Image-Based Wafer Navigation

Michael Lifshits, Roman Goldenberg, Ehud Rivlin, Michael Rudzsky, and Mike Adel

***Abstract*—Microscopic imaging is used in most core technology processes where integrated circuit (IC) digital images reveal important information. We present a new method for navigation on wafers that is based on localization of microscopic eye-point images using a previously acquired wafer map. It is fast enough for in-line microscopy and robust to visual changes occurring during the manufacturing process, such as contrast variation, rescaling, rotation, and partial feature obliteration. The method uses geometric hashing, a highly efficient technique drawn from the object recognition field. This approach proved to be highly reliable when tested on typical wafer images.**

***Index Terms*—Geometric hashing, image-based navigation, image matching, integrated circuits manufacture, integrated circuits measurements, navigation, pattern matching.**

I. Introduction

METROLOGY and inspection play an ever more critical role in process control of semiconductor manufacturing. Broadly speaking, these applications may be broken down into defect detection, classification and review, and critical dimension (CD) and overlay metrology [1]. Both e-beam and optical methods rely on optical imaging-based wafer navigation solutions. High wafer throughput is an important requirement for metrology and inspection systems due to the high value and enormous quantity of wafers being manufactured. Usually, imaging is the first, but can also be the only step in the "see–measure–control" sequence in semiconductor manufacturing. Therefore, the need for rapid nondestructive imaging-based techniques is growing [2], [3]. According to the ITRS [4], there is also a clear requirement for a paradigm shift in the role of metrology from off-line sampling to in-line control, as real-time integrated metrology is essential for manufacturing complicated devices.

Development of innovative metrology methods, designed to give faster and more detailed and useful information which would help establish more automated process control, is required. Various types of semiconductor manufacturing equipment call for rapid and precise wafer positioning so that operations such as lithography, cutting, and inspection can be performed to extremely tight tolerances. For example, the speed and accuracy of wafer scanners, optomechanical machines that perform the exposure part of the photolithography, are largely determined by the performance of a wafer stage, the part of the wafer scanner that positions a wafer under the lens system. In this paper, we refer to the act of directing a tool from one point on the wafer to another as "navigation on the wafer." The problem of navigation while maintaining the exact wafer coordinates on the stage is essential to achieve increased yield, improved quality, and consequently, reduced manufacturing cost.

Let us consider a metrology/inspection application as an example of navigation on the wafer. The required measurement or inspection is to be performed on specific features printed at known positions on a wafer. In order to perform a metrology or inspection step one must first arrive at that feature. Without feedback, the stage cannot identify whether it has arrived at the exact location and, therefore, may miss the feature due to less than ideal environmental conditions such as vibrations, which contribute to stage inaccuracies. Moreover, if multiple movements are made, position errors will continue to accumulate. Modern metrology and inspection applications require highly accurate positioning of the feature to be measured in the field of view of the tool. Thus, dead reckoning is not sufficient for wafer navigation, and sensory feedback is required.

Current metrology techniques commonly adopt the following solution. An additional large and easier to track feature is selected as an "acquisition target" near the desired metrology/inspection feature. First, the nearby area is searched for the acquisition target using a low magnification level; then, the location of the feature to be measured/inspected is detected by knowing its relative position. The measurement or inspection is performed using the highest magnification available. This process is time consuming and nonrobust as the correlation-based matching techniques used for pattern localization are highly sensitive to possible changes in the visual appearance of the feature. Such changes occur during the manufacturing process and may include nonlinear contrast variation, color inversion, rescaling, rotations, and partial feature obliteration [5]. In this paper, we propose a different solution for wafer navigation, which is fast enough for in-line microscopy and robust to in-process variations. The same methodology can be used to identify known defects and perform many other tasks essential for semiconductor manufacturing.

Furthermore, one of the most time-consuming tasks associated with automated metrology and inspection applications is recipe setup. For an operator to "train" the tool during the recipe setup process to identify the acquisition features and record their relative location to the features under test traditionally requires the placement of a product wafer on the stage of the tool. An attractive alternative to this procedure under consideration today is the option of "waferless" recipe setup which relies on the availability of acquisition and test feature location information

Manuscript received March 3, 2003; revised June 20, 2004. This work was conducted as part of the Wafer Fab Cluster Management (WFCM) Consortium supported by the Chief Scientist Office, Israeli Ministry of Industry and Trade, under the "MAGNET" program.

M. Lifshits, R. Goldenberg, E. Rivlin, and M. Rudzsky are with the Computer Science Department, Technion, Haifa 3200, Israel (e-mail: protezhe@cs.technion.ac.il; romang@cs.technion.ac.il; ehudr@cs.technion.ac.il; rudzsky@cs.technion.ac.il).

M. Adel is with the Optical Metrology Division, KLA-Tencor, Migdal Haemek 23100, Israel (mike.adel@kla-tencor.com).

Digital Object Identifier 10.1109/TSM.2004.831939

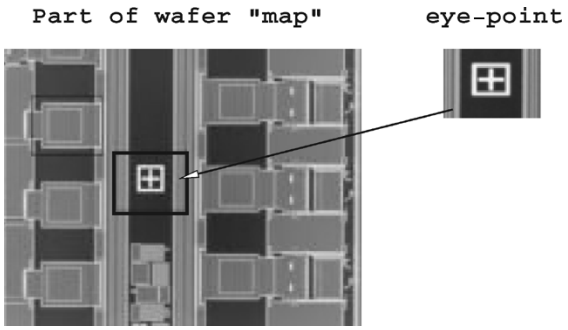

Fig. 1. Example of the real eye point within a wafer map.

from reticle layout data, eliminating the need for the wafer/operator interaction during the train step. This is of particular value in "high-mix" manufacturing environments such as semiconductor foundries where a large proportion of operator and tool time is expended on the recipe setup process. The wafer navigation methodology disclosed in this publication may readily lend itself to enhancing the "waferless" setup option as will be described.

Our algorithm uses advanced image processing techniques and geometric hashing [6]–[10], a known object recognition scheme, to perform image-based navigation on the wafer. As in the object recognition framework, image-based navigation aims to recognize the image at a current location or, in other words, to find a match in a set of images known in advance. Rapid and precise matching is achieved by finding a correspondence between geometric features extracted from the images. The algorithm exhibits high tolerance to run-time process variations as it is scale, rotation, and translation invariant. Moreover, the geometric nature of the features used by the algorithm makes it insensible to illumination changes, which is essential for wafer navigation.

The paper is organized as follows: Section II describes the navigation problem as an object recognition task. In Section III, our basic approach for wafer navigation is presented and an important enhancement to the basic algorithm is discussed. Section IV concentrates on various options available for feature selection and their influence on navigational system performance. Robust and efficient verification is described in Section V. Experimental results obtained using a wafer navigation system based on the proposed algorithm are presented and analyzed in Section VI. In the same section, we provide a systematic evaluation of the overall system performance and accuracy. Section VII is devoted to conclusions and future work directions.

II. Navigation Problem as an Object Recognition Task

Object recognition is a known problem in the computer vision field. Recognition is achieved by finding the correspondence between a given object and a set of predefined objects. In the model-based object recognition approach, the descriptions of previously known objects are prepared in terms of various properties, such as shape, color, etc. These descriptions are referred to as "models." A given query object is then matched to one of these models.

