Patterned Wafer Segmentation

Pierrick Bourgeat[ab], Fabrice Meriaudeau[b], Kenneth W. Tobin[a], Patrick Gorria[b]

[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 is an extension of our previous work on the image segmentation of electronic structures on patterned wafers to improve the defect detection process on optical inspection tools. Die-to-die wafer inspection is based upon the comparison of the same area on two neighborhood dies. The dissimilarities between the images are a result of defects in this area of one of the die. The noise level can vary from one structure to the other, within the same image. Therefore, segmentation is needed to create a mask and apply an optimal threshold in each region. Contrast variation on the texture can affect the response of the parameters used for the segmentation. This paper shows a method to anticipate these variations with a limited number of training samples, and modify the classifier accordingly to improve the segmentation results.

Keywords: Wafer inspection, wavelet transform, segmentation, thresholding.

1. INTRODUCTION

As semiconductor device density and wafer area continue to increase, faster and more sensitive automatic inspection tools are required. The size of the defects is becoming smaller, and harder to detect [1], [2], [3] and [4]. This paper introduces an improvement of our previous work [5], [6], on the image segmentation of electronic structures on patterned wafers to improve the defect detection process on optical inspection tools.

Die-to-die wafer inspection is based upon the comparison of the same area on two neighborhood dies, using the assumption that they are identical, except for the defects. The dissimilarities between the images are a result of defects in this area on one of the die. The two images are subtracted, and a threshold level is selected to locate any abnormality. This threshold is established upon the noise level in the difference image, to improve the signal-to-noise ratio. The noise level can vary from one structure to the other, within the same image since multiple structures coexist in the field of view. Therefore, the measure of noise within the whole image is not relevant for each individual type of structure. Segmentation is needed to create a mask of these different regions. This mask is then used to produce a measure of noise for each structure in the difference image, and apply an individual threshold in each region.

For this work, segmentation is performed using the discrete wavelet transform [7] and the "à trous" algorithm [8], [9], and [10]. This algorithm is well adapted to discriminate local frequencies of the repetitive pattern, and it is restricted to principal directions that correspond to the geometric patterns found on integrated circuits. The weakness of this method is its sensitivity to contrast variation and small texture variation. In our previous work [5], a local correction was applied to remove the non-uniformities. This is sufficient in the case of small variations. However, in some cases where the variations become very important like large process variation or bad focus selection, the classifier needs to be trained with many different samples that cover all the variations contained within the die.

The usual way to train a classifier on this type of data is an empirical approach. The classifier is trained with randomly selected samples, and then is tested over the whole set of data. The areas where the classifier performs poorly are used to extract new training samples. These new samples are added to the original set to retrain the classifier, until the best performances are obtained. This method is not realistic in the in-line inspection process when dealing with huge amount of data. It would require storing all the images of a die to process them off-line, and therefore a huge amount of memory would be needed. Meanwhile, it is time consuming for an operator to go through the iterative cycle of training the classifier, and testing its performances until they become acceptable. This paper introduces an original method to

anticipate the impact of the variation on each feature, and modify the classifier consequently to accommodate these variations. 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 and the "à trous" algorithm. Next, the stress polytopes classifier [11], [12] is described, as well as the modifications introduced to correct the variations.

