UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

Departamento de Comunicaciones

Final Project

Scanner application for the UPV Card to control attendance to class

FINAL PROJECT

Authors :

Peral de León, Jorge

Arévalo Jaramillo, Laura Marcela

Subject:

Tratamiento digital de imagen y video

Tutor/a:

Herranz Herruzo, José Ignacio

GANDIA, 2020-2021

Table of Content

Image Index

Tables Index

1 Introduction

This project came from the necessity of controlling and restricting the attendance of big groups of people due to the current situation involving COVID-19.

The project's main objective was to design and program a full-stack application, built-in specifically for the *Universitat Politécnica de Valéncia* (UPV), to facilitate the control of attendance in its classes, courses, etc.

To do this it was necessary to implement an SQLite database, to simulate the UPV's real one or as the base for a new one, and a function to manage to do queries with the data extracted from the card; apart from the main functions to detect and collect information on the cards.

The application is a real-time, name and QR-Code detector, designed to be placed outside the classrooms to grant or deny students access depending on the subject they are enrolled in, and also making available a table at the end of the day as an attendance list. For this, it would be necessary the placement of a screen and a webcam.

2 Graphical User Interface (GUI)

This section explains every component present in the user's interface shown in Illustration 1. Their purpose, how they work, which functions or callbacks they call if they do so, and where are they placed.

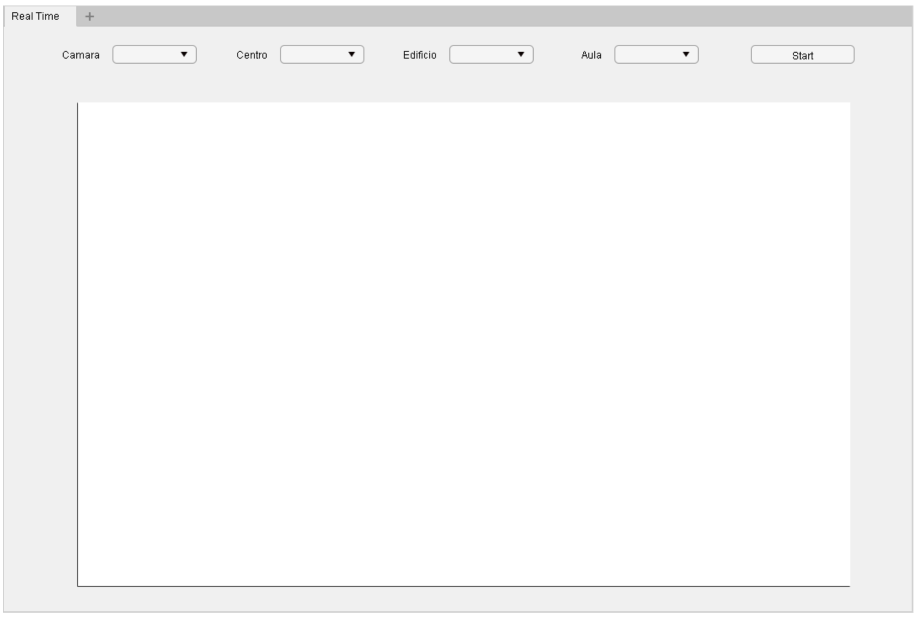


Illustration 1 UPV Scanner GUI main page

2.1 TabGroup

All the components present in the application are embedded in two TabGroups instead of using the main UIFigure. The reason for this is to make it easier to upgrade the application on future releases making available the creation of different tabs. These tabs could serve different purposes as configuration or camera control for mask-wearing.

2.2 UIAxes

The UIAxes Object is where the footage from the webcam is displayed. As Matlabs App Designer currently doesn't have video-oriented objects, to help the users interact with the application snapshots are taken from the webcam in an infinite loop and displayed on the UIAxes with a red rectangle placed on top to mark where the card should be placed. An example of how the video would display shows non the Illustration 2.

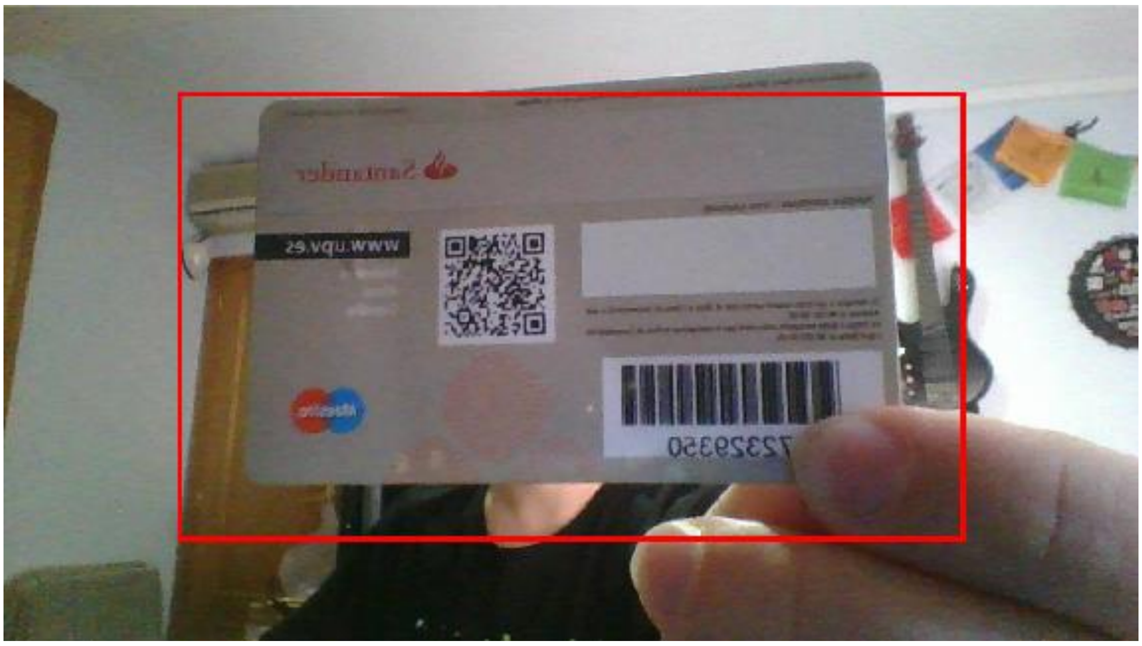


Illustration 2 Format of the video displayed on the UIAxes

2.3 Drop Down Menus

As the application is designed to function on an infinite loop and has no stop option other than closing it the dropdown menus are where the information to call the databases must be selected. There are four of them: Camera, Center, Building, and Classroom.

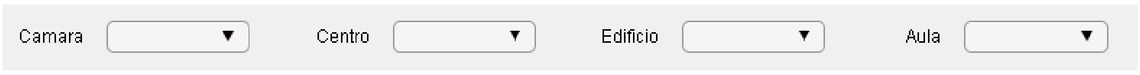


Illustration 3 Drop Down Menus on the main interface

2.3.1 Camera

A drop-down menu, labeled as 'Camera' that calls the function CamaraDropDownOpening function that calls to the Matlab function 'webcamlist' and changes its items to show the available webcams. This allows the user to select the camera that wants to use for the application, but all the cameras would work at the same resolution ('1280x720').

2.3.2 Center

It's the only drop-down menu that has its item values predefined. As it is designed exclusively for the UPV the possible values are EPSG, Vera, and Alcoy. Once the option is selected it calls to the CentroDropDownOpening function. The function called takes the value selected and queries the database to find out the available options for buildings on that campus.

2.3.3 Building And Classroom

The drop-down menu has its item values defined by the function CentroDropDownOpening, so the values are not available until the function is called. When the value is selected calls the function EdificioDropDownValueChanged to query for the available classrooms in that building to make them available on the classroom dropdown.

2.4 Start Button

As the values from the dropdown menus need to be configured before the UIAxes starts displaying an infinite loop of snapshots taken from the webcam it is necessary a button that indicates the program where to start. For this purpose the start button it's placed in the right corner.

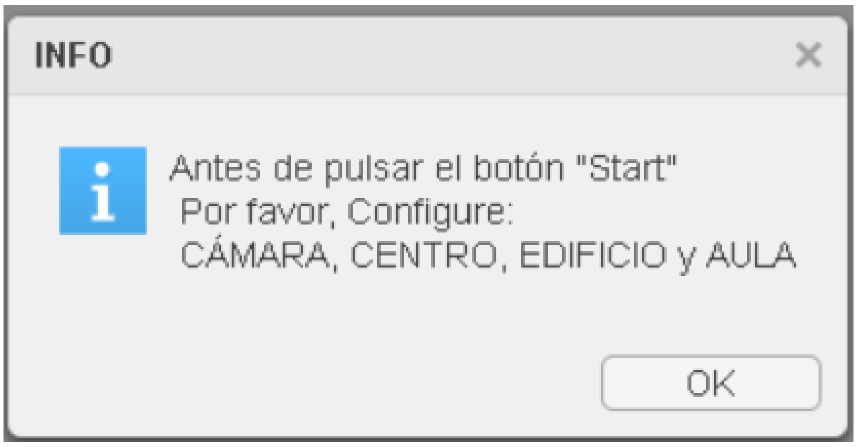
When the button is pushed calls the function StartButtonPushed.

2.5 UIAlert

UIAlerts are displayed on the screen and used to inform the user on different aspects of the startup configuration, and whether or not is granted to pass.

There are three of them: INFO, SUCCESS, and WARNING.

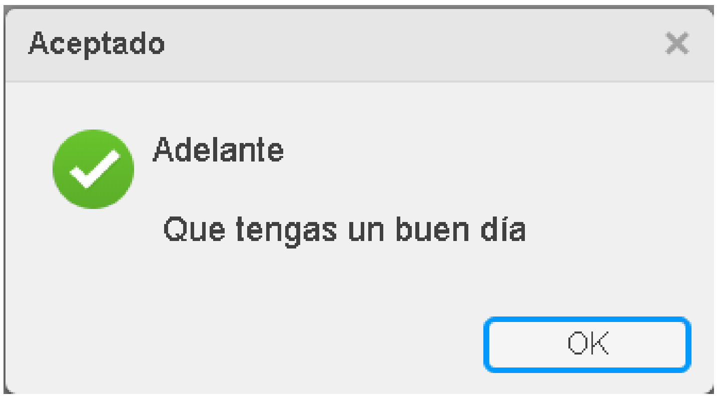
INFO, shown in Illustration 4 is displayed thanks to the StartUp function and informs the person/people in charge of configuring the program that at first he/she must configure the dropdown menus to the wanted values.


INFO

Antes de pulsar el botón "Start"
Por favor, Configure:
CÁMARA, CENTRO, EDIFICIO y AULA

Illustration 4 Info UIAlert screen used to inform the user on different aspects of the startup configuration

SUCCESS, shown in Illustration 5 screen used to inform the student that he/she can pass. Its trigger comes from the code returned by the function *ComprobarBD*. When the returned code is 200 the access is granted and the SUCCESS alert must be displayed.

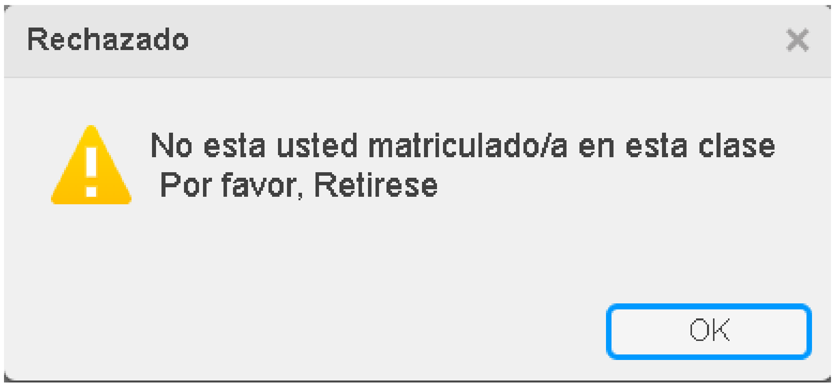

Aceptado

Adelante

Que tengas un buen día

Illustration 5 Success UIAlert screen used to inform the student that he/she can pass

Last but not least, WARNING (shown in Illustration 6). As SUCCESS depends on the return code from *ComprobarBD*, but in this case, the trigger codes are 204 (There are no classes in that classroom TODAY) or 401 (The student does not have access to the class).


Rechazado

No esta usted matriculado/a en esta clase
Por favor, Retirese

Illustration 6 Warning UIAlert screen used to inform the student that he can't pass

3 Code

This section is a review of the key line son the code of the different functions implemented on the application.

3.1 *UPVScanner*

Review on the main functions of the mlapp file out from the object creation statements, and their flowchart.

3.1.1 Properties

Private properties of the application:

- webcamObject: Where the value of the webcam selected by camera dropdown will be placed.
- imageObject: Object where the image of UIAxes will be saved.
- Sprintf():
- Sqlite(): would be saved in the 'conn' variable

3.1.2 startupFcn

This function is called when the UIFigure is created. It only has one statement which is:

- uialert(): displays an alert dialog box, with an info Icon, informing the person/people in charge of configuring the program that at first the dropdown menus must be configured to the desired values.

3.1.3 CamaraDropDownOpening

This function only returns the values from the function webcamlist and places them on the possible values on the camera selection dropdown.

3.1.4 CentroDropDownOpening & EdificioDropDownValueChanged

- fetch(con,SQL query): Imports the results form the queries to the database saved in app.conn. The SQL queries are:
  - In CentroDropDownOpening the query looks for the buildings on the table 'Aulas' where the center is equal to the selected value on the dropdown.
  - In EdificioDropDownValueChanged the query looks for the classrooms on the table 'Aulas' where the building is equal to the selected value on the dropdown.

- cat(): fetch function to the SQLite database returns a cell array with the values on the table separated by columns. To work only with the data necessary this function is used to eliminate the not desired columns and resizing the array to a single column. An example of how the output values are displayed in Table 1.

