Determining Line Positions in Scanning Electron Microscope Images of Extreme Ultra Violet Lithography Resist Test Patterns

David Idell, Leonid Yankulin

Abstract

Line Edge Roughness (LER) is one of the parameters used to qualify resists for production in optical lithography. Generally, LER above 5% of the Critical Dimension (CD) is not tolerated. In this paper we show how we created the model, prepared the data for our simplified model, and how we used the Bayesian Method with the Markov Chain Monte Carlo sampling utilized by the Metropolis-Hastings algorithm to determine the defining parameters of Scanning Electron Microscope (SEM) images.

Introduction

Lithography is critical in semiconductor manufacturing, and must be constantly improved to keep pace with Moore's law. One promising successor to the current technology is EUV lithography; however an acceptable LER must be achieved for commercial viability.

Currently, LER analysis requires scientists to progress through each SEM image taken and click their mouse on a computer program to determine line location. The program is then able calculate the location of the remaining lines. Since the LER is often plotted as a function of dose, focus, and resist, it is often necessary to analyze thousands of images before a correlation can be determined. In practice, this equates to days of lost productivity.

Model

Our method was to use Matlab to accomplish the task of analyzing an SEM image to automatically obtain the line slopes, x-offset of the line grouping, the line widths, the spacing between lines, the number of lines, the color of the lines, and the color of the space between them, on the grayscale.

First, it was necessary to create a forward model for the Bayesian problem. The model accepted the seven parameters and produced a matrix of integer values representing pixels and their grayscale color. The images recorded by the SEM are 512x512 resolution, however it took Matlab .2 seconds to produce a model. By reducing our model size to 51x51, we had 100 times left pixels, and were able to reduce the time per model to .015 seconds. The model also treated the image as only being two colors, one for the line, and one for the black. This does not take into account the noise, LER, variation in line thickness and space thickness, and the true color of the line which varies as a function of the distance from the edge.

Data

The initial approach used the full 512x512 pixel image generated by the SEM. However, each Log Posterior calculation took over .2 seconds, so the data was down-sampled to 51x51 reducing the Log Posterior calculation to .03 seconds. We then touched up the down-sampled image to reduce the nuisance parameters that were not included in the model.

Bayesian Analysis

The core of the automatic image analysis problem is Bayesian probability, which uses data and prior probabilities to generate a probability density function. The general Bayesian equation is given below:

(1) $$prob(hypothesis \mid data, I) \propto prob(data \mid hypothesis, I) \times prob(hypothesis \mid I)$$

Given the quality SEM images, the data used in the algorithm will generally have Gaussian noise present. Because of this noise, a Student-T distribution is used when comparing the forward model to the data. The equation for the log of the probability using a Student-T distribution is shown below

(2) $$\log P = -\frac{N}{2} \log \sum_{i=1}^{N} (F_i - D_i)^2 + C$$

P is the probability associated with the forward model, F. D is the data (the input image matrix). N is the number of data points in the data.

Results

The algorithm was run on the original downsampled data, and it was able to determine the location of the first line correctly. However, since the model was unable to cope with the negative x-offset, the pixilation, and variation of colors, and the varying thickness of the lines and spaces, the remaining lines were shifted, and all the colors were not true colors.

The algorithm was then used on the optimized data, and it produced perfect results after many iterations on a simple model. The down-sampled original and optimized data used and the corresponding solutions produced by our algorithm are shown below:

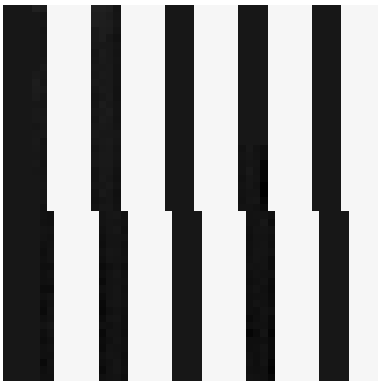

Figure 1 - Optimized data (51x51 px)

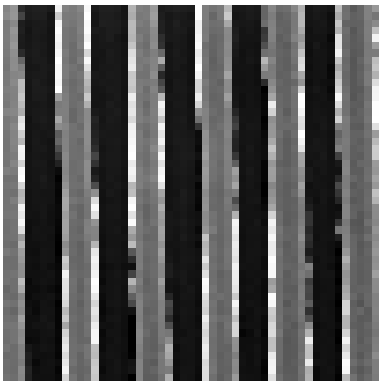

Figure 2 - Down-sampled data (51x51 px)

The following graphs show the progress of the seven parameters across five thousand iterations. The graphs are shown side-by-side to enunciate the differences in the behavior of the algorithm when dealing with optimal data and real data. Each graph contains thirty lines, representing the values of the thirty samples throughout the iterations.

