Edge Detection using Adaptive Mean Filtering

Tanmay Gulati
160905320
VI - C : 48

Shiven Sharma
160905386
VI - C : 60

Yash Raj Sarrof
160905390
VI - C : 61

Abstract

In simple thresholding, the threshold value is global, i.e., it is same for all the pixels in the image. Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions. The Wiener filtering executes an optimal tradeoff between inverse filtering and noise smoothing. It removes the additive noise and inverts the blurring simultaneously

Objective

Parallelize the Adaptive edge detection process done using Weiner filter, demonstrate in both CUDA and MPI. The main motivation is to remove noise from the image, then apply any of the commonly known filters to detect images. We apply sobel filter to find horizontal edges. Although the only difference to the algorithm to find vertical edges, would be to take the transpose of the filter and use that as the base filter instead.

Introduction

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convolving the image with an operator (a 2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions. Variables involved in the selection of an edge detection operator includes :

- Edge orientation (horizontal/vertical)
- Amount of Noise in environment
- Edge Structure (steep change in intensity at edge/gradual change in intensity)

Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels and Weiner Filter provides for one such solution.

Literature Review

The base paper for this project [1] takes a concept of signal processing i.e. adaptive mean filtering and applies that to a common application in image processing, which is edge detection. Adaptive mean filtering is standard technique of reducing noise in signal processing. In [2], various variations of mean filtering can be found, and thus different variations to this filtering method can be applied for this application as well. A performance analysis of such varied methods has been done comprehensively in [3]. This algorithm was one of the earliest developments in image processing, and more recent versions of Weiner filter can be found as well as in [4]. Also, parallel implementations of this algorithm have been done as well, like in [5], where LMS filter is used on some superscalar architecture.

Methodology

The following steps need to be followed to implement the algorithm. Parallelizing in CUDA is done at pixel level, however with MPI it is done at the row level, where each row concurrently gets executed. Since the number of threads required for a pixel level computation is not supported by MPI. Unless specified, all the steps are parallelized in the same manner as stated above.

- The image needs to be read into our programs for both CUDA and MPI, since we are using C as our base language, this is done through a publicly available library named **stb_image.h**. The library function returns the image in a 1D character array. We do all the necessary declarations here itself. It should be noted that this part of the algorithm cannot be parallelized in any way whatsover.

- Since the image is in character array, we need to cast all the elements to an integer, so that arithmetic operations can be performed on the images. The final 1D integer array is stored as **image**.

- The input image is padded with 1 layer of 0 intensity pixels. The same is done by having the whole image copied into a new 1-D array of length **(row+1)*(col+1)**, with the first, last column and row made as 0, others having intensity values of the original image with the mapping **image_pad[(i)*(cols+1) + j] = image[(i-1)*cols + (j-1)]**

- Compute the local means and local variances of the pixels of the input image, by considering a neighbourhood of 3*3, and store it in the 1D arrays of **localmean** and **localvariance**.

- Find the global variance of the image as a whole and store it in the variable **variance** Using which **localvariance** array is modified to be greater than at least the global variance, i.e. **variance**.

- Inverse adaptive mean filtering is applied to the image, with the following equation being applied to each and every pixel in the image, and output is stored in **filtered**.

$$filtered[i*cols+j] = image[i*cols+j] - \frac{variance}{localvariance[i*cols+j]} * (image[i*cols+j] - localmean[i*cols+j]) \tag{1}$$

- Convolve any filter meant to be used for edge detection to the padded version of the filtered image. For demonstration, we have used a horizontal sobel filter.

- Finally, again since we are using C, we need to save the image with the detected images. The same is done through a publicly licensed library **stb_image_write.h**. Again this can only save character arrays, thus the image is casted to a char, and the final image is saved as **final_image.h**.

Results

On an average the runtime of the overall algorithm is reduced by about 60-70% , while using CUDA, and about 25 - 30 % while using MPI. The difference is a result of the fact that we are not able to make as many threads in MPI as in CUDA. Thus whereas we make all the pixels of the image directly compute steps on it's own in CUDA. In MPI only the rows or the columns of the image can be parallelized.
A sample input in figure 1 and corresponding output in figure 2 is shown here. Thus the algorithm definitely works well for images with no noise.

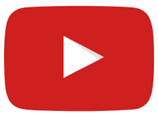

Figure 1: Clean Input Image

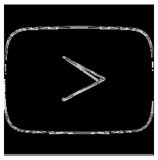

Figure 2: Output Image with horizontal edges

However looking at figure 3 , the algorithm manages to detect edges and higlights even in highly noisy images as can be seen in figure 4. Thus if edge detection is part of a preprocessing step, it would help a lot of applications, in the sense that the immense processing power of a GPU could be used for more advanced applications.

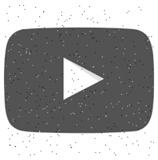

Figure 3: Input Image with noise

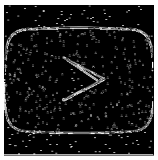

Figure 4: Output Image

Limitations and Possible Improvements

For the purposes of this project the filter size has not been tamparerd, varying filter sizes could provide for better results. Also, the accuracy of the algorithm severely depends on the function taken for making the image more immune to the noise and is now, probably too outdated to be used in mainstream applications. Especially with the boom of deep learning and the excellent results achieved with convolution neural networks, it would make better sense to use that for any image processing applications with GPU and fast processing available at hand, since the potential trade off achieved in favour of time, does not justify using this algorithm anymore. Another area of improvement could come up in adding more feature engineering, and improving the algorithm to parallelize further.

Conclusion

Adaptive Mean filtering, and using that to negate the effect of noise although works well, but with other alternatives opening up to negate noise, it has almost lost relevance, and in case the noise is not all that much, then using a fixed filter makes much more sense, since the computations are far lesser. However, it can still be used as part of preprocessing.

References

[1] M. Woodhall and C. Lindquist, "New edge detection algorithms based on adaptive estimation filters," in *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No. 97CB36136)*, vol. 2, pp. 1695–1699, IEEE, 1997.

[2] S. Weib, *On adaptive filtering on oversampled subbands.* PhD thesis, Ph. D. thesis, Signal Processing Division, University of Strathclyde, Glasgow, 1998.

[3] R. Bitmead and B. Anderson, "Performance of adaptive estimation algorithms in dependent random environments," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 788–794, 1980.

[4] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction wiener filter," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1218–1234, 2006.

[5] D. Allred, V. Krishnan, W. Huang, and D. Anderson, "Implementation of an lms adaptive filter on an fpga employing multiplexed multiplier architecture," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 1, pp. 918–921, IEEE, 2003.