Foreground Segmentation

Jingwen Yang and Peter Szabo

I. INTRODUCTION

In our current work we implemented multiple foreground segmentation algorithm. Starting from the background subtraction algorithm towards more complex models, called single Gaussian and Mixture of Gaussian. We aimed to avoid multiple ghost type of errors, so we implemented different background subtraction algorithms. To reduce error shadow removal was implemented. Finally we experimented the effect of different parameter setup, and detailed our experiences in the Exultation section.

II. METHOD

A. Foreground Segmentation Algorithm

1) Basic algorithm: The algorithm measures the absolute difference between the initial (in our case the first) frame, called background, and the current frame. If it is bigger than a given threshold, it is going to be considered a foreground pixel.

2) Blind and selective running average: The basic algorithm performed in several cases poorly, since it could not adapt the changes in the background. We introduced an updating background model, in order to adapt the dynamic changes and to suppress the ghost errors and stationary objects.

During blind update, we dynamically updated the current background by weighted combination of the previous background and current frame. The ratio of how much current frame we want include, was called alpha parameter, and it was responsible for the adaptation speed.

The selective running average follows the same principle, the only difference is, that the update only happens on background pixels.

3) Suppression of ghosts (2): Ghost (2) is the error, when a stationary object, which used to belong the background disappears from the image. In order to overcome this error a counter is introduced for every pixel. It gets incremented every time a pixel is detected as a foreground, and is zeroed once it is turns into background.

Once the pixel reaches a certain threshold, it will be turned to background. It means that a pixel did not changed for a long time, and it could be possible a background pixel.

B. Shadow detection

This method is responsible for shadow removal. Shadows can appear as false positive results for foreground, and highly decrease our accuracy, since they move in the same manner as the targeted foreground.

In order to detect, and remove the shadow, we use a chromaticity -based approach. The main idea is, that when shadow is reflected on an object, the pixel is not going to take a completely new color, but it is only going to change its intensity value. Also the reflection is going to behave the similar way, but with higher intensity, but it was not the scope of our work.

Our method, inspected several factors in the HSV color space. If the intensity ration of the current frame and the background is between a certain region, and saturation difference is moderate (smaller than a threshold), and the hue difference, or its inverse is small enough we classify a picture as shadow.

C. Single Gaussian Model

In order to operate more accurately in a wider range of scenarios, a complex background model seems a good approach. We define a Gaussian for each pixel.

The algorithm has two steps, detection and update. During update we update the Gaussian parameters based on the new images, and during detection we make a decision, based on the Gaussian, if the pixel is foreground or background. It is background if it inside the Gaussian, inside a certain band of mean, otherwise it is foreground.

D. Mixture of Gaussian

The following method is the extension of the previous one. In this case, for each pixel we assign 3 Gaussians, each having 3 parameter, the mean, variance and weight.

This method has also 2 crucial step, detection and update. In every iteration, we take one pixel, and check which Gaussian does the new pixel belong to. If none, we delete the one with the lowest weight and assign it to the current one. If it belongs to one we update the parameters of that Gaussian, based on the pixel. The weight is also updated, increased if the pixels belong to one, and decreased for the rest.

The decision about being foreground or background is based on the weight. The combination of the first Gaussians with the highest weigh is going to decide if we are foreground or background. If the first weight exceeds the threshold itself, it is the only one being background, otherwise we combine with the following Gaussians with the highest weight.

III. IMPLEMENTATION

A. Functions for Task 1

The following task was implemented to gray level images. The background subtraction is implemented in matrix operations. In the constructor the alpha parameter and the decision between the type of background update is added. While blind

update, combined with the ghost (2) removal is implemented with matrix operations, the selective update is implemented pixelwise.

We experienced with matrix-wise execution, but it strongly reduces the background image resolution resulting further error. The matrix-wise implementation can be found in comments. Also the explanation the logic of the code, is also provided in the comments.

B. Functions for Task 3 - Shadow detection

In this task the background subtraction and shadow removal algorithm were implemented and used, on RGB images.

C. Functions for Task 4 - Single Gaussian

The methods are written for gray scale images, and during the implementation matrix operations were used. The parameters were added in the constructor, the $alpha\_gaussian$ parameter is responsible for the Gaussian parameter update speed. The initial variance is the variance of the first frame, and the mean is the mean of the first frame. The update is happening blindly. We also experimented with selective update, they can be found in the code, but since it resulted worse result we used the blind approach.