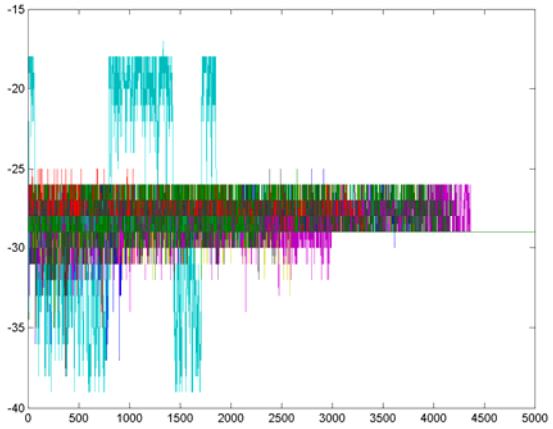

Figure 3 - slope - optimal data

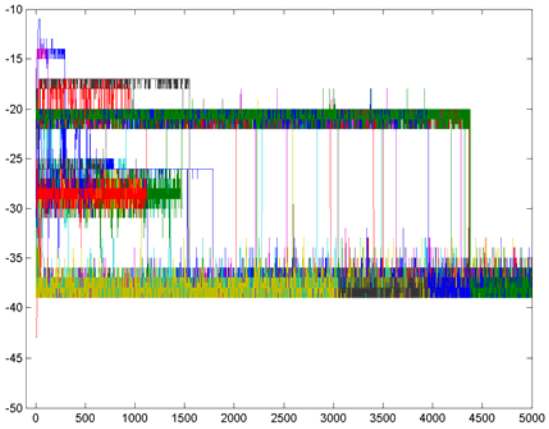

Figure 4 - slope - down-sampled data

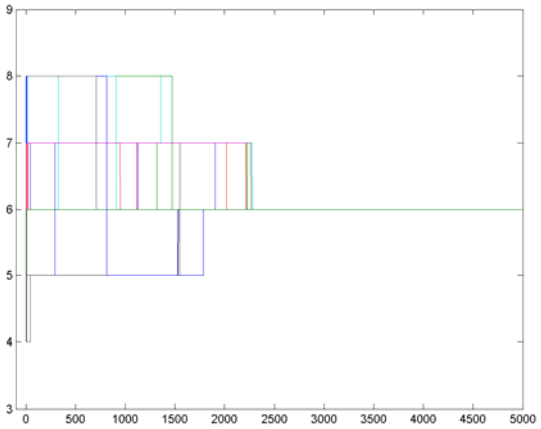

Figure 5 – x-offset – optimal data

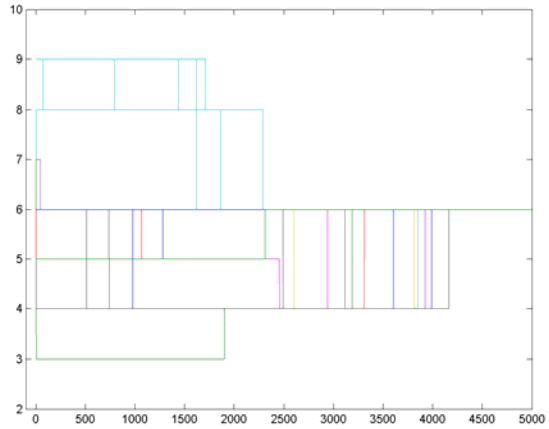

Figure 6 – x-offset – down-sampled data

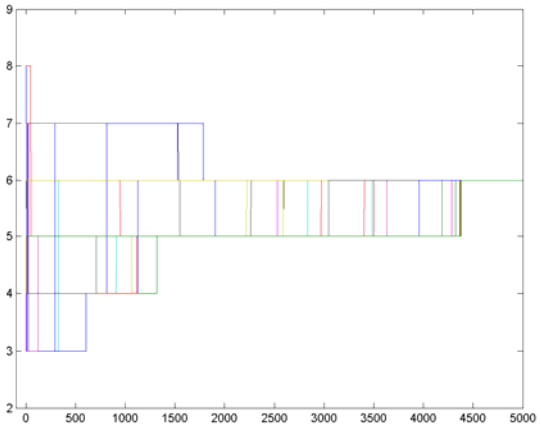

Figure 7 – line width – optimal data

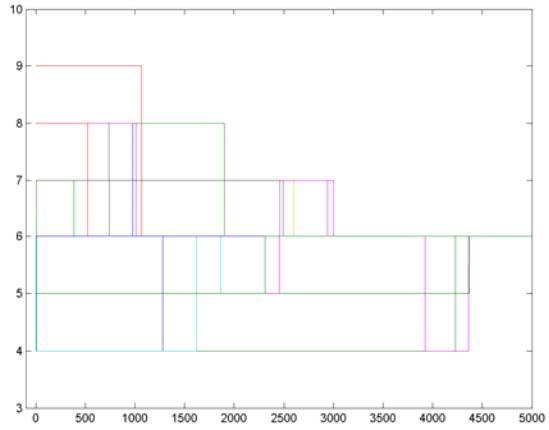

Figure 8 – line width – down-sampled data

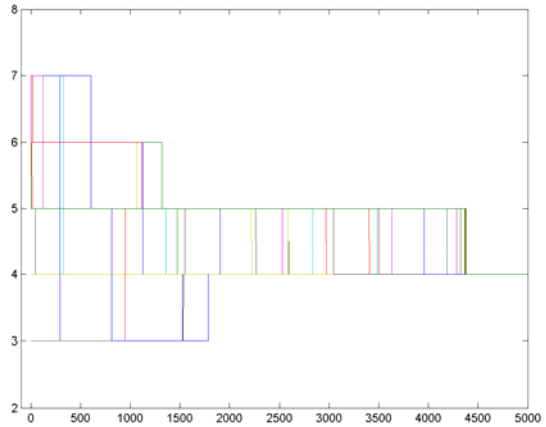

Figure 9 – space width – optimal data

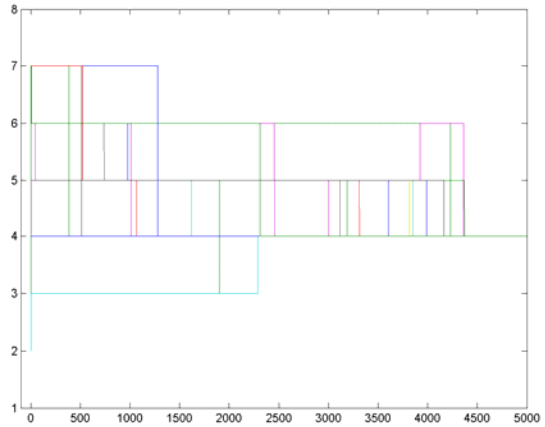

Figure 10 – space width – down-sampled data

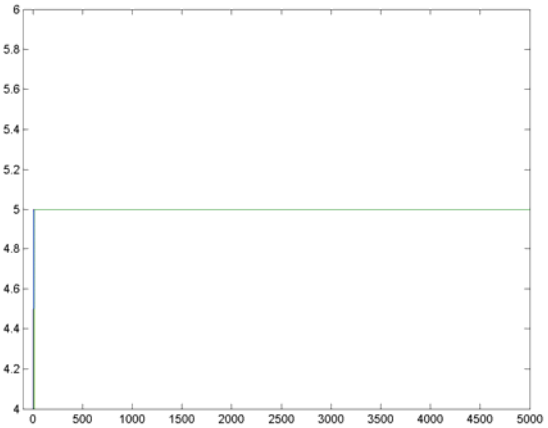

Figure 11 - # lines – optimal data

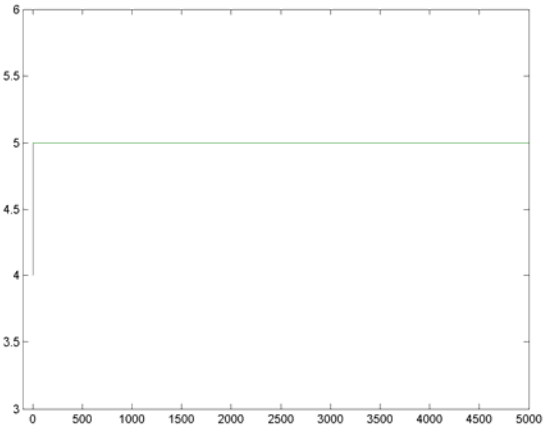

Figure 12 - # lines – down-sampled data

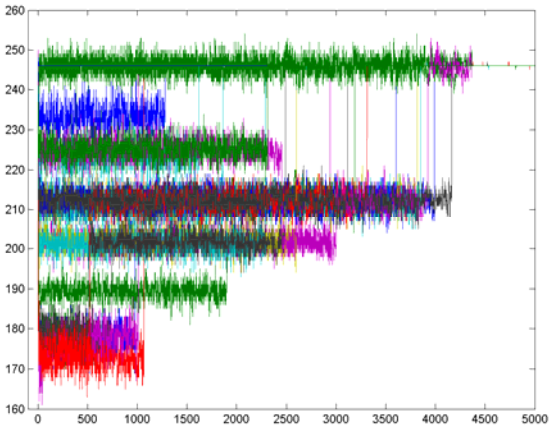

Figure 13 – white value – optimal data

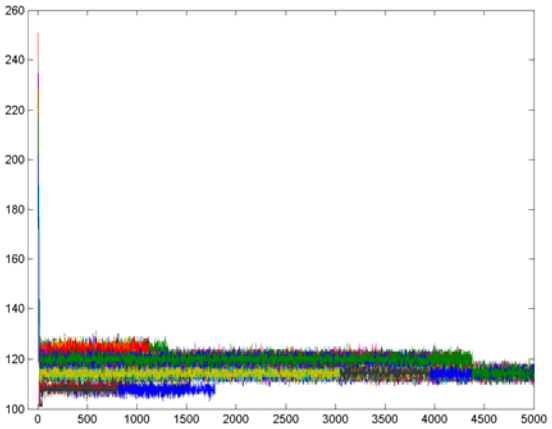

Figure 14 – white value – down-sampled data

