**Two Dimensional LMS Adaptive Filter**Devon Quaternik

ELEN 431

December 7, 2016

**Abstract**

The LMS gradient approximation appears in many 1 dimensional applications, but these do not work well in 2 dimensional settings. Part of this has to do with the stationarity of the signal, the other with neighborhood considerations of images. 1D tend to ignore correlations in one direction, which for image data is naïve. The two dimensional LMS, recreated in MATLAB, is able to better account for the statistics of an image. The results show the recreated TDLMS for a two dimensional adaptive line enhancer (TDALE).

**Introduction**

When dealing with images, the two most common problems are with regards to compression and enhancement. One dimensional approaches tend to focus on stationary signals, something that an image is inherently not. They also tended to introduce blurring or gray-level effects that would harm the integrity of the image. The TDLMS is able to take into account for local statistics of images, and use them to form to the weights. It takes in your noisy signal and a desired one, as well as a few setup parameters, such as the step and filter size. It does not require any a priori knowledge about the image, and can be demonstrated to improve as it moves throughout a given image. It is an extension of the one dimensional Weiner solution, which is seen in the update equations. When used in image enhancement, the desired signal and the input are the same, separated by a delay. There are limitations to this algorithm, though. Being that a proper Weiner solution would require an infinite data set, this algorithm never reaches the Weiner solution. It also requires a proper step update size to properly track. Too small and local data will be missed, causing blur; too large and intensity values can become distorted. There are also issues of initial weights. While this will work for any given set of weights, there are better choices. Most try to keep all the weights summed to one, to ensure the local intensity average is preserved. In the following sections, I will cover the methods I used in this recreation, as well as the results I was able to obtain.

**Methods**

The TDLMS, like the original LMS, uses a gradient estimate to find the Weiner solution to a two dimensional filter. It then follows the method of steepest descent in progressing through the gradient to minimize the Mean Square Error. To form its output, it takes a neighborhood of values from the input image, point multiplies them by the weights, and sums over the entirety. This output is subtracted from the desired to get the error, which is used to update the weights.

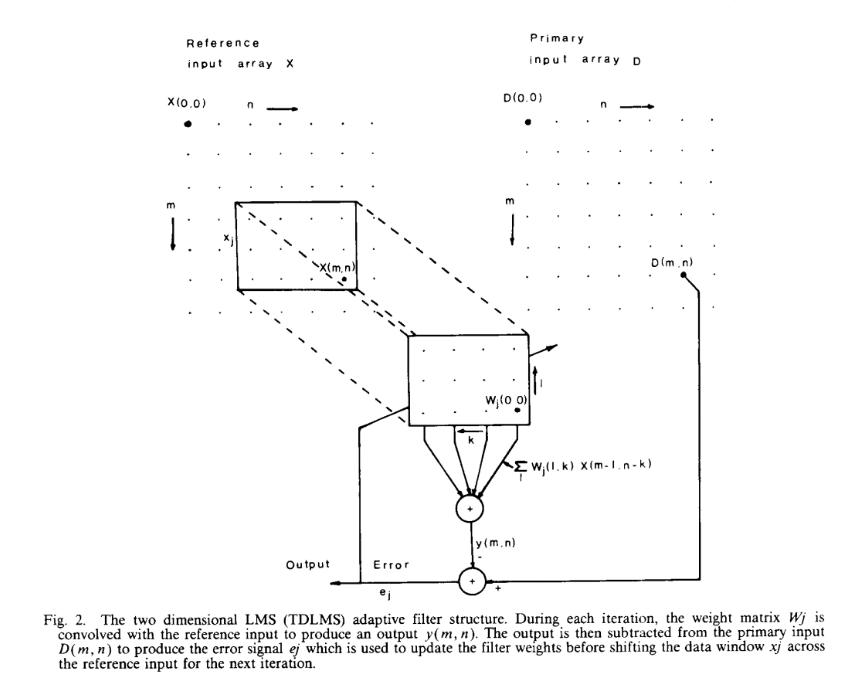


Figure 1: The structure of the 2D LMS adaptive filter. Right: Equations for output, error, and next weight.

