Content-based segmentation of patterned wafer for automatic threshold determination

Pierrick Bourgeat[ab], Fabrice Meriaudeau[b], Patrick Gorria[b], Kenneth W. Tobin[a]
[a]Oak Ridge National Laboratory, P.O.Box 2008, Oak Ridge, TN 37831-6011, USA
[b]Le2i Laboratory – Univ.of Burgundy, 12 rue de la fonderie, 71200 Le Creusot, France

ABSTRACT

This paper introduces a segmentation algorithm suitable for semiconductor wafer images generated by optical inspection tools. The primary application of this work is content-based region segmentation for automatic threshold selection during recipe generation in die-to-die wafer inspection. Structures associated with different functional areas lead to different levels of noise in the difference image during the defect detection process. The ability to automatically create a mask to separate the different structures and materials is necessary to determine local thresholds for each area and thus to improve the signal-to-noise ratio. A supervised segmentation based on the discrete wavelet transform is used to segment a whole die to create a mask. During the inspection, the mask is applied on the difference image, and the threshold is automatically set as a function of the noise within the region and the thresholding coefficient specific to that region. Preliminary segmentation results are very promising. The use of the segmented region in content-based threshold defect detection improves the number of defects detected, and reduces the number of false detections. This paper will show the performance of the segmentation method on optical microscope wafer images, and the subsequent improvement of the defect detection process.
Keywords: Wafer inspection, discrete wavelet transform, segmentation

1. INTRODUCTION

Every year, semiconductor manufacturers spend over $700 million on wafer inspection equipment. As design rules are getting smaller, automated wafer inspection has become a big challenge for the semiconductor industry [1], [2]. This paper introduces a segmentation algorithm suitable for semiconductor wafer images generated by optical inspection tools.

The primary application of this work is content-based region segmentation for semi-automatic threshold selection during recipe generation in die-to-die wafer inspection. In die-to-die wafer inspection, images of corresponding areas on neighborhood dies are aligned and subtracted to detect defects. A threshold level must be determined to optimize the signal-to-noise ratio (SNR), improving the number of correct detections while minimizing false positives and missed detections. Structures associated with different functional areas lead to different levels of noise in the difference image during the defect detection process. The ability to automatically create a mask to separate the different structures and materials is necessary to determine local thresholds for each area and thus to improve the SNR. During the learning operation, the user trains a classifier to identify the different regions and sets a thresholding coefficient for each of those regions. Then, a whole die is segmented to create a mask. During the inspection, the mask is applied on the difference image, and the threshold is automatically set as a function of the noise within the region and the thresholding coefficient specific to that region. Thus, the threshold is based upon the measured noise and the type of region.

Wafer patterns are mostly a composite of repetitive geometric structures with well-known principal orientations. The discrete wavelet transform is efficient in discriminating the local frequency, thus making it appropriate for the segmentation. Preliminary segmentation results are very promising. The use of the segmented region in content-based threshold defect detection improves the number of defects detected, and reduces the number of false detections.

First, we discuss the correction applied to the images to remove the non-uniformity induced by the imaging system and the semiconductor process variations, followed by the features selections using the wavelet transform [3] and the “à trous” algorithm [4], [5]. Next, we describe the stress polytopes classifier [6], and the modifications we introduced for

this particular segmentation work. Then, we give a short description of the method used to apply the segmentation to the defect detection. Finally, we present preliminary results on the segmentation performances, and the defect detection improvement.

