Motion Detection Using Simple Image Filtering

Cao Ning
cao.nin@northeastern.edu

Yihao Huang
huang.yihao@northeastern.edu

Abstract

The purpose of this project is to explore different filters that can be used to detect motion in image sequences captured by a stationary camera. The images mostly contain a stationary background with small moving objects passing in front of the camera. The main goal is to compare the effectiveness of various algorithms and filters in detecting motion by analyzing the differences in intensity between the background and the moving objects. The findings, including the most effective algorithms and threshold cutoffs, are available in the project's Github code repository.

1. Introduction

Detecting motion in image sequences captured by a stationary camera is critical for various applications, including object tracking, event recognition, and video surveillance. In this project, we investigate the use of simple techniques for motion detection that leverage the stationary background present in the image sequences. By observing the differences in intensity between the stationary background and moving objects in the foreground, we aim to detect motion.

To accomplish this, we design, implement, and compare several filters based on their performance in detecting motion. We also design The project code is available in the Github repository linked in the Appendix. The project's outcomes provide useful insights for practitioners and researchers working in the fields of computer vision and image processing.

2. Description of Algorithms

The initial task of our project involved converting a sequence of image frames into grayscale using the cv2 package in Python. We utilized the cv2 imread function to read the images and transformed them into grayscale using cvtColor. To speed up the processing part of the program, we read all the images in the folder and stored them in a numpy array.

Next, we used a 1D differential operator to calculate the temporal derivative and then took the absolute value of the output. After that, we applied a threshold function to map the pixels in our images to 0 or 1 based on whether they exceeded a particular threshold value.

Once we had our mask, we used the numpy multiply to combine the mask with the original image to highlight only the pixels or objects that are in motion. Initially, we used the multiply function, which should theoretically do the same thing since our mask is binary.

3. Experiments

We implemented Gaussian filters with various kernel sizes and sigma values to compare with the naïve filter.

In addition to using a single filter for computing derivatives, we also employed multiple two-dimensional filters for smoothing the image sequence. These filters help to reduce noise in the image and make it easier to detect motion accurately. There are various types of two-dimensional filters that can be used for smoothing, including box and Gaussian filters. We also explored different sizes and sigmas for Gaussian filters.

Our main focus was to develop a thresholding mechanism through noise standard deviation. We choose the threshold value by selecting the best factor. This enabled us to accurately identify events even when the images changed.

3.1. Temporal Difference

The temporal difference (TD) filter is the first method for approximating the temporal gradient. We define a 1D filter with the kernel

$$0.5 \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

3.2. 1D Derivative of Gaussian

We also use a derivative of Gaussian (DOG) method for approximating the temporal gradient. This method is similar, except that we calculate our 1D kernel using the equation for the derivative of the 1D Gaussian distribution a chosen σ_t value.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\frac{\partial f(x)}{\partial x} = -\frac{(x-\mu)}{\sigma^2} \cdot f(x)$$

we also normalize the full kernel at the end.

3.3. Box Filter

For spatial smoothing, we have box filter and 2d gaussian filter. There are two sizes for box filter 3x3 and 5x5. For each pixel in each image, instead of considering the pixel's raw value we consider the average value of it and its neighbors within some n×n neighborhood. We achieve this by defining an n×n kernel.

$$1/n^2 \cdot \begin{bmatrix} 1 & ... & 1 \\ ... & 1 & ... \\ 1 & ... & 1 \end{bmatrix}_{n \times n}$$

3.4. 2D Gaussian Filter

The 2d gaussian filter is another filter for spatial smoothing. The 2d gaussian filter is shown below.

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}$$

We then calculate the value of each element in the kernel such that the (i,j) element is f(x_i,y_i) where x_i and y_i are the i^{th} and j^{th} elements in $[\frac{-n+1}{2}, ..., 0, ..., \frac{n-1}{2}]$. We also normalize the full kernel at the end. This provides us with maximum element in the center pixel and the neighbor of pixels attenuate exponentially with the distance from the center.

