Applied Video Sequence Analysis: Object Detection and Classification

Mary Chris Go and Sebastián Andrés Cajas Ordóñez

I. INTRODUCTION

Video analysis requires complex methods for segmentation. One routine is the blob analysis which is the main focus of this laboratory report. All methods discussed are compiled in one framework and are implemented using C++ using OpenCV library. The datasets that were tested are assumed to be captured from a still camera. The methods are straight forward and sequential. It starts with dealing with foreground segmentation mask to extract blobs. The grass-fire algorithm is also done and the blobs are eliminated and classified as people, objects, or cars. Moreover, the last part includes stationary foreground detection that was implemented based on the paper "Stationary foreground detection for video-surveillance based on foreground and motion history images [1].

The main objective of the paper is to implement a foreground segmentation through a pipeline of step-by-step methods. It mainly discuss blob extraction, blob classifier, and stationary foreground extraction. All methods were evaluated qualitatively through visual observation.

This conclusion will be further proven and analyzed on the next sections. The paper is organized as follows: Section II, we describe all the methods tried. Section III expounds all the methods with how the functions are implemented. Section IV shows the dataset used to evaluate each method. Section V reports the analysis of the results and its comparison against each other. Section VI concludes the paper by summarizing all the findings. Lastly, Section VII shows the division of labor.

II. METHOD

There were three various approaches that were implemented. First is the blob extraction that is based on sequential Grass-Fire. Second task is the blob classification based on the aspect ratio feature and the simple statistical classifier. The last to implement is the stationary foreground extraction.

A. Blob Extraction

We based the blob extraction from Grassfire algorithm. This method deals with extracting the Binary Large Objects. These objects are obtained from the foreground masks using the mentioned strategy. From these blobs, we filter them. The ones who pass the threshold will be the probabilities for classification.

B. Blob Classification

Now the subsequent step is the classification task, which will be decomposed into blob classification and blob tracking, the first one characterized by the labels and the second, by the IDs of each detected object. A detected blob can be a person, object, person, or unknown. This method is based on aspect ratio and a simple Gaussian statistical classifier.

C. Stationary Foreground Extraction

The implementation of this method was wholly based on the paper "Stationary foreground detection for video-surveillance based on foreground and motion history images" [1]. First, background subtraction is applied. This first step is for detecting the foreground. We implement this through obtaining Foreground History Image (FHI) through updating the foreground temporal variation in terms of foreground and background detection.

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

$$FHI_t(x) = FHI_{t-1} + w^f_{neg} \cdot \sim FG_t(x) \tag{2}$$

The two weights are used to manage the contribution of background and foreground detections from the foreground mask. When the pixel is considered foreground, the foreground score increases. It will only decrease if the pixel belongs to the background model. We then normalized the foreground history image to the range [0,1] using the video frame rate (*fps*) and the stationary detection time. (t_{static}).

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

$$SFG_t(x) = \begin{cases} 1, & SFG_{t-1}(x) \geq \eta \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

These equations are used to detect the foreground objects even they become stationary.

III. IMPLEMENTATION

All implementations are in the same project. Varying parameters control the modes through a series of command line arguments.

A step-by-step instructions can be found through comments in the source files.

A. Functions for Blob Extraction

Object detection and classification presented on this work is based on foreground segmentation, which first will be focused on extracting the background using the threshold techniques we have already worked, or using OpenCV in-built functions, plus a post-processing stage that will be aimed at improving problems with the background extraction, such as noise due to low and high intensity movements,

illumination changes, camouflage and related problems with the camera's distortion.

Blobs com from Binary Large Objects, and means that we will focus on two aspects, first, determining the foreground masks, which will be performed by the Sequential Grass Fire algorithm, which will assign an ID for each detect object and surround it with a rectangle, which will be stored on a Rect type file, containing information related to its [x,y] position, as well as the width and height. Then this information is stored into one blob object, which will finally be pushed into a bloblist.

