Lab 2

Sebestyén Németh, Bendegúz H. Zováthi and Jaime Cortón González

I. INTRODUCTION

This laboratory report presents different segmentation techniques, written in C++ using the OpenCV library. On the development of the lab we were asked to develop a blob detection for object classification pipeline by using 'Grass-fire algorithm' among other techniques. In this case, not only the detection of objects was required, but also the undesired blob subtraction (static, noise...ect), the metric dependant classification of the blobs and the correct bounding box plotting.

II. METHOD DESCRIPTION

A. TASK 1 - Blob extraction

On this initial task we implemented a routine for blob extraction based on the sequential Grass-Fire algorithm. This implementation also had a size dependant blob removal.

B. TASK 2 - Blob classification

For this task we were asked to implement a blob classification routine based on the aspect ratio feature and the simple statistical classifier or Gaussian classifier. Labels for the 3 classes were: **PERSON**, **OBJECT** or **CAR**. Aspect ratio was computed by the following formula:

$$Blob_{AR} = \frac{Blob_{Width}}{Blob_{Height}} \tag{1}$$

For this method the following parameters were correspondent to each Class:

- **PERSON:** $\mu = 0.3950$, $\sigma = 0.1887$
- **CAR:** $\mu = 1.4736$, $\sigma = 0.2329$
- **OBJECT:** $\mu = 1.2111$, $\sigma = 0.4470$

C. TASK 3 - Stationary foreground extraction

For this task we implemented a stationary foreground detection and subtraction pipeline based on the paper *"Stationary foreground detection for video-surveillance based on foreground and motion history images"*[3]. This algorithm will work with a counter frame or **Foreground history image** that will be used for the foreground analysis and the posterior static object subtraction. This algorithm consisted on the computation of the Foreground History Image (FHI), the Motion History Image (MHI) and combine them to get the Stationary History Image (SHI). At this exercise our task was to implement the algorithm without considering the MHI and SHI variables in order to compute the Stationary Foreground Mask (FHG).

III. IMPLEMENTATION

A. TASK 1 Implementation

The Grass-Fire algorithm that we implemented, used the `floodFill` function with a `cv::Rect` structure for storing the bounding box of each detected blob.

First, we coded the `extractBlob()` function. An initial check of a valid connectivity was required (either 4 or 8).

On the foreground we have the following pixel values:

- Background = 0
- Foreground = 255
- Shadow = $\lfloor \frac{255}{2} \rfloor$

Having that clear, we initialized a double for loop over all the pixels, and everytime a foreground pixel was detected, we increased the `blobId` by 1 and we called the function `floodFill` over an already created aux matrix (`fgmask.convertTo(aux, CV_32SC1);`) for its update (blob neighbouring check and burning of the already checked pixels). After this, we just needed to update the initialized the blob with its Rect reference and then push back to the bloblist.

This function was applied to every frame, so we have to clear the bloblist every time for every frame for this task by using `bloblist.clear();` code line.

After this initial blob plotting we can see that there were some undesired blob detections, due to noise, image artifacts or illumination changes. To solve that we were asked to implement the `removeSmallBlobs()` function.

On this part, we defined a `MIN_WIDTH` and `MIN_HEIGHT` equal to 20 pixels, so that we can remove the small enough blobs by a simple if condition. We inputed the bloblist (`bloblist_in`) and we filtered out to another bloblist just the desired blobs (`bloblist_out`)

B. TASK 2 Implementation

On this task we used the aspect ratio of every blob and we check if that specific AR belonged to some of the 3 Classes (**PERSON**, **CAR** or **OBJECT**), and with which weight. The condition for belonging to a class, followed this equation:

$$|\mu - Blob_{AR}| \leq \sigma \tag{2}$$

If the given blob belonged to multiple classes, the target class was selected by minimizing the relative distance to the class' mean:

$$cls = argmin_i(\frac{|\mu_i - Blob_{AR}|}{\sigma^2}) \tag{3}$$

On the implementation of the `classifyBlobs()` function, we checked each blob's and AR and we chose the label

according to the formulas above. The only issue in here was to be aware of the indexing on the for loop to make sure that we were changing the labels on the original blob and not on a copy of the blob. With the **`for (cvBlob &blob : bloblist) { ... }`** syntax that problem should be solved (instead of the initial indexing method that copies the blob on a different variable.)

C. TASK 3 Implementation

The basic considerations in here are the FHI (Foreground History Image) and the SFM (Stationary Foreground Mask) computations, which were called **`fgmask_history`** and **`sfgmsk`** on the code.
The input parameters were the following:

- **FPS** = 25
  Frame rate, how the video was recorded
- **SECS_STATIONARY** = 1.5
  Waiting time to consider objects to be stationary
- **I_COST** = 1
  Increment cost for stationarity detection
- **D_COST** = 3
  Decrement cost for stationarity detection
- **STAT_TH** = 0.5
  Threshold to select stationary objects

The FHI was updated at each iteration in two ways. If the pixel is a foreground, then we add to the history image the foreground mask weighted with the **I_COST**. In case the pixel is a background (which is the inverse of the foreground mask), we subtract it from the history image weighted with the **D_COST** value, as you can see at eq.4. In order to avoid negative pixel values we applied a **`max()`** function, which gives 0 value to the updated FHI if it gets lower.

