Machine Vision and Intelligent System

Project 1
Advanced segmentation

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
Our problem 3
III. Algorithm 5
Automatic thresholding 5
Image blurring 6
High-pass filter 6
Regions selection 7
Result size sorting 7
Results analysis and selection 8
Image cropping and result 9
IV. Experimental results 9
The method and criteria to check the algorithm 9
V. Related and further work 12
Comparing with other work 12
Further work 12
VI. Conclusion 13

I. Introduction

As in the previous task, which was about segmentation of several basic grayscale pictures, we are still going to process segmentation on this project. Segmentation is the process of highlighting regions on a picture to be able to extract them. This can be done regarding several characteristics of the regions, as shapes, color or texture for example. Most of applications are about gathering data about a special object need this process since it allows to extract the object from its background and every other detail in the pictures. Several processes have already been implemented, and most of them are based on the differences between to areas or on the shapes of objects. Still it is not that easy and the process of segmentation does not stop at only applying some kind of high-pass filter…

Some applications require indeed much more developed algorithm for the result to be acceptable. Even for basic applications, minimizing the error we can get from the segmentation on different pictures, in other words different situations, is a great challenge. To differentiate a difficult situation from another one, let's look at an example. On a production chain, we want to check some characteristics of our products. It is pretty easy to realize it since the system does not have to worry about variation in the background color, the orientation of the picture or the brightness on this scene. In this last case he can just fix it with an enclosed case and a lamp lightning the area. This way the system can count every product, checking their quality and sort them regarding their type for example. In this case it is pretty easy to develop a segmentation process since the conditions will always be almost the same. In another example, let's say we want to track a person in everyday situation, for instance in a street or in a park. Here we want the algorithm to be portable, that means that it should be able to work on every different situation. Two factors are changing around here, the person by its size, weight or sex and the background that can include a lot of details which give a noisy image easily miss understandable.

Overall, the complexity of segmentation is most of the time remaining in the possibility of the user to influence the situation he wants the system to process on. This is really important for the following of the project, since some choices must be made about several situations.

II. Problem definition and overview

Our problem

What we want to do here is a part of car's license plates recognition. As every machine vision problem and as explained earlier, this global project can be split in a part where we must process and segment the picture to focus our attention on an area and one part where we must recognize every character in this area. For this project, we are going to focus on the first part, especially on the segmentation of the whole license plate of the car.

This time the application topic is determined and those images are not simple to process anymore. We have some RGB colors with lot of detail which include some noise on the pictures, and it is not possible to process some threshold selection anymore, at least it is not the easier way. The method we are going to use is based on region properties, with a shape filter that only give us the high variations (the limits between two regions). Each contour will determine one region from which we will be able to get the size or other characteristics.

We will focus on some characteristics of the plate to get some results. The more details we can get on this region, the easier it will be to find it in the picture. The algorithm expects that there is always a plate to find in the picture, for example an application where the machine vision part is launched when a car detection is triggered. We are going to work on this kind of pictures:

Figure II-1: Example of processed pictures

On these two images we see different facts that can make our project a bit hard. First, plate does not have the same size, the same format, cars does not have the same color neither the plates, and background of the picture can be very noisy.

Another point is that those plates are not always centered, it is even not positioned or oriented the same way from one picture to another.

At the end we expect to get a cropped picture of the plate, to be able to reprocess it as we want for further application. This will allow us to remove all the information we do not want and continue the project with other possibilities.

This project is really important since being able to recognize license plate allow the user to perform application around car facilities and driving, since it is almost the only way to identify a specific car. The color, the model or the brand are not enough for most of applications, and are maybe more difficult to recognize than the license plate. Being able to recognize those plates allow us to provide road surveillance, parking or access checking…

As we can see on all those pictures, the only characteristics of the plate we can see from one image to another is the shape of it (a horizontal rectangle or at least a parallelogram/trapeze regarding the angle) and the several characters inside. In addition, from one picture to another, the size is not changing that much.

This is a great application and we will provide an algorithm that can be used on several plate types and situations.