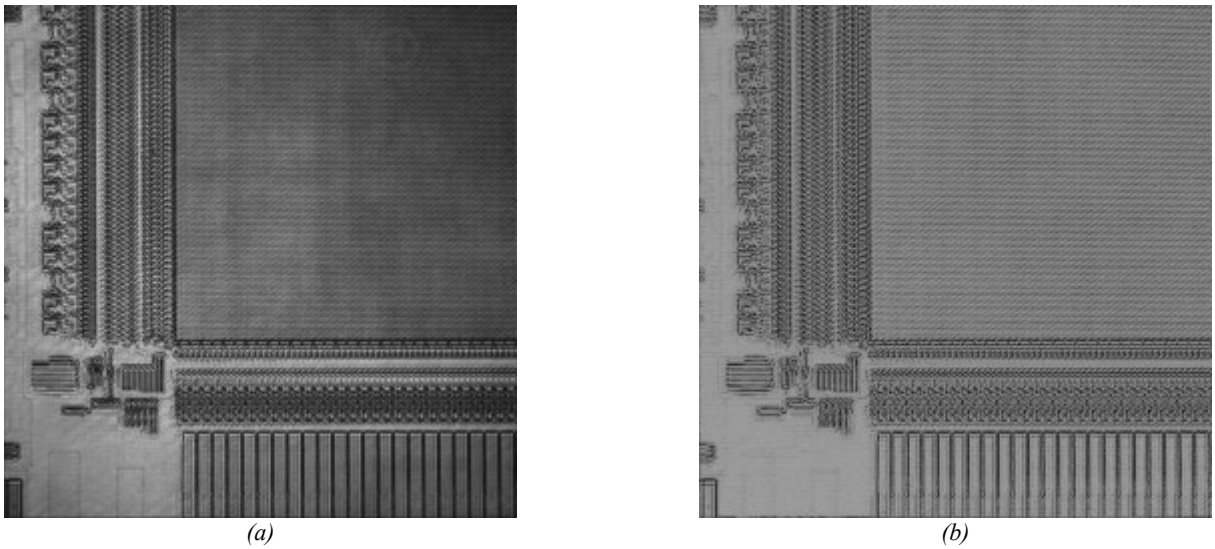
2. IMAGE CORRECTION AND FEATURES EXTRACTION

The selection of good features is one of the most important parts of the segmentation work. The wavelet transform produces good discrimination between the different structures, but is also extremely sensitive to the variations induced by the imaging system and the process variations. Therefore, it is very important to correct the images before the feature extraction, to remove these variations as much as possible.

2.1. Image preprocessing

The bright field illumination used in the imaging system produce a gaussian type illumination, where the image is brighter in its center, and darker near the borders. This non-uniform illumination creates contrast variations over the image that can change the wavelet response. The texture of the electronic structures is also affected by focus changes, and process variation over the wafer. When small enough, they produce a contrast variation that can be partially corrected.

The normalization is performed using the statistics in the neighborhood of each pixel. The mean value is used to correct the illumination, and the standard deviation is used to correct partially the contrast variation. Each pixel is divided by the sum of the mean and the standard deviation values in a 5 by 5 sliding window. The illumination variation as well as some of the contrast variation is removed (**Figure 1**).



(a) *(b)*

Figure 1. Image correction with (a) the raw image, and (b) the corrected image

2.2. The wavelet transform

Once the images are normalized, the features can be extracted. The wavelet transform [7] is based on horizontal and vertical filtering that is well designed for the geometric structures found on semiconductor devices. It allowed an efficient discrimination of the horizontal and vertical local frequencies.

The "à trous" algorithm is used to process a fast wavelet transform that is translation invariant [8], [9], and [10]. Unlike the classical wavelet decomposition, where the image is sub-sampled at each decomposition level, the "à trous" algorithm works with a constant image size, and the filters' kernel is up-sampled by adding zeros between the coefficients. It creates an overcomplete decomposition, that leads to some redundancy in the information, but also an invariance in translation, which is a requirement in this application. The image size is kept constant during the whole decomposition, and there is a good spatial localization of low frequencies.

Two filters are necessary to perform the wavelet decomposition: a low pass filter (LP), and a high pass filter (HP). Coifman's orthonormal filters are used [13]. **Figure 2** shows one level of the wavelet decomposition.

