# Object detection and classification

Jingwen Yang and Peter Szabo

## I. INTRODUCTION

In the report we present a foreground segmentation based blob extraction and blob classification method. The implementation of foreground extraction was not the scope of the report, in our work we highly focus on blob extraction and classification. For extraction we implemented the Grass-Fire algorithm from scratch. For classification we used the previously defined object classes. In our work we investigated the effect of different parameters, and exploited the strengths and weaknesses of our approach, and tried to prose solution to them.
We used Ubuntu and Eclipse with OpenCV as a tool to design and implement our solutions. Also our code used an initial skeleton for blob painting and image visualization.

## II. METHOD

The foreground based object detection and classification method consists of 3 main steps: foreground segmentation, blob extraction and blob classification. During the exercise our task was to implement blob extraction and blob classification. Foreground detection was the scope of the previous lab, so we won't detail them in our current work.

### A. Blob extraction

In our work, we performed blob extraction with the help of the Grass-Fire algorithm. The algorithm is based on connected component analysis, it examines a neighborhood of a selected pixel and looks for similar pixels. This process is repeated at each neighbor as long as no more left. The similar pictures are labeled, and removed from the frame. The algorithm finishes, when no more pixel of interest left. Finally small blobs are removed to filter out the blobs, whose width or height is below a threshold. This is necessary to get rid of noisy detections.
In our work we ignored the shadows, and only used the background for classification.

### B. Blob classification

To classify the blobs, a classifier needs to be chosen, which makes its decision based on one or multiple selected feature. In our work we used simple Gaussian classifier, which used aspect ratio as the only feature.
Aspect ratio is the ratio of the width and the height of the blob.
The principle of the classification is that we have the parameters (means and variance) of given objects. And during the classification we compute the distance between the unknown object and the model object, and make a decision based on it. Since the classifier was the Gaussian classifier we decided to examine if the unknown objects aspect ratio in inside the Gaussian or not. If its inside multiple Gaussian, we examine which one is the closest.
For the classification we had 3 pre-trained Gaussian models (with aspect ratio features): person, car and object. If none of them matched the object remained unknown.
Multiple distance metrics were provided, but we finally used the absolute difference of means, since it is a simple but efficient method for simple Gaussian models.

### C. Stationary Foreground extraction

This routine is based on the paper "Stationary foreground detection for video-surveillance based on foreground and motion history images", where we measure the foreground temporal variation to get a Foreground History Image FHIt(x).

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

$$FHI_t(x) = FHI_{t-1}(x) - w^f_{neg} * !FG_t(x)$$

where $!FHI_t(x)$ is the logical NOT operation; $w^f_{pos}$ and $w^f_{neg}$ are two weights to manage the contribution of the foreground and background detections. We should increase the $FHI_t(x)$ if they belong to foreground and reset it to 0 when they are background. This frequently happens in crowds where a static region is occluded by fast moving objects which cause camouflage errors. Hence, penalization weight $w^f_{neg}$ should decrease $FHI_t(x)$ at a higher rate than positive one $w^f_{pos}$ without resetting to 0 for increasing robustness against foreground errors.

After obtaining $FHI_t(x)$, we normalize it to the range [0, 1] considering the video framerate (FPS) and the stationary detection time (SECS_STATION) :

$$normalized\_FHI_t(x) = min\{1, FHI_t(x)/(FPS * SECS\_STATION)\}$$

Finally, stationary detection mask is obtained by thresholding. If $normalized\_FHI_t(x)$ is bigger than the threshold, then it is recognized as a foreground pixel, otherwise it is recognized as a background pixel.

## III. IMPLEMENTATION

Foreground segmentation was already implemented, using the opencv built in Background subtraction method.

### A. Blob extraction

For the Grass-Fire algorithm we used the opencv FloodFill algorithm. We used 8 connectivity analysis, on a binary mask, and gave each region a unique id starting from zero. In addition we also implemented the recursive flood fill

algorithm for 8 connectivity analysis from scratch. It return smallest possible rectangle that can be fitted on the object, its x and y coordinate and the width and height of it. Before moving to the next pixel the algorithm checks that we should not exceed the image, and also takes into consideration how much neighbor we want to visit.
Storing the blobs we utilize the cvBlob struct. It stores the rectangle, as x, y, the upper left coordinates, and width and height of the rectangular blob. Also it stores and unique id, among with class label, which is unknown so far.
To get rid of noisy detection, we remove the small blobs. We create and empty list, and only add blobs with a certain width and height.