III. Algorithm

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

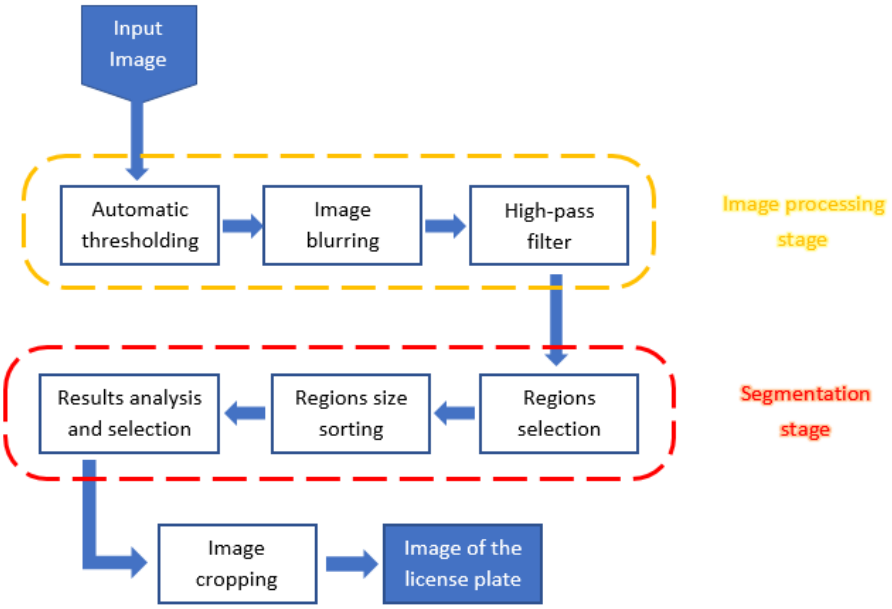


Figure III-1:Block scheme of our license plate searching algorithm

Our application consists of two main parts, that are composed of several functions which can be seen independently. These block does not detail that much the functions used and each process we use (e.g. for the region size sorting), the goal is just to have an overview to understand easily how the algorithm process.

We are going to explain each block with the function.

Automatic thresholding

First, we process to an automatic threshold using the Matlab function graythresh() that allows us to return an optimized threshold level to turn the image to binary. This function uses Otsu's method, which start by assuming that we can find to class of pixels in the picture: foreground and background pixels. To discern them, the function spread every pixel in two group to have a minimal combined spread between these two groups.

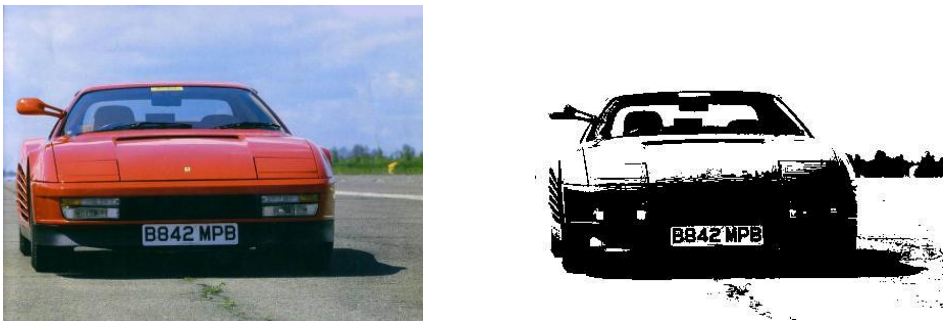


Figure III-2: Auto-threshold process from RGB picture

As we will see later, this method is sometimes causing some issues in the plate detection. By fixing some unknown variables of the problem, the user should be able to use this method without any problems. We will discuss about this point in part “IV. Experimental results”.

Image blurring

As explained in the previous report, blurring the image is useful to remove the noise on the image. As we can see right before, the image that we get at the end of the threshold process is pretty noisy. However, it is important to think about the level of the blur we are applying. Indeed, it blurs contours since it is working as a low pass filter every high intensity variation are lowered.

We process on a binary image so the blurring will be very effective around each limit between black and white areas since it will remove the noise without impacting the variation. Indeed, the picture is still binary after this process, so we don’t have any issue with high-variations that can get smoothed. We are going to process an average filter, which use an averaging seed on each pixel of the picture.