I implemented the equations in the ‘TWDLMS.m’ MATLAB file based on Figure 1 above. The method of updating the regressor requires the appending of two rows and two columns of zeros to ensure the proper function. The function should work with any uniform boundary applied, as it would only affect one row and one column of pixels, but zeros were chosen to mitigate the effects. The selected also will have a large effect on the image. This is illustrated below in the output images. For small values of , you are not able to see much effect. The larger shows signs of overstepping, as the gray levels are inconsistent. This does not necessarily make it a bad choice, though, as the choice of will depend on the image set as well as the noise involved. It does have some slight effects when run on a normal image, but it is only a slight degradation. These are demonstrated in the cross sections of the original and the final image.

In selecting initial weights, you also should consider your noise and images. Because this algorithm improves as you move through the image, the initial effects of your filter will be seen much more clearly early on. For instance, when starting with all zeros, the image will start off very dark, and as the algorithm progresses it will become lighter to match the original. Similarly, if started with an averaging filter, the beginning of your output will appear blurry compared to the rest. The original paper tried to account for this by ensuring the values of all weights summed to one, preserving the average intensity of the image. I was unable to recreate this portion, causing some values of my image to go above one or below 0.

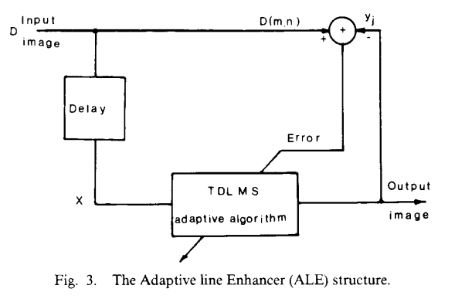


Figure : The 2D Adaptive Line Enhancer structure

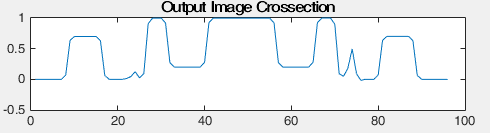
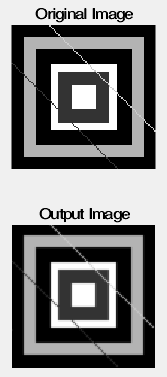
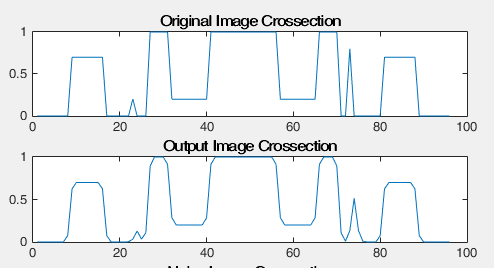
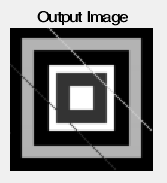
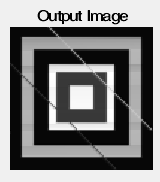
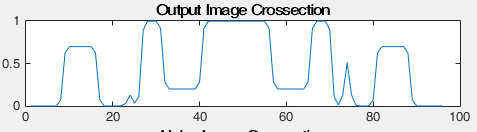
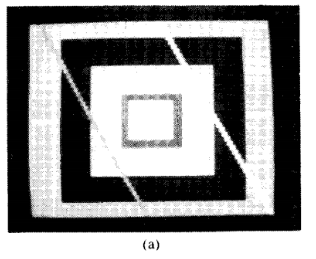
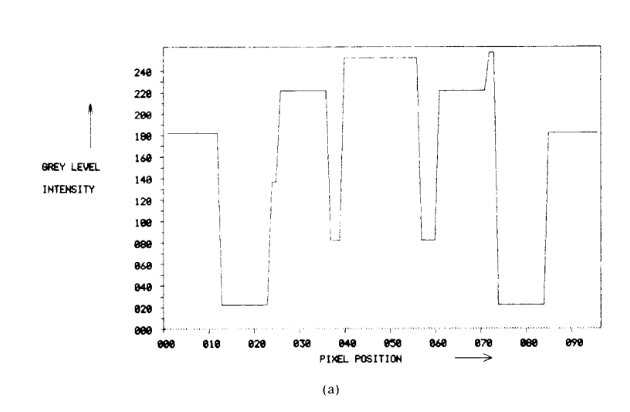
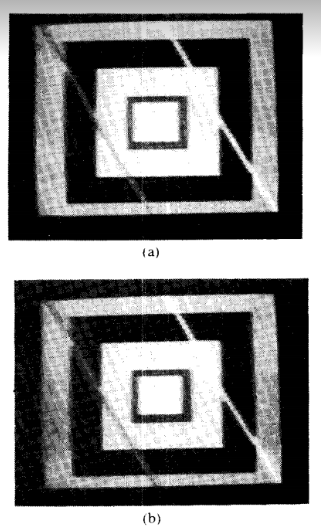
********The ‘TDALE.m’ has the MATLAB script that generated the test image as well as ran the overarching image enhancer. It is set up to run an ensemble of the TDLMS and display the results it gives back for the final run. It saves every set of error images as well as coefficients for viewing later. It was modeled after the block diagram given in the original paper, which I have attached in Figure 2. As such, it needed a delay for the image. If you submit the same image as desired and the input for the TDLMS, the weights do not update because the error is always zero. The delay prevents this from happening, and forces the weights to update. To implement the delay, a 0 was added at pixel position (1,1) and each pixel was pushed one place to the right. Pixels at the end of a row would be placed in the first position of the next row. The last pixel is destroyed, but in a large enough image this should be negligible in effect. The TDALE only taking the noisy image gives it its utility. In most real world situations, you do not have a restored image to base off of.

