Comparing Parametric and Non parametric approaches to Background Subtraction

Stefan Quach
stefan.quach@wustl.edu

December 22, 2020

Abstract

Background subtraction has applications in many computer vision problems, such as tracking. In the case of tracking, background subtraction can provide a preliminary estimate of moving objects. Here, two different approaches to background subtraction are compared: mixed Gaussian and kernel density estimation. The mixed Gaussian approach is a classical approach, but involves learning parameters using expectation maximization, thus inherently having some delay. The non-parametric approach of kernel density estimation adapts to fast changes and also provides more flexibility in tuning. As such, we show that this non-parametric approach yields much better results compared to the parametric approach with our naive implementations.

1 Introduction

Many tracking algorithms begin by detecting unusual movement in the current frame. This process is often called background subtraction, as it isolates moving objects in the foreground. By using background subtraction, only the moving objects are seen, making some problems much easier, like tracking. On first impression, background subtraction and be done simply by comparing the current frame with the past frames, however backgrounds are not always static, either due to noise or small movement in the background (i.e., leaves in the wind). Another complication is background detection algorithms often detect shadows, since they move with the object.

The goal of this project will be to implement and compare a parametric and non-parametric approaches to background subtraction. The two methods compared will be the Mixed Gaussian Model [3], and the Kernel Density Estimation approach [1].

2 Background & Related Work

One of the classical approaches to background subtraction is the mixed Gaussian model. The basis of this approach starts with modeling every pixel's probabilty of corresponding to the background as a single gaussian. However, this approach struggles to differentiate between objects. As such, [2] uses a weighted sum of three different Gaussians to help differentiate and represent the road, vehicles, and shadow. [3] and [4] are generalizations of this approach, using K gaussians, where K usually ranges from 3 to 5. An expectation maximization algorithm is then used to learn the parameters of the model. This approach is detailed further in Section 3.1.

One downfall of the parametric model is the time required for the model to approach a solution, thus failing to adapt to rapid changes in the environment. This can be simply depicted as a tree branch swaying in the wind. At one time, the pixel could correspond to a leaf, while in the next frame it could correspond to the sky [1].

3 Proposed Approach

3.1 Mixed Gaussian Model

The mixed Gaussian model begins by modeling each pixel's intensity as a mixture of K multivariate gaussians, according to the dimensionality of the pixel, shown in Equation 1. X_t is a single pixel at a given time t, d is the dimensionality of a pixel, ω_i is the weight associated with that Gaussian, μ_j and Σ are the jth mean and covariance matrix associated with X_t.

$$P(X_t) = \sum_{i=1}^{K} \frac{\omega_i}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)} \tag{1}$$

For simplicity, each color channel is assumed to be independent and have the same variance, thus

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \tag{2}$$

Each pixel will maintain its own set of parameters (μ,σ,ω), where expectation maximization (EM) algorithm is then

used to iteratively learn the correct values these parameters at a rate of α. If a pixel is more than 2.5 standard deviations from any of the first B distributions, then it is classified as a foreground pixel. B is defined below, where ω is sorted by ω/σ and T is some threshold between 0 and 1.

$$B = \arg\min_{b}(\sum_{i=0}^{b} \omega_i > T) \tag{3}$$

3.2 Kernel Density Estimation

Kernel density estimation is the approach detailed in [1], which bases itself in the Gaussian mixture model. It begins by maintaining N previous frame samples, and modeling the probability of the pixel being in the background as shown in Equation 4. This is a similar expression to the multivariate gaussian shown in Equation 1, however, each channel has its own variance σ_j and the means are the previous samples.

$$Pr(x_t) = \frac{1}{N}\sum_{i=0}^{N}\prod_{j=0}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{(x_{tj}-x_{ij})^2}{2\sigma_j^2}} \tag{4}$$

To classify a pixel as backgroun or foreground, any value below a certain threshold probability th will be marked as a foreground pixel. This threshold remains constant and global to every pixel.

To estimate the standard deviation for each pixel and channel, the median absolute difference is taken between consecutive samples of the same pixel. This assumes that the most consecutive samples come from the same distribution. Thus, the standard deviation can be calculated using Equation 5.

$$.\sigma_{ij} = \frac{|x_{ij} - x_{i+1,j}|}{0.68\sqrt{2}} \tag{5}$$

[1] also implements a second stage of filtering in order to remove false positives, but this was not implemented and is discussed in Section 5.

4 Experimental Results

For simplicity of implementation, and to reduce computational loads, the following methods were only implemented for grayscale images (one-dimentional pixels). The kernel density estimation method was implemented for color images, however this turned out to be computationally infeasible, and thus grayscale images were used.

4.1 Functionality Test

For each model, a small video (240x320) of a small remote controlled car is used to test functionality of the model. This video was purposefully made to be easy for background detection as the background is very static (no moving leaves, bushes, etc) and there is only one moving object with no shadow. In terms of choosing parameters, the mixed Gaussian model initially used $\alpha = 0.3$, $K = 3$ and $T = 0.5$. With this configuration, the video was able to play at roughly the normal framerate, with no noticiable lag. One frame of this configuation is shown below in Figures 1 and 2.