Output from fetch	Output on CentroDropDown Opening	Output on EdificioDropDownValueChanged
{'EPSG'} {'1A'} {'10'}	{'1A'}	{'10'}
{'EPSG'} {'1A'} {'11'}	{'1A'}	{'11'}
{'EPSG'} {'1A'} {'9' }	{'1A'}	{'9' }
{'EPSG'} {'1B'} {'20'}	{'1B'}	{'20'}

Table 1 Outputs from fetch function and cat function on the different DropDown Values

- unique(): As SQL-based databases cannot have multiple values attached to one when querying most of the values returned are the same. To only display one time each of the different values for classes and buildings the function unique is called. This function returns a single array where the values are not repeated.

3.1.5 StartButtonPushed

It's the main function of the application due to being the one in charge of start displaying and processing the images taken from the webcam to identify the UPV student cards. The most important parts of its code are:

- snapshot(): Takes snapshots from the webcam stored in the app webcam object.
- insertShape(): Places a red rectangle, with the following dimensions [200 100 880 500], on top of the snapshot to mark where the users must place their card to be detected as shown in Illustration 2.
- imshow(): This function displays the resulting image from the function insertShape as paren ton the UIAxes object making the images visible on it as a video.
- detectorQR(): The function is the one in charge of detecting if there is a QR -code. It returns the cropped image of the QR-code if there is one, else it returns an empty string.
- ExtractorDatosQR(): This function returns the name, ID, state of the card (active or expired), and the center the student is registered. This is done by taking the cropped image resulted from the function detectorQR, decoding the HTML encoded in the QR-code, and taking the information needed from the URL.
- ComprobarBD(): With data extracted from the recoNombre or the ExtractorDatosQR calls the database with the different values to extract if the student should or not be granted access to the class, returning a code as a response.
- recoNombre: this function makes it possible to detect the characters corresponding to the student's name on the front of the card.
- uialert(): Depending on the code returned by ComprobarBD it displays a response in the UIFigure. Said responses are shown in Illustration 5 and Illustration 6.

3.2 DetectorQR

This function is responsible for identifying whether or not there is a QR code in the images extracted from the webcam and if there are any, and cuts where said code is. An example of its input and output are shown in Illustration 7 and Illustration 8.

Its main functions are:

- imbinarize(): To detect a QR code (BLACK) and separate it from the background of the card (dark gray), a high threshold of 0.85 is assumed using the imbinarize function to convert the resulting grayscale image, from the rgb2gray function to a logical/binary (0 or 1) image.
- strel(): The structuring element (SE) is defined as a 3x3 pixel square, since the QR-code patterns are very small elements and with not very high resolution. So if the SE were larger it would unite some objects or erase them.
- imopen(): This operator mixes erode and dilate by eroding the object and then filling in the small details. Done with the SE explained before.
- imcomplement(): saves the complement of the image after the open operation to mark the QR-code as objects.
- bwpropfilt: Extracts objects from binary images using their properties. This function is used twice. The first time keeps only the first 50 objects with higher robustness, to remove objects with branches or too flattened. And last, it is used

to keep only the first 6 objects with greater área, to eliminate small objects that may have been left.

- regionprops(): It is used to measure the eccentricity and the centroid properties, and the minimum properties that would be given with a caliper (MinFeretProperties) that would be used to differentiate between objects.
- imcrop: This function is called twice:
  - First, at the beginning where it crops the image to the size marked by the red rectangle in the application to eliminate the background and set a similar size for all photos.
  - Last, when it crops the QR-code from the first cropped image to return it to the function ExtractorDatosQR.

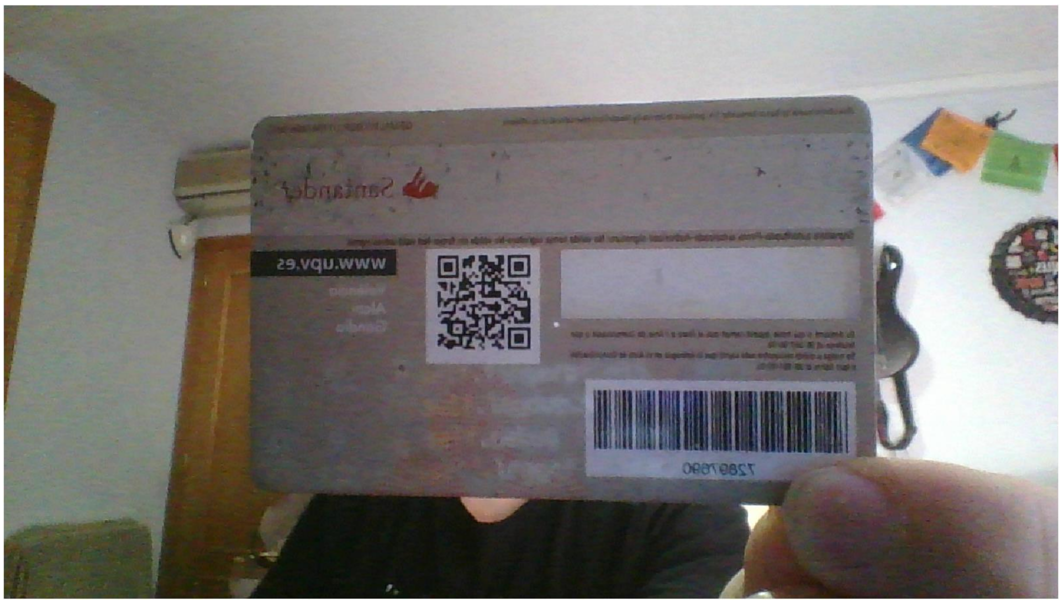


Illustration 7 Input example for the function DetectorQR

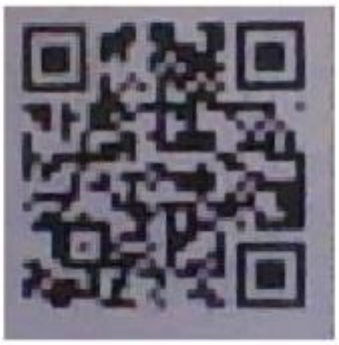


Illustration 8 Output example for the function DetectorQR

3.3 ExtractorDatosQR

This function is used to extract all the information from the QR-code. This function returns the name, ID, state of the card (active or expired), and the center the student is registered. This is done by taking the cropped image resulted from the function detectorQR, decoding the HTML encoded in the QR-code, and taking the information needed from the URL. It accepts different formats as inputs (text or uint8). The main functions used are:

- readBarcode(): This function gets an image of the cropped QR-code and decodes it. If the QR code is readable, this action returns a string with the URL of the student's data page. Else it returns an empty string. This last statement would end the function.
- webread(): Given the URL by the readBarcode function it returns all the HTML code from the said URL. The code returned would be where all the information from the student would be taken.
- findElement(): Finds in the HTML tree the table elements, where all the student information is embedded, saving it in a subtree.
- extractHTMLText(): Extracts the text from the tables subtree.

In the end, it would give a response like the one in Table 2.

```
Nombre =
   "Andres Madariaga Revuelta"
DNI =
   "72175198"
Estado =
   "ACTIVO"
Centro =
   "EPSG"
```

Table 2 Example of output from the function ExtractorDatosQR

3.4 RecoNombre

This function checks the name of the student on the front of the card, as an alternative to QR code recognition. A camera frame must be entered as input, and the output is a text string with the full name.

3.5 ComprobarBD

The function takes the Center, Building, and Classroom, where the subject is carried out, from the application's interface, from the dropdown values and queries the database with that data and the actual hour and day at the moment. But the student's name is taken from the ExtractorQR or recoNombre functions.

The function mainly returns four different codes:

- 000: Null input values.
- 204: There are no classes in that classroom today.
- 401: The student does not have classes at that hour in that classroom or just today.
- 200: The student has class at the time and has been granted access.

3.6 Test Database

It’s mainly a mock database to act like the UPVs real one or as a foundation base for a new one. There are three tables on the database:

- Alumno (Student)
- Aula (Classroom)
- Asignatura (Subject)

They have the following data:

Table	Columns	Primary Keys
Alumno	nombre, dni	nombre, dni
Aula	centro, edificio, numero	centro,edificio,numero
Asignatura	nombre,dia,inicio,final,centro,edificio,aula,alumno	____________________

Table 3 Databases tables, columns, and primary keys

In Asignatura the columns stand for:

- Nombre: Name of the subject.
- Dia: Day of the year between 1 and 366. For example, the date 08/02/2021 would be 39.
- Inicio and final: The hours at which the subject starts and ends.

4 Description Of The Algorithm

4.1 DetectorQR

First, the function is given as input a snapshot taken by the webcam as is displayed on Illustration 9.

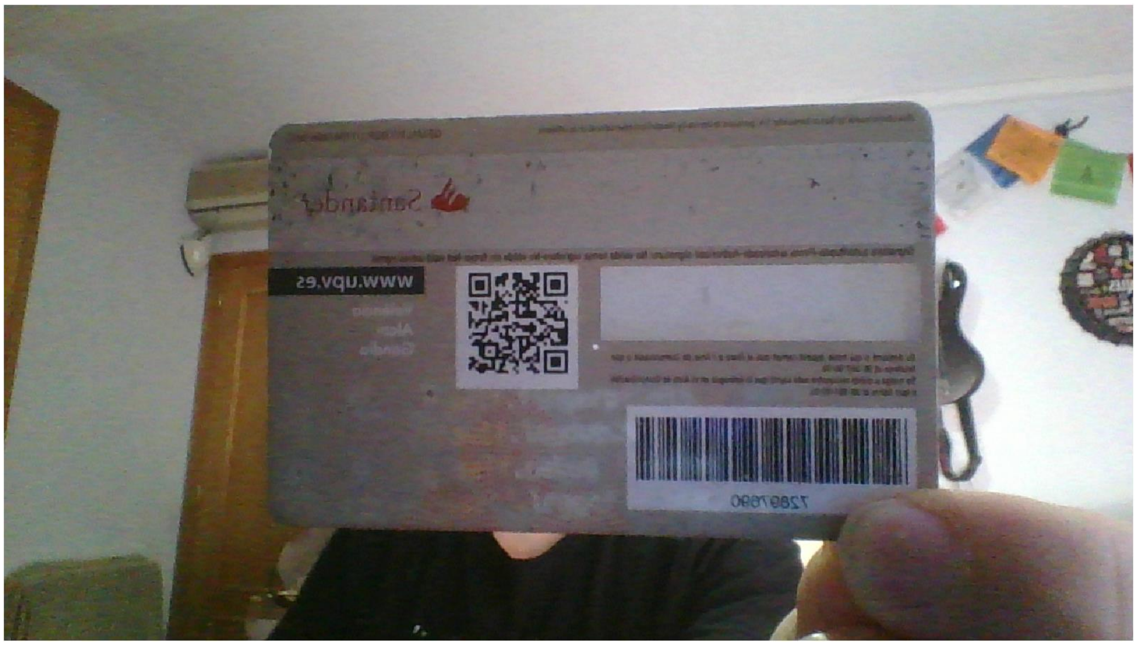


Illustration 9 Image taken by the webcam and placed between the red margins indicated on the interface

After that, the image is cropped in a rectangle equal to the one displayed in the main interface as in Illustration 2. This is done to eliminate the background and set a similar size for all photos. An example of the image cropped in Illustration 10.

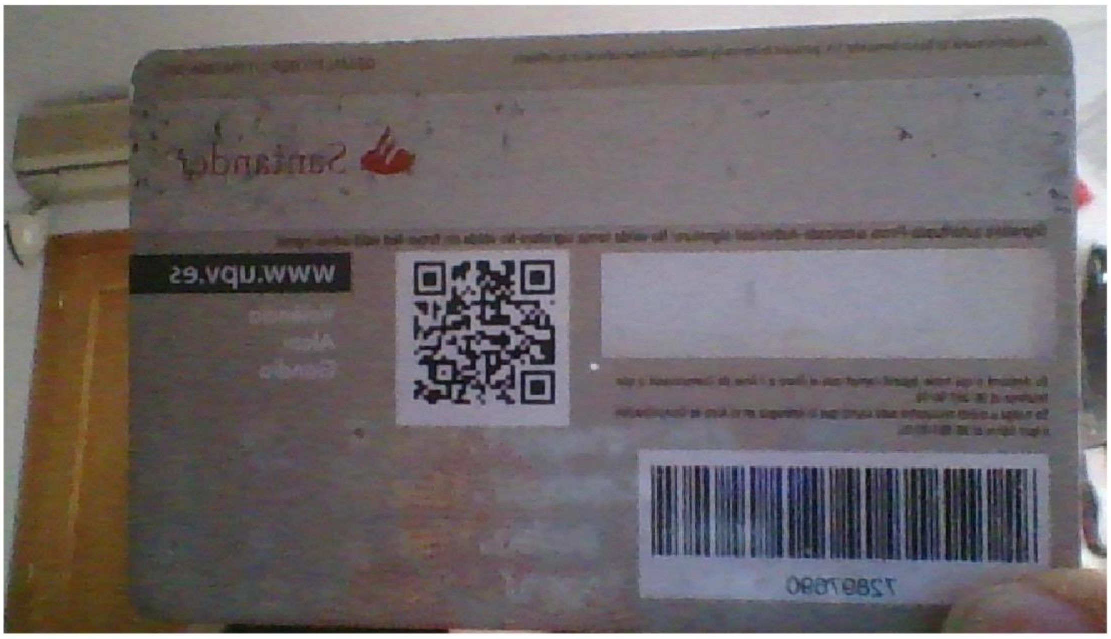


Illustration 10 Cropped image between the margins indicated on the interface

After being cropped the image is transformed into a grayscale image as shown in Illustration 11. This image is turned to grayscale to be turned into black and white images as shown in Illustration 12.