We refer to a partial image of the wafer (e.g., the current field of view of the microscope imaging system) as the wafer "eye point." Positioning on the wafer is defined in the following manner: given an eye point of the wafer, determine its exact coordinates on the wafer map. Thus, the process of wafer navigation can be carried out by repetitive vision-based positioning. Accordingly, map-based navigation can be interpreted as model-based object recognition as follows. Suppose that we already have a description of the wafer structure—a wafer map (the process of its construction will be explained in the next section). The wafer map can be divided into many adjacent parts to be identified during navigation. These correspond to a model set in the object recognition framework and the eye point plays the role of a query object. Matching the current eye point to one of the previously constructed parts of the wafer map during navigation is essentially the same as associating a query object to one of the known models in object recognition. An example of the wafer eye point and corresponding part of the wafer map is shown in Fig. 1(a).

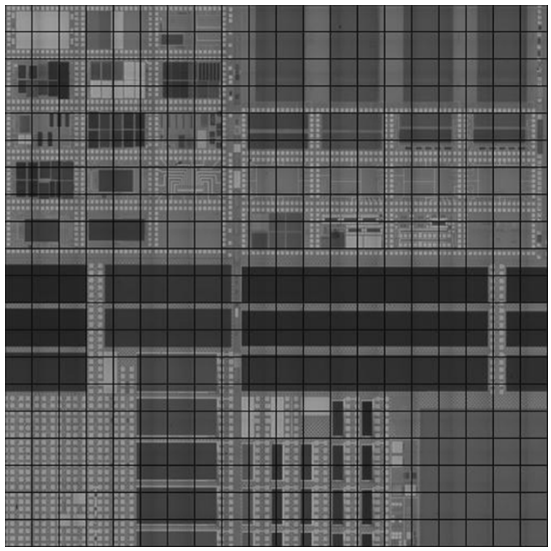

Fig. 2. Wafer map constructed by the combination of many adjacent partial wafer images.

Various approaches have been applied to vision-based localization. For an extensive survey on the developments in this area, we refer the reader to [11]. In our case, in order to handle the enormous amounts of geometric structures contained in wafer images, we choose to address the positioning problem using a highly efficient technique from the object recognition field called geometric hashing [6]–[10]. Matching between a current eye point and the wafer map is achieved by spatial correspondence of geometric features extracted from the images. The technique successfully deals with various visual transformations observable in semiconductor manufacturing such as two-dimensional (2-D) rotations, translations, and uniform scale.

III. Navigation Algorithm

A. Constructing a Wafer "Model's" Database for Navigation

Before one can begin navigation on the wafer, he or she must construct the wafer map. This can be performed in two different ways (for an example of a wafer map, see Fig. 2).

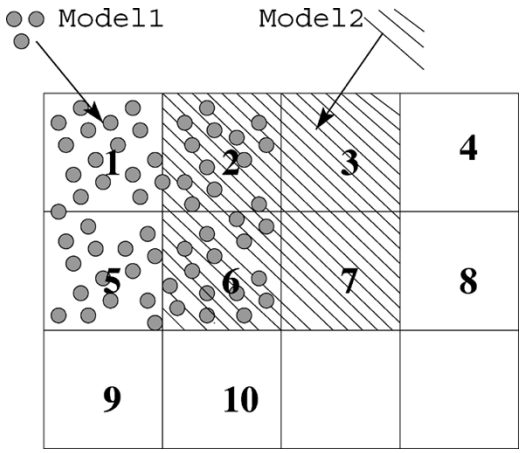

Fig. 3. Every four adjacent small tiles/models are merged into a single four-times-larger model. For example, the first model (marked with small circles) consists of tiles 1, 2, 5, and 6, the second model (marked with angled straight lines) consists of tiles 2, 3, 6, and 7, and so on. Note that tiles 5 and 6 will be part of the 5, 6, 9, and 10 larger model.

1) From the combination of multiple partial images of the wafer lying near each other: In this case, small adjacent wafer images are first obtained and then integrated into one big map. Small images are captured by a microscope imaging system moving above the wafer surface according to some grid structure. The process is similar to tiling the whole map with small square tiles (images).

2) From a wafer layout: Every wafer is first designed and thus has a corresponding layout file describing the 2-D shapes on the different layers of a chip by a limited set of graphics primitives. One has to parse this file and build the wafer map based on the primitives in it. We used the Caltech intermediate layout file format (CIF).

This resembles the approach of optical die-to-die and die-to-database comparisons with the reference frame. After a large scale wafer map is constructed, it is decomposed into many models, which will be recognized during navigation. The process is similar to separating a mosaic back into its single tiles. The resulting models are formed by small images, which may be similar to the ones used to build the wafer map. As a final step of the preprocessing, the representations of each model (or small partial image) are inserted into a database. When analyzing a current eye point during navigation, the database is searched for the most suitable model among all the precompiled ones. The map coordinates of the matched model are well known.

In order to assure that any selected eye point will fall entirely into one tile/model, we replace every four original adjacent small tiles with a single four-times-larger one and use it instead (see Fig. 3). As soon as the large model containing the eye point is detected, the navigation task can be completed easily using a local search. Needless to say, this modification uses redundant information, as almost all of the original small models are contained in the new four larger ones (see Fig. 3). This does not, however, increase the number of models.

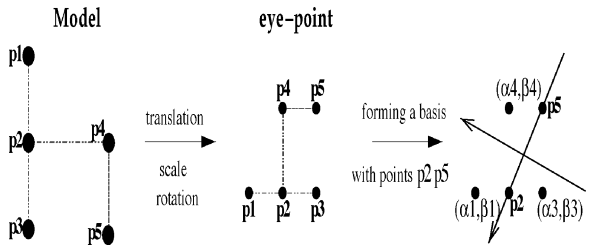

Fig. 4. Determining the hash table entries when points 2 and 5 define a basis. Models are allowed to be rotated, translated, and scaled.

B. Navigation Algorithm

Our algorithm uses geometric hashing, introduced by Lamdan and Wolfson [12], which is based on the indexing approach described in [13]. The algorithm solves the problem of navigation by applying object recognition in the following way. We assume a set of predefined geometric models $M_1, \ldots, M_n$, defining a wafer map, and a query eye-point image Q, formed from one of the models. The task is to find the corresponding model M_i given the query eye-point Q.

It is assumed that the models are defined by a set of geometric features (e.g., corner points) and that the same features can be extracted from the query eye-point image. A model can undergo similarity transformations to form the eye point: it can be rotated, translated, and uniformly scaled. One way to make feature points invariant under this class of transformations is to represent them in the coordinate frame formed from the points themselves. For example, we may arbitrarily choose an ordered pair of model points to form a basis and describe the rest of the features in this coordinate frame. As there are multiple ways to choose a basis, we are faced with a combinatorial problem of finding the right one to match a model to the eye point.

The algorithm copes with this problem by shifting the computational burden to the off-line learning stage. Instead of going over all feasible eye point/model bases couples and trying to match them, all possible model representations are prepared in advance and stored in a hash table for efficient access. Thus, a query eye point projected onto an arbitrarily chosen basis has a matching model representation already stored in the hash table.

