Object Detection and Classification

Baglan Aitu and Alberto Ruiz

I. INTRODUCTION

In this assingment we have developed routines for blob extraction using Grass-Fire algorithm and classification using the output of the *Mixture of Gaussians* (here and henceforth *MOG2*) background subtraction module provided by OpenCV, along with detecting stationary foreground blobs in new provided video sequences.

II. METHOD

Methods implemented are mainly two. These both are implemented after we do the foreground segmentation (implemented in previous assignment, we won't focus on this), and are blob extraction and blob classification. In blob extraction, we are able to detect "blobs", but not to distinguish what they are. Blob classification will be responsible of it.

A. Blob extraction

We were required to implement a routine for *blob extraction* based on the ***sequential Grass-Fire*** algorithm. This technique is used to extract blobs, and consists in traversing the image pixel by pixel with a connectivity element. Pixel-wise speaking, we examine its neighborhood and check whether similar pixels exist. If do, they are labelled and removed from the frame. If not, continue the traversal.
In addition, task 1 also includes a template for small blobs removal, which is just remove those blobs lower than a threshold.

B. Blob classification

Once blob are obtained, next task is to develop a classifier, in order to distinguish them. In this case, a simple Gaussian classifier is used. We are given mean and standard deviation of the possible detections (Person, car, object and unknown, bounding boxed with blue, green, red and white, respectively). We compute the distance between the unclassified blob and the model object, and make a decision. Due to working with Gaussians, we check whether the aspect ratio is inside the Gaussian.

C. Stationary foreground extraction

This task is important as it affects to plenty of situations, such as crowded places where a static region is occluded by fast moving objects, causing camouflage errors. In the following section the whole applied technique will be deeply explained.

III. IMPLEMENTATION

A. Blog extraction

The implementation of the sequential Grass-Fire algorithm has been done with the help of the OpenCV *FloodFill* function [1]. This required the structuring element (4 or 8-connectivity) applied to a binary mask. Once blobs are returned (x and y coordinate, along with height and width and id), we store them in the cvBlob struct.

B. Blob classification

The feature for making the classification was the aspect ratio. As we previously have Gaussians for the objects, if the absolute difference between its mean and each models' is lower than the std times a ratio parameter, then it belongs to one of them. If we get the particular case that it belongs to more than one, we look for the closest one.

C. Stationary foreground extraction

The applied technique for stationary foreground extraction follows the article from Ortego et al. [2]. The aim here was to get the *Foreground History Image* (*FHI*) by measuring the foreground temporal variation of a frame sequence. It was given by the following formulas:

$$FHI_t(x) = FHI_{t-1}(x) + w^f_{pos} * FG_t(x) \tag{1}$$

$$FHI_t(x) = FHI_{t-1}(x) - w^f_{neg} * \tilde{F}G_t(x)) \tag{2}$$

$\tilde{F}G_t(x)$ indicates the logical NOT of the operation, w^f_{pos} and w^f_{neg} are two weights to manage the contribution of the foreground ($FG_t(x) = 1$) and background ($FG_t(x) = 0$) detections. Sometimes it was required to increase the ($FG_t(x)$) if it belongs to foreground and reset it otherwise. Nevertheless, temporally sparse errors in foreground detection might cause loosing correct stationary detections. Commonly happens in crowds were a static region is occluded by fast moving objects which cause camouflage errors.
Once $FHI_t(x)$ was obtained, we normalized to [0-1] range and work with the frame rate as below formula shows:

$$\overline{FHI}(x) = min(1, FHI(x)/(fps * t_{static})) \tag{3}$$

Finally, we normalize to get $\hat{FHI}_t(x)$. If this is greater than a threshold, then it is foreground. It will be background otherwise.