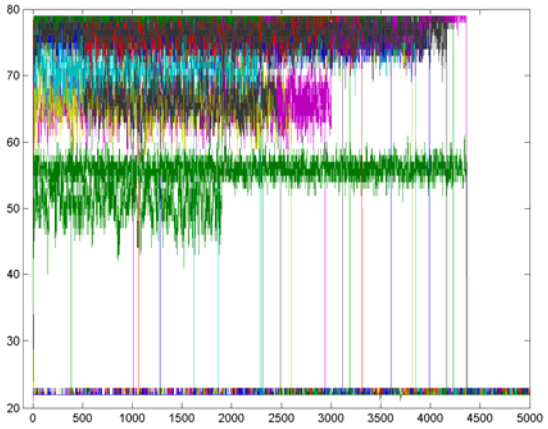

Figure 15 – black value – optimal data

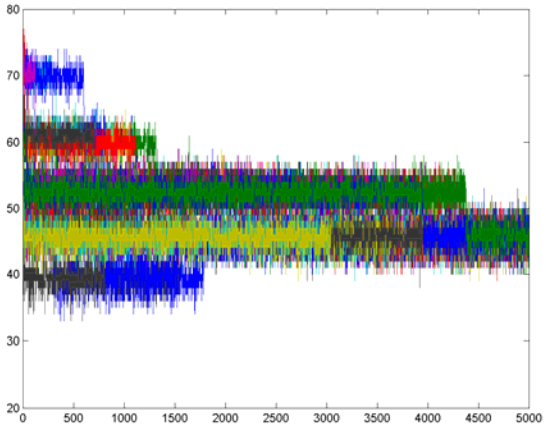

Figure 16 – white value – down-sampled data

From the comparisons of the progress in both cases for each parameter, one can observe that the algorithm will converge faster and more accurately for the optimal data than for the down-sampled data, as one would expect.

Conclusions and Future Work

There are many special considerations regarding the use of the MCMC algorithm for an image analysis problem. For example, one must consider the theoretical possibility of lines with an essentially infinite slope, causing the algorithm to never be "satisfied" with any one slope, and hence causing it to never fully converge on an answer. There are also special cases concerning identical probability peaks in the answer space. Such peaks occur when two different models have the same probability of being the best result. This can happen in a dual-color image when, for example, white lines with a given spacing look identical to black lines with that spacing as a line width, and have white spacing of the white line width.