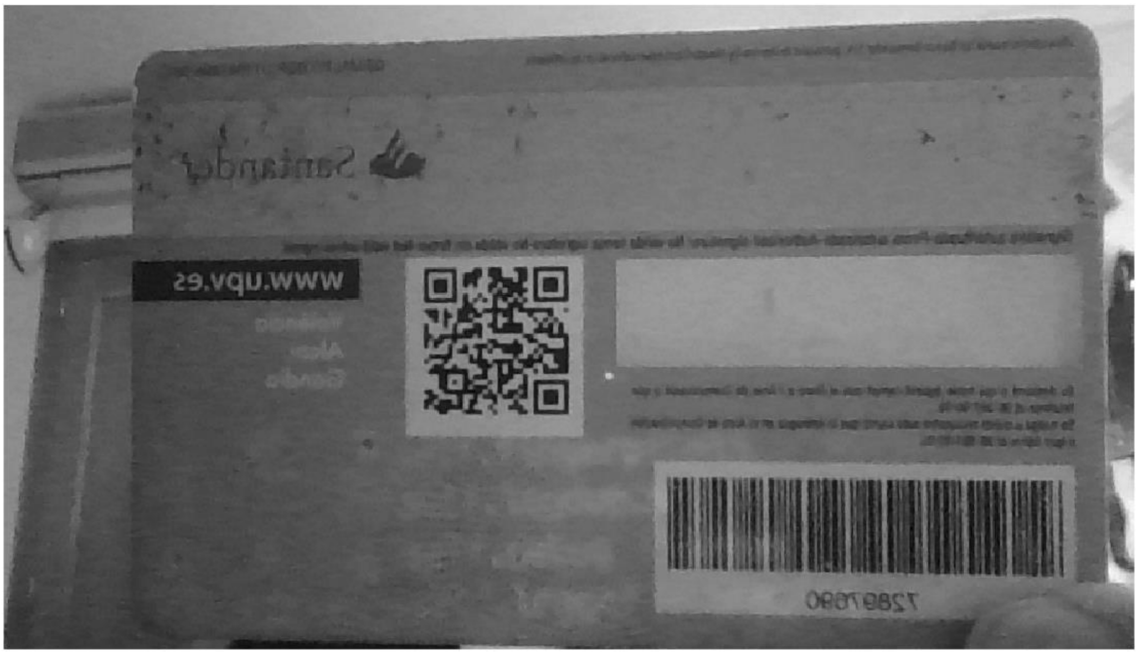


Illustration 11 Cropped RGB image turned to a grayscale image

After being turned into grayscale it must be turned into a binary image, for the later detection of objects. For this, the imbinarize function is used with an adapted sensitivity and the threshold being placed at 0.85. Due to the QR-code being black over a dark gray background, a high threshold must be selected. The resulting image of the operation is Illustration 12.

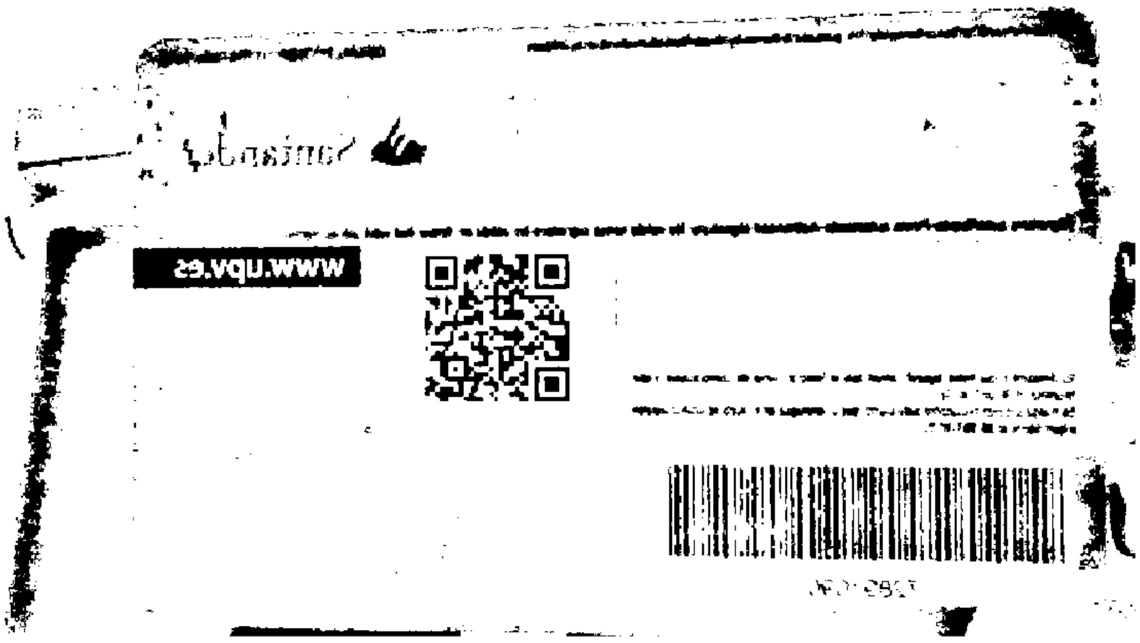


Illustration 12 Binarazed image after being passed to grayscale by the rgb2gray function

For detection and labeling reasons for objects to be detected they must be White, otherwise, they would be detected as background. Due to this the black and White image is turned to its complementary shown in Illustration 13.

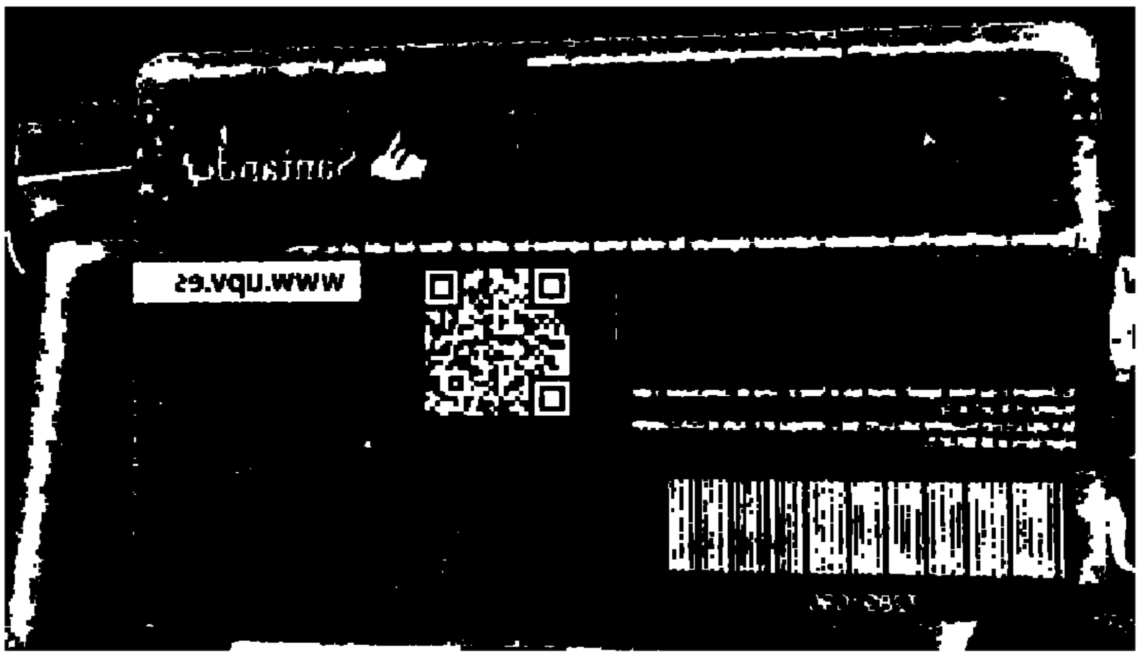


Illustration 13 Complementary image of the card after being imbinarize

After being turned to its negative image, an open operation is applied to the image to eliminate possible unions between the patterns and their bounding boxes, as shown in Illustration 14.

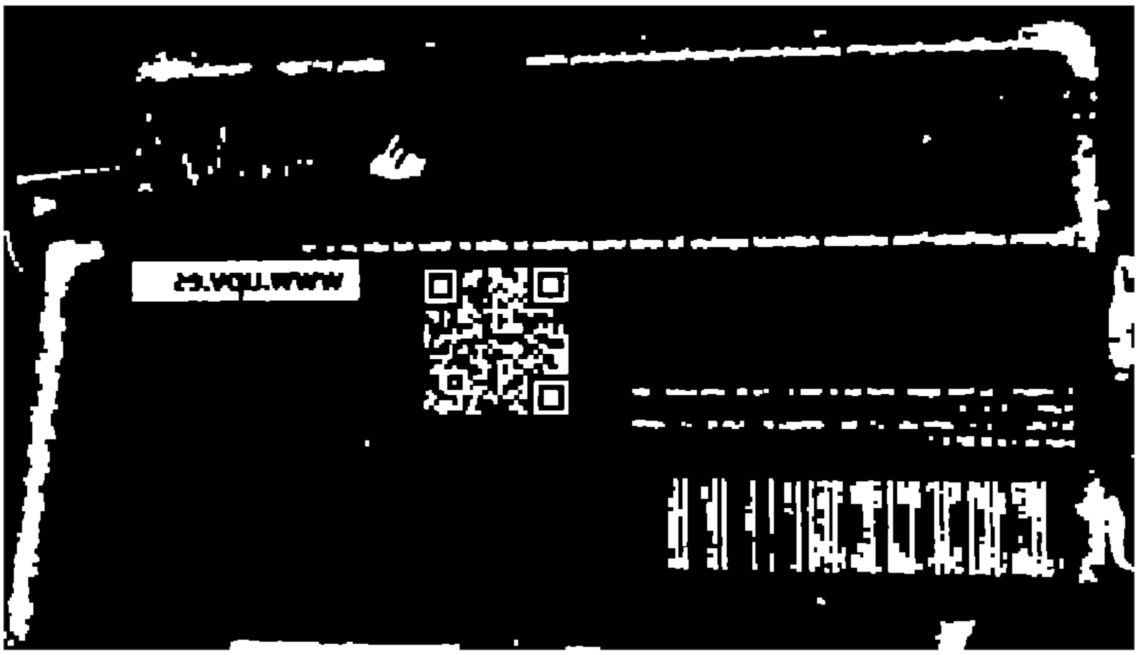


Illustration 14 Black and white image after being processed by an open operation

After the open operation, the objects in the image are filtered by the bwpropfilt by their solidity, and only living the first 50 with the greater one. The solidity of the QR-code location patterns is almost one due to its area and convex área being almost the same, calculated by the equation:

$$Solidity = \frac{Area}{Convex\ Area}$$

The resulting image is displayed in Illustration 15.

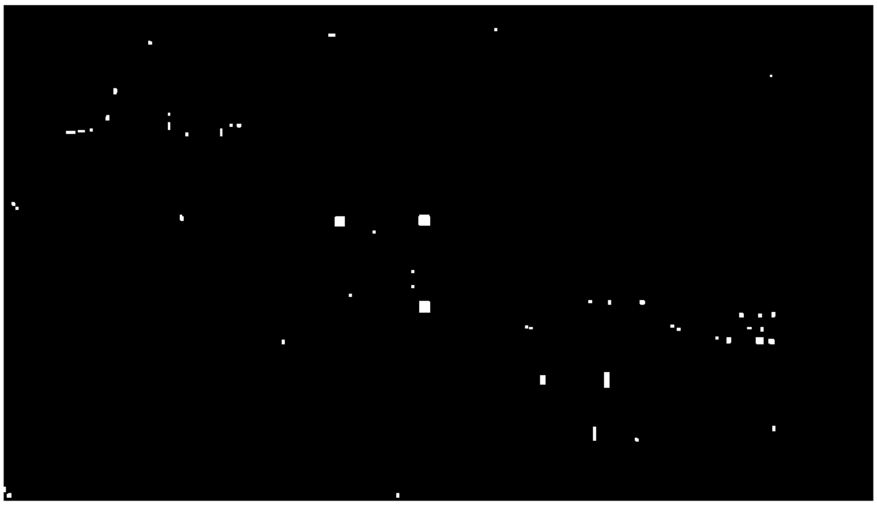


Illustration 15 Objects present on the image when taken the fifty with the biggest solidity through the function bwpropfilt

To avoid possible mistakes a second call to the bwpropfilt is made but this time using the areas and taking only the first 6 objects with the biggest ones.

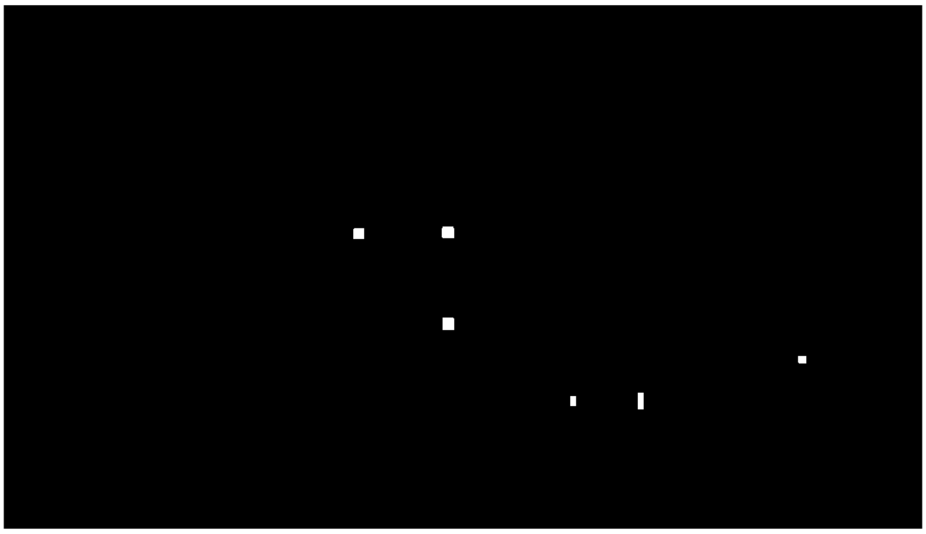


Illustration 16 Objects present on the image when taken the six with the biggest areas through the function bwpropfilt

At last, it instantiates the objects where the number of patterns and the position of their centroids will be stored. On a for loop from first to the last object saved looks for the objects with its minimum Feret diameter is between 10 and 11, and has an eccentricity between 0.19 and 0.45, the position of its centroid is saved and 1 is added to the number of patterns. That would return an array of positions x and y of the objects like the one shown in Table 4.

