Motion Detection Using Simple Image Filtering

Yash Mewada, Pratik Baldota

Project 1 Submission

Submitted in partial fulfillment of the requirements for the course of ***EECE5639***

Feb 20, 2023

Northeastern University

***Abstract*—There are several algorithms used for motion detection. One of them was covered in this project. The goal of this project is to detect motion in image sequences taken by a stationary camera. The vast majority of the pixels in the sequence are from a stationary background, with only a few moving objects in front of the camera. To detect moving objects, the technique analyses the temporal evolution of pixel values. The program reads in a series of grayscale image frames and computes the temporal derivative by applying a 1-D differential operator to each pixel. The absolute values of the derivatives are then threshold to produce a 0 and 1 moving object mask. The results are then displayed by combining the mask with the original frame. Exploring variations on the above basic outline is one of the project requirements. For example, the program can compare the results of various temporal derivative filters. To improve the results, we can also apply a 2D spatial smoothing filter before the temporal derivative filter. To optimize the mask, the threshold can also be changed. It is necessary to submit a report that details the algorithms, experiments, parameter values, and observations. The report should include representative images of both "good" and "bad" results. An appendix contains the code.**

***Index Terms*—Gaussian filters, Spatial Smoothing filters, Temporal Derivatives.**

I. Introduction and Motivation

A. Background

The main idea behind motion detection is to find the difference between two consecutive frames of the camera and perform filtering on top of it to detect the change in pixel intensity. This goal was briefly covered in this project.

B. Approach and description of algorithms

We explored a total of 4 algorithms in this project. They are stated below.

1) Grayscale Conversion
2) Spatial Smoothing
3) Temporal Derivative of frames with different filters
4) Estimating the noise in the image for thresholding selection.

II. Introduction and Motivation

A. Grayscale

To make the computation timeless the input images were first converted to grayscale. This states that a value where all the channels of pixels in an image are equal, making them a Grayscale image.

B. Spatial Smoothing

Spatial smoothing algorithms are separable algorithms that are mainly used for smooth images. The reason they were explored in this project was that as this focused on motion detection based on pixel intensity, it also changed the intensity of surrounding pixels causing the false detection of motion. The possible background noise of original frames could have significant negative effects on accurate motion detection. And thus a 2D spatial smoothing filter is considered to be applied to the frames before applying the temporal derivative filter. Here we tried 3×3, 5×5 box filters, and $2D$ Gaussian filters with different standard deviation σ. We tried the value of σ as 1.0,2.0 and 3.0 for testing purposes.

C. Temporal Derivative of frames

Here temporal derivative i.e subtracting two consecutive image frames results in first-order differentiation of the image frames. Since the temporal derivatives of each pixel are to be analyzed for large gradients, the different temporal derivative filter has different corresponding result.
After performing the temporal derivative, a $0.5[-1, 0, 1]$ 1D filter was convoluted on top of it to detect the gradient in the images. The same process was done with a 1D differentiation of the Gaussian filter.

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

$$\frac{d}{dx} g(x) = \frac{1}{\sqrt{2\pi}\sigma} \left(-\frac{x}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} g(x) \tag{2}$$

$$\frac{d}{dx} g(x) = -\frac{x}{\sigma^2} g(x) \tag{3}$$

D. Estimating noise for threshold selection

We learned in the previous lecture about the algorithms used for estimating the noise in an image. We used this algorithm in this project to estimate the noise in temporal derivative images. After temporal derivatives are performed, they still contained some grayscale values, hence a mask must be created using this temporal derivative by thresholding the pixel values of the derivative of images. One common approach is to use a multiple of the standard deviation of the noise as the threshold. A commonly used value is 3 times the standard deviation, which corresponds to the range that includes approximately 99.7 of the noise values if the noise is normally distributed .

We found that the images had noise of around 3. Hence we choose threshold 9 to filter out the false detection of motion.

III. Values of parameters used

As mentioned in the above section, the most crucial parameter that needed to be fine-tuned was threshold selection. We tried different experimental values but they somewhere failed to work for all cases.
Also as the general case was $3 * \sigma$ for threshold, in our case it resulted in 5 times.

IV. Output

A. Output of applying the smoothing filters

Fig. 1. 3X3 Smoothing filter

Fig. 2. 5x5 Smoothing filter

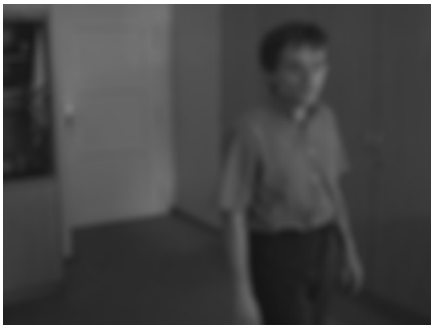

Fig. 3. Gaussian Smoothing filter

From the above output, it is visible that smoothing filters smooth the details in images and we will see further that this smoothing is not good for better motion detection in the images as they blur out the details in the images. Increasing the value of σ in the Gaussian filter smoothens the image.