2. IMAGE CORRECTION AND FEATURES EXTRACTION

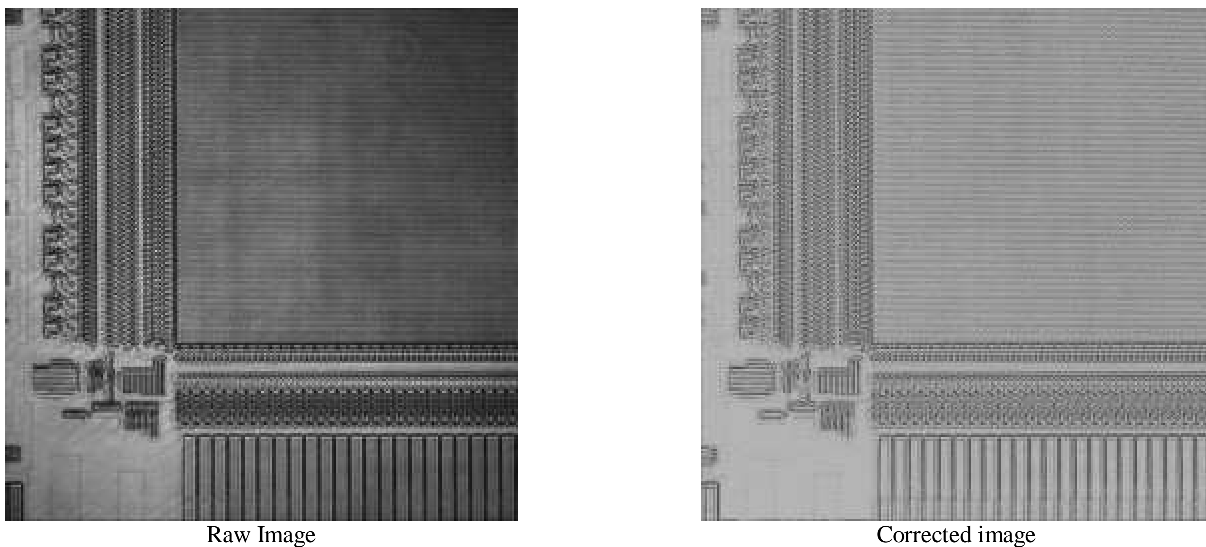
The wavelet transform gives a set of good features to discriminate the different textures of a semiconductor image. It is also impaired by a great sensitivity to variations induced by the imaging system, and small process variations on the semiconductor. Thus, a preprocessing step is required to reduce these variations, and their effects on the features.

2.1. Image preprocessing

The imaging system used for this work is based on a bright field illumination, where the light is focused on the wafer through the imaging objective. This system produces images with a gaussian-type illumination, corresponding to the power distribution of the laser. The resulting images are very bright in the middle, and dark near the borders. This type of illumination creates contrast variations over the image, affecting the wavelet response.

Texture variations are also caused by focus changes due to the height changes on the wafer surface. The auto-focus takes out most of it, but small variations are sufficient to affect the texture. In the same manner, small process variations over the wafer can affect the texture. This type of variations is usually small enough, and can be corrected as a small contrast variation.

To correct the contrast variations and the non-uniform illumination, the image must be normalized. The normalization must be done carefully, so that the level of noise is not increased. If we use the classical method, where for each pixel, the local mean is subtracted, and the result is divided by the local standard deviation, the level of noise is tremendously increased in the resulting image, especially in the areas with no structure where the standard deviation has a very small value. In other words, the noisy areas are divided by a very small number that increase the noise amplitude, and creates an artificial random texture, that makes the segmentation work harder.



Raw Image | Corrected image

Figure 1. Image correction

To perform a meaningful correction, each pixel is divided by the combination of the local mean and standard deviation values. This way, the mean value is modulated by the standard deviation, and for the same mean value, a pixel in an area with a strong contrast is divided by a number slightly bigger than a pixel in an area with a weak contrast. The corrective image is created by summing for each pixel the local mean value and the local standard deviation value in a sliding window. The size of the sliding window is a trade-off between the amounts of contrast correction, and the amount of

information that is removed. It is empirically set at 5 by 5. This way, the contrast variations are corrected as much as possible, and there is not too much information removed from the texture. The input image is divided by this corrective image, which removes completely the illumination variation, and some of the contrast variations (Fig. 1).

