A simple detector for crowd counting using OpenCV and Python

Martí Gelabert Gómez

November 14, 2022

Contents

1 Introduction

The assignment consists in **counting** the number of people that appear on the given images. There was not an stablished way to aproach this problem. In this case I have taken the decision of using the computer vision algorithms seen in class and not relay on deep learning or any kind of artificial inteligence algorithm. This is because not by de amount of data needed for training, due to there are really god datasets out there, like MS COCO, but because of the effort of building a good performing detector architecture. Use an already set framework as YOLO or Mask-RCNN would be the way to accomplish the task, but I would be thinking about it as "cheating". Therefore, this way, using the content seen in class, I will be more cautious about my decisions and It will be much easier to justify them.

In the following document we will be focusing on the process taken and the algorithms used, their implementation and their performance.

2 procedure

The program takes the following steps :

1. Import images as black and white.
2. Apply Adaptative Histogram equalization to the images.
3. Substract the background to the images using the image with the background.
4. Apply an erosion algorithm to obtain small highlighted areas.
5. Apply a thresholding algorithm to binarize the image.
6. Apply a dilation operation into the binarized images to expand the whites.

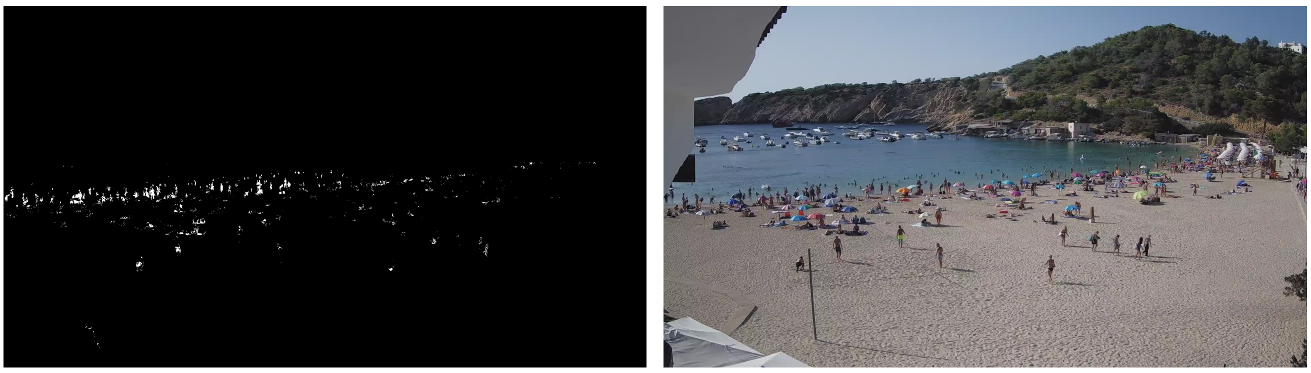
7. Use a contour algorithm to extract the diferent regions containing persons.
8. Obtain the bounding boxes from these regions.
9. Count them and compare the number of detections to the real cuantity.

3 algorithms

In this section we will discuss the algorithms selected and the output we obtain from them in the application of our problem.

3.1 Background Removal

Background removal is a technique that allows to remove the background from the image, this way, the output will be only the foreground highlighted in the form of a binarized image as it can be ilustrated on the figure 1.


(a) Output of background substraction using cv2.threshold(image, 100, 255, cv2.THRESH_BINARY)

(b) Original 1660320000.jpg

Figure 1: Reference images

CLAHE

In the case of our problem, first we have imported all the images in black and white integer values using `cv2.imread(img,0)`. Then for a better extraction, it has been applyed an **algorithm of CLAHE** seen previously on the course to try to uniformize the ilumination of the images. With a more uniform images, we will be able to distinguish more precisely our foreground and artifact like shadows casted by objects on the images may have less impact on the **binarization process**.