D. Functions for Task 5 - Mixture of Gaussian

The methods are written for gray scale images, and blind update. At first we tried to work it in matrix scale. The parameters alpha_Gaussian, weight_Threshold and number of gaussians used were added in the constructor, the $alpha\_gaussian$ parameter is responsible for the Gaussian parameter update speed, and the $weight\_threshold$ is going to be used for deciding how many gaussians are describing each pixel. The initialization of the first gaussian model is calculated from the mean and variance of the first frame, while the other gaussian models will be assigned random values for the mean mask and the variance mask. As for the weights, the first gaussian will be assigned one and the others will be initialized as zero. We update the background pixels based on the gaussians selected by weights, we update the means, variances and weights of gaussians each frame.

E. How to run the program:

Each program can be run by itself. They require the dataset for running (they maybe differ at different parts). The datasets can be found on the link below: https://drive.google.com/drive/folders/15b$_p$2$YlQNX6aR09eLFqFo0Hfu$

IV. DATA

We have two datasets to run our foreground segmentation algorithms.Both provide a realistic, camera-captured (no CGI), diverse set of videos. They have been selected to cover a wide range of detection challenges and are representative of typical indoor and outdoor visual data captured today in surveillance, smart environment, and video database scenarios.

The video dataset is for testing and visualizing how the algorithms work, which includes the following challenges: illuminance change, hot start(dynamic background), shadows, intermittent object motion.

The sequence dataset is used for evaluation our code. The four sequences are filmed from different angles, which can efficiently evaluate the robustness of the algorithms. It is accompanied by accurate ground-truth segmentation and annotation of change/motion areas for each video frame.

TABLE I: Videos for Testing Purposes

Name of the Video	Number of Frames	Challenges
hall.avi	299	Appearance of stationary object
empty office.avi	620	Global illuminance change
stationary objects.avi	539	Stationary object location change
eps hotstart.avi	299	Background image has moving object
eps shadows.avi	668	Shadow removal

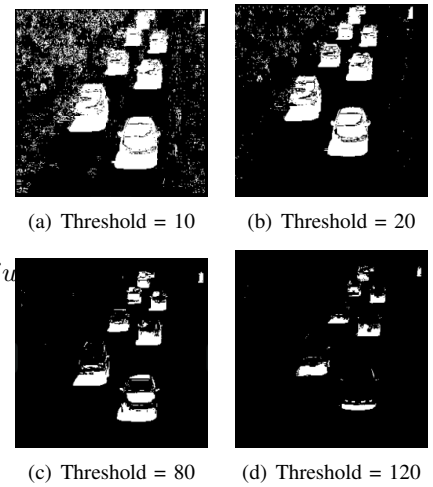
TABLE II: Sequences for Evaluation Purpose

Name of the Sequence	Number of Images
Highway	1700
Office	2050
Pedestrians	1099
PETS2006	1200

V. RESULTS AND ANALYSIS

A. Background subtraction

In the first task the only parameter was, the threshold. The higher it was, the less difference did we accept between the current pixel and the background. By increasing the parameter, we exclude more pixel, or by decreasing the value, it is the other way around. The effect of the different threshold value can be observed on Figure 5. During our future work, we chose the parameter as 20, since it seemed the most robust among a wide variety of environment.



(a) Threshold = 10 (b) Threshold = 20

(c) Threshold = 80 (d) Threshold = 120

Fig. 1: Parameters of background subtraction, at frame 821 from Highway data set

On Figure 1, we can observe the inherent error of static background model, when the initial image contains foreground information.

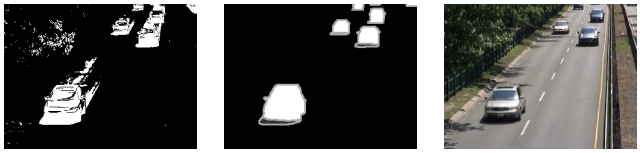
On the upper right corner we can see, that at the initial pixel there was not only background but a car, which was not supposed to be background, and since our model is stationary, it is not going to disappear from it.

(a) Result of blind update (b) Groundtruth (c) Original image

Fig. 2: Effect of α parameter in blind update compared to ground truth on frame 1231 at Office data set

B. Background model update

During background model update we implemented blind a selective update and experimented with different parameters of Alpha. Alpha was responsible of how fast our background adapts to the foreground. We started with a relatively high alpha parameter (0.4). We observed that the object were accurately detected. But when they were moving slower or stopped for a short time, they rapidly disappeared. This can be observed in the case of the office example on Figure 2.



