A NEURAL-NET BASED, IN-LINE FOCUS/EXPOSURE MONITOR

by

Pamela M. Tsai

Memorandum No. UCB/ERL M94/55

28 July 1994

A NEURAL-NET BASED, IN-LINE FOCUS/EXPOSURE MONITOR

by

Pamela M. Tsai

Memorandum No. UCB/ERL M94/55

28 July 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A NEURAL-NET BASED, IN-LINE FOCUS/EXPOSURE MONITOR

by

Pamela M. Tsai

Memorandum No. UCB/ERL M94/55

28 July 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

The calibration of defocus distance and exposure time in lithographic equipment for integrated circuits fabrication is currently performed manually. An automated approach promises better consistency and reproducibility at a lower cost. The two critical parameters that determine the performance of a lithographic stepper are the defocus distance and the exposure time. Currently, the optimal settings are selected after examining a pattern that has been projected several times across one wafer. Each projection is done under a different combination of exposure time and defocus. The "best" pattern is chosen by an experienced operator, who looks for the image that appears to be the sharpest, having the most vertical sidewalls, and whose critical dimensions are the closest to those of the desired pattern. The focus and exposure settings corresponding to this image are then selected as the settings to use. This, for example, is done when choosing the best exposure and identifying current focus in using a SMILE or Bossung plot. This calibration procedure has to be repeated periodically since the stepper, the light source and the chemicals tend to age. Calibration is also necessary whenever maintenance is performed, or whenever the machine is configured for the patterning of a new layer.

In this project we applied a two dimensional pattern recognition network which was trained to choose the "best" developed image. We collected a database of digitized optical calibration images generated on our stepper and tagged with a qualification code supplied by a human expert. A feed forward network was trained using the backpropagation training algorithm to recognize key aspects of the patterns exposed under different stepper settings. We used image processing techniques (such as edge extraction and convolution) to pre-process the data before it was presented to the neural net. Results show that using image processing techniques and neural nets can identify the optimum settings for the stepper with a success rate as high as 96%. Also, this procedure can be extended to ensure that the exposure and focus settings are optimal on a run-to-run basis in a production environment.

Acknowledgments

I would like to thank Professor Costas Spanos for being such a wonderful research advisor and for his valuable guidance during this project. I am also grateful to Professor Bernhard Boser for his useful advice on dealing with neural networks, and to Professor Andy Neureuther for reading my project report. Fariborz Nadi made extremely essential contributions to this project from the very beginning to the very end. I am also indebted to Debra Hebert and Maria Perez for helping me in the Microlab and to Christopher Hylands for installing the image-acquisition equipment. Special acknowledgments go to Sov Leang, John Helmsen, and Derek Lee for their important insights into SAMPLE, and to Joe Weber for teaching me how to use Khoros. The BCAM group has been so wonderful; many thanks are due to Eric, Sherry, Zeina, Mark, Sean, Raymond, Roawen, Dave, Shang-yi, Sov, Tony, Crid, Herb, Manolis, Fariborz, and David. Finally, Andy Abo is especially appreciated for his encouragement and support.

This project was supported by the Joint Services Electronics Program (F49620-93-C-0014) and by the SRC (93-MP-700) and (94-YP-700.) The experiments involved with this project were performed in the University of California at Berkeley Microfabrication Laboratory.

Table Of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview

Regular monitoring of semiconductor equipment is essential to the semiconductor industry to ensure that the quality of the product and the efficiency of the processes are as high as possible [1]. Automating this monitoring promises better consistency and reproducibility at a lower cost. This project focuses on automating the calibration of a lithographic wafer stepper by using image processing techniques and neural networks to ensure that this important piece of equipment is operating under the appropriate settings. In order to accomplish this, we used optical microscope images of resist features that were exposed by a lithographic wafer stepper under different focus and exposure settings. This method is applied on the GCA 6200 wafer stepper in the Berkeley Microfabrication Laboratory. This wafer stepper is a 1 Megabit generation stepper; however, the techniques described in this project can be extended to today's more aggressive technologies that utilize 64 Megabit and 256 Megabit steppers which can resolve features as small as 0.25 µm. It is important to have some understanding about optical projection lithography in order to understand the experiments in this project; therefore, lithographic wafer steppers are discussed in the next section.

1.2 Optical Projection Lithography

The photolithographic steps are very important in the semiconductor processing sequence because they define the patterns on the different layers on the wafer. One of the main items involved in photolithography is the mask. Masks are glass with chrome islands forming the desired patterns. These patterns are transferred to a light sensitive material called photoresist [2].

Lithographic wafer steppers are used to expose the photoresist through the mask with high-intensity ultraviolet light. Wherever there are openings in the chrome of the mask, the light can pass through and expose the resist. Currently, there are two kinds of photoresists that are typically used in the Berkeley Microfabrication Laboratory, a G-line resist, which is made to be spec:ally exposed at a wavelength of 435 nm, and an I-line resist, which is for exposure at wavelengths of 365 nm. Both of these are positive resists. As can be seen in Figure 1, a positive resist that has been exposed to UV light will be washed away in the following development step. On the other hand, a negative resist will remain on the surface wherever it is exposed [2].

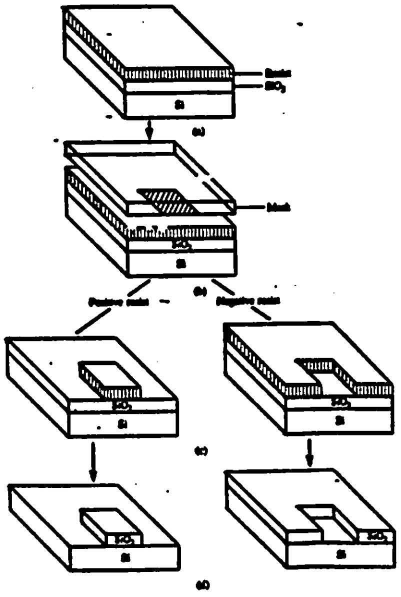


Figure 1 Resist and silicon dioxide patterns following photolithography with positive and negative resists [2].

A lithographic wafer stepper is a type of projection printer. See Figure 2 for a schematic of a projection printer. The GCA 6200 wafer stepper (GCAWS) used in the Berkeley Microfabrication Laboratory has the capability to project a pattern, and then to move, or "step", the wafer stage and project the pattern again elsewhere on the wafer, under different exposure conditions, if desired. The GCAWS uses an exposure wavelength of 365 nm, the wavelength with which the I-line resist is usually used. Figure 3 shows a drawing of a typical lithographic wafer stepper [3].

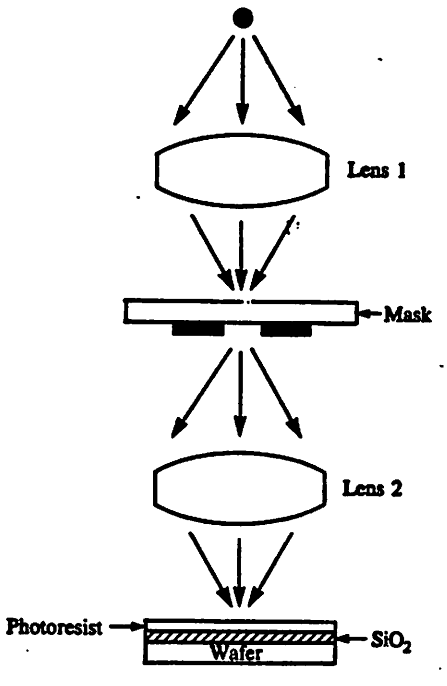


Figure 2 Artist's conception of a projection printer [2].

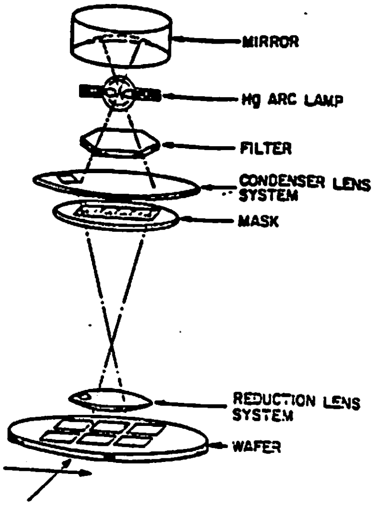


Figure 3 A typical lithographic wafer stepper [3].

There are several parameters associated with an optical projection printer (and therefore with a wafer stepper.) One of them is called the numerical aperture (NA), which is a measure of a lens' capability to collect diffracted light from a light source and to project it onto a surface. Its defining equation is as follows [3]:

$$NA = n \sin\theta \tag{1}$$

where n is the refractive index (of air in this case) and 2θ is the angle of acceptance of the lens. Typical values for NA are between 0.16-0.63. As can be seen in the Figure 2, there are two lenses in an optical projection system, the condenser lens and the projection lens. The condenser lens is collecting the light from the source and projecting it to the mask, and the projection lens is collecting the light from the mask and projecting it to the silicon surface. There are values of NA associated with each of these lenses. The ratio of the NA's for these lenses (NAc/NAo) is σ, the partial coherence value of the system. Typical values for σ are between 0.4-0.85 [4].

The GCAWS that was calibrated in this project has an NA of 0.28 and a σ of 0.5. The theoretical limit of resolution for this stepper can by found by using the following equation:

$$linewidth = k_1 \times \lambda / (NA) \qquad (2)$$

where λ is the wavelength of the light used the in exposure and k_1 is a constant which ranges from 0.4 to 1.2, depending on resist type and stepper characteristics [4]. For a value of k_1=0.61, this equation yields a linewidth equal to 0.8 μm. This implies that 0.8 μm lines are the smallest lines that the GCAWS can clearly resolve.

Another important factor to consider when dealing with an optical exposure tool is the exposure time. The longer the resist is exposed to the UV light, the larger the dose of energy that is imparted to it. Areas of resist that are supposedly protected by the chrome on the mask (especially at the edge of an opening) may have received some exposure from the UV light if the exposure time is too long. As shown in Figure 4, areas that should remain dark can indeed receive some exposure because of diffraction effects [5]. As a result, for positive resists, too much of the resist will be developed away and features will become smaller or even disappear.

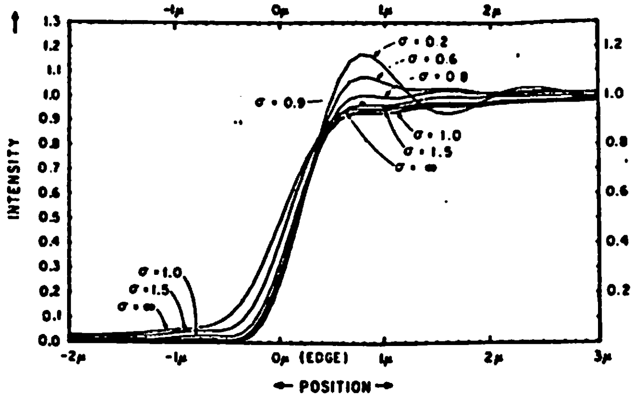


Figure 4 Aerial image intensity vs. position for a knife-edge pattern at a wavelength of 0.436 nm and numerical aperture of 0.28 for various amounts of partial coherence [5].

On the other hand, if the exposure time is too low, then the areas may not be fully exposed where they are supposed to be. Therefore, extra resist would remain after development, resulting in larger features and uncleared resist. Figure 5 shows an example of an overexposed image, and Figure 6 shows an underexposed example. In these examples, the features (i.e., the squares, lines, and X's) are resist "islands" on the substrate. Note the difference between those two images and the ideal image shown in Figure 7.

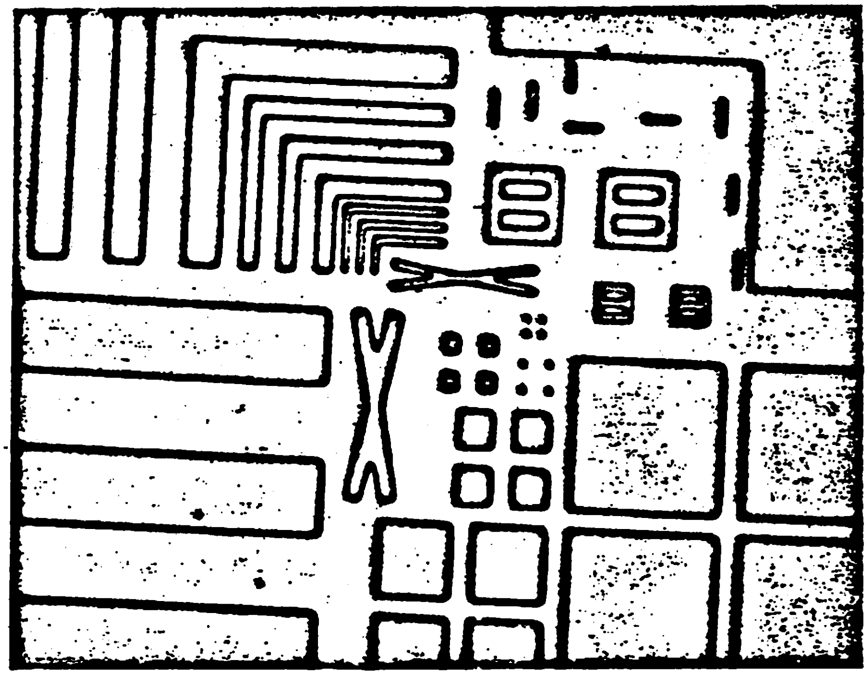


Figure 5 An overexposed image.

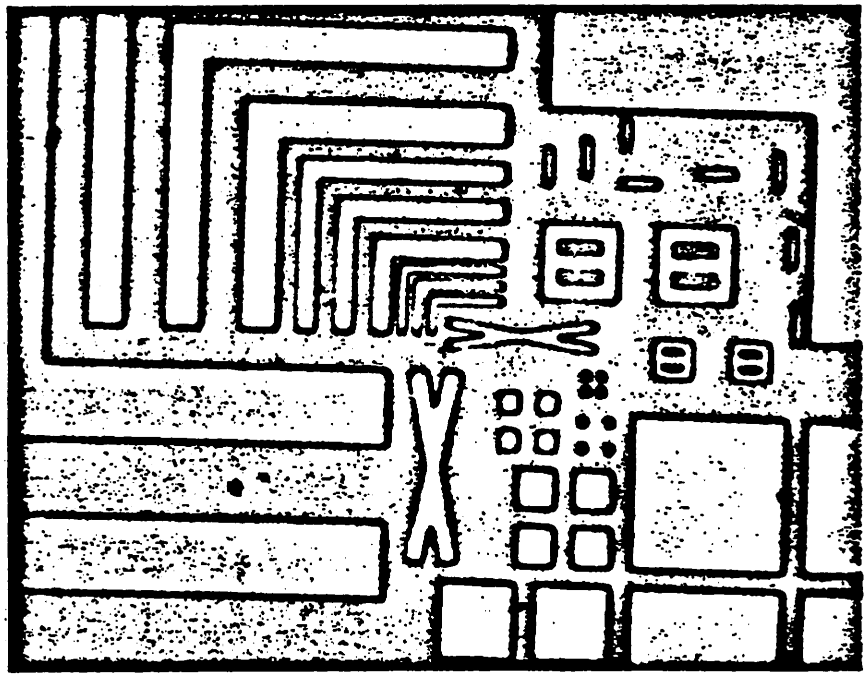


Figure 6 An underexposed image.

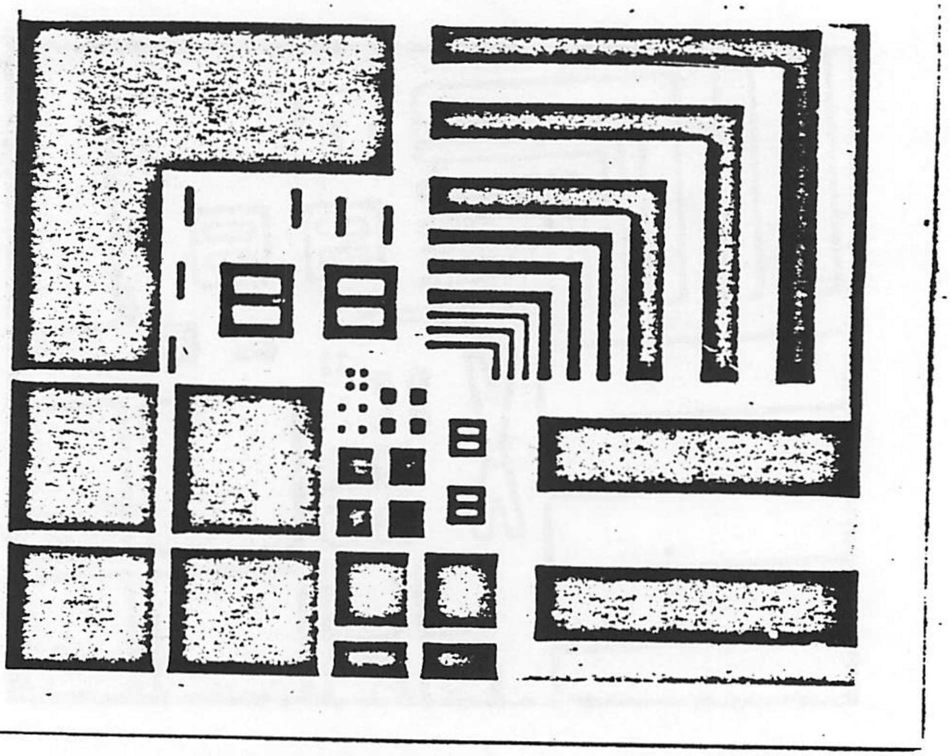


Figure 7 The ideal image.

The ideal exposure time for a stepper may drift over time, since changes in the light source, the chemicals, and the cleanroom environment may cause the ideal exposure time to change as well. For the microlab stepper, the ideal exposure time usually ranges between 0.6-1.0 seconds for the G-line resist, and from 0.25-0.50 seconds for the I-line resist. Calibration of exposure time must therefore be performed periodically [1].

