Implementation of Video Foreground Detection based on GMM

Kecheng Yang
kyang17@ubishops.ca

Yong Wang
ywang18@ubishops.ca

Michael Ekubay Negassi
mnegassi17@ubishops.ca

Jiaqi Gan
jgan17@ubishops.ca

Bill Morrisson
btalla18@ubishops.ca

Computer Science Department
Bishop's University
Sherbrooke, Canada

***Abstract*—this project implemented a foreground detection algorithm GMM by using OpenCV. Firstly, a mixed Gaussians model is used to simulate each background in each pixel of each frame of the video. The number of mixed Gaussians of each background can be adaptive. Then in the test phase, the new coming pixel of new frame will be matched by using GMM. If the pixel can match one of Gaussian, it is considered as the background. Otherwise, it is detected as the foreground. Since the GMM model is continuously updated and learned through iterative process, it is robust to process dynamic backgrounds.**

Keywords—Mixed Gaussians model, GMM, Foreground detection

I. INTRODUCTION

Video foreground detection technology is an important branch of video monitoring systems. It is applied to monitor the appearance of new targets in a particular scenario. When new targets appear in the video, the algorithm will segment it out for providing a basis of further target identification and tracking. Since the background of the camera video is always fixed, human or other moving objects become foreground correspondingly. So it is reasonable to extract moving foreground by using background modeling.

At present, the foreground detection methods are mainly divided into Interframe Difference Algorithm, Average Background Method, Optical Flow Method, Foreground modeling method, Background Non-parametric Estimation, and Background modeling method. This project is going to implement adaptive background mixture models of Gaussian for real-time foreground tracking.

The earliest application of the mixed Gaussian model in the computer vision domain is proposed by Stauffer et al [1], which is mainly used for video surveillance. This system is stable and self-learning enough to run outdoors for more than 16 months. KaewTraKulPong et al. [2] improved the training model of GMM, and divided the training process into two steps: the first L frame uses EM algorithm to perform weight, mean and variance updating; the later process use the same method to iterate to achieve better detection results. Furthermore, Zivkovic et al. gave a comprehensive discussion of GMM theory, making the use of GMM theory not only limited to the field of computer vision but also further detailed the theory into the foreground detection of background subtraction in [3], that is, adding the prior parameter estimation and achieving good effect and stability.

In this paper, based on the algorithm proposed in [2][3], update the Gaussian model to achieve those three parameters in the model. Finally, the algorithm is implemented by using wrapper functions in OpenCV.

II. PRINCIPLE OF BACKGROUND MODELING

A. Introduction of background modeling

Using statistical methods to achieve background modeling and moving target detection is a better method in the current target detection solution. This type of method characterizes the

background and establishes a background model. Different monitoring scenarios have different characteristics, so the background model contains single-modal and multi-modal. In a single-modal scene, the color value distribution of each background pixel is concentrated, and a single distributed probability model, therefore, can be simply used to describe the background. However, in the multi-modal scene the color value distribution of each background pixel is quite scattered, and it is necessary to fit the background description with a plurality of distribution probability models. In actual scenes, the background model is often multi-modal due to changes in illumination and slight disturbances in the background such as slight sway of branches in an outdoor scene.

B. Background modeling with GMM

Background modeling using Gaussian mixture models is one way to describe multi-modal scenes. This method can directly detect the pixel belonging to the target in the new frame image while continuously updating the background model, thereby reducing the calculation of the difference between images and the binary image, and effectively improving the detection speed. The basic ideal of the algorithm is to establish a mixed Gaussian model for the color value of each pixel in the image. Through a period of sampling observation, according to the persistence and variability of each Gaussian distribution, it can determine which distribution is more similar as the real background. Then if the color value of a pixel in the image does not conform to that Gaussian distribution, it is considered to be the target point. The specific calculations and derivations show as below:

Let the observations of the pixel position (x_0, y_0) in the image be a period of time:

$$\{X_1, \ldots, X_t\} = \{I(x_0, y_0, i) : 1 \le i \le t\}$$

By modeling the observations with mixed Gaussian distribution, the probability of the color value of current pixel is obtained as:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \times \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

Where K is the number of Gaussian distributions (usually 3 to 5); $\omega_{i,t}$ is the estimated value of the weight, that is, the probability that the pixel belongs to the i^{th} Gaussian distribution at time t; $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian distribution; η is the Gaussian distribution probability density function:

$$\eta\left(X_t, \mu_{i,t}, \Sigma\right) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)}$$