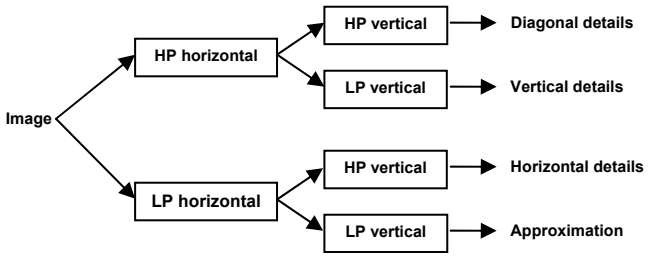


Figure 2. Wavelet decomposition

Between each decomposition level, the filters are up-sampled to match different frequencies. Practically, for each level, 2^{level}-1 zeros are inserted between each coefficient (**Figure 3**). Thus, the number of meaningful coefficients to convolve with the image is kept constant during the whole decomposition, so the processing time is almost the same for each level, allowing fast wavelet decomposition.

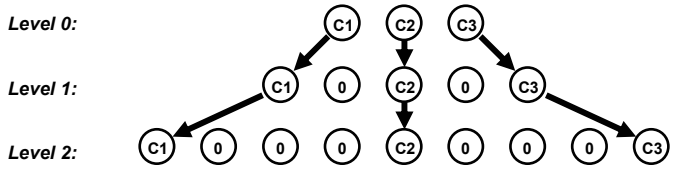


Figure 3. Wavelets kernels for three levels decomposition

Semiconductor images are mostly a composite of geometric structures with horizontal or vertical orientation. For that reason, only horizontal and vertical details are kept for the segmentation. Diagonal details are ignored, because they are not good discriminating features, and they carry most of the noise. Three levels of decomposition are used to produce six different features.

Because of their waveform, the wavelet coefficients can not be used directly in a classifier. That is why the local estimate of the wavelet standard deviation is used in a 17 by 17 neighborhood as a texture feature. This is a good way to smooth the wavelet response, and obtain a uniform response when a frequency is matched, which is more suitable for the classifier.

3. CLASSIFICATION

The classification is performed using the stress-polytopes classifier [11], [12]. This is a statistical classifier designed for high dataflow segmentation. The parameter space is clustered in a small set of hypercubes called stress-polytopes, where each hypercube contains samples of a single class. Thus, the classification of a new point is performed by comparing its features with the boundaries of the hypercubes to verify a membership relation. When the set of hypercubes is small, this is a very fast way to perform the classification.

3.1. Training1

Once the features are extracted, each training point is associated with 6-dimentional vector. These vectors form the input to the classifier for the training. The 6-dimentional parameter space is sliced in hypercubes corresponding to the different classes. A hypercube is created around each training point, so that it only includes points of the same class. This is performed using the distance to the closest neighbor of dissimilar class for each side of the hypercube. The distance *l(X, Y)* between two vectors $X(x_1, \ldots, x_N)$ and $Y(y_1, \ldots, y_N)$ is measured using the L-infinite norm, also known as maximum distance:

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

For each training point, the distance *l* is measured with every other point of dissimilar class. In each direction, the shortest distances set the boundaries of the hypercube. In *N* dimensional space, *2N* boundaries are enough to define a hypercube. To prevent any overlap between the hypercubes of different classes, these distances are multiplied by a coefficient *R* that must satisfy the condition (2):

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

The hypercubes are fully described once there are two boundaries in each dimension defined by a neighbor in each direction. This means that with *N* parameters, there are 2*N* directions where to look for a neighbor. If no neighbor is found in one direction but there is a closest neighbor in the opposite direction, then this distance is used in both directions. Otherwise, if there are neighbors in one dimension, then the shortest distance in every other dimension is used; as a result, limits cannot go to the infinite and saturate the parameter space. Using (1) and (2) guarantees that there is only one class per hypercube, and there is no overlap between the hypercubes of different classes. This results in clustering the parameter space in hypercubes with fixed boundaries.