Another important parameter of a lithographic projection system is the defocus distance. There is a certain distance separating the projection lens and the image plane that will produce the clearest image. If the distance between the lens and the image plane is moved from this ideal distance by, say, Δ, then the image will become blurrier and less well defined. This distance Δ is known as the defocus distance (see Figure 8.) Therefore, if the distance separating the projection lens and the image plane is "just right", then the defocus value is zero, and the corresponding image will be clear and well defined. If the lens is 1 µm further than the ideal distance from image plane, then the defocus distance is 1µm, and the corresponding image will be blurry [6]. Defocus distance can measured in Rayleigh Units; the Rayleigh defocus distance is given by

$$RayleighDefocusDistance = \frac{\lambda}{2\left(\cdot NA^2\right)} \tag{3}$$

For our system, which has an NA =0.28 and λ = 365 nm, a Rayleigh Unit is 2.32 µm.

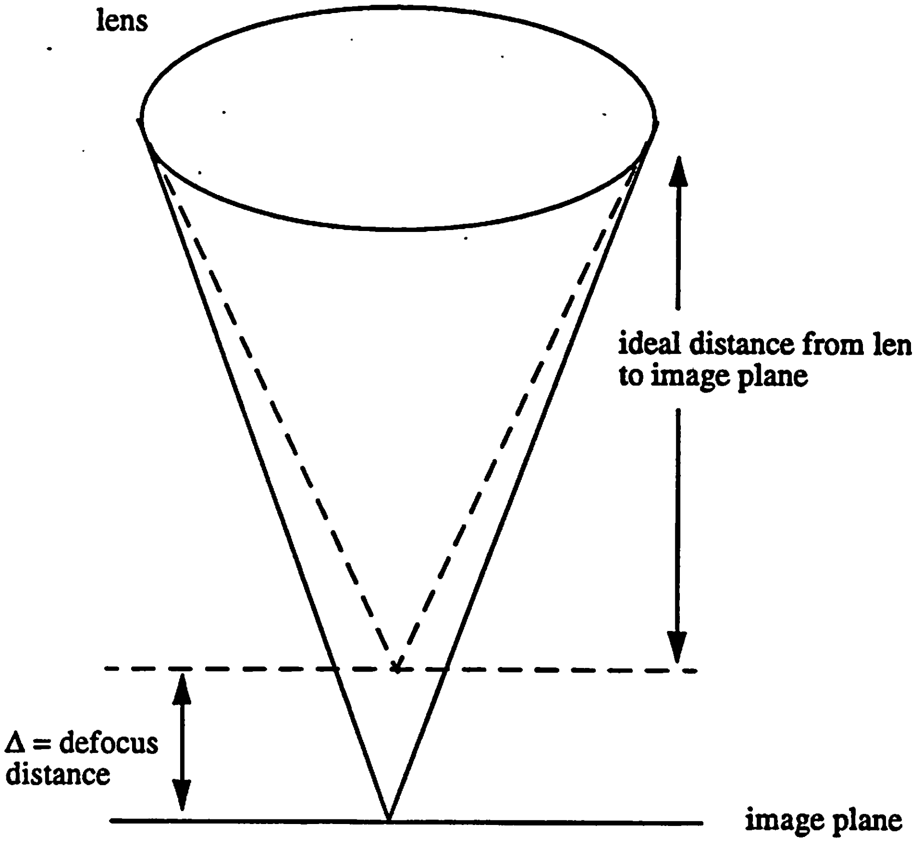


Figure 8 A visualization of defocus distance.

As can be seen in Figure 9, if the defocus distance is not zero (i.e. the stepper is poorly focused,) then the light intensity reaching the wafer will be poorly imaged [4]. This results in a decrease in linewidth and poorly defined edges and corners, giving the features a "fuzzy" appearance. Two-dimensional features, such as squares, have been found to be especially sensitive to focus [4].

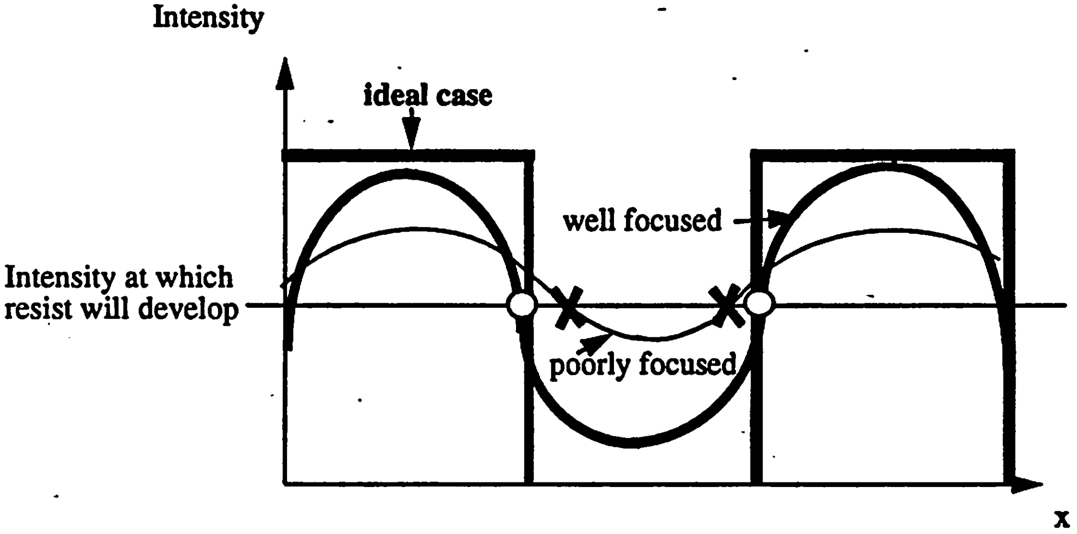


Figure 9 Light intensity corresponding to a mask of lines and spaces for a the ideal case, a well focused case, and a poorly focused case. The poorly focused case would produce a smaller linewidth and poorly defined edges.

See Figures 10 and 11 for an example of a focused pattern and a poorly focused pattern. The image will look the same if it is overfocused or underfocused by the same amount, i.e. if the focal plane is above or below the ideal position by the same amount. For example, if the setting is +1 Rayleigh Unit above the ideal distance, the resulting image will look the same as if it were exposed with the focus setting -1 Rayleigh Unit from the ideal distance.

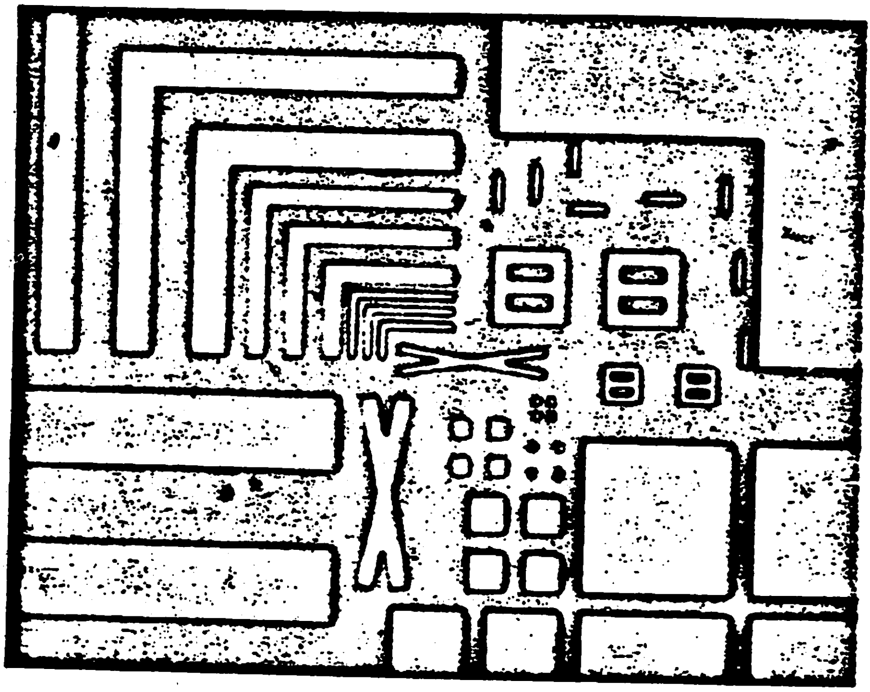


Figure 10 A well focused image.

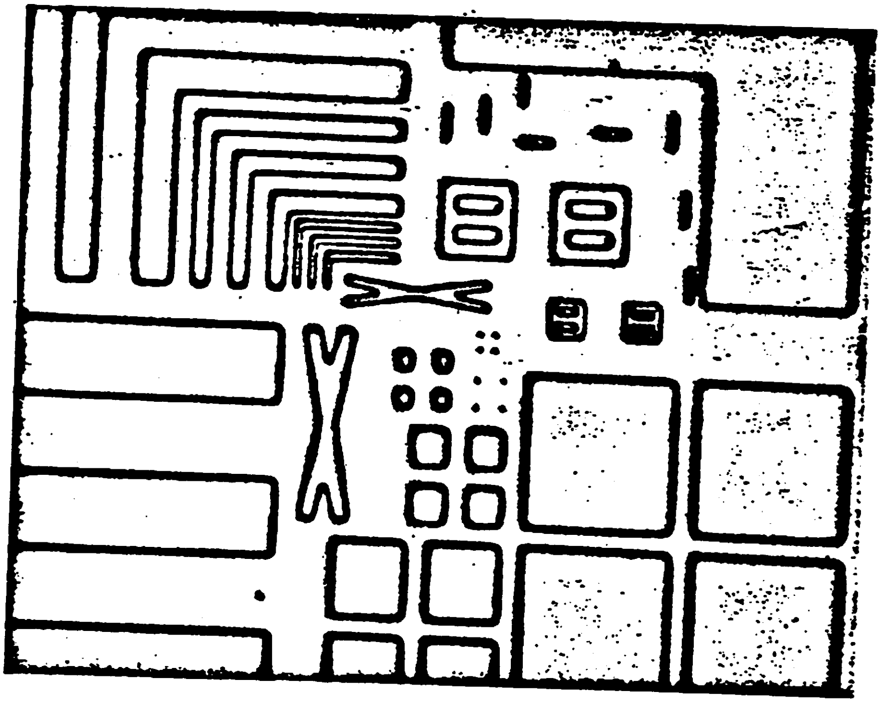


Figure 11 A poorly focused image.

The GCA 6200 wafer stepper was calibrated in this project. During the original installation of this piece of equipment, the "ideal" focus setting was somewhat arbitrarily set to be around 300 "divisions" [7]. It is from this "ideal" reference baseline that all changes in focus are made. Therefore, one can think of this value as corresponding to the zero defocus case discussed before. This value should remain approximately the same from run to run; however because the stepper ages and its components may drift over time, this value may change over time, and has to be adjusted accordingly to produce the best results. This value can take on values from about 125-375. Therefore, calibration of the ideal focal distance must be performed periodically to find the correct value that should be used.

One important thing to keep in mind is that there are 0.37 μms in a "division." Therefore, when the focus setting is moved from, say, 300 to 304, the automatic lens focus system null position moves 4 x 0.37 = 1.48 μm, inducing a 1.48 μm change at the focus plane. This value is approximately 64% of a Rayleigh Unit which, for this system, is about 2.32 μm.

1.3 The Manual Method of Calibration

The calibration of defocus distance and exposure time in lithographic equipment for integrated circuits fabrication is currently performed manually. An automated approach promises better consistency and reproducibility at a lower cost. The two critical parameters that determine the performance of a lithographic stepper are the defocus distance and the exposure time. This calibration procedure has to be repeated periodically since the stepper, the light source and the chemicals tend to age. Calibration is also necessary whenever maintenance is performed, or whenever the machine is configured for the patterning of a new layer.

Currently, the optimal settings are selected after examining a pattern that has been projected several times across one wafer [1]. Each projection is done under a different combination of exposure time and defocus, forming a 7x7 matrix of images performed under different settings. The "best" pattern is chosen by an experienced operator, who looks for the image that appears to be the sharpest, having the most vertical sidewalls, and whose critical dimensions are the closest to those of the desired pattern. Figure 12 shows the mask from which these resist images were made. Figure 13 shows a case where the resist was overexposed and poorly focused, which the human operator can determine by the disappearance of small features and a general blurriness of the corners. Figure 14 shows an underexposed case, where there is still resist left behind after the development step. As can be seen in Figure 15, which shows the results of ideal focus and exposure settings, there is no leftover resist, the features look sharp and well defined, and are not too small. Therefore, the human expert would choose this image as the "best" and would reset the stepper settings with this image's corresponding focus and exposure settings.

The above mentioned procedure is labor intensive, expensive and error-prone. Therefore, the objective of this project is to employ image processing techniques and neural network modeling in order to automate the visual analysis performed by the operator.

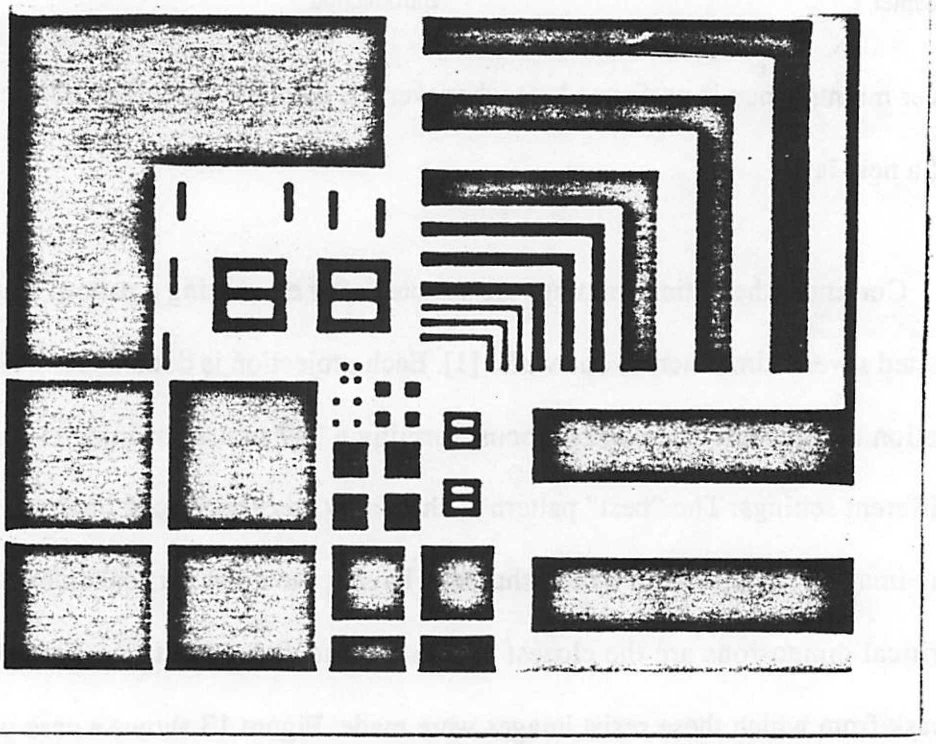


Figure 12 The mask from which the manual calibration images were made.

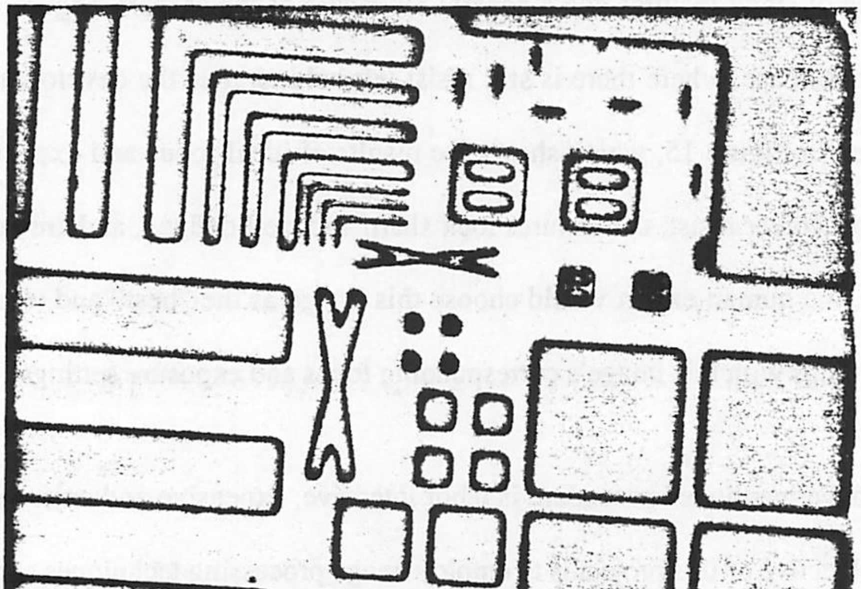


Figure 13 An overexposed, poorly focused image.

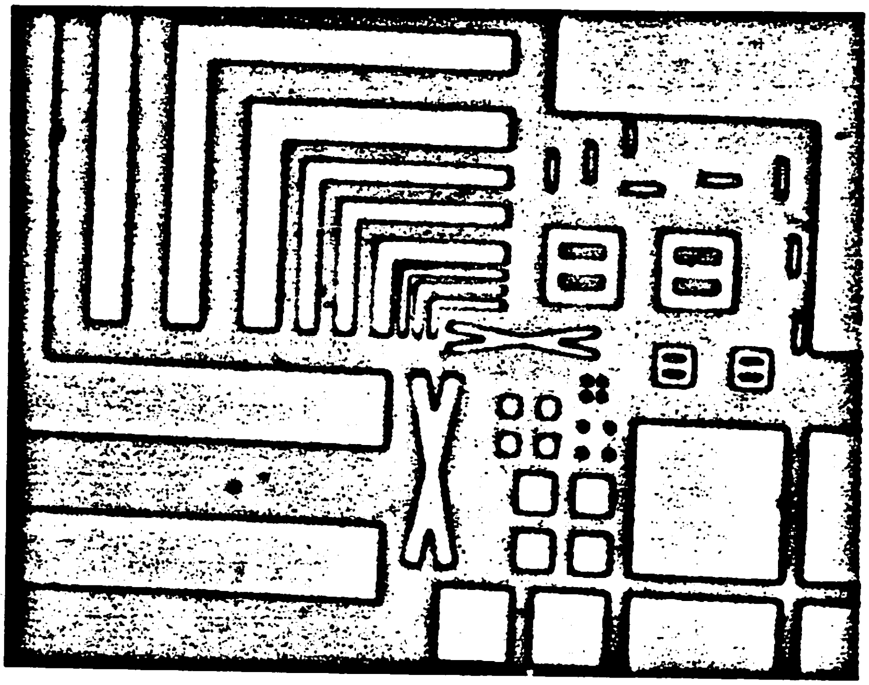


Figure 14 An underexposed image.