$$FHI \begin{cases} FHI_t = FHI_{t-1} + I\_COST \cdot FGmask_t \\ FHI_t = FHI_{t-1} - D\_COST \cdot BGmask_t \end{cases} \quad (4)$$

After computing the FHI we normalized its values in range [0:1] by dividing the FHI by **numframes4static** (if it is bigger than 1, we set the value to 1 by applying **`min()`** function). The **numframes4static** variable was initially computed as the multiplication of **FPS** and **SECS_STATIONARY**. This variable was designed for setting a value, how many frames should we wait to consider a pixel as stationary. In purpose to binarize the stationary foreground mask output, we compared the normalized pixel values with the threshold **STAT_TH** (eq.5).

$$SFG_t \begin{cases} 1 & \text{if } \overline{FHI} \geq \textbf{STAT\_TH}, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

IV. DATA

The dataset used this assignment was provided by the Universidad Autónoma de Madrid and Video Processing and Understanding Lab [1]. The dataset includes 6 *.avi* and 2 *.mpeg* RGB video files with 10-20 seconds duration, recorded with low frame rate (25 frames per second). The videos contain more complex sequences, where we can deal with problems like moving objects, reflections and shadows. The used datasets were namely:

1) *"ETRI/ETRI_od_A.avi"*
2) *"PETS2006/PETS2006_S1/PETS2006_S1_C3.mpeg"*
3) *"PETS2006/PETS2006_S4/PETS2006_S4_C3.avi"*
4) *"PETS2006/PETS2006_S5/PETS2006_S5_C3.mpeg"*
5) *"VISOR/visor_Video00.avi"*
6) *"VISOR/visor_Video01.avi"*
7) *"VISOR/visor_Video02.avi"*
8) *"VISOR/visor_Video03.avi"*

V. RESULTS AND ANALYSIS

A. TASK 1 Results

Fig. 1: Task 1 blob plot for frame 283

B. TASK 2 Results

Fig. 2: Task 2 blob plot for frame 231, with the correspondent classification

Nevertheless, the AR is ot a robust enough condition for the classification, perspective variations can affect performance as seen on the figure below:

Fig. 3: Task 2 error blob plot for frame 283

C. TASK 3 Results

Analyzing the results we can conclude that the algorithm preserves the stationary information in comparison with the segmentation masks. This simple implementation resulted good performance visually on the images but for further improvement using also the Motion History Image (MHI) information could be more efficient.

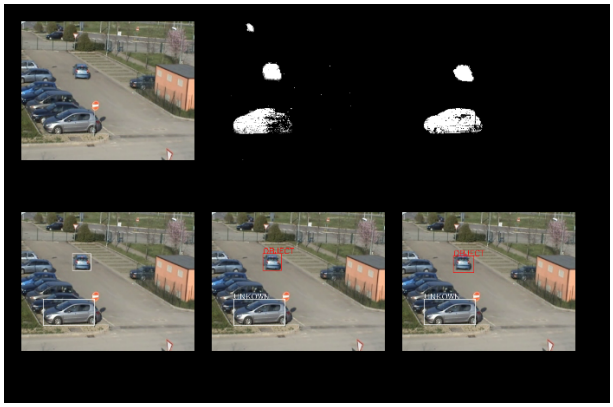

Fig. 4: Task 3 displayed results for frame 752 in VISOR/visor_Video00.avi video

VI. CONCLUSIONS

After developing this 3 tasks we can see that blob detection and plotting can give us really useful information about the frame that we are looking at. This kind of algorithms can be used in an automatized way for real-time object detection. The counterpart is that the performance results are highly dependent on the complexity of the scene. Scale differences between objets, perspective of the camera and object overlapping will determine the amount of errors appearing not only on detection but also on classification.

Specifically in Task 2, a classification based on the AR has a lot of limitations. On simple scenarios it can be sufficiently good but as the complexity of the scene increases this metric is not enough. On figure 2 we can see an example of an error due to the difference in scale and in the camera perspective.

Moreover, on Task 3, we observe that the Static foreground masks give us completely different results for different parameters or video sequences. We've been discussing that the method could be improved if this parameters auto-regulate themselves in an iterative way, so that they can adapt to scene changes. Again it is a method that works very well on controlled scenarios, but not that much in the case of multimodal complexity.

In the end, we've learned how to implement really efficient models that require very little computational cost, that work sufficiently good on certain type of scenarios. Complexity on the scene requires complexity on the models, but that is also affecting the computational cost of the model, that's why we should always evaluate that kind of trade-off so that we can deliver the more optimal algorithm in every case, because sometimes less is better.

VII. TIME LOG

We have worked on this project on a equally distributed time schedule of about 12 hours / person including the labs and the programming tasks.

Studying 2 hours / person, including the lectures, the individual research and the discussion about the main ideas of this topic.

Coding 5 hours.

Testing 2 hours.

Writing the report 3 hours.

REFERENCES

[1] Video Processing and Understanding Lab. http://www-vpu.ii.uam.es/webvpu/en/starting/who/http://www-vpu.ii.uam.es/webvpu/en/starting/who/

[2] Grass-Fire algorithm insights documentation https://posgrado.uam.es/pluginfile.php/1620377/mod_resource/content/

[3] *"Stationary foreground detection for video-surveillance based on foreground and motion history images"*, Ortego, D. and Sanmiguel, J.C., 2013 [75-80], 10.1109/AVSS.2013.6636619