Background Image

For the selection of our background image, in a starting point the **average image** was computed as the main way for accomplish foreground extraction, but the resulting image generally was really noisy and with ghosting effect very present on it, which decreased the performance of the algorithm as it can bee seen on the Figure 2. Even with a gaussian blur applyed to try to counter the ghosty figures, they are still really present on the image. The image averaging is a good idea generally for noise reduction and in this case to try to obtain a neutral image were the background is predominant, but there is not enought images to obtain a more sharp image to really have a good background. Therefore, we **end up** using the image 1660284000.jpg with a gaussian blur applyed as our background, this way we obtain better results on the **binarization process**.

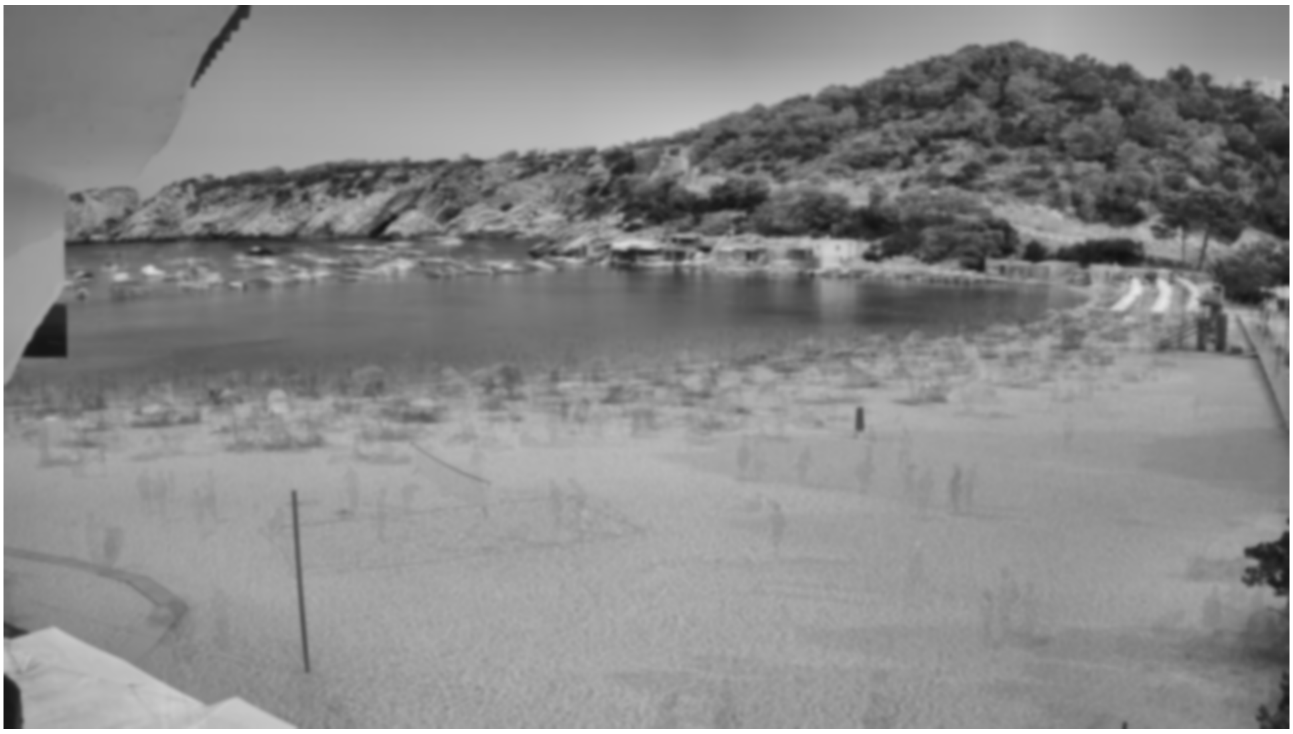


Figure 2: Average image with a gaussian blur applyed with a kernel of (11,11)

Substraction

Once we have selected our background, we should substract every image with our background, this way what we obtained has the foreground (in this case the persons) highlighted as we can see in the figure 3.

