

Machine vision and intelligent system

Project 2

Characters recognition

In this project report we are going to focus our attention on the development of license plate reading methods.

What are intelligent systems and machine vision?

Nowadays, the place of intelligent system in our daily life is huge and it is going to be even more important in the future. These systems consist of different sensors or actors components regarding their application domain, and are able to react to different input on those sensors in a more or less complex process defined by a code implemented by the user, to trigger actors or extract data.

When the intelligent system input is a picture, via a camera for example, which needs to be processed to gather information from it, we are talking about machine vision. This is a topic that is going more and more used in loads of applications as object tracking, face recognition, high-quantity production, metrology, land analysis... Through those examples we notice that this can be used on a wide array of domains. Actually, machine vision for intelligent system is a really interesting sector that can be used for most of the applications in almost every domain, at least those one that need the human eye and mind.

However, how is it possible to substitute the human ability to analyze a picture to extract data? In every machine vision application, two main parts can be noticed. The first one is about image processing. That include every step that helps process the picture regarding the goal we aim at, by improving the quality for example, changing the domain in which the information is initially get (linear to frequency, RGB to grayscale...), adding some filtering that allow the system to focus on a characteristic or a region of the picture. The second step is starting right after and use this enhancement of the information to recognize any object by comparing it to a data base. This way we can gather position, size, color, alphanumerical data and more from a region of an image.

Table of Contents

What are intelligent systems and machine vision?	1
I. Introduction	3
II. Problem definition and overview	3
<i>Our problem</i>	3
<i>How to solve this</i>	4
III. Algorithm	5
<i>Region sorting</i>	5
<i>Character comparison phase:</i>	7
<i>Image normalization</i>	7
<i>Comparison with the set</i>	7
<i>Percentage obtention</i>	8
<i>Selection with classifiers</i>	8
<i>Letters association</i>	8
IV. Experimental results	9
V. Related and further work	13
<i>Related work</i>	13
<i>Further work</i>	14
Conclusion	14

I. Introduction

In the last two projects, we were using some segmentation process to first being able to extract an object from its background, which was a low-level process since we only wanted to focus our attention on a part regarding its grayscale aspect. The second one was more advanced; indeed, we could not use thresholds segmentation anymore since the pictures was changing from one to another and a lot of details were represented in these pictures. We were here studying some cars picture to get the whole license plate of it. To do so, we used region properties and the result was pretty good, with 7/10 pictures with a successful result.

The thing is, we can't do that much with these plate picture. In fact, the next aspect of the machine vision must be implemented, as expressed at the very beginning of the report. As an example, a human need to first focus its attention on an area to first being able to read on this area. To read our plates, we will use all the plates extracted successfully from the pictures and going to use a training set to compare each character previously extracted with several test samples. A test sample is a picture of a letter for which we know the value and several of the same letter for each letter are building the test set together. This suppose that we are able to extract the characters from the previous picture.

Giving the ability to a computer to extract data which can be further processed in the intelligent system for the application goal is maybe difficult but can have a huge impact for technological applications and for our daily life. This is almost as giving a new kind of sensor to a system, giving him the ability not only to see, but to watch and to pay attention to its environment details by another way than physics values. Still, some of the applications will need a far advanced development. For some of them simple comparisons won't be enough anymore, and some learning algorithm must be implemented to train the computer to recognize different shapes according to a database.

As an approach to an intelligent system that use machine vision, our algorithm is an approach to this wide topic of image recognition, which has shaken the world over the last years, to understand the basics of how it works and what problems we can struggle with.

II. Problem definition and overview

Our problem

As explained earlier, we would like to allow our computer to read the characters we can find on a license plate. A huge part of the work has already been done by extracting the plate from the car's image. What is done is resumed in the scheme below:



Figure II-1: Extraction of the license plate from a picture in the last project.

This way we want to give to the computer the ability to read the characters on the plate. In the experiment part, we will have this kind of pictures to process on:

As we can see, we have several types of plates and several types of characters, with different sizes, different formats and additional characters we don't want to pay attention to on them.

How to solve this

To answer this question, we will develop an algorithm which consist of 3 steps. Each step will be explained



later and every process will be shown in a bigger scheme to have a better idea of the whole process.

Figure II-2: Several license plates we can obtain using the previous project.

The first step consists in the extraction of the character from the plate. As we did on the previous part, we are going to check the dimension using region properties, but also the relative surface, since every character bounding box should have almost the same size.