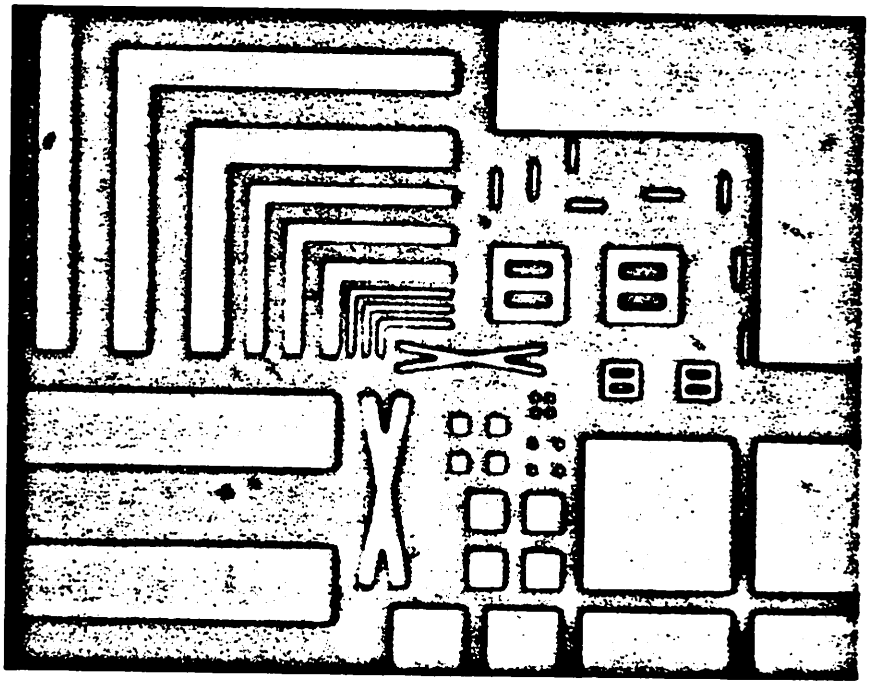


Figure 15 A well focused, well exposed image.

1.4 Outline of this Document

In this project, we investigate an automated way of choosing the best image using image processing techniques and neural nets on optical microscope images of resist features that were exposed by a lithographic wafer stepper under different focus and exposure settings. The image processing techniques that were used are described first, along with the Khoros[8] framework, within which they were implemented. A discussion of neural nets is then presented, with special emphasis on how they are used in this project. An experimental example is then shown, which describes the image acquisition setup, the new mask that was designed, the specific image processing techniques and neural nets used, and results for selected patterns. Some SAMPLE simulations were performed in order to investigate how sensitive feature size is to changing exposure time and defocus values, and to quantify how effective our calibration results are in keeping the linewidths close to their ideal values. Lastly, some other specific applications for this type of procedure are described.

Chapter 2 Image Processing

Image processing techniques were needed to preprocess the pictures before they were sent to the neural network, in order to quantify and simplify the data into a suitable representation. We could then use these values as the input to a neural network on which it could then be trained. Some image processing techniques that were used were edge extraction, pixel extraction, and convolution. These operations were implemented within the Khoros [8] framework, which is described at the end of this chapter.

2.1 Convolution

One image processing technique that was used to pre-process some of the image data was convolution. Convolution involves a kernel of numbers multiplied by each pixel and its neighbors in a small region, the results being summed, and the result being placed in the original pixel location [9]. This is applied to all of the pixels in the image. The original pixel values are used in the multiplication and addition, and the new derived values are used to produce a new image. Discrete convolution can be expressed mathematically:

$$g[m,n] \otimes f[m,n] \equiv \sum_{i=0}^{M} \sum_{j=0}^{N} f[i,j] \times g[m-i,n-j] \qquad (4)$$

$g[m,n]$ is the image and $f[m,n]$ is the kernel which is nonzero only for $0 < i < M$ and $0 < j < N$.

2.2 Edge Extraction

Another procedure that was applied to the images was edge extraction. This greatly simplifies other image processing procedures and minimizes the impact of noise. It demarcates the boundaries and enhances the visibility of small steps and other details [9]. See Figure 16 for an example of edge extraction.

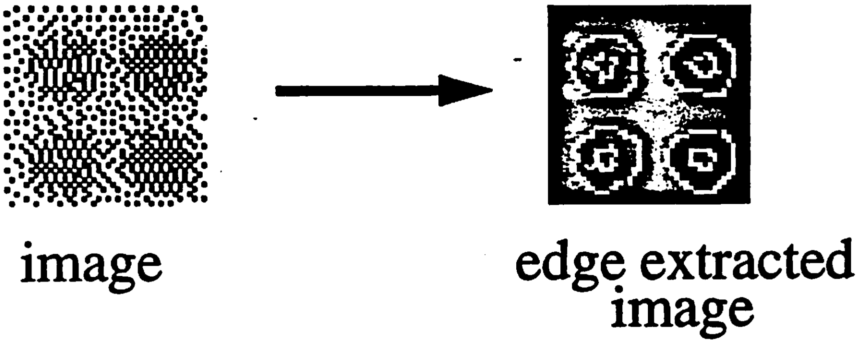


Figure 16 An example of edge extraction.

The Canny Edge Detector [10] was used as the edge extraction algorithm in this project. In this method of edge extraction, the image is convolved with the first derivative of a Gaussian function in two directions, resulting in a matrix. Local maxima in this matrix indicate a change in the intensity of the original image (i.e. a possible edge). The gradient of the matrix is then computed in order to give a vector perpendicular to the edge. The next step is to move along this gradient until a maximum is found; that point is then marked as an edge point. To prevent detection of false edges, there is a threshold value placed on the magnitude of the gradient of matrix, below which the output is ignored.

2.3 Using Edge Extraction and Convolution

All pictures and images were edge extracted prior to any convolution procedures in this project. Convolution can identify the location of a specific feature in a larger picture. If a small image of the feature in question is convolved with the larger picture, the pixel location of the maximum resulting convolution value would indicate the location of the feature in the large picture. See Figures 17-21 for a visual example.

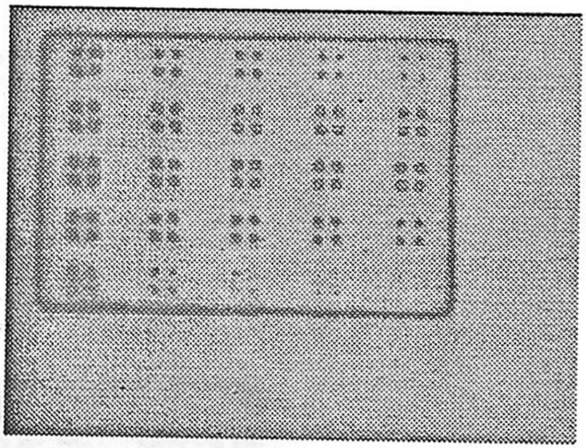


Figure 17 A large image.

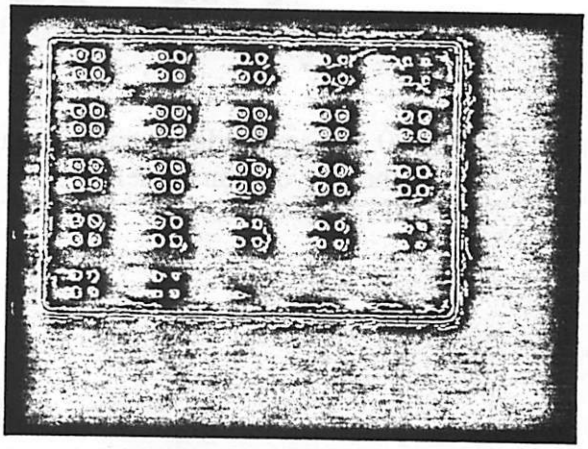


Figure 18 A large edge extracted image.

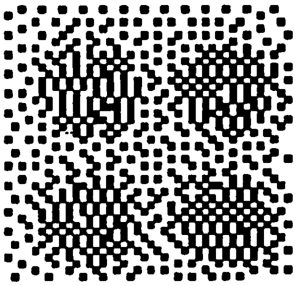


Figure 19 A subregion of the image (approximately 50 x50 pixels large.)

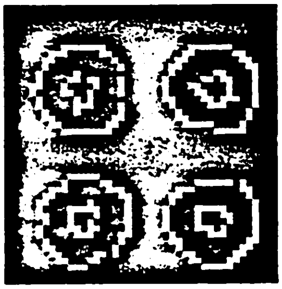


Figure 20 A small edge extracted image; this will be used as the convolution kernel.

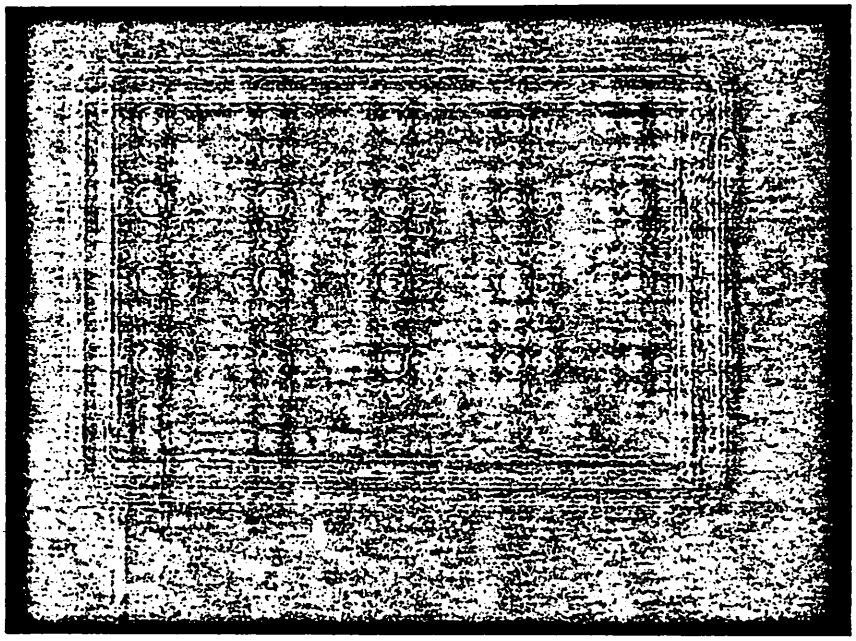


Figure 21 The convolution result of Figure 18 and Figure 20. The bright spots denote high convolution values and therefore indicate the most likely location of the pattern within the field of view.

An image obtained after a good exposure can serve as a kernel. We could then convolve this ideal image with other test images, which may have been overexposed, underfocused, etc. The convolution will produce a maximum value for the test image that most closely matches the ideal image. In this context, convolution can be seen as a two-dimensional "correlation" between an ideal image and the image being tested.

We were also interested in not only which test image produced the highest convolution value, but also what the "shape" of the convolution values around that maximum were for each test image. We recorded an area of convolution values for each test image. We could then be able to have information about each test image, independently of how it compares relative to the other test images. Figure 22 shows a convolution pattern for a test image.

This would be useful in determining by how much the focus or exposure settings were off (and in which direction it is necessary to move to correct it).

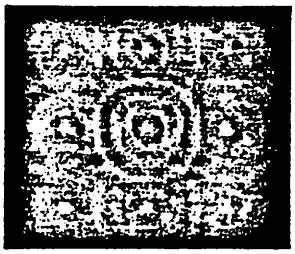


Figure 22 A convolution pattern for a test image.

2.3 Normalized Intensity Plots

2.3.1 One-dimensional Normalized Intensity Plots

Another image processing tool that was utilized in this project was extracting one-dimensional pixel cuts of a grey scale image and plotting the pixel intensities [8]. No edge-extraction is performed prior to the extraction of the one-dimensional cut. This method is especially convenient for line-space patterns, since much of the information in this type of pattern can be captured in a one-dimensional cut. In order to reduce noise, several adjacent one-dimensional cuts are taken, and their pixel values are averaged. As can be seen in Figure 23, which plots the average value of the pixels from five adjacent cut-lines from a 1 μm line-space pattern, the high intensities represent the bright areas of the images, and the low values represent the dark areas (the edges) of the image. The wider the bright areas in the intensity plot, the wider the corresponding line or space. Broad valleys imply that the line has thick edges, which in turn could infer that there is a substantial sidewall slope. The relative heights of the peaks (indicating lines and spaces alternatively) also convey valuable information about the conditions under which the line was exposed. The peaks corresponding to the spaces in overexposed images tend to be higher than the peaks corresponding to the lines. For underexposed images, the opposite is true; the peaks corresponding to the lines are higher than the peaks corresponding to the spaces. See Figure 24 and Figure 25 for examples of one-dimensional intensity plots of 1 μm lines that were exposed under different settings. All of the pixel intensity values were normalized to be between 0 and 255.

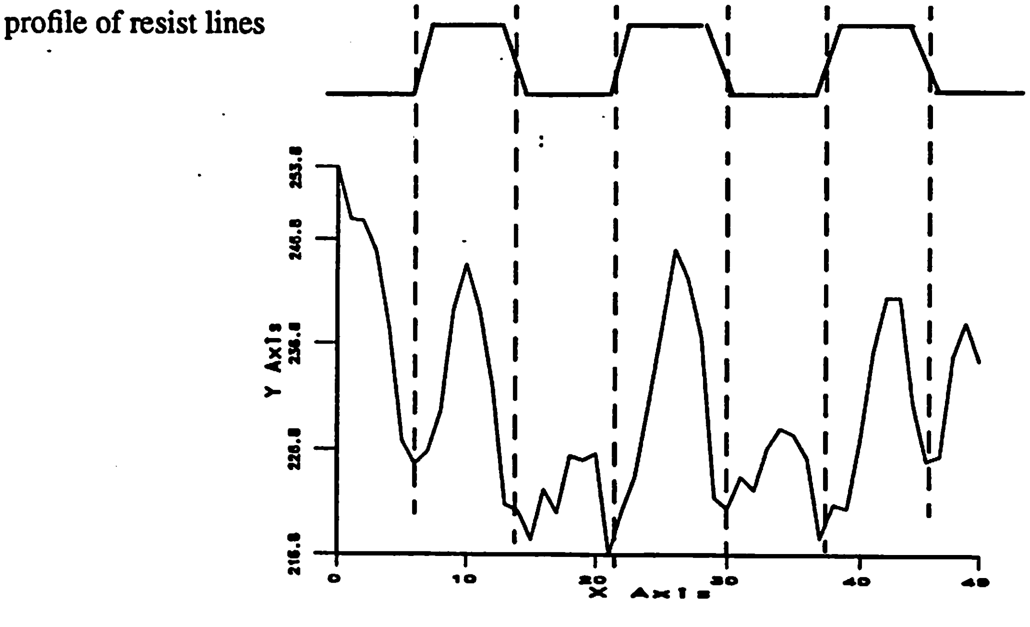


Figure 23 An example of an image intensity plot. Above is the corresponding profile of the resist lines. The dotted cut lines indicate how the edges in the profile correspond to the valleys in the intensity plot.

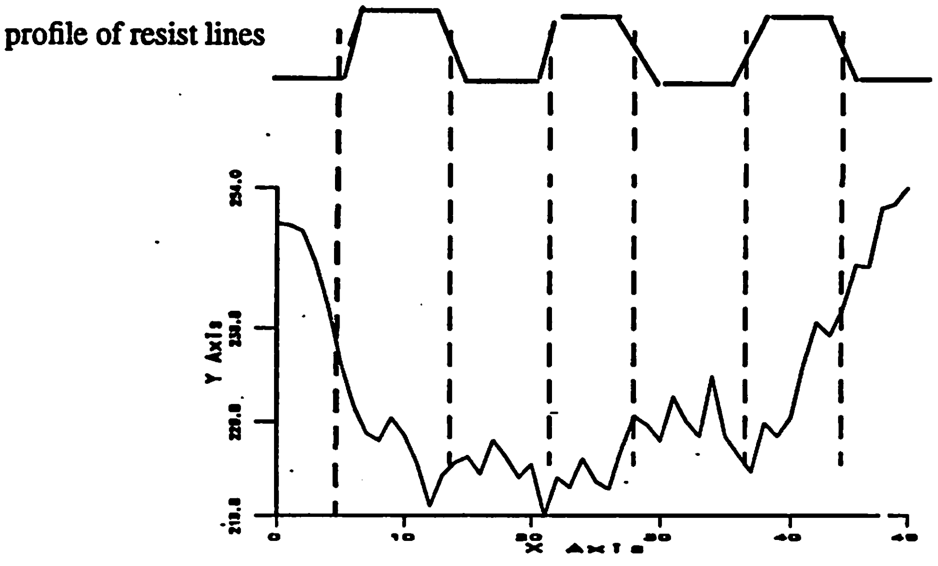


Figure 24 A one-dimensional pixel intensity plot of an overexposed line-space pattern.

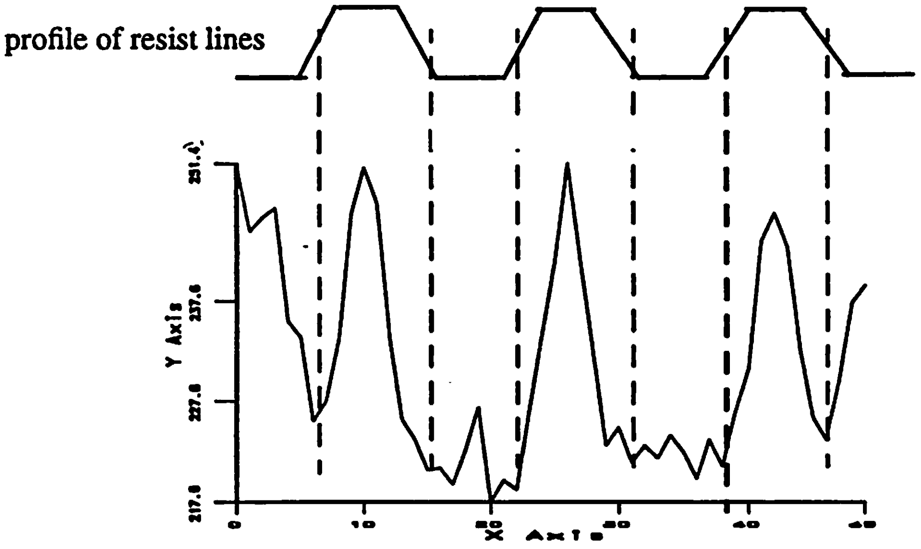


Figure 25 One-dimensional pixel intensity plot of an underexposed line-space pattern.