1) Sequential grass fire algorithm: This is a method for blob extraction. On it, we will count with two different images. The procedure starts with scanning the images from the top-left til the end of the image. We must initialize an ID counter, which will identify the objects that have the same value intensity (255 or 1). For this, We will go through all the images that have these values and give them to an auxiliary variable the ID counter value, to tag it with an ID. Secondly, we can assign the value we want the foremask to have. This procedure will be performed using the following functions and respective parameters:

```
cv::floodFill(output,Point(j,i),IDcount,&
    Selection,cv::Scalar(0),cv:: Scalar(0),
    connectivity);
```

So this function will take the image were we want to store the final mask containing the final output, in this case we have called it 'output', the top-left foreground pixelwise value from the first image through Point(j,i), the ID counter; Selection represents the rectangle object of type Rect, that will store all the information about '(x,y,width, height)'.

Once the blobs have been extracted, we pushed them back to the list of blobs which will store all blobs detected.

B. Function for small blobs suppression

This function will suppress small blobs, whose with and height is under an specified value. For performing this exclusion criteria, we extract the list of blobs and we verify that for each blob width and height, its correspondent value is higher than the specified threshold. Afterwards, the remaining blobs are pushed back to the blob list.

C. Functions for blob classification

The main function to implement here is the classifyBlobs. The function provides the mean and standard deviation models for each type of object. The function is able to classify objects into 5 categories: unknown, person, group, car, and object. It was based on aspect ratio feature and statistical classifier. The approach is applicable to person, car, and object since the aspect ratio models were already provided to eradicate the training process. Our goal is to use a simple statistical classifier (Gaussian) to identify each blob class based on the calculated distances between each blob feature and the reference standard deviation, iteratively, for each blob box from the bloblist. The expression applied can be seen below:

```
if ( (WED(feat_i, MEAN_PERSON, STD_PERSON)
< STD_PERSON) || (ED(feat_i, MEAN_PERSON)
< STD_PERSON ))
            blob.label  = CLASS(PERSON);
            bloblist[i] = blob;
```

Where WED and ED correspond to the Weighted Euclidean Distance and the Euclidean Distance, respectively.

D. Functions for stationary foreground extraction

There are five constants set on the first part of the implementations. These parameters are set to check the positivity and negativity of weights for history update. Alongside with these mentioned parameters, the fps and seconds were also set for the normalization version of the history mask. Last, a threshold value was set to determine the stationary mask.

For this part, the function extractStationaryFG was filled. This function has three arguments namely *fgmask*, *fgmask_history*, and *sfgmask*.The first argument is the analysed foreground frame. The second argument is simply the history of the foreground mask while the last argument is the stationary foreground mask.

From the formulas mentioned above, the foreground mask is stored and updated at each frame. With each update, the amount of time the pixel stayed as a foreground is stored. Therefore, the weight will either increase or decrease depending if it belongs to the background to the foreground. It will decrease if it is from the background and vice versa. For uniformity, the values are converted to float.

Furthermore, the value is then normalized. Through the set threshold, it will be determined where it belongs. If the stationary foreground is determined, it is treated as a simple foreground mask.

The trick in this method was to find the best learning parameter. After many trials, we learned that the smaller learning rate, the better time for the stationary object to not be part of the background through background segmentation method. A large amount of learning time such as 0.1 results the stationary object be removed from the stationary background

E. Running the code

Each method has its own source file and a makefile that makes the compilation easier. Each method has its own dataset provided so it should be noted the path folder is different for each case. The user should be able to provide it using the main.cpp file. One crucial parameter in running the code is the learning rate. The value should be between 0 and 1. Moreover, other parameters are already set per file but the user can tune them just in case it is needed. More details about the parameter can be seen in the implementation.

IV. DATA

There are three datasets used for this activity. Each method uses a specific dataset with numerous video sequences.

For blob extraction, ETRI sequence was used. It showed two people walking on a pavement. With the complex

background of various light conditions, the challenge comes from the tree shadows differentiating half of the frame.

The PETS2006 has three video sequences and is used for blob extraction and classification. It shows a hall where people walk through it. This has been used from the past exercises. It composed of many challenges like shadows, obstructions, and occlusions. Also, some people that passes by has objects such as suitcase with them which make the identification harder.

Last dataset is mainly for the classification and extraction of stationary foreground. It shows a parking lot where different cars appear in different frames. At one point, two cars become stationary and should be classified which is the main point of the method.

V. RESULTS AND ANALYSIS

Each task was evaluated on the appropriate dataset to comprehend the implementation of the algorithms. Different parameters were evaluated subjectively. There was no ground truth do test the robustness of the implementation objectively. All of the evaluation was done subjectively through visualization. All tasks were determined by adjusting the parameters until best results were obtained.

