Entropic Object Detection Applied to the Game of *Pétanque*

Axel HIPPOLITE
4th Year Student
Informatique Données & Usages
axel.hippolite@protonmail.com

Abstract

This paper presents how to use random variable entropies to calculate the Kullback-Leibler Divergence for the purpose of detecting objects in different images. We will apply these calculations to *Pétanque* in order to determine which ball is closest to the *cochonnet*.

Keywords: Entropy, Divergence, Detection

1 Introduction

As one of the pillars of image processing, object detection is an indispensable field today. Present in autonomous cars, surveillance cameras and social networks, for example, this technology is an integral part of our lives. The existence of multiple processes allowing its implementation makes it a very interesting subject. The aim of this paper is to present the work done on entropic object detection applied to the game of *Pétanque*. In *Pétanque*, the objective is to score points by placing your balls closer to the *cochonnet* than your opponent. Once all the balls have been thrown, the team with the balls closest to the *cochonnet* wins points. Points are awarded according to the number of balls close to the *cochonnet*.

2 Key-Question

In some rounds, the different balls are almost equidistant from the *cochonnet*. It is therefore impossible to choose with the eye to which team to distribute the points. So I was interested in how to implement an entropic object detection algorithm (here the balls and the *cochonnet*) which would allow to show instantly the closest ball to the *cochonnet*.

3 Project Progression

To facilitate its realization, the project has been broken down into different stages :

Data Acquisition
↓
Image Registration
↓
Entropic Detection
↓
Displaying Results

The work done for each step will be presented throughout the sections.

4 Data Acquisition

Before starting to apply the image processing procedures, it is necessary to know which images we are going to work on. However, we will see that some constraints were imposed and that it was necessary to respect them.

4.1 Constraints

As our algorithm will be based on the calculation of the Kullback-Leibler Divergence it is imperative to work on two images. Moreover, in its purpose, the algorithm must be applicable to a situation that could happen during a game. However, it is still necessary to work in relatively optimal conditions in order to be able to optimize the parameters and not immediately work with difficult situations.

4.2 Data Selection

By taking into account the different points previously discussed, it was possible to choose the data with which we would work.

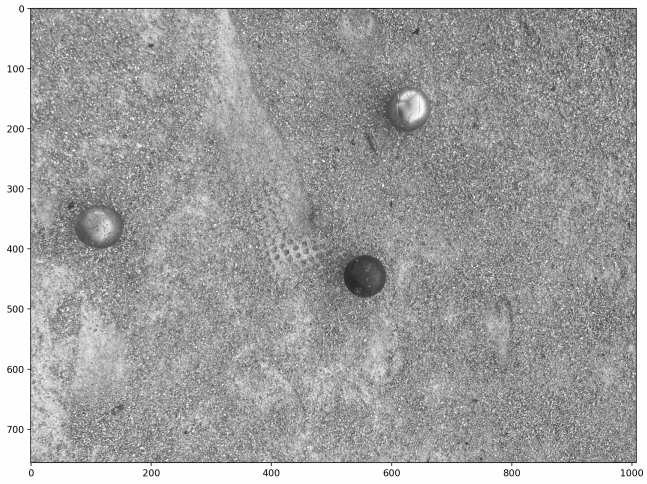

Figure 1: Representation of a realistic situation. We find the *cochonnet* and the balls positioned at different distances from it.

We find in the previous representation an optimal lighting as well as a ball whose color is easily distinguished from that of the ground. However, we also notice the presence of balls that can be confused with the ground and allow us to work in less than perfect conditions.

