Motion Detection Using Image Filtering

Miles, Lillian

miles.l@northeastern.edu

Shen, Matthew

shen.mat@northeastern.edu

Abstract

This project investigates motion detection using temporal intensity gradients in image sequences captured by a stationary camera. Motion is identified by detecting large temporal changes in pixel intensity. We implemented a simple 1D temporal derivative filter and a derivative-of-Gaussian (DoG) temporal filter with varying standard deviations. Additionally, we applied 2D spatial smoothing filters (3×3 and 5×5 box filters and Gaussian filters with varying standard deviations) prior to temporal processing. Adaptive thresholding was used to generate motion masks based on estimated background noise statistics. Experimental results show that moderate spatial Gaussian smoothing combined with a moderate temporal DoG filter produced the cleanest motion masks with minimal noise while preserving object boundaries.

I. Description of Algorithms

A. Temporal Derivative-Based Motion Detection

The motion detection framework in this study is based on the assumption that the camera remains stationary and that most of the scene consists of a static background. Under these conditions, the intensity of background pixels remains approximately constant over time, while pixels corresponding to moving objects exhibit significant temporal changes. Therefore, motion can be detected by analyzing the temporal derivative of pixel intensity values.

The simplest temporal derivative was implemented using the discrete filter 0.5[-1, 0, 1]. The idea behind this is that for each pixel location (x,y), we can let I(x, y, t) represent its greyscale intensity at time t. An approximation of the temporal derivative can be implemented using the 1D filter:

$$\frac{(I(x,y,t+1) - I(x,y,t-1))}{2}$$

which corresponds to the discrete filter 0.5[-1, 0, 1] used across time, providing an estimate of the rate of intensity change at each pixel.

While the simple temporal derivative is computationally efficient, it is sensitive to noise and small intensity fluctuations. To improve robustness, a derivative-of-Gaussian (DoG) temporal filter was also implemented. Instead of directly computing the difference between adjacent frames, the DoG filter smooths intensity values over a temporal window and then computes the derivative. The temporal filter is defined as:

$$G'(t) = -\frac{t}{\sigma^2} e^{-\frac{t^2}{2\sigma^2}}$$

The σ, or tσ as used in the project, controls the temporal scale of smoothing, with different values causing different results, which will be explained in a later section. The temporal derivative at each pixel is obtained by convolving this 1D filter with intensity values across neighboring frames.

The temporal filters were implemented in Python using mathematical operations and separated into functions.

B. Spatial Smoothing

Temporal derivatives can be significantly affected by noise, so to reduce false detections caused by it, 2D spatial smoothing was applied to each grayscale frame before computing temporal derivatives.

Three types of spatial filters were evaluated:

- 3x3 box filter
- 5x5 box filter
- 2D Gaussian Filter with standard deviation sσ

The box filters replace each pixel with the average of its neighboring pixels within the specified window. The Gaussian filter performs weighted averaging, where nearby pixels contribute more strongly than distant ones, leading to blurring or smoothing. Changing the value of sσ leads to different results, which will be covered in a later section.

It's important to note that spatial smoothing was applied prior to temporal filtering. This ensures that noise is reduced before evaluating temporal intensity changes.

The spatial filters were applied in code by creating functions and using the OpenCV (cv2) library, namely blur and Gaussian blur.

C. Motion Mask Generation and Thresholding

After computing the temporal derivative, motion regions were identified by thresholding the magnitude of the temporal gradient. Pixels with large gradient magnitudes will correspond to significant temporal intensity changes, thus very likely belonging to moving objects.

Since most pixels correspond to a stationary background, their temporal gradients are close to zero and can be modeled as zero-mean Gaussian noise. The noise standard deviation was estimated using the median absolute deviation[5]. In this case, the MAD is:

$$MAD = median(\nabla_t I)$$

The MAD was used because the standard deviation is sensitive to outliers. Since the majority of pixels belong to the static background, their temporal gradients cluster near zero. There will be a small number of high-magnitude motion pixels, or outliers, which will cause the standard deviation to be less accurate. The MAD provides a better estimate that is largely unaffected by these outliers. For a zero-mean Gaussian distribution, the relationship we get is:

$$MAD = 0.67449\sigma$$

So the standard deviation can be calculated as:

$$\sigma = \frac{median(\nabla_t I)}{0.67449}$$

This was then incorporated into the code.

II. Experiments and Values of Parameters Used

To analyze the performance of the motion detection, experiments were conducted on image sequences given for the project. All frames were converted to grayscale prior to processing. The goal of the experiments was to analyze the effect of temporal filtering, spatial smoothing, and threshold selection on motion detection accuracy.