A. Blob Extraction

The parameters for minimum width and minimum height are defined manually based on empirical observations, testing different sizes from 20 pixels until 140 pixels. We could notice that the sizes of blobs characterizing people under different perspectives can change, in particular due to scale change and the camera perspective, making people to look smaller and be confused as a false positive.

Since this values were set manually, we expect that a more robust method for detecting people based on body-part detection may bring better and more accurate detections. As shown in Figure 1, a person was detected. The figure showed appropriate bounding boxes in most frames.

The result for this method highly depended on the segmentation approach. The better detection of foreground, the better extraction of blobs. Therefore, it may not be the best approach for scenarios where there are overlapping objects.

Fig. 1. Classification task for 1 person.

B. Blob Classification

The result for this activity highly depended on the quality of the extraction of blobs. As shown in Figures 2, 3, and 4, the detector was able to do its job by successfully classifying a person, a car, and an object.

The considered classification blob approach was based on the aspect ratio feature, which is the ratio between the width and the height for each blob. Then, we use the Gaussian classifier to compare whether the Weighted or Euclidean distances fit under the standard deviation for each model, based on the provided reference std. This method showed to be quite variable and a lot of testing is require to find a good classification. Therefore, there are some limitations when handling only shape and motion descriptors on a multi-class task, because during the video sequences there are many objects that are similar between each other in terms of feature ratio and since this descriptor is not considering body-parts, there are many false positives and thereof, the algorithm is highly pronable to missmatch when the camera perspective is in a harsh position or the object appearance is too variable.

On certain occasions, light variations on the soil are also confused as objects, as well as the shadows also enlarge the blob sizes, sometimes occluding other objects.

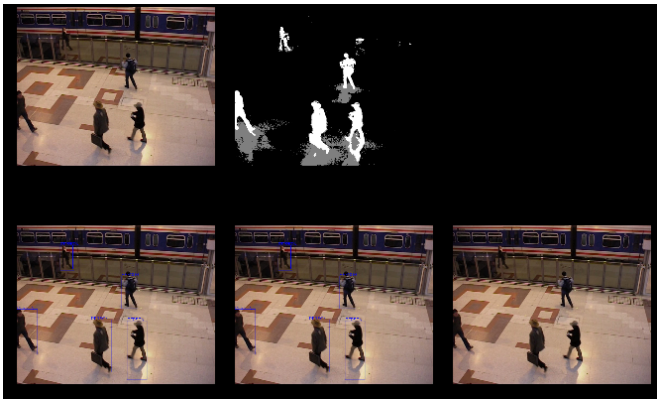

Fig. 2. Classification task for group of people.

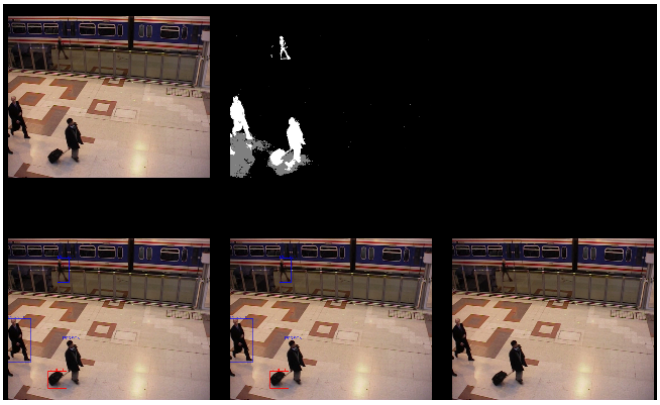

Fig. 3. Classification task for object detection.

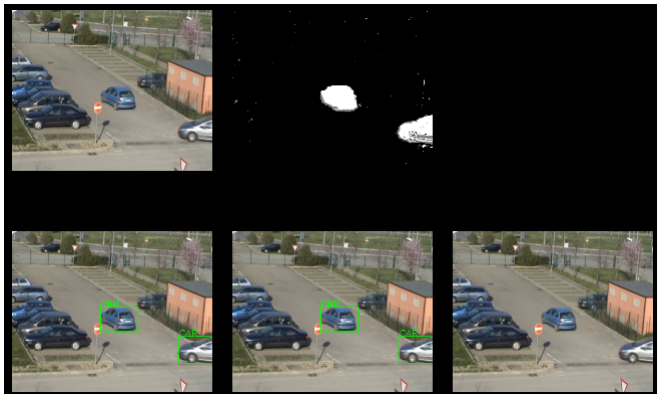

Fig. 4. Classification task for car detection.

