Machine Vision and Intelligent System

Project 1

Image Processing and Segmentation

In this project report we are going to focus our attention on the development of segmentation methods for basics application. Another report will further present a special application with license plates.

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

I. Introduction

In this project, we will look at a specific step of the image pre-processing: the segmentation. 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.

After a segmentation of our picture, we should be able to get the region we select, that means being able to separate it from other details without altering it. Basically, we should be able to draw only the object on a black screen. By doing that we would also being able to gather much more information such as the dimensions or the orientation.

II. Problem definition and overview

Our problem

Our task for this report is the segmentation of two grayscale pictures. Still, as having an algorithm where everything needs to be changed from a picture to another does not make any sense, we will design an algorithm that can segment the region the user wants after setting a few characteristics.

The first one (fig II-1) is representing several 3D objects from above, on a black background. The picture is lightly lighted from the top right which induce some noise on the overall image, especially on the top right and on the shapes angles.

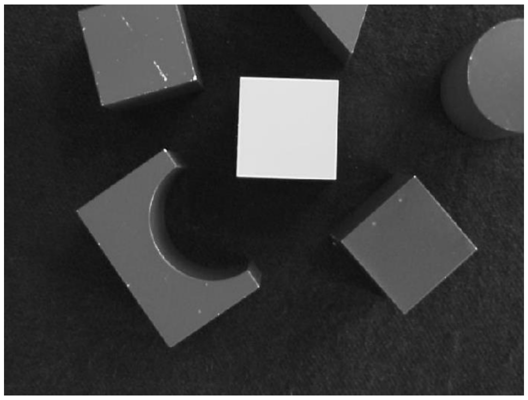

Figure II-1 First image processed

The second one is a really noisy image of what looks like a protein. This picture probably remains from a medical or biological application, where the microscopic image gets stretched to be displayed with a correct resolution. The limits between the two regions is really noisy, so we will need a method that is not affected that much by blurred edges (by not clearly defined edges).

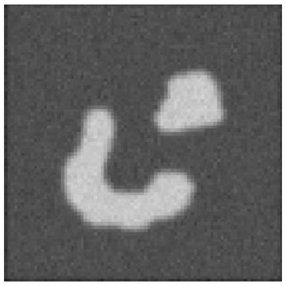
Figure II-2: Second image processed

Means to arrive to our goal

One fact about those grayscales pictures is that they are really simple. The information is coded with grayscale levels and not with RGB or particular textures for example. As the intensity levels' characteristic is the only thing that we can find from one to another, it makes sense to use it in our selection. The method used in this case is called thresholding.

Thresholding is using a value given by the user to split the multiplicity of intensity levels (256 form black to white) in a binary image. From 0 to the value given by the user, every pixel that are in this interval will be dropped to 0. From the value to 255, every pixel with an intensity value in this interval will turn white. This can be complexified by using several values to make multiple selection or interval selection.

The segmentation is still not the only important process to achieve our goal. Pre and post-processing are both simple but important parts of our project that we will explain later.

Difficulties

Basically, this is the weakness of segmentation. Every noisy area where it is really difficult to define some precise characteristics are most of the time getting several errors since it is difficult to discern two regions. On another hand, blurring is also an interesting and well used filter that remove a great part of the noise on most of the picture. This way, it must not be to hard neither inexistent for the process to be valid.

Another difficulty we will encounter is just about the reusability of this algorithm. As we want it to be able to adapt to the situation, the algorithm will use some basic technics that does not depend that much of the overall system. Indeed, those pictures can be far different from one to another, as we can see on our situation. This way we can not design a complex algorithm that will need to be changed a lot between each picture.

Applications

For everything to work correctly, we will give as input of the algorithm a grayscale picture, two threshold values and a seed regarding the size of the region the user wants to select for postprocessing. At the end we must get the region that match our conditions.

This application is really interesting since it can solve some basic segmentations problems really fast on almost every application. In addition, these grayscales pictures are used for a large number of topics such as medicine (e.g. RMI), production or metrology. As explained in the introduction, further development can be easily implemented controlling the conditions of the registration of the image regarding the application, that means our project is giving a basis that have a good successful segmentation rate for every application and can further be develop to earn better results.