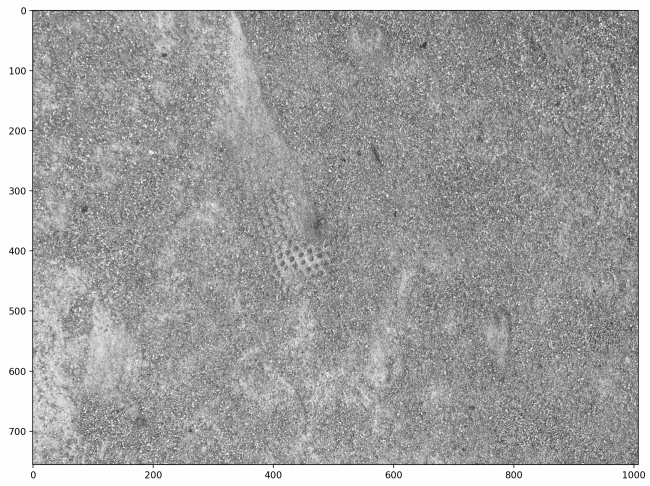

Figure 2: Representation of the same situation by removing all the balls and keeping the *cochonnet*.

This image will be used to calculate the Kullback-Leibler Divergence. Indeed, as the main difference with the first is the presence of balls, the application of this calculation should highlight them. It will also be used to calculate the coordinates of the jack to use them in the calculation of distance from the balls, and also to make the registration of the two images on this point.

5 Image Registration

In order to maximize the Kullback-Leibler Divergence (KLD), it is important to superimpose the two images on the same point, here the center of the jack. This is called image registration.

5.1 Mathematical Morphology

Before explaining the process used for image registration, it is necessary to introduce the concept of Mathematical Morphology which will allow us to facilitate the future treatments that we will apply to the image.

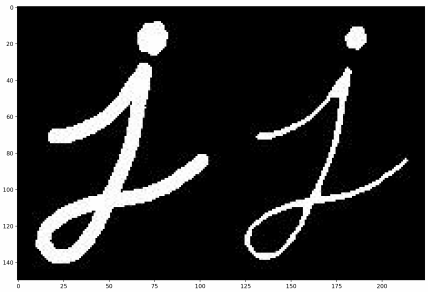

Figure 3: Representation of Erosion.

The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object. The kernel slides through the image. A pixel in the original image (either 255 or 0) will be considered 255 only if all the pixels under the kernel is 255, otherwise it is eroded (made to 0).

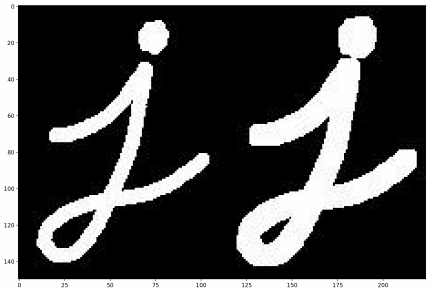

Figure 4: Representation of Dilatation.

Dilatation is just opposite of erosion. Here, a pixel element is 255 if at least one pixel under the kernel is 255. So it increases the white region in the image or size of foreground object increases.

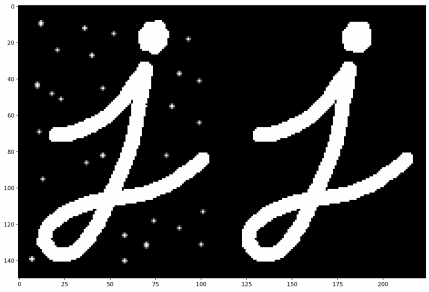

Figure 5: Representation of Opening.

Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. Opening is just another name of *erosion followed by dilation*.

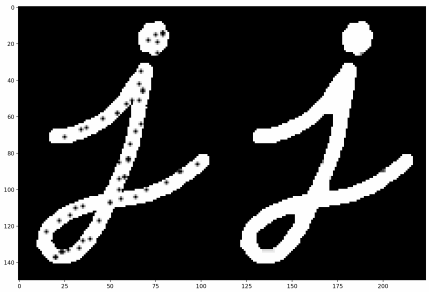

Figure 6: Representation of Closing.

Closing is reverse of Opening, *dilation followed by erosion*. It is useful in closing small holes inside the foreground objects, or small black points on the object.