Dealing with fixed boundaries in the classifier allows processing the data very quickly, but it removes any flexibility. It is not a real problem when the parameters response is consistent for a given texture, but in most application, unexpected variations in the parameter response can affect the performances. This is usually resolved by increasing the number of training samples, and go through a cycle of retraining and testing the classifier until the expected performance is reached. In the particular application of image segmentation of electronic structures, variation of the parameters can be correlated with the residual contrast variations that have not been completely corrected. In other words, for a given texture, some parameters will follow the contrast variation, but not all of them. It is technically difficult to train the classifier with a very large set of samples since there is no guaranty that the whole range of contrast variation will be represented. Therefore, a technique was developed to modify the hypercube's boundary according to the anticipated variations of the parameter for each class.

Since the contrast variation is produced by the variations induced by the system, the measure of correlation between the contrast and a particular feature is a good way to determine if the feature will be sensitive to focus and process variations. This measure will then be used to modify the boundaries of the hypercubes. When a parameter has a strong correlation with the standard deviation for a given class, it will be more likely subject to variations itself (Figure 4). That is why the corresponding hypercube needs to be enlarged in the direction corresponding to this parameter to allow more variation of the parameter. Similarly, if the correlation is weak, the parameter will be less likely subject to variations (Figure 5). Therefore, the corresponding hypercube can be reduced in the corresponding direction.

Figure 4 and Figure 5 show the difference of correlation with the standard deviation for two parameters on the same set of four different samples.

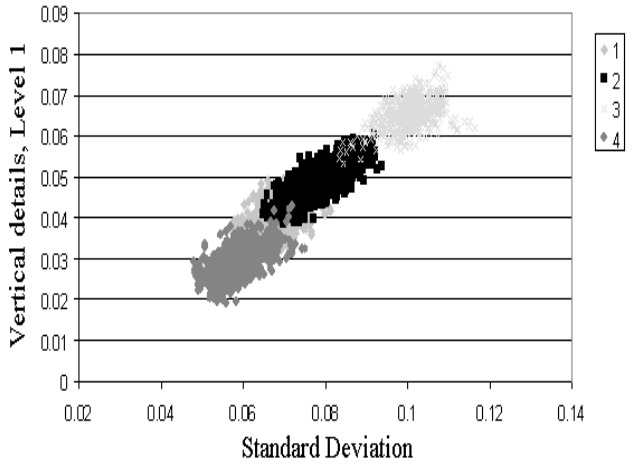


Figure 4. Representation of one parameter versus the standard deviation corresponding to 4 samples of the same texture taken on 4 different images.

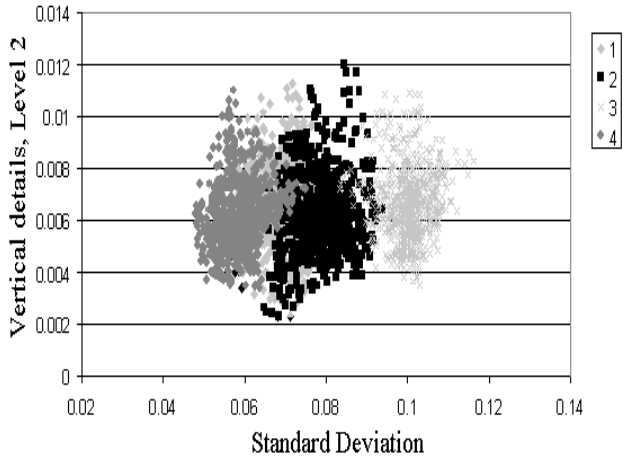


Figure 5. Representation of one parameter versus the standard deviation corresponding to 4 samples of the same texture taken on 4 different images.

This correction can be accomplished by making R a function of the correlation factors, the parameters, and the classes. This needs to be done carefully to insure that there is only one class per hypercube, and that no overlap between the hypercubes of different classes exists. Actually, for each parameter of each class, the expansion/retraction must be done according to the correlation coefficients of the neighbors of other classes. For the same parameter, the correlation coefficient can be extremely different from one class to the other one.

During the hypercube's creation step, the class of the closest neighbor is recorded. This way, the class of each neighbor in each dimension is known. The boundary between two neighbor points in one dimension is defined using the centroid of mass between the respective correlation coefficient.