	1	2	3
1	[341.041666666667 219.041666666667]	[426.542635658915 218.038759689922]	[426.961832061069 305.541984732824]

Table 4 Example of centroid positions returned by DetectorQR

If the number of patterns collected that match those specifications are greater or less than 3 it is understood that they are not the QR code patterns and an empty string is returned.

With the position of the centroids, each of the parameters that will be passed to the imcrop function is calculated. This is done by taking the x and y positions of the centroid of each object. Then, the minimum x and y of the 3 are taken (the minimum will always be the upper left pattern).

At last, the height and width of what is to be trimmed are calculated, adding a margin of the centroids. This is done by following the next equation.

$$h = max([y1\ y2\ y3]) - min([y1\ y2\ y3]) + 45;$$

$$w = \max([x1\ x2\ x3]) - \min([x1\ x2\ x3]) + 45;$$

In Illustration 17 a labeled photo is shown, to demonstrate where the program detects that the QR-code patterns are.

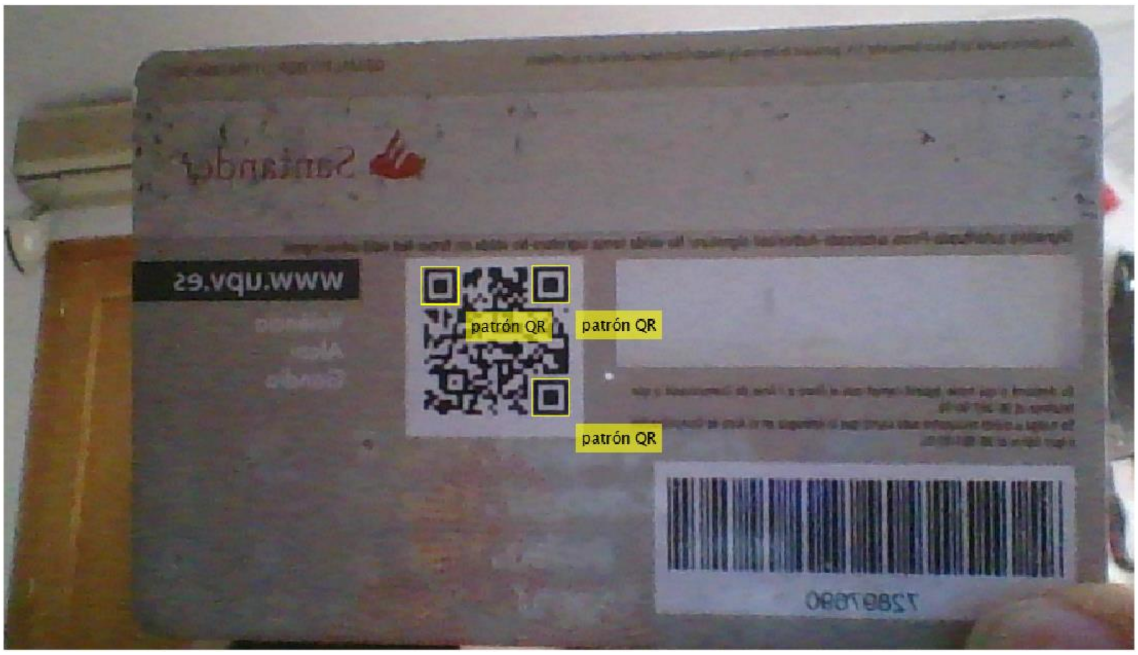


Illustration 17 Marked recognition patterns from QR detected by detectorQR function

With the positions calculated in the last paragraphs, the function imcrop is called following the next statement.

$$[min(x1\ x2\ x3)\ min(y1\ y2\ y3)\ width\ height]$$

So the final result and what the function returns is shown in Illustration 18.

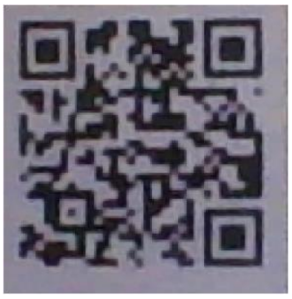
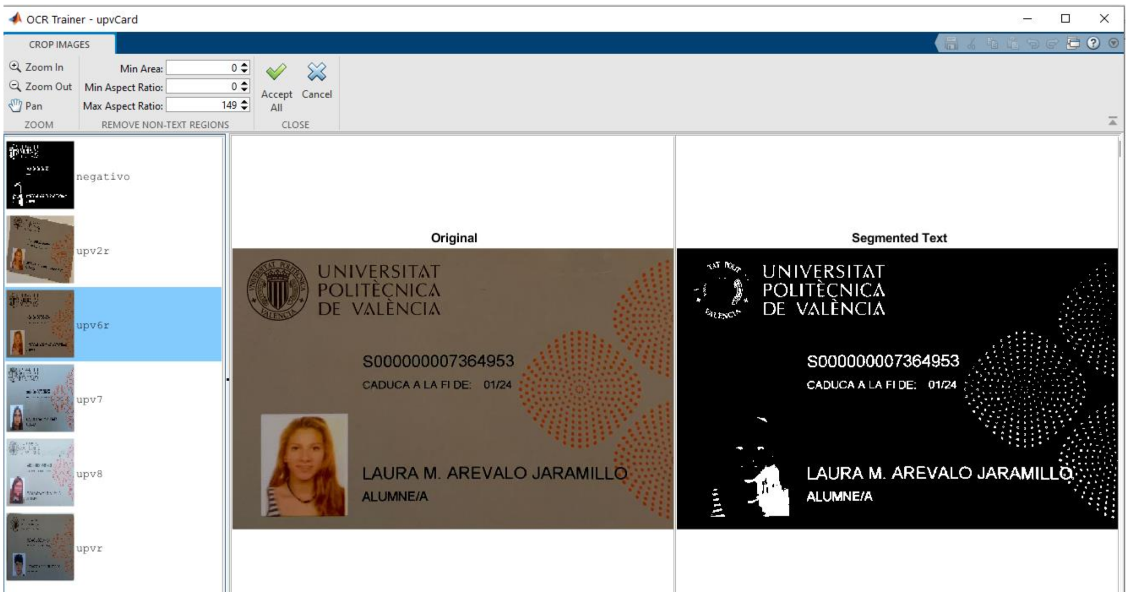


Illustration 18 QR-code cropped by the detectorQR function

4.2 RecoNombre

In addition to the detection of the QR code on the card, we have also considered the possibility of detecting the name on the front of the card. To do this, we use the OCR tools included in Matlab. First, we loaded images of different UPV cards into the OCR Trainer application to train the system and obtain our character recognition language.

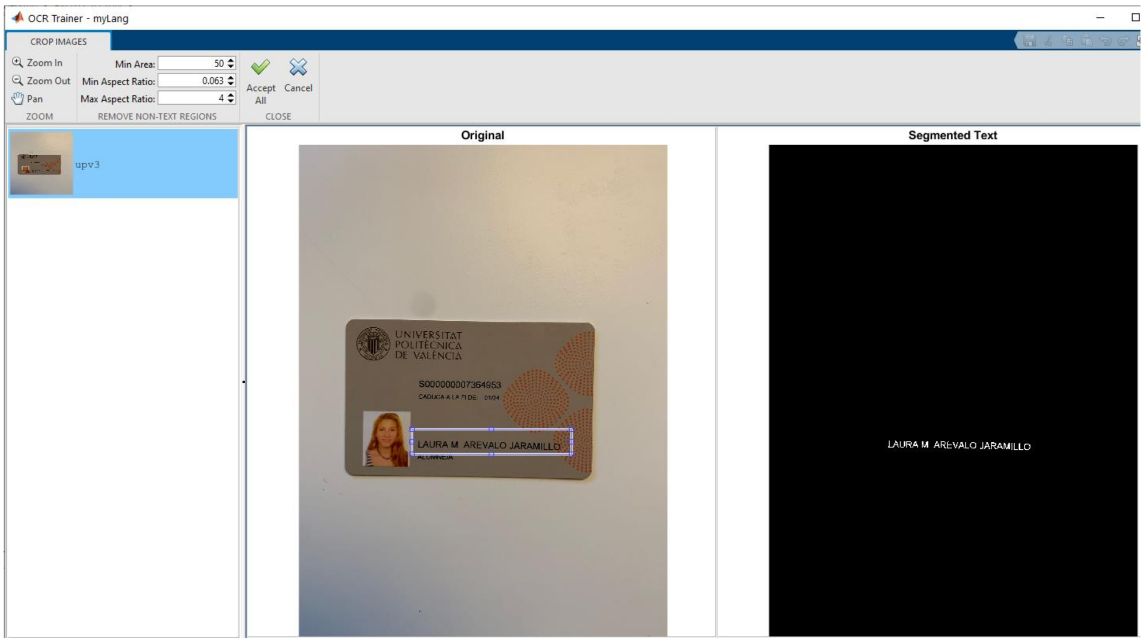


Min Area: 0
Min Aspect Ratio: 0
Max Aspect Ratio: 149

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 19 OCR Trainer App

The application automatically thresholds the input images and also allows us, optionally, to choose an area of interest ROI in which the text is located. This is very useful with images in which the thresholding does not give a good result, being able to work only in the area of the text.



Min Area: 50
Min Aspect Ratio: 0.063
Max Aspect Ratio: 4

LAURA M AREVALO JARAMILLO

Illustration 20 ROI in OCR Trainer App thresholding

Once they are ready, we can tell them to start training.

In a first analysis, the application already detects most of the letters but makes numerous errors. It is necessary to check all the results, reject the detected characters and reassign the wrong ones.

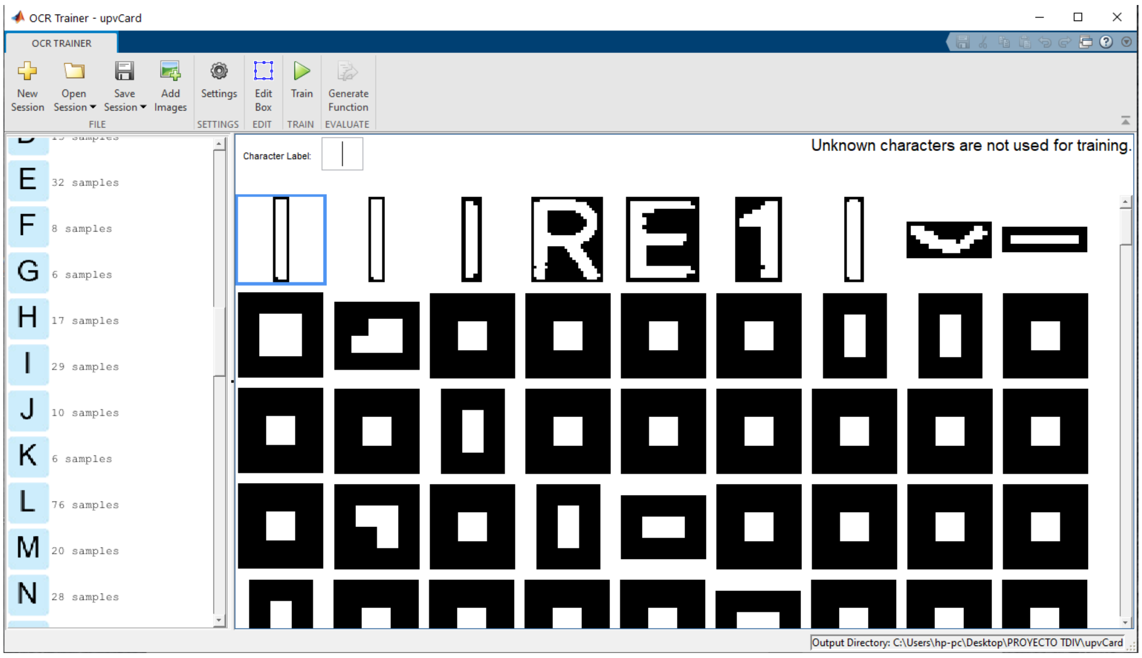


Illustration 21 First automatic detection

For example, the letter "A" has been detected as the symbol "/". We must correct it and add it to its corresponding character.

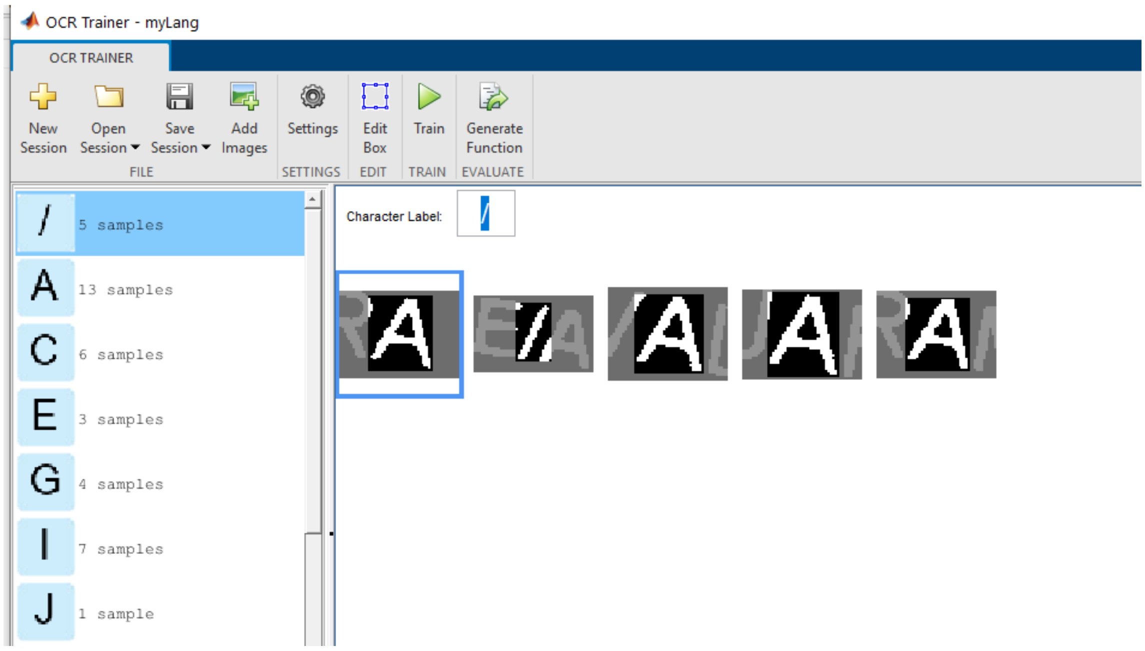


Illustration 22 Wrong character detection

When we consider that there are enough samples of each letter, we save the results and a code is automatically generated for use in the Matlab console.

```
function [ocrI, results] = upvCardOCR(I, roi)
% Location of trained OCR language data
trainedLanguage = 'C:\Users\hp-pc\Desktop\PROYECTO
TDIV\upvCard\tessdata\upvCard.traineddata';
layout = 'Block';
if nargin == 2
    results = ocr(I, roi, ...
        'Language', trainedLanguage, ...
        'TextLayout', layout);
else
    results = ocr(I, ...
        'Language', trainedLanguage, ...
        'TextLayout', layout);
end
```

Table 5 Matlab Code: auto-function to use an OCR trained language

This function can be applied directly to an image, but also allows working only on an ROI area of interest. It returns a text result and a graphic result marked on the input image itself.