Let $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ be the feature points of the eye-point Q and $M_i' = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\} \subseteq M_i$ be the corresponding feature points of the matching model M_i stored in the hash table. Let us denote the transformation model M_i to form the eye point by T; then, $Q = \mathrm{T}(M_i')$, or $\mathbf{q}_j = \mathrm{T}\mathbf{m}_j$ for $1 \leq j \leq k$. Consider an ordered points pair $(\mathbf{m}_1, \mathbf{m}_2)$ of M_i'. A vector $(\mathbf{m}_1 - \mathbf{m}_2)$ with another vector rotated by 90° form the basis of a 2-D coordinate frame. The coordinates (α, β) of any other point $m_j \in M_i'$ for $3 \leq j \leq k$, in this frame agree with

$$\mathbf{m}_j = \frac{\mathbf{m}_1 + \mathbf{m}_2}{2} + \alpha(\mathbf{m}_2 - \mathbf{m}_1) + \beta(\mathbf{m} - \mathbf{m}_1)$$

where we denote an end point of the rotated vector $\mathrm{Rot}_{90}(\mathbf{m}_1 - \mathbf{m}_2)$ by $\mathbf{m}$. These (α, β) coordinates will remain unchanged when any linear transformation is applied to the model points.

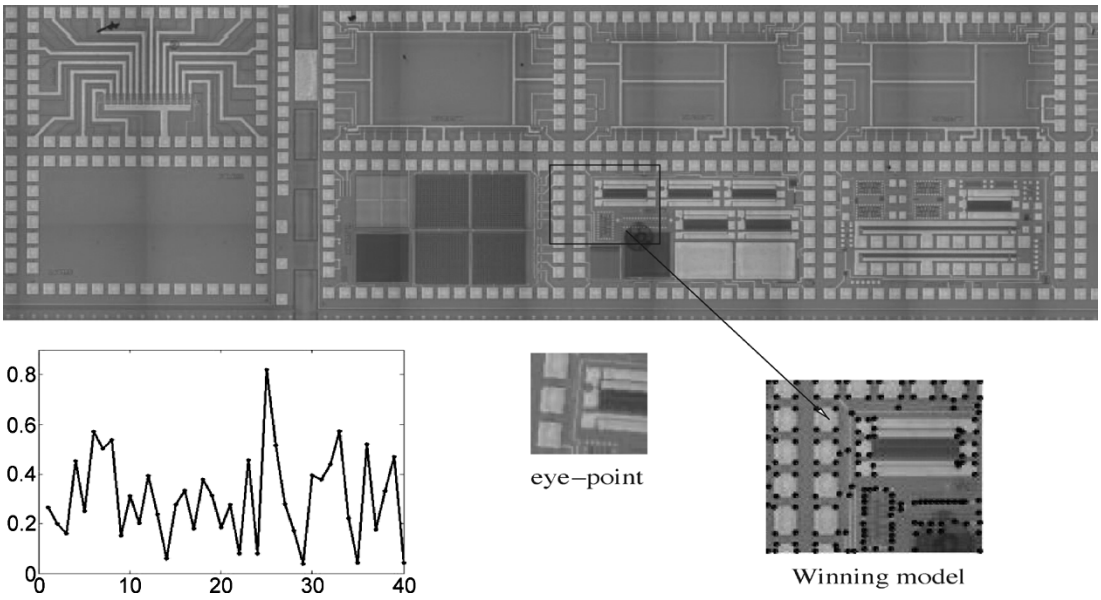

Fig. 5. Outline of the navigation process. Wafer map that is constructed from 40 model images is shown on the top. Voting results for the eye point shown in the middle are plotted on the left. Enlarged image of the winning model (25) with its feature points marked with black dots is presented on the right.

Application of a linear transformation T on the model M_i transforms the point $\mathbf{m}_j$ to

$$\mathbf{Tm}_j = \frac{\mathbf{Tm}_1 + \mathbf{Tm}_2}{2} + \alpha(\mathbf{Tm}_2 - \mathbf{Tm}_1) + \beta(\mathbf{Tm} - \mathbf{Tm}_2)$$

so the point $\mathrm{T}m_j$ has the same coordinates (α, β) in the frame formed by the ordered basis pair $(\mathbf{Tm}_1, \mathbf{Tm}_2)$. Thus, we further refer to coordinates (α, β) as invariant coordinates.

Assuming that the model M_i contains N_i feature points, there are $\binom{N_i}{2}$ different bases for that model. To form a transformation-invariant model representation, the invariant coordinates (α, β) are computed using each one of these bases $B_{\mu\nu} = \{\mathbf{m}_\mu, \mathbf{m}_\nu\}$ for every other model point. The corresponding entry $(M_i, B_{\mu\nu})$ is stored in the hash table with index (α, β). For example, consider the left side of Fig. 4. An invariant representation of a chair model comprising five dots is inserted into the database. An eye point containing the chair was rotated and scaled (in Fig. 4, middle). During navigation, when a pair of dots $\mathbf{p}_2$, $\mathbf{p}_5$ is used as the basis of a reference frame in the eye point (Fig. 4, right side), the entry ("Chair," $\{\mathbf{p}_2, \mathbf{p}_5\}$) is repeatedly addressed in the hash table at indexes $(\alpha j, \beta j)$, corresponding to the invariant coordinates of all the other points. As a result, this entry gets a sufficient number of votes and the eye point is correctly matched to the model of the chair.

Wafer navigation involves two stages: the *off-line preprocessing stage*, when most of the time-consuming work to compute the invariant model representations is done, and the fast *on-line navigation stage*, which uses the data prepared by the first stage to perform the matching and navigation. The efficiency of the second stage determines the actual navigation time. We now provide detailed description of the stages.

1) **Preprocessing Stage**

In the preprocessing stage, all models are processed individually, while their transformation-invariant representations are constructed and stored in a hash look-up table. For each model M and for every feasible basis b (composed of ordered features pair), it is necessary:

a) to compute the invariant coordinates of all the remaining feature points in terms of the basis b;
b) to use the computed coordinates as an index to the hash table and to store an entry $(M; b)$.

The complexity of this stage is $O(n^3)$ per model, where n is the number of points contained in the model. However, since this stage is executed off-line, its complexity is of little significance.

2) **Navigation Stage**

In the navigation stage, the invariant coordinates are computed from the eye-point features and are used as indexing keys to access the hash table and vote for the possible model matches. The model that scores the highest number of votes indicates the correspondence of the current eye point with that model. An example of a typical navigation process is presented in Fig. 5. Given an eye point with k feature points, the following steps are performed.

a) Choose a feasible pair of feature points as a basis b.
b) Compute the invariant coordinates of all the remaining feature points in terms of this basis b.
c) Use each computed invariant coordinate to index into the hash table and vote for all (M_i, b_j)s retrieved from this bin.
d) Build a histogram for all (M_i, b_j)s according to the number of received votes.
e) Establish a hypothesis of correspondence between an eye point and an instance of model M_i if (M_i, b_j), for some j, peaks in the histogram with a sufficient number of votes.
f) Verify all the hypotheses established in Step e)and repeat from Step a) if all of them fail verification.