Considering a vector X of class C_X, and a vector Y of class C_Y, (2) can be rewritten to fulfill the requirement with the corrected coefficient $R_C(x_p, y_p)$:

$$0 < R_C(x_p, y_p) + R_C(y_p, x_p) < 1, \forall p \tag{3}$$

With $R_C(x_p, y_p)$ the coefficient for the parameter p of class C_X with its closest neighbor of class C_Y.

To perform the correction, a weight $W_C(w_{c_1}, \ldots, w_{c_N})$ is assigned to each class. This weight is inversely proportional to the correlation with the standard deviation:

$$w_{c_P} = \frac{1}{\left| r_{c_P} \right|} \tag{4}$$

r_{c_P} : Correlation factor between the parameter p of class C, and the standard deviation ($0 < \left| r_{c_P} \right| < 1$)

Given a vector X of class C_X, and a vector Y of class C_Y, for each parameter p, the distance ratio $d(x_p, y_p)$ between x_p, and the centroid of x_p and y_p can be defined as:

$$d(x_p, y_p) = \frac{w_{y_P}}{w_{x_P} + w_{y_P}} \tag{5}$$

$$\Rightarrow d(x_p, y_p) = \frac{\left|r_{x_P}\right|}{\left|r_{x_P}\right| + \left|r_{y_P}\right|} \tag{6}$$

It is easy to show that:

$$d(x_p, y_p) + d(y_p, x_p) = 1 \tag{7}$$

Requirement (3) can be fulfilled using (7) and *R* defined as (2). The corrected coefficient $R_C(x_p, y_p)$ is described as:

$$R_C(x_p, y_p) = 2 \cdot R \cdot d(x_p, y_p) \tag{8}$$

$$R_C(x_p, y_p) = 2 \cdot R \cdot \frac{\left|r_{x_P}\right|}{\left|r_{x_P}\right| + \left|r_{y_P}\right|} \tag{9}$$

The coefficient $R_C(x_p, y_p)$ provides a way to control the expansion/retraction of the hypercubes so that two hypercubes of different classes will not overlap. Any expansion in a given direction is followed by a proportional retraction in the same direction of the hypercubes corresponding to the neighbor points of another class. Since the correction only affect the coefficient used to set the boundaries, the hypercubes will not be reduced to a point where they would exclude training points of the same class, or expanded to a point where they would include training points of another class.

Once all the hypercubes are created, they are merged inside each class. Two hypercubes are merged together if the newly created hypercube does not overlap with a hypercube of another class. This step reduces manageable number of hypercubes.

The full process is illustrated in Figure 6, Figure 7 and Figure 8 for the two dimensional and two classes' case. First, the hypercubes are defined for both classes using (1) and (2) as shown in Figure 6. Then in the case were the parameter p_1 of the class C_1 has a strong correlation with the contrast, but in the same time, the parameter p_1 of the class C_2 has a weak correlation with the contrast, then hypercubes of class C_1 can be extended in the p_1 direction, while hypercubes of class C_2 can be retracted in the same direction (Figure 7). This way, parameters of class C1 can afford more variation in the p1 direction. After this step, the hypercubes can be merged inside each class (Figure 8).

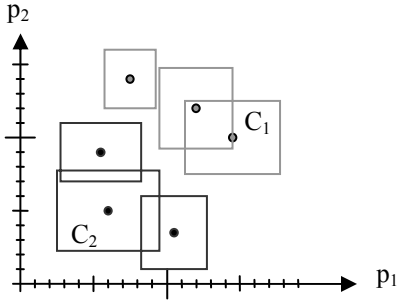


Figure 6. Define the hypercubes using the distance to the neighbor of different class

