Development of a method that is suitable to guess the material of the objects in a digital image

Author: Juan José Castillo Bueno

Tutor at Home University: Antonio José Albiol Colomer

Cotutor at Destination University: András Horváth

Final Thesis presented at the Széchenyi István Egyetem for the obtainment of the Graduated of Engineering in Telecommunication Systems and Technologies at the Universitat Politècnica de València.

Course 2016-2017

Győr, January 2017

Resumen

El propósito de este Trabajo Fin de Grado es el de presentar una metodología que permita reconocer de manera automática diferentes materiales presentes en las imágenes digitales.

El programa ha sido desarrollado en Python, y abarca seis tipos diferentes de materiales: vidrio, metal, papel, plástico, porcelana y madera. Para detectar el material del que está hecho el objeto se han calculado diversos parámetros RGB y HSV, los cuales han sido analizados estadísticamente y han permitido descubrir características únicas de cada material que lo identifican respecto a otros.

Durante la elaboración del presente trabajo, primero se han tomado imágenes de diferentes materiales utilizando diversos métodos, y posteriormente se han estudiado ampliamente sus parámetros de manera estadística. Una vez escogidos los parámetros más útiles para esta tarea, se han elaborado diferentes métodos de procesado de la información con el fin de comparar dichos parámetros en una nueva imagen con los obtenidos previamente en las imagenes de diferentes materiales, y así poder reconocer de qué material se trata.

Este programa ofrece unos resultados bastante interesantes, aunque cuenta con limitaciones similares a las del ojo humano a la hora de diferenciar entre diversos tipos de materiales.

Resum

El propòsit d'aquest Treball Fi de Grau es el de mostrar una metodologia que permeti reconèixer de manera automàtica diferents materials presents en les imatges digitals.

El programa ha sigut desenvolupat en Python i reconeix sis diferents tipus de materials: vidrie, metall, paper, plàstic, porcellana y fusta. Per a detectar el material del que està fet el objecte, s'han calculat diverses paràmetres RGB i HSV, els quals han sigut analitzats estadísticament i han permès descobrir característiques úniques de cada material que lo identifiquen respecte a altres.

Durant la elaboració del present treball, primer s'han tomat imatges de diferents materials utilitzant diversos mètodes, i posteriorment s'han estudiat àmpliament el seus paràmetres de manera estadística. Una volta triats el paràmetres més útils per a esta tasca, s'han elaborat diferents mètodes de processat de la informació amb el fi de comparar dits paràmetres en una nova imatge amb els obtinguts prèviament en les imatges de diferents materials, i així poder reconèixer de qui material es tracta.

Aquest programa ofereix uns resultats prou interesants, encara que conta amb limitacions similars a les del ull humà a l'hora de diferenciar entre diversos tipus de materials.

Abstract

The aim of this Final Thesis is to present a methodology to recognise automatically different materials present in digital images.

The program has been developed in Python, and considers six different types of materials: glass, metal, paper, plastic, porcelain and wood. In order to find out which material is the object made of, several RGB and HSV parameters have been measured and statistically analysed, so it has been possible to discover unique characteristics of each material that makes them identifiable among different options of materials.

The first task in the elaboration of this project was to take several pictures of various materials using different methods, and widely statistically studying their parameters afterwards. Once the most useful parameters for this assignment have been chosen, different information processing methods have been developed to compare these parameters in a new image with the same ones previously obtained for other images, in order to make possible to know which material it is made of.

This program offers such interesting results, even though it has similar limitations than a human eye when making a difference between diverse types of materials.

Index

Chapter 1. Introduction and aim

1.1 Motivation

This project has been created as part of an ERASMUS semester, and with the collaboration of the Universitat Politècnica de València[1], from Valencia (Spain) as home university, and the Széchenyi István Egyetem[2], from Győr (Hungary) as destination university.

The aim of the developed program is to detect which material is an object made of, by comparing the parameters of a new image with the parameters of different images stored as a databank. First, some interesting parameters are selected and calculated for all the images in the databank, and stored afterwards into an archive. When a new image is taken, these parameters are calculated, and compared to the data stored of the previous pictures. Using different methods, the program exposes the most suitable material.

One possible future application of this project is industry, where a material differentiation method is very useful to accelerate and automatize processes. For example, it can be very attractive for an automotive plant like the Audi's one in Győr[3], due to the variety of materials that conforms a car and the diverse treatments that each object requires. With a program like this, it is possible for the machines to make a fast decision about what to do with a component depending of its material.

Similar applications are extremely helpful in recycling plants, where the core of its work is based on separating objects in different categories and treating each one in a separate way.

During the elaboration of this project, some different comparing methods have been studied, but only two have reported interesting results. The first method is relates to calculating the 2-D or 3-D 'distances' from a new point, characterized by three normalized parameters of a test image as x, y and z axis, to different points stored that corresponds with three parameters of every image stored in the databank. The second method is based on a decision tree and consists in establishing a range of values for every chosen parameter of each material, according to the minimum and maximum values found in the databank, and classifying the material of a new object depending of its affinity with the previously calculated values.

In this project have been studied six types of materials: glass, metal, paper, plastic, porcelain and wood. As have been proved parameters used in this work only for these materials, it is not sure that its results would be congruent for other materials. In such a case, new parameters are needed to adapt the program to the new circumstances.

Furthermore, one basic limitation of this program is that works in a similar way as a human eye: finds patterns in objects made of the same material, and makes a difference depending on the pattern. This leads to confusion when patterns corresponding to a material are present in another one. A common example of this is when trying to analyse the material of a bottle of water, that usually made of plastic but that can show many bright spots depending on the light sources, like a glass-made object. To minimize the cases when this turmoil appears, pictures used in this project have been taken in the same light conditions and with the same camera, in order to avoid resolution and quality problems too.

Additionally, some future proposals are included in this document as a guideline for future improvements of this project, in order to remove the problems above described and to improve the results of the program.

[1] http://www.upv.es/index-en.html
[2] http://uni.sze.hu/en_GB/home
[3] http://www.audi.com/corporate/en/company/production-sites/audi-production-worldwide/gyor.html

1.2 Objectives

The aim of this project is to develop a program using the different libraries that Python offers, capable of making a difference between six different materials in a digital image, by analysing several parameters of each material and comparing them with the parameters of a new image. To achieve this objective, different goals are defined:

- Learn the basics of Python programming as starting point, in order to be able to understand the available tools that this language offers for this project.
- Consider the most interesting libraries of Python for image processing, and the kind of problems that can be solved with each one.
- Study about digital image processing, especially about the topics that can be applied to this project, such as RGB and HSV colour systems, mask definition, histogram representation and morphological operators.
- Define the characteristics that must have the images in order to have accurate results after analysing them.
- Integrate knowledge about digital image processing in Python to analyse the images in different colour systems.
- Statistically analyse the results for each colour system to create a databank of parameters.
- Find parameters with characteristic values for certain materials, so that it would be possible to make a difference between them.
- Study different methods to compare the values for the parameters of a new image to the values for the images in the databank.
- Test the different methods to check the accuracy.
- Improve accuracy as much as possible.
- Value alternatives to improve the efficiency of the program by evaluating other parameters, studying other ways of comparing the values or expanding the databank.
- Detect strengths and deficiencies of the program and the used methods.
- Make proposals to future improvements and applications.

1.3 Structure of the report

The different chapters that conforms this document are explained in details as follows:

- **Chapter 1: Introduction and aim.** This chapter is used to introduce the project, and why and for what it has been developed, including its objectives and the structure of this report.

- **Chapter 2: Basic concepts and libraries of Python.** Here the different modules and libraries of Python are explained in details, as well as the tools and structures used in this project.

- **Chapter 3: Theoretical concepts about digital image processing.** Concepts about digital image processing needed and acquired for the consecution of this work are developed, as well as the theoretical background.

- **Chapter 4: Methodology of work.** Division of tasks and time-scheduling of the work. The work process and different actions and decisions during the elaboration of the project are also included.

- **Chapter 5: Analysis of results.** Evaluation and comparison of the results obtained using different methods.

- **Chapter 6: Conclusions and proposals for future development.** As a conclusion, an understanding of the strengths and deficiencies has been include, as well as a guideline of actions and methods that could be implemented in the future to improve the results of this program.

Chapter 2. Basic concepts and libraries of Python

2.1 Features of Python programming language

Python language has a variety of characteristics that makes it the perfect language for digital image processing.

As it is a high-level language, its syntax is very intuitive, so it is possible to acquire a medium programming level of it very fast. In this language, there is no need of variable declarations, and it is easy to use dynamic objects like lists. Furthermore, it supports modules and packages that expand the fields where Python can be applied.

On the other hand, it is a slower program compared with other languages like C, and takes much more resources from the computer than static languages[1].

In the initial plan of this work, it was not conceived to be applied in time-dependant situations, so time is not a problem in our case. Whether there is a possible application where saving time is important, this program can be migrated to faster languages, but migration would not be an easy task due to the libraries and procedures used in Python.

2.2 Used libraries and data structures

2.2.1. Interesting libraries for digital image processing

Among the infinite variety of libraries that Python can offer, some interesting libraries which have helped in the achievement of this work[2]:

- **Operative system:** imported as *os*, this library allows to operate with the current working path and to change it to others. This has been really useful because of the structure of the folders in the project: a main folder that contains some subfolders for storing the databank parameters, the databank images, the resulting graphs of the analysis, the histograms of each parameter of the databank and the new test images. In this way, it is possible to move from one folder to another to read or write information using a simple instruction:

  os.getcwd[4] *():*

  Tthis instruction gives back the current working directory, so that it can be stored in a variable to go back to it in the future.

  os.chdir[4] *(dir):*

  Using this instruction, the current working directory can be changed, in order to take needed data from another directory, or to write data on it.

[4] https://docs.python.org/2/library/os.html

- **OpenCV 3.1[3]:** library oriented to digital image processing which includes several options to treat the images. Some instructions used in this project are:

  cv2.imread(image)[5]*:*

  To read an image from the working directory, specifying the complete name of the image as a string. This image must be stored in a variable that will contain three matrixes of values, each matrix corresponding to the values of blue, green or red of the image. Every value in a matrix will correspond to the amount of blue, green or red in a pixel of the image. It is important to point out that the matrixes of an image read by this instruction are sorted inversely of the traditional RGB order, so the first matrix corresponds to the blue channel, while the last one refers to the values of the red channel.

  cv2.imwrite(name, variable) [5]:

  To save an image in the working directory, specifying the desired name of the image, and the name of the variable which contains the image.

  cv2.calcHist[6]*(images, channels, mask, size, ranges):*

  This is one of the most relevant instructions used for this project, as it produces the histogram of the selected channel of an image and supports multiple options.

  The first field of the instruction refers to the image or images that will be calculated. This instruction allows the calculation of many histograms from different images at the same time, and the result would be a list of histograms corresponding to each image.

  The second field allows the selection of the desired channels to analyse. For example, if the image is defined in the RGB colour system, it is possible to choose one or more RGB channels to be analysed, resulting in a list of histograms or a single histogram for each image. Whether many images and channels are selected, the results will be sorted in lists of lists of histograms, in other words, lists of histograms of the different channels inside a list of histograms for each image.

  The third field is optional, and enables the use of a mask for selecting the area of the image to analyse. This fact is really useful to obtain histograms of an object in an image, excluding the background.

  The last two fields set the number of values that will contain the histogram, and the range of values that it will cover. In our case, the size of the histogram is always 256 values, from 0 to 255, corresponding to the natural range of values in RGB colour system.

  cv2.putText(image, text, org, fontFace, fontScale, color, thickness, lineType, bottomLeftOrigin)[7]:

  This instruction is used to write text in images, and it has been used to write some measures in the images of the histograms. The fields to specify are the image where it will write, the text to write, the location of the bottom-left corner of the text string in the image, the cv2 font of the text letters (in our case, *cv2.FONT_HERSHEY_SIMPLEX*),

[5] http://docs.opencv.org/3.0-beta/modules/imgcodecs/doc/reading_and_writing_images.html

[6] http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html

[7] http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

the font scale factor that is multiplied by the base size of the font (in our case, always 0.4), the text color (in RGB), the thickness of the lines (always 1 in this program) and the line type (cv2.LINE_AA). The last field expects a flag, and when true, the image data origin is at the bottom-left corner. In our case it is disabled by defect, so the origin will begin at the top-left corner.

cv2.getStructuringElement(shape, size)[8]*:*

This instruction creates a kernel that can be used in morphological operations. The first field consists on the desired shape for the kernel, and it can have a rectangular shape (*cv2.MORPH_RECT*), an ellipse (*cv2.MORPH_ELLIPSE*) or cross-shaped (*cv2.MORPH_CROSS*).

The second field is the size of the kernel, as a two-dimensional tuple.

For a cross-shaped kernel with a 3x3 size, we would obtain:

```
>>> kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))
>>> print kernel
[[0 1 0]
 [1 1 1]
 [0 1 0]]
```

Code 1 - Creation of a kernel

cv2.dilate(image, kernel)[8]*:*

This instruction applies a dilation to the image, using the specified kernel. This operation basically enlarges the black areas of the picture.

cv2.erode(image, kernel)[8]*:*

In a similar way to the previous instruction, this applies an erosion to the image with a certain kernel. This operation is the opposite of dilation, and enlarges the white areas of the picture.

cv2.morphologyEx(image, function, kernel)[8]*:*

This is a more complex instruction that applies different morphological transformations, depending of the selected function. Some of the available functions are the following:

- *cv2.MORPH_OPEN:* applies an opening operation, that consists on an erosion followed by a dilation. Its main application is to reduce the number of isolated white points in an image.
- *cv2.MORPH_CLOSE:* applies a closing operation, that is reverse of opening. It applies a dilation followed by an erosion, to reduce the number of isolated black points in an image.

[8]http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

- *cv2.MORPH_GRADIENT:* calculates the morphological gradient of an image, that consists on the difference between dilation and erosion. Its value depends on the differences between neighbour pixels.
- *cv2.MORPH_TOPHAT:* calculates the difference between the image and the opening of the image.
- *cv2.MORPH_BLACKHAT:* calculates the difference between the closing of the image and the image.

Only the morphological gradient has been used during the elaboration of the program.

cv2.cvtColor(input_image, output_image, code, number_of_channels)[9]*:*

This instruction converts one image from one colour system to another. The first two fields are the input and output images, that must have the same size, while the third field is the color space conversion code, that depends on the systems of the conversion. Here are some of the available codes that can be used for different conversions:

- *cv2.COLOR_BGR2GRAY:* converts the image from BGR system to grayscale.
- *cv2*.COLOR_GRAY2BGR: converts the image from grayscale to BGR colour system.
- *cv2.COLOR_BGR2HSV:* converts the image from BGR system to HSV system.
- *cv2.COLOR_HSV2BGR:* converts the image from HSV system to BGR system.

- **NumPy[4]:** this *Numerical Python* library contains powerful mathematical tools that are very useful for our purpose. Among the instructions that it offers, the following ones have been used in this project:

numpy.asarray (input, dtype = None, order = None).astype(dtype, order, casting, subok, copy)[10]*:*

The instruction *numpy.asarray* converts the input data to an array. The second and third fields represents the data type and the type of memory representation, but they will be empty in our case.

It is possible to add *.astype* at the end of the instruction to specify the type of data that the array will manage, for example, unsigned integers of 8 bits (*numpy.uint8*) or 32 bits float (*numpy.float32*).

numpy.convolve (N, M, mode = 'full')[11]*:*

This instruction implements the convolution of two arrays with the same dimensions, using one of the modes that this instruction manages:

- *Full:* mode by default, it returns an output with a (N+M-1) shape. The signals do not overlap completely at the end-points, so the boundary effects can be seen.
- *Same:* it returns an output with a length equal to the maximum of N and M, and boundary effects are visible.

[9] http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html
[10] https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.astype.html
[11] https://docs.scipy.org/doc/numpy/reference/generated/numpy.convolve.html

- *Valid:* it returns an output with a length equal to the maximum between N and M, minus the minimum of both plus one. The convolution product is given by the points where there is a complete overlap of the signals, so values outside the signal boundary have no effect.

numpy.dot(a,b)[12]*:*

Returns the dot product of two N-dimensional arrays.

numpy.power(a,b)[13]*:*

Calculates the power of the elements of the first array, raised to the elements of the second array. The first array is made up of the bases, and the second one of the exponents.

numpy.append(array, values, axis = None)[14]*:*

Append the values in the second field to the end of the array of the first field. The axis where values are appended can be set.

numpy.zeros(shape, dtype, order)[15]*:*

Creates an array filled with zeros, and with the specified shape.

numpy.concatenate((a1,a2,…), axis = 0)[16]*:*

Joins various arrays along the specified axis.

numpy.average(a, axis = None, weights = None, returned = False)[17]*:*

Calculates the weighted average along the specified axis. In the calculations of this work, all the average calculations have not been weighted.

numpy.median(a, axis = None, weights = None)[18]*:*

Calculates the median along the specified axis.

numpy.var(a, axis = None)[19]*:*

Calculates the variance along the specified axis.

[12] https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html
[13] https://docs.scipy.org/doc/numpy/reference/generated/numpy.power.html
[14] https://docs.scipy.org/doc/numpy/reference/generated/numpy.append.html
[15] https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html
[16] https://docs.scipy.org/doc/numpy/reference/generated/numpy.concatenate.html
[17] https://docs.scipy.org/doc/numpy/reference/generated/numpy.average.html
[18] https://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.median.html
[19] https://docs.scipy.org/doc/numpy/reference/generated/numpy.var.html

numpy.ndarray.shape(array)[20]*:*

Returns a tuple with the dimensions of the array.

- **Openpyxl[5]:** library that makes possible to read and write Excel 2010 files.

  Using this library, it has been build a data structure with all the parameters of each material, and stored in a Workbook while using them. After writing all the values into the Workbook, it has been saved into an Excel file.

  To store a value in a cell of an Excel sheet, it is as easy as saving the value into the position in the Workbook (*ws*) corresponding to the desired letter and number.