Figure : Left: Top: Original Image Bottom: Output with mu = 35\*10^-9. Middle: Top: Output with mu = 35\*10^-7 Bottom: Output with mu = 35\*10^-5. Right: Crossection across middle column for each image respective to listing order. X axis is pixel position, Y is gray value.

Figure 4: Left: Top: The original test image. Bottom: The cross section of the original test image. Right: Top: The effects with mu = 70\*10^-9. Bottom: The effects with mu = 35\*10^-9

**Results**

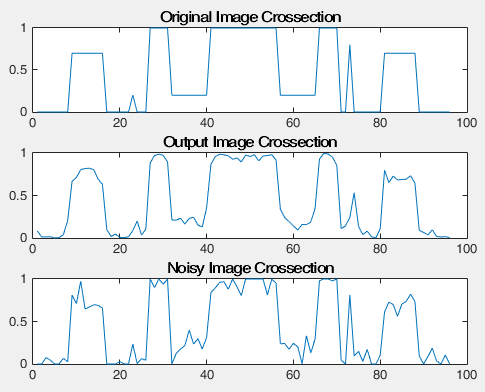
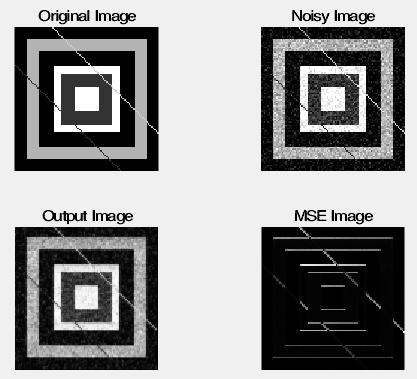
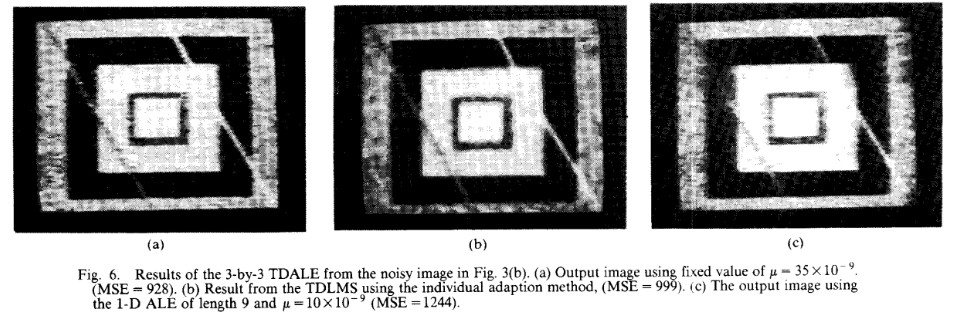
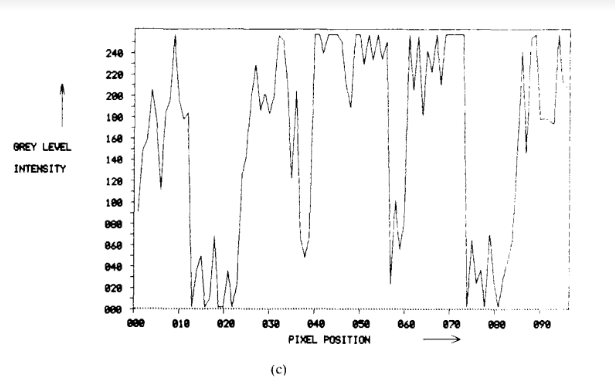
I was able to recreate the TDLMS and TDALE with some limited success. I had to use a much higher value of than the paper, but this could be due to differences in test images. When using the smaller values, it seemed my weights did not have enough time to update. This probably goes back to not having a real ‘desired’ image, but only simulated through delay. The algorithm does help cut noise, as is objectively seen in the cross sections. One interesting thing I found was in the MSE average image. You could see the lines getting darker as the algorithm progressed through the image. This implies the MSE is getting smaller as 0 is black in MATLAB. This shows the improvement the TDLMS was having. The effects of the TDLMS are explored on a non-noisy image in figure 3. It is shown in the cross section that when there is no noise, the filter is biased toward its initial condition. In this case I initialized as a Gaussian filter, and this is reflected in the shapes. Each different case shows different values of mu. Figure 4 shows the original papers results for similar cases