2.3.2 Two-dimensional Normalized Intensity Plots

In another effort to quantify the picture data into a representation suitable for presentation to the neural net, we also extracted two-dimensional areas of images and recorded the corresponding pixel intensities. This works best for small images, such as the two-squares feature shown in Figure 26 so that most of thc image can be captured in the extracted area. The ideal image of this feature is shown in Figure 27, along with the approximate location of the two dimensional area that was extracted. An example of a two-dimensional extraction is shown in Figure 28. Again, these values were normalized to be between 0 and 255.

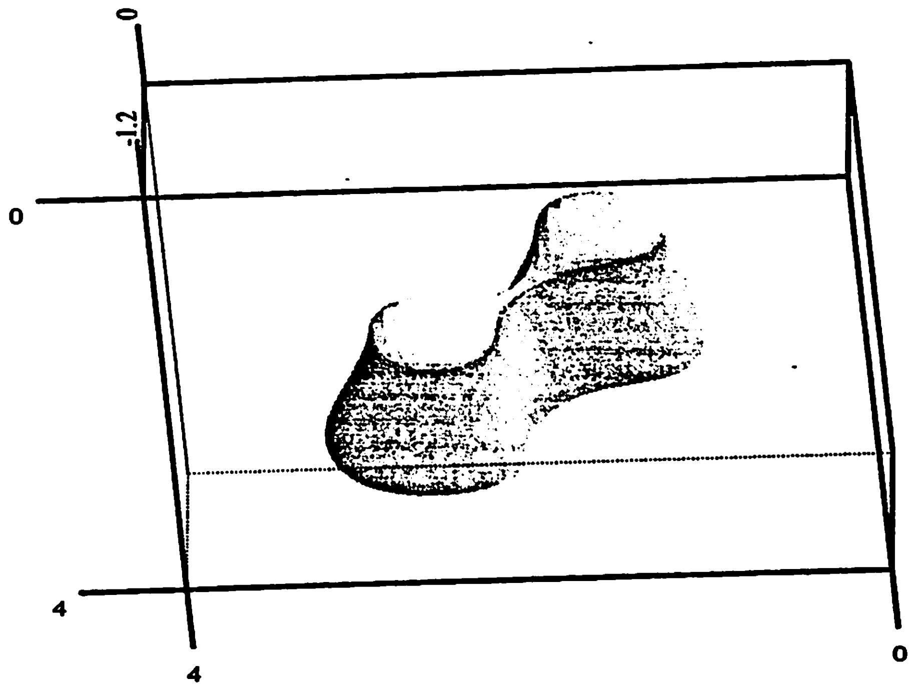


Figure 26 A SAMPLE 3D [11] simulation of the two 1 μm resist squares placed diagonally to each other.

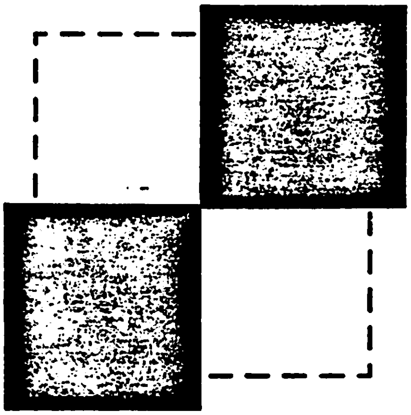


Figure 27 Ideal image of the squares and the area extracted.

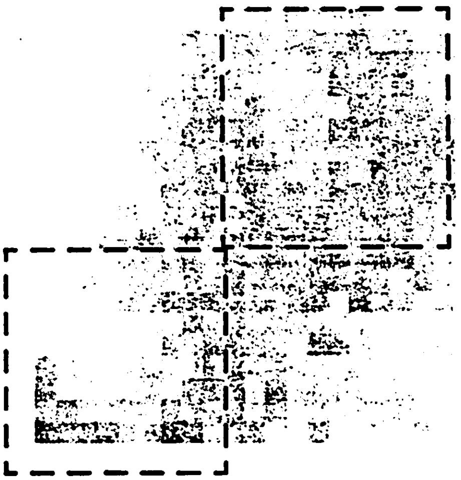


Figure 28 A 2 dimensional normalized pixel intensity plot of the two 1 μm squares placed diagonally. The dotted lines outline the ideal image of the squares.

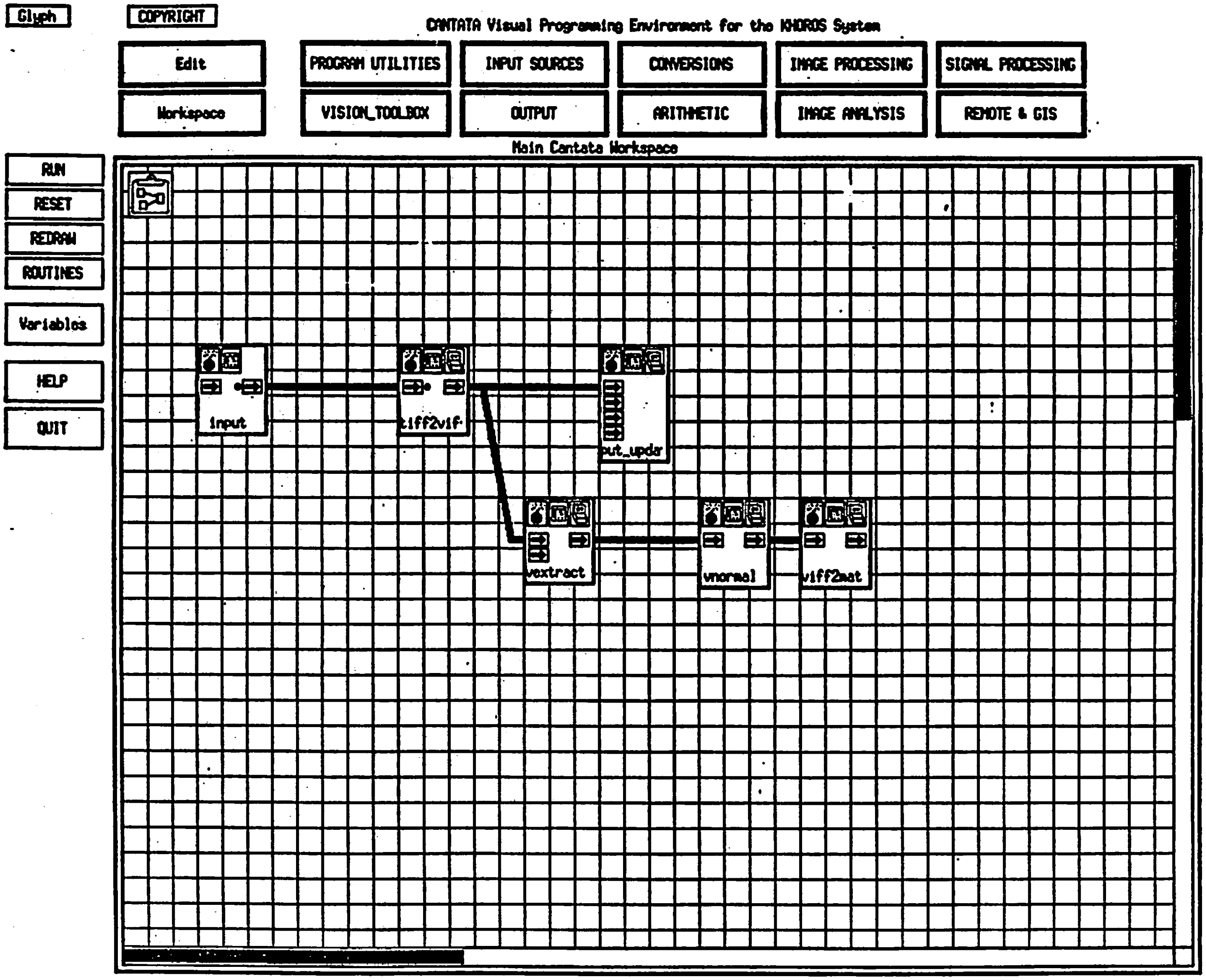
2.4 Khoros

The operations described in the previous sections were implemented within the Khoros [8] framework. The Khoros image processing software was used to preprocess the pictures. The Khoros system encompasses fundamental algorithms for image processing, and has an interactive image display/editor so that you can visualize the image data. A data flow graph, similar to block diagrams already used in image and signal processing, is used to access and connect routines from the processing library.

As can be seen in Figure 29, which is a simplified version of a workspace that was used in this project, the workspace contains blocks, or "glyphs", that each perform a certain function. The **input** glyph is where the input file is specified. The output of this **input** glyph is piped into the **tiff2viff** glyph, which converts the picture from a tiff format to a viff format, which is the only format that Khoros deals with directly. The output of this conversion glyph is connected to two different glyphs, as can be done with many of the Khoros glyphs. The **put_update** glyph displays the image fed into it. The **vextract** glyph extracts subregions from a picture; here we have it configured to extract a one-dimensional cut along the 1 μm line-space feature. These pixel values are then normalized to be between 0 and 255 with the **vnormal** glyph. Next the numbers are saved to a file using the **viff2mat** glyph, which will save the data in a matrix of numbers.

Other workspaces which contained routines to perform the convolution procedures, the averaging of pixel cuts and the two-dimensional pixel extraction were utilized in this project as well.

Figure 29 An examplc Khoros workspace which performs a one-dimensional pixel intensity extraction.



The previous discussion of image processing techniques dealt with the methods that were used to pre-process the pictures before they were sent to the neural network in order to quantify and simplify the data into a suitable representation. We could then use these values as the input to a neural network, which would then be trained using this data to recognize whether the resist feature in the image was processed under the correct focus and exposure settings. A discussion of neural nets is presented in the next chapter.

Chapter 3 Neural Network Theory

One of the goals of neural nets is to learn from examples and to generalize, things that the human brain can easily do. A neural net shares some basic qualities with the human brain, such as having a parallel distributed processing capability and being based on a model that emulates a neuron [12]. A neural net can be trained by presenting it with examples of inputs and their corresponding outputs. After the neural net has learned many of these examples, it can then be applied to new examples, where it now should be able to generalize and give the right output for these new examples.

3.1 Basic Background

The building blocks of neural nets are called neurons [12]. A simple neuron which is shown in Figure 30, is made up of input and output units. The links between them are assigned weights. This type of neuron sums the weighted inputs and passes the result through a nonlinear function. This nonlinearity could be a step function, or more commonly, a sigmoidal function (which can also be seen in Figure 30) and is described by

$$f(\alpha) = \frac{1}{1+e^{-(\alpha-\theta)}} \tag{4}$$

where θ is the threshold or offset.

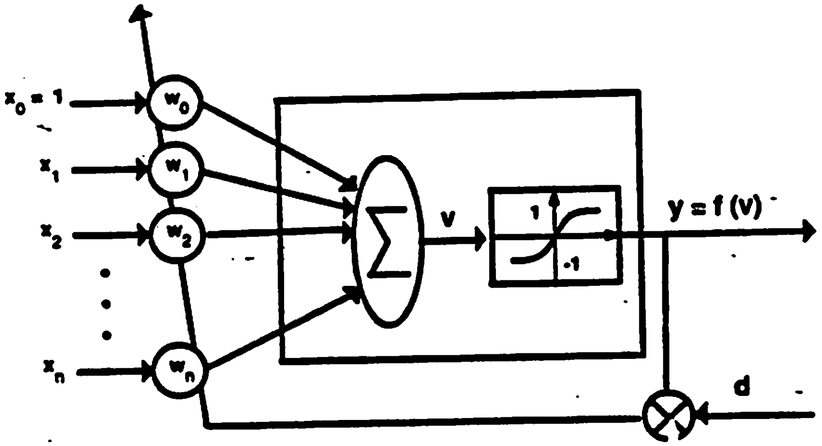


Figure 30 A formal neuron using the sigmoid function as its nonlinearity filter.

These types of neurons can be trained by adjusting the weights in the following way [13]:

Step 1. Initialize Weights and Threshold

Set $w_j(0)$ $(0 < i < N - 1)$ and θ to random values. Here $w_i(t)$ is the weight from input i at time t and θ is the threshold in the output node.

Step 2. Present New Input and desired Output

Present new continuous valued input $x_0, x_1 ... x_{N-1}$ along with the desired output d(t).

Step 3. Calculate Actual Output

$$y(t) = f_h(\sum w_i(t)x_i(t) - \theta)$$

Step 4. Adapt Weights

$$w_i(t+1) = w(t_i) + \eta[d(t) - y(t)]x_i(t), 0 \le i \le N-1$$

where *d(t)* = +1 if input from class A,-1 if input from class B.

In these equations η is a positive gain fraction less than 1 and d(t) is the desired correct output for the current input. Note that weights are unchanged if the correct decision is made by the net.

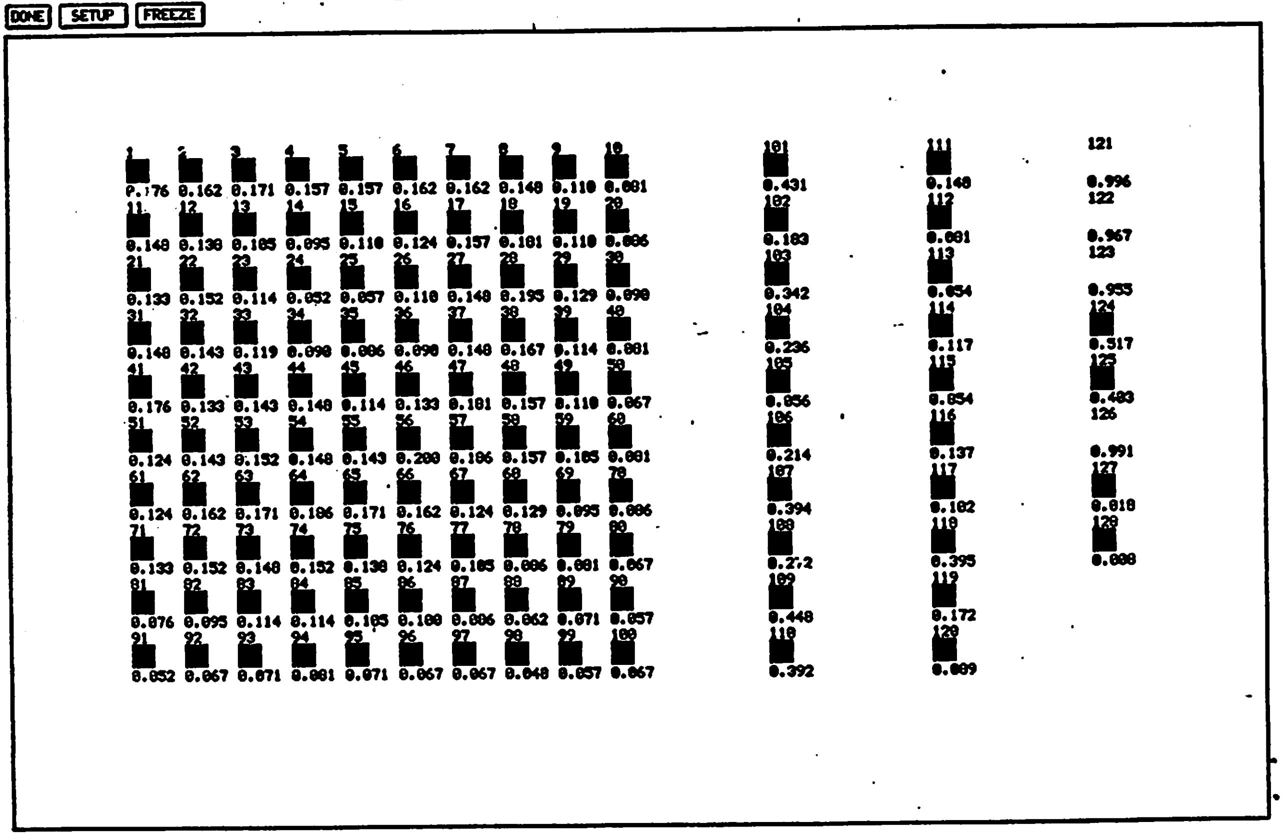
Step 5. Repeat by going to step 2

Other types of units that are neither input nor output units of the network are called hidden units and the play an important role in neural nets [12]. They make up one or more layers of nodes that are between the input and the output layers. The addition of a hidden layer enables the net to learn more complex examples. The training algorithm for this type of net is called the back-propagation algorithm. The general idea of this procedure is that it is trying to minimize a cost function equal to the mean square difference between the desired and the actual net outputs. Since the hidden units have no "desired" output value per se, their error is proportional to a weighted sum of the errors of the following layers, which had already been computed. Therefore, the errors are propagated backwards. The back-propagation training algorithm is described in more detail in [13].

3.2 Stuttgart Neural Network Simulator

We used the Stuttgart Neural Network Simulator (SNNS) [14] to simulate our neural networks. SNNS creates a simulation environment for the application of neural nets to various problems. The two main components of the SNNS simulator are the simulator kernel and the graphical user interface (GUI). The simulator kernel operates on all of the internal network data structures of the neural nets. The GUI gives a graphical representation of the neural networks and can be used to control the kernel during the simulation run. An example of a neural network display generated by SNNS is shown in Figure 31.

Figure 31 An example SNNS network display. This particular structure has 100 input nodes, 2 layers of hidden layers of 10 units each, and 8 output nodes.



3.3 Network Types

There were several different neural network types that SNNS was capable of simulating, such as the feed forward type, which simply has the different layers (the input, the hidden and the output) feeding into each other sequentially [12].

The Kohonen type network [12] uses a Self-Organizing Map algorithm. It is based on unsupervised learning. Self-Organizing Maps consist of two layers of units: a one dimensional input layer and a two dimensional competitive layer. Each unit in the competitive layer is linked to its neighbors with weights; all of the weights connected to a given unit can be described with a weight vector. Before starting the learning process, the competitive layer is initialized with normalized vectors. The input pattern vectors are presented to all competitive units. The dot product of the weight vector with the input vector is an indicator of how good the match is, and the best matching unit is chosen as the winner. The nodes in its neighborhood are also affected, so that similar input patterns tend to produce a response in units that are close to each other in the grid. Kohonen type networks are good for clustering input data, such as sounds in speech.

The time delay neural network [12] is made of neurons receiving inputs that have been delayed in time. This is good for tasks such as associating a letter with a corresponding phoneme in speech recognition.

