Image and Video Analytics Project 1

1. Introduction

The first project of this course is about object detection on a certain topic of interest from images. It is possible to do so without machine learning methods and the purpose of this project is to answer the question of "Can we apply the existing image analytics methods to provide an estimation of the number of people that are present in a certain image?". This project could be useful for lifeguards as they might need a tool to gather information on the occupancy of beaches automatically. This report will serve as a documentation of this project which consists of introduction, data description and analytics, implementation, results and accuracy, and conclusion.

2. Data Description and Analytics

The dataset being used for this project consist of 10 images of the Cala Vadella beach in Ibiza taken from a live web camera on the 9th of August throughout the day. The first image consists of an empty beach at 7 AM while the rest of the image is taken on the same day with the time being 9 AM, 10 AM, 11 AM, 12 PM, 1 PM, 2 PM, 3 PM, 5 PM, and 6 PM. The Images that were taken from 9 AM until 6 PM contain at least one person in the image. Each of the image containing people is annotated based on the position of the head and this data is saved into a CSV file consisting of the coordinate of each person's head position including the image file name.

3. Implementation

The implementation part is divided into five parts these are removing the background, binarizing the image, image post-processing, small region processing, and contour detection. Figure 1 shows the original image for image 1660050000.jpg which will be used throughout this report for an example.

Figure 1: Image 1660050000.jpg

a. Background Removal

For the first part on removing the background, since most of the crowd resides only on the lower half of the image, region of interest (ROI) can be defined. The empty beach image can be used for background removal. For each pixel of an image, the function `cv2.subtract()` were used with the empty beach and the result will be a temporary foreground. This foreground still has unnecessary ocean part which can also be removed by filtering out the pixels within range of the ocean pixels. To do so, by defining the pixel ranges of the ocean, the function `cv2.inRange()` will result in the ocean mask, and the ocean region can be achieved by using `cv2.bitwise_and()` to the image and the mask. Finally, the temporary foreground was subtracted with the ocean region and by doing so, the resulting image would consist of the pixels that are out of range, and these are the actual foreground that will be processed further. Figure 2 shows the results of the image after background removal.

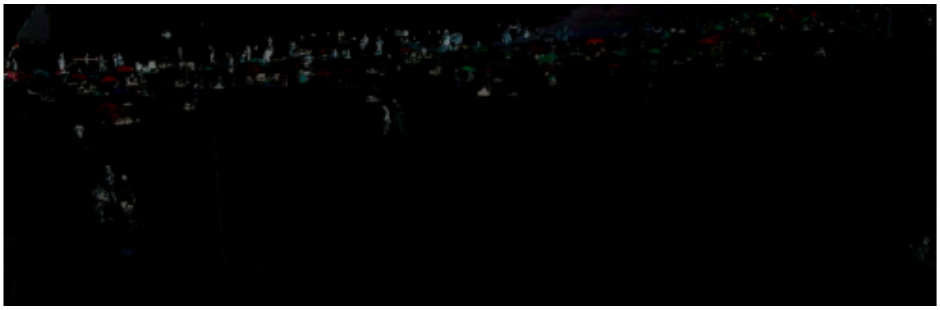

Figure 2: The ROI of image 1660050000.jpg after background removal

b. Binarization to the ROI Foreground

The binarization method being used for the foreground was adaptive thresholding. The reason for this is that adaptive thresholding selects an individual threshold for each pixel based on the range of intensity values in its local neighborhood. This allows for the thresholding of an image whose global intensity histogram doesn't contain distinctive peaks. Just like detecting text on paper, adaptive thresholding is useful for detecting details for images at different times of the day because of the sun's position. As for this project, the crowds during a different time of the day can be discovered using adaptive thresholding, first do a gaussian blur to the image, and then apply the function `cv2.adaptiveThreshold()` and the resulting would be a binarized version of the image as can be seen on Figure 3.

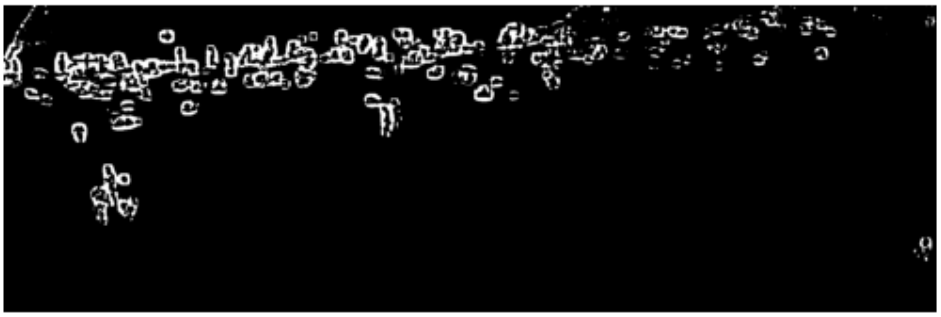

Figure 3: The binarized ROI of the image 1660050000.jpg after adaptive thresholding

c. Image Post-Processing

Further processing after the binarization consists of morphology operations to polish the resulting binarized image by calling the function `cv2.morphologyEx()`. The first function being used was the `cv2.MORPH_ERODE` with 4x4 kernel to remove the small white noise and

also split the detected binarized crowds so that the contour can detect them as separate. The second function was `cv2.MORPH_DILATE` with 3x3 kernel to make the remaining white region bigger, because the erosion from the previous function might shrink the actual object. The resulting morphed binarized image can be seen in Figure 4.