4. Observation

4.1. Temporal Difference

In the case of motion detection using temporal differencing (TD), the small number of samples considered can result in speckled and jagged masks due to variations in the background or the moving object. The masks can also show sudden changes from frame to frame during video playback. On the other hand, the difference of Gaussians (DOG) method considers a larger number of time steps while weighting them according to their proximity to the current time step, which reduces the impact of individual noisy samples. The DOG method produces more consistent masks that seem to flow better from frame to frame.

However, the DOG method has a failure mode at high values of size, where the kernel size increases, and pixels that recently but not currently observed motion are still considered to have seen motion. This can result in a light trail around moving objects in the frame. Also, at very high values of σ_t, many constant background pixels before and after the actual motion can enter the estimate of the temporal gradient, potentially obliterating motion detection.

Through the following Figure 1, we found that one-dimensional Gaussian performs better, as more noise is left after simple filtering. However, one-dimensional Gaussian derivative filters of different sizes also produce different effects on the image at different sigma values. By comparing Figure (b) and (c), we can see that when the size is the same, the larger the sigma value, the smaller the noise in the image. On the other hand, when the sigma value is the same, the larger the size, the more ghosting appears in the image. Specifically, Figure (d) has a size of 5 and Figure (b) has a size of 3. We can observe that with a sigma value of 1 in Figure (d), two persons are detected in the image.

Finally, both methods struggle to detect the middle of moving objects when the object is slow or the frame rate is fast. In such cases, pixels in the middle of a moving object with a uniform color may not appear to change within the time window where the object occupies those pixels.

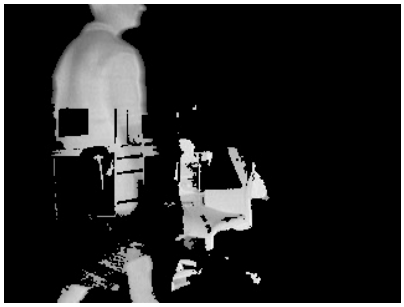
4.2. Spatial Smoothing

Spatial smoothing can help reduce noise and improve the quality of masks produced by motion detectors. However, in the case of motion detection based on the temporal gradient, it's essential to be cautious about using too large a window for spatial smoothing, as it can dilute the unusually high change in pixel value that indicates motion. For example, it could cause two people detected. Moreover, increasing the window size for spatial smoothing can cause a "halo" effect around moving objects, which affects the accuracy of motion detection.

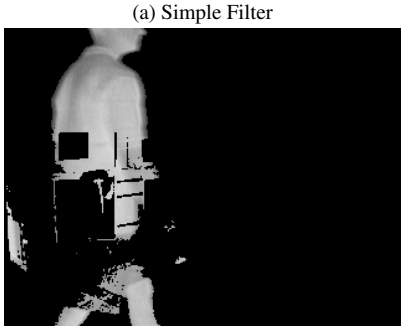
Box filtering is less effective than 2D Gaussian filtering for spatial smoothing in motion detection. A 3x3 smoothing provides little improvement, while a 5x5 smoothing already leads to significant failures in detecting moving objects. In contrast, 2D Gaussian filtering performs better with larger window sizes. This behavior is because the Gaussian filter gives higher weight to neighboring pixels and lower weight to pixels farther away from the pixel under consideration, which helps preserve the high change in pixel value that indicates motion while reducing noise.

However, when it comes to detecting motion through temporal gradient, it is crucial to bear in mind that we are identifying motion by observing an abnormally high alteration in pixel value with respect to a constant background pixel value. This implies that if our spatial smoothing involves an excessively large window and encompasses numerous adjacent background pixels, it will weaken the exceptionally high change in pixel value that we are searching for, leading to a failure in detecting motion. Moreover, if we attenuate overmuch noises, we are still going to miss out the pivotal pixels which have highly change comparing with the last image frame.