The complexity of this stage is $O(n)+O(t)$ per probe, where n is the number of points contained in the eye point and

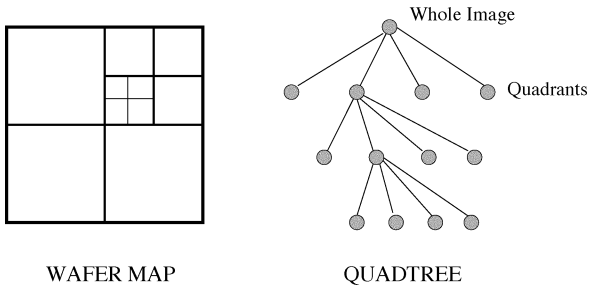

Fig. 6. Decomposition of the wafer map using a quadtree.

t is the complexity of verifying one model. Note that the complexity is independent of the number of models stored in the system, thereby allowing fast navigation even on very large-scale maps.

C. Performance Enhancement Using a Quadtree

Unlike absolute localization on the wafer, in the incremental localization discussed here, the initial position is assumed to be approximately known at the beginning of the navigation session. The goal is to refine the eye-point position estimation. Thus, it is possible to search for the eye point only in a relatively small "expectation region" on the wafer map, based on the known initial location. One can observe that in the case of wafer navigation the set of models is actually formed from the neighboring wafer image tiles. This allows us to refine the basic algorithm to utilize prior knowledge about the approximate eye-point position. The *quadtree* data structure can be used to reduce the number of models being examined by searching the hash table for models within the "expectation region" only. A quadtree is a spatial tree data structure that encodes an image hierarchically; see Fig. 6. Each node of the tree has up to four children. The root node represents the entire image; its children represent the four quadrants of the entire image, and their children represent the 16 subquadrants, etc.

The basic algorithm implementation uses a 2-D hash table, whose bins are accessed according to the computed invariant coordinates. Multiple entries within a single bin are organized in a linked list and retrieved altogether when the corresponding bin is accessed. The proposed enhancement replaces the linked list with a quadtree at each bin in the hash table. These trees correspond to the space partitioning of the global wafer map (see an illustration in Fig. 7). In this way, it is possible to access only the relevant part of each tree during voting and, thus, exclusively account for models from the "expectation region." To put it differently, the quadtree allows us to select any partial wafer area to be searched for the query eye point.

This approach improves the performance of the algorithm by reducing both the navigation time and the false matches. The approach addresses one of the undesirable effects of the proposed navigation method, which is a high population of certain areas of the hash table due to nonuniform distribution of coordinates in the invariant space. This effect results in an increased number of models getting many votes. This, in turn, makes it necessary to increase the number of candidates retained after the voting stage for verification, in order to get high reliability. Practically speaking, the quadtree approach reduces the number of irrelevant entries accessed in the hash table without actually removing any contained data.

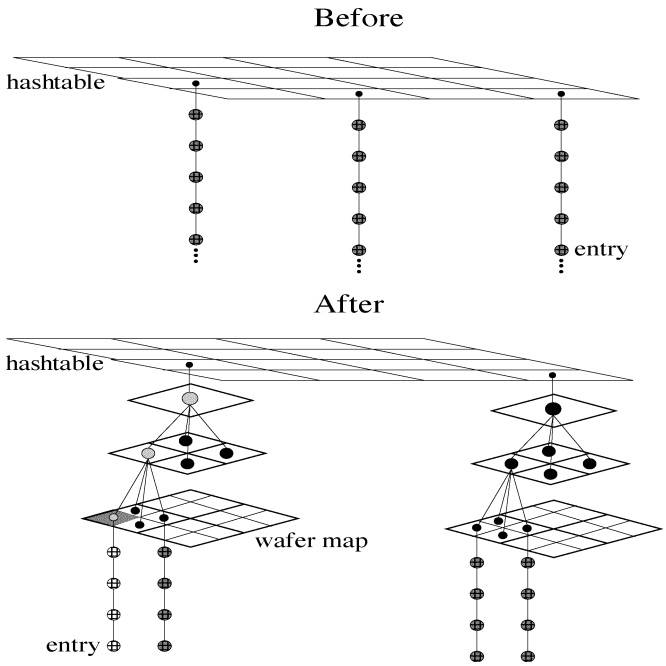

Fig. 7. Multiple entries within single bin are organized in a quadtree after the enhancement, as opposed to a traditional linked list organization. This allows us to access only the relevant part of each tree and not necessarily vote for all the entries stored in the corresponding bin when it is accessed. For example, if one is certain that the query eye point is located in the bottom left corner of the wafer map (the area marked by grey shading), it is possible to follow the corresponding path in the quadtree and to vote only for the relevant entries (colored light grey).

IV. Features Selection

The extraction of image features is one of the critical parts of the navigation algorithm. Extracted features are used to form invariant models' descriptions during the preprocessing stage and to determine which entries get votes during on-line navigation. Various properties of the extracted features play an important role in the navigation system performance. Here, we concentrate on the major ones: *quantity* and *consistency*.

System storage requirements may become a major issue as the number of features increases, since a model with n features produces n^3 records in the hash table. For instance, a system with 1000 models, each having 200 features, reaches $1000 * 200^3 = 8 \cdot 10^9$ records. A large number of hash table records significantly degrades navigation performance. Voting within highly populated hash table bins is time consuming. Furthermore, it can result in many wrong models receiving votes. On the other hand, too few features make matching unreliable.

In general, there are actual differences between the database model and the query eye-point images due to in-process material variations. Consequently, a fraction of the eye-point features will not match the features extracted from the model during the preprocessing stage. We denote these unmatched eye-point features as *inconsistent* features (outliers). These inconsistent features do not contribute votes to the correct model and by doing so reduce the ability of the system to successfully match the eye point to the reference model. Moreover, if any outlier is used to form a basis for voting during navigation, there is a good chance

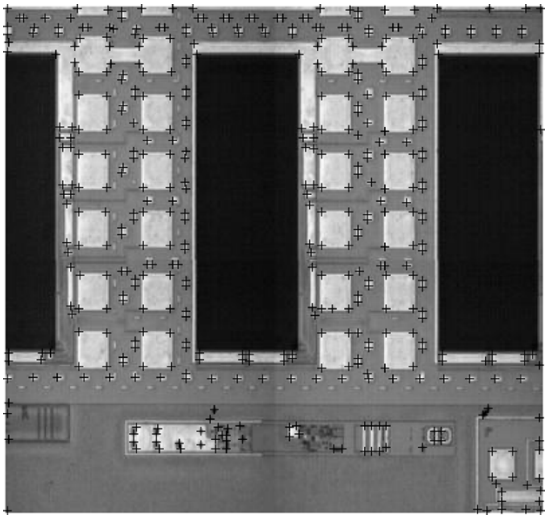

Fig. 8. 447 corners that were detected by the Harris corner detector in the sample model image.

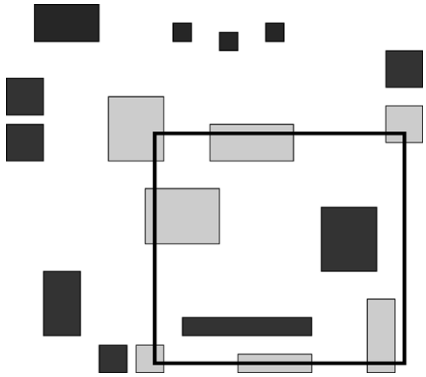

Fig. 9. Schematic representation of rectangular features detected in the model image and the query eye point (marked with a bold black line). Note that rectangles that are partially contained in the eye-point image (marked with light grey) cannot be taken into account during voting.

that the algorithm will fail and point to an incorrect eye-point location on the wafer map.