The DoG temporal filter was evaluated using three standard deviation values: 0.8, 1.5, and 3.0. These values were selected to represent increasing levels of temporal smoothing.

A value near 1 (0.8) will have minimal smoothing and emphasize rapid intensity changes. A moderate value (1.5) provides a balance of noise sensitivity and motion boundaries. A larger value (3.0) introduces strong temporal smoothing, reducing noise sensitivity but potentially attenuating fast or short-duration motion events. This range was chosen to examine the trade-off between motion sensitivity and noise suppression. If the temporal standard deviation is too large, motion would likely be over-smooth and not be informative.

As for spatial smoothing, the same values were chosen of 0.8, 1.5, and 3.0, for a similar reason; to explore light, moderate, and strong spatial smoothing. Small values preserve fine spatial detail but may allow noise to propagate into temporal gradients. Larger values provide greater noise suppression at the expense of edge sharpness and gradient magnitude. If the spatial standard deviation is too large, there would be too much blur.

The threshold multiplier used in experiments was 4.0. Multiple values of the threshold multiplier k were tested experimentally; when k was less than 3, there were many false positives due to background noise fluctuations. When k was greater than 5, there were less false detections, but there were less detections of smaller/slower-moving objects. K = 4 was a good balance of removing background noise but keeping motion regions.

III. Observations

A. Temporal Filtering

The performance of the motion detection algorithm was strongly influenced by the temporal standard deviation used in the DoG filter. Compared to the simple derivative filter $0.5[-1,0,1]$, the DoG filter with $t\sigma = 0.8$ produced visually equivalent results. The system was sensitive to motion and successfully detected fast-moving objects. This sensitivity came at the cost of increased background noise, as shown in Fig 1. Minor flickering intensity variations were classified as motion when there was a lack of more intense motion, so the results contained scattered noise, but the edges of the object in motion stayed sharp.

Increasing the temporal smoothing to $t\sigma = 1.5$ reduced noise while maintaining strong motion localization. Motion regions became a bit smoother, and fast motion was still reliably detected, however the object boundaries were less well defined.

For a large value of $t\sigma = 3.0$ noise was substantially reduced, as shown in Fig 1, but attenuated short-duration motion events. Boundaries became less precise, and some fast-moving objects were partially missed. Increasing $t\sigma$ improved noise suppression but reduced responsiveness to fast or subtle motion. Figures 2-4 show the results of each type of filter on different frames.