When the characters are extracted, we are going to start the comparison part. By spreading them in several small areas, in which the value is determined by averaging the binary pixel's intensity values, we can later compare each of them to several samples of letters which forms a reference set. Here we are using some supervised pattern recognition system.

The third step consist in the analysis of each percentage to tell which one is the more trustable and then return the string that correspond to these characters we can read on the plate.

III. Algorithm

To present the algorithm, we will first present the overall process, and then try to explain the role of each step.

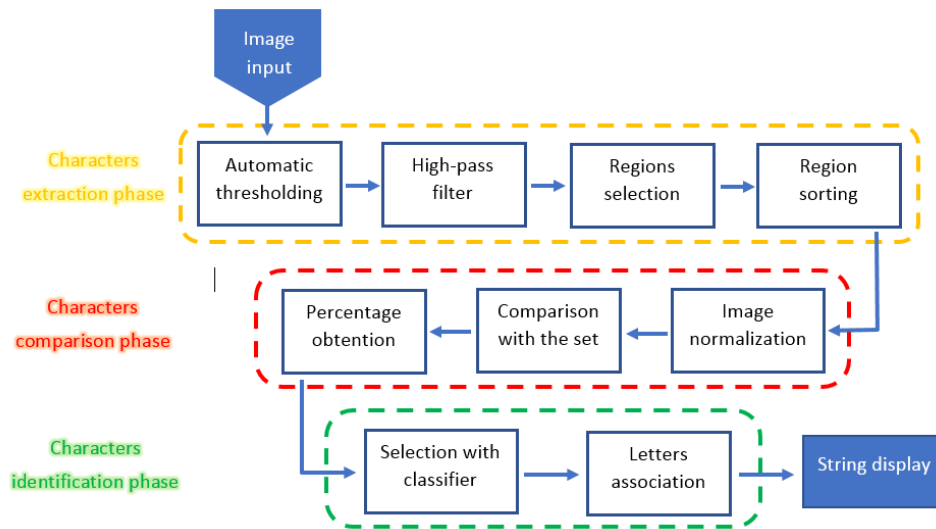


Figure III-1: Algorithm overview: block scheme

Our application is made of three phases, that are composed with several steps that we can see as blocks on the previous scheme. Each step has a special goal which is reach using functions. As these blocks do not give that much information about how our algorithm is working (especially what principles, conditions and functions we use), we are going to detail the important steps through further explanations.

About the automatic thresholding, the high-pass filter, and the regions selection, the processes used are basically the same as in the previous segmentation work. I advise you to take a look at the previous report called “Advanced segmentation” in the project 1. An important precision can still be made, we are using a Laplacian filter as a high-pass filter to get the contours.

This way we will start at the “Region Sorting” step. At this step, we get the bounding box of all the regions in the image. We then need to sort them to select only the chars. Once again, for further explanations about region properties, it is advised to check the previous report to have a better idea of this concept used in segmentation.

Region sorting

To chose the regions in which are included the characters of the plate, we are using several conditions in a specific order, to filter step by step and keep more advanced process for the end. Starting from the simpler ones help us optimizing our algorithm so that we have less regions to process heavier calculations on. The conditions are the following:

- Height and width of the bounding box not too big or too small: This step allow us to remove all of the little regions that represent noisy area or without information. This condition allows the user to also remove huge regions, such has the plates contours themselves. This step is not about selecting the characters, but more about rejecting every region that surely does

- not contain any information. That means that the threshold values should not be too close to the characters size. This condition is equivalent to 5% of the plate size, so that we can adapt to the size which allow a better automation process (we can use it on several plates without changing the value every time).
- The second condition is looking at the bounding box aspect for each region. Working only on independent surfaces does not really make a sense since we are working on rectangles. Here, as we are looking on characters, we want to be sure that we are working on vertical rectangles, where height is bigger than width. To do so, we just check that width/height is less than 1. We can tune this by selecting a more restrictive value (inferior to 0.7 for example), or even an interval of rates (between 0.3 and 0.7 for example). In the Algorithm, the value is just inferior to 1, but the code was realized so that the user is able to change it if it is needed.
 - The third condition is selecting regions by their relative size: here we don't want to work independently on each region anymore. The function is checking the surface (since we can now use it because regions are sorted by absolute size already) of each region and compare it to others' surface. The process is simple. For each region, we save in an array how much time there was a match with another region. At the end, we check what is the bigger value and we get the index of each regions that show this number in the array. We now have found our characters' region. Since we are working on a little area, and thanks to the previous sorting steps, it is almost impossible to get a big number of similar regions which are not our characters. In addition, we use an interval of surface, which is tunable. Basically, it goes from "surface to compare"-delta to "surface to compare"+delta. It is still fast since we are processing on 20 to 30 regions max, and the process is optimized to earn time regarding the number of operations done.