A. Selecting the Right Feature

Generally, as the complexity of the features increases, the number of features making up the models decreases, since an object can be described by a few complex parts or by many simple parts [14]. Thus, feature quantity is largely determined by complexity. When selecting the right type of features one has to take into account the following general ideas. Using simple features (e.g., corners) results in large amounts of hash table records. These features represent a more ambiguous interpretation of the model data (e.g., a few corners in the eye point may correspond to many corner triplets in many models); therefore, matching is similar to accumulating a lot of insignificant evidence for the presence of the correct model. On the other hand, utilizing more complex features (e.g., contour groupings or "virtual" polygons with vertices corresponding to simple feature points) results in fewer records stored in the hash table. These features have much more discriminative power, since a very special polygon is not likely to be present in many different models. Matching can be based on a very few such primitives, because each provides significant evidence for the presence of the correct model. However, reliable recovery of complex features is a difficult problem, particularly in the presence of noise and obliteration [14].

We adopted the above approach when selecting features to be used in our algorithm. In order to achieve fast and reliable navigation one would like to work with a moderate number (due to their medium complexity) of sufficiently consistent features. Adapting geometry-based features makes the navigation algorithm invariant to intensity changes. Typical wafer images contain rectangular structures; therefore, the objects being of particular concern are corners, straight line segments, and rectangles.

We used a Harris detector [15] to find corner features. As expected, using such simple features results in a high number of entries being stored in the hash table. Typical model images with which we worked contained more than 200 corners (see, for example, Fig. 8), reaching practically unmanageable numbers of hash table records. Storing such a huge amount of information negatively affects the performance of the navigation system, as it slows down the voting stage and gives rise to false matches. This undesirable effect can be reduced by grouping corners into an integrated feature, e.g., a rectangle. A rectangle's complexity is much higher than corners; consequently, they produce a manageable number of hash table records. Furthermore, rectangles are less likely to be found at random. We experimented on the wafer images with a typical eye point containing too few rectangles to produce reliable voting (see Fig. 9).

The following feature type allows us to take advantage of the higher reliability of rectangular features while maintaining enough hash table entries to make voting reliable. We chose, as features, distinct corners found with the Harris corner detector, which coincided with corners of detected rectangles. In this manner, we can account for all the rectangle corners contained within the eye point in Fig. 9.

In order to demonstrate the properties of different types of extracted features and give some basis for comparison between them, we performed the following test: we constructed a map from 70 models; then, for each model, ten different eye points were selected. Three types of features were extracted from all of the eye points: corners found with the Harris corner detector, distinct corners found with the Harris corner detector that coincided with the corners of detected rectangles, and rectangles. The average quantity of the different feature types are presented in Fig. 10(a). The interplay between quantity and complexity of features, which is schematically shown in Fig. 10(b), is consistent with the obtained test results. There are too few rectangles, as they are a relatively complex feature. The number of corners found with the Harris corner detector is, on the other hand, very high, since corners are much more simple features. Harris corners corresponding to rectangular ones are found in the desired moderate quantity and provide sufficient consistency.

V. Verification

The second stage of the navigation algorithm is verification. Given a set of candidate models that accumulated the highest number of votes, one has to determine which is the best match to the query eye point. For this purpose, it is essential to specify how to fit the candidate models to the eye point.

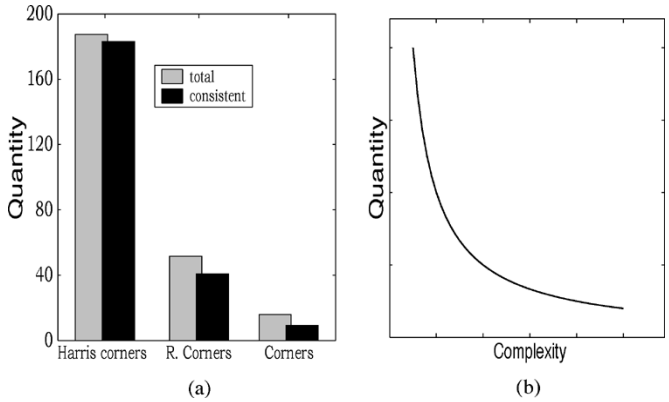

Fig. 10. Quantity of different types of features contained in typical eye points, which is randomly produced from 70 different models. Corners were found with the Harris corner detector. R Corners stand for distinct corners found with the Harris corner detector, which coincide with corners of detected rectangles. Total number of features within the eye point as well as the number of consistent features are shown by light grey and dark vertical bars, respectively, in the left subfigure. Interplay between quantity and complexity of features is schematically shown in the right subfigure: (a) average quantity and (b) quantity and complexity interplay.

A. Approximating the Best Solution

To form the eye point, the models undergo similarity transformation, which is a composition of translation, rotation, and isotropic scaling. Thus, fitting a model to an eye point should be done by a similarity transformation estimation. The eye point is characterized in terms of a feature points set $\{\mathbf{x}'_i\}$ in $\mathbb{P}^2$, and each of the candidate matching models is described by its feature points $\{\mathbf{x}_i\}$, where $\mathbb{P}^2$ is a projective space. First, it is essential to find all $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ point correspondences to compute a similarity transformation $\mathrm{H_S}$, which transforms a model to the eye point: $\mathrm{H_S}\mathbf{x}_i = \mathbf{x}'_i$ for each i. Two correspondences are enough to fully constrain $\mathrm{H_S}$, as the total number of degrees of freedom for similarity is four (one for the rotation, two for the translation, and one more for scaling), and every correspondence gives rise to two independent equations in the entries of $\mathrm{H_S}$. However, since the locations of points in the query eye point are not exact (due to noise), all of the correspondences should be used to determine the "best" transformation, given the data. Accordingly, $\mathrm{H_S}$ is calculated by finding the least squares solution of the overdetermined linear system.

B. Robust and Efficient Detection of a Maximum Point Correspondences Set

An important issue is how to efficiently find all of the correspondences. The navigation voting stage provides one corresponding basis (two point-to-point correspondences) between the candidate model and the eye point. This allows us to approximate the desired transformation $\mathrm{H_S}$ by $\widehat{\mathrm{H_S}}$ and then, after applying $\widehat{\mathrm{H_S}}$ on the candidate model, every model point $\widehat{\mathrm{H_S}}\mathbf{x}_i$ will correspond to the closest eye-point feature $\mathbf{x}'_i$. Formally

$$\mathbf{x}'_i = \arg\min_k d\left(\mathbf{x}'_k, \widehat{\mathbf{H}}_{\mathrm{S}}\mathbf{x}_i\right) \tag{1}$$

where subindex k indicates any eye-point feature and $d(\mathbf{x}, \mathbf{y})$ is the Euclidian distance between two points $\mathbf{x}$ and $\mathbf{y}$.

Thus, to compute all of the point correspondences it is possible to check the distance of each point $\mathbf{x}'_i$ to every transformed model point $\widehat{\mathrm{H}}\mathbf{x}_i$. If the model contains m points and the eye point contains n points, those interset distances are computed in $O(mn)$ time.