2.2. The wavelet transform

After the preprocessing step, the features can be extracted. The wavelet transform has some nice properties [3] that make it appropriate for these types of images. It is based on horizontal and vertical filters that are well designed for the geometric structures found on semiconductor devices. The horizontal and vertical local frequencies can be then be efficiently discriminated. To overcome the translation invariance problem, we use the "à trous" algorithm, where there is no decimation step [4], [5]. It creates an overcomplete decomposition, that leads to some redundancy in the information, but also an invariance in translation, which is a requirement in image segmentation. The image size is kept constant during the whole decomposition, and there is a good spatial localization of low frequencies.

For the wavelet decomposition, two filters are necessary, a low pass filter (LP), and a high pass filter (HP). We employ the Coifman's orthonormal filters [6]. One level of the wavelet decomposition is presented in figure 2.

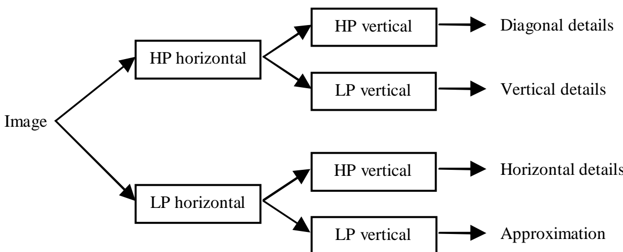


Figure 2. Wavelet decomposition

In the "à trous" algorithm, the decimation step is replaced by an upsampling of the filters at each decomposition level. Practically, for each level, 2^{level}-1 zeros are inserted between each coefficient. The upsampling step is presented in figure 3 on a three coefficients kernel with three levels of decomposition. The number of useful coefficients in the kernel is the same at every level, which results in the same processing time at each level.

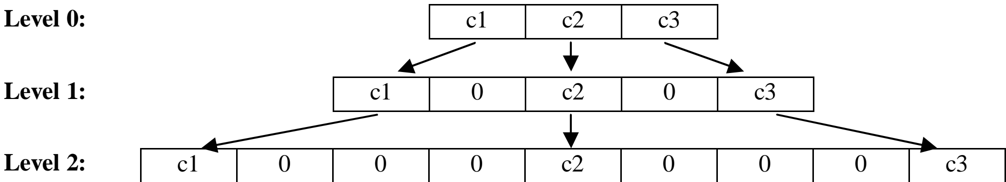


Figure 3. Wavelets kernels for three levels of decomposition

In the particular case of semiconductor images, we use three levels of decomposition, with only the horizontals and verticals details. Diagonal details are not relevant in term of texture discrimination, when the main directions for the structures are horizontal and vertical. Furthermore, diagonals details are obtained after horizontal and vertical high pass filtering, so the result carries most of the noise, which is not useful for the segmentation work.

The wavelet coefficients are hardly usable directly in a classifier, because their waveform does not give a uniform response when a frequency is matched. The response is made uniform by taking the local estimate of the standard deviation. This way, the wavelet coefficients are smoothed and more suitable for the classifier.

3. CLASSIFICATION

In the choice of the classifier, there is a tradeoff between the accuracy of the results, and the corresponding processing time. In-line inspection requires processing a huge amount of data very rapidly, but the accuracy is also extremely important, because misclassifications can lead to misdetection of defects. We found good results with the stress polytopes classifier [6] that can be easily transferred in FPGA architecture to increase its speed. This is a statistical classifier, where the parameter space is clustered in a small set of hypercubes called stress polytopes. These polytopes contain only points of a single class, thus the classification is performed with a couple of comparison to verify the membership with any of these hypercubes. This is a fast operation when the set of hypercubes is small.

3.1. Training

The training of the stress polytopes classifier is a two-step process. For the first step, hypercubes are created around each training samples, so they only include points of the same class. For the second step, hypercubes of the same class are merged together to reduce their number, and speed-up the classification.

During the training process, samples are selected in different areas, and processed though the wavelet decomposition. Each sample becomes a 6-dimensional vector that is used to create the input for the classifier. For each training sample, a hypercube is generated around it, so that it only includes points of the same class, and it does not overlap any hypercubes of different classes.