With this done, we can get those characters extracted from the plate, as we can see on the scheme below:



Figure III-2: Steps of the characters extraction. a) Binary (auto-threshold picture). b) Laplacian filter. c) Region detection. d) Region sorting: size and vertical rectangles. e) Result of relative surface analysis.

However, this solution is not perfect. We will discuss that in the experimental process, but it is important to know that some of the plate will not be able to extract the characters from it. In the experimental phase, we will still count them in the global result, but for each phases' results (extraction of characters and recognition), we are going to rate them independently. Indeed, this can help us to know if it is better to change the algorithm of one phase or the other, and therefore it optimizes the enhancement of the overall process.

Character comparison phase:

In this phase we are going to give the computer a set of alphanumeric characters. The computer will be asked to use it to recognize the characters in the plate. To do so, I made a set using paint and some Matlab image processing (cropping to the letter, and reversed auto-thresholding). Then I created a function that load every normalized character in an array. The order is from 0 to Z, so knowing the index it is possible to know the characters. I am using only one set of letters in this project. It is an evidence that using more will increase the rate of good results, we are going to discuss about that later. The letters I use can be seen below:

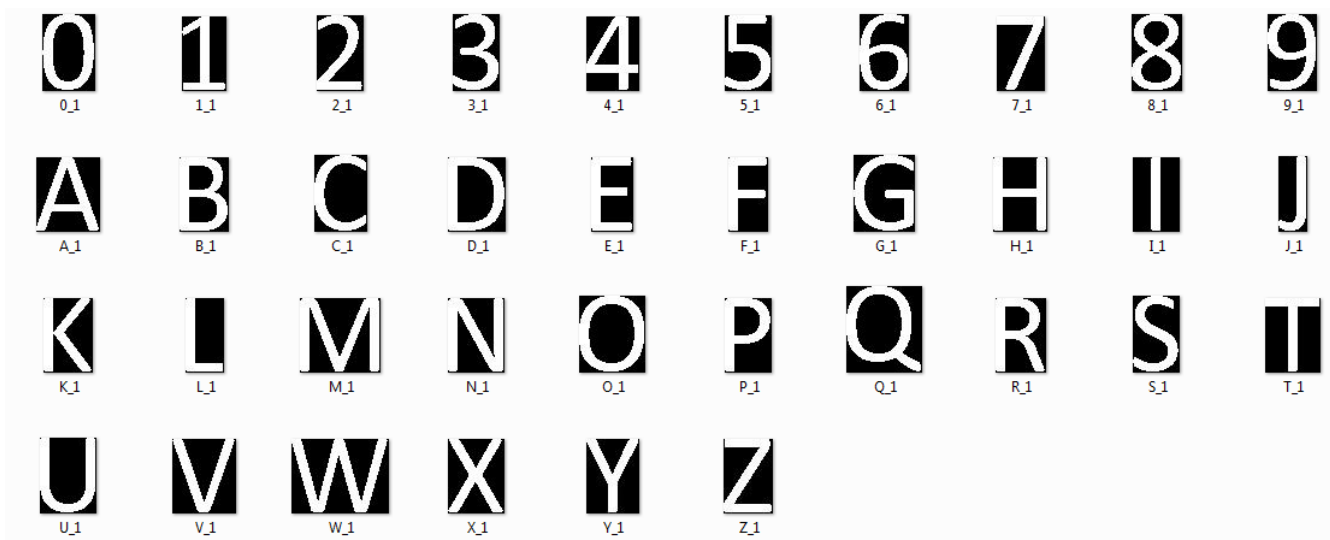


Figure III-3: Numbers and letters used as a test set.

I precise that every picture is then resized in Matlab before being concatenated in an 3D-array.

Image normalization

If we get in mind that we want to compare our extracted characters with those of the set, they must be on the same basis. There is several ways to do this, for example cutting both pictures in a number of area n , averaging the intensity of each pixel of the area to get the area value, and then compare both array of area to each other. The more each array is close to a pixel, the more our result is close to the reality, but the more time it take to do the calculation.