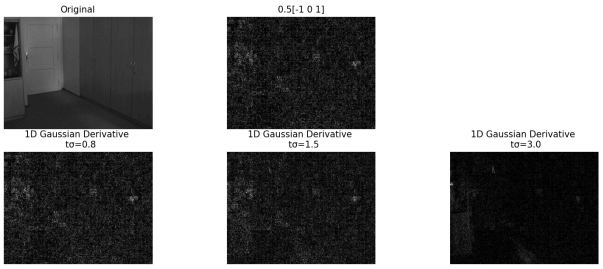

Figure 1: Demonstration of Noise with Temporal Filters

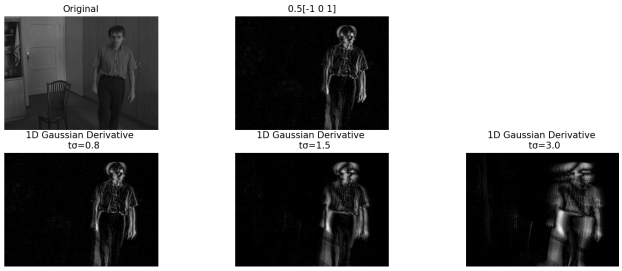

Figure 2: Temporal Derivative Comparison

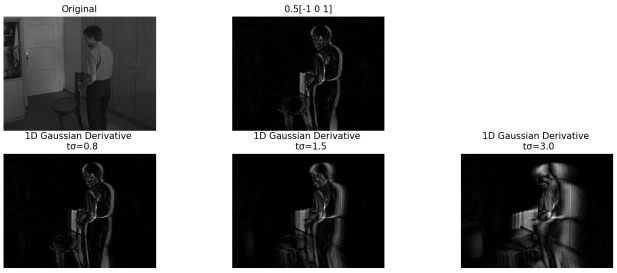

Figure 3: Temporal Derivative Comparison

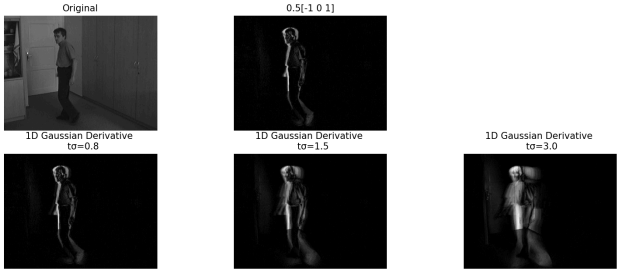

Figure 4: Temporal Derivative Comparison

B. Spatial Smoothing Observations

Spatial Gaussian smoothing also had a significant effect on motion detection performance. Part two results feature each combination of spatial and temporal filters in a single 24 image grid for comparison. The grid of results clearly shows that as tσ increases (from top to bottom), noise suppression is improved, and as sσ increases (from left to right), blur and smoothing for each image is increased.

With the 3x3 and 5x5 box filters it is evident that the smaller filter results in a lot of speckle noise, while the 5x5 filter results in less frequent but more intense noise. The 3x3 box filter did however, produce clear motion regions with well-defined object edges when there were significant objects in motion.

A small spatial standard deviation of sσ = 0.8, preserved edges, but noise was not well suppressed before temporal differentiation. The results again exhibited speckle noise and were visually equivalent to the results for the 3x3 box filter. This filter also produced clear motion regions with well-defined object edges.

Increasing the smoothing to sσ = 1.5 led to reduction in noise. This setting preserved important structural information while improving mask coherence, resulting in visually clean and reliable motion detection.

However, when the spatial standard deviation was increased further to sσ = 3.0, there was excessive blurring, which obscured edges. These results, as shown in figures 5-9, indicate that moderate spatial smoothing or a 3x3 box filter provide the best compromise between noise suppression and preservation of motion detail, while excessive smoothing degrades detection performance.

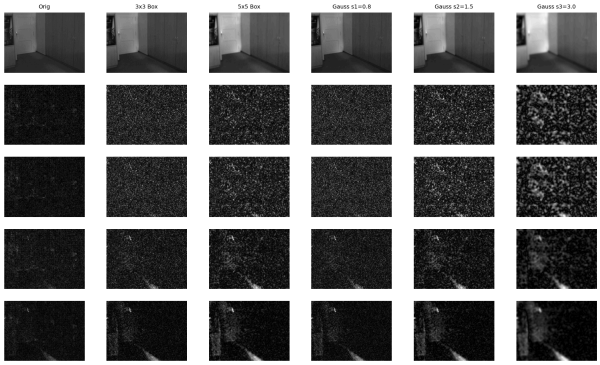

Figure 5: Demonstration of Noise with Spatial Smoothing Filter

Figure 6: Spatial Smoothing Comparison Results

Figure 7: Spatial Smoothing Comparison Results

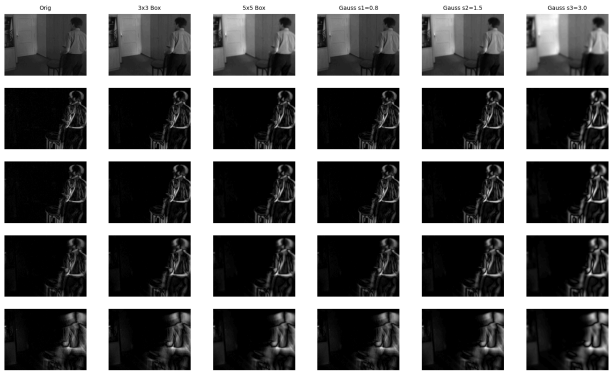

Figure 8: Spatial Smoothing Comparison Results

Figure 9: Spatial Smoothing Comparison Results

C. Part Three

The dynamic thresholding strategy using a Median Absolute Deviation (MAD) approach demonstrated good performance. The threshold multiplier k critically affected detection quality.

For low values ($k<3$) the algorithm became overly sensitive, resulting in a high false positive rate and cluttered motion masks. Much of the background noise was classified as motion. Large values ($k>5$) suppressed noise effectively but weakened the motion response, causing slow or small movements to be missed. A moderate value of $k=4$ provided the best tradeoff between sensitivity and false alarm suppression. At this setting, motion masks were clear and continuous, and background speckle noise was minimal while still maintaining reliable detection.