B. Output of different temporal derivatives

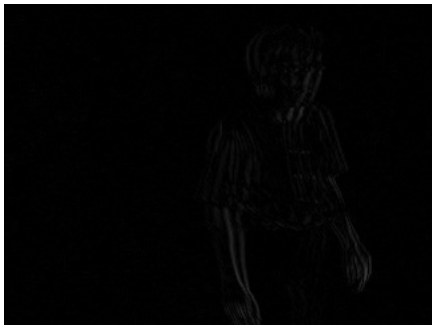

Fig. 4. 1D temporal derivative

Fig. 5. 1D Gaussian derivative

The above image might not be visible properly because the gaussian function blurred the image causing fewer edges/gradients to be detected. Furthermore, this was also tested after applying the smoothing filters. The output is as below. For

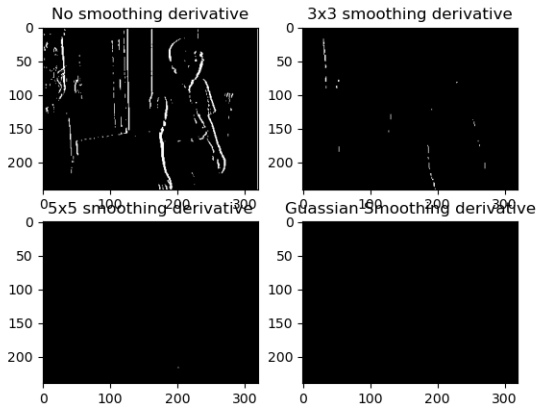

Fig. 6. smoothing derivative

experimental results we also performed direct subtraction of two consecutive images, although it is a drawback which is

explained in the later part of this report, we performed it. Although it looks the same as 4 but it is not we found that in

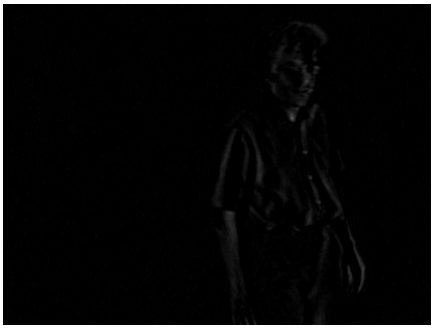

Fig. 7. Subtraction of two consecutive images

our analysis part.

C. Thresholding on basis of noise estimation and creating a binary mask

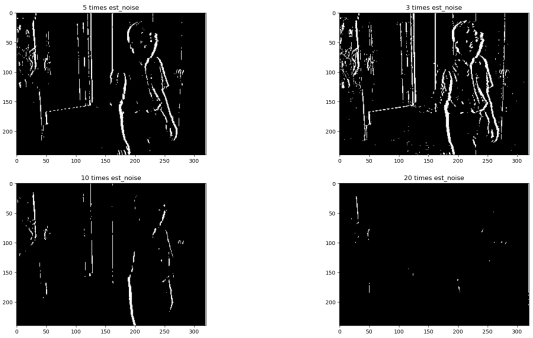

Fig. 8. Creating a binary mask based on thresholding the noise values

More detailed images are attached to this report. We tried different values for constant here to find the best threshold value.

D. Final detected output with different filters

Red and white color in the below images suggests motion detection based on the previous frame The reason Figure:

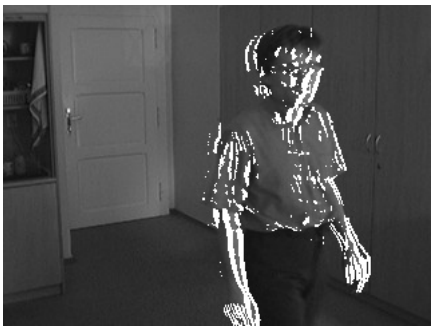

Fig. 9. Motion detection with 0.5*[-1,0,1] filter

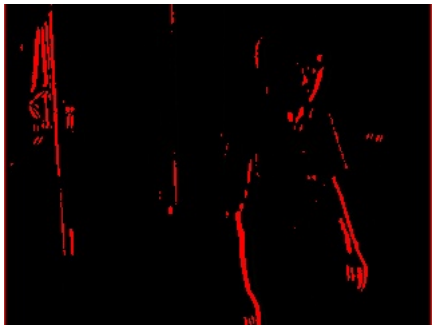

Fig. 10. Motion detection with 3x3 filter

10 is completely different from others is that while generating it, it was not multiplied with the original frame to generate the output. Also, the detected edges in it suggest that the threshold value did not work for this specific frame.

Fig. 11. Motion detection with 5x5 filter

Fig. 12. Motion detection with gaussian $\sigma = 2$ filter

*E. Comparing **Good** and **Bad** images*

As this project was focused only on motion detection, we will be quantifying only the output images. As from the above images, it is quite clear what caused the images with less motion detection. If we compare images 9, 10, 11, and 12 images we can see that 9 outperformed all the other types giving us the sweet spot between the threshold pixels and true detection. We can say that it is a good output in terms of motion detection.

Now the other images are also able to detect the motion but we

Fig. 13. Motion detection with gaussian $\sigma = 3$ filter

found that they are not able to detect relatively small motion due to the fact that they depend on **smoothing filters** that smoothens the edges in the images making it harder to detect the motion.

V. Observations and conclusions

We mentioned above that we also performed image subtraction as experimental data. We found that the temporal derivative also provides us with the **direction** and **speed of motion** which the image subtraction fails to do so.

The temporal derivative filter is sensitive to the direction of motion, as it produces positive and negative responses for motion in opposite directions. This can be useful for distinguishing between objects that are moving in different directions, or for detecting the direction of motion.

This approach can be used and is used in real-world applications like **Object Tracking, Motion Segmentation (what we did in the above section)** and **Video Stabilisation**.

Also, this is a very shallow implementation of the motion detection algorithm because here we are not considering the change created by the shadows in the intensity of the pixels which can cause more errors and false motion detection.