Understanding various Techniques for Background Subtraction and Implementation of Shadow Detection.

Harsha Varwani  Heena Choithwani  Kajal Sahatiya  Shruti Gangan  Tina Gyanchandani  Dashrath Mane  
VES Institute of Technology, Chembur  
varwani.harsha@gmail.com

Abstract

In this paper, we present steps involved in Background Subtraction and their techniques. However, a prominent problem in these is presence of shadow in the foreground. Hence, we discuss two techniques for efficient shadow removal. Further, we conclude with a method to reduce considerably the processing overhead. Here, we take alternate frames and apply background subtraction algorithms on them. This does not impact much on the output data, however, resources are greatly saved.

Keywords: Background subtraction, shadow removal

I. Introduction

Identifying moving objects from a video sequence is a fundamental and critical task in video surveillance, traffic monitoring and analysis, human detection and tracking, and gesture recognition in human-machine interface.

A common approach to identifying the moving objects is background subtraction, where each video frame is compared against a reference or background model. Pixels in the current frame that deviate significantly from the background are considered to be moving objects. These foreground pixels are further processed for object localization and tracking. Since background subtraction is often the first step in many computer vision applications, it is important that the extracted foreground pixels accurately correspond to the moving objects of interest. Even though many background subtraction algorithms have been proposed in the literature, the problem of identifying moving objects in complex environments is still far from being completely solved. There are several problems that a good background subtraction algorithm must solve correctly. Consider a video sequence from a stationary camera overlooking a traffic intersection. As it is an outdoor environment, a background subtraction algorithm should adapt to various levels of illumination at different times of the day and handle adverse weather conditions such as fog or snow that modifies the background. Changing shadow, cast by moving objects, should be removed so that consistent features can be extracted from the objects in subsequent processing. The complex traffic flow at the intersection also poses challenges to a background subtraction algorithm. The vehicles move at a normal speed when the light is green, but come to a stop when it turns red. The vehicles then remain stationary until the light turns green again. A good background subtraction algorithm must handle the moving objects that first merge into the background and then become foreground at a later time. In addition, to accommodate the real-time needs of many applications, a background subtraction algorithm must be computationally inexpensive and have low memory requirements, while still being able to accurately identify moving objects in the video.

II. BACKGROUND SUBTRACTION ALGORITHMS.

Figure 1. Steps in Background Subtraction

Most of the background subtraction algorithms follow a simple flow diagram shown in Figure 1[3]. The four major steps in a background subtraction algorithm are

1. Preprocessing
2. Background Modeling
3. Foreground detection
4. Data Validation.
Preprocessing consists of a collection of simple image processing tasks that change the raw input video into a format that can be processed by subsequent steps.

Background modeling uses the new video frame to calculate and update a background model. This background model provides a statistical description of the entire background scene.

Foreground detection then identifies pixels in the video frame that cannot be adequately explained by the background model, and outputs them as a binary candidate foreground mask.

Finally, data validation examines the candidate mask, eliminates those pixels that do not correspond to actual moving objects, and outputs the final foreground mask. Domain knowledge and computationally-intensive vision algorithms are often used in data validation. Real-time processing is still feasible as these sophisticated algorithms are applied only on the small number of candidate foreground pixels. Many different approaches have been proposed for each of the four processing steps. We review some of the representative ones in the following subsections.

1. Preprocessing

In most computer vision systems, simple temporal and/or spatial smoothing are used in the early stage of processing to reduce camera noise. Smoothing can also be used to remove transient environmental noise such as rain and snow captured in outdoor camera. For real-time systems, frame-size and frame-rate reduction are commonly used to reduce the data processing rate. Another key issue in preprocessing is the data format used by the particular background subtraction algorithm. Most of the algorithms handle luminance intensity, which is one scalar value per each pixel. However, color image, in either RGB or HSV color space, is becoming more popular in the background subtraction literature. These papers argue that color is better than luminance at identifying objects in low-contrast areas and suppressing shadow cast by moving objects. In addition to color, pixel-based image features such as spatial and temporal derivatives are sometimes used to incorporate edges and motion information. For example, intensity values and spatial derivatives can be combined to form a single state space for background tracking with the Kalman filter. The main drawback of adding color or derived features in background modeling is the extra complexity for model parameter estimation. The increase in complexity is often significant as most background modeling techniques maintain an independent model for each pixel.