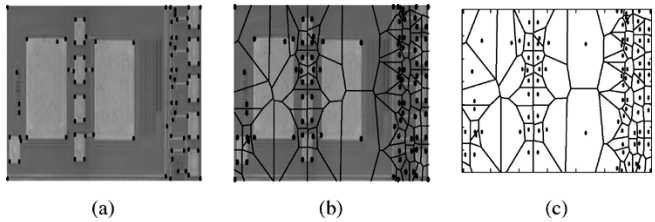

Fig. 11. Process of constructing the Voronoi tessellation of the eye point for verification acceleration: (a) eye-point feature, (b) Voronoi tessellation, and (c) Voronoi diagram points.

This computation can be accelerated by employing a *Voronoi tessellation* [16] for segmentation of the eye point image. Voronoi tessellation is partitioning of a plane with n points into n convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than to any other.

We start the verification by constructing the Voronoi tessellation from the points in the query eye point, which is done in $O(n\log(n))$ time [16] (see Fig. 11). This allows us to find the corresponding point of $\mathbf{x}_i$ in $O(\log(n))$ by checking what polygon within the Voronoi tessellation contains the transformed point $\widehat{\mathrm{H}}\mathbf{x}_i$ and choosing its center point. It follows that the time needed for point correspondences calculation is reduced from $O(mn)$ to $O(m\log(n))$.

In practice, the situation is complicated by the fact that some eye-point feature $\mathbf{x}'_i$ might be mistakenly reported and will not match any model point. The mismatched points, outliers, can severely disturb the estimated transformation and, consequently, should be identified. In order to make the verification robust to outliers, one has to obtain a big enough set of inliers from the presented correspondences so that the transformation can be re-estimated in an optimal manner (solving the overdetermined linear system as previously explained).

For this propose, a general robust estimator, the RANSAC algorithm [17], is used. It is able to cope with a large proportion of outliers and has the important property of reporting an explicit number of inliers, which is important to intuitively reject false positive candidates. Given an initial putative set of correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, computed from the model to the eye-point primary basis correspondence, the algorithm performs the following steps. A sample of two corresponding points is randomly selected and the appropriate approximation of the transformation $\widehat{\mathrm{H}}$ is calculated. Then, the number of inliers consistent with $\widehat{\mathrm{H}}$ is computed in terms of correspondences for which $d(\widehat{\mathbf{x}}'_i, \mathbf{x}'_i) < t$, where $\widehat{\mathbf{x}}'_i = \widehat{\mathrm{H}}\mathbf{x}$. These steps are repeated several times and the $\widehat{\mathrm{H}}$ with the largest number of inliers is chosen (in case of a tie the solution having the minimal standard deviation of inliers is chosen). We would like to choose a threshold t such that with a probability α the point is an inlier. It is assumed that the position error is Gaussian with zero mean and standard deviation σ. Thus, the square of the point distance d^2 is the sum of the squared Gaussian variables and follows a chi-squared χ^2 distribution with two degrees of freedom (as d^2 is the sum of squared x and y measurement errors). The cumulative chi-squared distribution $F(k)$ represents the probability

TABLE I
VALUES OF DISTANCE THRESHOLD t FOR WHICH THE PROBABILITY THAT THE POINT IS AN INLIER EQUALS α

α	0.9	0.92	0.95	0.99
t^2	4.6 σ^2	5.05 σ^2	5.99 σ^2	9.21 σ^2

that the value of the χ^2 random variable is less than k. Hence, to guarantee with probability α that the point is an inlier, one has to choose a distance threshold $t^2 = F^{-1}(\alpha)\sigma^2$. Some values of t associated with different levels of certainty that the point is an inlier are tabulated in Table I. For example, if one wants to make sure that an inlier is incorrectly rejected only 5% of the time, he or she will choose α as 0.95 and accordingly $t^2 = 5.99\sigma^2$.

VI. EXPERIMENTS

This section presents a number of experiments demonstrating both quantitative and qualitative aspects of the proposed navigation algorithm. The benefits gained from exploiting the quadtree enhancement, described in Section III-C, are also presented. We performed tests on both synthetic and real data sets. Synthetic images were chosen for their ability to isolate the indexing mechanism from other issues involved in navigation, such as feature extraction.

A. Test Data Sets

To create data sets we constructed two database model collections (wafer maps). The first, a synthetic map, consists of 30 synthetic models with different polygonal objects. Every synthetic model has 12–16 corner features. Examples of the models and the whole synthetic map are shown in Fig. 12 and Fig. 13. The second real wafer map covers an area of 4.5 × 9 mm on the wafer surface. It was constructed from real wafer images obtained on KLA-Tencor 5200XP overlay metrology tool using a 750-μm field of view. Examples of the images and part of the whole wafer map are shown in Figs. 14 and 15.

B. Performance Fine-Tuning Experiments

The main performance criterion for our navigation algorithm as well as for any indexing scheme is whether the correct model gets significantly more votes than others. If it does not, sequential comparison with a large number of the database models is required during the verification stage, making the indexing scheme insignificant and unnecessary. Consequently, if all the models are stored in a candidate list sorted by the number of received votes, one of the best criteria for algorithm performance is how close the true model is to the top of the candidates list. Since ground truth is available, it is possible to analyze indexing efficiency even if subsequent verification fails.

In the following, we present a number of tests exploring different aspects of our navigation system while the main criteria for algorithm performance is how close the true model is to the top of the candidates list.

1) Discrimination Power Test: This test explores the dependance of the algorithm's discrimination power on the number of features in the query eye point. We varied the number of features in the query eye point that was arbitrarily chosen from the wafer map and for each number performed a set of 100 iterations. At each iteration we analyzed the distribution of votes among all models and registered the position of the correct model within the candidates list. Results are shown in Fig. 16(a). It can be seen that the algorithm is very discriminative when the number of features is not small (more than eight) and the position of the true model is very close to the top of the candidates list. An increasing features number further improves the results as each additional feature point provides supplemental evidence for the presence of the correct model. There is a substantial reduction in the number of models to verify, as, on the average, the correct model/basis got into the 30 most voted-for models, which is less than 1% of all possible model/bases combinations.

2) Discrimination Power Test With Quadtree Enhancement: This test investigates algorithm performance improvement

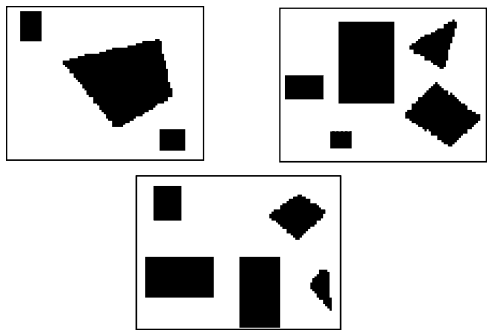

Fig. 12. Examples of the synthetic model images, each containing 12–16 corners features.

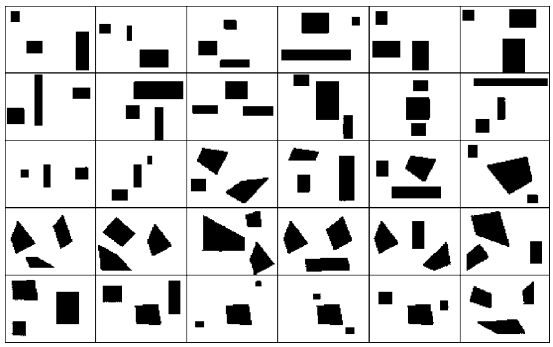

Fig. 13. Synthetic map constructed from 30 model images.