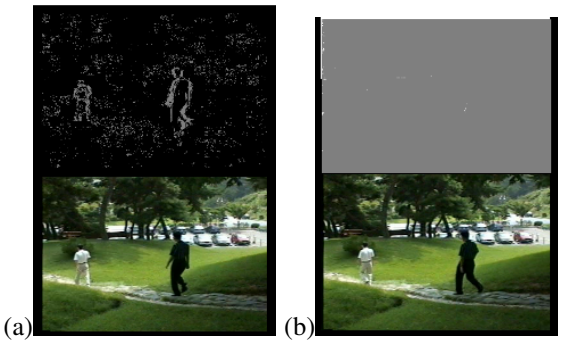

Fig. 1: (a) η = 0.5 (b) η = 1

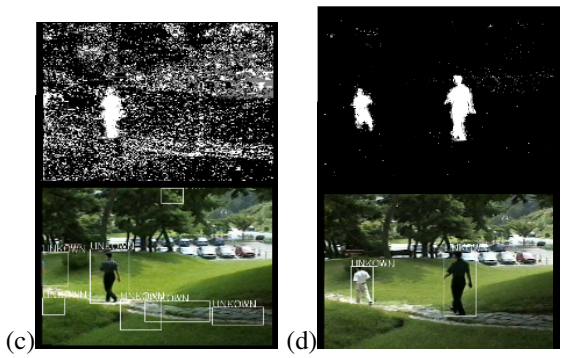

Fig. 2: (c) η = 0 (d) η = -1

Fig. 3: (a) τ = 10 (b) τ = 15 (c) τ = 20

D. Running the code

To implement the algorithms, the first thing is to choose the correct path for the input (video files) and output folders (the folder where results are stored) in the main function. The user can change the values of parameters at his own discretion. Once this is made, we create the appropriate makefile to just type 'make' in terminal to compile and build the program. And lastly, type '*./xxxx*' being *xxxx* the name of the .exe file.

IV. DATA

In this lab, we have a dataset of 3 videos: *ETRI*, *PETS2006*, and *VISOR*.

The first video group *ETRI* has the walking men along the path in the park. Most of the background environment is uniformly green with minor fluctuations (leaves stirring). Also there are some static objects (cars) behind the park. Some of them are occluded with trees.

The second video group *PETS2006* is more complex than the first video. It demonstrates the metro station with walking people with and without bags, suitcases. Some of them walk alone, and some of them walk in pairs; interact with each other which leads to the occlusion and difficulty of choosing a proper person model. It is need to mention that there are also object reflections which makes a classification task more challenging. Moreover, one person in the center of the scene abandons different objects (bag, suitcase, big board) in 3 parts of the video.

Third video group *VISOR* displays the parking lot with stationary and moving cars. In the upper part of the frames, the traffic road was filmed with moving cars which detection is dependent on the threshold value.

V. RESULTS AND ANALYSIS

A. Blob extraction

We noticed that blob extraction is dependent on the background subtraction (MOG2). The better background subtraction we do, the better blob extraction can be obtained.

Fig. 1 shows poor (a) and no foreground segmentation (b). Consequently, there are no blob extractions at all.

Fig. 2 shows that noisy (c) and correct (d) blob extraction cases. In the case of (c) there are false detections because of low learning value. When the η set to -1, it automatically chooses proper learning rate value which gives correct blob extraction at the output.

These results show that above discussed method is not solution for all "complex" scenarios as overlapped and partial object detections are not supported.

We also tested connectivity types (4 and 8) for the given video groups. Theoretically, 4-connectivity should provide more or equal blobs than 8-connectivity, while the last one should prove this. In practice, unfortunately, we were not able to distinguish any difference.

In Fig. 3 it is shown different cases of small blobs removal. As can be seen, it depends on the set threshold (width, height) value. In the case of τ=10 (Fig. 3, a) there are plenty of false detections. To avoid it and obtain correct results, it is better to set a high value of threshold. In our case, the best value is 20.

B. Blob classification

Blob classification depends on the blob extraction. Fig. 4 demonstrates how false detections of extraction lead to misclassification of stationary objects. In the video (b), there is one stationary false detection in the bottom side of the frame which also occludes people passing nearby. It happens because the width and height of a person seems different to the model. To overcome this problem, the more complex model needs to be introduced which is robust to partial occlusions. For example, the feature based model which detects human body parts [3].

In fact, in this lab blob classification has the similar issues with the blob extraction part as it also depends on the threshold value. For example, in *VISOR* video groups, cars are not classified until they reach a certain value of width and height.

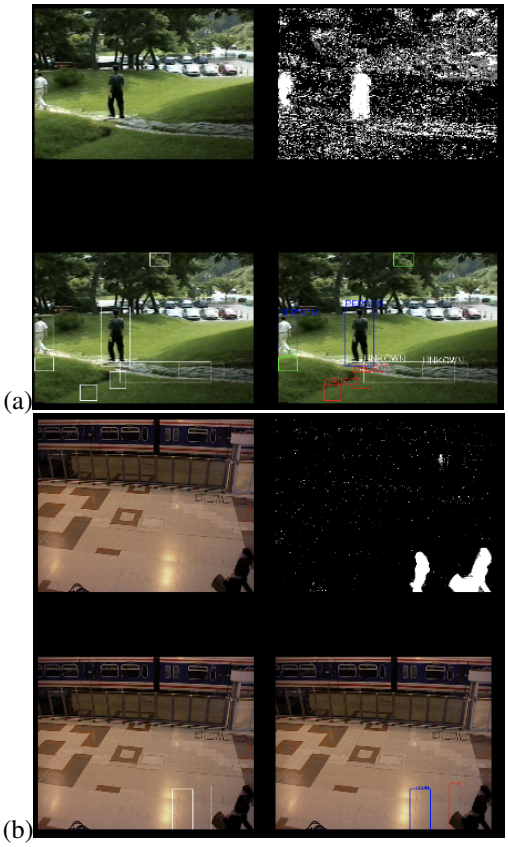

Fig. 4: Misclassification because of false detections

C. Stationary foreground extraction

In this part, the aim is to detect the stationary objects. For that purpose, several additional parameters were introduced. They should be tuned to obtain the best result. For instance, the time parameter which decides foreground object counting as a stationary object.