In our case, we have few letters to check and few pixels in each. This way I choosed not to use this technic, but size the extracted char to the normalized size of characters in the set, and then compare each binary pixel in both pictures.

To do the resize, I created a function based on the Matlab function "imresize(I, [r c])".

Comparison with the set

As explained, comparison is done on each pixels of both pictures. Each extracted characters of the license plate are stored as a "rectangle" (with the data format [x y width height], where x and y are the coordinates of the top-left angle of the bounding box) in an array. Then we compare each of the cropped images (of the plate) to the rectangle with all of the 36 alphanumeric chars, using two for loop to check each pixel.

The goal is to obtain a number that represent the similitude between the extracted characters and each letter in the set.

Percentage obtention

This number will be expressed as a rate between 0 and 1. However, we are not only taking the rate of likeness, but also the rate of differences. The obtention of each of them is done as follow. As a reminding, the white pixel (intensity level 1) are constituting the characters.

For the likeness, we want to check if two pixels at the same position in both pictures we are comparing have the white value. This is done by incrementing a counter each time both pixels have an intensity value of 1.

For the differences, we want to check if the pixel values are not the same. If this happen, we increment another counter.

Each counter for both values is concatenated in a 36 by 2 by x array, where x is the number of characters to compare. On the “n” raw are register both the difference and the similitude rates between the character to recognize and the “n” characters in the set.

To obtain rates, we just divide those numbers by the number of pixel we have in the picture.

Selection with classifiers

Here we are going to use the data we gathered in the previous part to select which character corresponds the best to those on the plate. Here we want to choose the letter which is the closest to our character. At the beginning, after several tests, the process has been coded as follow.

First, we check the likeness of each sample of the set with the initial character in the plate. Here we are taking every letter which has a similitude rate bigger than 95%, so that the algorithm is not only taking the bigger one. If we have only one character in this limit, we keep it as the final decision, however we need further process if we don't have any characters selected or if we have several of them. For example, if two letters look the same as a character, it is not a 50/50 choice to take the one which will have the bigger similitude rate. Instead, we are looking at the difference rate.

Imagining that we got an “I” and a “T” in our selection, both will have a pretty similar likeness rate. However, the difference rate will be different, since we are going to focus on the areas of each characters which are not the same. The lowest difference rate is the one we are going to trust and the one we will return as the result of the comparison.

In the algorithm, I did not invert extracted characters from the plates, that is why I am taking the opposite in terms of rates checking (<5% and the biggest difference). This is the same as explained right before at the end.

Letters association

We manage to drawback a “char” value between 0 and Z using a dictionary. It is defined in a 1 by 36 array, where we are storing each character that we can find in the plate using “char” variables.

Knowing the index of each letter, we are then able to find the one which was selected after the test. We store every letters in a string variable and display it in Matlab command window.



IV. Experimental results

The experiment is done using seven plates we successfully segmented with the algorithm of project 1 – license plate segmentation. We are going to process the whole algorithm on each of them, and then we will look at the results, paying attention about several characteristics:

- Are the characters extracted correctly? The numbers of characters and their final aspect are determinant factors.
- Are the characters recognized correctly? The quality of the recognition is not determined only by the whole string value, but by the characters independently. This way we can adjust our algorithm specifically to one issue or another.

If the first characteristic is not good enough, even if the result will still be written, we are not going to go further in the study because that means it will impact the reading.

Plate n°1:

Plate	Characters extraction	String result
		<p>letters =</p> <p>B848RPB</p>

Here we can see that the characters extraction is working correctly, indeed there is nothing to perturb us in the plate reading.

However, the recognition makes some mistakes. The 4th and the 5th characters are not correct, we therefore have 5 correct over 7 characters. As we can see, the algorithm manages to make the difference between the B and the 8 (characters 1, 2 and 7 on the plate), which is not an easy task even by looking at the plate itself.

However why do we have those two mistakes on pretty easily readable characters? In addition, the M has the same font as in our set. The factor that has the biggest impact here is that the letters do not have a really good quality after the thresholding, as we can see on the picture below. This noisy image is changing the shape of every characters, which, after the resize of those very little regions in terms of pixels number, the information is not clear enough anymore.



Figure IV-1: Thresholded image of the plate

Actually, this was spotted by looking at the rate we gather from the image. We can see that there are not any letters we can remark compared to the others for both the 2 and the M. We can see that on those bar characters below. Actually, we notice the same result on every other chars of the plate.