There are many more types of neural nets available in SNNS, such as the adaptive resonance theory models, the Hopfield net and the Jordan net. We used the feed forward type in this project, because its structure was the same as the one of our problem.

3.4 Network Structure

There are several things to keep in mind when designing the structure of a feed-forward neural net. One issue is representing the data in the input layer. Usually, the number of input values describing the example is the number of input nodes used in the input layer.

Representing the output is more complicated and depends on the example. For instance, for the network to identify something as being one of three things, one could use three output nodes where one of the three nodes is set to 1 and the other two to 0. An alternative way to represent this output would be to have only one output node that can take three different levels in order to distinguish between the three different things, such as 0, 0.5, and 1. There are many more possibilities, and the output representation can be chosen after investigating several different ones. The output representation that produces the highest success rate in identifying things correctly should then be used.

The number of hidden layers and nodes should be kept to the minimum number possible that will still allow sufficient learning by the net [15]. If there are too many hidden nodes, the net will lose the ability to generalize, as it will be learning the training examples too specifically. So although the net might appear to have learned the problem very well during the training stage, it will do poorly when applied to new examples. We have found that in our problem, two hidden layers were sufficient.

3.5 Training: Optimum Stopping Point

When a neural network is being trained, the weights are altered to decrease the error on the training patterns. As can be seen in Figure 32, which shows the error curves of an example neural net being trained, the network has the largest amount of error in the initial stages of training, because it has not had enough iteration cycles to fully learn the examples yet. As it goes through more training cycles, the training error decreases. However, another issue to consider when training a net is the error curve of a trained net as it is being applied to new examples (the "testing" data.) At some point in the later stages of learning, the network may "overfit" the training data and the testing error would start to actually increase, even though the training error continues to decrease. It can be seen that after a certain number of training cycle iterations, (in this case, around 1500) the testing error of the net increases. This is because the net will have become overtrained on the specific training examples and will have lost its ability to generalize enough to correctly identify new data [16]. Therefore, the optimum training stopping point should be where the testing error curve attains its minimum.

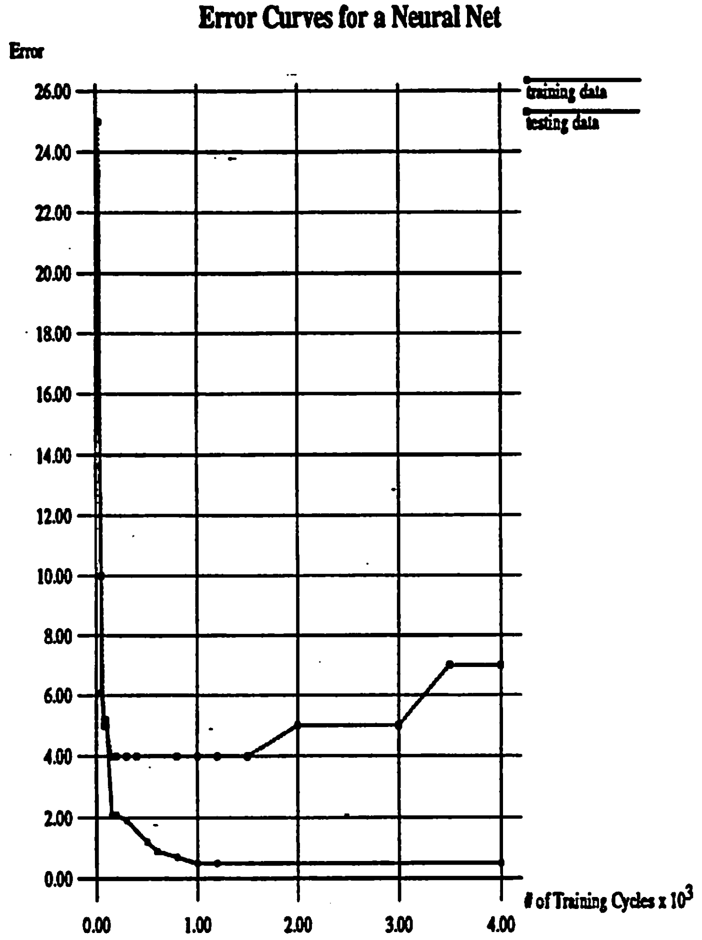


Figure 32 Training and testing error curves vs. # of training cycles of a neural net being trained.

Chapter 4 Implementation and Experimental Results

This chapter explains the image grabbing setup, the new mask that was designed, the specific image processing techniques and neural nets used, and results of how successful selected patterns were in identifying focus and exposure settings. Not only were we trying to identify the "best" settings, but we were also interested in identifying the specific focus and exposure settings for a given image, so that we could indicate in what direction and by how much to adjust the settings to attain the ideal conditions.

In order to understand how our experiments were designed, it is necessary to be familiar with the exposure tool. See section 1.2 for some background information on lithographic wafer steppers. The processing steps before the resist image inspection were the same as the current method of calibration used in the Berkeley Microfabrication Laboratory; an oxide layer 5000 A thick was grown on the wafer, and the resist was applied, exposed and developed according to standard Microlab parameters [1].

4.1 The Image Acquisition Setup

Figure 33 is a schematic of the image "grabbing" setup in the Microlab. Attached to an optical microscope is a Sony CCD camera which is hooked up to a Sanyo color video monitor/digitizer which is connected to a UNIX workstation. It is therefore possible to "grab" images of the exposed and developed resist on the wafer from the microscope and have them sent to UNIX, where further processing of the picture can be performed. Also attached to the microscope is a video monitor where the images can be viewed in the state in which they will be "grabbed". Whatever image is seen in the monitor is the resulting picture that is stored in UNIX.

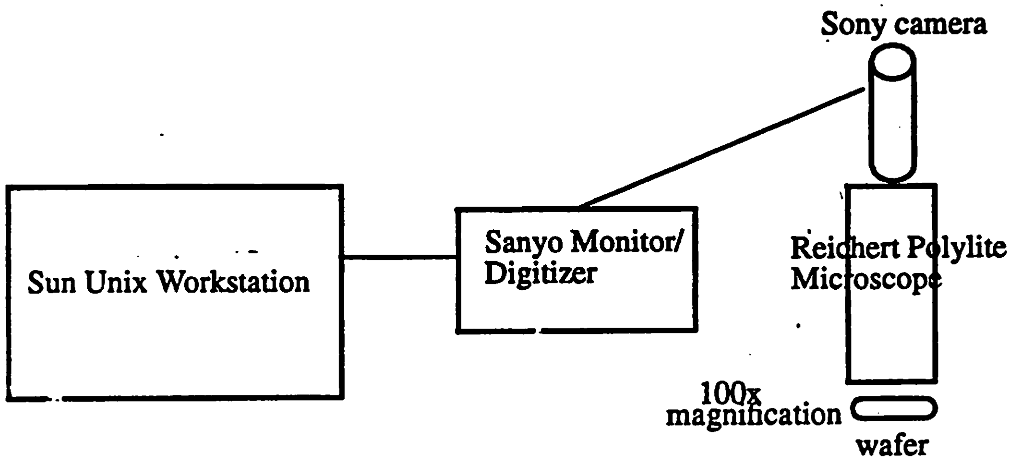


Figure 33 Schematic of the image grabbing setup in the Berkeley Microfabrication Laboratory.

We used the Reichert Polylite #30064 microscope, which is connected to output ports so that it can produce photographs, hardcopy printouts and UNIX image files. The images were captured with a magnification of 1000x, the highest power that was available. The resolution of the camera under this magnification is about one pixel per 0.1 μm. For example, a one micron wide line captured in the picture would appear in the image to be ten pixels wide.

4.2 New Mask Design

A new mask was designed so that a pattern processed under the full range of focus and exposures could be incorporated into one viewing field of the microscope. A picture is equivalent to one video monitor which can contain about 50 µm x 30 µm worth of information under 1000x magnification. See Figure 34 for a picture of a monitor screen worth of data. (The previous test mask only allowed one focus/exposure setting per picture.)

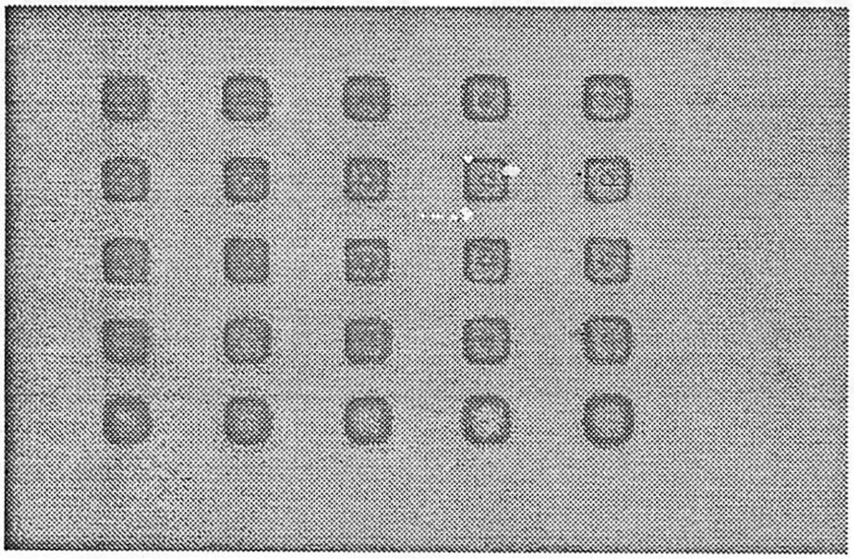


Figure 34 A monitor screen worth of data. This image shows the full matrix of the different focus (varying vertically) and exposure (varying horizontally) settings.

The new mask incorporated several different small features, each fitting within a 6 µm x 6 µm area, which would all be stepped and exposed in a 5x5 matrix of focus and exposure settings. The stage of the GCAWS would move in very small steps for each exposure (9 µm in the x direction, and 6 µm in the y direction) so that one picture could contain all twenty-five different combinations of focus and exposure settings for a feature. Figure 35 shows several features stepped on the wafer.

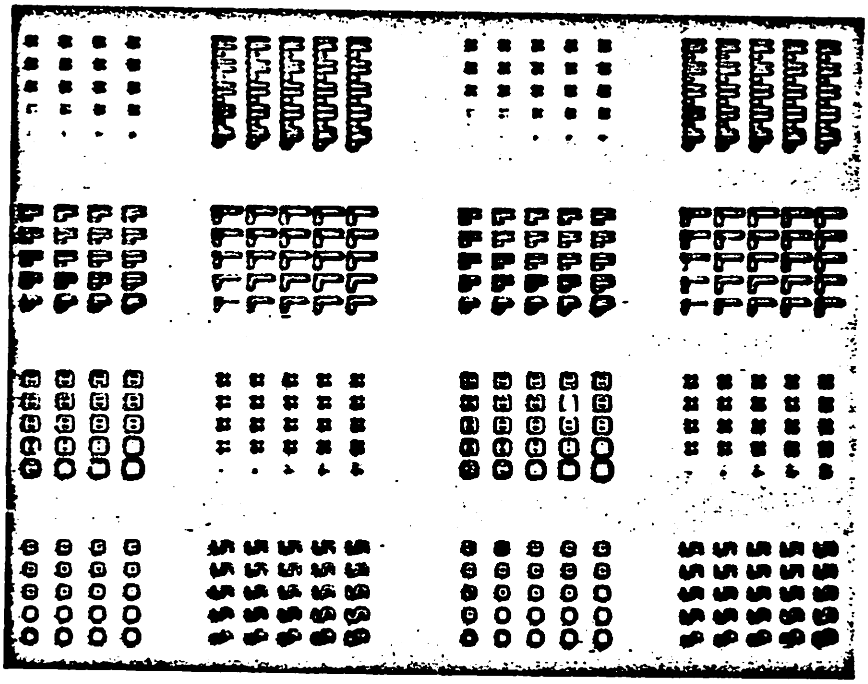


Figure 35 Several features stepped on the wafer.

All the patterns were each repeated forty-eight times on the mask, so that it could produce several hundred (768) monitor screens worth of data per wafer. This way, very few wafers were needed in order to provide all of the necessary training and testing data for our experiments. Figure 36 shows a picture of the mask.

Figure 36 Picture of the mask.

There are sixteen different features on the mask, eight of which are clear field features that would produce photoresist islands of the shapes on the wafer, and another eight, identical to these except they were done in the dark field manner so that there was resist everywhere on the wafer except in the shapes, where the oxide underneath was revealed. See Figure 37 for an example of resist islands and Figure 38 for an example of resist "valleys". We collected a database of these digitized calibration images that were generated on our stepper. A human expert had examined these images and identified the best one.

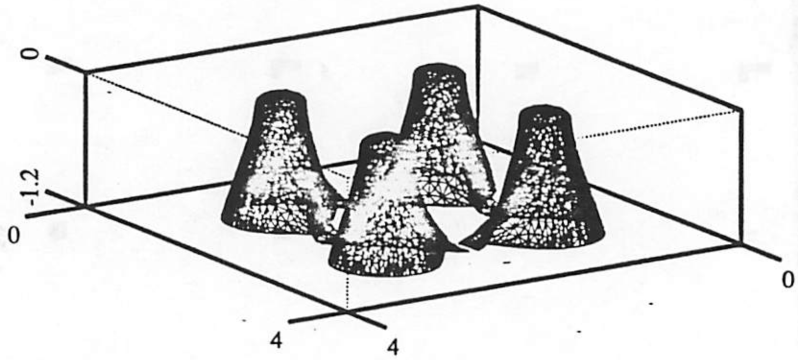


Figure 37 SAMPLE 3D [11] simulation of four 1 µm square "islands" of resist.

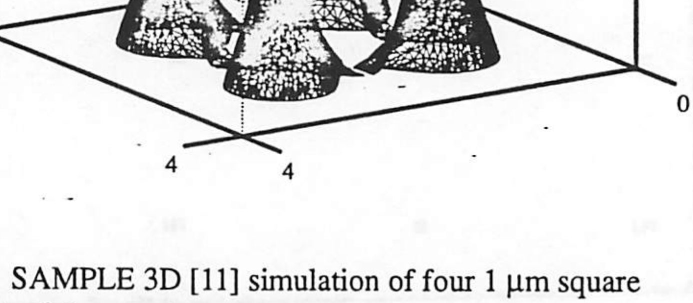


Figure 38 A SAMPLE 3D [11] simulation of four 1 µm square "valleys" of resist.

4.3 Conditions of Exposure

The experiment covered the same range of exposure and focus as in the runs performed for manual calibration, except that the focus and exposure conditions here were stepped only five times each, as opposed to seven times each (the number of steps used in the current, "standard" method.) Therefore, the range spanned by the exposure time values (for the G-line resist) would be, for example, from 0.65 seconds to 0.95 seconds for both the standard run and our run, but the standard run would contain the exposure times of 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, and 0.95 seconds, while our new conditions would consist of exposure times of 0.65, 0.725, 0.8, 0.875, and 0.95 seconds. The same concept is used when selecting the focus range and values for our experiment. See Table1 for an example matrix of settings for the "standard", current method, and Table 2 for an example matrix of settings for this experiment.

Table 1: Range of Exposure and Focus Settings used in the standard run.

exp (sec)	0.65	0.70	0.75	0.80	0.85	0.90	0.95

foc (div)	306	310	314	318	322	326	330

Table 2: Range of Exposure and Focus Settings used in this Experiment

exp (sec)	0.65	0.725	0.8	0.875	0.95

foc (div)	306	312	318	324	330

As can be seen in the above tables, the size of the focus setting increment for this experiment was 6 "divisions", which is equivalent to a 2.22 μm change in the focus plane. This value is approximately one Rayleigh unit of defocus for this system. (see section 1.2 for a discussion of defocus distance.) We performed the training exposures so that the "best" setting would be in the center of the matrix so that we could get a balanced range of overexposure, underexposure, and focus settings in the matrix.

4.4 Specific Image Processing and Neural Networks Used For Automatic Calibration

We used image processing techniques to pre-process the data before it was presented to the neural net because we needed to quantify the information contained in the pictures in order to get a suitable representation of the data. All images were processed using KHOROS, and SNNS was used as the neural network simulator.

4.4.1 Intensity Plots Method using 1 μm Lines

We investigated the 1 μm lines and spaces pattern (which is shown in Figure 39) to see if it could accurately indicate the focus and exposure conditions of the resist.

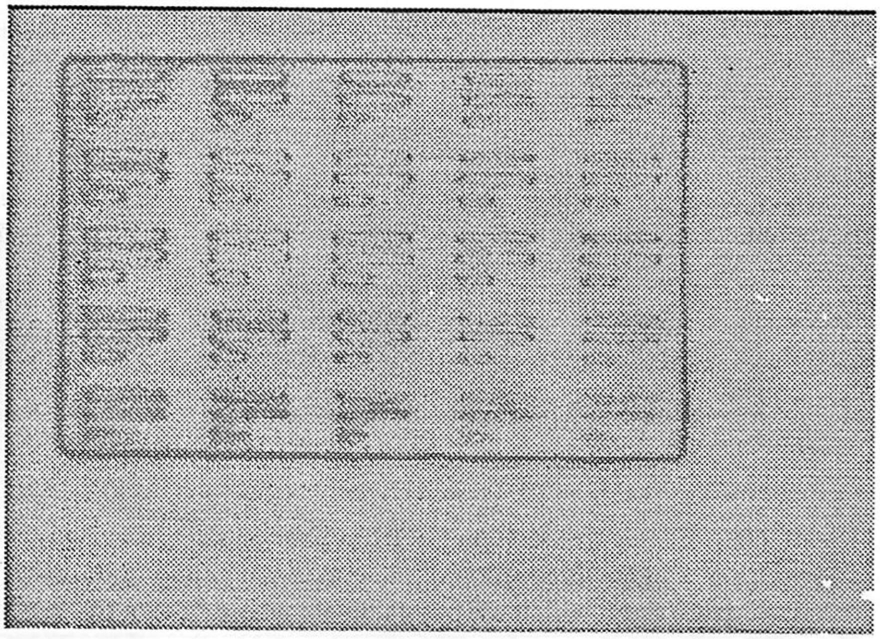


Figure 39 A grabbed monitor screen image of 1 μm lines and spaces.