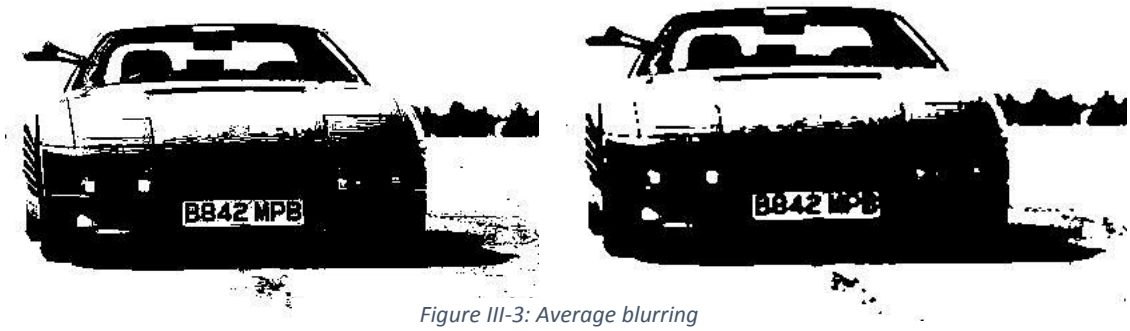


Figure III-3: Average blurring

We notice that this blurring aspect is maybe not the best for the character recognition if they are not big enough, since we lose a lot of information on them. Still, as we only want to segment the plate, it is not a problem.

High-pass filter

To end this pre-processing stage, we are going to highlight the variation from one level to the other one using a high variation detection function. To do so, we are using here a Laplacian seed on the picture. This will give us the contours of the variations, on which will be able to work more easily.

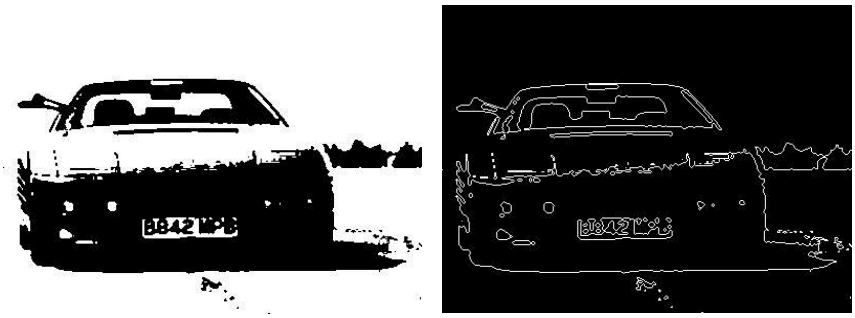


Figure III-4: Edges detection

As we can notice, each pixel inside a region is rejected to 0 (black) and those on the limits are set to 1 (white). They are a lot of others method (for example subtraction of a dilated region by the original one) but this one is precise and fast.

Regions selection

Here we enter in the segmentation stage. As we explained in the part "II. Problem definition and overview", we are going to segment the plate using region properties. The job is done by the Matlab function "regionprops()" which allow the user to select all the regions in the image and pre-compute several characteristics of them.

In the code, BoundingBox, Images and Extrema are pre-charged. BoundingBox contain the coordinate and size of the rectangle that include all the region, Images are returning the image in the BoundingBox, and Extrema consist of an area which includes the position of the height extremes pixels of the region inside the boundary box. However, this last characteristic was used as a test and the final algorithm does not need it, we are discussing about this point in part "IV. Experimental results" and "V. Related and further work".

If we display those BoundingBox using the Matlab function "rectangle()", we can see something like this:

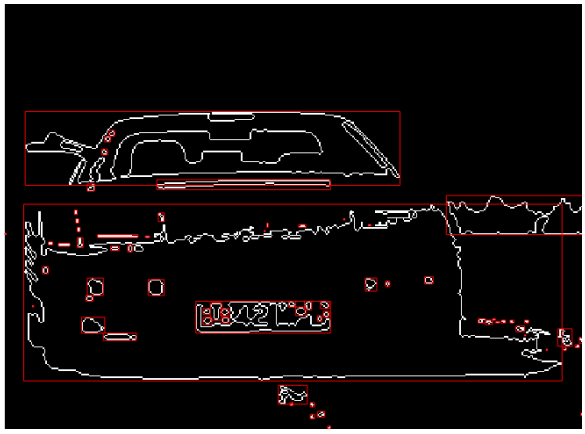


Figure III-5: All the different regions in the picture

As we can see there is a lot of different regions on this picture, that is why we are going to process some filtering about the plate characteristics to be able to segment it.