For simplicity of calculation, assuming that the three components (R,G,B) of the pixel point values are independent and have the same variance, the covariance matrix can be written as:

$$\Sigma_{i,t} = \sigma_k^2 I$$

Thus, A Gaussian mixture model of the color values of the observed pixel points (x_0, y_0) is established. For the pixel point (x_0, y_0, t) in the input image, compare its color value with existing K Gaussian distributions to determine whether it matches the existing Gaussian distributions. If it matches, this pixel can be seen as background point which satisfies the formula:

$$\left|\left(X_t - \mu_{i,t-1}\right)\right| < TH \times \sigma_{i,t-1}$$

Where $\mu_{i,t-1}$ is the mean of the i^{th} Gaussian distribution at time t-1, and TH is a threshold value that is usually 3.0. $\sigma_{i,t-1}$ is the standard deviation of the i^{th} Gaussian distribution at time t-1.

If it does not match any Gaussian distribution, the color value of the input pixel is used as the mean value to establish a new Gaussian distribution, which replaces the distribution with lowest probability and the lowest weight among the previous K distributions. Thus, re-establish the background model.

If there is a matching Gaussian distribution, the parameters in the background model are updated as follows:

$$\omega_{i,t} = (1 - \alpha)\,\omega_{i,t-1} + \alpha\,(M_{k.t})$$
$$\mu_i = (1 - \rho)\,\mu_{i-1} + \rho X_t$$
$$\sigma_t^2 = (1 - \rho)\,\sigma_{t-1}^2 + \rho (X_t - \mu_t)^T (X_t - \mu_t)$$

Where α is the model learning rate, for the matched Gaussian distribution $M_{k.t}$, $t = 1$, otherwise, $t = 0$ while it does not match. The formula for μ_i, σ_t^2 are only for matched Gaussian distribution, and the parameters of the remaining unmatched Gaussian distribution remain unchanged. ρ is the parameter learning rate, defined as:

$$\rho = \alpha \sum_{i=1}^{K} \omega_{i,t} \times \eta(\mu_k, \sigma_k)$$

However, after finding a matching Gaussian model, it cannot be completely determined as a background point because the background model may also contain certain noise and interference factors. They will not stay in a certain pixel for a long time in the image, so their weights should be small. Therefore, in the process of sorting Gaussian models, it is necessary to set a threshold T (usually $T = 0.7$). If the first sorted Gaussian distribution's weight is greater than T, then let $B = 1$. Otherwise, accumulating the weights of the Gaussian models in order until their weight is greater than T. B is defined as:

$$B_j = arg(\sum_{i=1}^{K} \omega_{j,t+1}^{i} > T)$$

Now, it realized the updating of GMM which can characterize the multi-modal features of the scene; it can quickly adapt to changes in the background, even when there are illumination changes in the scene, small-scale repetitive motion. If the target enters the scene and stays in the background for a long time, the Gaussian mixture model can also update the background model in time.

III. IMPLEMENTATION

It is also a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. One

important feature of this algorithm is that it selects the appropriate number of Gaussian distributions for each pixel. (Remember, in the last case, we took a K Gaussian distribution throughout the algorithm). It provides better adaptability to varying scenes due illumination changes etc. We call wrapper mog2 in our main activity, design a simple GUI to present the original video resource as input and detected real-time result as output for comparison.

A. *Object construction*

Our project is based on MOG2 which contains constructors: BackgroundSubtractorMOG2(). The major mission of constructor is to configure parameters.

```
BackgroundSubtractorMOG2::BackgroundSubtractorMOG2()
{
    frameSize = Size(0,0);
    frameType = 0;

    nframes = 0;
    history = defaultHistory2;
    varThreshold = defaultVarThreshold2;
    bShadowDetection = 1;

    nmixtures = defaultNMixtures2;
    backgroundRatio = defaultBackgroundRatio2;
    fVarInit = defaultVarInit2;
    fVarMax  = defaultVarMax2;
    fVarMin = defaultVarMin2;

    varThresholdGen = defaultVarThresholdGen2;
    fCT = defaultfCT2;
    nShadowDetection =  defaultnShadowDetection2;
    fTau = defaultfTau;
}
```

```
BackgroundSubtractorMOG2::BackgroundSubtractorMOG2(int _history,  float _varThreshold, bool _bSha
dowDetection)
{
    frameSize = Size(0,0);
    frameType = 0;

    nframes = 0;
    history = _history > 0 ? _history : defaultHistory2;
    varThreshold = (_varThreshold>0)? _varThreshold : defaultVarThreshold2;
    bShadowDetection = _bShadowDetection;

    nmixtures = defaultNMixtures2;
    backgroundRatio = defaultBackgroundRatio2;
    fVarInit = defaultVarInit2;
    fVarMax  = defaultVarMax2;
    fVarMin = defaultVarMin2;

    varThresholdGen = defaultVarThresholdGen2;
    fCT = defaultfCT2;
    nShadowDetection =  defaultnShadowDetection2;
    fTau = defaultfTau;
}
```

history: if learningRate is not set, history will be used to calculate the current learningRate. LearningRate decreases while history increases and the background updates slower.