5.2 Key-Point Detection

We know that an image is composed of three layers : Red, Green, Blue (RGB). The goal is to subtract all these layers between them to highlight the position of the *cochonnet*. The *cochonnet* is always brightly colored, one of these layers will always be predominant. Thus, among the matrices resulting

from these subtractions, one will have extremely smaller values than the other two. We will use this matrix to continue the processing.

$$DiffRed(i,j) = Red(i,j) - Green(i,j) - Blue(i,j)$$
$$DiffGreen(i,j) = Green(i,j) - Blue(i,j) - Red(i,j)$$
$$DiffBlue(i,j) = Blue(i,j) - Red(i,j) - Green(i,j)$$
$$\forall (i,j) \in \mathbb{R}^n * \mathbb{R}^d$$

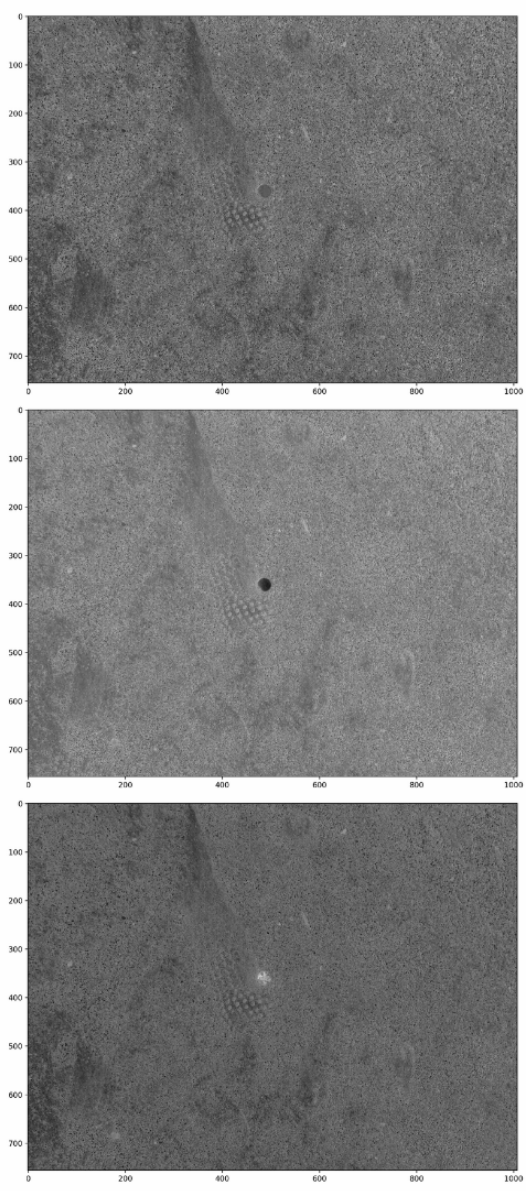

Figure 7: Representation of the 3 matrices resulting from the difference of colors.

We notice that on the difference matrix compared to the Green layer, the values in the vicinity of the jack are much smaller than those of the other difference matrices. We will therefore binarize it with a threshold of 60% of the minimum. To improve the binarization, we will use one of the principles of mathematical morphology seen previously: an opening to eliminate the residual noise.

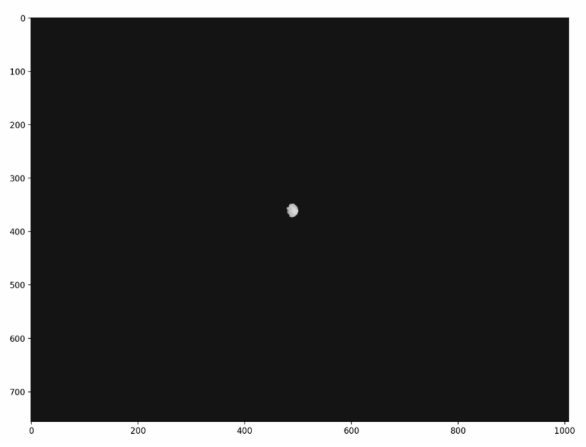

Figure 8: Representation of the Binarized Image.

By finding the contours of our binarised object and then calculating its center of mass, we can easily find and display the center of the *cochonnet*.