2. Background Modeling

Background modeling is at the heart of any background subtraction algorithm. Much research has been devoted to developing a background model that is robust against environmental changes in the background, but sensitive enough to identify all moving objects of interest. We classify background modeling techniques into two broad categories - non-recursive and recursive. They are described in the following subsections.

2.1. Non-recursive Techniques

A non-recursive technique uses a sliding-window approach for background estimation. It stores a buffer of the previous L video frames, and estimates the background image based on the temporal variation of each pixel within the buffer. Non-recursive techniques are highly adaptive as they do not depend on the history beyond those frames stored in the buffer. On the other hand, the storage requirement can be significant if a large buffer is needed to cope with slow-moving traffic. Given a fixed-size buffer, this problem can be partially alleviated by storing the video frames at a lower frame-rate r. Some of the commonly-used non-recursive techniques are described below:

2.1.1. Frame differencing

Frame differencing uses the video frame at time t - 1 as the background model for the frame at time t. Since it uses only a single previous frame, frame differencing may not be able to identify the interior pixels of a large, uniformly-colored moving object. This is commonly known as the aperture problem.

2.1.2. Median Filter

Median filtering is one of the most commonly-used background modeling techniques. The background estimate is defined to be the median at each pixel location of all the frames in the buffer. The assumption is that the pixel stays in the background for more than half of the frames in the buffer.

Median Filtering has been extended to color by replacing the median with the medoid. The complexity of computing the median is O(Llog L) for each pixel.

2.2. Recursive Techniques

Recursive techniques do not maintain a buffer for background estimation. Instead, they recursively update a single background model based on each input frame. As a result, input frames from distant past could have an effect on the current background model.

Compared with non-recursive techniques, recursive techniques require less storage, but any error in the background model can linger for a much longer period of time. Most schemes include exponential weighting to discount the past, and incorporate positive decision feedback to use only background pixels for updating. Some of the representative recursive techniques are described below:
2.2.1. Approximated median filter
Due to the success of non-recursive median filtering, propose a simple recursive filter to estimate the median. This technique has also been used in background modeling for urban traffic monitoring. In this scheme, the running estimate of the median is incremented by one if the input pixel is larger than the estimate, and decreased by one if smaller. This estimate eventually converges to a value for which half of the input pixels are larger than and half are smaller than this value, that is, the median.

2.2.2. Kalman filter
Kalman filter is a widely-used recursive technique for tracking linear dynamical systems under Gaussian noise. Many different versions have been proposed for background modeling, differing mainly in the state spaces used for tracking. The simplest version uses only the luminance intensity Karmann and von Brandt use both the intensity and its temporal derivative, while Koller, Weber, and Malik use the intensity and its spatial derivatives.

2.2.3. Mixture of Gaussians (MoG)
Unlike Kalman filter which tracks the evolution of a single Gaussian, the MoG method tracks multiple Gaussian distributions simultaneously. MoG maintains a density function for each pixel. It is capable of handling multi-modal background distributions. Since MoG is parametric, the model parameters can be adaptively updated without keeping a large buffer of video frames.

III. Foreground Detection
There are two types of B/F detection methods: –

1. Non-adaptive
It depends on certain numbers of video frames and do not maintain a background model in the algorithm.
• B/F detection based on two frames.
• B/F detection based on three frames.
   • Drawback of non-adaptive methods
     ○ The non-adaptive methods are useful only in high-supervised, short-term tracking applications without significant changes in the video scene.
     ○ When errors happen, it requires manual re-initialization.
     ○ Without re-initialization, errors in the background accumulate over time.

2. Adaptive Background/Foreground Detection
It maintains a background model and the parameters of the background model evolve over time. Adaptive B/F detection is chosen for more and more VCA applications because of the limitation of non-adaptive methods. A standard adaptive B/F detection method maintains a background model within the system.

Video Content Analysis (VCA) is the capability of automatically analyzing video to detect and determine temporal events not based on a single image[4].