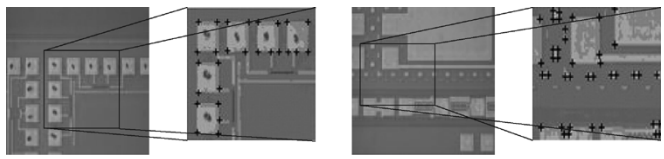

Fig. 14. Examples of the real wafer images used as models in the algorithm.

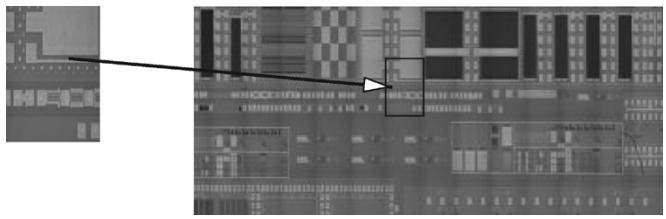

Fig. 15. Part of a real wafer map constructed from 72 images, some of which are shown in Fig. 14.

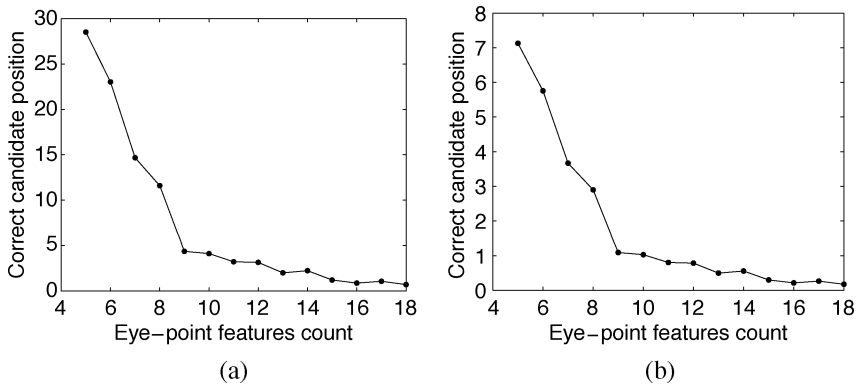

Fig. 16. Discrimination power test. Average position of the correct model in the candidates list is plotted against the number of features in the eye point: (a) basic algorithm and (b) using quadtree enhancement.

when employing the quadtree approach described in Section III-C. In Fig. 16(b), the results of the same test as before are shown, while using a quadtree with depth two for efficient access to the hash table entries (quadtree implementation is taken from [18]). The discrimination power of the algorithm is improved by a factor of four as expected, since only one-fourth of the models actually participate in navigation. It can be seen that the position of the true model in the candidates list is four times closer to the top.

3) Quality of Bases Test: Given an eye point, the first step of the navigation algorithm is to select a basis in order to create an eye-point invariant representation. The choice of two points, used to form a basis, greatly influences the algorithm's performance. This phenomenon is discussed in [19]–[21], among others. For example, a large separation of the basis points results in an invariant model description that is less sensitive to noise; i.e., the computed invariant coordinates will have smaller noise-induced inaccuracy. Optimally, to form a good basis with good selectivity power, one should select a pair of points maximally separated from each other and so that all the remaining points would be located close to the center of the formed coordinate frame.

We performed the following test, this time using both synthetic and real wafer images, to demonstrate the influence of basis selection on the performance of the algorithm. At each iteration, a random eye point from the wafer map was selected and navigation with two different bases was performed. While the first basis was randomly selected as before, the second was selected according to the following measure, which is based on [20]

$$Q_{ij}^{m} = \sum_{k=1,k\neq i,j}^{n} \frac{1}{2}\log\left(\frac{4\left\|(p_i p_j)\right\|^4}{\left(4\left\|(u_k, v_k)\right\|^2 + 3\right)^2}\right) \qquad (2)$$

for the basis points (p_i, p_j) of model m where u_k and v_k are the coordinates of the third point k in the coordinate frame defined by this basis.

In Fig. 17, the average position of the correct model in the candidates list is plotted against the normalized quality of the selected basis. It can be seen that (2) is a good measure for bases quality, as highly ranked bases produced a very high-vote count for the correct model; consequently, that model was set at the top of the candidates list.

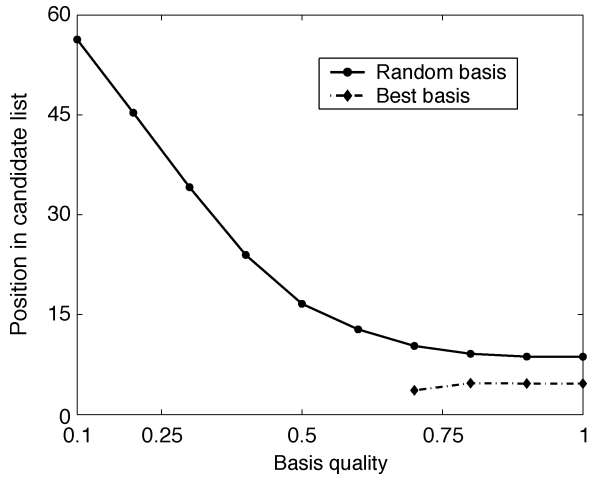

Fig. 17. Quality of bases test. Solid line corresponds to bases where the points were randomly selected. Results for bases having the best quality are shown with the dashed-dotted line.

C. Overall System Test

This test demonstrates the capabilities of the proposed navigation algorithm and provides a systematic evaluation of its performance. During voting, an eye-point invariant description is calculated and used to index the hash table and vote for all the accessed entries. The description is based on a pair of features, which form a basis, as explained in Section III. In many practical situations, there is a good chance that one of the points used to form a basis will be reported by mistake and hence not match any model point. Therefore, one should make multiple attempts using different bases (e.g., different descriptions) to ensure with sufficiently high probability that at least one of them is free of outliers. We evaluated the navigation algorithm performance by varying the number of different eye-point feature bases being used in voting.

We tested the algorithm on a total of 10^4 different navigation tasks to obtain a statistically meaningful measure of its performance. Each time we selected a random eye point and then, if the correct location on the wafer "map" was reported by the algorithm (ground truth was available due to the nature of data set formation), the result was considered to be true positive (TP). In case of an incorrect location or if no location was found (simply because none of the database models got enough votes), the result was regarded as a false positive (FP) or miss, accordingly.

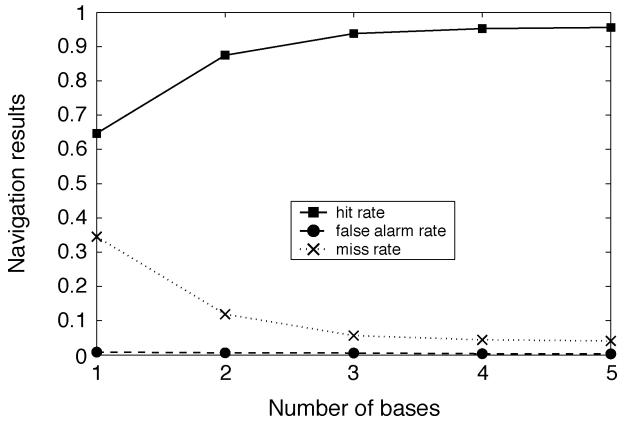

Fig. 18. System behavior with a different number of bases used in voting.