  It is showed in the following image the process for saving the name of the material, the name of a histogram and different parameters into different cells, where *row* is the number of the selected row.

```
"""Writes data in a Workbook"""
ws['A'+'%d' % row]=mat
ws['B'+'%d' % row]=hist
ws['C'+'%d' % row]=av
ws['D'+'%d' % row]=med
ws['E'+'%d' % row]=var
ws['F'+'%d' % row]=std
ws['G'+'%d' % row]=rd
ws['H'+'%d' % row]=cm
ws['I'+'%d' % row]=sk
ws['J'+'%d' % row]=sp
ws['K'+'%d' % row]=p90
ws['L'+'%d' % row]=p95
```

Code 2 - Filling an Excel sheet

- **Pickle[2]:** *Pickle* module is capable to convert an object hierarchy into a byte stream, and vice versa. This module has been used to store a *Dictionary* structure into a *.p* file, using the following instruction:

  pickle.dump (object, file [, protocol])[21]*:*

  Writes the object into an open file, with a determined protocol. In this project, only *'wb'* protocol has been used to open the file for writing in binary mode.

  pickle.load (object, file [, protocol]) [21]*:*

  Loads the object from an open file, with a determined protocol. Only 'rb' protocol has been used in this program, to open the file for reading in binary mode.

- **Matplotlib[6]:** this library is used for 2D plotting, and produces a great variety of quality figures. The instructions from this library that have been used are the following.

[20] https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html
[21] https://docs.python.org/2/library/pickle.html

matplotlib.pyplot.clf()[22]*:*

Clears the current figure.

*matplotlib.pyplot.plot(*args, **kwargs)*[22]*:*

Plots lines or markers using pairs of x, y arguments, and also supports different options like labels and different line and marker styles.

*matplotlib.pyplot.legend(*args, **kwargs)*[22]*:*

Places a legend on the axes with different options like the location or the font size.

*matplotlib.pyplot.axis(*args, **kwargs)*[22]*:*

Set properties for the different axis, with various available options like '*equal',* to have the same length in both axis, or establishing the minimum and maximum values for each one.

*matplotlib.pyplot.xlabel(s, *args, **kwargs)* and *matplotlib.pyplot.ylabel(s, *args, **kwargs)*[22]*:*

Set the x or y axis label of the current axis, with different options available.

*matplotlib.pyplot.savefig(*args, **kwargs)*[22]*:*

Save the current figure into a *.png* file, with the specified name.

2.2.2. Structure of the databank

The values of the different parameters must be arranged and stored so that they can be easily identified and used for new calculations. For that purpose, two files have been created with different objectives:

- **Excel sheet**: in order that users can consult easily the values of each parameter, all the data is stored in an Excel sheet. Each value takes one cell depending on the parameter that it represents, the concrete channel and the image from which it was calculated. This is an example of some parameters of the 'Glass1.jpg' image.

[22] http://matplotlib.org/api/pyplot_api.html

	A	B	C	D	E
1	Material	Histogram	Average	Median	Variance
2					
3	Glass1	Blue Chan	93,55384	96	480,0664
4	Glass1	Green Cha	112,3348	115	636,3937
5	Glass1	Red Chanr	121,4618	125	773,1752
6	Glass1	RGB histo	365,1246	371	49218,08
7	Glass1	Grey histc	112,9153	116	653,8601
8	Glass1	Hue	19,61369	20	39,18858
9	Glass1	Saturatior	59,58961	59	88,06741
10	Glass1	Value	121,463	125	772,7856

Illustration 1 - Structure inside an Excel sheet

- **Data.p file:** as an Excel sheet is sometimes hard to manage when the number of the cells increase, and due to the need of remembering the letter and number where each parameter is located, another file has been created. This kind of file is easy to manage, as it is filled in with a dictionary structure, native of Python, that works with very intuitive indexes. When working with dictionaries there is no need to memorize cell position, but to sort data with easy names.

  Data is now indexed using identifiers that classify each value by the name of the parameter, the channel and the image where it comes from. This is an example of storing the average of the grayscale histogram of an image:

```
if flag==False:
    Data['Gray'].update({'Average':hgray_average})
else:
    Data['Gray']['Average']=np.append(Data['Gray']['Average'],hgray_average)
```

Code 3 - Saving data into a dictionary

The flag shows if the dictionary has some previous value or not. If it is empty, the first position in *'Average'* inside *'Gray'* would be filled in with the value of the average. If there is already a value in that position, the new value would be appended afterwards.

After filling the dictionary with all the calculated values, the data structure can be saved in a *Data.p* file using *pickle.dump*, and can be read back at any time using *pickle.load*.

Chapter 3. Theoretical concepts about digital image processing

This chapter covers the theoretical information needed to understand the development of the project, and the decisions and actions that have been taken during its elaboration, from the first beginning until the conclusions after the last results.

3.1. Discretization principle

Real images are composed of different physical variables that depends on different continuous variables:

$$\boldsymbol{I(\theta, \varphi, \lambda, t)} \quad \textbf{[1]} \qquad \textbf{(1)}$$

The visual information of an image depends on the wavelength of the incoming light (λ), the direction of the light in relation to the position of a human observer (θ, φ), and the time that takes to a human to correctly process an image.

Human light sensors do not have an homogeneous density along the eye, which changes depending on the **direction** of the incoming light. The best resolution is obtained when the light from the object comes from the centre of our field of view, of 60 lines per degree. This means that an average human can make a difference between two lines separated 1/60 degrees in the central region of his field of view.

Furthermore, our eyes can only detect three colour channels, corresponding to red, green and blue colours depending on the **wavelength** of the light.

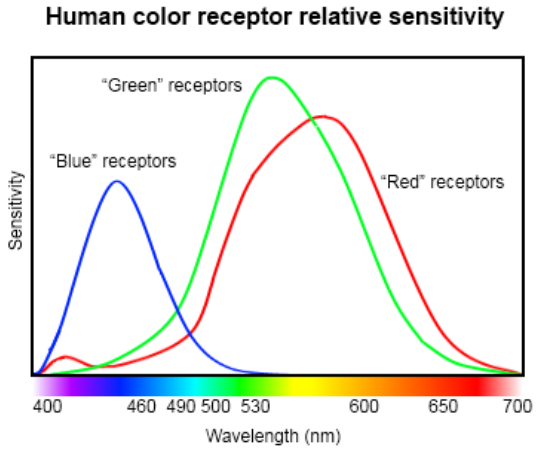


Illustration 2 - Human sensitivity for different colours

As we can see in the picture, human eyes have three types of light-detection sensors or cone cells corresponding to a different colour. Receptors of blue respond better to frequencies between 420 and 440nm, while receptors of green have a better response between 534 and 545nm and receptors of red, between 564 and 580nm[7].

Finally, the **time** that takes to a human observer to correctly perceive an image varies from 0.05 seconds to 0.1 seconds, so 10 to 20 images can be perceived in a second by an average person.

We can conclude that our brain processes discrete images, and fills the gaps between one image to the next one, and between discretization points.

Computers, on the other hand, can only work with discrete values, so will be needed an image discretization that lets us work with it. This procedure consists on taking a pixel grid, where each pixel has a unique value corresponding to the value of the centre of the cell, the average value of the cell or the average value of a subset of the cell.

To reproduce the spectrum of the light in a digital image, it is needed to replicate the work principle of the eye sensors. For a human brain, a colour is a mixture of three different wavelengths, corresponding to red, green and blue, that depends on the spectral sensitivity function of each colour channel. In computing, every pixel will have an 8-bit unsigned integer value for each RGB channels, which result in a mixed colour per pixel and a value from 0 to 255 per channel.

For this reason, RGB is the most famous colour system in digital image processing, but not the only existing one. There are more colour system depending on the number of channels, like in the cameras used in satellites to predict weather forecast.

Videos are formed by a sequence of **frames**, images taken at specific time-steps (frame rate) during a certain period of time (exposure time).

When deciding which format to use to store a digital image in a computer, it is important to take in account that a high compression needs more computing time, and that an optimal compression method is image-dependant. A lossless format that offers a high compression, or a slow compression for large images, is PNG, and it has been used in different parts of this project.

3.2. Basics of colour theory

The idea behind colour theory is to build a set of base colours, and combine them to produce different mixed colours. This mixing can be additive, when the spectra of base colours is weighted and added together, or subtractive, when the different spectra of base colours are subtracted when together[1].

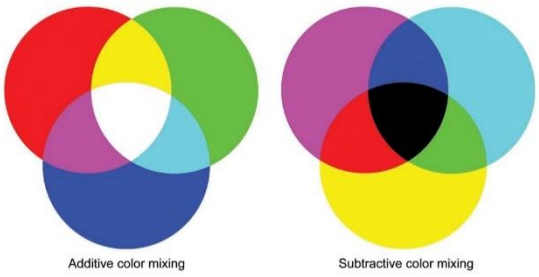


Illustration 3 - Additive and subtractive colour mixing

3.2.1. RGB system

RGB system is an example of additive colour system, but it can't reproduce all the existing colours [1]:

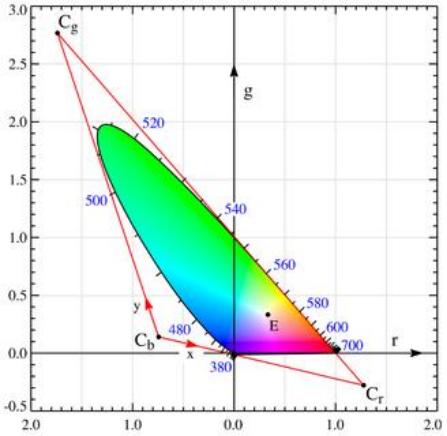


Illustration 4 - RGB chromaticity

In this graph, we can see that not all the colours are reproducible due to the rounding of negative values for red, green and blue to zero.

This lead to the conclusion that many colours in the pictures are not the exact colours of a real and non-digital image, but an approximation to them.

3.2.2. XYZ colour system

XYZ-system is an additive system that can be calculated from RGB in a simple way[1]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

Similarly to RGB, all the values must be non-negative in XYZ.

Furthermore, Y component is proportional to the brightness:

$$Y = 0.2126R + 0.7152G + 0.0722B \quad (3)$$

The next image shows the chromaticity diagram with the reproducible colour of the XYZ-system. Note that the perceived colours from the following picture may not correspond exactly with the XYZ colours, as it depends on the computer screen where it is reproduced, or the printer in case of paper.

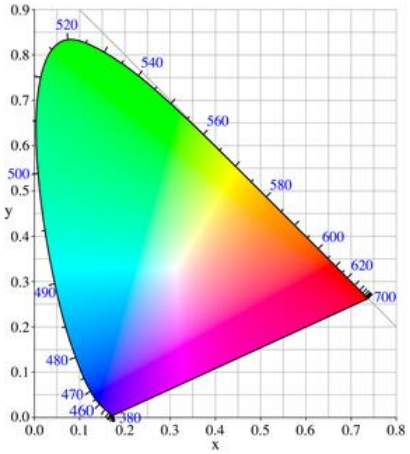


Illustration 5 - XYZ chromaticity

3.2.3. HSV-system

HSV-system consists in three channels: hue, saturation and value.

- **Hue (H) :** comparison of the actual colour and the most similar pure colour
- **Saturation (S):** vividness of the colour
- **Value (V):** brightness of the colour

The conversion from RGB colour system to HSV is as follows [1]:

$$H = \begin{cases} 0\,, & if\ Max = Min \\ 60º \cdot \frac{G-B}{Max-Min} mod\ 360º, & if\ Max = R \\ 60º \cdot \frac{B-R}{Max-Min} + 120º, & if\ Max = G \\ 60º \cdot \frac{R-G}{Max-Min} + 240º, & if\ Max = B \end{cases} \tag{4}$$

$$S = \begin{cases} 0, & if\ Max = 0 \\ \frac{Max-Min}{Max} = 1 - \frac{Min}{Max}, & in\ other\ case \end{cases}$$

$$V = Max/255$$

Where $Max = \max(R, G, B)$ and $Min = \min(R, G, B)$

From these equations, we can conclude that the ranges for the HSV coordinates will be:

- **Hue :** from 0º to 360º
- **Saturation:** from 0% to 100%
- **Value:** from 0% to 100%

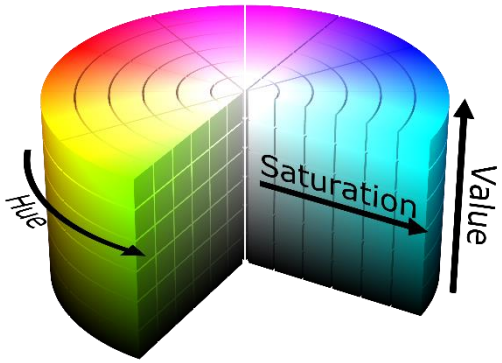


Illustration 6 - HSV coordinates

Many graphics system use HSV-coordinates, but scaling their values into [0,100], [0,255] or [0,180]. In our case, HSV-coordinates will have values between 0 to 255 in Python.

There are lots of colour systems, each one defined for a different purpose, but we will focus in this project in RGB and HSV systems, to study different parameters of the images.

3.3. Usage of histograms

Histogram is a diagram that shows how the numerical values of a set are distributed in a range. This is very useful when analysing diverse parameters of a colour channel, such as red, green, blue, the three at once or one of the coordinates of HSV system.

For example, for the following image, which contains very bright points and a few dark areas, the histogram for RGB components would be the next:

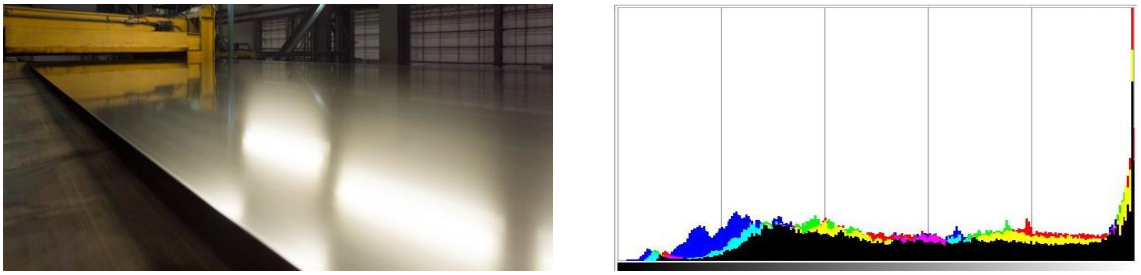


Illustration 7 - Steely material with bright areas and its RGB histogram

As we can see in the images above, there are many bright points that appears in the histogram as a strong pike in the right side (bright pixels). On the other hand, there is a wide range of colours that are represented in the histogram as a plateau in the central values.

As there are only a few dark points, and not completely black, the left side of the histogram (dark pixels) have very few values.

In a well-balanced image, with not very bright or dark pixels, the histogram would not have strong pikes at the left or right side.

During the elaboration of this project, RGB, grayscale and HSV histograms have been obtained and analysed.

To select only the pixels corresponding to an object, and not the background, a mask can be created using Python or GIMP. A **mask** is a map of zeroes and ones corresponding to the pixels from the object that are desired to analyse, and the pixels from the background that would be discarded.

3.4. GIMP

GIMP[23] is a GNU image manipulation program that is widely used in digital image processing. During the elaboration of this project, GIMP 2 has been used for multiple purposes.

The main window that appears when starting GIMP is the following.

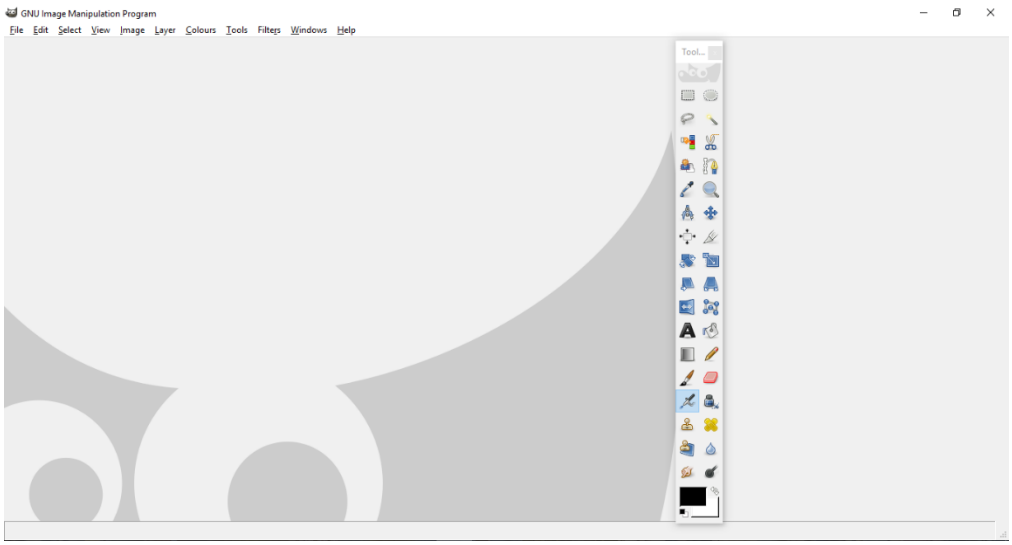


Illustration 8 - GIMP interface

A variety of features can be found in the toolbox, that are very helpful to perform many actions in digital image processing.

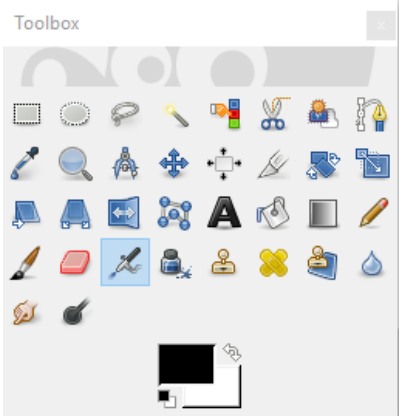


Illustration 9 - GIMP Toolbok

[23] https://www.gimp.org/

The features from the toolbox that have been more used in this project are described next:

- *Free select tool*: lets to select the contour of an object when creating a mask.
- *Colour picker tool*: to identify the value of a pixel in different colour systems.
- *Bucket fill tool*: to paint all the pixels in an area with the same colour, like completely white or black colour for making a mask.

In addition to these tools, histograms for RGB and HSV systems have been also generated with GIMP at the beginning of the project. The details of the generation process will be discussed later.

Chapter 4. Methodology of work

This chapter covers the entire development process of this project, as well as the actions and decisions taken and the time schedule of the tasks. Moreover, the entire status of the project in each timeline point during its elaboration is explained in details.