III. Algorithm

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

Overview of the algorithm

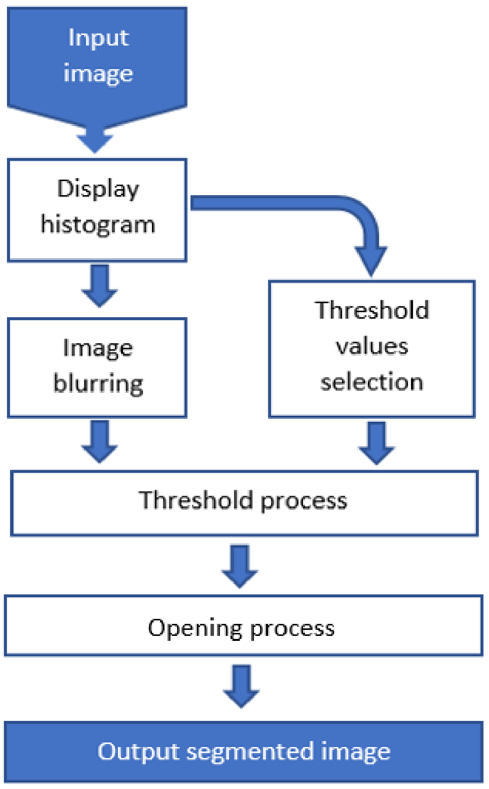

Figure III-1: block scheme of the segmentation algorithm

Our application is made of three main parts: pre-processing, segmentation, and post-processing. It can be resume by this block scheme:

This algorithm will first ask the user for a picture. When it is red using matlab function imread(), it will compute and display the histogram of the picture. From this histogram, the user will be able to choose two values between which the region he wants is included.

Then the picture is blurred and using this new picture and the threshold's values we can process to the threshold selection.

Once this is done, the picture is going into the post-processing part which consist in morphological processing. The closing process help us to have a clearer result since it removes the remnant noise and smooth contours.

If needed, we can at the end get the real image with only the region we want by inverting the image we get from the previous step and subtracting it to the real picture. Even if it is not mentioned in the block scheme, we will take some time to talk about that and try it on our pictures to see the result.

Histogram and threshold values selection

The histogram of a picture is a graph that show the number of pixels as a function of the intensity level. We can see an example right below, in figure III-2:

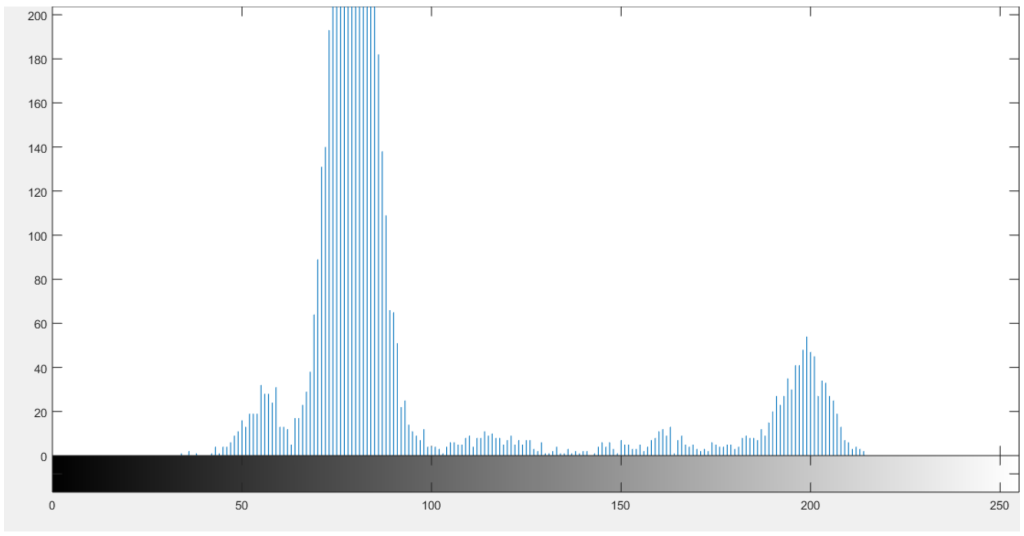

Figure III-2: Example of a histogram

The histogram is used to determine some areas by intensity. In fact, more pixels occupying a pixel intensity interval are supposed to be in the same area for those simple grayscale pictures. For example, in our images with several pieces on it, if some are in metal and other in plastic, the metallic pieces are supposed to be brighter (at a medium-low level of intensity) so we can select them using this concept.