Before proceeding with OCR recognition, the image has to be prepared. The first thing we have to do is to detect where the card is located. To do this, we threshold the image, remove the small objects (the card will theoretically be the largest object), and fill in the gaps. This gives us the card frame, which we can locate and use to crop the image:

```
%Escala de grises
I=rgb2gray(I);
%Umbralización
T = adaptthresh(I,0.4,'ForegroundPolarity','dark');
BW=imbinarize(I,T);
%Negativo para tener objetos en vez de fondo
BW1=imcomplement(BW);
%Rellenar huecos y eliminar objetos pequeños
BW2=imfill(BW1,'holes');
BW3=bwareaopen(BW2,100000);
%Detectar el BoundingBox y utilizarlo para recortar
s = regionprops(BW3,'BoundingBox');
RBW=imcrop(I,s.BoundingBox);
```

Table 6 Matlab Code: Image preparation

Here you can see the different phases of the code:

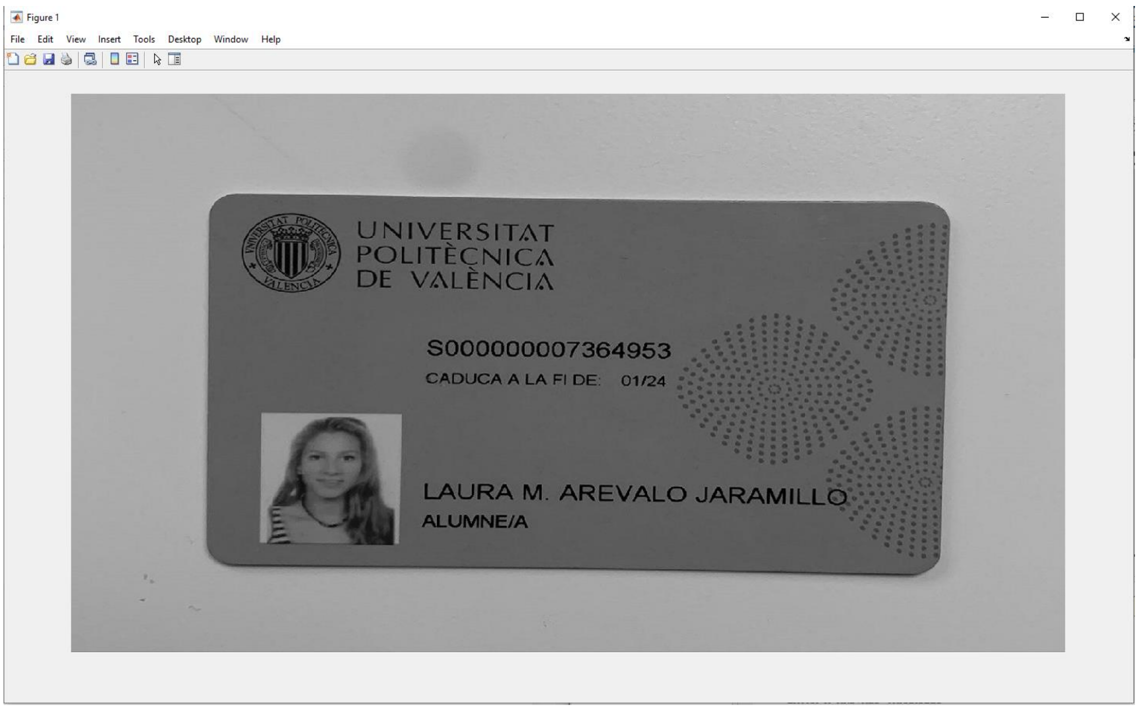


Figure 1
File Edit View Insert Tools Desktop Window Help
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 23 Original image

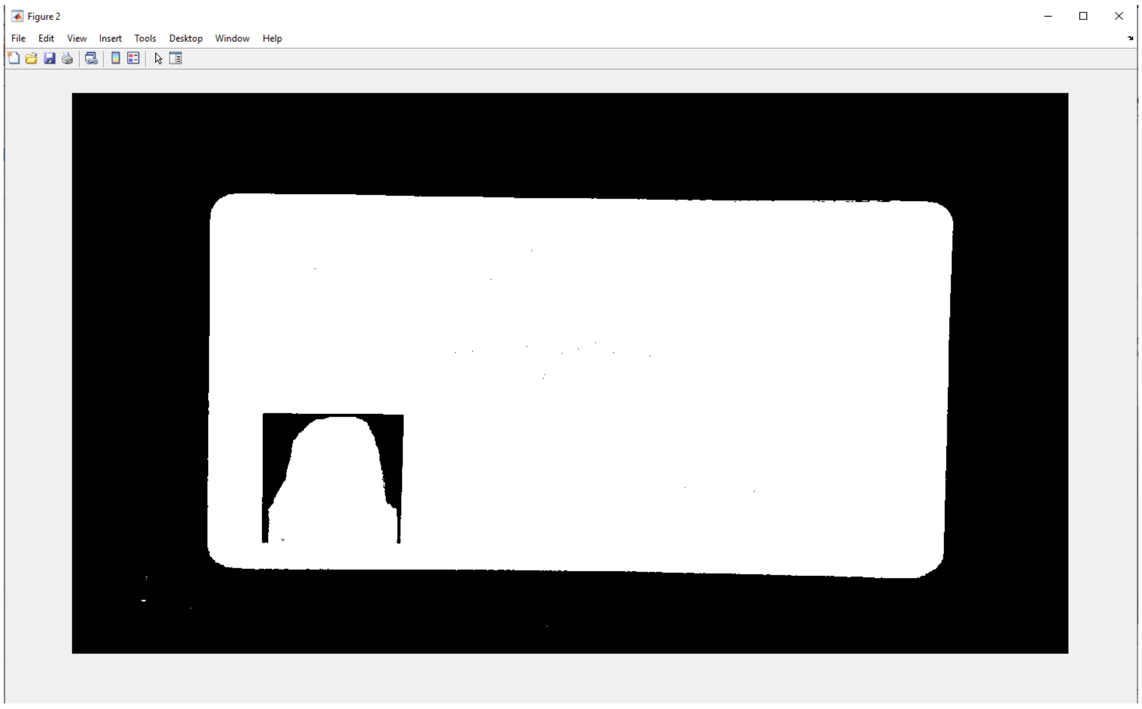


Illustration 24 Threshold image

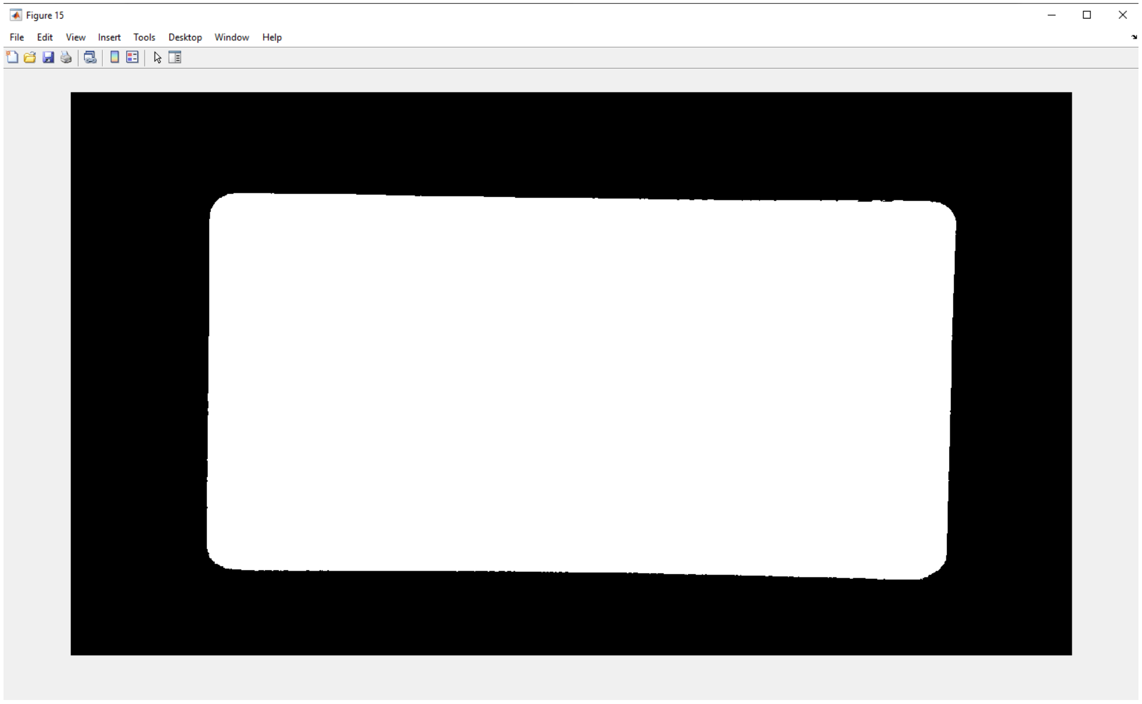


Illustration 25 Gap filling and removal of small objects.

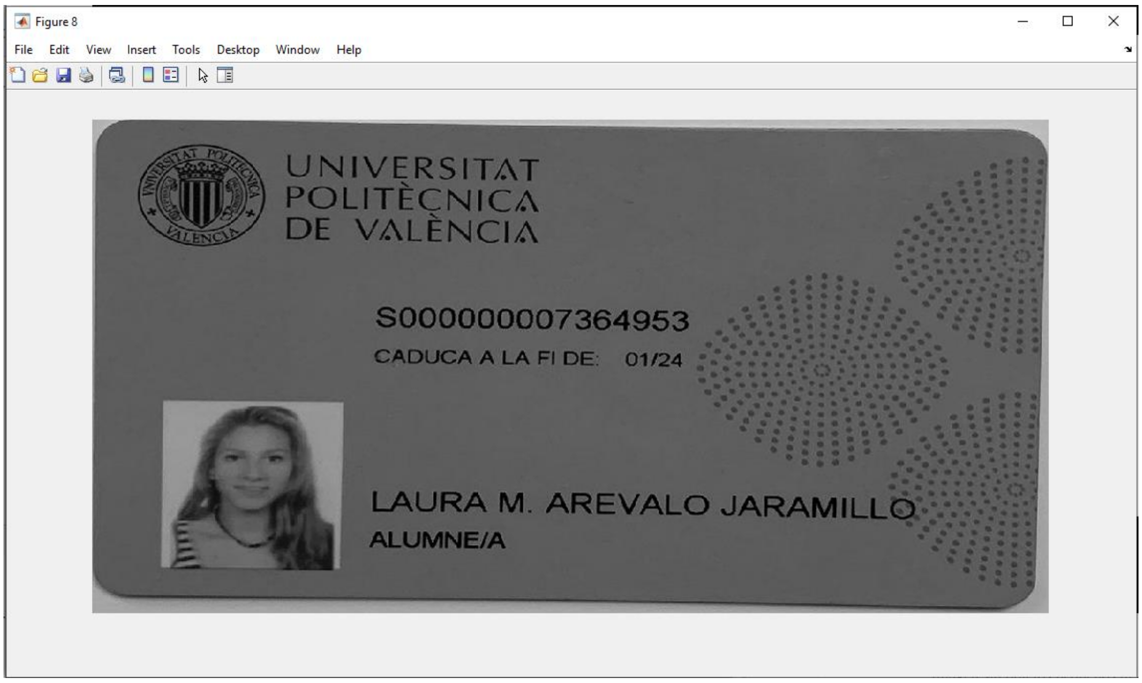


Figure 8
File Edit View Insert Tools Desktop Window Help
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 26 Image cropped according to BoundingBox

Once cropped, we perform again a sensitive segmentation to obtain the name correctly. We use local thresholding to avoid possible problems due to lighting faults. Once this is done, we invert the image to be able to work with these objects.

```
%Escala de grises
gris=rgb2gray(I);
%Umbralización
T = adaptthresh(gris,0.4,'ForegroundPolarity','dark');
segm= imbinarize(gris,T);
%Negativo de la imagen
nega=imcomplement(segm);
```

Table 7 Matlab Code: Image preparation for name detection

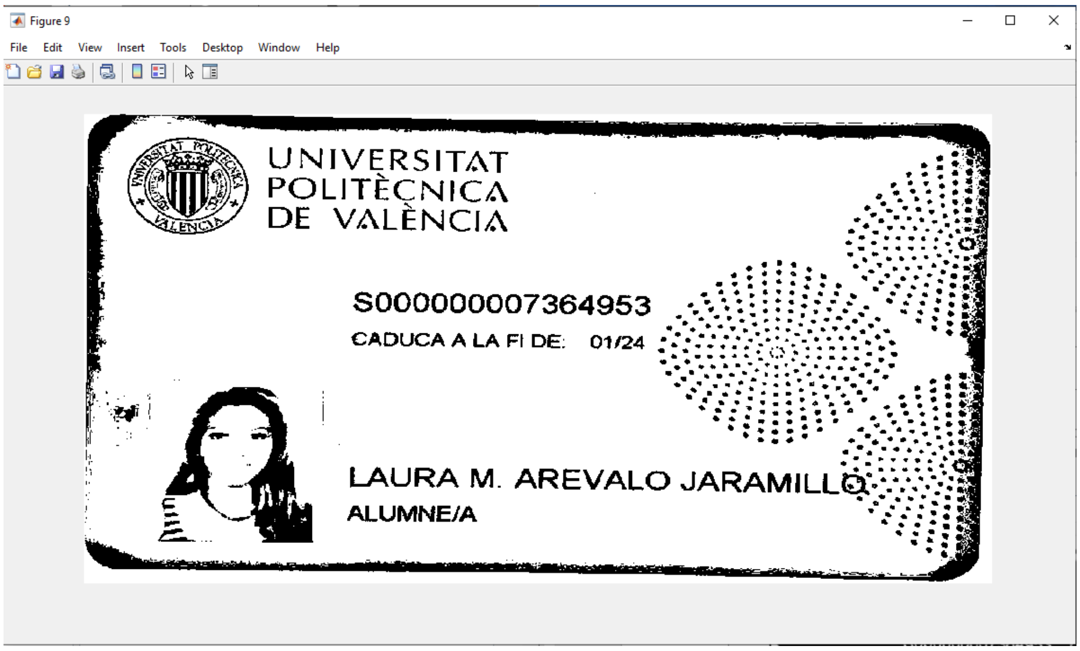


Figure 9
File Edit View Insert Tools Desktop Window Help
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 27 Image with local thresholding

We always have to get the inverse of the result so that our objects are real objects and not background (objects in white color, by default MatLab, defines them as background).