Figure 1: Mixed Gaussian Foreground

Figure 2: Original image

In Figure 1, there are some false positive pixels seen, which can be reduced by lowering α. However, this can also lead to the loss of some real foreground pixels. Additionally, due to the model adapting slower from the lower α, a trail of foreground pixels is left behind the car as the pixel's distributions adapt. The foreground picture of the same frame, but with $\alpha = 0.1$ is shown in Figure 3.

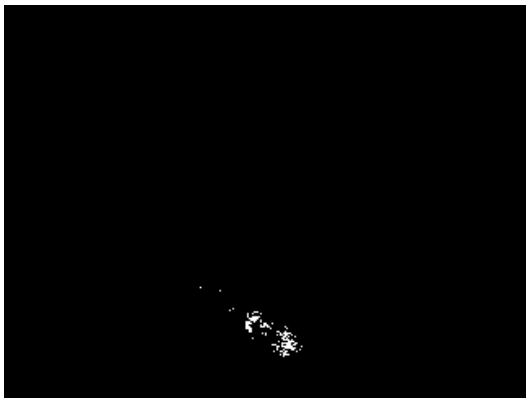

Figure 3: Mixed Gaussian Foreground mask with $\alpha = 0.1$

Increasing K results in more computation, as more gaussians must be computed. At around $K = 5$ the frame rate began to dip on this small video.

The Kernel density approach was much more computationally expensive, as a sufficient number of samples must be acquired for adequate performance. After some tuning, the $N = 8$ and $th = 0.02$ were chosen. A specific frame from the same video is shown below in Figures 4 and 5.

Figure 4: KDE Foreground

Figure 5: Original image

In terms of tuning, lowering N decreases computation intensity, but also decreases the stability of the subtractor. With lower N, the filter tends to cause some background pixels to fluctuate between foreground and background. The better parameter to change is th, since it directly affects the false positive rate. Thus, lowering th also lowers the number of false positives, but also runs the risk of causing some false negatives.

4.2 Performance test

In order to evaluate the performance of each algorithm, a video from a highway surveillance camera was taken. This video contains much more irregularities, such as the bush on the right side of the frame. Furthermore, shadows are much more of a problem, since the angle of the camera can easily see the cars' shadows. The video is much larger, thus there was significantly more lag in each frame. For the mixed Gaussian model, $K = 3$ and $\alpha = 0.1$ were chosen. Frame 10 of the video is shown below in Figure 6 and 7

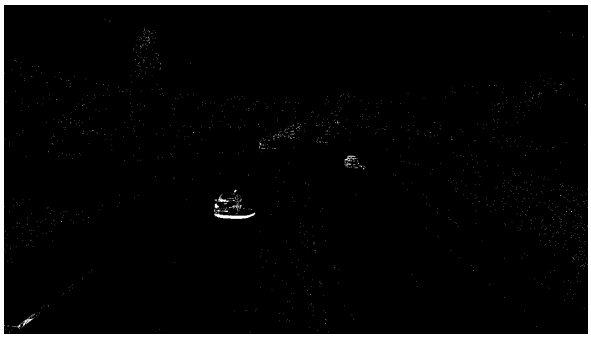

Figure 6: Mixed Gaussian Foreground

Figure 7: Original image

Within the image, this model fails to really capture the full body of every car, often missing the windshield or the roof of certain cars. Additionally, the subtractor also detects the shadow of the car very distinctly. Furthermore,

there are many false positives within the bushes and the trees.

Shown below in Figure 8 shows the same frame as Figure 6, but running the Kernel density subtractor with $N = 8$ and $th = 0.01$.

Figure 8: Original image

This subtractor detects the full body of the car, unlike the mixed gaussian model. However, like the gaussian model, it also detects the shadows of the cars and detects the bushes and trees as foreground. However, there were less false positives in the background trees and landscape.

5 Conclusion

Overall, the Kernel density estimator was better at successfully detecting more of cars in their entirety on the highway. However, in their current, naive implementations, they still falsely detect moving foliage and shadows within the frame. The non-parametric approach does allow for tuning without risking causing trails, and adapting to fast changes in the environment.

However, due to the naive nature of the implementation, the computation intensity was quite high. The original paper [1] proposed a method of accelerating the computation of gaussians using a lookup table. To further speed up runtime, a partial evaluation of the sum in Equation 4 if often sufficient [1].

In terms of future work, [1] also outlines a method for suppressing false detections. This involves probabilities in a neighborhood around a given pixel, thus making it more likely that a foreground pixel will be next to a foreground pixel. This clustering can help suppress false detection of moving foliage and improve the erroneous detections in 8. However, this implementation would likely suffice for any low-framerate application.

Acknowledgments

It was extremely helpful to have a naive Python implementation of the gaussian mixture model while making my own implementation. Code for the mixture model was adapted from Github user rajan951. Their code is linked here.

References

[1] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In D. Vernon, editor, *Computer Vision — ECCV 2000*, pages 751–767, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[2] N. Friedman and S. J. Russell. Image segmentation in video sequences: A probabilistic approach. *CoRR*, abs/1302.1539, 2013.

[3] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pages 22–29, 1998.

[4] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, 1999.