Result size sorting

One of the things that we first see, is that some regions are only some littles areas that we are not interested in. To remove this, we can apply a simple condition on the size of the region we want to check. To do so, we can process with two ways. The first one is checking the surface of the region, the other one is checking both the height and the width.

The thing is if we use the surface, that does not reject the possibility of getting the same kind of value without having the same shape at all. Furthermore, it taking more resources because of the 'for' loop, checking values is faster than multiplying and checking values.

Applying this concept for the too large areas, we can both reject area that are too big and too small. However, if we choose a too small interval for this, we are going to miss license plates on other pictures. This way we still keep a reasonably large interval.

Another condition to reject other areas is using the rate of the width of the region over the height. For a horizontal rectangle, this rate should be inferior to 1. We set this condition like this, but knowing the dimension of the plate and the distance between the camera and the car can improve the results significantly, thanks to the adjustment of all those conditions.

If the region passes these conditions, we add it to our list and we can display them like this:

Figure III-6: Result of the region sorting

As we can see, we still have to additional regions in the selection that we will need to reject.

Results analysis and selection

In this part, the first step is to check how much regions we still have in the selection. We envisage three different cases. The first one is that we do not have any result in the selection. That means that the plate did not passed the test and, at a larger scale, that no rectangle shape passed the test. In our plate detection application, which means we suppose there always is a plate to find in the picture, that means there was a mistake, and we display a message that suggest the user to tune some preprocessing or sorting values.

For the two situations left, a counting system where we check the number of elements in the region is used. The minding is based on the fact that we must have several characters in the plate.

If only one region is present, we are checking how much elements it possesses. In the case there is less than 5 regions inside the selected one, we do not use it and display an error message, explaining that the plate was not found in the picture. The user is invited to change the values according to its application to have better result. If the area presents a sufficient number of regions inside it, we use it as the license plate. This reduce the number of false negative errors we can have with this algorithm.

If we have more than one region in the selection, we count the number of regions in each of them. If we consider that the remnant noise in the picture is balanced all over the image, the region that will have the

most regions in it is the plate. Still, it is not a sure process but it works most of the time, the only annoying situation is when a region englobes the plate region.

Image cropping and result

Once getting one final result, we use the BoundingBox corresponding to the picture which later allow us to crop and extract/display the final result.

Figure III-7: Cropped plate of the original image

Cropping the initial picture allow us to reprocess a pre-processing step. This is really interesting since we can do a new auto threshold specific to the plate or other functions. Earlier, we could not process a specific processing because we had to be careful about the information loss. Still, this process can take some time to run, and some applications maybe do not need to reprocess to have good results.

IV. Experimental results

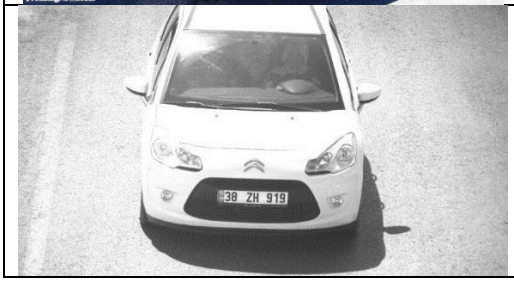
The method and criteria to check the algorithm

To test our algorithm, a part that is really important for me in this application is the automation. This way, we are going to set our variables to values that will stay constant during the application.

Those values are:

- Blurring: fspecial('average',3);
- Edges detection: fspecial('laplacian',0.2);
- Interval conditions for ratio:
  - delta = 0.20; % 20% tolerance for the y/x calculation
  - conditionMin = 0.33 - delta;
  - conditionMax = 0.33 + delta;
- Interval conditions for the distances:
  - Height >= 20 && width >= 60;
  - Height <= 150 && width <= 300;
- Num min of regions inside the final result: 5

We are using a sample of pictures more or less difficult to analyze, with various style, car's color, background… Counting the fails and successes will help us to give a rate about the performances of this algorithm. Also, the different cars pictures will help us discuss about some strong and weak points of this plate segmentation code.

Initial picture	Processed plate	comments
		Here we got a successful segmentation of the plate. Even if the fact that the shadow allows us to have clear edges of the white plate, the small size of the area we wanted to segment and the background that is not united do not make things easy.
		Even if this plate is in the middle of the screen and that it is not too far or too close, the image has a lot of details in it, which show that our algorithm is strong against additional information.
		Since the plate is too big, the algorithm is not taking it in the selection. In addition, it presents another region that have more than 5 regions inside, which generate an error we cannot detect.
		This plate is little, not centered, and the image has a lot of different details in it.
		Here the process fail because of the too small size of the plate. In addition, because of the same reason as the previous error, we don’t have an error message.