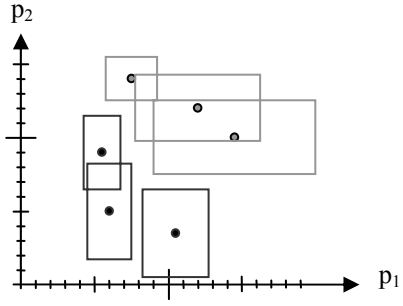


Figure 7. Correct the hypercubes using the correlation information

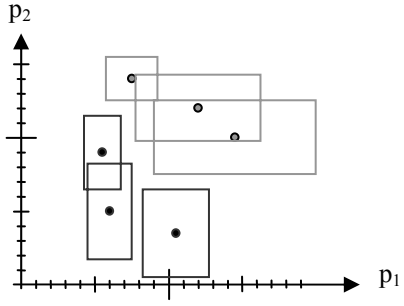


Figure 8. Merge the hypercubes within each class

3.2. Segmentation

During the classification process, each point is tested with the hypercubes to verify the membership relation. The hypercubes do not cluster the whole space; therefore, some points do not belong to any hypercubes. In that case, an artificial point for each class, which corresponds to the mean of the training points of the same class, is used. The closest neighbor sets the class of the point.

4. RESULTS

The segmentation tests were performed on a memory wafer, where 3 different areas need to be segmented:

- the DRAM (**Figure 9**) area which is a fine regular texture that shows a lot of process variations
- the logic area (Figure **10**) which is a composite of coarse textures
- the blank area (Figure **11**) which does not contain any structure

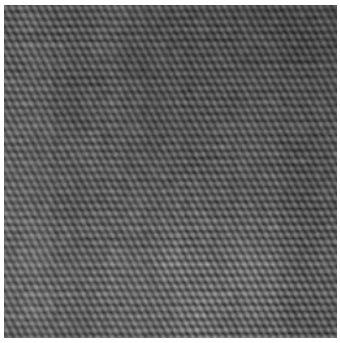


Figure 9. DRAM Area

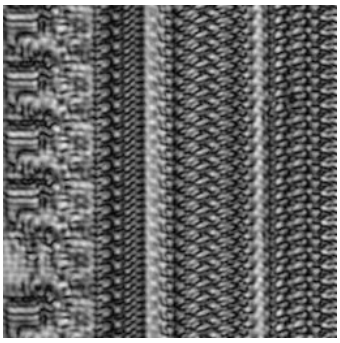


Figure 10. Logic Area

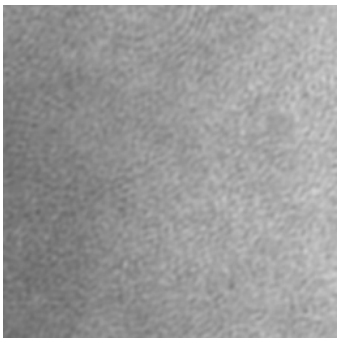


Figure 11. Blank Area

Since the Dram shows most of the variations, the test is done by comparing the results obtained with the same training sample for the logic and the flat area, but different samples for the DRAM in locations where the contrast is different. In each case, a single sample on the DRAM area is used. Two samples of the logic area are needed to include two different types of structures and one sample on the blank area. Images for the segmentation come from a column of 170 images and a row of 170 images taken across three dies (Figure **12**) to include the maximum contrast variation. Each image is 472 by 472 pixels, and the segmentation is performed pixelwise (Figure **13**). The results for the two training sets with and without correction are presented in **Table 1**. Training sets 1 and 2 are also combined for comparison purposes.