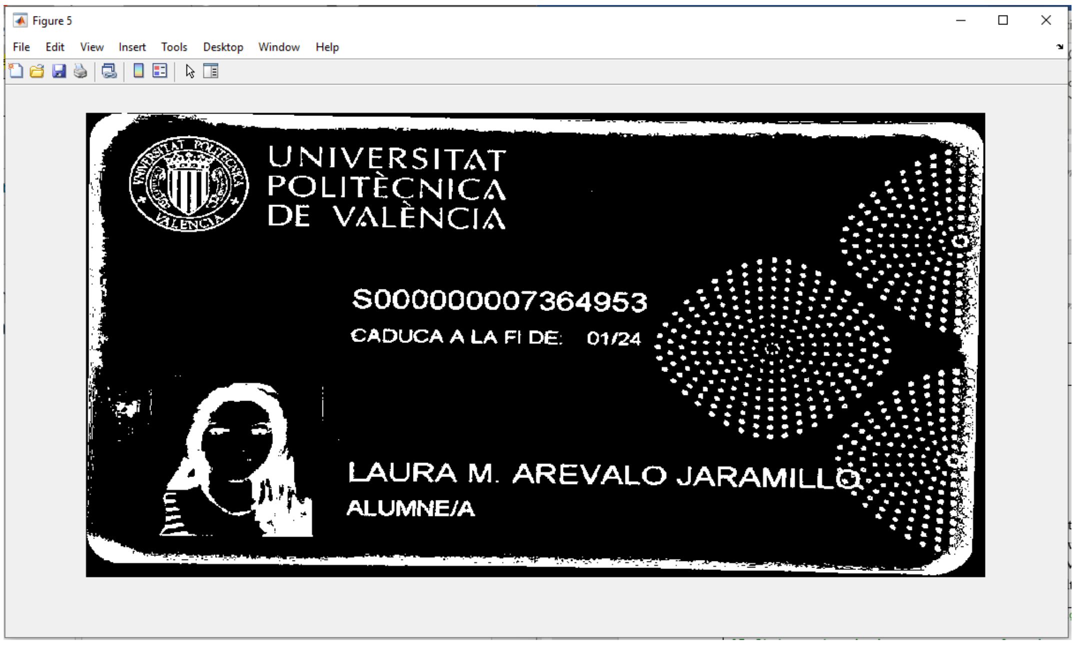


Figure 5
File Edit View Insert Tools Desktop Window Help
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 28 Negative image

As we only intend to detect the user's name, we are interested in delimiting the area where it is located. To do this we rescale the image so that it always has the same size and we look for the area of the name. With the Rectangle tool of Matlab, we can mark the ROI on the image and get back its position and size. The result is as follows.

```
%Reescalado de la imagen para que siempre tenga el mismo tamaño
res=imresize(sup,[500 850]);
%Definir region donde se encuentra el nombre
roi=[220,350,510,80];
```

Table 8 Matlab Code: Image resize and ROI definition

The last step is to run the function that created the OCR Trainer application and evaluate the results.

```
%Reconocimiento de caracteres
[ocrI, results] = upvCardOCR(res, roi);
%Resultado en texto
results.Text
%Resultado de manera gráfica
figure
imshow(ocrI)
```

Table 9 Matlab Code: OCR text recognition

5 Results

5.1 DetectorQR

The code performance is Good but since it's a real-time QR-code detector and it is designed to be working on open areas. So the position of the card and the angle it is placed affects how the objects extracted are perceived.

On one hand, there are times where the QR-code patterns are filtered by the bwpropfilt so the detection is not possible. An example of this kind of performance is shown in Illustration 29 and Illustration 30. In this case, it is due to the detection of more objects on the background with a greater solidity than the QR-code patterns.

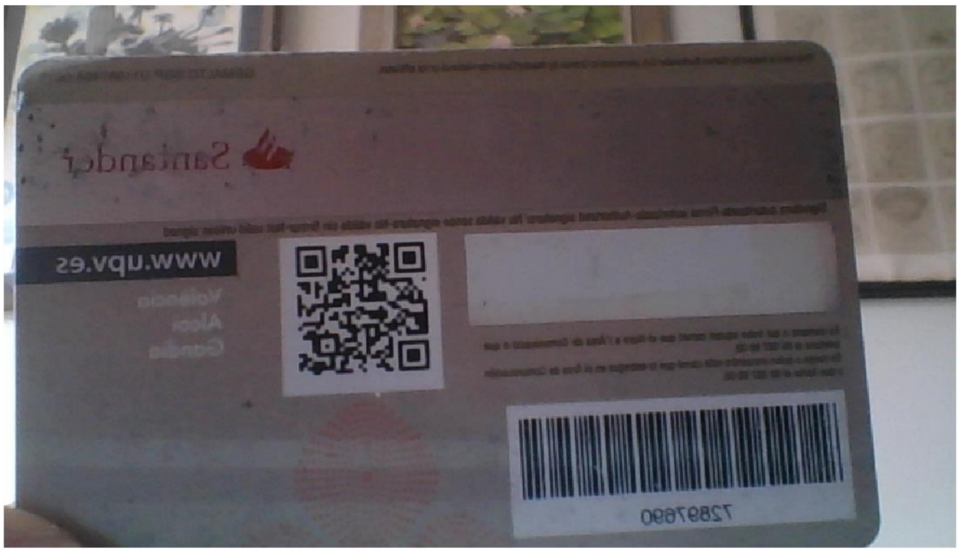


Illustration 29 No pattern detection original photo

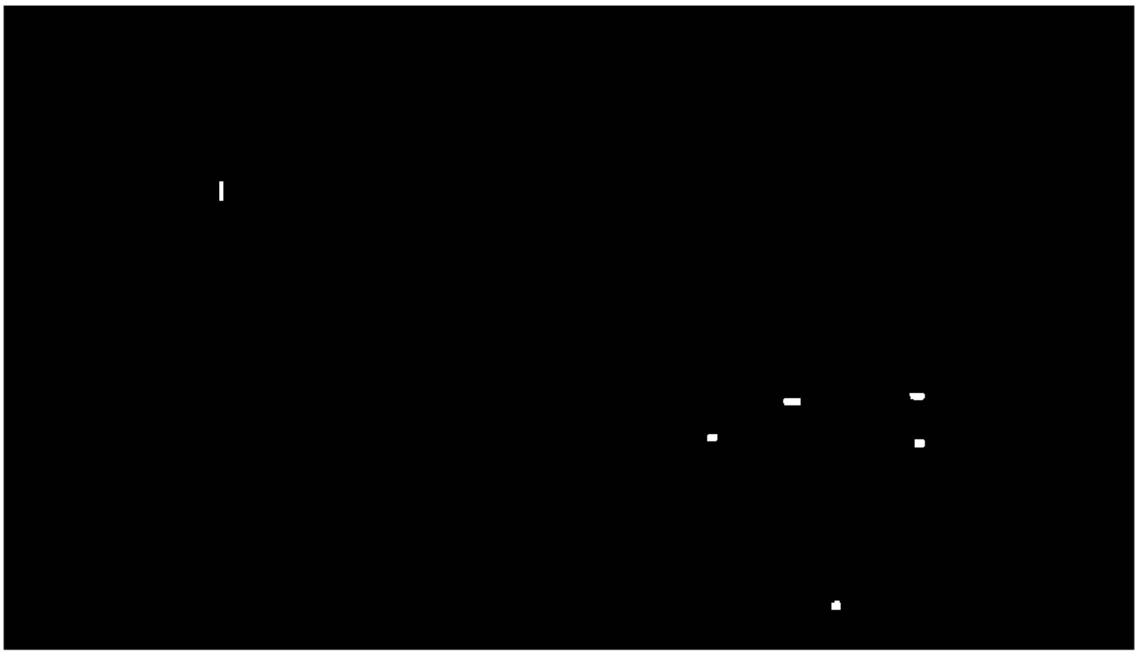


Illustration 30 No pattern detection black and white objects photo

On the other hand, there are times where not every pattern of the QR-code is detected. In Illustration 31 and Illustration 32 the only pattern detected is the bottom right one.

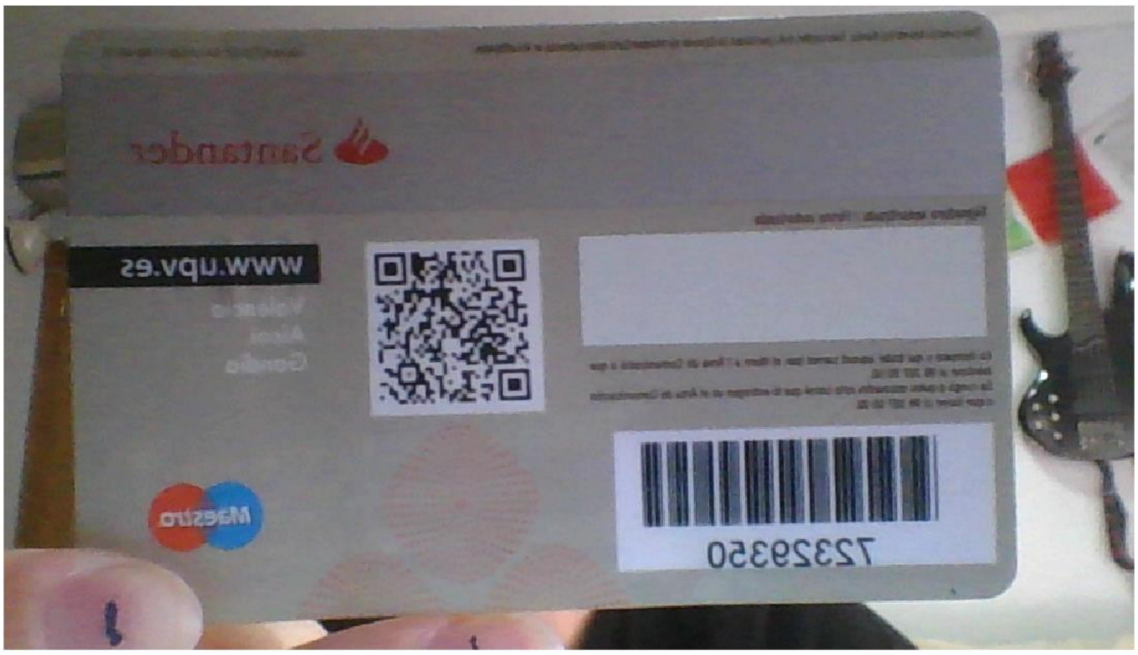


Illustration 31Only one of the patterns detected the original image

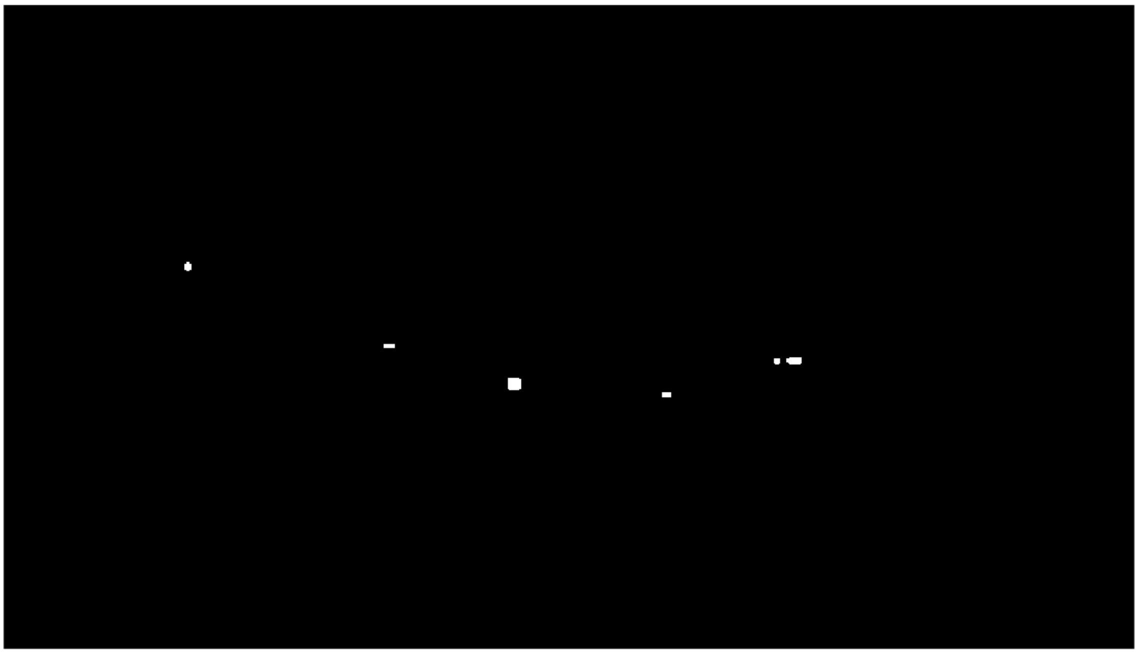


Illustration 32 Only one of the patterns detected objects black and White

Last, there are times where the detection is done correctly and all the patterns are detected, cropping the QR-code as shown from Illustration 33 to Illustration 38. But on these kinds of occasions, the QR-code is too blurry due to the motion or the image resolution. In these sorts of performances even though the detection is completed, it is not possible to extract their information through the ExtractorDatosQR function.