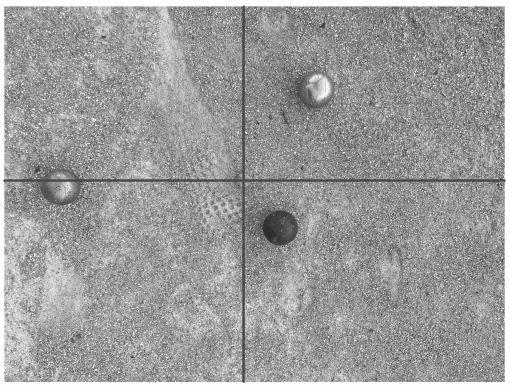

Figure 9: Representation of the Center of the *cochonnet*.

5.3 Reframing

Once the coordinates of the *cochonnet* have been obtained, all that remains is to crop the two images so that they have the same coordinate system. In other words, the correct number of rows and columns are removed from each image so that the *cochonnet* has the same coordinates in both images.

6 Entropic Detection

The aim here is to detect changes between the two images: the one with and without the balls in order to highlight their positions.

6.1 Pooling

Pooling consists of going through the matrix of an image with a kernel (here 3x3) and associating to each pixel a variance and an average from the values of the kernel. By repeating this process on the two images, we obtain two pairs for each pixel, each containing a variance and a mean. It is these two pairs that we will use to calculate the Kullback-Leibler Divergence.

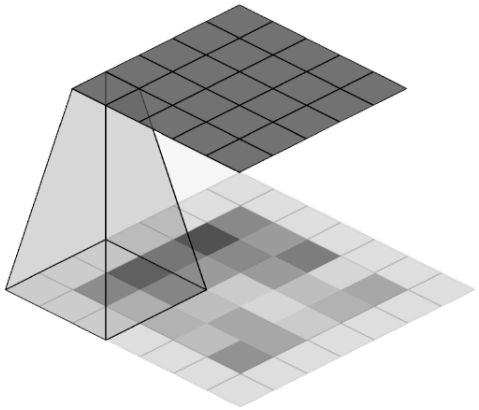

Figure 10: Each cell of the new matrix contains the mean and variance of the 3x3 kernel values.

6.2 Kullback-Leibler Divergence

Assuming that the distribution of the kernel values follows a Gaussian distribution, we could have the classic definition of the Kullback-Leibler Divergence. However, keeping the mean in the equation will create a bias because whether there is a ball or not the mean can be the same. It is therefore important to remove the terms with the mean :

$$KLD(i,j) = \frac{1}{2} * \left(\frac{Var_{2(i,j)}}{Var_{1(i,j)}} + \frac{Var_{1(i,j)}}{Var_{2(i,j)}}\right)$$

Thus, we will calculate the divergence between the variance-mean pairs of each image. As the calculated values are very close, it is difficult to discern variations in the divergence. We will therefore calculate the log of the divergence: this will allow us to stretch the values and thus accentuate the variations.

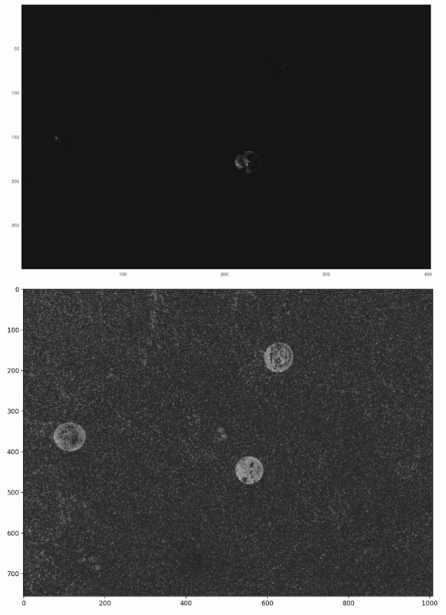

Figure 11: KLD without *log* (Upper Figure) VS. KDL with *log* (Lower Figure).