Basically, on this example, if we want to select the area of pixel that is kind of gray (the huge peak we can see on the left), we will choose our threshold value to be between 60 and 100. We will see later what this implies. Those are two of the three values we ask the user to tune.

Image blurring

As explained earlier, blurring the image is a process that can help us to remove the noise on a picture. In the code, the filter used is a gaussian blur. Basically, that means that we process to a convolution between our image and a gaussian function.

Still, this is a low-pass filter so we have to be careful with the use of this one since it can destroy every detail in the image. Indeed, a low-pass filter is erasing high-frequency values, that we can traduce of high variations. That means if the variations between two values is to high it will block this value by smoothing it, resulting in a global blurring of the picture.

Threshold process

Even if the principle of threshold is still very simple, several possibilities are available. This function is made so when we give it a grayscale value and two threshold values, it will reject to black (intensity level 0) all pixel that have an intensity value outside of this interval, and set to white (intensity level 1) all pixel that had their initial intensity value in this interval.

From a grayscale image with 256 levels inside, we have now a binary image in which the white area represents the region we want to select.

Closing of the image

Morphological operations are useful when it come to enhance some binary images. Indeed, it helps to clean borders of regions, to remove noise induced by the threshold process, to fill regions…

Actually, morphological processing is using mostly two functions. The first one, the dilation, is growing the regions according to a seed. That means everything gets filled and closed, but at the end the region is also bigger. The second one is the erosion, which do the exact opposite, removing noise and thin areas.

When combining both, we are able to use each of their qualities and this way enhance our region selection, without getting the size changing drawback. Still, the region can be lightly altered depending on the seed used, especially its contours and corners.

The seed is playing a huge role in this process. This more or less little array will pass over the all picture. To explain the process, we will consider that the information (i.e. the selected region and the seed) will be white, which means a 1 in binary. In dilation, if the origo (the center in most cases) of the seed (1) is equal to a pixel in the picture, we will transform every pixel of the processed picture in range of the seed by 1. The erosion is doing the same, but the condition to apply is that the seed must be different, and it sets every pixel in range to 0.

Processing to an erosion function and then to a dilation function is called opening, and the opposite is called closing.

Here we chose to process opening since it allows to remove noise without paying attention about what is closed to the contour of our shapes. In fact, another process will induce the fusion of our objects with the noisy areas close to them and our result will be bigger than it should be.

Getting the original texture

As an additional post processing, it an be interesting in several applications to display the real initial object, maybe if we later want to process recognition on it using other characteristics.

We can do that by using the binary picture as a mask. To do so, we need to invert the initial picture, so that every binary pixel with a value equal to one will be zero and reversed. Then we subtract this mask to the initial picture (subtracting the region pixels by 0 and the others by 255). At the end we got the initial object on a black background.

IV. Experimental results

Test method and criteria to check the algorithm

To test the algorithm, I used several grayscale pictures that are far different to each other. Some are close to the issues exposed earlier, others are casual. I change values of the threshold according to the histogram each time and I change the size of the seed. At the end I get a result of the picture on which I check several criteria:

- Is the segmented region close to the object contours?
- Are the contours altered?

- Is the final image noisy?
- Is the process long to run?

All those factors will help us to know if our algorithm can be trusted for further applications.

Test results

Since we just want to talk about the results of the algorithm, only the initial and the result of each picture will be displayed. A discussion on results and the experiment settings will be also mentioned.

Test 1: *figures:*

This picture is usual, it only shows some kind of noisy areas.