(a) Result of blind update (b) Groundtruth (c) Original image

Fig. 3: Slow background update on frame 1004 at Highway data set

We lowered the alpha parameter (0.04). The adaptation slowed down, but the slower moving objects were detected better, and they remained foreground for a longer time when they stopped moving. But the faster moving object resulted a worse accuracy. A new phenomenon emerged, the object were leaving trails behind them as Figure 3. The reason behind that, is if an object is moving fast, with lower alpha the background needs more time to adapt, as long as it does so, the old place of the moving object will be considered as foreground. Moreover, implementing a counter is very useful in case we have a very low adaptation speed. As Figure 4 shows, even with low alpha, the background adapts more decently if we have a counter.

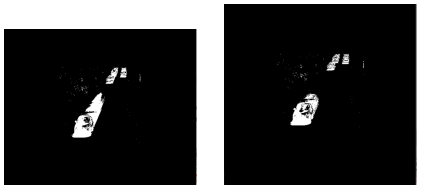
The disappearing of object is also benefitable, since it overcome the ghost (2) type of error. This can be achieved by extending the model update with the counter. This is responsible for removing the object that remain foreground for too long. As we can see the initial car from the Figure 1, upper right corner disappears over time. The choice of counter was set to ten, after multiple experiment. It proved to be slow enough, not to delete slow foreground object, but small enough detect inappropriate foreground objects over time.

Method	Recall	Spec	FPR	FNR	Prec	FMeas
FgSegNet	0.9977	0.9999	0.0001	0.0023	0.9978	0.9978
Param1	0.5910	0.9881	0.0118	0.4089	0.6157	0.5514
Param2	0.6652	0.9741	0.0258	0.3347	0.4405	0.4928
Param3	0.4089	0.9944	0.0055	0.591	0.6655	0.4509
Param4	0.4894	0.9965	0.0035	0.5106	0.8419	0.6085

TABLE III: Comparison of our results with other methods, Param1: $\alpha = 0.04$ with counter, Param2: $\alpha = 0.04$ without counter, Param3: $\alpha = 0.4$ without counter, Param4: Simplified Self-Organized Background Subtraction

By choosing the parameters we had to take into account the different environment, even though in the Highway example a faster adaptation seems like a better choice, on other datasets, where there are slower object (like persons), a high alpha may decrease accuracy.

Our results indicate that the current method is robust against the slight change of background, by progressively adapting the background model. The prefect example for this case, if when the room gets lighter, but by adapting our model, we will not encounter any false detection.



(a) Background without counter (b) With counter and its threshold at 10

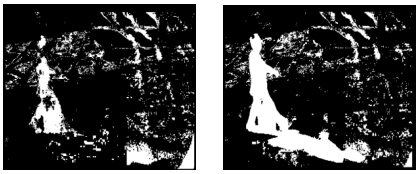
Fig. 4: Difference by extending a counter wit low α = 0.04

In Table 1, we can see the comparison of our results with different algorithm. As we can see, our method is still far from the best ranking method, called FgSegNet, but on the other hand it performs better than the worse one on the webpage on baseline, the Simplified Self Organized Background Subtraction. We can clearly see the tradeoff between parameters. In the Param2 composition we have relatively decent FNR, but on the other hand lot more False Positive detentions. By increasing alpha to a higher value, it is visible, that FPR rate drops, but we will have an increased number of FNR. This confirms our observations.
It is worth to notice, that by including the counter, we can observe higher precision, without decreasing the recall. The FMeasure is the highest with the counter.

C. Shadow removal

In the case of shadow removal, we have 4 parameters. Alpha and Beta responsible for the intensity band filter, and there is a threshold for saturation and hue difference.
First of of the we calibrated the alpha and beta value.

The alpha was the lower bound for the background, since it seemed to be the bottleneck in most of the cases. We experimented with multiple values. When it was higher we removed less background.



(a) $\alpha = 0.01$ (b) $\alpha = 0.5$

Fig. 5: Result of different alpha parameter with $\beta = 0.9$ Saturation and Hue threshold = 80

With 0.5 we could see, that I scenarios, where the intensity difference between the shadow and the person was small, we performed better. On the other hand if alpha was 0.01, when the contrast was higher, it resulted the better accuracy, but when the intensity differences were slower, it filtered out part of the foreground.

The results are shown on the Table 4.

Although, we can see, that more shadow got removed, the in other environments it decreased the accuracy in a way that we rather kept alpha at 0.5.