The boundaries of the hypercubes are described using the L-infinite norm, also called, maximum distance. Considering two vectors $X(x_1, \ldots, x_N)$ and $Y(y_1, \ldots, y_N)$, the L-infinite norm $l(X,Y)$ between the vectors X and Y is defined as:

$$l(X,Y) = \|X - Y\|_\infty = Max_{1 \le p \le N} |x_p - y_p| \tag{1}$$

In each direction, the shortest distances with points of dissimilar class set the distance to the closest neighbor. These distances are multiplied by a coefficient R to set the hypercubes' boundaries. To prevent any overlap between the hypercubes of different classes, R must satisfy the condition (2):

$$0 < R < \frac{1}{2} \tag{2}$$

The value of R is used to set the gap between the hypercubes. With R close to 0, the hypercubes are small, with large gaps between hypercubes of different class. With R close to 0.5, the hypercubes are large, with small gaps between hypercubes of different class. In this case, there is less space available for the fusion to be efficient, and the number of hypercubes may be higher at the end. Formulas (1) and (2) guarantee that there are only samples of the same class inside each hypercube, and that there is no overlap between hypercubes of different class.

In [6], the classifier is only used to separate two classes. During the creation of a hypercube, if no neighbor is found in one direction, the boundary is set to the infinite. When using several classes, it saturates the parameters space, and in some cases, leads to misclassifications. Therefore, a modification of the algorithm is needed to generate finite boundaries.

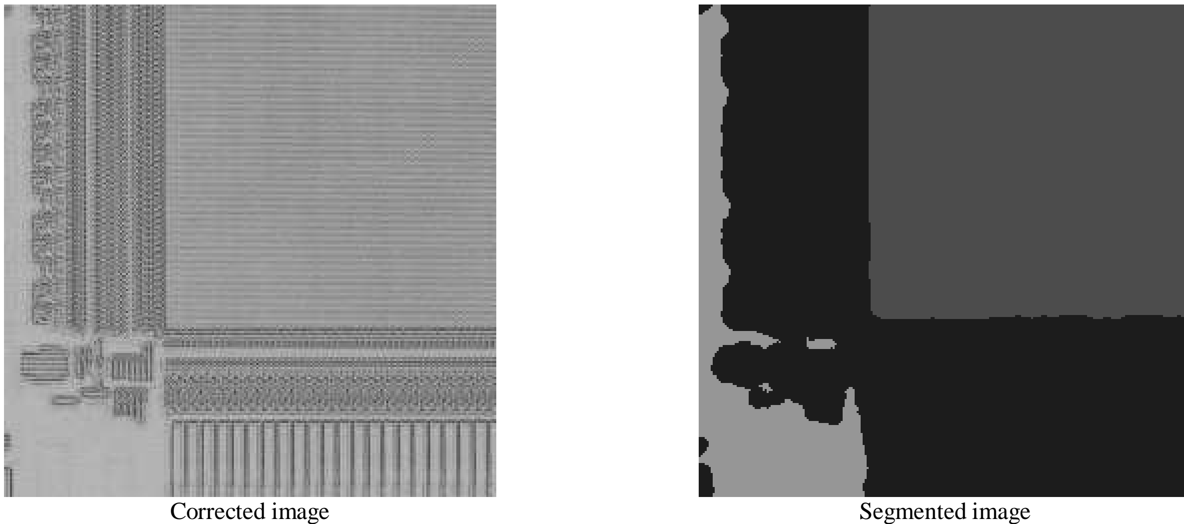
In the cases where no neighbor can be found in a particular direction, an artificial boundary is set using the distance to the closest neighbor in the opposite direction. If there is no neighbor in the opposite direction, then the smallest distance to the closest neighbor in any other direction is used to set the boundary. This way, all the boundaries are defined so that they cannot be at the infinite, and saturate the parameters space.