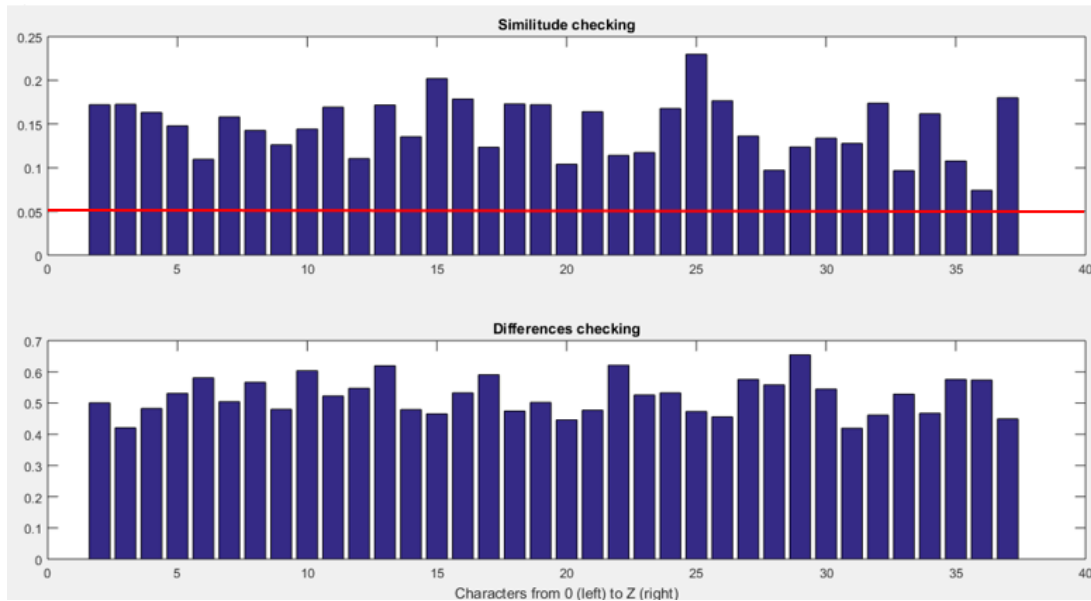


Figure IV-2: Bar chars of the similitude and the difference between the "2" in the plate and each sample of the set. The red line is showing the value under which the similitude must be select a char.

The result can probably be improved by adding some processing, but this is difficult to know if the quality will get better since we are working on a plate which is really little in terms of pixel dimension. Getting the car closer before taking the picture is probably the best option.

Anyways, we still manage to get the right characters on the other ones, which is not easy regarding the previous explanation. That is overall a good point for our algorithm.

Plate n°2:

Plate	Characters extraction	String result
		letters = 40T07

In this plate, it is possible to see an issue with our character segmentation method. As we are using region properties, the C and the second 0 are not taken in the selection since they are touching the contour of the plate. Therefore they are part of this big region that don't pass the filter. This can be fixed using another extracting method, as we will see later.

About the characters, I don't count the Chinese sign in it since it is not in my set (then it can't be recognized). On the other ones, we get 2 right characters over 4. We can see that 7 and eight are pretty close, same for G and 0.



Here a really interesting part is that this plate has white characters. With a function that check if the information is white or black, we are allowed to continue checking characters without any manual modification with the same set of characters.

Plate n°3:

Plate	Characters extraction	String result
		<pre>letters = NOLRNF1</pre>

With this plate we are back to black characters. We notice that on a plate that has some additional information, this character extraction result is really good. About the characters reading process, we got a pretty good result with 5 letters found over 7. We still have issues with O/O/C, and the M error is explained by the fact that our M in the set is going right to the bottom. Actually, the one on the plate has more similitude with an N than with our M.



Plate n°4:

Plate	Characters extraction	String result
		<pre>letters = HOR5SH1T</pre>

This plate with black characters is showing a really good result. First, the plate has a lot of additional information we don't really want to know about, and it still manage to do a really good segmentation, so that this additional information does not add noise in our reading result.

About the letters, it is really interesting because this time we managed to discern the 5 and the S, the I and the 1, and every character are red correctly. 8 over 8 here.



Plate n°5:

Plate	Characters extraction	String result
		<pre>letters = KN3RXR</pre>

Here the segmentation has missed one letter. The "I" is indeed too thin to pass the filter. To counter this, it is maybe better to use the "same position" filter than the "same surface" one.