Another consideration would be to automatically pre-determine the black and white values of the model before running the algorithm. This might be done using a dual-Gaussian fit to the color histogram data, which is shown in Figure 17.

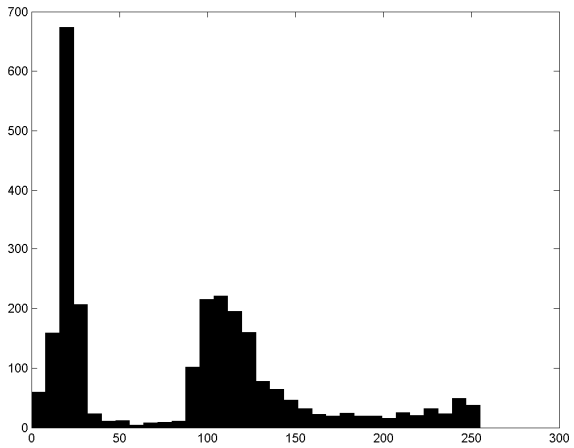

Figure 17 – Histogram grayscale (1-256) color data for the down-sampled image

From the double fitting, the resulting black and white values can be passed into the MCMC algorithm and used whenever a log probability is calculated. This process will speed up the algorithm and make it more accurate over a lesser number of iterations.

References

D.S.Sivia, J.Skilling. *Data Analysis – A Bayesian Tutorial.* New York: Oxford University Press, 2006.

Kevin Knuth's Matlab MCMC algorithm

I CSI451/551, A PHY 451/551, *Bayesian Data Analysis and Signal Processing,* class notes