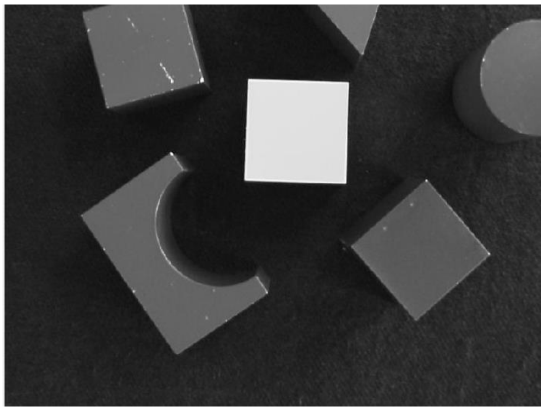

Figure IV-1: Initial

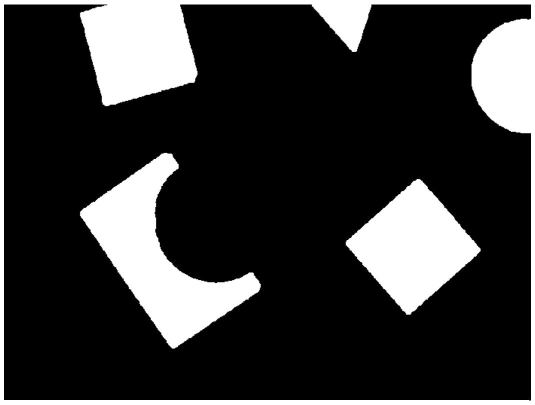

Figure IV-2: Processed

On this picture, we chose to select those gray areas, selecting th1=80 and th2=160 with a circled seed of 5-pixels radius.

We see that the regions correspond to the initial ones, that the image is not noisy at all and that the process was really fast. Still, one negative aspect is the angles of this "U" shape piece that are a bit rounded by the seed. Changing the seed does not help in this case, as the shapes are all in different orientations.

Test 2: *noisy shape:*

This picture is interesting since we can test the resistance of our algorithm to weak contours. As we do not see the exact contour it will be difficult to say if the contours are altered, but the first criteria can still be analyzed since we have enough information.

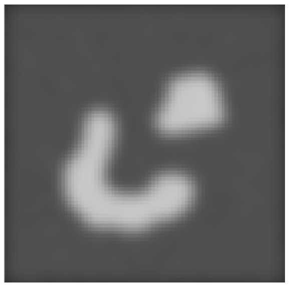

Figure IV-2: Initial

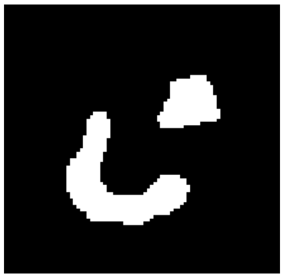

Figure IV-1: Processed

We get these results with th1=180, th2=220, and a 1-pixel radius seed (a cross).

On this process, we see that seems to be really close to the first image, which is a good point for the resistance of the algorithm to blurred or noisy pictures. Once again, we can say that contours are not really smooth, even if we don't have the clear picture to compare to. Anyways, this code is running even faster on this picture, since it is only an 84x84 picture.

<u>Test 3: *fingerprint:*</u>

This picture is interesting because a lot of details are present on it. It is easy to process the segmentation of it since all of the background is white, but it is helpful to check if the algorithm is altering the information that much.

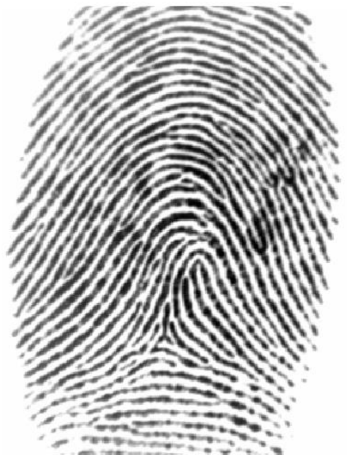
Figure IV-3: Initial

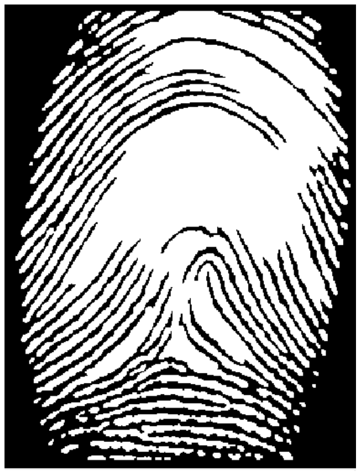
Figure IV-4: processed

First I used th1= 0 and th2=220, with a circular seed of 1-pixel radius. The result is deteriored by the function that I use to fill the region to remove the noise present inside the regions I want to select. The I tried with a lower threshold level, but information starts being lost on the peripherals areas. The best I could fine is th2=160.