At this point, the number of hypercubes corresponds to the number of training samples. This number needs to be reduced as much as possible in order to increase the speed of decisions during classification. Inside each class, we try to merge the hypercubes together. Two hypercubes are merged together by defining the smallest hypercube that includes both of them. The fusion is successful if the newly obtained hypercube does not overlap any hypercube of a different class. This step reduces the set to a manageable number of hypercubes.

3.2. Segmentation

The segmentation is performed pixelwise. For each pixel, the corresponding vector is compared with each hypercube, until the membership relation is fulfilled. The membership is verified if every component of the vector is included inside the boundaries of the hypercube. This comparison is simplified by the design of the hypercube, because it is a simple comparison between each parameter value of the vector with the corresponding boundary of the hypercube.

The parameter space is not completely clustered, which means that some points will not be included inside any hypercube. This case should be rare enough, when the classes are well discriminated. Nevertheless, these points have to be assigned to a class. The classification of these points is performed using the Euclidian distance between the vector, and the average vector for each class based on the training samples. The point is assigned to the class with the closest average vector. An example of the segmentation is presented on figure 4.



Corrected image Segmented image

Figure 4. Image segmentation

4. DEFECT DETECTION

The defect detection is performed with a comparison of the same area on two different dies. The images are subtracted, and a threshold is applied to the difference image in order to find the defects. The threshold level is based upon the noise level in the difference image. If no segmentation is performed, then the noise level is measured over the whole image. When several structures leading to different levels of noise are found on the same image, the measure of noise is an average between the noisy areas and the less noisy areas. Therefore, the threshold is not optimum for each area. An optimum threshold is defined so that most of the defects are detected, and few false detections are generated. It is usually better to miss some defects and reduce the number of false detections, because the process of reviewing all the defects to find the real one is extremely time consuming.

With the segmentation, the measure of noise is performed on each segmented region of the image. Therefore, the threshold is set at an optimum level for every type of structure.

During defect detection, the threshold $T(R(C))$ is defined as:

$$T(R(C)) = K(C) * \sigma(R(C)) + \mu(R(C)) \tag{3}$$

With:

$R(C)$: the segmented region of class C
$\sigma(R(C))$: the standard deviation over the region R of class C

$\mu(R(C))$: the mean value over the region R of class C
$K(C)$: the sensitivity coefficient that is preset for the class C

We consider to be a region every part of the image that has been segmented in the same class. In other words, for a given class, the threshold is set using the measure of noise over all the individual parts of the image that belongs to that class. During the defect detection, the operator set the coefficient $K(C)$ for each class, so that the threshold level is optimum for the specified region.

5. RESULTS

In order to test the segmentation algorithm, two different tests are performed. In the first part, the segmentation performances are tested on a real case with the segmentation of the same area over several dies. In the second part, a defect detection is performed in this same area using the previous results of the segmentation.

4.1. Segmentation performances

For this test, we scanned an area of 11 by 11 fields of view (FOV). Each FOV is an image of 432 by 432 pixels. The same area was scanned on 20 different dies over the same wafer. The segmentation is performed pixelwise with three different classes:

1. The DRAM area, which is composed of a fine regular texture
2. The decode bars, which is a coarse, irregular texture
3. The flat area, which does not have any structure

The resulting segmented mosaics were compared with a mosaic manually segmented. The table figure 5 gives the misclassification rate for each segmented die.

Die	Misclassification Rate	Die	Misclassification Rate	Die	Misclassification Rate	Die	Misclassification Rate
0	2.30%	5	2.37%	10	2.24%	15	2.28%
1	1.99%	6	2.53%	11	2.22%	16	2.49%
2	2.57%	7	2.55%	12	2.61%	17	3.51%
3	3.07%	8	3.61%	13	3.69%	18	3.36%
4	2.76%	9	3.41%	14	2.84%	19	2.88%

Figure 5. Misclassification rate on several dies within the wafer

The average misclassification rate is 2.76% on this example. Most of the misclassification occurs at the boundary between two different textures.