		The European plate pass also the test, even if the picture has lot of details inside since we don't see only the car. In addition, we see that there are a lot of regions that match conditions in the algorithm, and we still have the plate at the end.
		This Image is pretty difficult. The quality is not really good, there is a lot of details and we have a shadow that pass over the plate. Still, the result is pretty good.
		Here the plate is a bit far and tiny which is not easy to detect. In addition, some characters are written just down the plate, so algorithms based on characters detection would have some troubles when this one does not.
		Here we managed to get a blue Chinese plate. The plate is in the middle of the picture and its size is pretty good, however it shows we still can get the plate from different colors.
	x	This picture does not give us any results because the plate contours does not show themselves on the threshold processing. The yellow color is too close from the white, it would be possible without using the auto-threshold.

		Here we got a message that tells us to change parameters because no plate was find.

From those 10 pictures, we manage to get the plate from 7 of them. From the 3 others, we have one which is displaying an error message, and 2 that display a wrong area of the initial image.

This algorithm is really interesting since it allows the user to segment the plate of almost all the images without paying attention to the color, the side, the details in background, the country… Still for some other development some algorithm can be more precise to add features or cover weakness such as the orientation (the boundary box doesn't follow the rectangle but stick to the outer points of the regions) and the size of the picture mostly. The color and low-quality edges are also an issue, but it is due to the thresholding, some additional processing functions must be developed for those one.

V. Related and further work

Comparing with other work

A huge advantage we have with this algorithm is the ability to pass from one image to another without tuning the function's setting. Some other algorithm which aim to recognize characters of license plate are using directly the characters recognition, and this is a method that work nicely. The thing is once again it is a bit difficult to implement, in addition it needs a data cluster. If this kind of method is used on an image where characters are presents, it is possible that some errors show up. Our algorithm gets a quick result with a 70% accuracy and 80% trustability; the only weakness compare to this kind of character recognition method is that we are not sure to have a result about the plate. If that happen, we cannot go on further processing or recognition.

Further work

To improve our method, we can do several enhancements. The implementation of some of them can be seen in the folder PROTOTYPES.

The first idea is to try to have a more precise idea of the region characteristics. Instead of the bounding box that include the whole region, it could be interesting to use the region itself. We know that the region should be a rectangle, sometime a parallelogram or a trapezoid regarding the position of the camera. For further development we are going to consider the case of the parallelogram. To discern it, we can use several properties:

- Each opposite side must be parallel with each other (using direction of each lines);
- Each angle must be the same as its opposite (using trigonometry);
- Each side must have the same size than its opposite (using distances);

For all of them, we are using the "Extrema" property of the region properties function. Getting each corner of the picture can help us getting distances, direction and angles of the parallelogram 's sides. Anyways, we are setting a percentage to create an interval centered on the value to compare. The first one did not work because of some division by zero in some case, I did not try further.

The distances should work, but sometimes it did not show any rectangles, further studies are needed to evaluate it. I did not try to implement a method with the angles for know that was my last plan because it is not that great to compute some trigonometrical functions in ‘for’ loops.

The other improvement concerns the size of the plate. We saw that several times, the fact that the plate is too far or too close can be an issue. To counter that, one idea is to process the algorithm normally. If there are no results, the result got too much/not enough regions inside it or there are too much results that have a high number of regions inside, we are going to restart the process, but on a zoomed (or dezoomed) version of the picture. This way we are able to check another time if we can see different results.

This was the initial idea, and that seems pretty complicate. We can use the same idea but with some modifications on the conditions or the factors directly. The ultimate idea is to compute on several version of the images and pay attention of each region selected to keep the areas that seems the most probable to be the license plate. Splitting the initial image in several areas that overstep each other (so that we don’t cut on the plate) is also a possibility.

VI. Conclusion

This license plate segmentation of cars is working pretty well since we are able to discover several plates without having any troubles, in situations that are really different to each other.

However, we still have some troubles on several pictures, and further development can help us to forget about several weaknesses.

This first step is achieved and we will try now to gather the characters to proceed to a recognition algorithm to write the car’s id in a file for example.