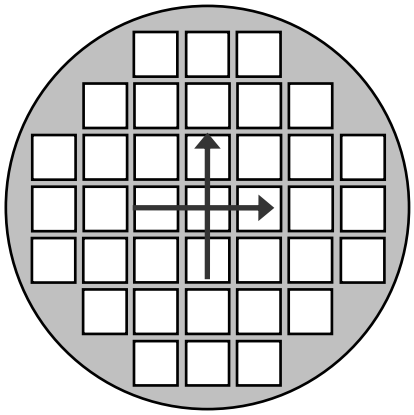


Figure 12. Inspection layout

Figure 13. Example of the pixelwise segmentation on one image

Misclassification Rate	***Column***			***Row***		
	Training 1	Training 2	Training 1&2	Training 1	Training 2	Training 1&2
Without correction	2.45%	7.85%	2.29%	3.04%	2.97%	2.95%
With correction	2.32%	2.36%	2.49%	2.94%	2.69%	2.87%
Improvement	0.13%	5.49%	-0.20%	0.09%	0.29%	0.08%

Table 1. Misclassification rate on the segmentation on one column and one row

These results show that without the hypercubes' correction and only one training sample in the Dram area, the misclassification rate can vary depending on the training sample. The segmentation on the column is really dependant on the selection of the training set, whereas the segmentation on the row is more invariant because the contrast variations are less important. In both cases, the best results are achieved when combining the two training sets.

The correction on the first or the second training set improves the results in both cases, and the segmentation is more invariant with the selection of the training samples. With this segmentation, very good results are achieved (less that 3% of misclassification) with a few training samples. The results with one training set with the correction are as good as using the combination of two training sets without correction. Figure 14 and Figure 15 show the improvement brought by the correction on the column images using the second training set. Figure 16, Figure 17 show the slight improvement brought by the correction on the row images using the second training set.

There are still limitations to the correction technique, especially when the variations are large, or if the inter-class distance is small. Furthermore, the correlation measure is accurate on uniform fine textures, but less accurate on non-uniform texture where the correlation value is averaged. We also notice that on the column, the correction used on the combination of the first and second training set slightly increases the misclassification rate. This is caused by the big difference between the two samples on the DRAM area. For some parameters, the two samples are so different that they do not overlap. Thus, the correlation value is dramatically increased, and the corresponding hypercubes can be over-stretched. Nonetheless, misclassification is below 3%, which is adequate for this inspection process.