4.2. Defect detection with segmentation performances

A memory area, is composed of a grid of decode bars, and inside each cell, a DRAM area is located. The DRAM area leads to little noise in the difference image, but the decode bars leads to very noisy area. For this test, we are only looking for defects in the DRAM area, and we ignore the decode bars, as well as the flat areas surrounding the memory blocks. The tests were performed using the previously segmented areas, and the graph of figure 6 gives the number of real defects against the total number of detection with and without segmentation for different value of $K(C)$. This curve is not the best way to represent the performances, because there are actually some real defects in the decode bars that are detected without the segmentation, but we do not have information on their location. Nevertheless, their numbers should be in the same range as the ones in the DRAM area.

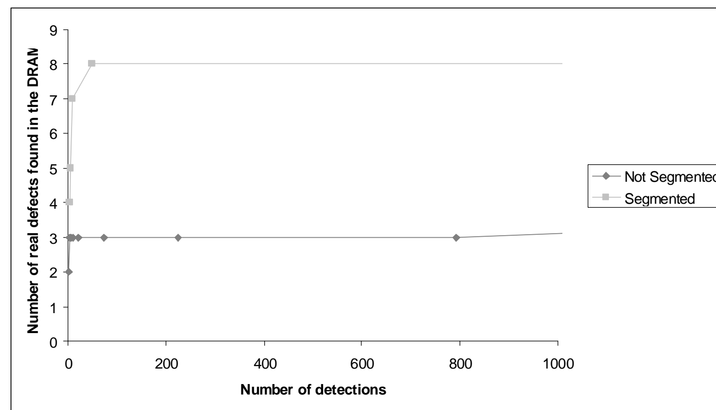


Figure 6. Defect detection performances with and without the segmentation

The graph of figure 6 shows that more defects can be detected in the DRAM area. The biggest one can be found with and without the segmentation, but the smaller defects can only be detected with the segmentation, without generating a huge amount of false detections.

6. CONCLUSION

In semiconductor inspection, the performance of the defect detection depends on several parameters: the number of real defects detected, the number of false positives generated, and the amount of time needed to perform the inspection. The ratio between the number of real defects found, and the number of false positives can be optimized using a mask manually created by an operator, but this operation is extremely time consuming, and reduces the overall performance of the tool. The use of the segmentation for content-based threshold defect detection, gives the same results in a smaller amount of time, thus improving the overall performances of the tool.
Very good results were obtained on this particular type of wafer. Nevertheless, the classifier may perform poorly with coarse or irregular textures, and some more work is needed to make the algorithm less sensitive to the type of texture.

7. REFERENCES:

1. K.W. Tobin, “Inspection in Semiconductor Manufacturing”, Webster’s Encyclopedia of Electrical and Electronic Engineering, vol. 10, pp. 242-262, Wiley & Sons, NY, NY, 1999.
2. “The National Technology Roadmap for Semiconductors: Technology Requirement”, Semiconductor Industry Association, 2001.
3. M. Unser, “Texture Classification and Segmentation Using Wavelet Frames”, IEEE Transactions on Image Processing, Vol. 4, n°11, pp 1549-1560, November 1995.
4. M. J. Shensa, “The Discrete Wavelet Transform: Wedding the A Trous and Mallat Algorithms”, IEEE Transactions on Signal Processing, vol. 40, pp 2464-2482, October 1992.
5. P. Dutilleux “An Implementation of the algorithme à trous to compute the wavelet transform”, in Wavelets: Time Frequency Methods and Phase Space. Berlin: Springer IPTI, pp 298-304, 1989.
6. J. Miteran, P. Gorria, M. Robert, "Classification géométrique par polytopes de contraintes. Performances et intégration", Traitement du Signal, vol. 11, n°5, pp 393-408, 1994.
7. J. Tian, “The Mathematical Theory and Applications of Biorthogonal Coifman Wavelet Systems”, Ph.D. Thesis, Rice University, February 1996.