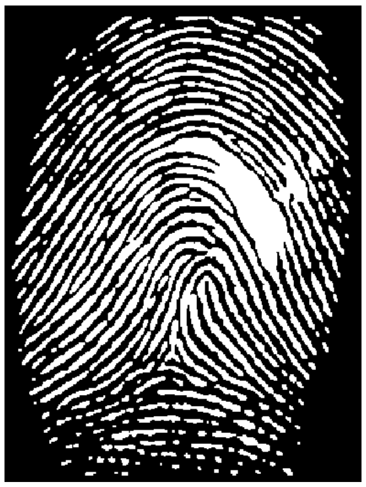
Figure IV-3: Final processed

Still the result is quite good but is not maybe the best for this application. Further modifications have to be done in the post processing, as removing the filling function in this case.

<u>Test 4: *Brain*</u>

This image is used as an application test, to see on some real case if the segmentation is precise enough for some application.

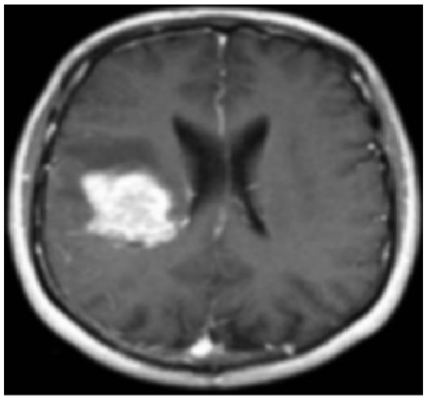
Figure IV-6: Initial

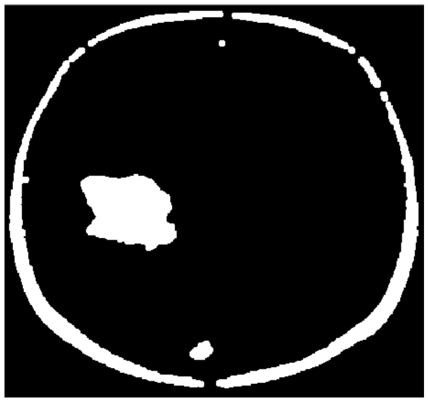
Figure IV-5: Processed

Th1=150, th2=250, seed = circle with a radius of 3 pixels.

The segmentation work well, we can see that the tumor is well segmented. From that we can gather it size and even more if we apply the result as a mask to get the texture and visible details. Still, some shapes can lead the computer to a misstake, for example those litle dots. With an enhanced technology or a better quality of the image it is possible to make the difference between those bones and the tumor.

Results discussion

After testing our threshold segmentation algorithm on several images, we can say that the job is done by this function. It still presents some weakness regarding the fact that it affects the contours sometimes badly, but most of the regions tested concord with the segmented areas are really good. In addition, this is done fast and can be used on a lot of different applications.

V. Related and further work

As we saw, thresholding is maybe not the best tool to be used for further development, even if it is working quite nicely on every images, it is sometimes a good idea to use more complex methods.

Some methods are high variations in the array to determine the limits of each regions. It is the case for example of the region growing and the snake contours.

Others method are using directly properties of objects. Some are really basic, using the size of those areas and other simple characteristics, and others are using some clusters to compare characteristics of each regions via a feature space.

However, the algorithm we have is far enough to segment this kind of simple grayscale pictures, since it allow a really fast process and then a high speed production for example. A point we can try to improve is using a region growing algorithm to be able to chose from different objects that have the same intensity level. In addition, being able to automate a bit more those characteristics can be really useful, even if most of the time those characteristics that were changing between each processing are fixed by the application.

VI. Conclusion

The algorithm we designed using thresholding selection allow the user to segment some basic images to use machine vision and intelligent system in several domains with good performances.

Indeed, the algorithm is running really fast allowing the user to gather more images and therefore accelerating the process which the intelligent system is checking, with a correct selection that does not lead to a lot of errors. Therefore, this algorithm is really interesting for high rate production to check for first defaults for example.

However, it is maybe not the best method to use if a huge precision is necessary, or the user has to use a high-quality camera to have more information and to be able to gather more data from the image.

Once again, the application can fix several unknown factors and therefore enhance the results of this algorithm. Some improvements can still be done regarding the choice of the parts that have the same texture, for example using the region growing algorithm or some region properties function which is even better, as we will see in the license plate recognition project.