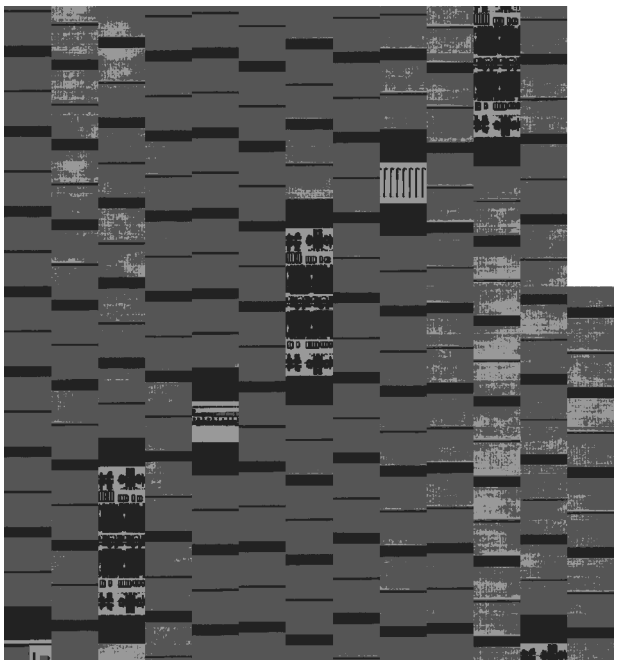


Figure 14: Segmentation on one row without correction

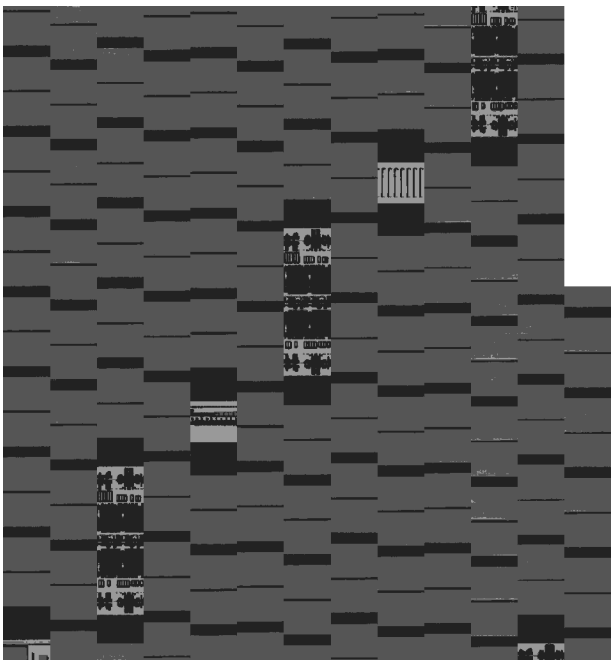


Figure 15: Segmentation on one row with correction

Figure 16: Segmentation on one column without correction

Figure 17: Segmentation on one column with correction

5. CONCLUSION

In wafer inspection, the performance of the segmentation is critical since the misclassification of an area can create a false detection, or increase the overall noise level in the area, that would result in a higher threshold with the risk of missing a critical defect. It looks like a trivial problem since we deal with regular textures, but it is not since the textures are subject to big variations that can be complex to correct. There are also limitations on the system flexibility to train the classifier since we have to work with huge dataflow. We develop an original method to anticipate the variations with a limited number of training samples, thus increasing the training speed, and the segmentation performances.

REFERENCES

1. K. W. Tobin, "Inspection in Semiconductor Manufacturing", *Webster's Encyclopedia of Electrical and Electronic Engineering*, Wiley & Sons, New York, NY, vol. 10, pp. 242-262 (1999).
2. K. W. Tobin, L. Neiberg, "Metrology Data Management and Information Systems", *Handbook of Silicon Semiconductor Metrology*, Marcel Dekker, New York, NY, pp. 679-703 (2001).
3. "The National Technology Roadmap for Semiconductors: Technology Requirement", Semiconductor Industry Association (2001).
4. J. Harrigan, M. Stoller, "Automated Wafer Inspection in the Manufacturing Line", *Solid State Technology*, 34(10), pp 69-72 (1991).
5. P. Bourgeat, F. Meriaudeau, P. Gorria, K.W. Tobin, "Content-based segmentation of patterned wafer for automatic threshold determination", in *Machine Vision Applications in Industrial Inspection XI*, Proc. SPIE 5011, pp. 183-189 (2003).
6. P. Bourgeat, F. Meriaudeau, K.W. Tobin, P. Gorria, "Patterned wafer segmentation", in *Quality Control by Artificial Vision VI*, Proc. SPIE 5132, pp. 36-44 (2003).
7. M. Unser, "Texture Classification and Segmentation Using Wavelet Frames", *IEEE Transactions on Image Processing*, 4(11), pp. 1549-1560 (1995).
8. M. J. Shensa, "The Discrete Wavelet Transform: Wedding the A Trous and Mallat Algorithms", *IEEE Transactions on Signal Processing*, 40(10), pp. 2464-2482 (1992).

9. P. Dutilleux “An Implementation of the algorithme à trous to compute the wavelet transform”, *Wavelets: Time Frequency Methods and Phase Space*, Springer, Berlin, pp. 298-304 (1989).
10. M. Feil, A. Uhl, “Real-time image analysis using MIMD Parallel à trous wavelet algorithms”, *Real-Time Imaging*, 7(6), pp. 483-493 (2001).
11. J. Miteran, P. Gorria, M. Robert, "Classification géométrique par polytopes de contraintes. Performances et intégration", *Traitement du Signal*, 11(5), pp. 393-408 (1994).
12. J. Miteran, *Performances et integration d’un algorithme de classification géométrique par apprentissage. Applications en traitement d’image*, PhD thesis, Universtity of Burgundy (1994).
13. J. Tian, *The Mathematical Theory and Applications of Biorthogonal Coifman Wavelet Systems*, PhD. Thesis, Rice University (1996).