THRESH_OTSU funciona bien salvo en los casos donde empieza a haber sompras en la imagen, a la hora de aplicar el thresholding obtendremos un número muy grande de falsos positivos en las pequeñas montaás que hace la arena, las dimensionas que tienen son similares a las de una cabeza humana por lo que si las eliminasemos estariamos eliminando buenas deteccionas.

3.1.1 Image Averaging

3.1.2 Image Substraction

4 Gabor Filter

A band pass filter generated by a function of various parameters.

$$filter(x,y;\sigma,\theta,\lambda,\gamma,\phi) = exp[-\frac{x^2-\gamma^2\cdot y^2}{2\sigma^2}]\cdot exp[i(2\pi\frac{x}{\lambda}+\phi)] \tag{1}$$

The parameters of ksize allows to select the size of our kernel filter, in case we are using a really big shape we will overlook details if the shapes are small. The same reasoning can be applyed with a small filter, may overlook shapes too big for it. Therefore, must be tested with different sizes to reach an idoneal spot, if your features are tiny or bigger, you have to take that in count.

If we are looking for **horizontal-like** features, applying an horizontal filter will allow us to maintain those characteristics and block the vertical ones and viceversa in the other cases.

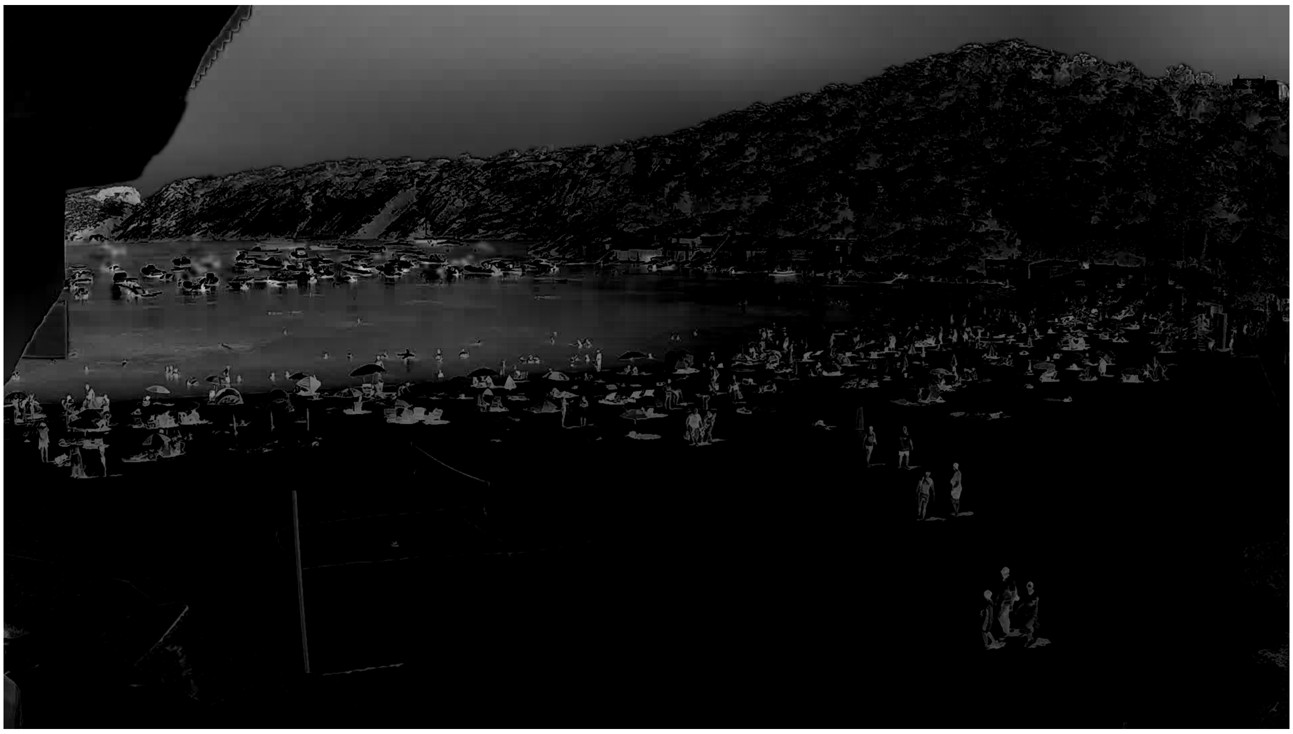


Figure 3: Image obtained by applying cv2.subtract(background, image)

