Foreground Segmentation

Mohammad Mohaiminul Islam and Zahid Hassan Tushar

I. Introduction

This report is generated as a guideline and analysis for the system designed for *Foreground Segmentation*, written in C++ using OpenCV Library. The objective of this work is to detect moving objects on a stationary and dynamic background. At the beginning a basic method was created with blind and selective update for the gray scale images, which was later remodeled to support color images. Right after that a routine was designed to filter out stationary objects that were appearing and disappearing in the frame. In the next stage of development, a shadow removal technique was introduced to mitigate the effect of shadows in the foreground mask. Further, advanced background subtraction algorithms were explored in search of improvement for the selective update models designed at the beginning as well as to accommodate segregation of background and foreground in dynamic background situations.

II. Method

Our foreground segmentation project can be divided into two subtasks: Background subtraction model and Shadow removal. The first one is to detect moving objects and assign them to the foreground while the other task focuses on separating the shadows of moving objects lurking in the foreground. We started our module by creating a background model based on frame difference. Then we implemented blind and selective updates of the background model using a learning rate. Finally a counter was integrated to suppress the stationary objects that appear or disappear. Alternatively two different approaches were investigated using single and multiple Gaussian models for background detection. Simultaneously we explored the HSV color space to leverage the chromaticity of individual frames to remove shadow.

A. Background Subtraction Models

1) Simple Frame Difference Model. As the name implies we took the frame difference between the background model and the current frame, and separated the significant dissimilar pixels which later to be marked as foreground pixels. (1) shows the algebraic expression.

$$|Image_t[x,y] - BKG_t[x,y]| > \tau \;\Rightarrow\; Fore_t[x,y] = 1 \qquad (1)$$

2) Selective Running Average Model. Two types of update algorithms were carried out to update the background model. In the blind update algorithm we took a percentage of the current frame pixels and incorporated it with the background. However, in the selective algorithm we only considered the current frame pixels that do not belong to the foreground, took a percentage of those pixels and updated the background model. (2) represents the selective algorithm where α determines the adoption rate.

$$BKG_{t+1}[x,y] = \alpha Image_t[x,y] + (1-\alpha)BKG_t[x,y] \;\Leftrightarrow\; Fore_t[x,y] = 0 \qquad (2)$$

3) Advanced Gaussian Based Background Subtraction Model. C.R. Wren et al. proposed a unimodal model [1] which was implemented to update the background. In this method, a Gaussian distribution was considered for each pixel location in the frame. If the new pixel belongs to the distribution, it is a background pixel and vice versa. Also for the background pixels, the mean and standard deviation of the Gaussians were updated per frame using selective update algorithm. We can mathematically put this method as shown in (3) and (4).

$$\mu_{t+1}[x,y] = \alpha Image_t[x,y] + (1-\alpha)\mu_t[x,y] \;\Leftrightarrow\; Fore_t[x,y] = 0 \qquad (3)$$

$$\sigma^2_{t+1}[x,y] = \alpha(Image_t[x,y] - \mu_t[x,y])^2 + (1-\alpha)\sigma^2_t[x,y] \;\Leftrightarrow\; Fore_t[x,y] = 0 \qquad (4)$$

C. Stauffer et al. investigated a multimodal background model based on Gaussian distribution [2]. This adaptive background mixture model uses multiple Gaussian distributions per pixel. In our case, we implemented five Gaussians per pixel. A normalized weight matrix was also incorporated to order the Gaussian models. During each frame, the models were sorted and their weights were added to meet an upper weight threshold (0.8 in our case). When a pixel belongs to the Gaussians of the background model, it is included in the background mask. However, when a pixel is found outside the available Gaussian models, it is placed into the foreground mask and a new Gaussian model is placed around that pixel while disregarding the lowest weight Gaussian. For each frame the weight, mean and standard deviation of the Gaussians were selectively updated using (5), (6), (7) and (8).