The algorithm of adaptive B/F detection is described below.
– fi : A pixel in a current frame, where i is the frame index.
– µ : A pixel of the background model (fi and m are located at the same location).
– di : Absolute difference between fi and m.
– bi : B/F mask - 0: background. 0 x ff: foreground.
– T : Threshold
– α: Learning rate of the background.
  i) di = |fi - µ|
  ii) If di > T , fi belongs to the foreground; otherwise, it belongs to the background.

2.4. Data Validation
We define data validation as the process of improving the candidate foreground mask based on information obtained from outside the background model. All the background models have three main limitations: First, they ignore any correlation between neighboring pixels; second, the rate of adaptation may not match the moving speed of the foreground objects; and third, non-stationary pixels from moving leaves or shadow cast by moving objects are easily mistaken as true foreground objects.

The first problem typically results in small false-positive or false-negative regions distributed randomly across the candidate mask. The most common approach is to combine morphological filtering and connected component grouping to eliminate these regions. Applying morphological filtering on foreground masks eliminates isolated foreground pixels and merges nearby disconnected foreground regions. Many applications assume that all moving objects of interest must be larger than a certain size. Connected-component grouping can then be used to identify all connected foreground regions, and eliminates those that are too small to correspond to real moving objects.

When the background model adapts at a slower rate than the foreground scene, large areas of false foreground, commonly known as ghosts, often occur. If the background model adapts too fast, it will fail to identify the portion of a foreground object that has corrupted the background model. A simple approach to alleviate these problems is to use multiple background models running at different adaptation rates, and periodically cross-validate between different models to improve performance.

Sophisticated vision techniques can also be used to validate foreground detection. Computing optical flow for candidate foreground regions can eliminate ghost objects as they have no motion. Color segmentation
can be used to grow foreground regions by assuming similar color composition throughout the entire object. If multiple cameras are available to capture the same scene at different angles, disparity information between cameras can be used to estimate depth. Depth information is useful as foreground objects are closer to the camera than background.

The moving-leaves problem can be addressed by using sophisticated background modeling techniques like MoG and applying morphological filtering for cleanup.

III. Measures for shadow removal

1. Using invariance against illumination changes [1]

A main difficulty with shadow is that intensity differences between shadows and background are often larger than differences between some foreground objects and background. In order to address this problem, we use window-based decision rules. By comparing statistical characteristics of the window, we explored several distance measures that are invariant against illumination changes. First, we used the Bhattacharyya distance by representing each pixel as a vector of red, green and blue. Bhattacharyya Distance measures the similarity of two discrete probability distributions [5]. It is used to evaluate the degree of similarity between two histograms. The Bhattacharyya distance is defined as follows:

\[
b = \frac{1}{8}(\sum_i (M_2 - M_1) \sqrt{\frac{1}{2}} - \frac{\sum_i M_2}{2} + \frac{1}{2} \ln \frac{\sum_i M_2}{\sum_i M_1})
\]

The first term represents the mean difference and the second term represents the covariance difference. We also tested a normalized mean difference:

\[
D_{\text{norm}, M} = 1 - \frac{|M_1 - M_2| + K_1}{|M_1 + M_2| + K_1}
\]

\[
D_{\text{norm}, ed} = 1 - \frac{\sigma_1 - \sigma_2 + K_2}{\sigma_1 + \sigma_2 + K_2}
\]

Where Ki is a constant, which is introduced to prevent zero denominators. On the other hand, the correlation coefficient provides similarity information about two sets of data.

Furthermore, it is invariant under any linear transformation.

\[
\text{Corr} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}
\]

The structural similarity (SSIM) index is a method for measuring the similarity between two images. The measure of image quality based is on an initial uncompressed or distortion-free image as reference. Structural similarity (SSIM) was proposed to measure image quality and is computed as follows:

\[
\text{SSIM}(x,y) = \frac{(2\mu_x \mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_2)(\sigma_x^2 + \sigma_y^2 + C_2)}
\]

We divided SSIM into two terms and use them separately as follows:

\[
\text{SSIM}_1 = \frac{(2\mu_x \mu_y + C_1)}{(\mu_x^2 + \mu_y^2 + C_2)}
\]

\[
\text{SSIM}_2 = \frac{(\sigma_{xy} + C_2)}{(\sigma_x^2 + \sigma_y^2 + C_2)}
\]

Figure 2. (a) input image, (b) difference image between the reference and input images, (c) Bhattacharyya distance, (d) correlation, (e) SSIM, (f) normalized mean difference.