### B. Blob classification

As we already mentioned, for classification we use aspect ratio. The unknown objects aspect ratio is examined, which Gaussian it belong to. It belong to a Gaussian if the absolute difference between its mean and the each models mean is smaller than the deviation times a ratio parameter. The latter parameter is the only parameter of the model. Finally if the object is inside a Gaussian we take a look if it's inside other Gaussians as well, and then we look for the closest Gaussian in the end.

### C. Stationary Foreground extraction

In the implementation we tried out both matrix-level operations and pixel-by-pixel operations. We do either increase or decrease the scores in history foreground mask according to the pixel is a foreground pixel or a background pixel. We set the parameters as follows:
$FPS = 25$,
$SECS\_STATIONARY = 5$,
$increasingWeight(I\_COST) = 1$,
$decreasingWeight(D\_COST) = 4$,
$threshold = 0.5$.
$learningrate = 0.0001$

As we initialize the history mask to 0s, we should be careful with the values of the mask, don't let it be lower than 0. Then we just compare the normalized value of history mask with the threshold and decide if it is a stationary foreground pixel or not. Save the value in stationary foreground mask.

### D. Running the code

The program runs without parameter. The number of connectivity can be modified in the 151. row of Lab2.0AVSA2020.cpp. It is going to use our implementation of flood fill, which can work with 4 or 8 neighbor connectivity.

## IV. DATA

In this lab we have basically three sets of videos to see how the algorithms work. The first video is two man walking on the path in the park. Both of them didn't stop and kept walking.

The second set is in the train station. There are many people carrying big suitcases, carts walking around, which increases the difficulty of the algorithms. Also the problem of interaction such as people walking in groups or passing by each other makes it more complex for choosing the right person model. Some people are running.

The third set of videos are filmed by the monitor in the parking lot, where the cars move around, stop in the middle of the path, or stop in the right parking position.

## V. RESULTS AND ANALYSIS

### A. Object classification with background subtraction

In our work we tested the accuracy of our method in 3 provided sequences: ETRI, PETS2006, VISOR.

Our results indicated the our approach accuracy highly depended on the set of parameters. Our goal was to find the best parameters combination which is the most robust among various conditions.
We experimented with 8 connectivity analysis. This approach results less, bigger object compared to the 4 connectivity analysis. As we can see in most of the cases it behaved accurately, but in case of close object it resulted false merge of multiple object. And example is visualized on Figure 1. As we can see, the 4 connectivity analysis separated the group of persons much better, but it introduces a lot of noisy detection, since it will less likely to overcome the small gaps between the fragment of objects, compared to the 8 connectivity.

(a) 8 neighbor connectivity (b) 4 neighbor connectivity

**Fig. 1:** Effect of neighborhood connectivity

Moreover we could observe, that the accuracy of the classifier highly depends on the accuracy of the foreground segmentation algorithm. We encountered multiple scenarios, when the classifier worked incorrectly due to the error of the background subtraction.
In case the background method was too strict, we observed that multiple boxes were created, and they resulted badly classified objects. A good example for that is can be seen on the left picture of Figure 2. The threshold may be too strict, and the head was classified as a separate object. Also this effect could be caused by camouflage. These are inherent challenges of background subtraction.
Also the disadvantage of the approach was that the stationary object could not be classified. As the objects stop moving the algorithm stops recognizing them. If we want to recognize them, we should apply different approach before classification. And example can be seen on the right (b) image of Figure 2.

(a) Same object into smaller pieces

(b) Not detected stationary object

**Fig. 2:** Dependency of foreground segmentation algorithms accuracy

During the blob extraction we filter out the smaller blobs. But how do we define the smaller blobs? This is a very challenging task to define it in a robust way. During our experiments, we observed, that if this value was strict, multiple smaller objects remained undetected. On the other hand if the threshold was small, a lot of false detection appeared. This can be observed in the Figure 3. As we can see on the left figure there are some false detection, in this scenario the threshold should be higher. In contrary, the same threshold seems too strict on the right scenario, when the cars in the background are not detected at all.

(a) False detection

(b) Lack of detection at the background

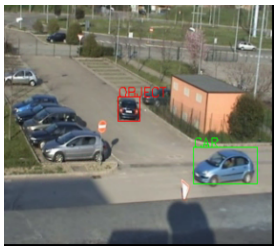
**Fig. 3:** Min width and height should be at least 10

The parameter of the classification was, that how do we define when is an object inside a Gaussian. It was determined by tau parameter, which measured that how big the absolute difference between the mean and the objects aspect ratio can be.