$$\mu_{t+1}[x,y] = \alpha Image_t[x,y] + (1-\alpha)\mu_t[x,y] \;\Leftrightarrow\; Fore_t[x,y] = 0 \qquad (5)$$

$$\sigma^2_{t+1}[x,y] = \alpha(Image_t[x,y] - \mu_t[x,y])^2 + (1-\alpha)\sigma^2_t[x,y] \;\Leftrightarrow\; Fore_t[x,y] = 0 \qquad (6)$$

$$W_{k,t} = (1-\alpha)W_{k,t} + \alpha M_{k,t} \qquad (7)$$

$$M_{k,t} = {}^{1}/_{0}\ (matched/not\ matched) \qquad (8)$$

4) Background Subtraction with Shadow Removal. While the above mentioned algorithms could easily detect the moving objects in the frame, they also incorporated the shadow of the objects in the foreground mask. To suppress these shadows, we exploited the chromaticity based shadow detection described in [3]. Since it is quite impossible to distinguish the chromaticity in the RGB color space, we shifted to the HSV color space which exhibits our desired property. In the algorithm, we used threshold in all the color channels to detect shadows. The (9), (10), (11) and (12) were employed for this method.

$$Image_t[x,y] = (IH_t[x,y], IS_t[x,y], IV_t[x,y]) \quad (9)$$

$$BKG_t[x,y] = (BH_t[x,y], BS_t[x,y], BV_t[x,y]) \quad (10)$$

$$\alpha \leq \frac{IV_t[x,y]}{BV_t[x,y]} \leq \beta \wedge |IS_t[x,y] - BS_t[x,y]| \leq \tau_S \wedge D_H \leq \tau_H \Rightarrow Shadow_t[x,y] = 1 \quad (11)$$

$$D_H = min\begin{pmatrix} |IH_t[x,y] - BH_t[x,y]|, \\ 360 - |IH_t[x,y] - BH_t[x,y]| \end{pmatrix} \quad (12)$$

III. IMPLEMENTATION

This project has been mainly written on C++ with the OpenCV Library package. Following are the system specification for the development of this project

- **OS:** Ubuntu 18.04.4
- **IDE:** Eclipse
- **Language:** C++
- **Library:** OpenCV

A framework file was provided to us which included 5 files: fgseg.cpp,fgseg.hpp, Lab1.0AVSA2020.cpp,ShowManyImages.cpp and ShowManyImages.hpp. We mainly worked with the fgseg.cpp, fgseg.hpp, Lab1.0AVSA2020.cpp files and kept unmodified ShowManyImages.cpp & ShowManyImages.hpp since their only functions was to display results in an external window. All of our methods were implemented in the fgseg.cpp class under different eclipse projects and fgseg.hpp contains the class definition of fgseg.cpp.

To use the program, one must specify the dataset path first along with the category. Then total number of category sequences and their names need to be mentioned. There are several parameters for different versions of the program. In the Foreground Detection task, only an optimal threshold value has to be set while in the Background Model Update task two parameters: learning rate & update type (selective/blind) need to be specified alongside the threshold. To work with the Ghost Removal task, a counter threshold is required. Those 3 tasks were built incrementally and finally combined together as Background Subtraction Model. Therefore, all of these parameters are needed to run this version of the program. Parametric background update model includes two approaches: Single Gaussian Model (SGM) and Gaussian Mixture Model(GMM). SGM requires a learning rate parameter only while GMM also needs the number of Gaussians to be used. Those two approaches can be accessed in separate versions of the program. Finally in the Shadow Removal version of the program which works with Background Model Update, requires four parameters: a, b, s_tau, h_tau. This version of the program works with color images while the rest of the versions of program only support gray level images. During each iteration of the program, the results folder needs to be cleaned to avoid overlapping of results from the previous run.

IV. DATA

For incremental development, AVSASlidesVideo dataset [4] was used. After developing the Task 01 i.e., foreground detection, background model update using selective running average and suppression of stationary objects that appear or are removed, we evaluated our module using the ChangeDetection datasets (2012 version, CDNet2012) [5] based on Baseline category . In the next stage, we performed evaluation on our shadow removal algorithm using the dataset [5] based on Shadow category. Finally we checked the performance of our Gaussian Mixture model algorithm on the dataset [5] using DynamicBackground Category.