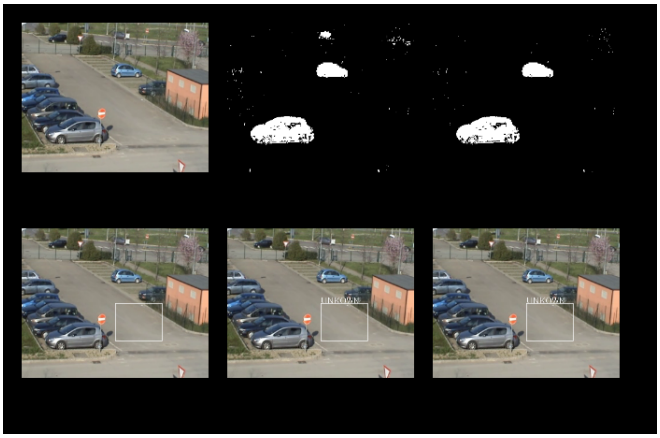

Fig. 5. These video frames are obtained from the implemented algorithm for extraction of stationary pixels.

C. Stationary Foreground Extraction

There are various parameters for this function. The *FPS* and *SECS_STATIONARY* are applied for the normalization. With the set parameters that can be seen in the source file, the stationary pixel were extracted. The weights also played a part in controlling the history update. *I_COST* was set for the positive value while the *D_COST* was set for the negative weight. This weights were assigned for determining the value of the background and foreground. Analyzing these weights, the ratio between them played the biggest factor. For example, the same value for the *I_COST* and *D_COST* means that the detected object has the same amount of time as foreground and background. To avoid this from happening, different values were set for both. As a result, the stationary car is immediately recognized after a few time of it stopping. The same thing happens for the second car the stopped after a few moments. Moreover, it goes back to the background once it changes its position. The last parameter mentioned is the *STAT_TH*. The set value is mainly for thresholding the history that was normalized.

The function was tested in all datasets. With a very low learning rate, more noise was present especially when there is lighting change as what we can observe from other datasets specifically in ETRI. Although with the main dataset, the objective was attained as seen in Fig 5. For a longer running time, it would be visible that the blobs in second frame will vanish. We also tried using a zero learning rate. This trial worked but it introduced a lot of false detection and noisy segmentation specially with lighting changes.

VI. CONCLUSIONS

All the methods were implemented well. Using the foreground segmentation, the blobs were correctly extracted. The methods were all simply written through a step-by-step process. Through this activity, we observe major changes by varying the parameters set slightly. For the first part, we have seen how the foundation of the algorithm was solely based on aspect ratio. This method could be more improve if more appearance features are incorporated and on a more acute object segmentation is performed during the blobs extraction. The most important parameter is the learning rate and it has demonstrated to be a key factor during foreground segmentation, having strong effects during the classification task. Observed from the activities, it is critical factor specifically for the last activity. All in all, the objectives were met. Moreover, more robust models can be look into more detail to improve the methods implemented for this activity.

VII. TIME LOG

Under this section is the detailed amount of time spent per step. All time logs include all methods such as setting up the environment, unforeseen events, and technical problems such as Eclipse and Visual Studio Code not working properly and virtual machine crashing. The mentioned time are considered per student as there is a collaboration for most parts.

- Task 1 = 6 hours, understanding the grass-fire algorithm and implementing several versions of it, as well as testing multiple ways to improve the blob extraction and filtering of smaller blobs.
- Task 2 = 6 hours, testing multiple combinations of multi-classification based on several object types, this included testing many types of performing the classification task, as the Gaussian classifier is not explicitly clear on the slides.
- Task 3= 3 hours: Writing of the algorithm, finding the right paramaters, adjusting learning rate
- Report Writing: 8 hours; analysis of the results and writing them through LaTeX

REFERENCES

[1] Ortego, D. Sanmiguel, J.C.. (2013). Stationary foreground detection for video-surveillance based on foreground and motion history images. 2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2013. 75-80. 10.1109/AVSS.2013.6636619.

[2] Google Drive Link (click for the directory)