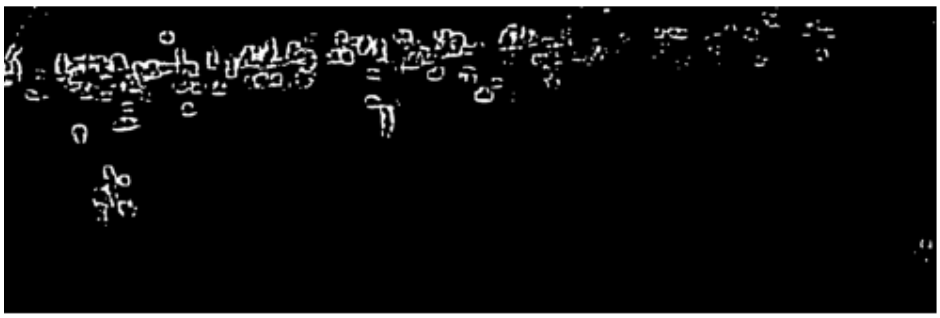

Figure 4: The binarized ROI image 1660050000.jpg after morphology operations

d. Small Region Processing

As can be seen in Figure 4, the region near the lifeguard section is almost completely erased including the crowd that is supposed to be detected. This can be retained by changing the configuration of the ocean removal values, the thresholding values, and the post-processing methods. To separate the processing, a new region for the lifeguard region was defined and a then similar process was conducted with different values for the background removal pixels and thresholding values, and only erosion was applied. In this way, the crowd near the lifeguard region can be detected without adding noise to the other region.

e. Contour Detection

The final part of the implementation is detecting the contour from the binarized images which was done using the `cv2.findContours()` function. To draw the contour with the bounding box, the function `cv2.boundingRect()` was used and the resulting values are the x and y coordinates of the starting point of the rectangle and also the weight and height of the rectangle. For the drawing, to make sure that the detected contours are not noisy, the area of the contour was calculated using the `cv2.contourArea()` function, and if the area is more than a certain value and less than a certain value, then it will be recognized as a person and was added to the list of boundary boxes, this is done similarly to each region depending on the distance of the region to the camera location.

The resulting image and the performance analysis of this implementation will be discussed in the following section.

4. Results and Accuracy

After doing the implementation shown in the previous section, the resulting image with bounding boxes of the recognized person can be seen in Figure 5.

Figure 5: Final result of the person detecting algorithm on image 1660050000.jpg

To measure the performance of the algorithm that was used in this project, the mean squared error (MSE) was used for the image level and the average accuracy metrics were used for the person level.

a. Image Level

Mean squared error (MSE) is the average of error squared, which are the average squared difference between the estimated values and true value. In this project, it means how big the error in the estimation of the crowd detection using this algorithm. To calculate the MSE, the actual number of people manually annotated in the labels CSV will be compared to the number of bounding boxes generated in the algorithm. The resulting MSE value for this project is 4327.67, it is relatively huge and the reason for this is the algorithm also still has some white noise as a person which is quite difficult to remove for some images. Another factor is for some people the algorithm generates more than one box because, during the processing part, the algorithm splits the person into more than one part, and the contour detects each of the parts as a different unit which would result in multiple boundary boxes.

b. Person Level

The metrics being used in this project are Recall, Precision, and Jaccard Index (JI). To do each of these, there needs to be a definition for True Positives (TP), False Positives (FP), and False Negatives (FN). In this project, TP is defined as the number of manually annotated persons detected in the final image, this can be measured by checking if each of the coordinates of a person exists in the final image. This can be done by looping through each coordinate and checking if any of the centers of the bounding box coordinate has the same value as the actual person coordinate. Since the labeling was done manually, a tolerance of distance between the actual and the detected bounding box is defined with 11. Hence for each actual person coordinate, if there is a bounding box that has a center coordinate value that has a difference of less than or equal to 11 with the actual person coordinate, the TP value will be incremented by

one. FN is defined as the number of annotated people that are not detected in the final image. FP is defined as the number of bounding boxes that are not representing the actual person, these can also be achieved by subtracting the number of bounding boxes from the TP value. With these metrics being defined for this project, the metrics can be calculated. The recall value of this project is 50.21% and the precision is 35.03%. The algorithm has a medium recall value and a medium-low precision value, which means that the algorithm considers most of the things are "persons" however this includes the actual "person" and the non-actual "person" which includes umbrellas, shadows, and ripples on the ocean side. The JI serves as a way of conceptualizing the accuracy of object detection. The JI value in this project is 20.56% which is quite low and the reason for this is the large FP values of the noises that are produced due to some images that need different treatments.

5. Conclusion

In this project, the question that was raised in the first section had been answered by looking at the implementation in section 3 and the results in section 4. The algorithm works sufficiently well with a medium recall value and medium-low precision value, although for the Jaccard Index (JI), the value is low. This algorithm is more suitable to detect crowds instead for counting the number of heads because, in this way, the small noise can be further removed, and the bounding boxes can be combined. In that way, to measure the accuracy, the area of the bounding boxes can be compared to the actual one. Removing white noises becoming a dilemma because in some regions the crowd is much smaller than the one closer to the camera. As for future improvements to the algorithm, region sectioning would be the answer to further polish the accuracy, and further image processing techniques could be explored. Overall, it is a fascinating project, and the results are quite satisfying.

The code can be accessed in https://github.com/realr3fo/Analysis_Image_Task_1