Figure 5: Left: Results of TDALE with TDLMS mu = 35\*10^-5. Right: The cross section of the middle of each image, except the MSE.

In Figure 5 I show the results for a noisy image with one value of mu. You can see from the MSE the improvement as the filter progresses. The cross-sectional images demonstrate the reduction in noise. It is interesting to note how you can see the Gaussian nature at the beginning, and it slowly stops the further along you go in the cross section. Figure 6 shows the original papers results for different selections of mu. The cross sections also shows reduction in noise and a tendency to look less and less like its initial filter. Overall the functionality of the algorithm shines through, even for less than ideal situations.





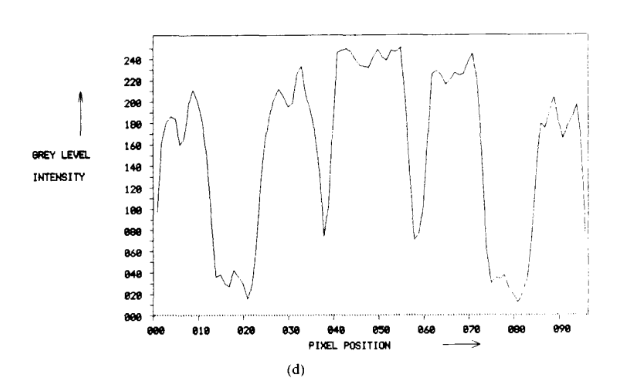


Figure : Top: The results of using a noisy image. Middle: The cross section of the original noisy image. Bottom: The cross section of the middle output

**References**

Diniz, Paulo Sergio Ramirez, Da Silva Eduardo A. B., and Sergio L. Netto. *Digital Signal Processing: System Analysis and Design*. Cambridge, U.K.: Cambridge UP, 2002. Print.

Hadhoud, M.m., and D.w. Thomas. "The Two-dimensional Adaptive LMS (TDLMS) Algorithm." *IEEE Transactions on Circuits and Systems* 35.5 (1988): 485-94. Web.