The parameter was a requirement, that the absolute difference of the means should be smaller than n times the variance. And it determined, how flexible we are with the boxes, how robust our algorithm is in different scenarios and different shape variations. The shape variations pose a hard challenge. Since we used only one parameter in the classification process, it was really hard to set a parameter, which was robust to pose changes. As we can see on Figure 4, a car can have multiple shapes, based on the direction it goes. A person can take multiple shapes based on age, appearance, or he/she can carry objects.

Since we only have 3 object types, we were allowing with this parameter. It resulted that one object could belong to multiple Gaussians, but afterwards the closest center of the Gaussian determined the final object type.

But being less strict with the Gaussians could result multiple possible false classification. But we observed, that in our data sequence the presence of other kind of object was negligible, only a few and well defined object was present. Regardless, we think it is worth to point out that in more scenarios it has to be taken into consideration that an allowing threshold could potentially classify unknown objects, for which there is no model defined.



(a) Parking car


(b) Further car from the back

(c) Person with another object

(d) Different age, and way of movement

**Fig. 4:** Variability of different objects

Partial occlusion could leads some false results. When a person stands behind something it could result a different height or width and a false aspect ration. Also multiple people can occulude each other. A solution to that could be to introduce new object type like GROUP of people. Moreover more complex models, based on more features could improve accuracy as well. We should also keep in mind, that intraclass variance could also lead some miss classification. It means that for example just between type of cars there are also big differences.

### *B. Stationary Blob Extraction and classification*

In this task we tested the accuracy of our method in only one set of sequences because this one can testify stationary foreground mask the best: VISOR.

In stationary foreground mask, only the stationary objects appear but not the moving objects. In the picture 5, the moving car only appears on the foreground mask but doesn't appear on the stationary foreground mask. We know in the Mixture of Gaussian background model the learning rate decides how fast the stationary object integrate in background, so if we set the learning rate as 0.0001, the stationary object will stay on the foreground mask for a while. But on the stationary foreground mask it should definitely stay longer, depending the paramaters $FPS = 25$ and $SECS\_STATIONARY = 5$ we set before. In the picture 6, we can see the blue car stopping at the crossing

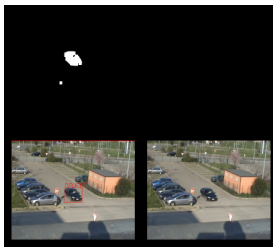


**Fig. 5:** moving object

**Fig. 6:** stationary object

is fading faster on foreground mask, slower on stationary foreground mask.

Here we have a problem of false detection, which is the shadow(illuminance change) detected as foreground pixels, and of course since it doesn't move, also detected as stationary foreground pixels.

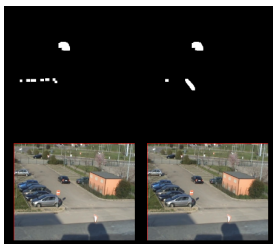


**Fig. 7:** shadows

## VI. CONCLUSIONS

To conclude we can say that the models are relatively simple, and works efficiently with the good parameter setup. It is also fast. The third task presented an approach for stationary foreground region detection by computing spatio-temporal variations of foreground data extracted from the video sequence. We encountered multiple weaknesses of the approach as well. Finding the correct parameters is very challenging especially in complex scenarios with possible factors, like occlusion, multiple type of objects. Also accurately defining a model is really difficult. We experimented the effect of given parameters in different scenarios, and concluded that fore a given task they could be set easier and more accurately. On the other hand making it more accurate results a worse overall performance, this is a trade-off we should be aware. It also highly relies on the accuracy of the foreground segmentation method. A further improvement could be the use of more features, and a good way of combining them, also the use of complex models for foreground detection, automatic tuning of algorithm parameters, and probably the use of region-level information. But we should keep in mind, that learning a classifier with more features is difficult, and makes the whole algorithm slower.

## VII. TIME LOG

**Blob extraction:** 7 hours. 6 hours initially understanding the code, implementing the Grass-Fire, and finally as the optional task implementing the flood fill algorithm. 1 hour of experimenting with the parameters.
**Blob classification:**
8 hours implementation of code and experimenting with the effect of different parameters **Stationary foreground extraction:** 8 hours implementation of code and playing around the configuration of the parameters
**Writing the report:** 10 hours, writing the report itself, collecting the data and the images, evaluation and explanation