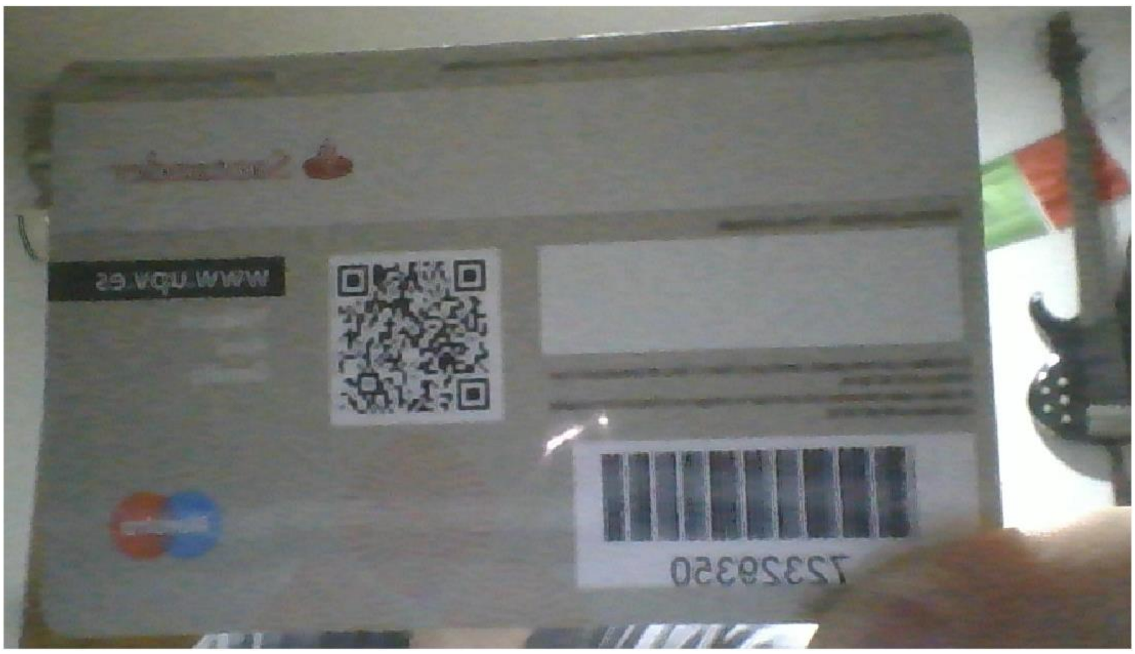


Illustration 33 Good detection of QR-code but not readable Original Image Example 1

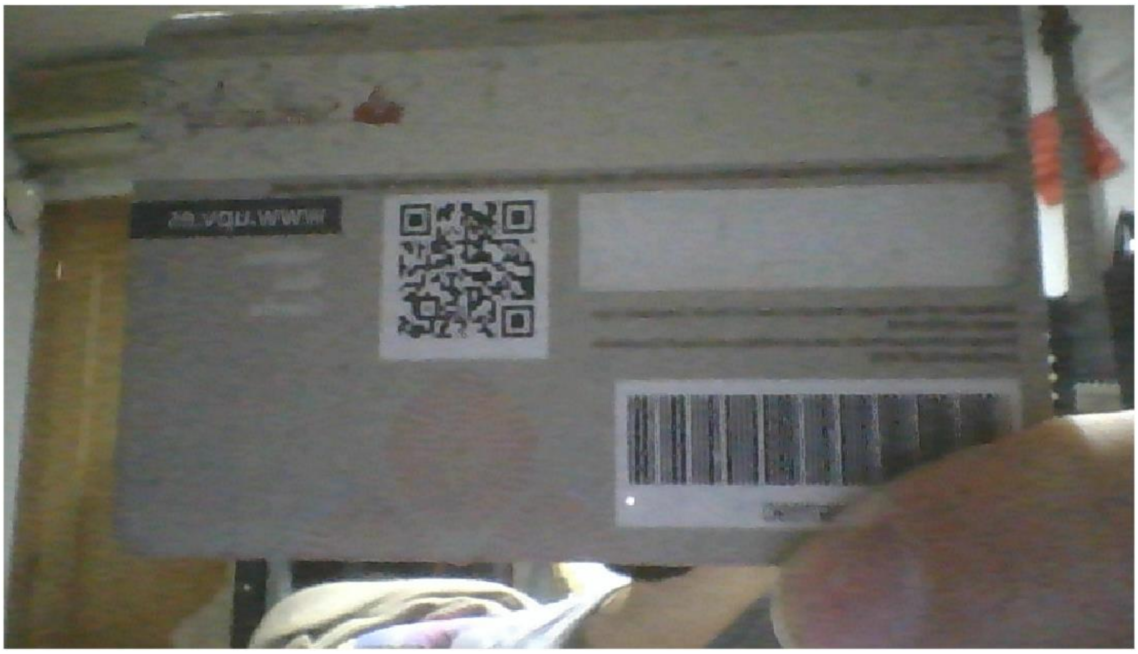


Illustration 34 Good detection of QR-code but not readable Original Image Example 2

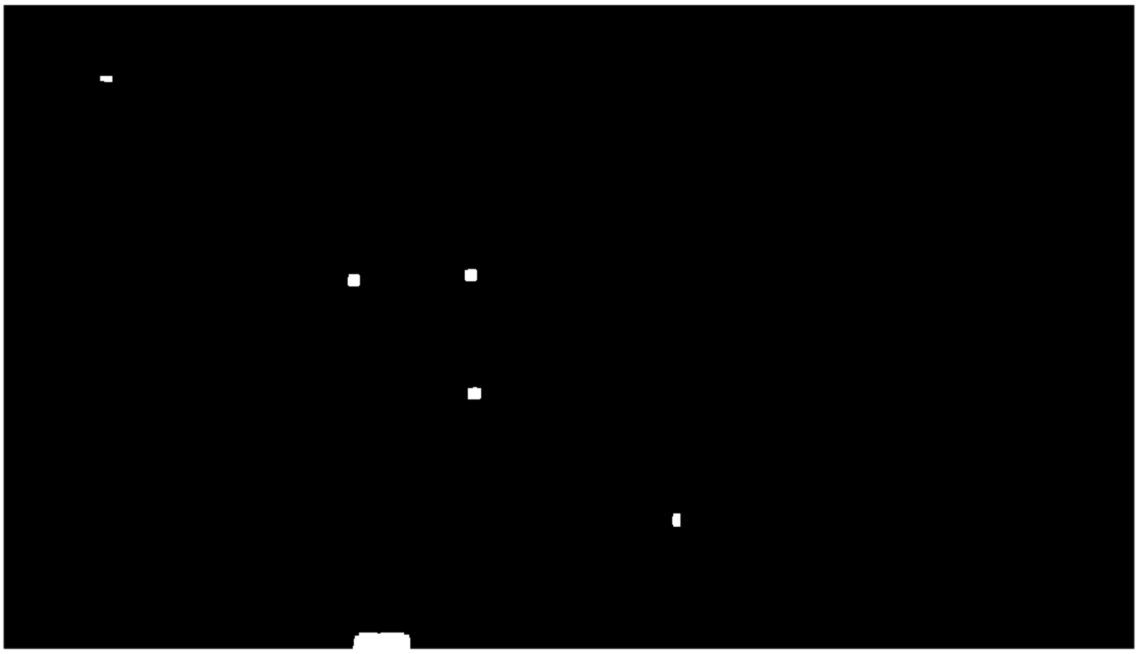


Illustration 35 Good detection of QR-code but not readable- Object detection- Example 1

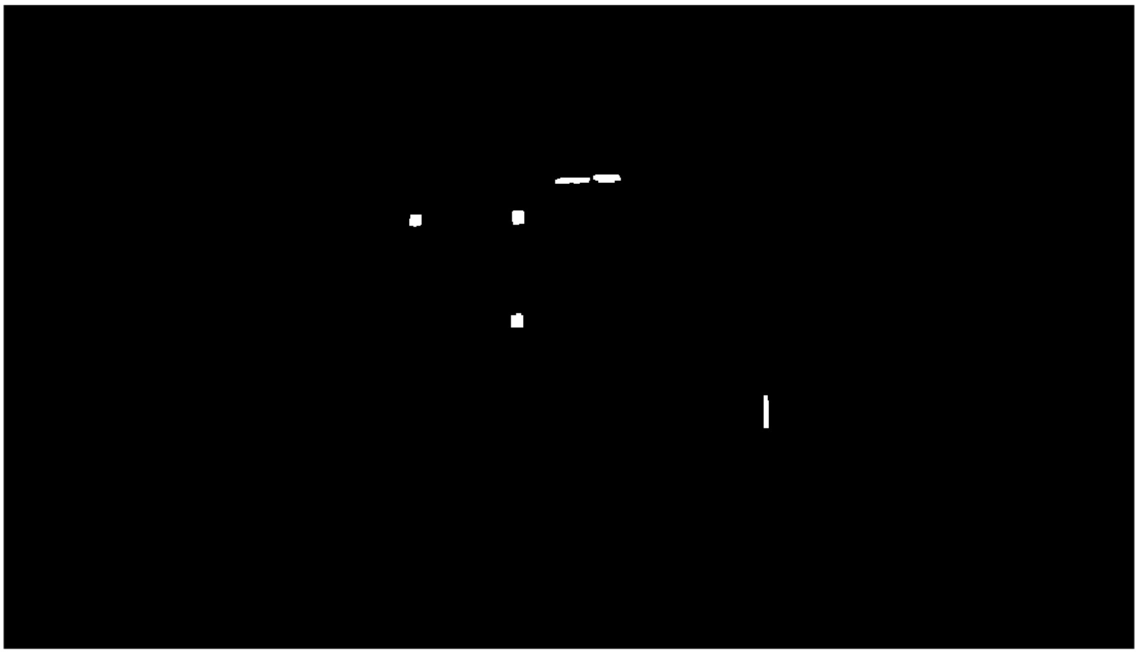


Illustration 36 Good detection of QR-code but not readable- Object detection- Example 2

Illustration 37 Good detection of QR-code but not readable- QR code cropped- Example 1

Illustration 38 Good detection of QR-code but not readable- QR code cropped - Example 2

An example of a good performance is explained and shown in 4.1.

5.2 recoNombre

This is an example of the result of the recoNombre function:

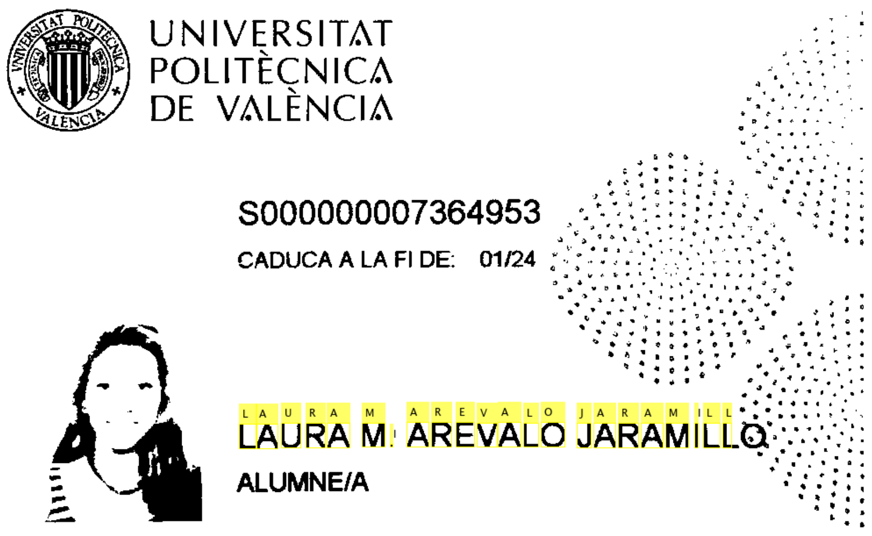

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
L A U R A M A R E V A L O J A R A M I L L
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 39 Name detection with point interference.

As can be seen in the resulting image, the text areas that belong to the name and do not have drawings or background patterns are recognized. We encountered a problem with compound or very long names, as they merge with the dot pattern of the card background. Also, compound names appear contracted with initials, which differs from the name database of the UPV website. A literal search for this name on the web would return no results.

To try to remove the dot pattern from the image we have tried different morphological operators and tools. As they are similar in size to text paths, the basic operators such as erode, dilate, open or close end up modifying the text too much without managing to remove the dots. We conclude that the best result is obtained with the *bwareaopen(I,n)* command. This operator removes objects with less than n pixels in the image. Set to 50 pixels it removes almost all the dots, except for those in direct contact with the text.