First, we applied the features to an image that has strong shadow presence (Fig. 2(a)). Fig.2(b) shows the difference image between the reference and input images, which shows the shadow problems. Fig. 2(c) is the result of the Bhattacharyya distance. Although it solved some shadow problems, it failed to completely remove the shadows. Fig. 2(d) is the result of correlation, which solved most of the shadow problem. SSIM and the normalized mean difference also showed good performance (Fig. 2(e) and 2(f)).
2. Pixel Classification using Chromaticity and Brightness Distortion

In this step, the difference between the background image and the current image is evaluated. The difference is decomposed into brightness and chromaticity components. Applying the suitable thresholds on the brightness distortion ($\alpha$) and the chromaticity distortion ($CD$) of a pixel $i$ yields an object mask $M(i)$ which indicates the type of the pixel.[2]

Pixel in the current image is:
- **Original background (B)** if it has both brightness and chromaticity similar to those of the same pixel in the background image [2].
- **Shaded background or shadow (S)** if it has similar chromaticity but lower brightness than those of the same pixel in the background image. This is based on the notion of the shadow as a semi-transparent region in the image, which retains a representation of the underlying surface pattern, texture or color value [2].
- **Highlighted background (H)** if it has similar chromaticity but higher brightness than the background image [2].
- **Moving foreground object (F)** if the pixel has chromaticity different from the expected values in the background image [2].

As mentioned above the different pixels yield different distributions of $\alpha_i$ and $CD_i$. In order to use a single threshold for all of the pixels, we need to rescale the $\alpha_i$ and $CD_i$. Let

$$\hat{\alpha}_i = \frac{\alpha_i - 1}{\alpha_i}$$

$$\hat{CD}_i = \frac{CD_i}{b_i}$$

be normalized brightness distortion and normalized chromaticity distortion respectively.

Based on these definitions, a pixel is classified into one of the four categories B; S; H; F by the following decision procedure described in Table 1.

<table>
<thead>
<tr>
<th>Table 1. Classification of a Pixel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel Category</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>M(i) F</td>
</tr>
<tr>
<td>M(i) B</td>
</tr>
<tr>
<td>M(i) S</td>
</tr>
<tr>
<td>M(i) H</td>
</tr>
</tbody>
</table>

However, there might be a case where a pixel from a moving object in current image contains very low RGB values. This dark pixel will always be misclassified as a shadow. Because the color point of the dark pixel is close to the origin in RGB space and the fact that all chromaticity lines in RGB space meet at the origin, thus the color point is considered to be close or similar to any chromaticity line. To avoid this problem, we introduce a lower bound for the normalized brightness distortion ($T_{\alpha_{lo}}$). Then, the decision procedure Equation becomes as described in Table 2.

<table>
<thead>
<tr>
<th>Table 2. Classification of a Pixel (low RGB values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel Category</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>M(i) F</td>
</tr>
<tr>
<td>M(i) B</td>
</tr>
<tr>
<td>M(i) S</td>
</tr>
<tr>
<td>M(i) H</td>
</tr>
</tbody>
</table>

Figure 3 shows the result of our algorithm applying on a sequence of a person moving in an indoor scene. The upper left image is the background scene, the upper right image is the input sequence, and the lower left image shows the output from our background subtraction. The lower right image shows only foreground region after noise cleaning is performed.

V. Conclusion and discussions:

In this paper, we introduced various Background Modeling and subtraction techniques with a prominent problem bring inclusion of Shadow as a Foreground
Object. Further, we see methods to overcome this problem. After applying the shadow removal using invariance against illumination changes, we see improvements in obtaining the foreground data when compared to traditional background subtraction methods. However, this method requires a lot of processing. We may reduce this time by considering alternate frames instead of all the frames. Also, the method of pixel classification involves the similar problem of excessive computation. Also, the method involving Chromaticity Distortion fails in case of identifying Foreground Objects in case of monochromatic video sequences. With some effort, we can reduce the overhead by dividing the image into segments and performing the algorithm on individual segment.

VI. References: