 Le Chesnay Cedex, France [2000]

[17] Jae Young Choi, Yong Man Ro, Konstantinos N. Plataniotis, *Boosting Color Feature Selection for Color Face Recognition,* IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 20, NO. 5, [MAY 2011]

[18] Charles Goodwin, *Practices of Color Classification, Cognitive Studies*: Bulletin of the Japanese Cognitive Science Society, 3 (2), 62-82. [1999]

[19] Hans de Heij, *Banknote design for visually impaired*, DNB Occasional studies, Vol7, Number 2, [2007]

[20] Masato Aoba, Tetsuo Kikuchi, Yoshiyasu Takefuji, *Euro Banknote Recognition System Using a Three-layered Perceptron and RBF Networks*, Vol. 44, No. SIG7 (TOM 8) IPSJ Transactions on Mathematical Modeling and Its Application [May 2003]

[21] Nan C. Schaller , *Color Conversion Algorithms,* Computer Science Department, Rochester Institute of Technology [2003]

[22] Philippe Colantoni, *Color Space Transformations,* 2003, http://colantoni.nerim.net/download/colorspacetransform-1.0.pdf

[23] Jae-Kang Lee, Seong-Goo Jeon, Il-Hwan Kim, *Distinctive Point Extraction and Recognition Algorithm for Various Kinds of Euro Banknotes,* International Journal of Control, Automation, and Systems Vol. 2, No. 2, [June 2004]

[24] Guillaume Dave, Xing Chao, Kishore Sriadibhatla, *Face Recognition in Mobile Phones,* Department of Electrical Engineering, Stanford University [2010]

[25] Tarek M. Mahmoud, *A New Fast Skin Color Detection Technique,* World Academy of Science, Engineering and Technology 43 [2008]

[26] Janusz Baran, *Cyfrowe przetwarzanie sygnałów,*

[27] Cezary Bołdak, *Cyfrowe przetwarzanie obrazów*, [2008]

[28] Michael T. Wells, *Mobile Image Processing on the Google Phone with the Android Operating System,* [2009]

[29] EE368 Digital Image Processing, *Tutorial on using Android for Image Processing Projects,* Spring 2010

[30] Bernd Girod *Lectures from image processins,* Stanford University:
http://www.stanford.edu/class/ee368/
(access 2010-2011)

[31] Blog Andorid-er [online]: *Draw a bitmap on view*
http://android-er.blogspot.com/2010/05/draw-bitmap-on-view.html
(access 05-2011)

[32] Android developers: *Developer resources:*
http://developer.android.com/resources/index.html
(access 2011)

[33] Raymond Lau, *Face Detection in Java – Haar Cascade with JJIL (how-to),* Agenda Tech blog[online], [January 2011]:
http://techblog.hk.agenda-asia.com/2011/01/13/face-detection-in-java-%E2%80%93-haar-cascade-with-jjil-how-to-2/
(access 05-2011)

[34] Deftafalcon's blog [online]: *Tutorial: Read/Write Android Emulator sdcard.img in Windows* [April, 2010]
http://deltafalcon.com/2010/04/mounting-an-android-emulator-sd-card-image-in-windows/
(Access 09-2011)

[35] Olly Oechsle, *Finding Straight Lines with the Hough Tranform* [online]
http://vase.essex.ac.uk/software/HoughTransform/
(access 05-2011)