Overall, the optimal configuration consisted of a 3x3 box filter or a 2D gaussian spatial smoothing filter with standard deviation $s\sigma$ with the simple 0.5[-1 0 1] temporal filter or a 1D DoG filter with standard deviation $t\sigma = 0.8$. These strategies were most effective in combination with adaptive thresholding to separate the edges of moving objects from background noise.

The results in Figures 10-14 show adaptive thresholding after using a 3x3 box filter and a simple temporal derivative (0.5[-1 0 1]).

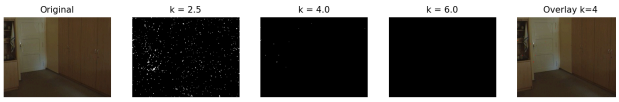

Figure 10: Demonstration of Noise in Thresholding and Overlay

Figure 11: Thresholding and Overlay Results

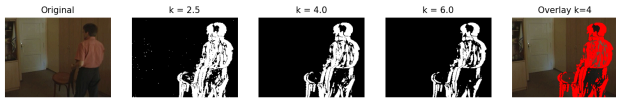

Figure 12: Thresholding and Overlay Results

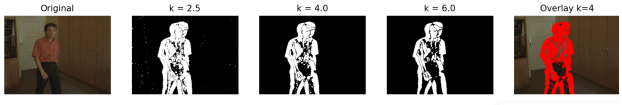

Figure 13: Thresholding and Overlay Results

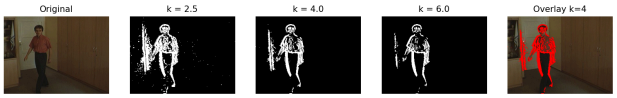

Figure 14: Thresholding and Overlay Results

IV. Conclusions

This project investigated motion detection through temporal intensity gradients in image sequences captured by a stationary camera. The results confirm that motion can be effectively detected by identifying significant temporal changes in pixel intensity, and quality of detection and noise suppression can be improved using appropriate spatial smoothing and thresholding strategies.

While the simple temporal derivative filter was more computationally efficient and capable of detecting rapid motion, it was sensitive to noise. The DoG temporal filter improved the system by incorporating temporal smoothing, but excessive smoothing reduced responsiveness to fast movement.

Applying spatial filtering prior to temporal differentiation reduced noise propagation. Small spatial standard deviations best preserved object boundaries but allowed speckle noise to remain. Larger spatial standard deviations introduced excessive blur and obscured the edges of moving objects.

The adaptive thresholding strategy based on the Median Absolute Deviation proved effective for separating motion from background noise. Experimental evaluation showed that a threshold multiplier of k = 4 achieved a strong balance between minimizing false positives and maintaining sensitivity to motion.

The experiments demonstrate that motion detection performance depends strongly on the interaction between temporal filtering, spatial smoothing, and threshold selection. The best results were obtained from a 3x3 box filter or a 2D gaussian spatial smoothing filter with standard deviation $s\sigma$, in combination with the simple 0.5[-1 0 1] temporal filter or a DoG filter with standard deviation $t\sigma = 0.8$. These strategies were most effective in combination with adaptive thresholding to separate the edges of moving objects from background noise. The study highlights the importance of tuning parameters to balance sensitivity and noise suppression in motion detection systems.

References

1. "Python: Grayscaling of images using OpenCV," *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/python/python-grayscaling-of-images-using-opencv/
2. "Matplotlib documentation," *Matplotlib*. [Online]. Available: https://matplotlib.org/stable/index.html
3. "Render Special Characters," *Visual Studio Code Marketplace*. [Online]. Available: https://marketplace.visualstudio.com/items?itemName=miku3920.vscode-render-special-chars
4. "Gaussian derivatives," *Hannibunny Object Recognition Book*. [Online]. Available: https://hannibunny.github.io/orbook/preprocessing/04gaussianDerivatives.html
5. H. Khaleel, "Estimation of noise in gray-scale and colored images using Median Absolute Deviation (MAD)," *ResearchGate*, Chapter 15. [Online]. Available: https://www.researchgate.net/profile/Hasan-Khaleel/publication/4355619_Chapter_15_Estimation_of_Noise_in_Gray-Scale_and_Colored_Images_Using_Median_Absolute_Deviation_MAD/links/00b7d53587bbf96ece000000/Chapter-15-Estimation-of-Noise-in-Gray-Scale-and-Colored-Images-Using-Median-Absolute-Deviation-MAD.pdf
6. "Thresholding strategy of estimating noise level," *Emergent Mind*. [Online]. Available: https://www.emergentmind.com/topics/thresholding-strategy-of-estimating-noise-level