V. RESULTS AND ANALYSIS

Our task 1 consists of 3 sub-tasks namely foreground detection, background model update using selective running average and suppression of stationary objects. In sub-task 1 the very first frame was used as the background model for the entire video to isolate the foreground. This fact led to some problems like being unable to adapt with the slow varying ambient light and stationary objects in the scene. In figure 1 which is an empty office room with changing light intensity. As there is no moving object in the scene, expected foreground mask would be completely black (top-left) but for the current method it can be seen that from figure 1 (bottom-left) it's not adopting with the change of light thus marking almost all the pixels as foreground pixels whose intensity values are changing in the scene. In figure 2 it can be observed that, even though there is only one mouse in the original scene, the foreground mask shows us two. When the person in the video sequence moves the mouse our algorithm marks it as foreground both in its initial position and current position.

In subtask 02 we have implemented a background model update using selective running average which actually updates those pixels that belong to the background depending on the previous background and current frame. We can regulate the adoption rate with the alpha parameter in (2). Due to the frame wise adoption it solved the first problem mentioned above with the changing ambient light. However this approach did not solve the problem of stationary objects which led us to the development of the 3rd approach. In the 3rd approach, namely suppression of stationary objects, what we are doing is we are counting the number of occurrence of every pixel in the foreground.

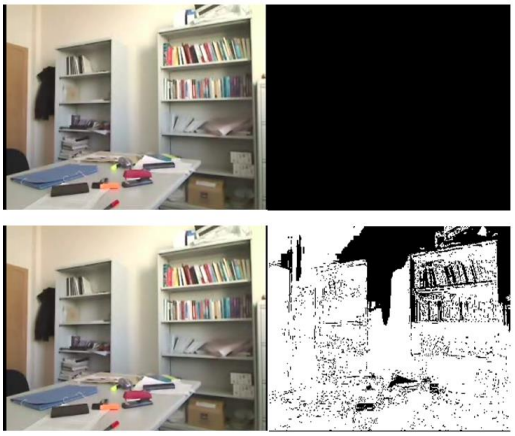

Fig. 1. **Foreground Detection for Ambient Light Change** (Top-left: current frame, Top-right: corresponding foreground mask, Bottom-left: next frame, Bottom-right: corresponding foreground mask)

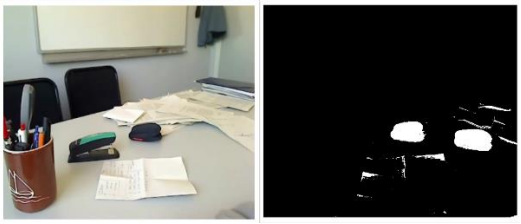

Fig. 2. **Foreground Detection for Stationary Object** (Left: current frame, Right: corresponding foreground mask)

The idea is that if an object (pixels) stays long enough without changing in subsequent frames it can be considered as background thus can be included in the background. This approach was successful in solving the problem with two mouse stated above.

Next the above method was tested on the baseline category of ChangeDetection datasets (2012 version, CDNet2012) [5] which yielded an F-Score of 74.25 %. Following TABLE. I shows performance of this method in details with different measures and parameter settings.

Though the discussed methods are decent for a somewhat non dynamic background and environment. In the case of dynamic backgrounds we need a more robust background subtraction model which is not hardly dependent on the various thresholds and can adapt with the changing scene. In order to cope with this kind of situation we have implemented a single (unimodal) and mixture of Gaussian (multimodal) based advanced background subtraction models. Unimodal models isolate background and foreground pixels depending on single Gaussian distribution whereas multimodal model uses multiple Gaussian distributions. We evaluated our model on the dynamic background category of ChangeDetection datasets (2012 version, CDNet2012) [5]. Although the performance of the Gaussian mixture model was expected to be better than the Single Gaussian model but in our case we found the opposite. One possible reason can be parameter tuning for both models, more extensive parameter tuning might yield an expected result. Another important factor that affects the background subtraction and ultimately object detection is the shadow of different moving objects.