varThreshold[Tb]: Variance threshold is used to judge the current pixel belongs to background or foreground.

nmixtures: The amount of Gaussian models.

backgroundRatio[Tb]: The weight and Variance threshold of Gaussian model. After the nmixtures models are sorted by weight, we only take the models whose accumulated value is greater than the backgroundRatio. In other words, if this value is too small, it is possible to use the biggest weight model as background.

fVarInit: the initial value of new Gaussian model, default as 15

fVarMax: during the procedure of updating background, it is used to limit the maximum values of variance of Gaussian model, default as 20

fVarMin: during the procedure of updating background, it is used to limit the minimum values of variance of Gaussian model, default as 4

fTau : Tau is a threshold on how much darker the shadow can be.Tau= 0.5 means that if pixel is more than 2 times darker then it is not shadow

B. *Algorithm execution*

```
void BackgroundSubtractorMOG2::operator()(InputArray _image, OutputArray _fgmask, double learning
Rate)
{
    Mat image = _image.getMat();
    bool needToInitialize = nframes == 0 || learningRate >= 1 || image.size() != frameSize || ima
ge.type() != frameType;

    if( needToInitialize )
        initialize(image.size(), image.type());

    _fgmask.create( image.size(), CV_8U );
    Mat fgmask = _fgmask.getMat();

    ++nframes;
    learningRate = learningRate >= 0 && nframes > 1 ? learningRate : 1./min( 2*nframes, history
);
    CV_Assert(learningRate >= 0);

    parallel_for_(Range(0, image.rows),
                  MOG2Invoker(image, fgmask,
                              (GMM*)bgmodel.data,
                              (float*)(bgmodel.data + sizeof(GMM)*nmixtures*image.rows*image.col
s),
                              bgmodelUsedModes.data, nmixtures, (float)learningRate,
                              (float)varThreshold,
                              backgroundRatio, varThresholdGen,
                              fVarInit, fVarMin, fVarMax, float(-learningRate*fCT), fTau,
                              bShadowDetection, nShadowDetection));
}
```

In the 13th line, we can see the detail of history and learningRate mentioned above. There are 3 cases:

- ***Bg_model(img,fgmask) or bg_model(img,fgmask,-1).*** The function input learningRate is -1, and the learning rate is calculated according to the history value, leaningRate=1/min(2*nframes,history)
- ***bg_model(img,fgmask,0).*** The function input learningRate is zero, and the background model stops updating.
- ***bg_model(img,fgmask,n).*** n is between 0 and 1, the background model update speed is n and the larger n is , the faster the update is. The internal performance of the algorithm is that the weight of the current frame participating in the background updating is higher.

The mog2 algorithm is mainly implemented in MOGinvoker.

C. *Bgmodel data structure*

After analyzing the source code, the data structure is saved as followed in bgmodel:

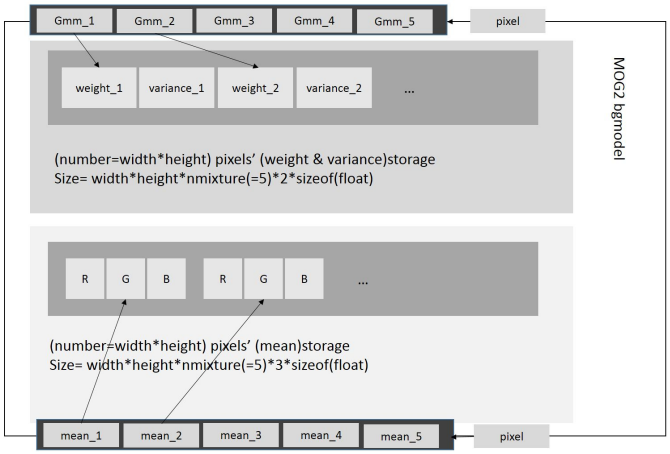


Fig. 1. Data structure of bgmodel

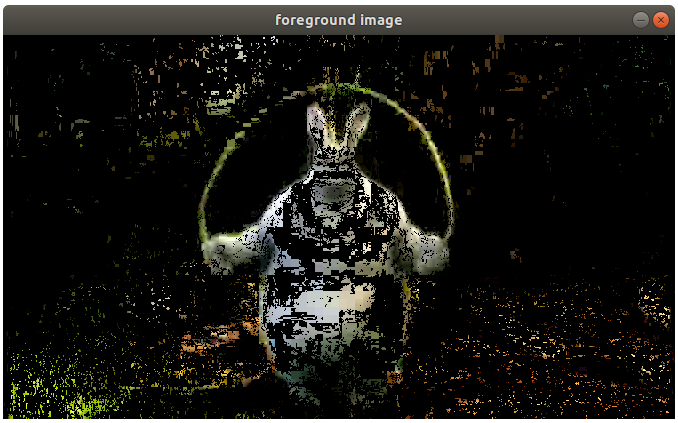
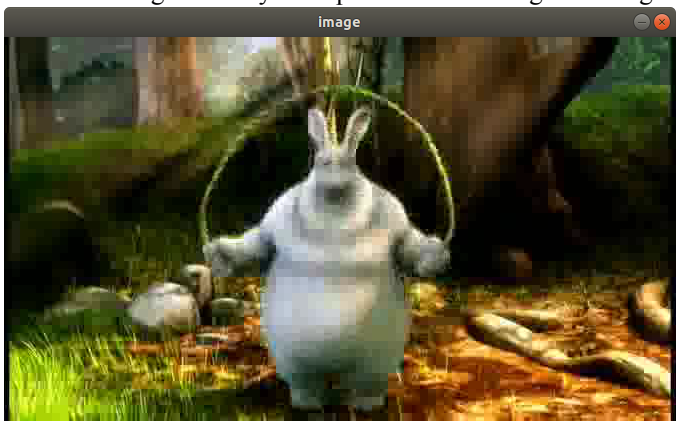
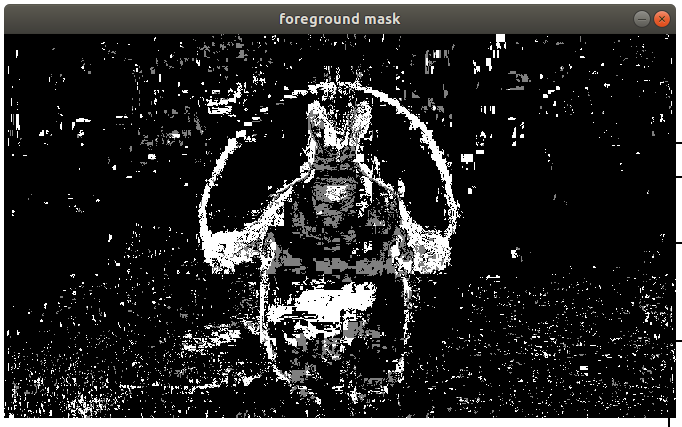


Fig. 2. Program outputs

IV. RESULT

We tested a dynamic video in a fixed scene including a rabbit doing rope skipping. The program illustrates the detected foreground by comparison with original image:





In the beginning, grass and leaves are on swing, and they are being considered as foreground by the initial model. As time goes by, models continually learn and update so that they are gradually seen as background. The rabbit is always jumping and moving to be detected as foreground. However, due to changes of light, the model cannot deal with shadow very well, so partial moving pixels are contributed into background.

CONCLUSION

This project uses GMM algorithm to detect the moving foreground from the camera video based on the mog2 class of opencv on Linux. The experiment presents that GMM has good adaptability and real-time to detect moving targets from background, through establishing and updating background mixed Gaussian distribution modeling.

CONTRIBUTION OF TEAM MEMBERS

Name	Work	Description
Yang	Coordination	Write paper, organize, design and test projects.
Wang	Implementation	Implement detection of our project, call opencv and its parameters.
Bill	Analysis and Implementation	Contribute further work on changes based on shadows, Finish and correct paper.
Negassi	Mog2 analysis	Read and learn opencv, principle and structure of mog2.
Gan	Researching	Find research references, provide project design requirements.

References

[1] C. Stauffer, W.E.L. Grimson , “Adaptive Background Mixture Models for Real-time Tracking” in 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 1999, page 2246

[2] P. KaewTraKulPongm, R. Bowden, “An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection”, in Video-Based Surveillance Sysyems, 2002, page 135-144

[3] Z. Zivkovic, F.V.D. Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” in Pattern Recognition Letters, vol. 27, Issue 7, May 2006, pages 773-780