4.1. Temporal distribution of tasks

The temporal distribution of this project is divided into weekly goals, as part of longer-term goals. Each week, results are checked and a goal for the next week is set, depending of the evolution of the project. Long-term goals are also checked weekly, and modified whether necessary. The weekly distribution of task has been as appears in the Gantt chart.

The long-term goals set at the beginning of September were the following:

1. Learn the needed knowledge about image processing during the month of September
2. Learn the basics of Python programming language during the month of September
3. Find good parameters for the selected images since October to December, and build an easy-to-manage databank
4. Find effective methods to compare the chosen parameters from a sample image to the ones from images in the databank, since December to the beginning of January.
5. Make conclusions about the work done, generate a report and prepare the slide presentation.

	SEPTEMBER					**OCTOBER**				**NOVEMBER**				**DECEMBER**				**JANUARY**			
Week			3	4	5	1	2	3	4	1	2	3	4	1	2			1	2	3	
Previous tasks			**X**	**X**	**X**																
Study the Python tutorial			**X**	**X**																	
Learn the basics of DIP			**X**	**X**	**X**																
Learn to use GIMP				**X**	**X**																
Analysis of parameters			**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**			**X**	**X**		
Obtain images of materials			X				X				X	X	X								
Produce histograms with GIMP			X																		
Produce histograms with Python				X	X	X	X	X	X	X	X	X	X					X	X		
Evaluate parameters of images				X	X	X	X	X	X	X	X	X	X	X	X			X	X		
Create a mask for each image					X		X				X	X	X								
Management of the databank						**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**				**X**	X		
Create an Excel file						X	X			X	X	X	X					X	X		
Plot parameters in 2D							X	X	X	X	X	X	X					X	X		
Split code									X					X							
Create a .p file									X	X	X	X	X					X	X		
Apply smoothness to histograms										X	X	X	X					X	X		
Elaborate decision-making methods														**X**	**X**				**X**		
Check the results																				**X**	
Write the report																				**X**	**X**

Table 1 - Gantt chart with the planification of tasks

As we can see, some tasks have repeatedly appear during the work. A description of each task is now introduced to the reader:

- **Previous tasks:** these objectives are previous to the elaboration of the program, and consists on the needed tools to obtain, interpret and use image data after the first weeks.
    - *Studying the Python tutorial:* in order to learn the basics about Python programming, and to be able to produce solid codes in this language. The tutorial from Python webpage[7] covers very interesting contents, like vectorization and dictionaries.
    - *Learn the basics about digital image processing:* basic concepts of digital image processing are needed to understand the diverse tools that are used in this project, and to be able to identify good parameters from an object.
    - *Learn how to use GIMP:* GIMP is an image manipulation program that is quite helpful, especially at the first stages of this work, to create masks and histograms of images.

- **Analysis of parameters**: these tasks covers the process from taking images and producing histograms, to evaluate parameters of those histograms in order to identify interesting values that can help in the material recognition.
    - *Obtain images of materials*: in certain occasions, new images must be taken as part of a problem solution, to have pictures with other conditions or light sources. Images used along the project elaboration come from different sources, such as cameras and webpages.
    - *Produce histograms with GIMP:* at the first week, when knowledge about Python was not proficient, histograms were taken using GIMP. These histograms gave information about RGB and HSV colour systems.
    - *Produce histograms with Python:* after the first week, all the histograms were taken using Python, and it made possible the calculation of different parameters and a wider range of options, such as writing text in images and smoothing.
    - *Evaluate parameters of images:* this task has been a constant during this project, since the real challenge is to find the clue parameters that allow to make a difference between materials. It refers to parameters values checking and comparing using different techniques.
    - *Create a mask for each image:* each time a new image is used, a mask is needed. A mask is basically a map of zeroes and ones that informs the *cv2.calcHist* instruction what pixels has to take in account when making a histogram. Pixels with a zero value are selected by hand using GIMP from the images, and are interpreted by Opencv as background.

- **Management of the databank:** these tasks are related with the storing, plotting and processing of the values in a databank.
    - *Create an Excel file*: this file acts like a reference where stored values of different parameters from every image can be consulted. The disadvantage of this kind of files is its difficulty to identify cells using Python, due to the need of remembering the row and column where the parameter to use is located.

- *Plot parameters in 2-dimensions:* using *matplotlib*, it is very easy to make graphs of pairs of parameters. First, interesting pairs of parameters must be picked, in order to obtain nice results. After the graph is produced, it is possible to analyse data very intuitively.
- *Split code:* after weeks of using a Python code file, it becomes too large to manage it efficiently. To make work easier, it is advisable to split that file into two or more codes, each one for a different purpose.

  Final versions of the project consist at least in three different Python files, for generating and analysing histograms and saving the data into files; to read from those data files and plot the information in graphs, and to identify the material of an object by comparing different data values.
- *Create a .p file:* this file contains a data structure with the different values for each parameter, but it is not understandable by humans. On the contrary, Python can read from it and overwrite it very easily, with no need to remember difficult locations as for Excel files.
- *Apply smoothness to histograms:* in order to avoid rounding errors and strange results for some parameters.
-

- **Elaborate decision-making methods:** write Python code that can decide the material that an object is made of. Various decision-making methods have been implemented, giving different results.

- **Write the report:** condense the information related with the elaboration and results of this project in a written document, explaining each part in details.

4.2. Development of the project

In this part of the chapter, the elaboration process of the project will be introduced and exposed in details.

4.2.1. First steps

The work about the project started at the third week of September, without previous knowledge about digital image processing nor Python programming language. For this reason, the first task was to study both of them to achieve a level enough to allow to produce great results in this project.

The first step was to work on the free Python tutorial of the developer's webpage, covering the first five units during the first week of study.

Additionally, despite the fact that the course *'Introduction to digital image processing'* was running in the same semester, results in the project would be achieved as soon as possible. During the first week, basic competences about colour systems and histogram obtention were practised using GIMP.

As an example, the process to obtain the RGB histogram of the image of a wooden object is shown hereunder:

The first step is to open the file with the image, choosing *'File'* in the upper bar of the main window.

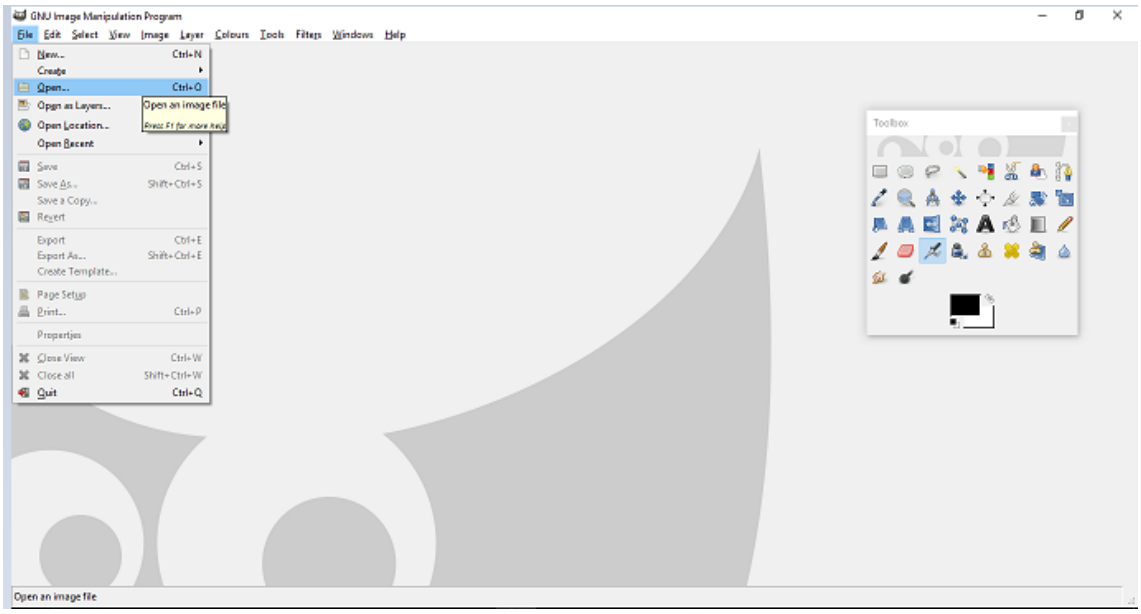


Illustration 10 - Opening an image with GIMP

After opening the file, the selected image will appear in the main window, ready to be processed.

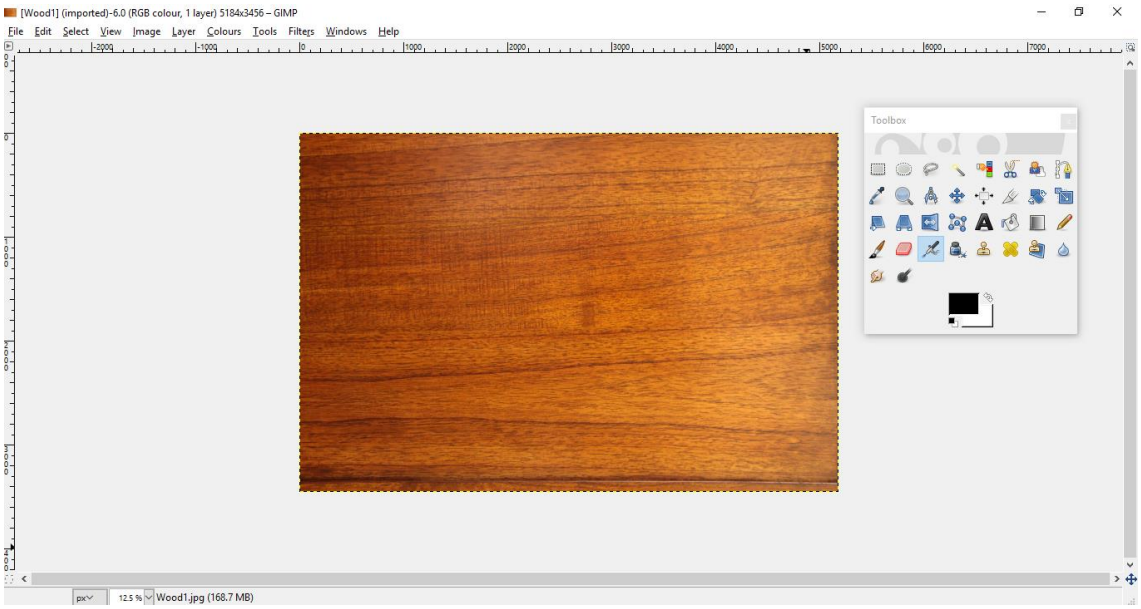


Illustration 11 - Image loaded in GIMP

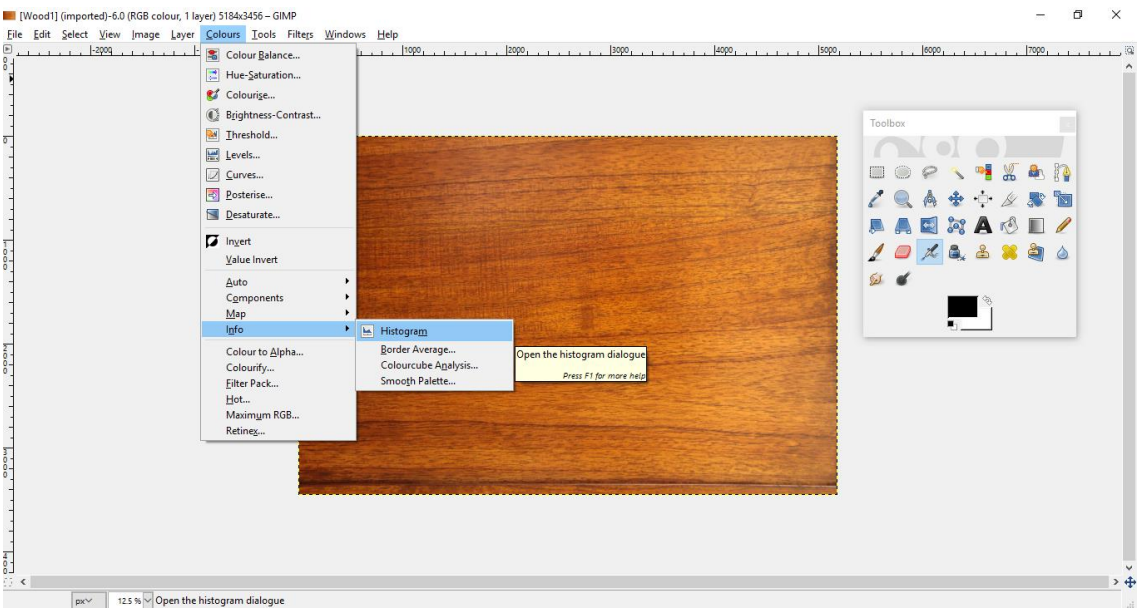


Illustration 12 - Making a histogram with GIMP

It is possible to select the channel of RGB system to be represented in the histogram, or the three at once, in the drop list *Channel*. The following image corresponds to the histogram of the three RGB channels from the previous image.

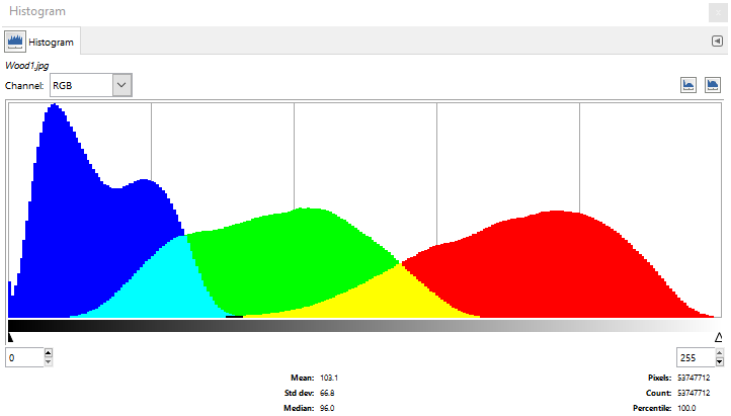


Illustration 13 - RGB histogram in GIMP

To get a representation in HSV systems, the image have to be converted to that colour system first, clicking in *'Colours'* in the upper bar, and then selecting *'Components'* and *'Decompose'* in the drop-down menu. In the following window, we must select *HSV* in the pull-down list, and click *OK*.

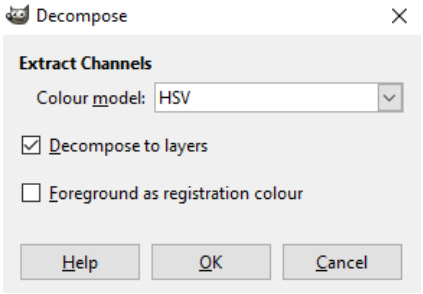


Illustration 14 - Decompose RGB image in HSV layers

This will convert the image to HSV-system, and opening the new image in a new window. This image is composed by three layers, each one corresponding to a HSV coordinate.

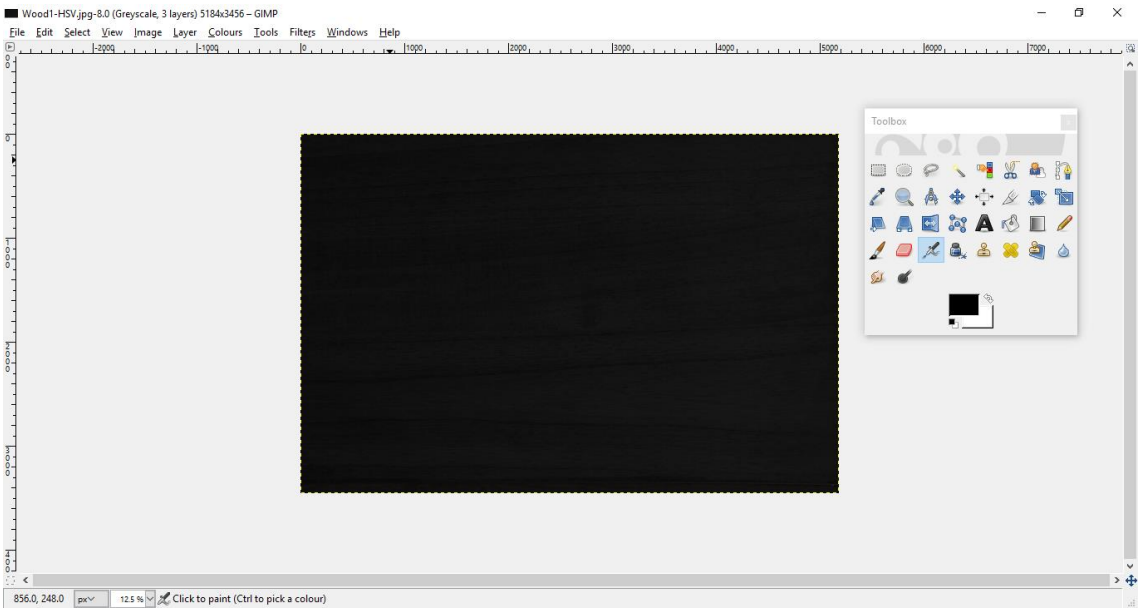


Illustration 15 - Image converted to HSV

The process to produce an HSV histogram is identical the previous explained for RGB. The histogram that now appears on the screen is a mixture of the histograms for hue, saturation and value, because the three layers are active. To select only one layer, first click on *'Windows'*, and then *'Dockable Dialogues'* and *'Layers'*. The following window will appear, showing the different layers that compound the HSV image.

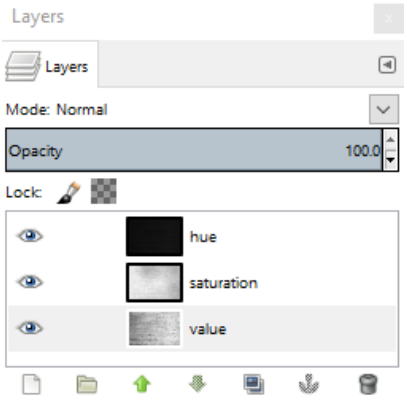


Illustration 16 - HSV coordinates in layers

By picking a layer, the histogram will change to the selected parameter. The next picture shows the histogram of hue for the image of wood, that presents a spike because the image doesn't have strong colour variations.