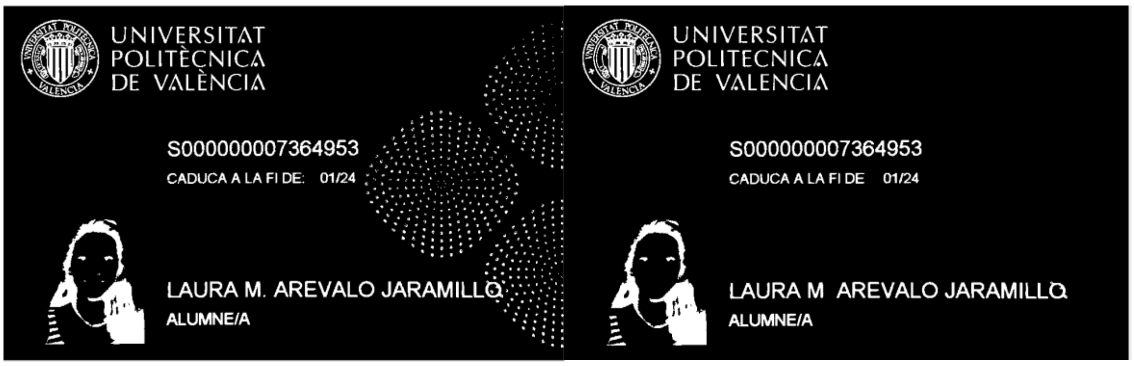

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
S000000007364953
CADUCA A LA FI DE: 01/24
LAURA M. AREVALO JARAMILLO
ALUMNE/A

Illustration 40 Points removal

The last letter cannot be detected correctly because the dots affect the shape too much:

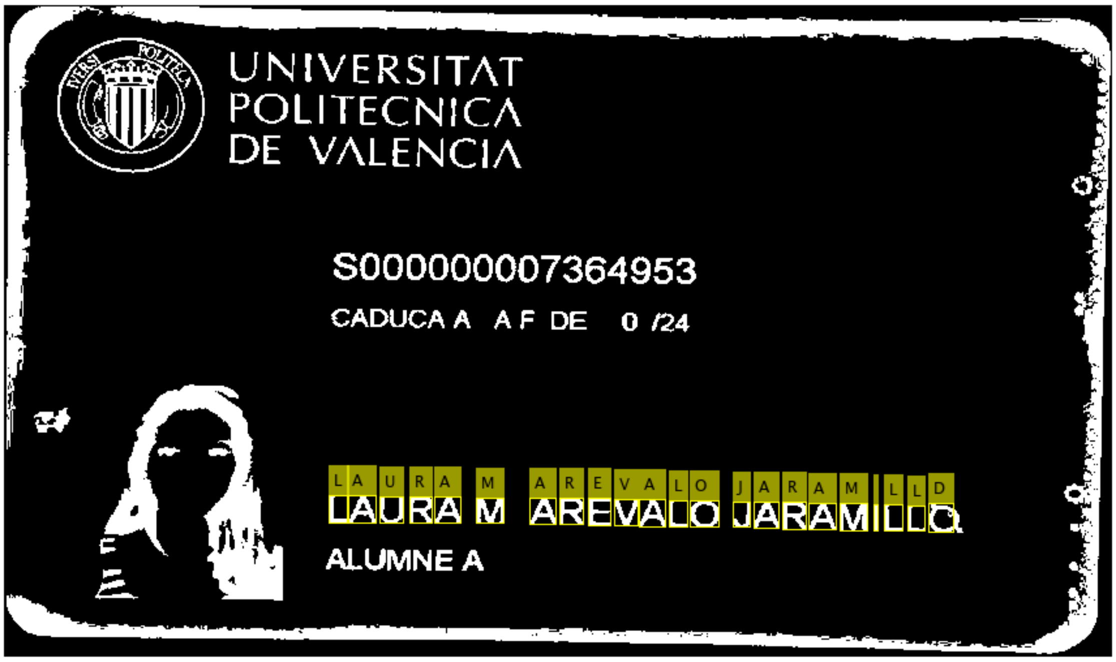


Illustration 41 Points modify the last letter

This case is considered extraordinary because it has a compound name and is too long, falling into the dotted area. In simpler cases, the result is positive:

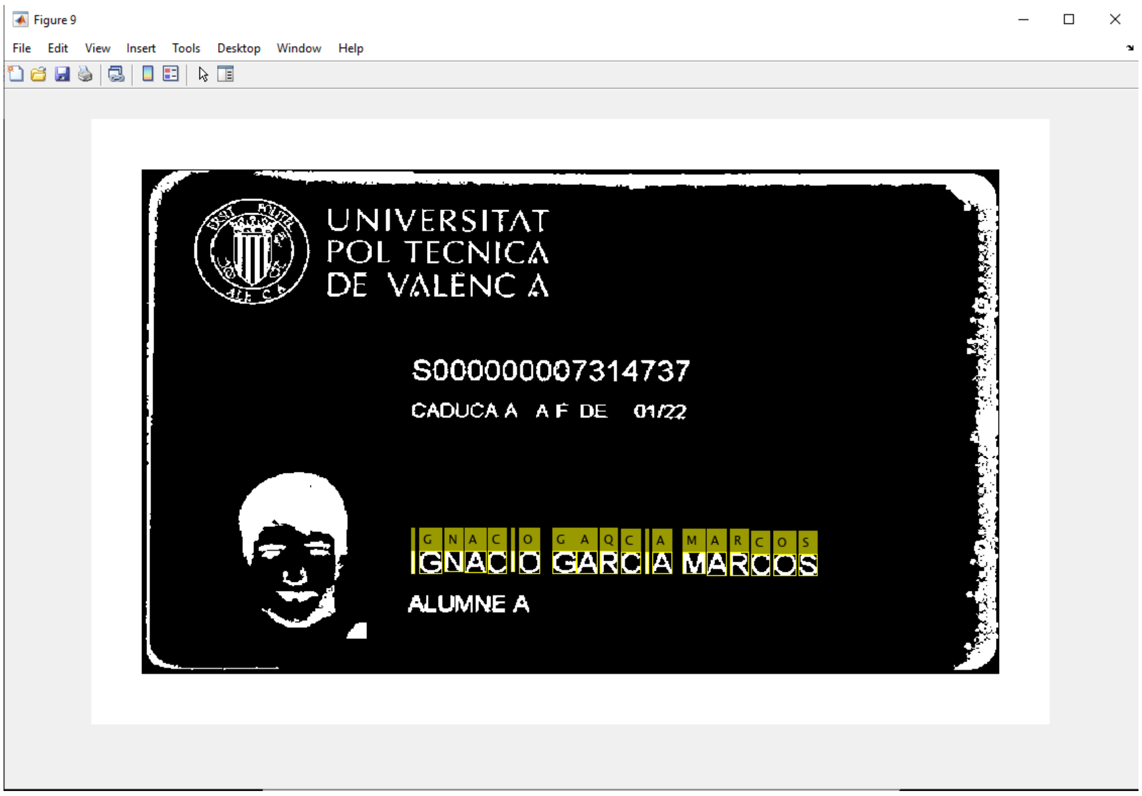


Illustration 42 Simpler case

In this simpler case, the result is still not perfect as one letter has been detected incorrectly.

```
Nombre =

    'IGNACIO GARQIA MARCOS

     '
```

Table 10 Matlab Code: Wrong recognition

6 Discussion Of Results And Possible Improvements

The recoName function was intended to be part of the main real-time application, but due to the way it has been implemented, it has proved to be invalid for this. A viable solution we found was to create a separate implementation that does not work in real-time. A new Tab has been included that allows you to open a file, and once stored, process the image and check the database to see if the name exists. Due to the large number of objects that are captured with the webcam, most of the time it is not possible to obtain a result.

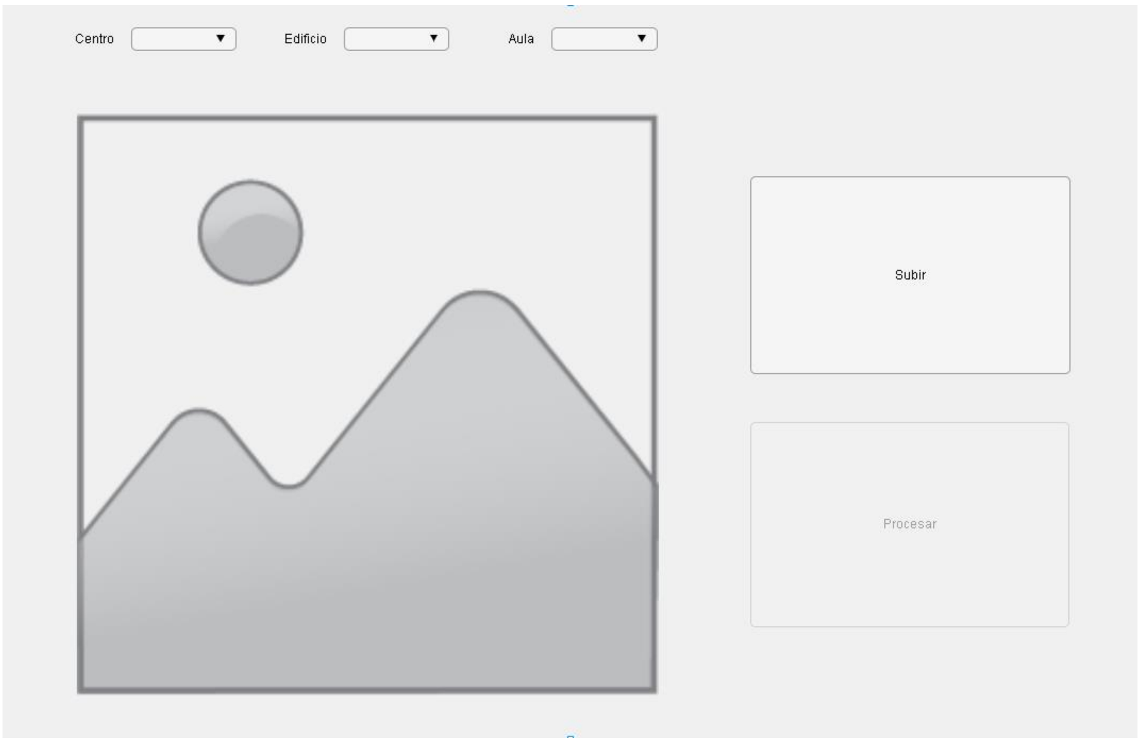


Illustration 43 New Non-real-time Tab

Besides, errors have been detected in this function that is due to a lack of language training. It has only been trained with 3 different cards, so it does not have many samples, not even the whole alphabet. In a future implementation, it should be trained with a wide range of images.

Due to the possible failures in the detection of characters in this function, it would be convenient to implement a function that compares the result with the results of the UPV databases, allowing a certain margin of error. Thus, the detection of a name with 1 wrong letter in each first name or surname would be considered valid.

On detectorQR, the dependence on the proximity and angle of the card is too big. Even though the results are good, an interesting take to this could be the use of a supervised binary linear classifier neural network based on *Gradient Descend*. This take could improve the recognition making not necessary the placement of a red rectangle and would eliminate most of the bad matches that happen on the actual code.

Also, there are discordances between the names written in the cards and the names written in the QR-code URL, when the person has multiple first names. To avoid this problem an algorithm should be implemented to differentiate when the person has multiple first names and look for the first letter match.

7 References

POLIFORMAT. Unidad 2.1 – Segmentación de Imágenes

https://poliformat.upv.es/access/content/group/GRA_11298_2020/Transparencias%20de%20clase/Cap2.1%20-%20Segmentacion%20de%20imagenes.pdf

[Query: 26/11/2020]

POLIFORMAT. Unidad 2.2 – Detección de bordes

https://poliformat.upv.es/access/content/group/GRA_11298_2020/Transparencias%20de%20clase/Cap2.2%20-%20Deteccion%20de%20bordes.pdf

[Query: 01/12/2020]

POLIFORMAT. Unidad 2.3 – Mofología de Imágenes

https://poliformat.upv.es/access/content/group/GRA_11298_2020/Transparencias%20de%20clase/Cap2.3%20-%20Morfología.pdf

[Query: 05/12/2020]

POLIFORMAT. Unidad 2.4 – Descriptores

https://poliformat.upv.es/access/content/group/GRA_11298_2020/Transparencias%20de%20clase/Cap2.4%20-%20Descriptores.pdf

[Query: 01/12/2020]

MATHWORKS. readBarcode

https://www.mathworks.com/help/vision/ref/readbarcode.html

[Query: 20/12/2020]

MATHWORKS. regionprops

https://www.mathworks.com/help/images/ref/regionprops.html

[Query: 20/12/2020]

MATHWORKS. imopen

https://www.mathworks.com/help/images/ref/imopen.html

[Query: 25/12/2020]

MATHWORKS. Videos and Webinars

https://www.mathworks.com/videos/webcam-support-89504.html

[Query: 30/12/2020]

MATHWORKS. Parse HTML and Extract Text Content

https://www.mathworks.com/help/textanalytics/ug/parse-html-and-extract-text-content.html

[Query: 05/01/2020]

MATHWORKS. database

https://www.mathworks.com/help/database/ug/database.html

[Query: 05/01/2020]

MATHWORKS. fetch

https://www.mathworks.com/help/database/ug/sqlite.fetch.html

[Query: 07/01/2020]

SQLITE. Command Line Shell For SQLite

https://sqlite.org/cli.html

[Query: 07/01/2020]

MATHWORKS. uialert

https://www.mathworks.com/help/matlab/ref/uialert.html

[Query: 10/01/2020]

MATHWORKS. OCR Trainer App.

https://es.mathworks.com/help/vision/ug/train-optical-character-recognition-for-custom-fonts.html

[Query: 16/01/2020]

MATHWORKS. BoundingBox.

https://es.mathworks.com/help/matlab/ref/polyshape.boundingbox.html

[Query: 16/01/2020]

MATHWORKS. bwareopen.

https://es.mathworks.com/help/images/ref/bwareaopen.html

[Query: 18/01/2020]

8 Time & Work Management

In the first week of work, a Gantt chart was made to plan the project correctly. Started defining the objectives and to do some initial research. Decided that would finally detect both the QR code and the name and started with the programming of the different functions.

Research work has been carried out while programming, learning new Matlab functions, and testing solutions.

Finally, the documentation has been written and future improvements to the project have been evaluated. All the planning has been fulfilled, except for problems in some of the functions that took longer to solve, or that could not be solved at all. Attached to this report is the Gantt chart.

9 Authors' Declaration of Originality

We, *Jorge Peral de León,* and *Laura Marcela Arévalo Jaramillo* declare that We are the sole authors of this report it hasn't been done by other people or copied and that We have not used fragments of text from other sources without proper acknowledgment.