These images were processed by the one-dimensional cuts/intensity plot method (see section 2.3 for more details.) A neural net with these values was trained to give us the appropriate focus and exposure settings. We used three wafers that included the "lines" feature on them. Two of these wafers had the I-line resist and the other wafer had the G-line resist. Twenty-four monitor screens worth of pictures were digitized; each of the monitor screens contained twenty-five images, resulting in a total of 600 individual images.

The first step was to locate the feature (in this case the lines) so that the appropriate one-dimensional cut could be taken. To do this, we used convolution. The pixel location of the peak convolution value would indicate a reference point in the image, which would indicate exactly where the feature was, since the convolution value would be highest where two images overlap. From this reference point, we can then figure out where to take the one-dimensional cut. First, we had to extract the edges of the images to simplify the convolution procedures and to minimize noise sensitivity. (See Figure 40.)

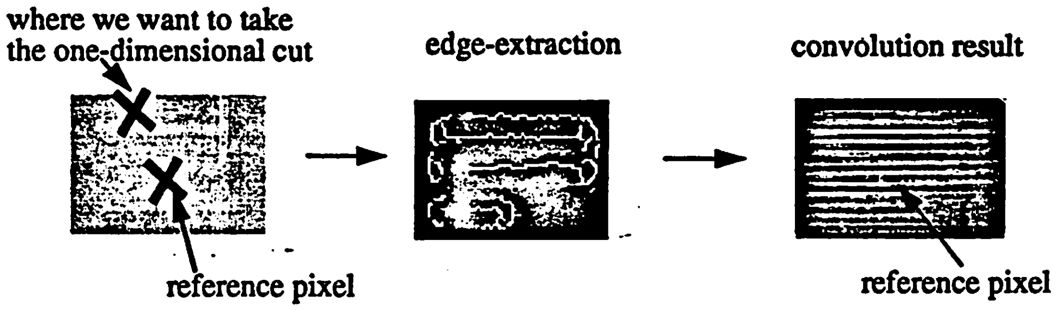


Figure 40 Edge extraction on the 1 μm lines and spaces pattern and the convolution result. The location of the maximum convolution value indicates the reference pixel from which we can calculate where to take the one-dimensional cut.

Next, a "best" image was chosen by looking at the matrix of images that were produced, keeping in mind that the ideal exposure setting would yield clear features, and that focus effects are symmetrical (i.e. images focused 1 μm above the ideal setting and images focused 1 μm below the ideal setting would look the same. Therefore, the focus setting about which the images were "symmetrical" would be the "ideal" one.) This image was then extracted and made into a smaller picture. This picture was then convolved with the entire test picture (consisting of all twenty-five different settings); the result of this convolution is seen in Figure 41. The pixel location of the peak convolution value for each of the twenty-five regions would indicate the location of each of the images (i.e. give the pixel coordinates of the location of the image), and therefore indicate where to take the one-dimensional cut.

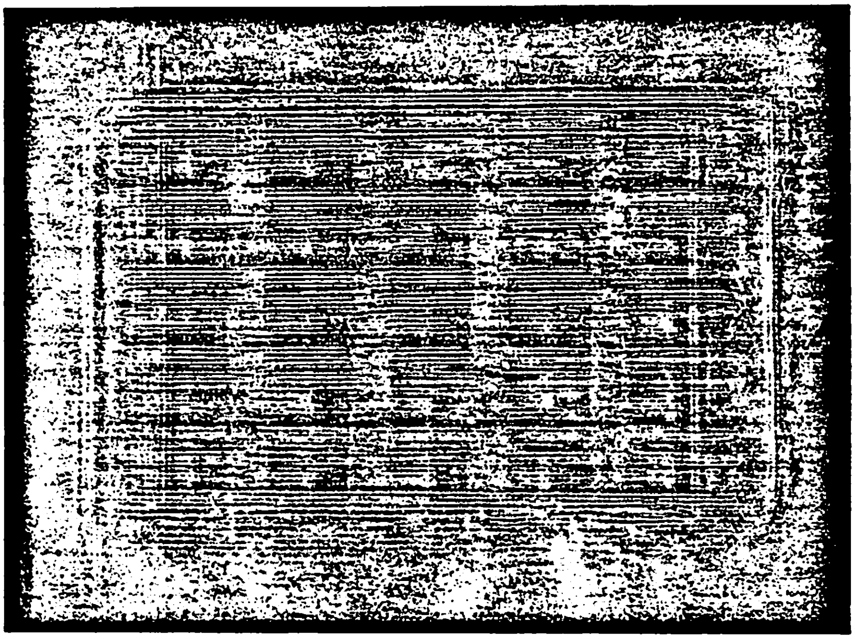
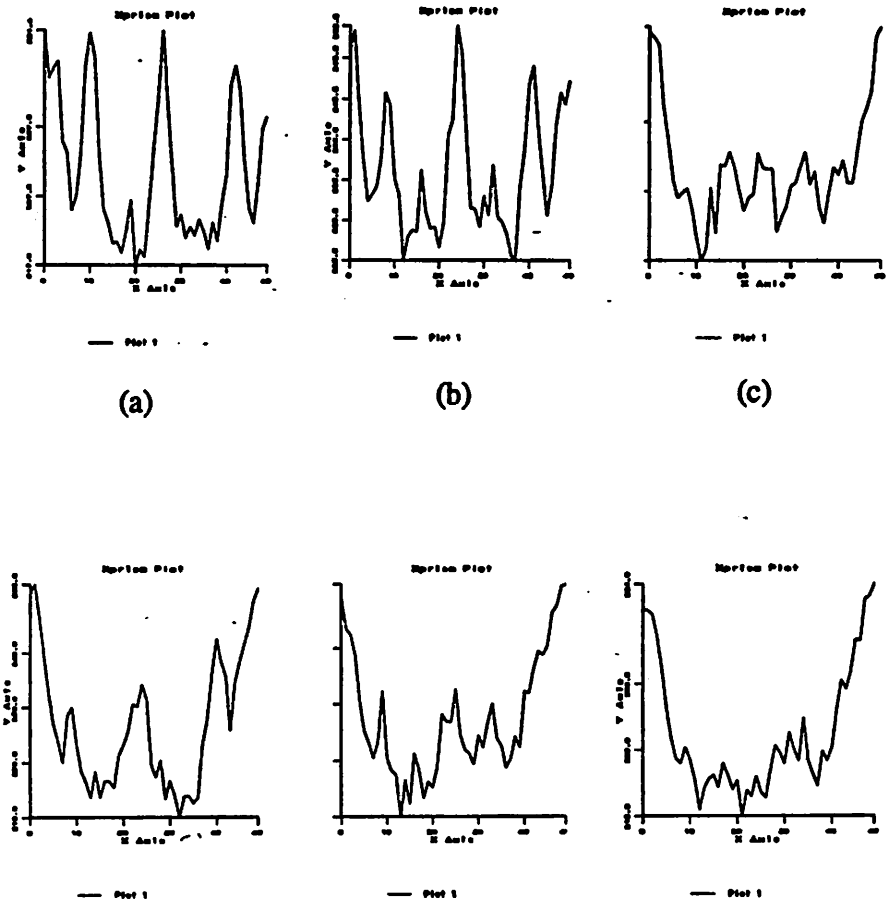


Figure 41 Convolution result using ideal "lines" as the kernel across the field of view.

Once we knew where the lines were, the next step was to extract the pixel intensity values from the lines, as described in section 3.2. For each of the twenty-five images, a one-dimensional cut fifty pixels long was taken across the lines and spaces. These pixel intensities were normalized between 0 and 255 and then recorded. The peaks correspond to the light colored areas of the picture, and the valleys represent the dark (i.e. the edges of the lines) parts of the picture. This one-dimensional cut was done for the images resulting from all the different focus/exposure settings. (Actually, the one-dimensional cut was an average of five adjacent cuts on the line; this was done to reduce noise [see section 2.3.1].) Figure 42 shows normalized line intensity plots for different exposure and focus settings. The fifty intensity values for each setting format the input to the neural network, representing one image setting. The corresponding outputs, which indicate how far off each of the settings is from the ideal setting, would be assigned to each of them so that the network could then be trained.



(a) (b) (c)

(d) (e) (f)

Figure 42 Line intensity plots examples of the 1 μm lines-spaces pattern for different exposure and focus settings: (a) good focus, underexposed (b) good focus, good exposure, (c) good focus, overexposed, (d) poor focus, underexposed, (e) poor focus, good exposure, and (f) poor focus, overexposed.

SNNS was used to simulate a feed forward network trained with the backpropagation training algorithm to recognize key aspects of the line pixel intensities patterns for the lines exposed under different stepper settings. The feed forward type of network was chosen because it fit the structure of our problem. (See section 3.2 for more detail on different network types.) The fifty intensity values for each setting were normalized to be between 0 and 1 and were then presented to the 50-node input layer of the neural network as the input values. We used two hidden layers of ten nodes each. The number of hidden nodes was chosen after experimentation to be the fewest number of layers that would produce a small enough error without an over excessive number of cycles.

The eight output nodes represent the different possible settings of focus and exposure. There were five different focus values and five different exposure values. Each of the output nodes would indicate where in the 5x5 focus/exposure matrix the image would be. Since being overfocused by a certain amount would render an image to look exactly like being underfocused by the same amount, only three output nodes were needed to represent the five different focus settings, as it is symmetrical around the best focal setting. The output node assignments for each of the twenty-five different settings were as follows:

Table 3: Neural net outputs for the different focus and exposure settings.

	under exposed by 2 settings	under exposed by 1 settings	perfect exposure	over exposed by 1 settings	over exposed by 2 settings
focus off by 2 settings	100 10000	100 11000	100 11100	100 11110	100 11111
focus off by 1 setting	110 10000	110 11000	110 11100	110 11110	110 11111
perfect focus	111 10000	111 11000	111 11100	111 11110	111 11111

As can be seen from the previous table, if a node was set to 1, all the previous nodes for the parameter (exposure or focus) were also set to 1, introducing a cumulative affect. This was done to capture the continuity of the process within the neural net in order to enhance the learning process [15]. A visualization of an example output of the net is shown in Figure 43.

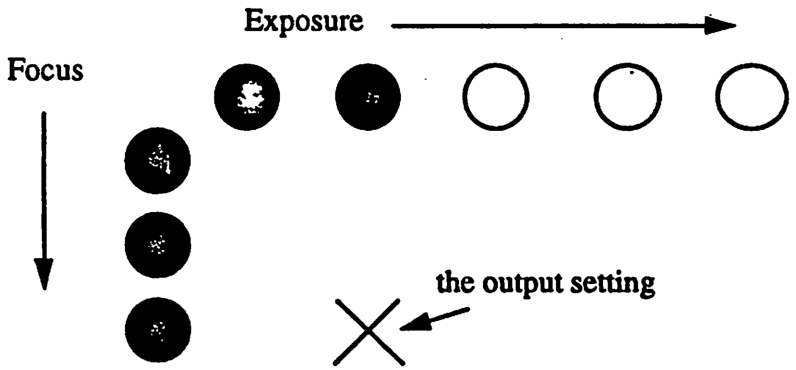


Figure 43 A visualization of an example output of the neural net. Here, the focus and exposure setting is indicated by the darkened circles, which indicate a 1. In this case, the output is indicating that the image is underexposed by 1 setting and is perfectly focused.

The neural network was then trained with 425 example images, representing all the different settings, for 10,000 cycles. While training the network, we were also testing it, to see how well it was learning. There were 100 testing images. It can be seen from Figure 44 that at around 400 training cycles, the testing curve hits its minimum error. The testing curve errors were calculated to be the percent identified incorrectly, where a testing output was considered to be "correct" if the setting was identified exactly right, or if it was off by one step in either focus or exposure. As can be seen in Figure 44, at its optimum state, the trained neural net could identify 96% of the test images' focus and exposure settings correctly. The results are plotted in Figure 45 and Figure 46 which show the correct settings vs. the settings predicted by the neural network for focus and exposure. There were five possible exposure settings and three focus settings.

As was noted previously in this section, the interval step size for the focus settings in this experiment was approximately one Rayleigh Unit of defocus. This is a relatively large amount of deviation from the ideal focus setting. Usually, we would like to be able to detect a deviation of half a Rayleigh Unit of defocus from the ideal. Further experimentation is needed to confirm that the techniques used in this project can accomplish this.

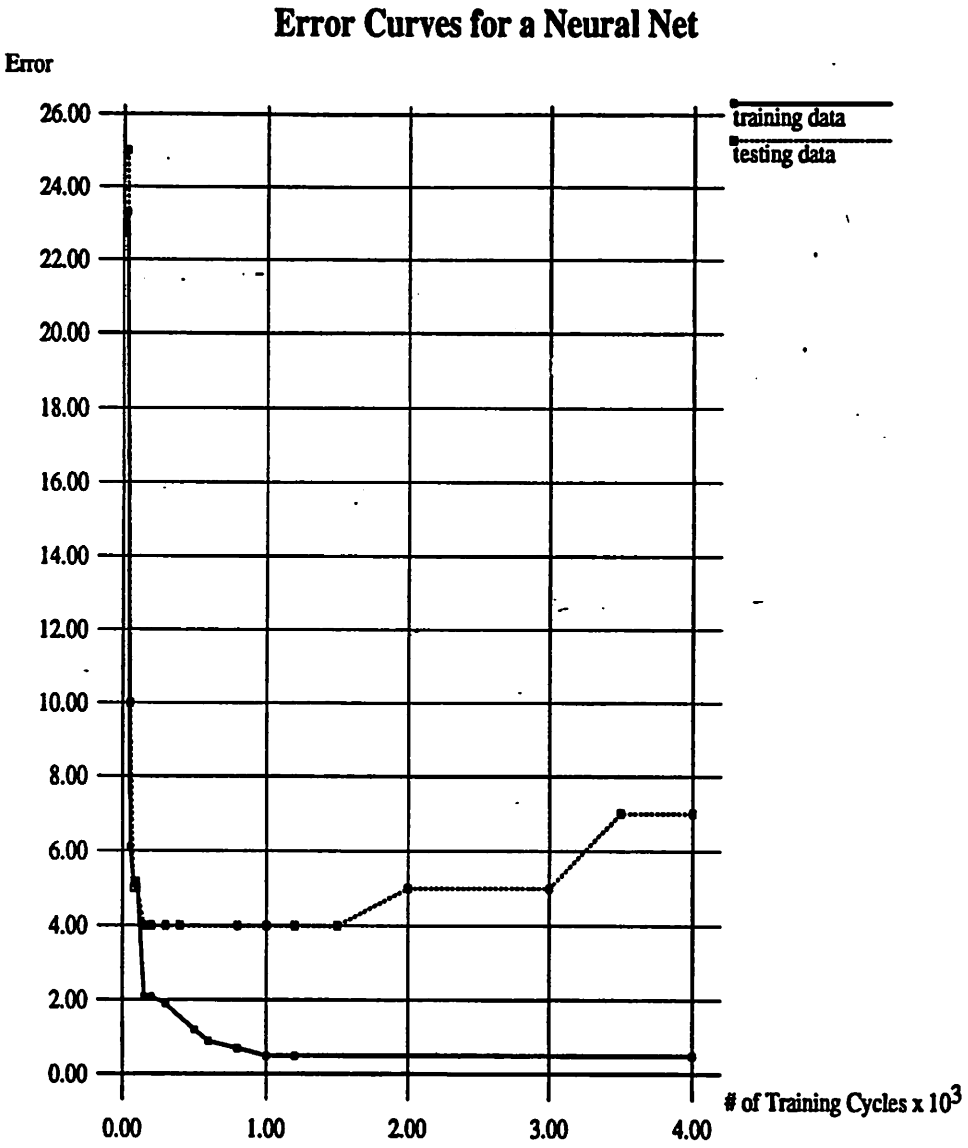


Figure 44 The training and testing error curves for a neural net trained to recognize the focus and exposure conditions of 1 μm lines and spaces.

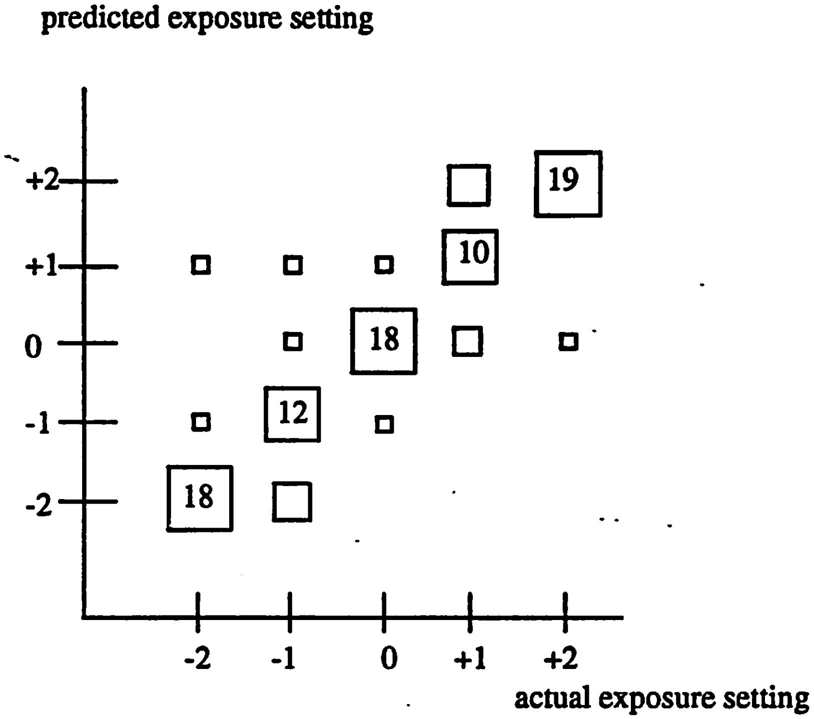


Figure 45 Plot of the exposure settings estimated by the neural net vs. the actual exposure settings. The area of the square is proportional to the number of occurrences. The settings units are as follows: underexposed by 2 settings is -2, underexposed by 1 setting is -1, perfect exposure is 0, overexposure by 1 setting is +1 and overexposure by 2 settings is +2. These tests represent a variety of focal settings.