Based our results such like the following Figure 2 and 3, we found that gaussian2D works better than Box filter when we apply them to smooth. In addition, If we gradually increase the size of the image, we will observe more ghosting in the image, which is unfavorable for our motion detection.



(a) Simple Filter



(b) Gaussian1D with σ=1 & size = 3

(c) Gaussian1D with σ=3 & size = 3

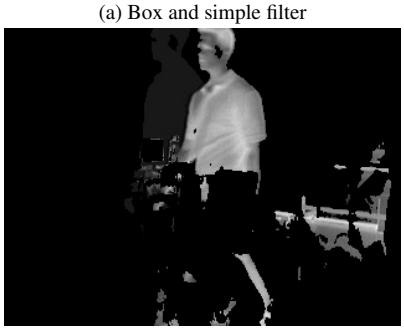
(d) Gaussian1D with σ=1 & size = 5

Figure 1

Moreover, the larger the sigma value, the less noise there will be in the image. In conclusion, the first-order Gaussian derivative has a better effect than the simple filter, and the two-dimensional Gaussian filter is better than the Box filter.

Overall, larger size and sigma will reduce more noise and improve the quality of masks produced by motion detectors. But larger window size(5x5) will detect two people which is a big failure in our experiment.

(a) Box and simple filter



(b) Box and Gaussian1D with σ=1 & size = 3

(c) Box5 and Gaussian1D with σ=1 & size = 3

Figure 2. Box filter for smoothing

4.3. Estimate thresholding

Thresholding is a crucial aspect of our studies and can have a significant impact on the detection of motion. Our experiments show that if the threshold is too low, then random fluctuations caused by noise in environmental light and camera electronics will be detected as motion. On the other hand, if the threshold is too high, sections of actual moving objects may go undetected. Furthermore, if smoothing is used, a lower or higher threshold can increase or decrease the halo effect.

Based on our empirical results, we found that the best performance for the TD case with no smoothing was achieved with threshold values around 10.

In the Figure 4, we vary the threshold to get the mask and see what works best.The approach we use for determining a suitable threshold is to calculate the average of the

(a) Gaussian2D and Gaussian1D with σ=1 & size = 3

(b) Gaussian2d with σ=1 & size = 3 and simple filter

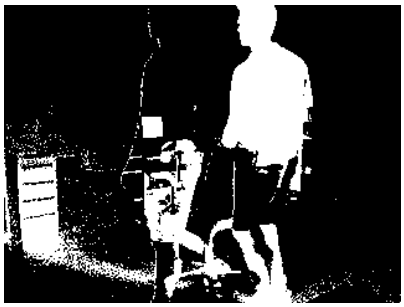
(c) Gaussian2d with σ=2 & size = 3 and simple filter

(d) Gaussian2d with σ=1 & size = 9 and simple filter

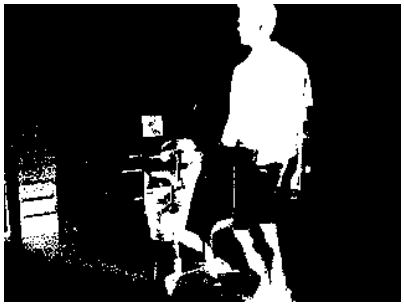
(e) Gaussian2d with σ=5 & size = 3 and simple filter

Figure 3. Gaussian2D for smoothing

(a)



(b) 3 times of standard deviation



(c) 5 times of standard deviation

Figure 4. Threshold

standard deviation of the temporal gradient for each pixel. Since the background is relatively constant and the moving objects occupy a relatively small part of each frame, we can treat the variation of these pixels as additive white Gaussian noise. This calculation can provide a reasonable estimate of the standard deviation of this noise distribution. Setting the threshold to three times of the standard deviation should be sufficient to ignore 99.7% of noise. After experiment, we select five times of the standard deviation as our threshold.

5. Appendix: Code

Please see our git repository for copies of our code. https://github.com/vegetablesB/EECE5639Project