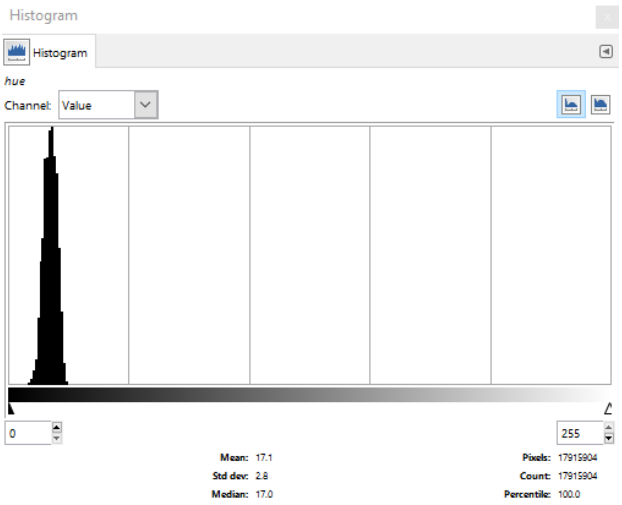


Illustration 17 - Histogram of hue

Using this method to take histograms, the first differences between materials appeared. At that time, only four different materials were analysed, wood, steel, glass and plastic, attending to each one's histogram shape. Analysed images were taken from the internet.

For **RGB histograms**, wood showed a hill for each colour channel with overlap, and which heighs depend on the photography.

Steel, on the contrary, has a great dependence of the light sources, and its histogram is compound by pikes of the three colours in the same region for strong light sources, or by a wide and high hill of red, green and blue in the centre of the spectrum.

Glass presents a similar case than steel, as it depend on the light source. For smooth and non-coloured glass, a great spike appear in the white colour area, while some spikes for different colours and very dispersed values appear for rough and coloured glass objects.

Histograms for plastic objects depend on the kind of plastic in the object. For thin plastic there are small separated hills for each colour, while for rigid plastic, there are spikes and hills for different colours, depending on the colour of the object.

For **HSV histograms**, there are no general patterns that can be detected only looking at them, or results are very noisy that it is impossible to distinguish a contour.

Here are some examples of the histograms taken. The first one corresponds to the RGB histogram of an object made with wood, while the second one represents the hue of a object of steel.

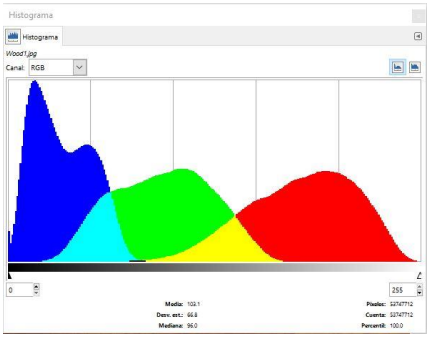


Illustration 19 - RGB histogram of a wooden object

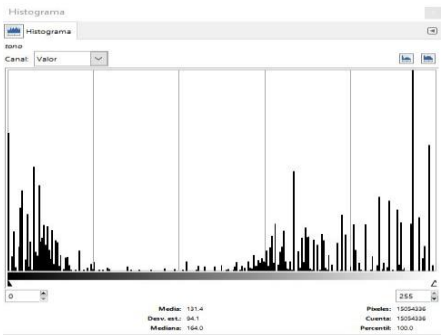


Illustration 18 - HSV histogram of a steely object

This first results are an evidence of the need of a more precise method to produce and analyse histograms. Thus, all the histograms taken after the second week are produced using Python code.

4.2.2. Starting with Python

As the tools that GIMP can offer to process histograms are very limited, successive histograms are produced and analysed using Python and its libraries *NumPy* and *Opencv*.

The most important instruction used to produce a histogram is *cv2.calcHist*, written like in the example:

```
hB=cv2.calcHist([img],[0], None, [256], [0,256])
```

Code 4 - Calculating a histogram using OpenCV

In the example, the histogram of the blue channel of an image is stored in a variable. The first field of the instruction contains the name of the image to analyse, while the second field refers to the first coordinate of the BGR system, that is blue in this case. No mask has been used in this first histogram, that contains 256 values in a range between 0 and 255.

The first histograms produced with Python code described the values for RGB channels separated or joint in a unique array of the triple of length, and also for grayscale and HSV system. For each histogram, some statistical parameters are measured:

- **Average:** the statistical average is defined as the value inside a range, in our case from 0 to 255, that represents the central tendency either of a probability distribution or of a randon variable characterized by that distribution [9]. It refers to the average value of the pixels.

$$\mu = \frac{h[i] \cdot i}{\sum h[i]} \tag{5}$$

- **Median:** also called 50th percentile, it is the value inside the range from 0 to 255, that separates the higher half of the pixel values from the lower half. For getting the median of a histogram, it is needed to calculate the cumulative histogram, in order to find the value between 0 and 255 where the number of pixels is equal or higher to the half of the total amount of pixels.

  A cumulative histogram is the result of creating a histogram where each position from 0 to 255 has the value of the original histogram in the same position, plus all the values in previous positions. The position of the histogram where the value is equal or higher than the half of the number of pixels (or 0.5 if we divide the value in every position by the sum of all the values in the other positions), corresponds to the median of the histogram.

- **Variance:** measures the how much are the pixels spread out from the average value.

$$\sigma^2 = \sum_{i=0}^{255} \frac{h[i] \cdot (i-\mu)^2}{\sum h[i]} \tag{6}$$

  For four images of every type of material, calculations were made to their histograms of red, green, blue, RGB, grayscale, hue, saturation and value, and the results were printed in the same image of each histogram. An example is the following histogram of the blue channel of a steely object.

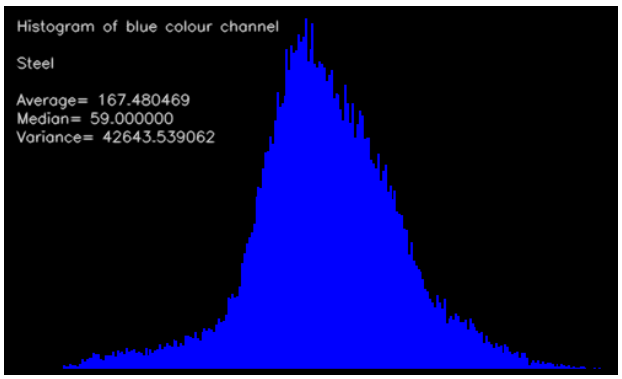


Illustration 20 - Histogram of blue of a steely object with some calculations

As we can see, every image contained the information about the average, median and variance of the histogram. In the image above there was a mistake in the calculation of the variance, that was solved in later modifications of the code.

4.2.3. Deeper analysis of histograms

Parameters presented in previous paragrahps were calculated for the entire image, including the background. Thus, those calculations were incorrect, and the need of a mask to analyse only the object is showed.

Illustration 21 - An image of some plastic objects

In the example image, only one plastic object must be analysed. Otherwise, the results would be inaccurate. To create a mask for only one object, the process is as follows:

First, select the *'Free select tool'* in the toolbox.

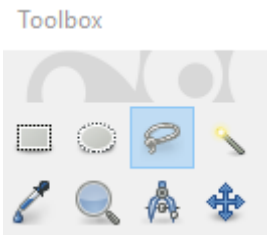


Illustration 22 - Location of the free select tool in the toolbox

With this tool selected, make click and move the mouse inside the desired object, following its external contour, to select the object, leaving a space to the border of the object in order not to

take pixels from the background accidentaly. A dashed line has appeared in the contour to show that the object has been selected.

Illustration 23 - Dashed line following the contour of the selected area

Now, to select every pixel in the image from outside of the selected area, press *CONTROL + I*, or click on *'Select'*, *'Invert'* in the upper bar. Now, a dashed line has appear in the borders of the image, showing that the entire image without the chosen object is selected.

Illustration 24 - Selection of the background

To create the mask, pick up a completely white colour by making click in the coloured rectangle in the bottom of the toolbox, and writing 255 in the R, G and B textboxes that appear in the new window.

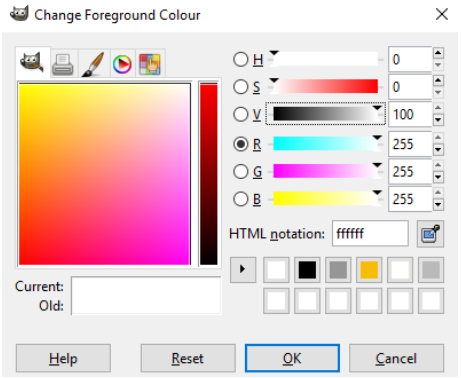


Illustration 25 - Choosing a pure white colour

Now, click on *OK* and make double click in the *'Bucket Fill Tool'* in the toolbox. This opens a new window with different options.

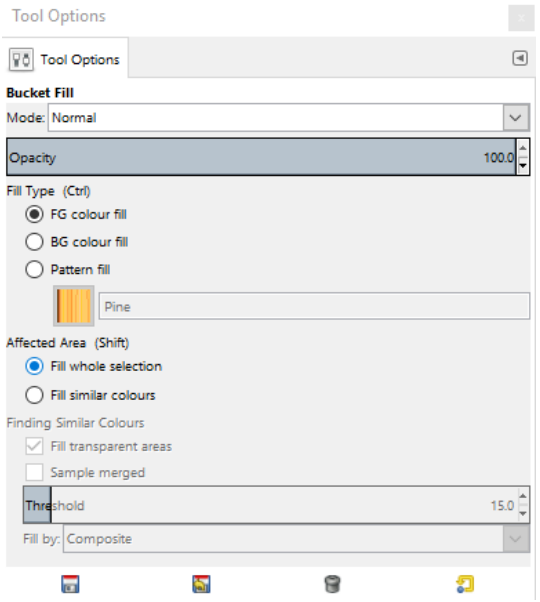


Illustration 26 - Selected options in the bucket fill tool window

It is important to make sure that the selected options appears as in the image, specially the mode in *Normal* and the affected area as *'Fill whole selection'*. This option fills the entire selected area with a colour, and not only areas with similar colours in the image. The mask of the image would look like in the next example, after filling the outside of the object with a pure white colour.

Illustration 27- Image without the background

It is important to save the image in PNG format, in order to have no losses in the image.

After doing this process with the four image of each material, all the masks are read and processed by Python in order that the *calcHist* instruction can understand them.

calcHist can only read masks composed by maps of zeros and ones, so each mask has to be converted in Python. The next code does that operation:

```
premask=cv2.imread('Example.png')
mask=(premask[:,:,0]==255)*(premask[:,:,1]==255)*(premask[:,:,2]==255)*1
mask=mask.astype(np.uint8)
mask=1-mask
mask_test=mask*255
cv2.imwrite('Example_maskBW.png',mask_test)
```

Code 5 - Processing a mask with Python

First, the program read the image generated with GIMP, and sets a condition for the pixels. Pixels with a value equal to 255 in the three RGB channels have a '*True'* value, while other pixels would have *'False'*. When pixels values are multiplied by one, their values change to zero in case of *False* or one in case of *True*.

Now, pixels inside the object have a zero value while pixels outside have one. If *calcHist* process this mask, it would detect the object as background (due to its value of zero) and the background as the object, so an inversion of the values are needed. This is the cause of the operation $mask = 1 - mask$, where the resulting value would be zero if the previous one was one, and vice versa.

To check that the mask is successfully made, its value is multiplied by 255 and saved in an image file, where the white areas correspond to the pixels of the object while the black areas are the background pixels. Otherwise, if the image of the mask is stored without multiplying, the resulting image would be entirely black due to the short difference between ones and zeroes in the spectrum. The image of the example mask would look like this, after the multiplication:

Illustration 28 - Mask multiplied by 255

As it was explained, the pixels of the plastic object are ones in the mask, while the background corresponds to zeroes.

Furthermore, values for grey pixels that have an undetermined value of hue, that OpenCV considers zero, are deleted from all the histograms of hue in order to have more accurate results.

After focusing in the objects of the images, more statistical parameters were added to the calculation in Python:

- **Standard deviation:** evaluates the dispersion of the values in the histogram, and is defined as the square root of the variance.

$$\sigma = \sqrt{\sum_{i=0}^{255} \frac{h[i] \cdot (i-\mu)^2}{\sum h[i]}} \quad (7)$$

- **Relative deviation:** is the ratio between the standard deviation and the average of the histogram values. This parameter is really important, because estimates the dispersion of the values depending on the average value of the pixels.

$$RD = \frac{\sigma}{\mu} \tag{8}$$

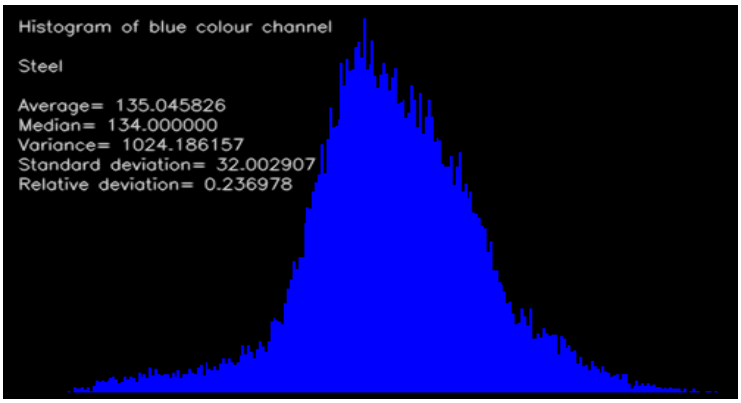


Illustration 29 - Histogram of the blue channel for a steely object showing more parameters

The image corresponds to the steel object analysed previously without applying a mask. As we can see, the values for the average and the median have changed, and the value for the variance has been corrected. Standard and relative deviation values are included in addition.

4.2.4. Creation of a databank

The next step in the analysis of parameters was the creation of a databank file, where the values can be saved and consulted. At first, the values were stored into an Excel file, depending on the channel and the parameter that represents.

The library *Openpyxl* is used in this project to manage Excel files in Python, using its feature *Workbook*. A Workbook is a variable that contains the data in a structure identical to an Excel file, and that can be saved as an Excel sheet. It is needed to open the Workbook first, using the following instructions:

```
from openpyxl import Workbook
wb=Workbook()
ws=wb.active
```

Code 6 - Opening a Workbook with Openpyxl

The Workbook is now active and can be filled with information. For this intention, a function is defined to write values in each cell, specifying its row and column:

```
def wr_data(ws,row,mat,hist,av,med,var,std,rd):
        ws['A'+'%d' % row]=mat
        ws['B'+'%d' % row]=hist
        ws['C'+'%d' % row]=av
        ws['D'+'%d' % row]=med
        ws['E'+'%d' % row]=var
        ws['F'+'%d' % row]=std
        ws['G'+'%d' % row]=rd
        return ws
```

Code 7 - Filling a Workbook

This function is called once for every histogram, and saves the name of the image, the channel of the histogram, and the average, median, variance and standard and relative deviations in different columns. As we can see, Workbook allows the writing of different data types, such as numeric data or string.

The variable *row* is a counter that increases once after saving all data values of a histogram in a row, so all the information of an image is stored in consecutive rows depending on the analysed histogram. The resulting structure looks like the following inside the Excel file:

Material	Histogram	Average	Median	Variance	Standard Deviation	Relative Deviation
Wood1	Blue Channel	31,26212	28	300,4314	17,33296	0,55444
Wood1	Green Channel	96,09756	96	869,3007	29,48391	0,306812
Wood1	Red Channel	184,3285	187	1009,745	31,77649	0,172391
Wood1	Cumulative histogram	142,4626	143	4355,53	65,99644	0,463254
Wood1	RGB histograms	103,8961	96	1553,929	39,41991	0,379417
Wood1	Grey histogram	115,0794	116	789,8466	28,10421	0,244216
Wood1	Hue	12,45415	13	3,991139	1,997784	0,160411
Wood1	Saturation	213,3449	216	395,4454	19,88581	0,09321
Wood1	Value	184,3285	187	1009,745	31,77649	0,172391
Wood2	Blue Channel	44,96245	43	398,0897	19,95218	0,443752
Wood2	Green Channel	119,1066	118	403,72	20,09278	0,168696
Wood2	Red Channel	189,2705	189	218,9843	14,79812	0,078185
Wood2	Cumulative histogram	149,0381	150	3906,907	62,50526	0,419391
Wood2	RGB histograms	117,7799	118	1270,649	35,64616	0,302651
Wood2	Grey histogram	131,6683	131	332,6025	18,23739	0,13851
Wood2	Hue	15,43082	15	1,075431	1,03703	0,067205
Wood2	Saturation	195,9618	196	520,0214	22,80398	0,11637
Wood2	Value	189,2705	189	218,9843	14,79812	0,078185

Table 2 - Structure inside the stored Excel file

Notice that the names in the column '*Material'* refers to images stored in the databank, and not to materials guessed by the program, so the material of each image was non previously. That feature was implemented afterwards in another way, and will be explained later in this document.

At this point, when a data acquisition process has been set, more pictures were included in the previously analysed ones, in order to expand the databank.

A total of 32 images were taken from the internet, 8 for each material, and analysed following the previously explained process of mask creation, histogram calculation and saving into the Excel file. The databank grew up to 12 images for each material, among wood, glass, steel and plastic.

4.2.5. Analysis of parameters

After setting a databank with all the parameters of histograms, it is time to develop a way to see effectively the relations between materials, in order to find unique values or characteristics that makes a difference between them. Previously during the project, images were studied observing histograms first, and comparing numeric values of the parameters for each histogram in later stages. Nevertheless, due to the large number of images and the low efficiency of the used methods, where images are analysed one by one, is needed a more effective method.

The first method to analyse all the images at once was plotting the values for a pair of parameters and identifying the material that corresponds to each image. To make plotting possible, the library *matplotlib* was used, in particular, its *pyplot* tools.

To plot a pair of selected parameters, a new function is defined.

```
import matplotlib.pyplot as plt
#-----------------------------------------------
def plot_data(ws,celly,cellx,letter1,letter2,xlabel,ylabel,name):
        plt.clf()
        datax=[]
        datay=[]
        numbers=range(1,49)
        #Parameter of y axis
        for y in range(celly,480+celly,10):
                datay.append(ws[letter1+'%d' % y].value)
        #Parameter of x axis
        for x in range(cellx,480+cellx,10):
                datax.append(ws[letter2+'%d' % x].value)
        #One line for each material
        line1, =plt.plot(datay[0:12],datax[0:12],'bo',label='Wood')
        line2, =plt.plot(datay[12:24],datax[12:24],'ro',label='Glass')
        line3, =plt.plot(datay[24:36],datax[24:36],'go',label='Plastic')
        line4, =plt.plot(datay[36:48],datax[36:48],'yo',label='Steel')
        #Create the legend chart
        plt.legend([line1,line2,line3,line4],['Wood','Glass','Plastic','Steel'])
        plt.legend(fancybox=True,framealpha=0.5)
        #plt.axis([-1,13,0,300])
        plt.ylabel(xlabel)
        plt.xlabel(ylabel)
        plt.savefig(name)
```

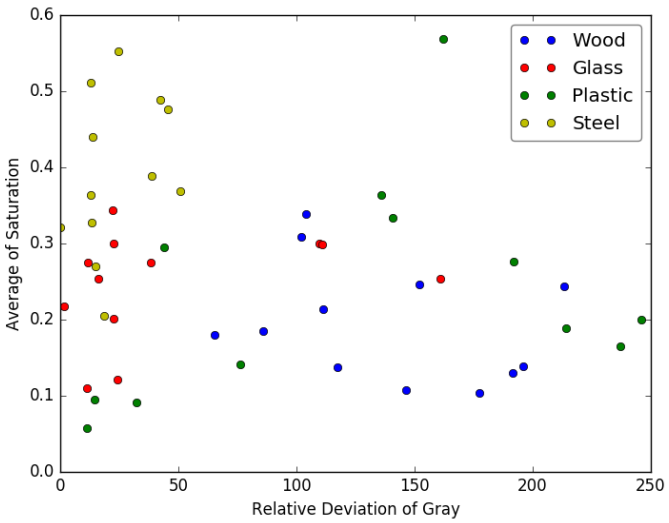
Code 8 - Code for plotting a pair of parameters

This function is employed whenever there is a need to plot data. It takes the Workbook where data is stored, and the letter and number that identifies the cell of the first value of a parameter.

Additionally, it uses labels for x and y axis, and a name for saving the resulting plot into an image file.

At first, it cleans the window of existing plots and collects data from all the cells corresponding to the parameter of the same channel that the present in the beginning cell. As there is a separation of 10 cells between each value and the next one for the same parameter, this function takes values from 10 by 10.

After collecting all the values, it classifies them according to their position in the Excel sheet, which corresponds to a specific material.



Plot 1 - Relative deviation of gray over the average of saturation

For each material, the point that represents a value in the 2-dimensional plot has a different colour: blue for wood materials, red for glass, green for plastic and yellow for steel objects.

Furthermore, it includes a legend and some labels for the axis, as we can see in the following example image.

For perfect results, all the circles of the same colour (same material) should be in the same region of the graph, clearly separated from other materials, but it very rare in reality.

As we can see, values are spread and sometimes are closer to values for other materials than to the same material ones. Using the current statistical parameters it is very difficult to find differencies between materials.

To have a wider vision of the characteristics of each material, more statistical parameters are introduced:

- **3rd Central moment:** is a needed calculation to obtain skewness, and it is defined as:

$$\mu_3 = \mu(h^3) = \frac{h[i]^3 \cdot i}{\sum h[i]^3} \tag{9}$$

- **Skewness:** is a measure of the asymmetry of the distribution about its mean. It can be defined as the ratio between the 3rd central moment and the standard deviation raised to its third power.

$$sk = \frac{\mu_3}{\sigma^3} \tag{10}$$

- **Spikiness:** is a quantification of strong variation between the values of neighbour pixels. The mathematical definition is the following:

$$sp = \frac{\sum_{i=0}^{254}\left(\frac{h[i]-h[i+1]}{\sum h[i]}\right)^2}{255} \tag{11}$$

  High spikiness values correspond to histograms with big variations between neighbour pixels, that is, the presence of many spikes in the histogram.

Illustration 30 - Two histograms with different value for spikiness

The first image has a low value for spikiness because of its smooth contour, while the second image has a higher spikiness due to the presence of three spikes.

- **Percentile:** is a value from 0 to 255, that contains the number of pixels from 0 to that position that represent a % of the total. 25th percentile is the value that contains the pixels, from 0 to that position, that represent the 25% of the total pixels in the image, while the 80th percentile contains the 80% of pixels, and the 50th percentile corresponds to the median by definition.

  To calculate a certain percentile, it is needed to calculate the cumulative histogram previously. In this project, the 25th, 90th and 95th percentiles are used.

- **Slope:** is a measure of the steepness of the histogram, or in other words, the ratio between an initial value and an ending value. In our case, 25th and 95th percentile are used to calculate slope:

$$Slope = \frac{95_{th}\ percentile}{25_{th}\ percentile} \tag{12}$$

  A high slope refers to a high amount of pixels concentrated at the beginning of the histogram, while a low slope means that the majority of pixels are reached in high values of the histogram.

After including these parameters into the databank, all the old and new parameters were deeply analysed in pairs by plotting the values in 2-dimensional charts.

When all the possible combinations of parameters are studied, pairs are sorted in 'good pairs' and 'bad pairs'. Also, red, green and blue histograms are discarded, since the same colour can be in different materials.

As it is shown in the following images, a certain colour does not give information about the type of material of the object.

Illustration 31 - Different materials with the same colour

Furthermore, histograms of value coordinate in HSV-system are also discarded, due to its dependence on the intensity of light.

After the parameter analysis, even so-called good pairs of parameters do not provide critical information to be able to make a difference between materials. Moreover, the structure of the code is too complicated to work efficiently, so a more compact structure is needed.

4.2.6. Restructuring the program

Before this stage of the project, all the program code was contained in one unique Python file, that finally included the creation of masks, histogram calculation and parameters measurement and plotting.

The structure proposed at this point is much more efficient and simple. It includes two code files, each one with a different function.

- **Capture data file:** this file is only used whenever a new image or measurement is added to the databank, or when existing calculations must be modified.

  It reads an image from the *Pictures* folder together with its mask, calculates all the histograms and measures the entire set of parameters for each histogram. Afterwards, it saves histograms into image files to the *Graphs* folder, and values for each parameter into an Excel file and into a new .p file to the *Data* folder. Utility and features of the .p file are discussed later in this chapter.

- **Plot data file:** it is an interactive file that reads data from a .p file in the *Data* folder and asks the user which parameters wants to plot. An example of an execution appears in the next image.

```
>>>

Channel for x axis: 'Gray'

Parameter for x axis: 'Spikiness'

Channel for y axis: 'Hue'

Parameter for y axis: 'Relative Deviation'

Do you want to set the axis limits?'no'

---Plotted from P file---

Would you like to plot again?False
>>>
```

Code 9 - Execution of the code for plotting

When executing this code, the program asks for a channel (*Blue*, *Green*, *Red*, *RGB*, *Gray*, *Hue*, *Sat* or *Val*) and for a parameter (*Average*, *Median*, *Variance*, *Standard Deviation*, *Relative Deviation*, *3rd Central Moment*, *Skewness*, *Spikiness*, *90 Percentile*, *95 Percentile*, *25 Percentile* or *Slope*).

After selecting the channel and parameter for each x and y coordinate, there is an option to set the axis limits or not. If yes, the program asks the user for the minimum and maximum value for x and y axis.

Afterwards, the resulting plot is saved into an image file in the *Plots* folder.

A new data structure is also defined in order to manage data more easily. This data consists on a Python dictionary, a data structure similar to an array, where the positions are not integer numbers, but very intuitive indexes.

The dictionary that contains all the values for the different parameters, channels and images is called *Data*, and it is composed by different indexes for each one:

```
Data={}
for a in ['Blue','Green','Red','RGB','Gray','Hue','Sat','Val']:
    Data[a]={}
    Data[a]['Average']={}
    Data[a]['Median']={}
    Data[a]['Variance']={}
    Data[a]['Standard Deviation']={}
    Data[a]['Relative Deviation']={}
    Data[a]['3rd Central Moment']={}
    Data[a]['Skewness']={}
    Data[a]['Spikiness']={}
    Data[a]['90 Percentile']={}
    Data[a]['95 Percentile']={}
    Data[a]['25 Percentile']={}
    Data[a]['Slope']={}
```

Code 10 - Different indexes of the dictionary

Data contains one index for every channel, which contains an index inside for each parameter. To select a parameter of a certain channel from an image, it is as simple as typing *Data['Selected channel']['Selected parameter'][Index of the image]*.

The index of an image is an integer number, unique for every image.

To write data into this file, it is the same process than saving a value into a variable. If the index where to write is empty, the value for that index is changed from *empty* to the new one. If there are one or more values in that index, the new value is concatenated with the one.

```
if flag==False:
    Data['Gray'].update({'Standard Deviation':hgray_sd})
else:
    Data['Gray']['Standard Deviation']=np.append(Data['Gray']['Standard Deviation'],hgray_sd)
```

Code 11 - Code for storing a value in an index of the dictionary

This piece of code illustrates very well how to write a new value into *Data*. To export the dictionary to a file, the library *pickle* is used.

```
data_file=open("Data.p","wb")
pickle.dump(Data,data_file)
```

Code 12 - How to export a dictionary using pickle

On the other hand, reading values from *Data.p* is also very easy:

```
Data=pickle.load(open("Data.p","rb"))
val = Data['Gray']['Spikiness'][1]
```

Code 13 - How to read a dictionary using pickle

This instruction writes the value for the spikiness of the grey channel of image in position 1 into a variable.

4.2.7. Results enhancement

Despite changing the structure of the program, results remained similar than previously, so it is time to think about why values of parameters are so overlapped in all the charts.

The first possibility is that variations in the values of the histograms due to rounding errors (pixel values are arranged in integer values from 0 to 255) affect to the measurement of some parameters, such as spikiness.

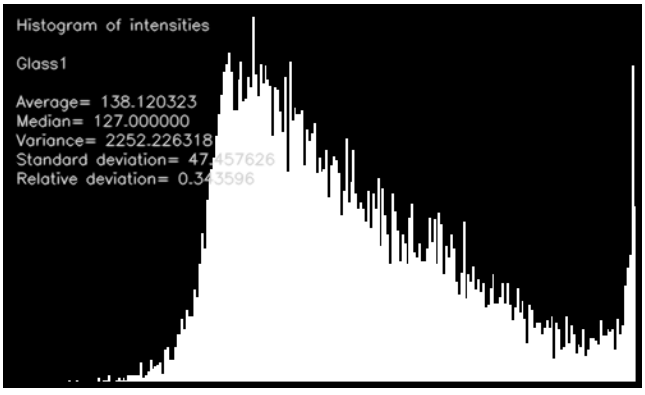


Illustration 32 - Histogram with so many spikes

For that reason, a smooth is required to avoid this little spikes in histograms, applying filtering to the calculated values.

Matlab is used to generate the filter, employing the *Filter Designer Tool*.

```
>> fdatool
>>
```

Code 14 - Instruction in Matlab to open the filter designer tool

Filter Designer Tool window looks as in the next image:

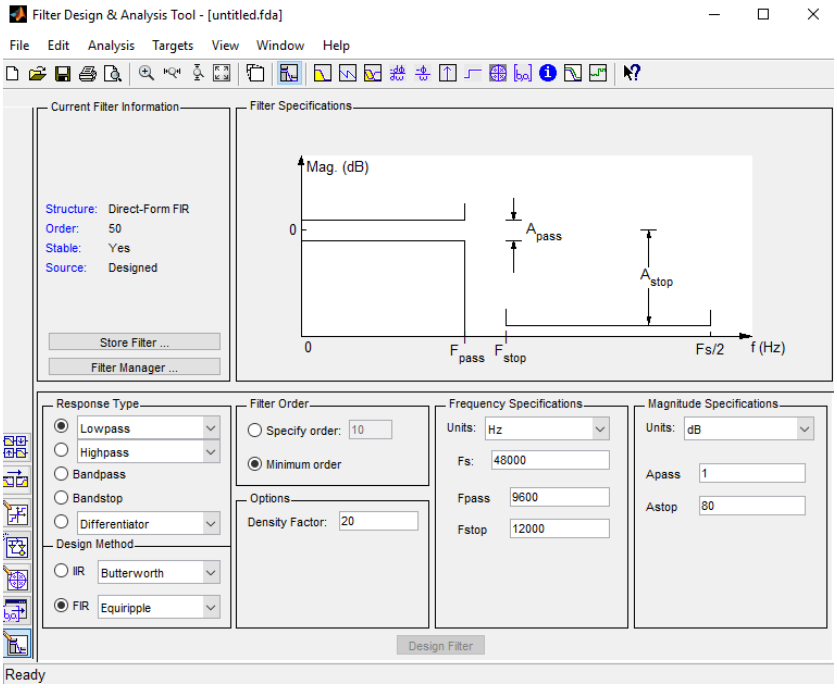


Illustration 33 - Window of the filter designer tool

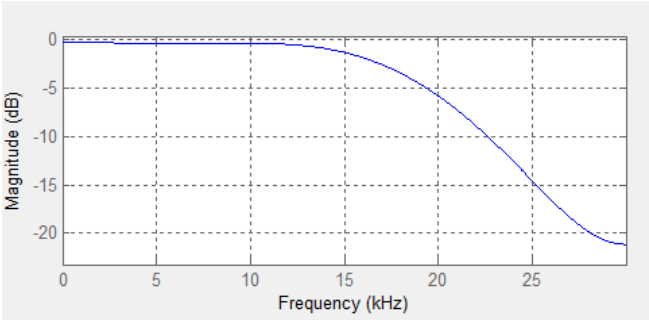


Illustration 34 - Example of magnitude response of a low pass filter

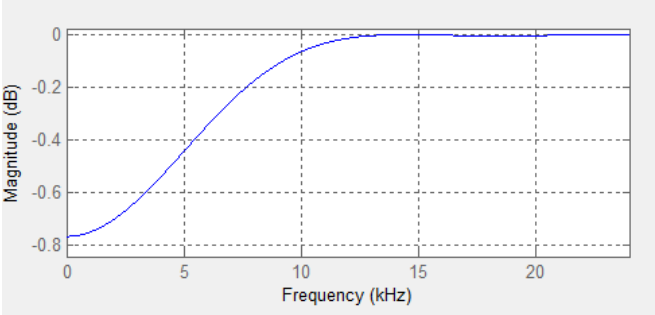


Illustration 35 - Example of magnitude response of a high pass filter

The first image corresponds to the magnitude response of a low pass filter in Db, while the second one is the magnitude response of a high pass filter.

A low pass filter removes the high-frequency information, while the high pass removes the low frequencies.

As the desired filter removes little variations of the histogram values, that correspond to the high frequencies, it would be a low pass filter.

When selecting the characteristics of the filter, an important parameter to take in account is the kind of response the filter would have. It can be a FIR filter or an IIR filter.

- **FIR filters[24]:** finite impulse response filters have an impulse response of a finite duration, and whose output depends only in past and present values, so it is a casual system.

  FIR filters have linear phase response, and are very stable, while requires no feedback.

[24] https://en.wikipedia.org/wiki/Finite_impulse_response

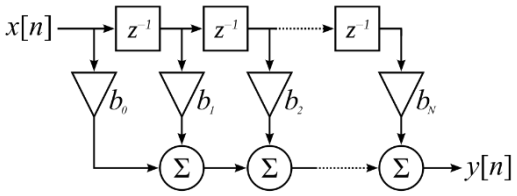


Illustration 36 - Example of a FIR filter structure

- **IIR filters[25]:** infinite impulse response filters have an impulse response that continues indefinitely. They are non-casual systems that needs constant feedback, and have a non-linear phase response that may produce instability.

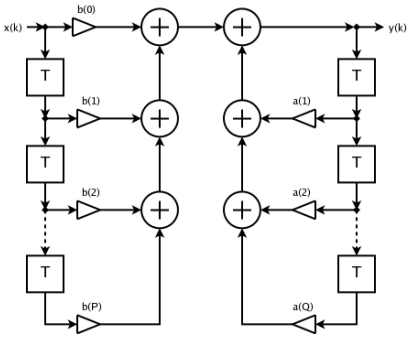


Illustration 37 - Example of a IIR filter structure

The designed filter is a FIR one, to avoid instability, with a high order (255) to have a sharp decline in the response over the cut-off frequency, that is 0.5 because it is normalized between 0 and 1. It is also designed using a Hamming window[26], which minimizes the size of the maximum side lobes. The filter response is shown in the next picture.

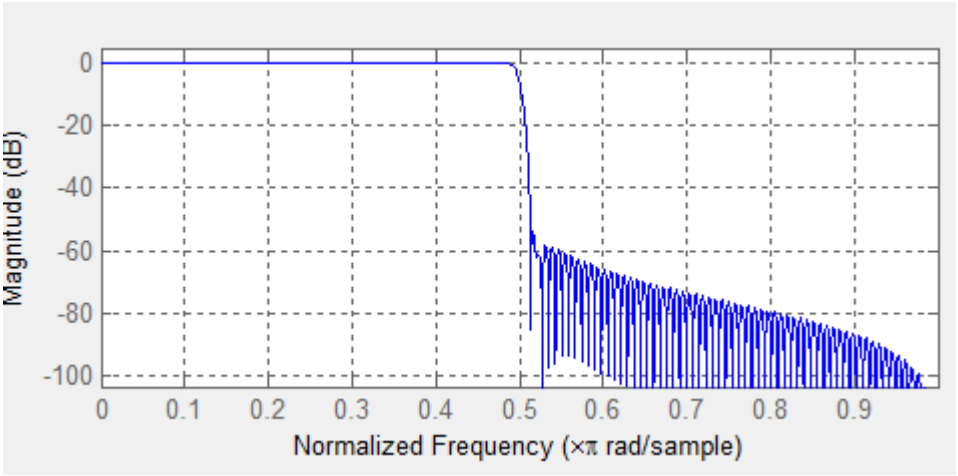


Illustration 38 - Magnitude response for a FIR filter of 255 order, Hamming window and 0.5 of normalized frequency

[25] https://en.wikipedia.org/wiki/Infinite_impulse_response
[26] https://en.wikipedia.org/wiki/Window_function#Hamming_window

After exporting the coefficients of the filter into a text file, they were read by Python using the following instructions.

```
text_file=open('filter_values.txt','r')
w=text_file.read().split(',')
for i in range(len(w)):
    w[i]=np.float32(w[i])
h=np.convolve(h[:,0],w,'same')
```

Code 15 - Making the convolution with the filter in Python

The values in the text file are read and arranged into a vector, that is convolved with the filter values. After convolution, values present less little spikes than before.

Results after applying smoothness are better than without filtering, but values were still very spread out in the charts.

A possibility is that, as the images taken from the Internet are very diverse, results are not as consistent as they would be in reality.

Illustration 39 - Example of images taken from the Internet

For this reason, new images of patterns, and not objects, are taken from the internet in order to avoid such differences between them. Some of the new images are the following.

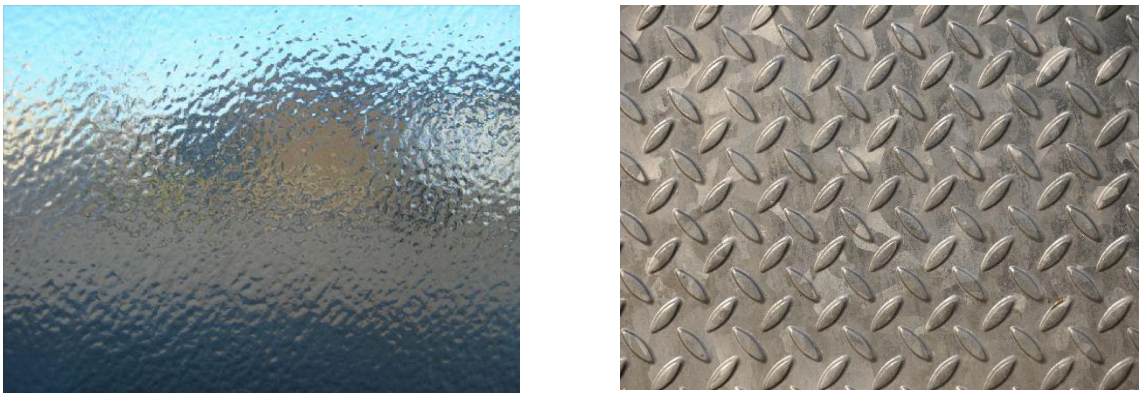


Illustration 40 - Example of patterns taken from the Internet

After analysing the new images, results were even worse than before, due to the low information that this kind of images gives. It is very difficult for a human observer to make a difference between materials only attending to the pattern, and sometimes can lead to confusion.

As explained in the introduction, an example is the texture of a bottle of water. Sometimes they made of plastic or glass, but depending on the light sources, there can be a confusion.

The next image is an illustrative example of this kind of situations.

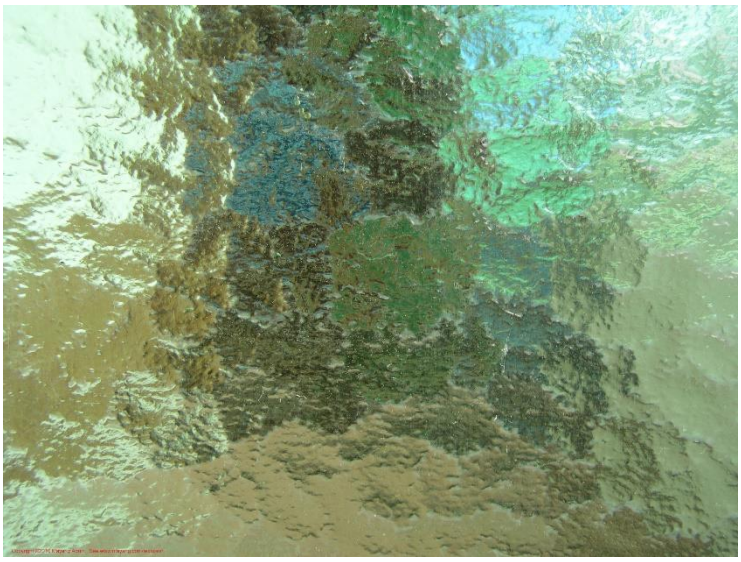


Illustration 41 - Pattern with a material that is difficult to detect

This pattern is part of a material that is very difficult to guess even for a human. Some observers can consider it as a plastic object, while other can find it glass.

It is the same case for this program, which had big problems to distinguish materials using only patterns. A proposed solution is to take pictures of different materials in the same light conditions, and with the same camera. Moreover, the settings of the camera must not be in automatic mode, so that all the conditions are equal for every image.

Images were taken using the camera of a mobile phone, so quality was not so high, but a difference between materials could be perfectly done by a human observer.

Furthermore, all the images were taken with the same dark background, very important for transparent materials, and using two light sources always in the same positions.

Some of the new images are the following ones.

Illustration 42 - Example of photographs taken in the laboratory

Moreover, some new materials are introduced, as new pictures are from five different materials: glass, steel, and wood, which were analysed in previous stages of the project, and porcelain and paper. Plastic objects are not included in this analysis due to a lack of time for taking more photos.

When taking the pictures, objects appeared very small in the images, but it was impossible to take better pictures due to the low quality of zoom of the camera and the position of the light sources, that created shadows over the objects.

The masks for the new images were made by GIMP, and the images were processed by the program in Python using the methods explained before, saving the values into an Excel file to consult them at every time, and into a .p file, to read them later easily.

Furthermore, a better filter was developed with a lower order, in order to avoid incorrect values in the coefficient and problems with the phase response. The magnitude and phase responses of the current filter appear in the following images.

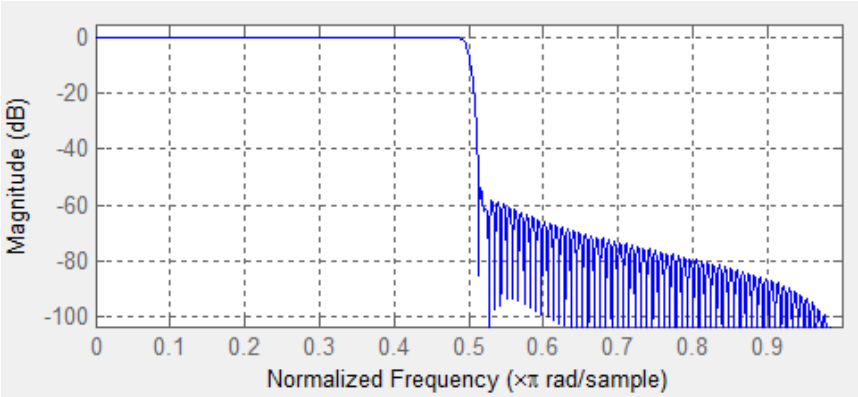


Illustration 43 - Magnitude response of a low pass filter of a high order

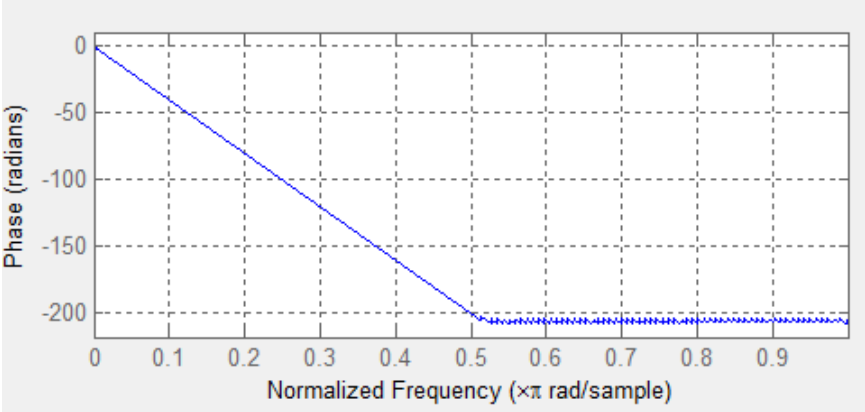


Illustration 44 - Phase response of a low pass filter of a high order

Due to the high order of the filter, its phase response reaches the -180º before the cut-off frequency, so the results for the convolution can be incoherent.

The magnitude and phase responses for the proposed filter of a fifth order are the next.

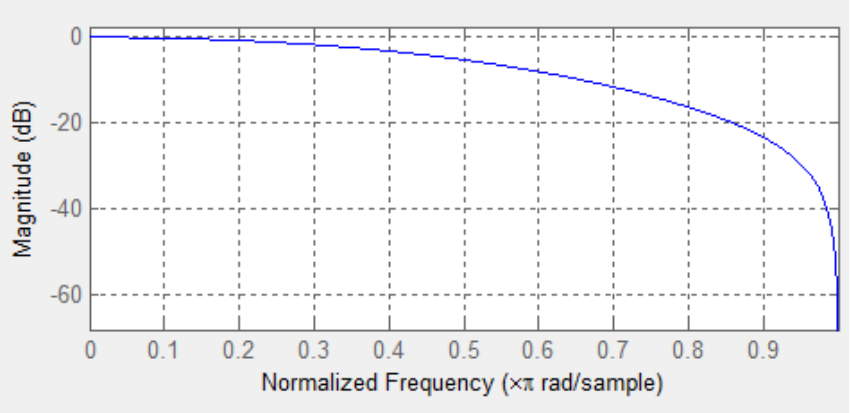


Illustration 45 - Magnitude response of a low pass filter of a low order

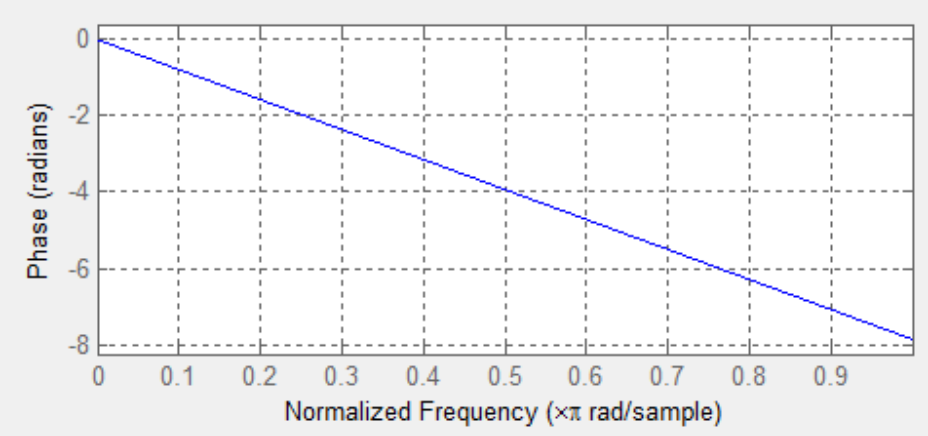


Illustration 46 - Phase response of a low pass filter of a low order

As we can see, if this new filter there are no problems with the phase, which is far of the -180º (phase inversion). However, its magnitude response is not as selective in frequency as in the previous filter, because it still has a -20dB gain for 0.8 normalized frequency.

The effect of the new filtering appears in the second image below, while the previous one has no filter smoothing applied.

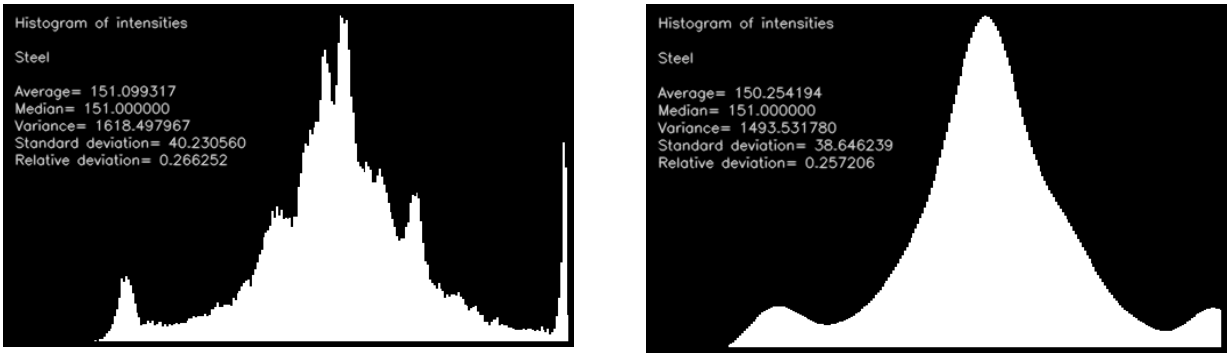
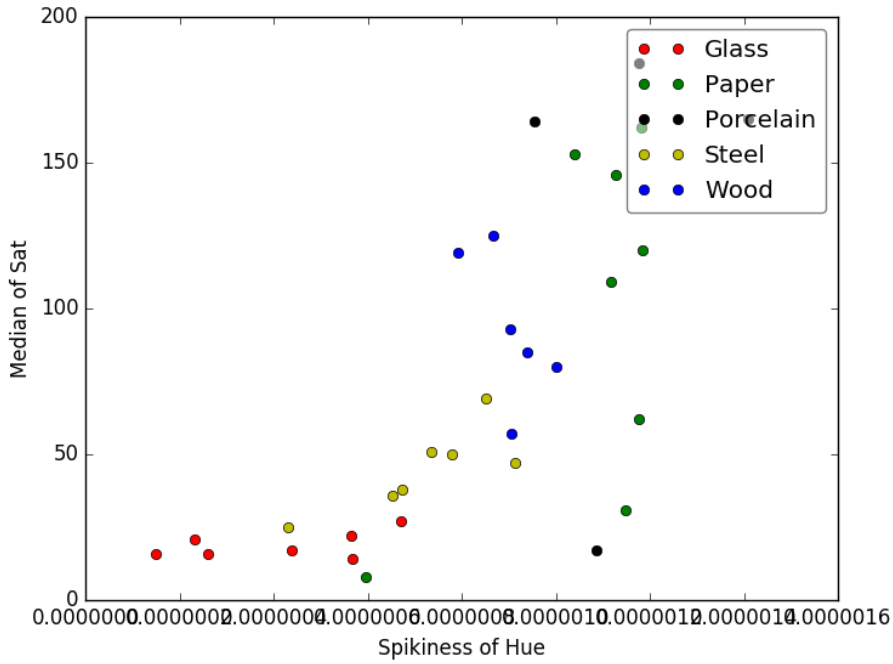


Illustration 47 - Effect of smooth in histograms

The evaluation of the results for both filters shows that the values obtained using the new filter are more promising than the previous ones.

However, the best results do not even reach the minimum clarity in the charts to make a difference between materials. One of the best results is shown hereunder.



Plot 2 - Spikiness of hue over median of saturation

The objective of separate the values of different materials in areas is in part achieved.

Porcelain values appear very far from other values, even though glass, wood and steel objects are in general have some of their values mixed in other areas.

A possible reason why some values are still overlapped, or areas are not well defined, is the low quality of the taken images. Sometimes, they are very noisy that pixels have inaccurate values. In addition to this, objects have a small size in the objects, and the amount of pixels may be insufficient.

Illustration 48 – Noise due to the low quality of the camera

The previous images illustrate the low quality of the zoom of the camera, so new pictures are taken with a higher quality camera and in similar conditions. Dark background is used again, as well as the same two light sources.

The new set of pictures, the last one used as databank in this project, is composed of 62 images in total, distributed in different materials:

- Seven images of glass objects.
- Thirteen images of metallic objects.
- Eleven images of plastic objects.
- Ten images of porcelain objects.
- Thirteen images of wooden objects.
- The eight images of paper objects taken in the previous set, due to the lack of time.

The following images are an example of the pictures in the new set, each one for a different material.

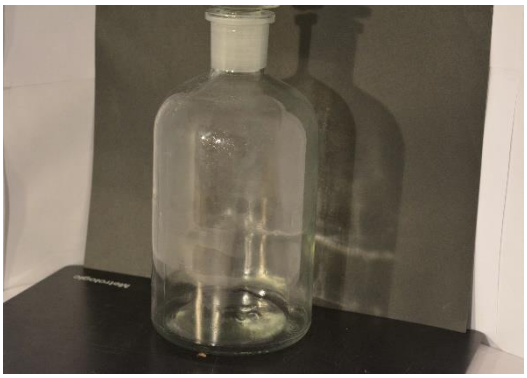


Illustration 54 - Object of glass included in the final set

Illustration 54 - Object of plastic included in the final set

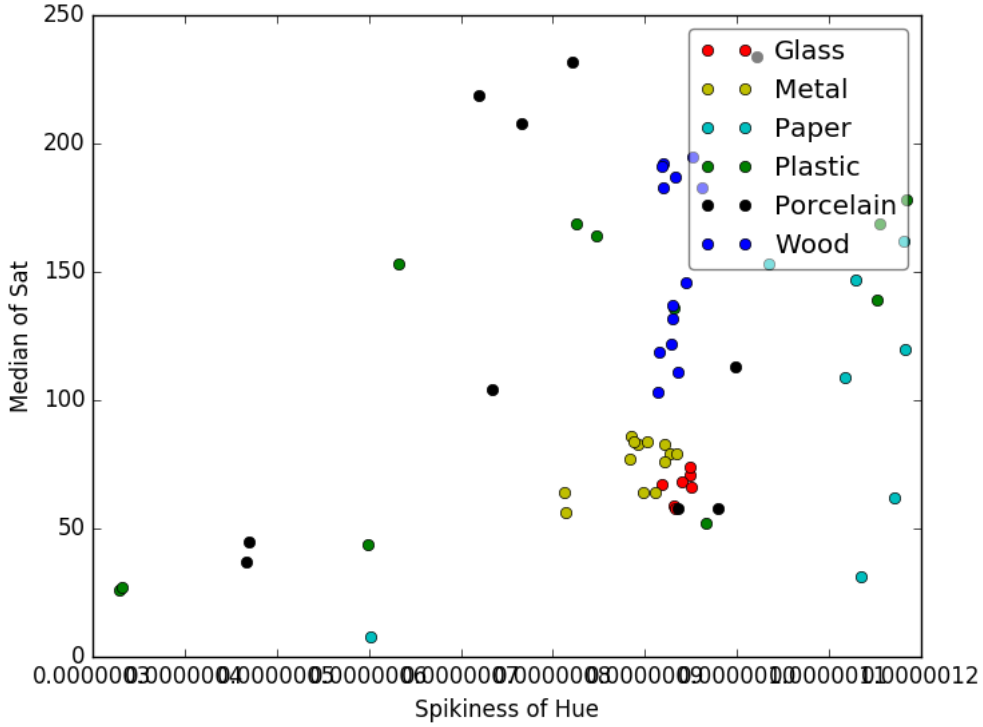
Illustration 54 - Object of porcelain included in the final set

Illustration 54 - Object of wood included in the final set

Illustration 54 - Object of metal included in the final set

Illustration 54 - Object of glass paper in the final set

Results are slightly better for these new images, but it was still difficult to make a difference between all the materials. Only some materials could be theoretically detected using various charts. An example of new result is the plot in the next image.



Plot 3 - Spikiness of hue over median of saturation

In this image, wooden materials can be distinguished from other materials looking at this graph, but the rest of values overlap with each other.

As the image quantity and quality, the filter and the equal conditions of the images were revised, the next step is finding a more effective method to differentiate materials using the existing databank.

4.2.8. Elaboration of decision methods

As the current structure of the code contains two files for calculations and for plotting, a new file is created to use the existing parameters in the databank to take a decision and guess the material what an object is made up of.

Various methods are developed during the elaboration of the work, obtaining different rates of accuracy, and are described in this part of the chapter. This is the first time that the code itself generates a valuation of the results, and not a human observer comparing values or describing histograms.

Before implementing a new decision method to get new results from the program, a flow diagram must be constructed to understand the new resolution process, and to translate it into Python code.

The common characteristic among these decision methods is that, basically, they work in the same way. In all of day, the program asks for a new picture that the user has to select, and after that it compares certain parameters from this picture with the existing values stored in the databank.

The differences between the diverse methods is the comparison process between the test image and the images from the databank.

The decision methods realised in this project are hereunder detailed:

- ***Method 1: Two-dimensional distance***

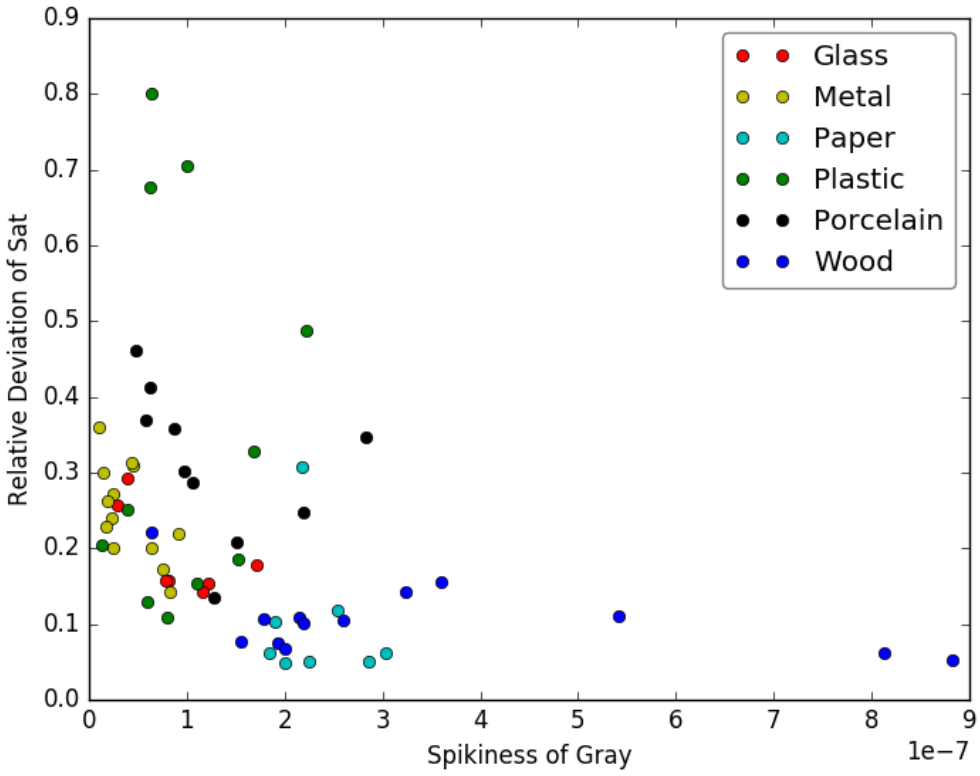
The first decision-making method developed in this project consists in comparing the two coordinates of a point corresponding to a new image, with the previously calculated values of the stored images.

This comparison is based on the closest points to a new one in a two-dimensional chart, measuring distances to these points and selecting the five shortest ones. The most common material among the selected points describes the material of the object in the new image.

The two pairs of parameters selected for this method are the ones that showed better results in previous stages, which are *spikiness of grey channel* and *relative deviation of saturation* on the one hand, and *median of saturation* and *spikiness of hue.*

The chart representing the current values for *median of saturation* and *spikiness of hue* appears in the previous page, while the chart for *spikiness of grey and relative deviation of saturation* is the following.

Notice that all the values that appear in charts are normalized to the maximum value for each parameter, so all values for different materials have the same weight when calculating the distance.



Plot 4 - Spikiness of grey over relative deviation of saturation

For *spikiness of grey* and *relative deviation of saturation* measurements, the program successfully detected eight objects of metal, five of plastic, nine of porcelain, and twelve of wood, while using *spikiness of hue* and *median of saturation*, it could detect four objects of metal, three of paper, three of plastic, six of porcelain and all the wooden objects.

Due to the disparate results of both pairs of parameters, emerged the idea of mixing both, and even more, to have a deeper comparison between materials. This lead to the next method developed during this stage of the project.

- ***Method 2: Iterated calculation of two-dimensional distances***

The following method is a modification of the previous one, which calculates the two-dimensional distances from the new point to the existing ones, similarly to the previous method.

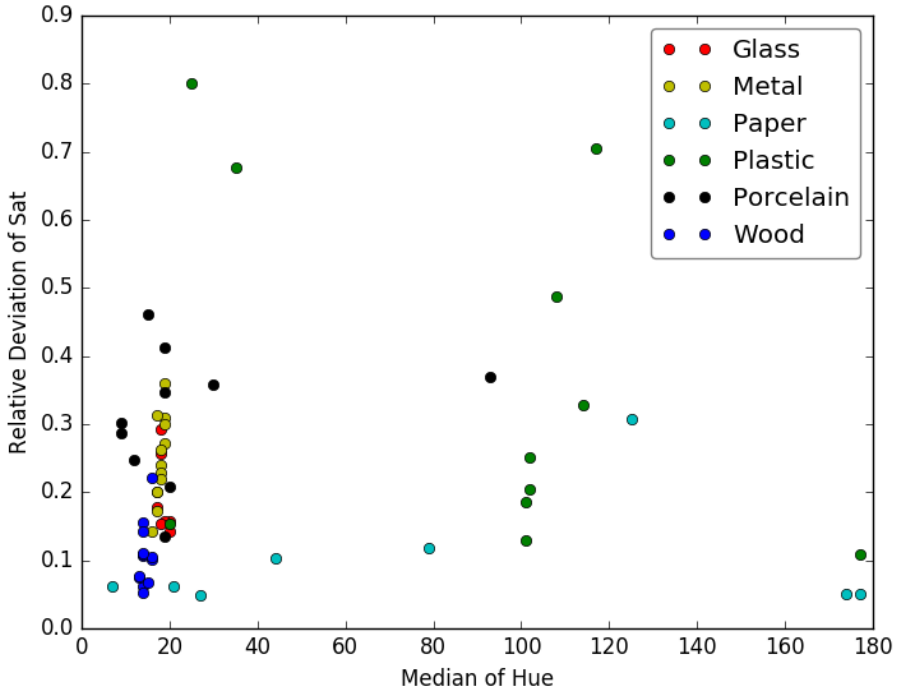
The main difference is that the first method measured distances only for a pair of parameters, while releases the same task in each iteration, calculating different parameters each time.

Material identification comes from the material that is closer to the new point in all the charts.

For every iteration, the materials of the closest points are saved along with the five closest points of other iterations. The material that appears more often among the five closest points in all iteration is the material decided by the program to be the one that the object is made up of.

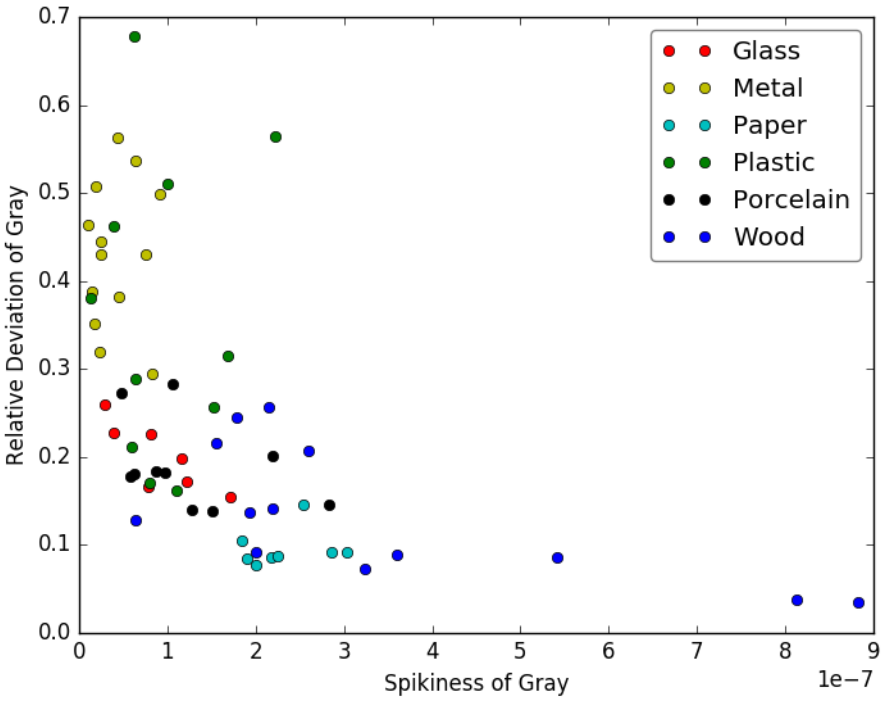
Pairs of parameters selected for this method are, apart from the previous two couples already introduced, *median of hue* and *relative deviation of saturation*, and *spikiness and relative deviation of the grey channel*.

Chart for *median of hue* and *relative deviation of saturation* is the following one.



Plot 5 - Median of hue over relative deviation of saturation

Chart for *spikiness and relative deviation of the grey channel* corresponds to the next image.



Plot 6 - Spikiness of hue over relative deviation of grey

The reason by this two couples of parameters is that, for the first histogram, metal values are concentrated very close of each other, while paper values are relatively separated from other materials.

On the other hand, the second parameters show a histogram where values for metal are relatively separated from other materials, as well as some values for wooden objects. Also, some values for paper are close to each other.

Results for this method were different than for the previous one, successfully guessing one object of metal, six of plastic, all the porcelain and twelve of wood. However, these results are only a difference rather than an improvement.

A possible cause is that using this method not only easily identifiable materials are better guessed, but the most difficult ones accumulate distances to materials that do not correspond with the one that object is made up of.

For example, in the case of paper, it is often successfully detected with the two first pairs of parameters used in the previous method, but in the new pairs, they can be mixed up with plastic or wood materials. As a result, in this second method they are identified as wood objects in all the images.

As this method has proved to be inefficient, a new method is developed as a enhancement of the first method.

➢ *Method 3: Three dimensional distance*

This result is an improvement of the first method in some aspects.

The first one is that three dimensional distances are calculated now, instead of two dimensional ones, so that values can be distinguished in an easier way in a three-dimensional space. For this reason, three parameters are now used to describe the characteristics of each material.

On the other hand, a new set of parameters related with a certain channel that was discarded previously, HSV Value, has proved to be a clue factor when improving results.

New parameters are related to the morphological gradient of the value channel, that describes the variation of the brightness between neighbour pixels of the image, and different statistical measures are used to describe this new characteristic. Their definitions are similar to the described previously for more general parameters.

- **Average of the gradient of brightness:** this parameter represents the middle difference of brightness of the pixels in the image. It is higher for wooden objects where the present of seams results in a high gradient value for neighbour pixels.

Illustration 55 - Example of image with seams

- **Median of the gradient of brightness:** it measures the brightness value below which a half of the pixels is situated.

- **Variance of the gradient of brightness:** it gives information about how are the values for variations of brightness spread out from the middle value. It can be a measure of how the values for brightness variations are distributed along the surface of the object.

- **Standard deviation of the gradient of brightness:** evaluates the dispersion of the variation in brightness.

- **Relative deviation of the gradient of brightness:** it is the ratio between the standard deviation and the average value of the gradient of brightness.

To apply the morphological gradient into an image, is created a rectangular kernel that is applied along the image. The kernel used has the following coefficients:

```
>>> kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
>>> print kernel
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

Code 16 - Coefficients of the kernel

Finally, the next instruction calculates the morphological gradient in the surface of the object:

```
h_Val_gradient = cv2.morphologyEx(hsv[2], cv2.MORPH_GRADIENT, kernel)
```

Code 17 - Instruction to calculate the morphological gradient

After calculating and including these new parameters into the database, the three-dimensional distances from a new point to the others are calculated with the previous parameters and with the *average of gradient of brightness*.

For *spikiness of grey*, *relative deviation of saturation* and *average of gradient of brightness*, the program successfully detects seven objects of metal, four objects of plastic, twelve objects of wood and all the objects of paper and porcelain.

On the other hand, for *spikiness of hue*, *median of saturation* and *average of gradient of brightness*, the program guessed four objects of metal, four of paper, six of porcelain and all the wooden objects.

As we can see, results for *spikiness of grey*, *relative deviation of saturation* and *average of gradient of brightness* are more efficient than the other set, and that using other methods, but it is not accurate enough in some aspects.

In order to achieve better results, this method is extended to more dimensions and parameters, that are explained hereafter.

➢ *Method 4: Another methods for 3 or more dimensional-distances*

To expand the previous method, various parameters are added to the calculation, bringing different new methods with diverse results. *Spikiness of grey*, *relative deviation of saturation* and *average of gradient of brightness* are used again.

- Including the *third central moment of hue* to the previous set. Successfully detected materials include ten objects of metal, one of paper, five of plastic, six of porcelain and twelve of wood.
- Including the *spikiness of hue*. Materials that were satisfactorily detected correspond to five objects of metal, three of paper, seven of porcelain and twelve of wood.
- Another method consists in measuring three-dimensional distances as in *Method 3*, but replacing the *relative deviation of saturation* parameter with the *relative deviation of RGB*. Results for this method include successful detections for twelve objects of metal, six of porcelain, twelve of wood and all the objects made up of paper.

- The last method of this category consists of the measure of five-dimensional distances, including the *median of saturation* and the *relative deviation of grey* to the previous set of parameters. Materials successfully detected correspond to one object of glass, twelve objects of metal, five objects of plastic, nine objects of porcelain, twelve of wood and all the objects of paper.

As the results of these methods are not very accurate in general, other solution is proposed to enhance the effectiveness of the program. This last idea of improvement is a completely change for the decision-making method, that reveals much better results.

➢ *Method 5: Decision tree*

This last method supposes a big change compared to the previously introduced methods, as it does not work measuring distances but ranges of values for each material.

The core of this method consists in the establishment of various ranges of values for certain parameters that represent a difference between a parameter and the others. For this purpose, all the ranges of values for each parameter and material are studied, identifying the maximum and minimum values for each one. These two values are included as the upper and lower limits of a condition to identify a material, with a tolerance in both limits to cover possible values out of this range.

There is a counter for each material, which increases when a parameter of the new image fits into the range for a material. At the end of calculations, the counter with the highest value represents the material found for the new object.

Parameters that are selected for this method are *relative deviation of grey, 95 percentile of RGB, variance of hue, median of hue, average of the gradient of brightness, variance of the gradient of brightness and relative deviation of gradient of brightness.* They were selected due to the interesting values for a type of material that make easy to identify it among the different materials.

Results for this method are the most promising, as the program can detect five objects of glass, eleven of metal, six of paper, nine of porcelain and all the objects of plastic and wood.

A deeper analysis of results is presented in the next episode.

Chapter 5. Analysis of results

This chapter is a summary of all the results obtained during the elaboration of the project, as well as a comparison between different methods and techniques.

At the very first stages of work, results were very intuitive and simple, only based in the contour and patterns observed in histograms made by GIMP. This type of results are completely inefficient, but served as an introduction to digital image processing and histogram interpretation.

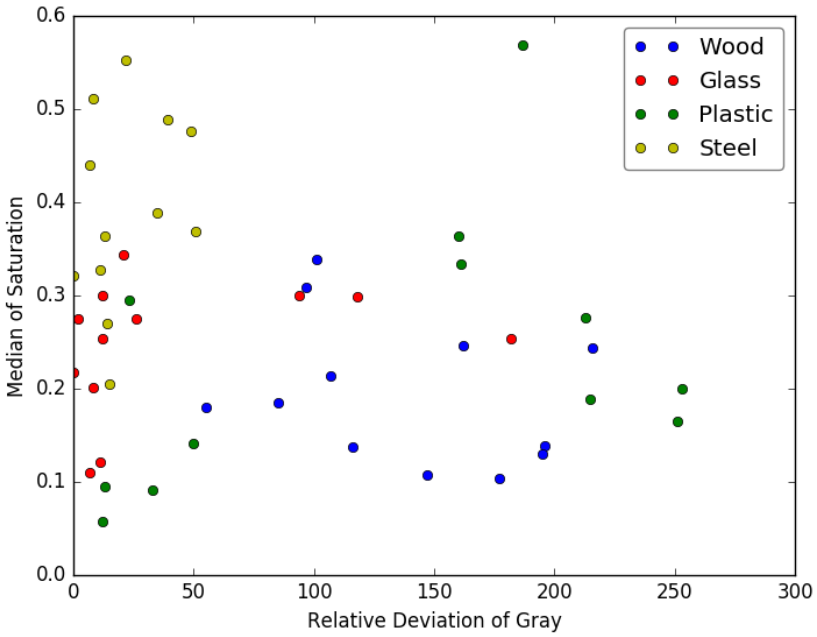
When histograms began to be made with Python, it allowed to measure quantitatively their values, that were used in many statistical calculations. More parameters joint to the first calculations of average, median and variance, and gave a more extensive view of the characteristics of each material. Furthermore, the use of masking brought measurements that are more realistic.

After creating a databank to store the values of parameters, it was possible to consult them at every time very easily. On the other hand, numbers for rows and columns had to be remembered in order to plot certain parameters. For this reason, a .p file was created to handle these values in an easier way.

The first tool that led to useful comparisons was two-dimensional plotting, that allowed to watch how values for different parameters are distributed in a chart. This was the most used comparison method during the project.

By plotting different parameters, many diverse results were obtained:

- In the first plots, values for different materials appeared mixed up with each other, although sometimes it is easy to identify values for wooden objects as describing a relatively separated area.

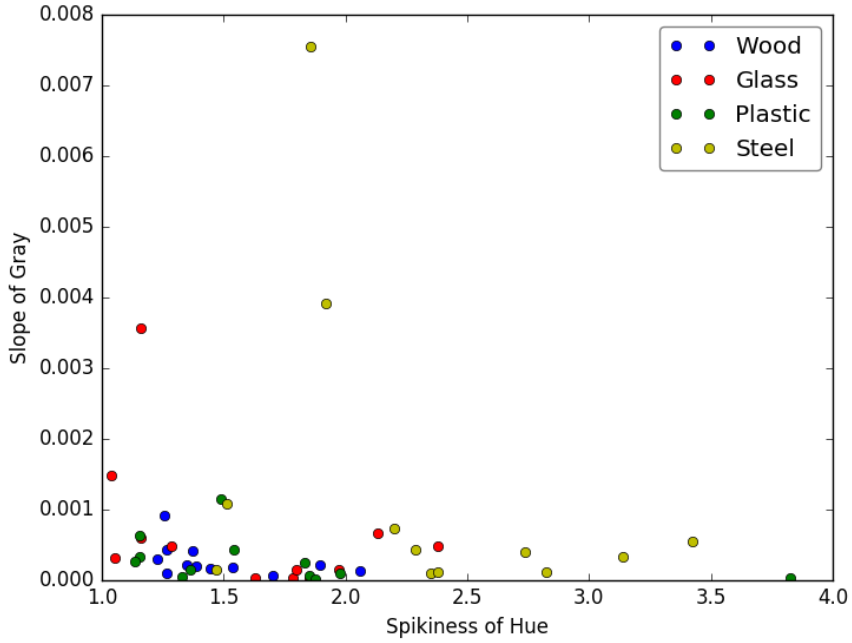


Plot 7 - Relative deviation of grey over median of saturation

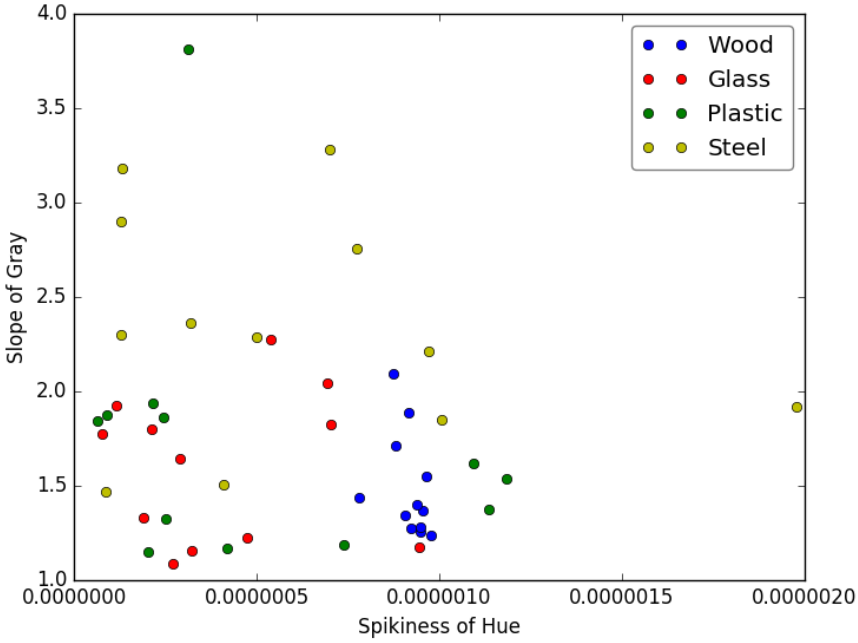
- After applying smoothing using a high order-filter, plots using *spikiness* parameter were more trustful. On the other hand, distribution of values for other parameters did not change very much.

  In some cases, wood values appeared separated, while in other occasions steel also appeared separated from other materials, though some points of glass or plastic overlapped in those regions.

  The following images show the different results when applying smoothness or not.

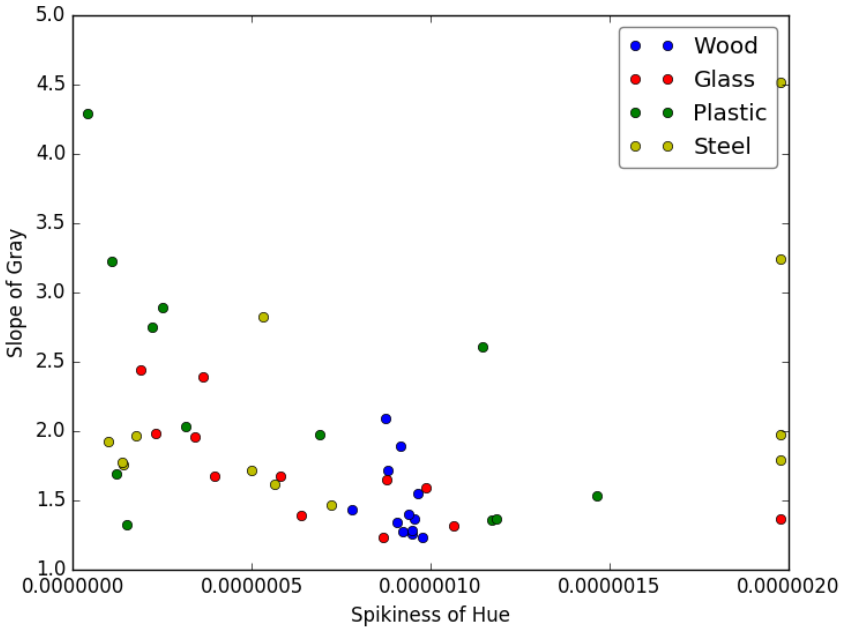


Plot 8 - Spikiness of hue over slope of grey without filtering



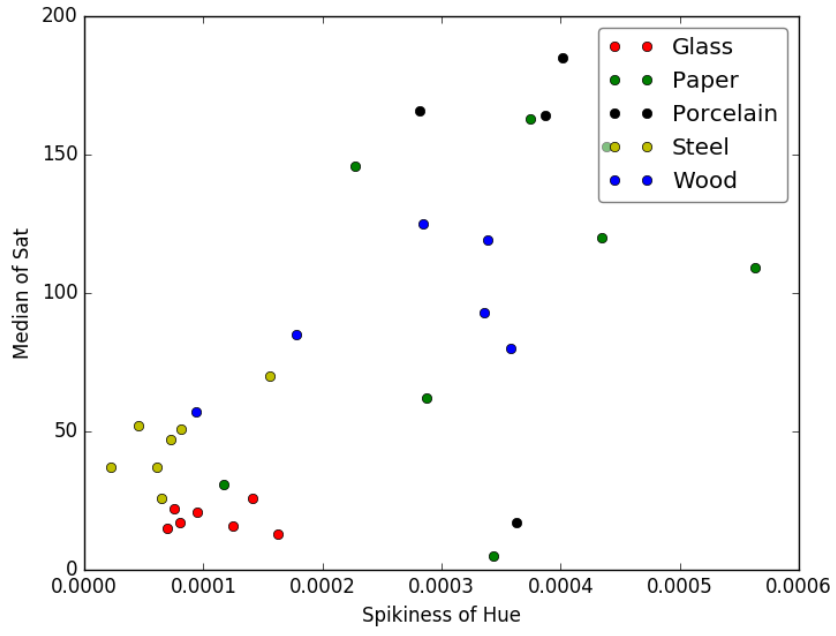
Plot 9 - Spikiness of hue over slope of grey with filtering

- Analysing patterns instead of the first images, results went worse due to the few information that a pattern can give, related to the material that it is made of. An example is the following image, compared with the previous one for the same parameters. Hereunder, parameters are more mixed up so it is impossible to identify a single material describing a separated region in the x-y plane.

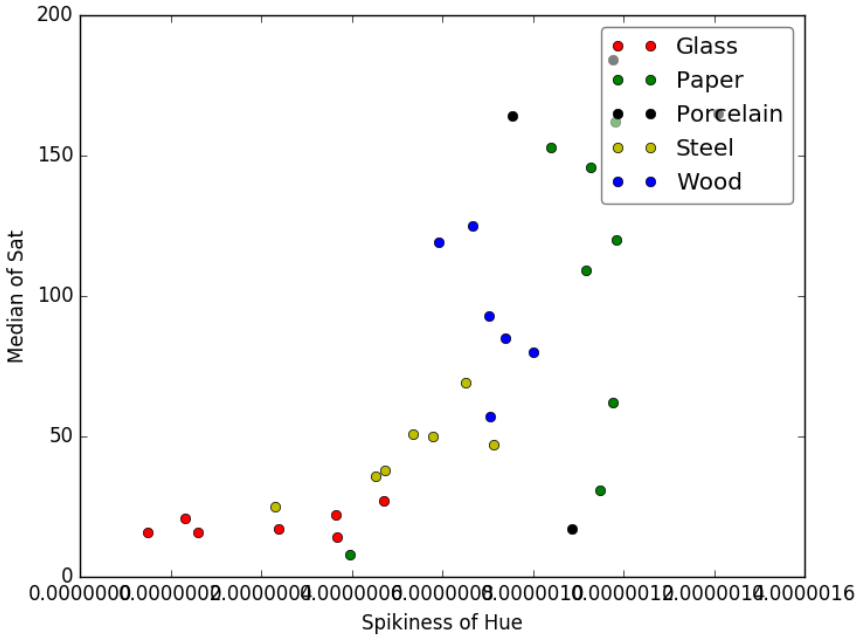


Plot 10 - Spikiness of hue over slope of grey using patterns

- When analysing pictures taken in the laboratory using the camera of a mobile phone, better results were obtained, as more materials described separated areas. Materials are more identifiable when using a filter with a low order.

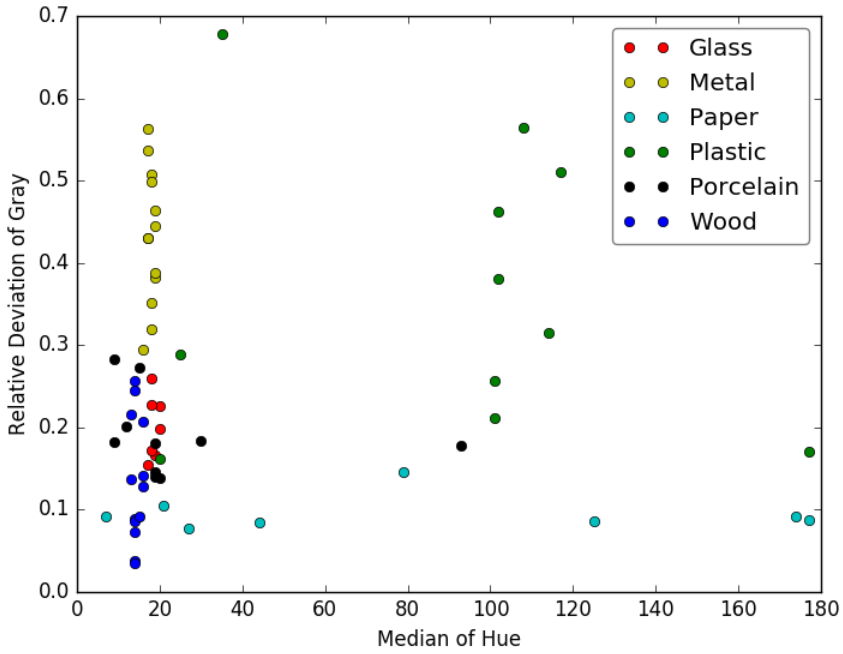


Plot 11 - Spikiness of hue over median of saturation using photographs taken with a mobile phone, using a filter of a high order



Plot 12 - Spikiness of hue over median of saturation using photographs taken with a mobile phone, using a filter of a low order

- With the pictures taken in the laboratory with a higher-quality camera, more pictures were included. Results like the following one reveals the need of a more sophisticated method for detecting materials.



Plot 13 - Median of hue over relative deviation of grey, using pictures taken with a higher-quality camera

- The different results for each decision-making method are compared in the following table. Note that these results are obtained using the last pictures taken in the laboratory

as databank values, and after that, choosing one of these pictures and trying to detect its material.

Material to detect	**Two-dimensional distances (*first set of parameters*)**	**Two-dimensional distances (*second set of parameters*)**	**Iterated two-dimensional distances**	**Three-dimensional distances (*first set of parameters*)**	**Three-dimensional distances (*second set of parameters*)**	**Four-dimensional distances (*first set of parameters*)**	**Four-dimensional distances (*second set of parameters*)**	**Three-dimensional distances (third set of parameters)**	**Five-dimensional distances**	**Decision tree**
Glass	0%	0%	0%	0%	0%	0%	0%	0%	14.3%	**71.4%**
Metal	62%	30.8%	7.7%	53.8%	30.8%	76.9%	38.5%	92.3%	92.3%	**84.6%**
Paper	0%	37.5%	0%	100%	50%	12.5%	37.5%	100%	100%	**75%**
Plastic	45%	27%	54%	36%	0%	45%	0%	0%	45%	**100%**
Porcelain	90%	60%	100%	100%	60%	60%	70%	60%	90%	**90%**
Wood detected	92.3%	100%	92.3%	92.3%	100%	92.3%	92.3%	92.3%	92.3%	**100%**
Total	54.8%	46.8%	46.8%	66.1%	43.5%	54,8%	43.5%	61.3%	75.8%	**88.7%**

Table 3 - Comparison of the accuracy of the different methods

For the calculation of the parameters related with the gradient of brightness, a mask was not applied at first. The following chart compares results obtained when applying and when not applying a mask in those calculations.

After applying a mask, an erosion is used to make the object smaller and avoid problems with edges. Morphological gradient is applied afterwards.

Material to detect	**Decision tree without applying mask**	**Decision tree applying mask**
Glass	**14.3%**	**71.4%**
Metal	**76.9%**	**84.6%**
Paper	**100%**	**75%**
Plastic	**72%**	**100%**
Porcelain	**80%**	**90%**
Wood	**84.6%**	**100%**
Total detected	**74.2.7%**	**88.7%**

Table 4 - Comparison of accuracy of the last method with and without applying mask

As we can see in the previous chart, the highest accuracy achieved in this project is 88.7% when using the decision tree method after applying a mask and erosion, and using a low-order filter.

To have a more realistic evaluation of results, as previous detections were for images whose values were already in the databank, a test method is develop to simulate the analysis of new images.

This method is based on leaving one-half of the images out of the databank, and use them as sample images. Results for this method are very successful, as they reveal that the program would detect successfully different new images. These results of the test are shown in the following chart.

Material to detect	Successful detection
Glass	100%
Metal	85.7%
Paper	75%
Plastic	100%
Porcelain	100%
Wood	100%
Total detected	93%

Table 5 - Results of the test of accuracy

Notice that the limits for the ranges of each material are changed in this case to be adjusted only to the images in the databank, and not to the sample images that were previously in the databank.

Chapter 6. Conclusions and proposals for future development

6.1. Conclusions about the obtained results

As it was shown in the previous chapter, the highest accuracy achieved by the program is 88.7% due to many factors. Different conclusions emerge because of the results obtained during and after the elaboration of this project.

- Among the decision methods discussed in this work, the most successful is the **decision tree**, as it considers different ranges of parameters to classify the material.
- One limitation is that sometimes, pictures do not give information enough to make a precise decision, as objects can look as made up of other material. Examples for this **confusion** can show up when examining a bottle of water, or a coloured object made of glass.
- **Ranges** in the decision tree are set by hand after looking for interesting parameters and values for each material. They depend somehow on the programmer's criteria, so the effectiveness of the program can change. Wider limits lead to overlaps in the values for different parameters, and tighter limits may exclude possible values of a new image that would belong to that material, so a compromise may produce the best results.
- The **lack of more images** taken in the laboratory may be a cause for a lower accuracy, as more image could lead to different limits for the range that are not contemplated.
- **Different filters** produce different results, and they are dependent on many parameters. Type of window, cut-off normalized frequency, order and length of the impulse response are the most important, and produce numerous types of filters. However, the one used in this project tested its effectivity compared to many others.
- Difficulty to guess a material also depends on the **conditions** when the picture was taken. Pictures from the Internet are quite difficult to analyse, as they may be computer-made. On the other hand, photographs taken with a camera can be more easily analysed, of course depending on the light sources and on the background for transparent objects.

6.2. Proposals for future development

By way of conclusion, the following proposals would serve as a guideline for future developments

- Analysis of images using **different methods**, like plotting values for materials in different charts, depicting an area that contains all the in the chart and identifying possible materials when the points of new images are inside the area.
- A solution that could bring results that are more accurate would be **artificial intelligence**, using neural networks to recognize materials by discovering patterns in the images, or by learning the parameters for each material and comparing them with the new image in a more effective way than using a decision tree.
- Increase the **number of materials** that the program can detect, including cardboard, stone and fabric. Metal can also be divided into different types, like aluminium or copper among others.
- **Collect more pictures** and build a larger databank.

- Include the program into an industrial machine to **test its practical efficiency**.
- **Migrate the program** into C or C++, which are faster languages, to get immediate results in real applications.

Chapter 7. Bibliography

[1] - Course of 'Introduction to Digital Image Processing' at Széchenyi István University, András Horváth

[2] - Python Software Foundation, 'Python 2.7.13 documentation', https://docs.python.org/2/

[3] - OpenCV, 'OpenCV 3.0.0-dev documentation', http://docs.opencv.org/3.0-beta/index.html

[4] - SciPy.org, 'NumPy v1.12 Manual', https://docs.scipy.org/doc/numpy/index.html

[5] - Eric Gazoni and Charlie Clark, 'Openpyxl' - A Python library to read/write Excel 2010 xlsx/xlsm files'', https://openpyxl.readthedocs.io/en/default/

[6] - Matplotlib, 'matplotlib.pyplot', http://matplotlib.org/api/pyplot_api.html

[7] - Wikipedia, 'Cone cell', https://en.wikipedia.org/wiki/Cone_cell

[8] - GIMP homepage, 'The Free & Open Source Image Editor', https://www.gimp.org/