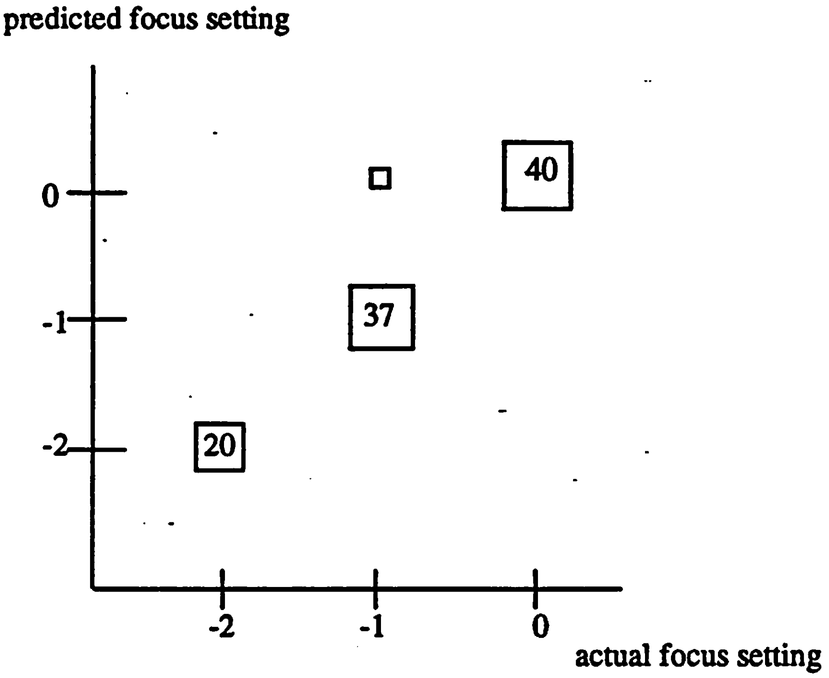


Figure 46 Plot of the focus settings estimated by the neural net vs. the actual focus settings. The area of the square is proportional to the number of occurrences. The settings values are as follows: under or over focused by 2 settings is -2, under or over focused by 1 setting is -1, and perfect focus is 0. These tests represent a variety of exposure settings.

4.4.2 Other Methods

In addition to the 1 μm line-space intensity plots, other methods and features were examined for their potential in identifying focus and exposure settings.

Extracting a Two-Dimensional Area From an Image

Extracting a two-dimensional array of pixel values from a picture of a feature and sending that array of numbers to the neural net is another way of quantifying and simplifying the picture data. The feature on which this method was used is two island squares placed corner to corner, which can be seen in Figure 47. Note that in even the ideal focus and exposure case, because of their small dimension (1 µm x 1 µm), the small squares appear to be rounded. This is because it is reaching the diffraction limits of the GCA 6200 wafer stepper's optical system, which as was mentioned before, can only clearly resolve down to about 0.8 µm lines. We used one wafer that included this two-squares feature on it; this wafer was coated with G-line resist. Twenty-one monitor screens worth of pictures were digitized. However, this feature was very sensitive to changing focus and exposure values; for some of the extreme settings, no pattern was present. Therefore, there were about eighteen images in each picture (as opposed to the expected twenty-five), resulting in a total of 372 individual training images.

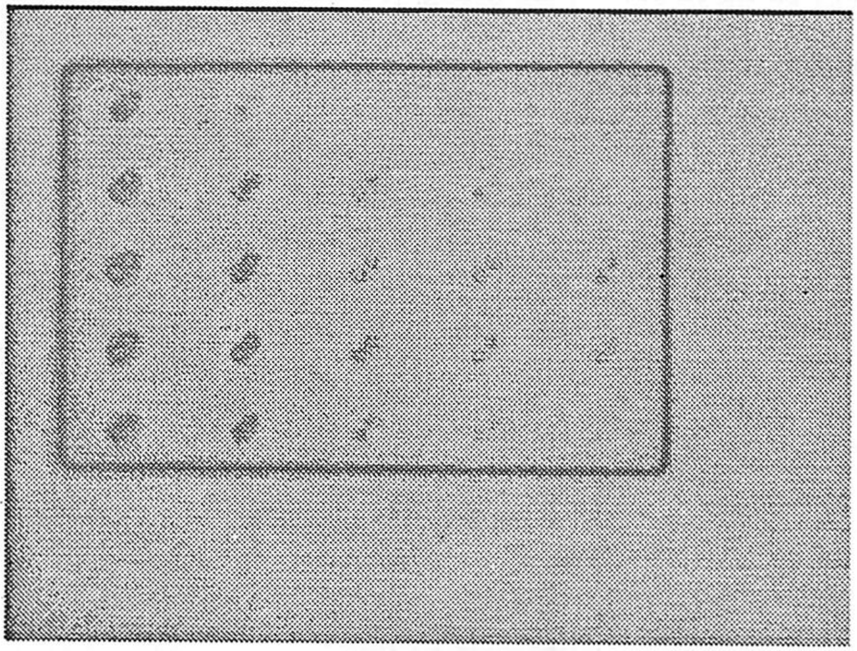


Figure 47 A monitor screen of the 2 squares feature.

As before, the first step was to locate where in the field of view the two-squares feature was so that the appropriate two-dimensional segment could be extracted. To do this, we again used convolution techniques to identify the location of this feature. Once we knew the location of the two-squares features in the picture, we extracted the pixel intensity values, as described in section 3.2. For each of the images, a two-dimensional area of 20x20 pixels was taken. These pixel intensities were normalized between 0 and 255 and then recorded. Figure 48 shows an example of an area segment. This is equivalent to sending the neural net the normalized image directly. This two-dimensional extraction was done for the images resulting from all the different focus/exposure settings. The 20x20 intensity values for each setting (after being normalized between 0 and 1) would then be the input to the neural network, representing one image setting. The corresponding outputs, which indicate how far off each of the settings is from the ideal setting, would be assigned to each of them so that the network could then be trained.

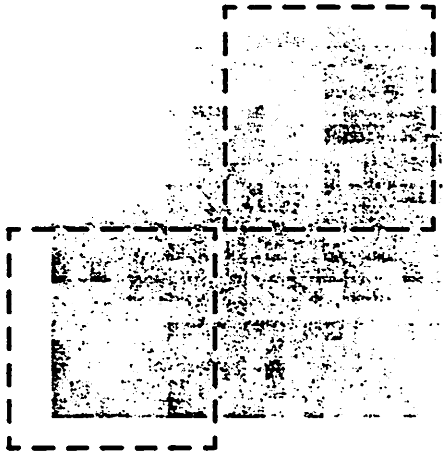


Figure 48 A normalized 200x200 pixel segment of the image of the 2 squares feature. One pixel corresponds to 0.1 µm.

SNNS used to simulate a feed forward network trained by using the backpropagation training algorithm as before. The structure was similar to the one used before, except that now there are 400 input nodes, and 2 hidden layers of 15 nodes each. (For the one-dimensional data from before there were 50 input nodes and 2 hidden layers of 10 nodes each.) This two-dimensional representation is more complex than the previous one-dimensional one, thereby necessitating more hidden nodes. The neural network was then trained with 290 example images, representing all the different settings, for 10,000 cycles. While training the network, we were also testing it against an independent set of patterns, to see how well it was learning. There were 82 testing images. At its optimum state, the trained neural net could identify 81% of the test images' focus and exposure settings correctly.

Convolution Method with Two Squares Feature

Another method that deals with this same feature consisting of the two diagonal squares uses only convolution procedures. (See section 2.3 for a detailed description of convolution.) The convolution will produce a maximum value for the test image that most closely matches the ideal image. We were also interested in not only which test image produced the highest convolution value, but also what the "shape" of the convolution values were for each test image. We recorded an area of convolution values for each test image by considering a 10x10 pixel area of convolution values surrounding the highest convolution value for the image, which would have the peak at the center, and get smaller far from the peak. Each setting would then have a "shape" of 10x10 convolution values associated with it, which could then be sent to a neural network as the input and have corresponding outputs assigned to it. The results of the convolution can be seen in Figure 49. For computational economy, the small picture was used as a convolution kernel over the twenty-five small regions of interest in the big picture (where the features were) and skipped over the "informationless" space between them.

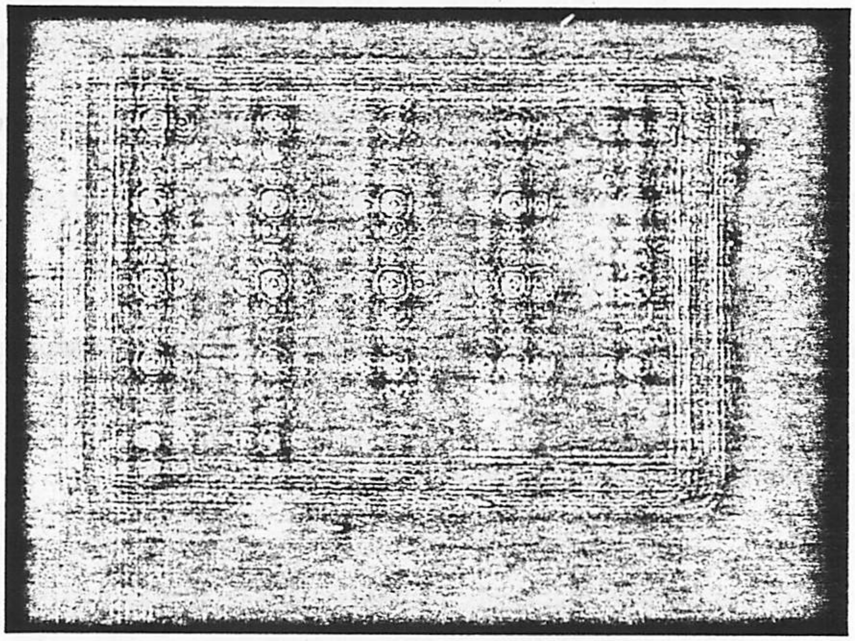


Figure 49 Convolution result of convolving a small 2 squares image with the large image. The black spaces between each of the 25 areas were skipped over, since they convey no information.

Again, we used the twenty-one monitor screens worth of pictures that were grabbed; there were approximately eighteen images per picture (as opposed to the expected twenty-five), resulting in a total of 372 individual images.

This area convolution method was done for the images resulting from all the different focus/exposure settings. The 10x10 pixel array of convolution values for each setting would then be the input to the neural network, representing one image setting. The corresponding outputs, which indicate how far off each of the settings is from the ideal setting, would be assigned to each of them so that the network could then be trained as it was before with the other nets.

SNNS was again used to simulate a feed forward network trained by using the back-propagation training algorithm as before. The structure was similar to the one used before, except that now there are 100 input nodes, and 2 hidden layers of 15 nodes each. The neural network was then trained with 290 example images, representing all the different settings, for 10,000 cycles. While training the network, we were also testing it, to see how well it could generalize. There were 82 testing images. The trained neural net could identify 67% of the test images' focus and exposure settings correctly. It can be seen that this method with this feature did not yield as good results as the other methods.

Convolution Method with Four Squares Feature

The same convolution method was also applied to another feature, the four island square, which is shown in Figure 50. Again, we recorded a 10x10 pixel array of convolution values resulting from convolving an ideal image with each of the test images for each of the images. The data was taken from three wafers, two of which were coated with I-line resist, and the other with G-line resist. A total of twenty-four monitor screens were grabbed, which results in approximately 577 images.

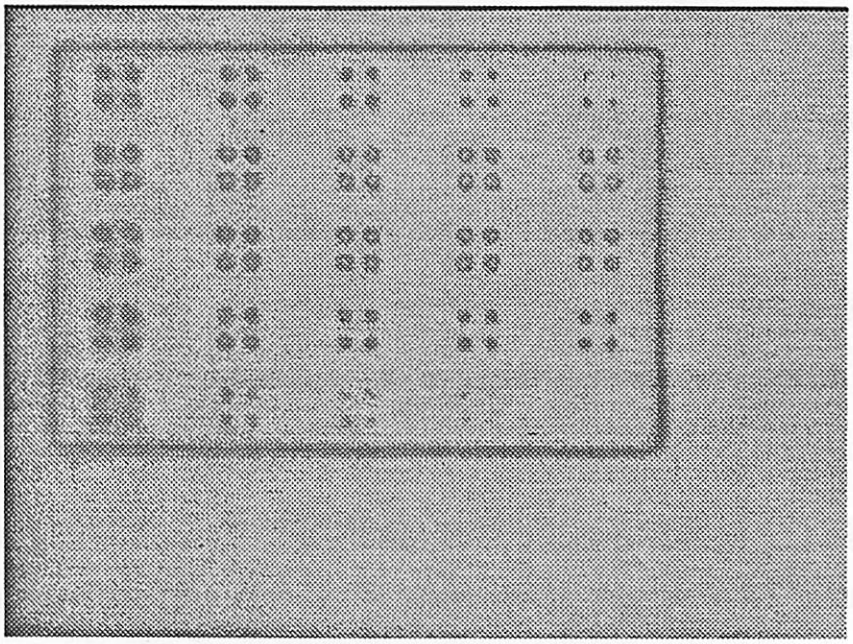


Figure 50 A monitor screen of the 4 squares pattern.

The 10x10 pixel array of convolution values for each setting would again be the input to the neural network, representing one image setting. SNNS was again used to simulate a feed forward network trained by using the backpropagation training algorithm as before. The structure was the same as the one used for the two-squares feature, with 100 input nodes, 2 hidden layers of 15 nodes each, and 8 output nodes. The neural network was then trained with 438 example images, representing all the different settings, for 10,000 cycles. While training the network, we were also testing it, to see how well it was learning. There were 139 testing images. At its optimum state, the trained neural net could identify 70% of the test images' focus and exposure settings correctly.

It is interesting to note that even though two-dimensional features, such as squares, are more sensitive to focus conditions than "line" features that convey one dimensional information [4], the feature with the squares was not as successful as the feature with the lines in this project. This may be because of the specific methods we used to quantize and process the images of the features.

The following table summarizes the results for the various features and methods that were investigated.

Table 4 Summary of testing success rates for different features and methods.

feature	method	chapter reference	# of training patterns	# of testing patterns	testing success rate[1]
1 μm lines and spaces	One Dimensional Pixel Extraction	4.4.1	425	100	96%
2 Squares placed diagonally	Two Dimensional Pixel Extraction	4.4.2	290	82	81%
2 Squares placed diagonally	Convolution	4.4.2	290	82	67%
4 Squares	Convolution	4.4.2	438	139	70%

[1] Testing success rate was defined to be the percentage of settings the neural net estimated "correctly", where a "correct" estimate was if it estimated the settings exactly right or was off by one setting in either focus or exposure.

Chapter 5 Error Analysis for Possible Control Application

Feature size can depend on the focus and exposure conditions of the lithographic wafer stepper. For example, if the resist is overexposed, the linewidth may become smaller than desired because more of the resist will have been exposed (and therefore developed away) than was originally designed for on the mask. On the other hand, if the resist is underexposed, the linewidth may become larger, as not all of the resist would have cleared in the spaces between the lines. The linewidth may decrease if the focus is too far off from its ideal setting because less of the line will have been clearly imaged. See section 1.2 for more detail on how focus and exposure conditions can affect feature size. Focus conditions may also affect the edge taper width of a line of resist. The further away the focus settings are from the ideal focus condition, the more significant the edge taper width. This is because the outline of the feature would be unclearly defined, resulting in a very non-vertical sidewall.

We wanted to know how much the linewidth would vary if it were exposed under the different conditions of the matrix of settings that we had used for our experiment. We also wanted to investigate how well our calibration of these settings would be able to keep the linewidth close to what was desired. We used SAMPLE [17] to simulate what the I-line resist profile would look like under the different focus and exposure settings for a 1 µm line feature. The units of defocus had to be adjusted for the simulations; from previously performed experiments [18], it had been found that for every 8 “divisions” of defocus in the GCAWS, there was 1 µm of defocus in the SAMPLE simulations. (See section 1.2 for a discussion on defocus distance.) We only simulated 3 different focus settings because of the symmetry involved (underfocused images look the same as overexposed images.) Figure 51 shows an example of a SAMPLE profile, and how linewidth was measured to be the width of the line at 50% of the resist height.

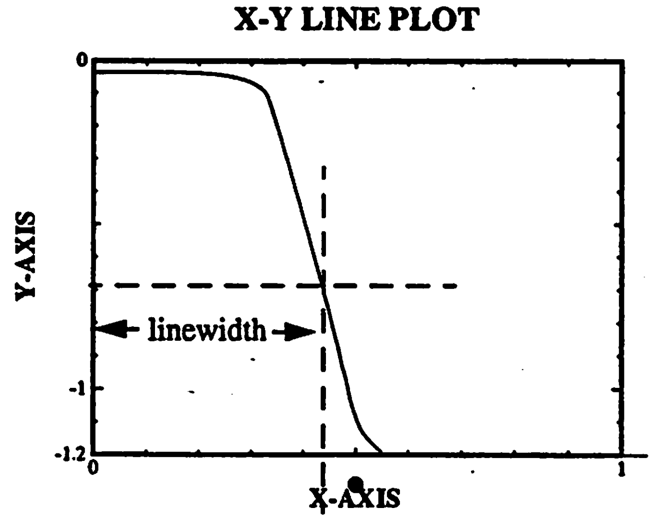


Figure 51 An example of a SAMPLE simulated line-space pattern.

The following table shows a typical range of settings of focus and exposure and what their simulated linewidths (in μm) would be under those conditions. It can be seen that the simulated linewidth decreases as the exposure values increase and as the focus values stray away from the ideal. These results are plotted in the surface plot shown in Figure 52.