The image is then binarised with a threshold of 60% of the maximum divergence value before applying an opening to remove residual noise and a closing to reconstruct the interior of the balls.

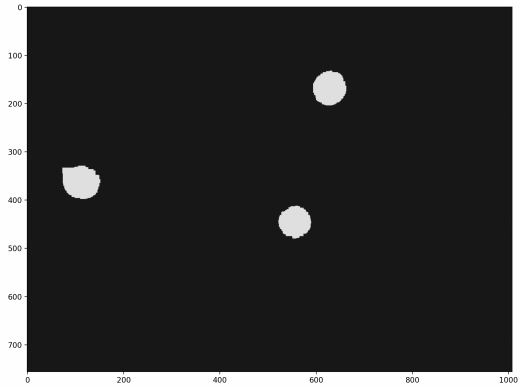

Figure 12: Representation of Binarised Divergence.

6.3 Displaying Results

By calculating the contours and centres of mass of each region of interest, the coordinates of each ball in the image are easily obtained. All that remains is to calculate the distance between the balls and the *cochonnet* and to highlight the closest ball.

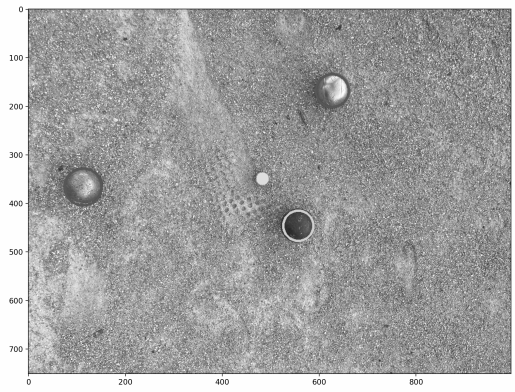

Figure 13: Finale Output.

7 Tests

The test protocol consists of placing two balls at a certain distance from the *cochonnet* and reducing the difference in distance between the balls to see how the algorithm reacts. The tests are carried out in optimal conditions : plain background and perfectly uniform *cochonnet*.

Tests Results	
Distance Difference	Right Decision ?
100mm	Yes
50mm	Yes
5mm	Yes
1mm	No

8 Limits

We have seen that the algorithm is relatively efficient. However, the environment during data acquisition can very easily disturb the results obtained. Indeed, depending on the inclination of the sun, the shadow of the balls projected on the ground will erroneously detect the object: not only the balls but also their shadows will be detected, thus modifying the calculation of distances.

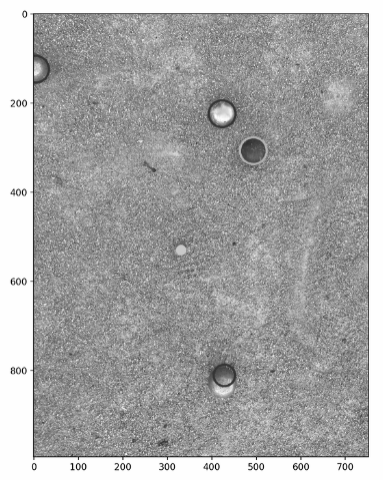

Figure 14: We can see on the ball at the bottom of the image that the area detected corresponds to the area between the ball and its shadow

A second problem is the execution time of the algorithm: even by reducing the size of the image the result is obtained only after about ten seconds, which remains problematic if one wants to use it during real games. It is therefore necessary to find the right balance between image quality and execution time. Another element that makes it impossible to apply this process to real games is the fact that you have to use an image with the balls and an image without the balls. Indeed, it is unimaginable to remove the balls to take a picture during a round.

9 Conclusion

At the end of the day, even though the algorithm proved itself in the tests under optimal conditions, we saw that there is still room for improvement. It would also be interesting to continue the tests to see how the algorithm reacts in an unfavourable environment and perhaps to transform it into an application to generalise its use.

References

OpenCV (V2) *Documentation*, Morphological Transformations.

Axel Hippolite, *github.com/AxelHippolite/DATA731-EntropicObjectDetection*.