5 What form do the People have?

In general the shapes that a person can describe may suffer alot of distorsion depending of the angle where the frame was took, the person posture, etc. Not always will be a perfet pose to the researchers to easily identify if an object is a person or not. There is no key shape that we could use, but we can try use common sense. The images gathered, are related to people moving standing up, swiming or just taking sunbathing.

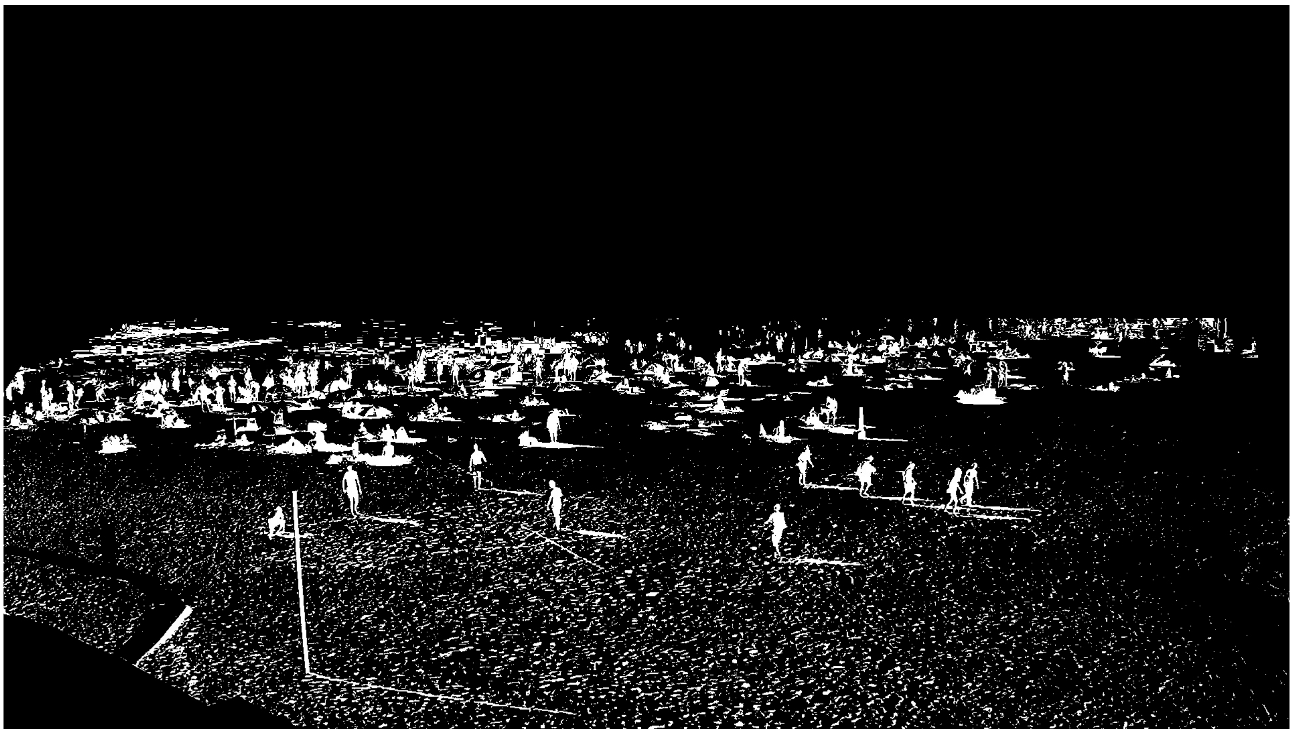


Figure 4: Binarized image using OTSU and the empty beach image