We also implemented a chromaticity based shadow detection method and evaluated the method on the Shadow category of the ChangeDetection datasets (2012 version, CDNet2012) [5]. This approach yielded a F-Score of 69.61 % for the shadow category.

However, all the results are dependent on the settings of the parameters used and this setting varies in each application scenario. Same setting of the parameters may provide the best response for a certain case while leading to worse performance for another. The trick is, one has to play around to find an optimal set of parameters for their application.

VI. CONCLUSIONS

In conclusion, we found that detecting objects is very challenging. Extensive parameter selection is required for different application scenarios. While static backgrounds are easy to handle, dynamic background application demands more advanced tools. Although chromaticity based shadow detection method provides a decent performance but it can be further improved by more sophisticated approaches. Further, non-parametric methods can be explored to compare how they respond in the same scenario our program was tested.

TABLE I. PERFORMANCE EVALUATION

Algorithm/ Method	Parameter Setting	Overall	Recall	Specificity	FPR	FNR	PWC	Precision	F-Measure
Background Subtraction with Stationary object Suppression.	tau=45, alpha=0.03, ghost_counter=200	0.800	0.643	0.995	0.004	0.356	1.975	0.888	0.742
Shadow Removal	tau=45; alpha=0.03; counter=200;a=0.3; b=0.5; s_tau=60; h_tau=80;	0.813	0.663	0.990	0.009	0.336	2.261	0.738	0.696
Single Gaussian Model	alpha=0.5; USD= 1.8; LSD=1.8;	0.984	0.663	0.990	0.009	0.336	2.261	0.738	0.696
Gaussian Mixture Model	alpha=0.01 ; Wth=0.8;	1.047	0.502	0.955	0.044	0.497	4.916	0.177	0.237

VII. Time Log

- **Foreground detection** 2 hours: Understanding the framework provided by the lab took 1 hour, 15 minutes to write the algorithm and 45 minutes for debugging and finding the optimal threshold.
- **Background model update** 7 hours: 1 hours to design the blind update model, 3 hours to design the selective update, and 3 hours for debugging the algorithm. It took longer than expected due to data type errors in OpenCV and carrying out all the operations in the Matrix level.
- **Ghost removal** 3 hours: 20 minutes to design the structure, 2 hours to implement it in the C++ using OpenCV and 40 minutes to debug the algorithm.
- **Background model improvement** 10 hours: testing the algorithm for several combinations of parameters. Finally a set of optimal values were found for each parameter.
- **Shadow removal** 12 hours: 2 hours to design the algorithm, 1 hour to design a basic background subtraction model supporting color image,1 hour to write the code of the algorithm and 2 hours to find the appropriate data types in C++ to carry out the algorithm. 6 hours for extracting optimal thresholds for different parameters.
- **Single Gaussian Model** 6 hours: 2 hours to design the layout of the algorithm, 3 hours to implement it using only matrix operations and 1 hour to debug the algorithm.
- **Gaussian Mixture Model** 10 hours: 3 hours of planning and designing, 4 hours to implement the algorithm and 3 hours for debugging and finding suitable data types in C++.
- **Report** At least 12 hours: writing, proof-reading and editing.

References

[1] C.R. Wren et al., “Pfinder Real-Time Tracking of the Human Body”, IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(7):708-785, July 1997

[2] C. Stauffer, W.E.L. Grimson, “Adaptive background mixture models for real-time tracking”, in Proc. of IEEE Computer Vision and Pattern Recognition, June 1999, vol.2, pp. 246-252

[3] [3] R. Cucchiara, C. Grana, M. Piccardi and A. Prati, "Detecting objects, shadows and ghosts in video streams by exploiting color and motion information," Proceedings 11th International Conference on Image Analysis and Processing, Palermo, 2001, pp. 360-365.

[4] http://changedetection.net/ accessed at 08/03/2020

[5] https://posgrado.uam.es/mod/resource/view.php?id=642053 accessed at 08/03/2020