One more thing to notice that our algorithm is good condition is very accurate at removing moving object's shadows. In the case of static background, its is adapting very slowly.

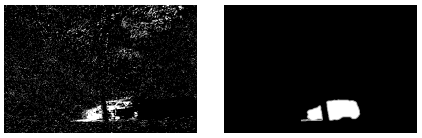
	Rec	Spec	FPR	FNR	PBC	Precision	FMes
$\alpha = 0.5$	0.8153	0.92349	0.07650	0.18465	8.16346	0.40213	0.49669
$\alpha = 0.01$	0.45771	0.93694	0.06305	0.54228	8.53843	0.37620	0.34630

TABLE IV: Results of shadow removal with $\beta = 0.9 HueTh = 80 SaturationTH = 80$

Our results show, we achieved a decent recall, but the precision with its 40% is not very strong. It can be the result of different conditions. We had a lot of outdoor videos, where the contract were not different. Moreover the static background model could result false results. For example during our frames, the brightness changed, which resulted false detection in the foreground, and this reduced the accuracy of shadow separation. In the future work it may be worth to extend the background update in RGB field, to enhance accuracy.

D. Robust background models

1) Gaussian model: In this task there is a new parameter $alpha_g aussian$. The higher it is, the more we update the background pixel. We initialize it as 0.02. In the real case, we tried several values for it, they didn't show much difference. As the picture below shows, there is much noise in our result. Also, the shape of the object is not well segmented. According the published results on the changeDetection website, most of them used alpha = 0.001 or 0.002, which is way smaller than our alpha. Maybe it is better to use a smaller tuning value.



(a) $SingleGaussian$ (b) $GroundTruth$

Fig. 6: Result of Single Gaussian Model with $alpha_g aussian = 0.02$

Rec	Spec	FPR	FNR	PBC	Precision	FMes
0.2356345049	0.9526940252	0.0473059748	0.7643654951	5.7186391906	0.0452565754	0.0654031919
Overall	1.1184712797					

TABLE V: Results of single gaussian with $\alpha = 0.02$

The figures above doesn't look really good, obviously the Gaussian Model we coded has more space to improve.

2) Mixture of Gaussian: The mixture of Gaussain model takes forever to run because we are updating k gaussians pixel by pixel and the algorithm is fulfilled with multiple conditions. Therefore we couldn't make it to finish running on the testing dataset and evaluation dataset.

VI. CONCLUSIONS

A. Blind and selective update

We encountered a lot of difficulties during the calibration of parameters. In case of alpha parameter, if it is high, objects will be detected more accurately as long as they move. But when they stop, they will be turned into background rapidly. In contrary, as the alpha parameter gets lower, they will be turned into background slower, but the "trail" appears.

This shows the importance of setting the alpha parameter correctly, and indicated that in different scenarios, different alpha may be optimal.

It can be concluded, that even though we are using a relatively simple algorithm, it still shows acceptable results in many different scenarios. By extending it with additional error removal method even better results can be achieved.

B. Shadow removal

The calibration of the parameter was a very difficult task. During evaluation we met with a lot of different scenarios. Based on our observations we can draw the conclusion, that finding the ideal parameter for each scenario is a complex and challenging task. On the other hand if we calibrate our algorithm to certain scenarios we could achieve way higher accuracy on that certain case. Creating the ideal parameters in outdoor environment is still a difficult task due to the constant change in brightness and shadow intensity.

It also depends on out application, what is more important to us, we can make sure to get ride of most all of the shadow, but that would result less accurate foreground model.

C. Single Gaussian

The idea of single gaussian model should overcome all the problems solved by previous tasks like shadow removel, progressive update and so on. Since we are running it on motion datesets, the parameters should be tuned based on gradual and sudden change of the background.

D. Mixture of Gaussian

Challenges such as shadowing, occlusions and illumination changes ideally should be overcomed by mixture of gaussian model. However, doing mixture of Gaussian models in pixel level is not a good idea. Algorithms should definitely be polished and refined to matrix level. But due to the time limit and ability limit, we didn't make it work perfectly.

VII. TIME LOG

The required time:
Task1: 20 hours of implementing it (understandig C++, learning OpenCV pixel and matrix operation, implementing algorithm, debugging the code, rewriting the code)
Task2: 4 hours (downloading dataset, creating own results, debug issues)
Task3: 14 hours
Task4: 12 hours
Task5: 20 hours
Writing the report: 20 hours (writing the report, create data for evaluation, run the code, drawing the conclusion).