Table 5: Simulated lw (μm) for different focus and exposure settings.

	under-exposed by 2 settings	under-exposed by 1 setting	perfect exposure	over-exposed by 1 setting	over-exposed by 2 settings
focus off by 2 settings	0.907	0.887	0.874	0.858	0.842
focus off by 1 setting	0.915	0.904	0.887	0.872	0.860
perfect focus	0.918	0.904	0.890	0.876	0.865

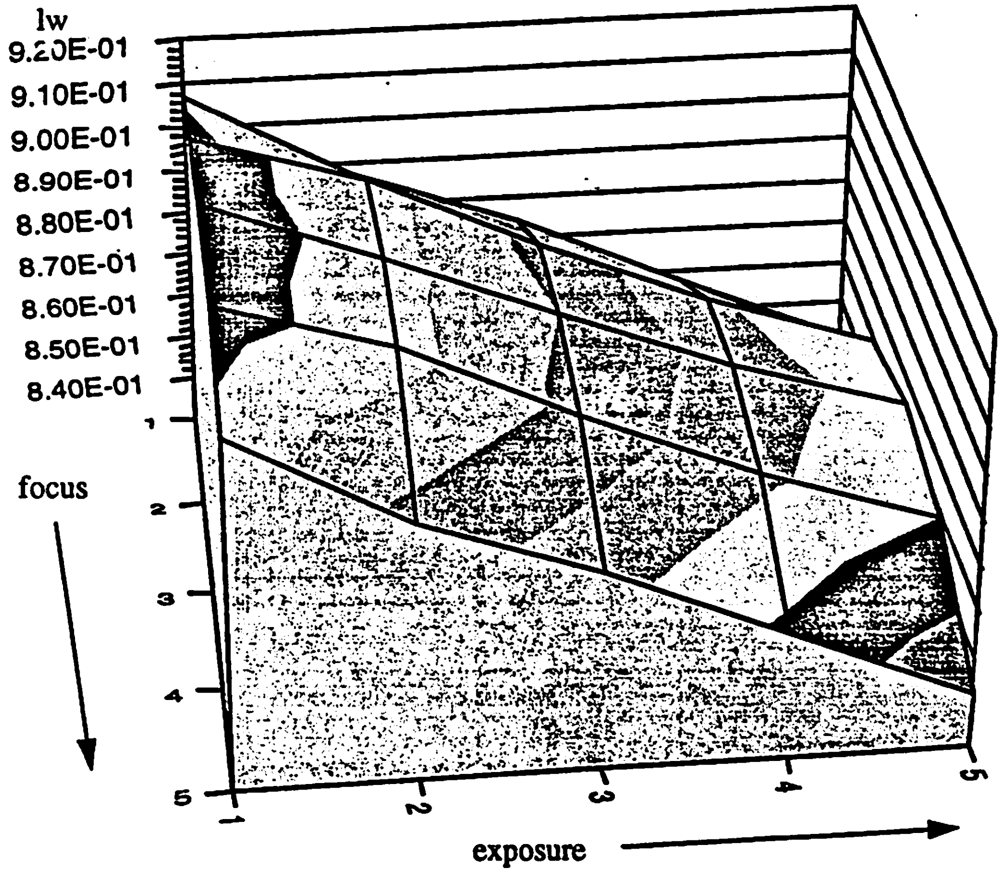


Figure 52 A surface plot of the linewidths as the exposure and focus settings are varied throughout the settings of the matrix.

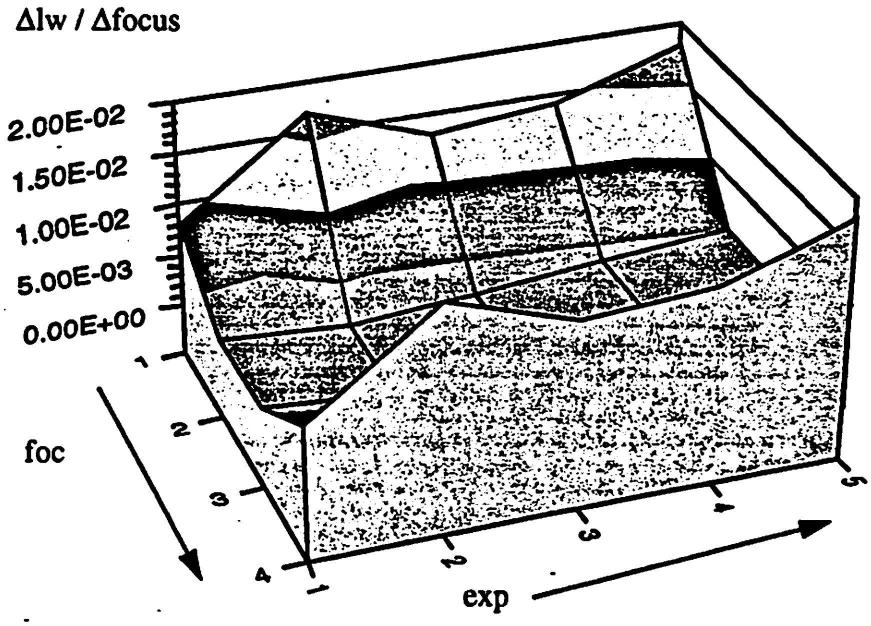


Figure 54 Surface plot of Δlw / Δfocus (as exposure is held constant) as a function of focus and exposure.

Edge taper width was measured to be the horizontal distance from the top of the resist to the extrapolated bottom (on one side of the line only), as can be seen in Figure 55.

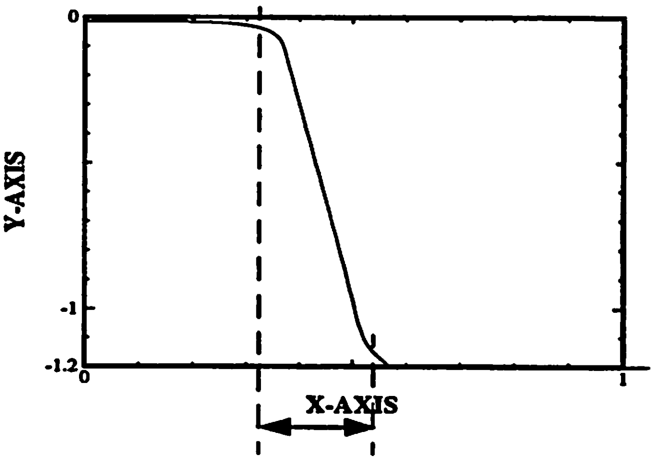


Figure 55 How edge taper width is measured in the simulated profiles.

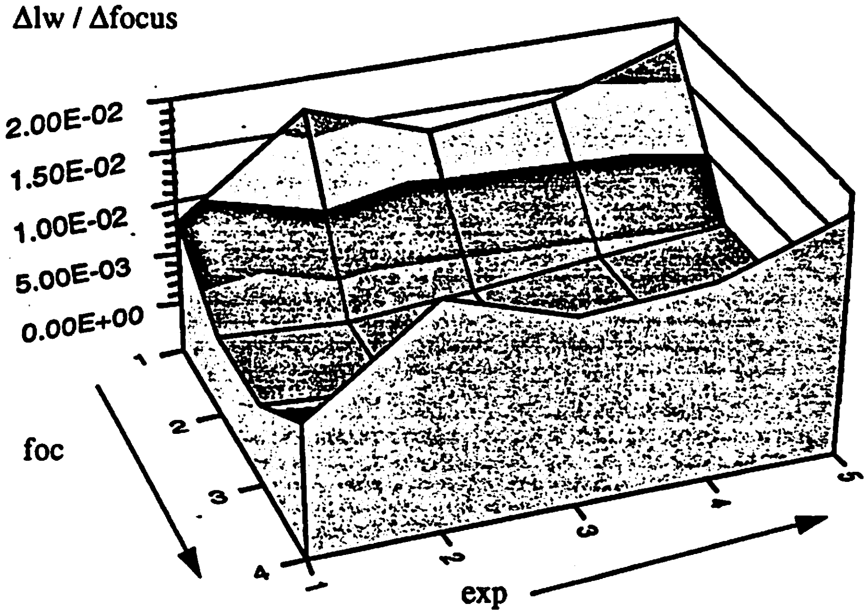


Figure 54 Surface plot of Δlw / Δfocus (as exposure is held constant) as a function of focus and exposure.

Edge taper width was measured to be the horizontal distance from the top of the resist to the extrapolated bottom (on one side of the line only), as can be seen in Figure 55.

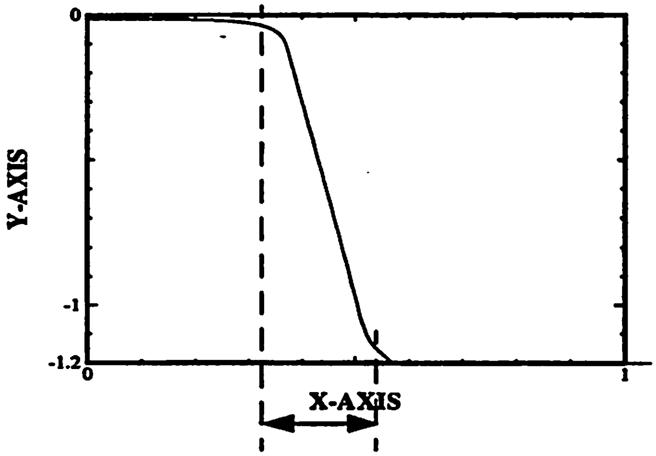


Figure 55 How edge taper width is measured in the simulated profiles.

The following is a table of the edge taper width of the simulated lines for the different settings. It can be seen that the further away the focus settings were from the ideal, the broader the edge taper width. However, these changes in the edge taper widths are rather small; this might imply that using cross sectional SEMs of the resist lines to evaluate focus settings may not be as sensitive as the techniques described in this project. Changing the exposure time settings did not seem to affect the edge taper width.

Table 6 Simulated edge taper width (μm) for different focus and exposure settings.

exp→ foc↓	under-exposed by 2 settings	under-exposed by 1 setting	perfect exposure	over-exposed by 1 setting	over-exposed by 2 settings
focus off by 2 settings	0.22	0.21	0.20	0.21	0.20
focus off by 1 setting	0.19	0.19	0.17	0.17	0.17
perfect focus	0.18	0.17	0.16	0.16	0.17

As was stated previously, our system predicted the settings to be exactly right or off by one interval in either focus or exposure 96% of the time. Therefore it would be able to indicate by how much and in what direction to move in order to get to the ideal settings most of the time relatively accurately. If it were off by one setting interval in either the exposure settings or the focus settings, the linewidth tolerances would be much smaller (i.e. be within 0.015 μm of the desired linewidth) than if the calibration procedure had not been used and the settings were to run the full range of the matrix, which could produce delta linewidths as large as 0.048 μm. So under the current experimental setting, our calibration procedure is sensitive enough that it can identify and correct inappropriate settings that would lead to linewidth changes as little as 0.02 µm. Greater magnification can reduce that number. Also, because convolution methods can be used to identify the exact location of the lines (see section 2.2,) no precise alignment of the image is required. Therefore, these techniques are very inexpensive and convenient to utilize.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

In this project, we investigated an automated method of choosing the best image (and therefore the best setting) when calibrating a lithographic wafer stepper that utilized image processing techniques and neural nets. In order to accomplish this, we used optical microscope images of resist features that were exposed by a lithographic wafer stepper under different focus and exposure settings.

Convolution and pixel intensity extraction were used to pre-process the picture information before it was sent to the network. Feed-forward nets were then trained with the backpropagation algorithm to identify focus and exposure settings for a given image. An image grabbing setup was installed, and a new mask was designed especially for use in the experiments in this project.

Results for selected patterns were presented; the best results, which utilized a one-dimensional pixel extraction technique performed on 1 μm lines and spaces, showed that our system could recognize the focus and exposure conditions within one setting for a given image 96% of the time. It is interesting to note that even though two dimensional features, such as squares, are more sensitive to focus conditions than "line" features that convey one dimensional information [4], the feature with the squares was not as successful as the feature with the lines in this project. This may be because of the specific methods we used to quantize and process the images of the features.

From SAMPLE simulations that were performed in order to investigate how sensitive linewidth was to changing exposure time and defocus values, it was found that by using the calibration techniques described in this project, we can identify inappropriate settings that would lead to linewidth changes of less than 0.02 µm.

6.2 Future Work

There are other applications for the techniques described in this project that are currently being investigated. One further application is the run-to-run monitoring of stepper conditions in a production environment. By incorporating some features on a production wafer, the focus and exposure settings can be evaluated during production. A picture of the features could be taken, and the image could then be processed and sent through a neural net, which could then indicate by how much focus and exposure settings were off. It would then be possible to determine in which direction to move in order to be at the optimum settings. (See Figure 56.)

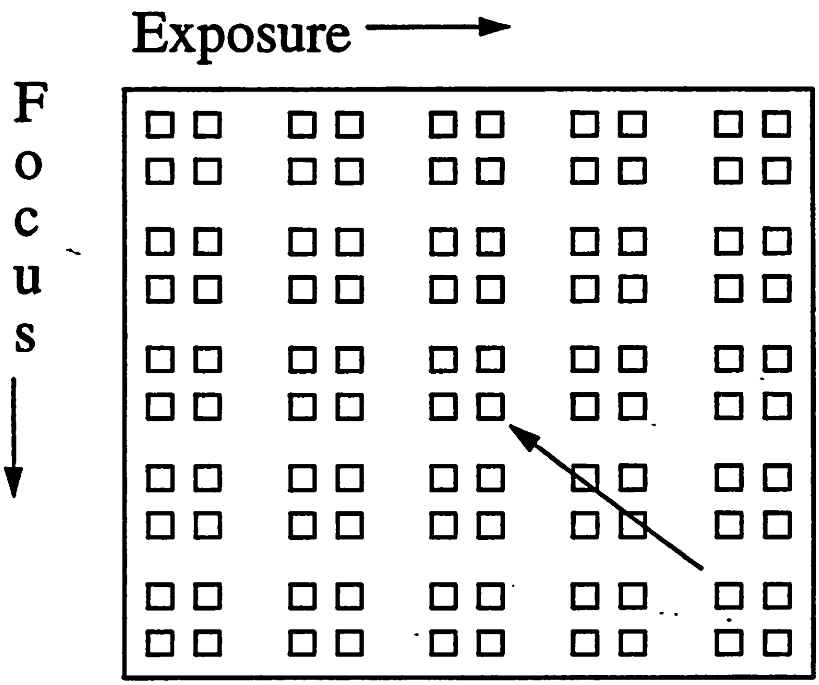


Figure 56 The neural net would be able to identify how far off and in which direction the focus and exposure settings are, and then be able to indicate how to change the settings in order for the stepper to perform optimally.

Also, inexpensive CD measurement applications can be developed using techniques similar to ones described here, namely the line intensity plots. An image of the lines could be grabbed, and one-dimensional cuts could then be taken across the lines to form intensity plots. If a pixel to µm ratio is established, then the number of pixels between the valleys (representing the dark edges in the picture, and therefore the edges of a line or space) could be converted to µm to measure linewidth. (See Figure 57.)

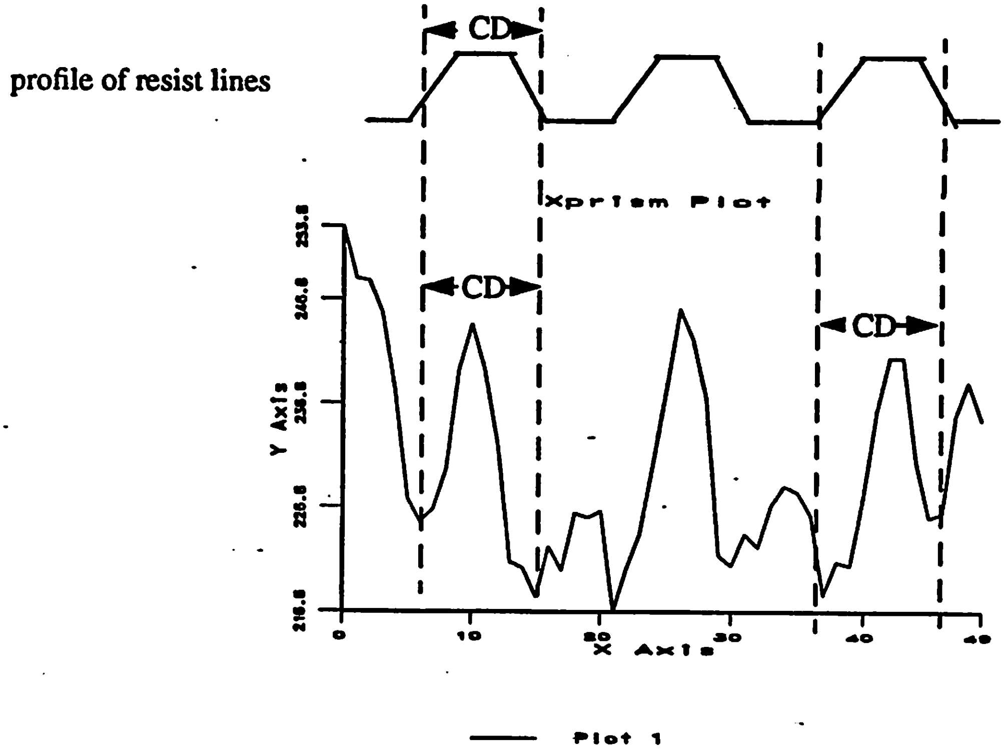


Figure 57 An example of an image intensity plot. Above is the corresponding profile of the resist lines. The cut lines indicate how the edges in the profile correspond to the valleys in the intensity plot, and therefore how CDs of the lines can be measured by looking at the distances between the valleys in the intensity plot.

References

[1] D. Hebert et al, *The Microlab Manual*, University of California at Berkeley, 1994.

[2] R.C. Jaeger, *Introduction to Microelectronic Fabrication*, Addison-Wesley Publishing Co, Reading, MA, 1988.

[3] S. Wolf, R.N. Tauber, *Silicon Processing For the VLSI Era*, Lattice Press, Sunset Beach, CA, 1986.

[4] A. Neureuther, class notes, EE243 Advanced Processing and Layout Class, University of California at Berkeley, 1994.

[5] A.R. Neureuther, W.G.Oldham, *Simulation of Lithography*, University of California at Berkeley, 1986.

[6] J. Helmsen, Private Communication.

[7] GCA Corporation, *Maintenance Training Instruction Manual*, 1984.

[8] The Khoros Group, *Khoros User's Manual*, University of NM, 1991.

[9] J.C. Russ, *The Image Processing Handbook*, CRC Press, Inc., Boca Raton FL, 1992.

[10] A. Stolcke, *The Canny Edge Detector*, CS280 Computer Vision Class, University of California at Berkeley, 1990.

[11] E.W. Scheckler, *A User's Guide for SAMPLE-3D v1.0: Lithography Simulation,* University of California at Berkeley, 1991.

[12] I. Guyon, *Neural Networks and Applications,* AT&T Bell Laboratories, Holmedel, NJ, 1988.

[13] R.P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-21, April 1987.

[14] SNNS Group, *Stuttgart Neural Network Simulator User Manual*, University of Stuttgart, Germany, 1993.

[15] F. Nadi, Private Communication.

[16] R. Reed, "Pruning Algorithms-A Survey", IEEE Transactions on Neural Networks, pp. 740-747, September 1993.

[17] The SAMPLE Group, *SAMPLE User Guide*, University of California at Berkeley, 1991.

[18] S. Leang, Private Communication.