If we count this mistake, we have 4 characters correct over 7. The one which cause us some issues are the one when the threshold is losing some information, as in plate one. Anyways, this would not have change anything for the M since it does not have the same profile as the one in our set, therefore looking more like an N.

Plate n°6:

Plate	Characters extraction	String result
		<pre>letters = LWAT</pre>

Here the biggest problem is the character extraction. If we can't do anything for the first T, it would have been possible to extract the O and the second L using a position filter.

Here there is not the much to say about the characters reading.

Plate n°7:

Plate	Characters extraction	String result
		<pre>letters = RTPLSI</pre>

Here the character extraction is done properly. The plate is really clear in the char area, but the additional information close to the top and bottom borders are not disturbing our method.

Concerning the letters, they are really clear to read. Here two characters are not read correctly. The reason to that is the font used in the plate. It is different as the one in our set of chars. For example, the l in the plate has two bars, ours got only the vertical part. Therefore, it looks more similar to a T for our algorithm. The one is the same, but the tilted bar on the top left is not long enough for the computer to notice it.

Overall results:

Before talking about the strength, weakness and improvements to be done, it is important to gather those individual results to have a more concrete idea.

Globally, from seven different plate we got the following results:

Plate N°		1	2	3	4	5	6	7
Overall process	Corrects chars	4	2	5	8	4	4	4
	Total chars	7	6	7	8	7	7	6
	Success rate	0.63						
Extraction process	Extracted chars	7	4	7	8	6	4	6
	Total chars	7	6	7	8	7	7	6
	Success rate	0.87						
Reading process	Correct reading	5	2	5	8	4	4	4
	Extracted chars	7	4	7	8	6	4	6
	Success rate	0.75						

With this tab, we see that token independently, each part of the algorithm is pretty good, but overall the performances are only around 63%.

Actually, both of each part can be improved using several enhancements.

V. Related and further work

Related work

There are many ways to do this task. One of them is in the characters extraction. It is about using the dimension of the plate since everything is official and standardized. Therefore, we can cut it between each character knowing their position. This method is valuable only for a defined application, and we actually got a good result for a method that can be applied in every country.

It is also possible to detect them using a vertical counter. When we have 0 or a really low value in the counter, that means there is not information, and therefore we can cut the plate at this plate. I tried several tests with this method, but that was depending too much about the plate, since sometimes there

was some additional scripts (for example in the plate n°4) that causes problems, or some noise that had a too heavy impact on the result (plate n°6).

Further work

I'll give here two really important feature that can drastically increase our success rate. I did not implement them in the algorithm because of the time I got to get this project to an end.

The first one concern the segmentation of characters in the plate. I think that adding a method that check about the position of each region in the plate, and keep all of them which has a relatively close y position from one to another, as it is done with the surfaces.

This can allow a less constraining condition on the surfaces (maybe only on the height, since it must be every time the same, therefore we can extract the I or those thinner letters) with still a good filtering. However, I do not have concrete ideas about those letters that are touching the border or other regions which keep them out of the selection. Maybe combining it with another cutting technic exposed above can be an idea.

The second one concern the reading of each characters. As we saw in the results, sometimes the characters are not understood correctly by the computer because of fonts problems or too big or noisy letters. This is due to the fact that the algorithm uses only one set of characters with one font and one size.

Comparing with several sets can allow the user to gather several rates for each letter, as we did. Therefore, we just have to take the average of each rate for the letters (for example the similitude rates for the letter A), and we are in the same situation than now, but with a more accurate result. Statistically, this will help to get better results, but that also need some time to check every pixels of each samples of each sets. Moreover, it may be interesting to think about the way of getting a global rate. Averaging and getting the closest letter is maybe not the better way, even if it is probably enough for our license plate application.

Conclusion

After several steps, we manage to get an algorithm that can, from a license plate picture, extract the characters and read them. Overall, with an effectiveness of 63%, it is maybe not the best algorithm for a practical application.

However, this is calculated over a large panel of license plates, with different size and colors, different fonts and characters localization, and sometimes with additional information. In addition, we get a result of 75% for the reading of the characters using only one set of samples.

Still, there are several things that can be improved in this algorithm in both the extraction and the reading phases. In addition, it can be a lot better organized and optimized.

This project was really interesting and it allows to have good notions of machine vision and intelligent system. Giving us the opportunity to use now a new door through the sensors and the automation's world.