As [2] states, the incrementation cost need to be as high as possible, while decremention as low as possible. Fig. 5 shows the issues appeared from wrong values of incrementation and decrementation. As can be seen there, moving objects appear (b) and the ghost (a) behind the cars remains on a foreground mask.

As we use MOG2, its learning rate decides how fast the foreground object incorporates into the background model. By trying several values of parameter (Fig 6,7,8), we decided to set the lower threshold value. In that case an object stays longer on the foreground mask which is enough to observe it as a stationary object.

Nevertheless, there is another false detection caused by shadow since it is stationary (Fig. 9).

VI. CONCLUSIONS

The performance of blob extraction and classification depend on the foreground segmentation quality. Algorithm was improved drastically by tuning proper parameters. The learning rate is one of the main parameters as it decides how fast a background learns. The model is not robust since it

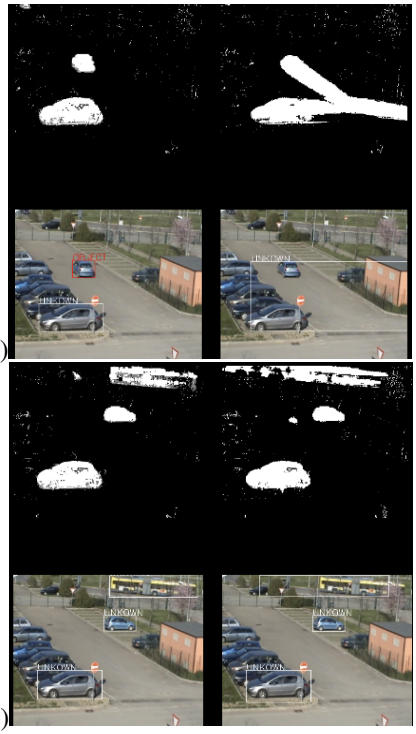

Fig. 5: False detections of stationary objects

Fig. 6: No detection (η = 0.1)

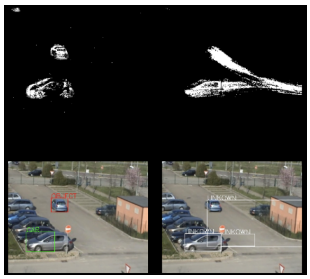

Fig. 7: False detections (η = 0.001)

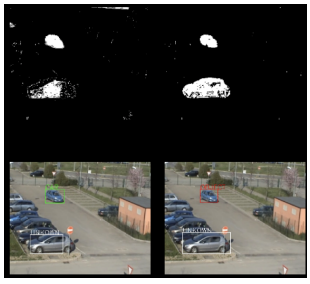

Fig. 8: Correct detection (η = 0.0001)

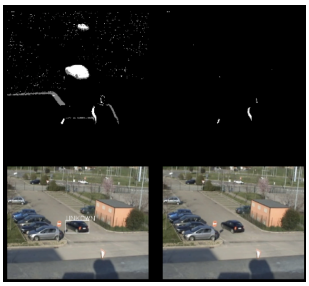

Fig. 9: Shadow as a stationary foreground object

is a holistic model. To overcome partial occlusion issues, more complex (feature based) models can be introduced. The problem is it will be very slow and computationally expensive.

VII. TIME LOG

Split of 50 % in coding, trying results, writing the report.
Task 1: 8 hours (understanding the code, trying different values of η, and implementation of functions)
Task 2: 10 hours (implementation and solving errors, saving results for report).
Task 3: 20 hours (coding, solving mistakes, understanding them, tuning parameters, saving results for report).
Report: 15 hours (writing, giving appropriate format, final check).

REFERENCES

[1] https://docs.opencv.org/3.4.4/d7/d1b/group__imgproc__misc.html

[2] Diego Ortego. Juan C. SanMiguel, *Stationary foreground detection for video-surveillance based on foreground and motion history images*. IEEE International Conference on Advanced Video and Signal Based Surveillance, 2013

[3] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester and Deva Ramanan *Object Detection with Discriminatively Trained Part Based Models*