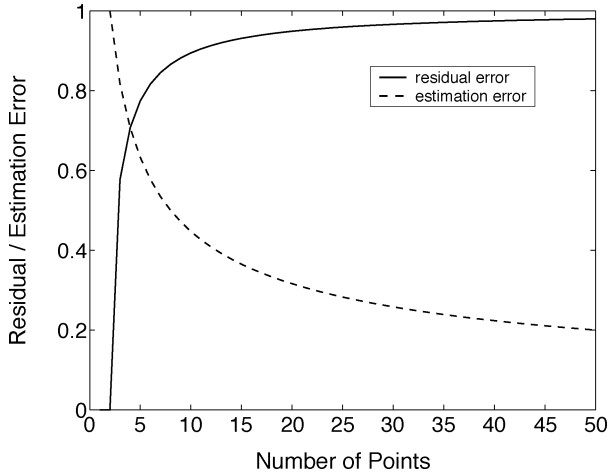

Fig. 19. Error in pixels when the number of sample points used to estimate the transformation varies and the measurement error level is assumed to be one pixel.

The summary of the obtained results is presented in Fig. 18. The hit rate $\mathrm{HR} = \sharp\mathrm{TP}/\sharp\mathrm{Tests}$ reaches 95% with a 4% false alarm rate and a 1% miss rate when four bases are used. A 100% HR is not achieved as the constructed map contains areas difficult for navigation, that is, areas having no distinguishable features or filled with repetitive geometric structures. Note that even a human would have serious difficulties in solving the task of self-localization for "degenerate" eye points selected from these unfavorable areas. Generally, we found that the algorithm performed well for eye points from most of the wafer areas.

D. Accuracy Estimation

The inaccuracy of the navigation result may be formulated as follows. Assuming the eye-point features are measured with a Gaussian error of standard deviation σ, it can be shown that [22]:

the root mean square (RMS) **estimation error** (distance of the estimated point location from its true value) is

$$\epsilon_{\mathrm{est}} = \sigma \left(\frac{2}{n}\right)^{\frac{1}{2}}$$

and the root mean square (RMS) **residual error** (distance of the measured from the estimated value) is

$$\epsilon_{\mathrm{res}} = \sigma \left(1 - \frac{2}{n}\right)^{\frac{1}{2}}$$

where n is number of correspondences used.

The graph of these errors is shown in Fig. 19. Note that for 50 sample points the estimation error is 0.2 pixels. Thus, if an eye-point image of size 200 $\times$ 200 pixels is taken at resolution of 50 μm, our algorithm provides navigational accuracy of 0.05 μm.

E. Testing the Algorithm Robustness to Image Change

The aim of this test is to provide the experimental analysis of the algorithm behavior when the wafer images are distorted (e.g., due to possible process variations). Creating a full and statistically meaningful model of the algorithm's performance in the presence of various process variations is beyond the scope of this work. Therefore, we analyzed a number of extremely distorted wafer images in order to develop a procedure which systematically distorts "clean" wafer images to simulate possible visual changes. We tested our algorithm on these systematically distorted images.

We concentrated on the factors having the most negative effect on the navigation performance: gray scale intensity changes and image rotations. We transformed the original image intensity according to a scheme that uses cubic spline interpolation of specially placed points to create five intensity remapping functions. Note that such a simulation accounts for possible local dynamic range variations, since these result from the nonlinearity of the underlying remapping functions. Color inversion was also simulated by a function produced with collinear points, linearly remapping the brightest intensity to darkest. This *ad hoc* simulation does not model all relevant intensity changes but produces images coherent with a number of real-world examples that we analyzed. We also rotated the images up to 8°. The average results of navigation performed on randomly chosen and systematically distorted eye-point images are shown in Figs. 20 and 21 along with some examples of these images. The navigation results were interpreted as in the overall system test (see Section VI-C). Note that in both cases the HR (the rate of correctly matching the eye point) does not drop below 0.8 even for severely distorted eye-point images.

VII. Conclusion

We presented a new method for navigation on wafers, based on the geometric hashing technique. The method is invariant to changes in visual appearance, such as nonlinear contrast variation, scale, rotation, and partial obliteration. A quadtree enhancement of the basic geometric hashing algorithm was proposed, improving its computational performance by allowing efficient access to the hash table entries. We showed that combing basic feature types leads to consistent features in a desired moderate quantity. We also showed how verification can be significantly accelerated by applying a Voronoi tessellation of the eye point. Extensive experimental analysis demonstrates the high reliability of the proposed method.

The same principles can be used to identify known defects (by efficiently searching a defects database) and to perform many other tasks essential for semiconductor manufacturing. In the future, we plan to implement a distributed system in order

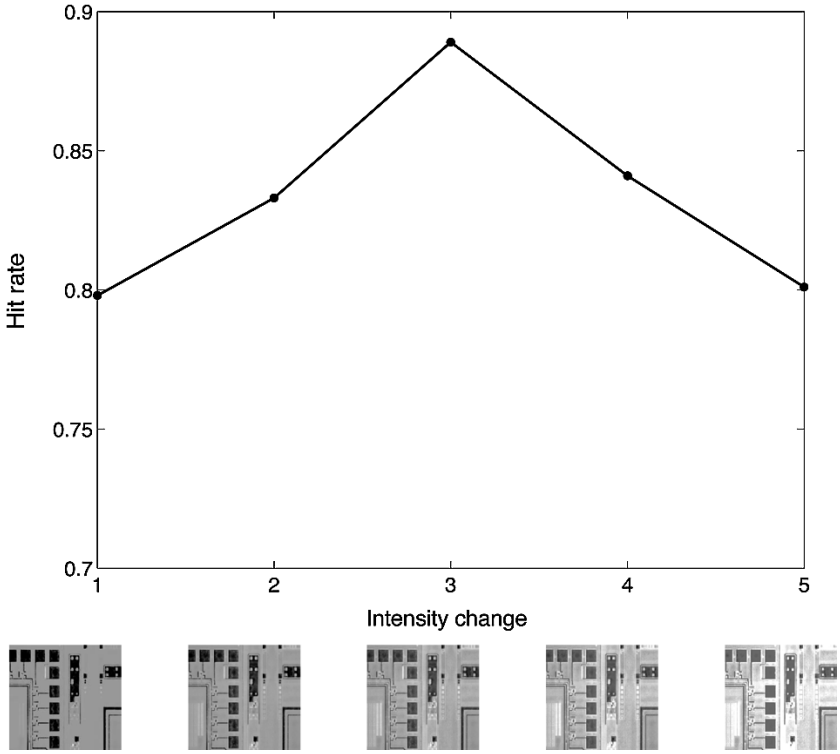

Fig. 20. Average navigation results performed on systematically distorted images simulating real-world intensity changes. Images under the graph are the examples corresponding to intensity changes 1–5, respectively. Note that the peak corresponds to the intensity change that linearly inverts colors (bright turns into dark and vice versa).

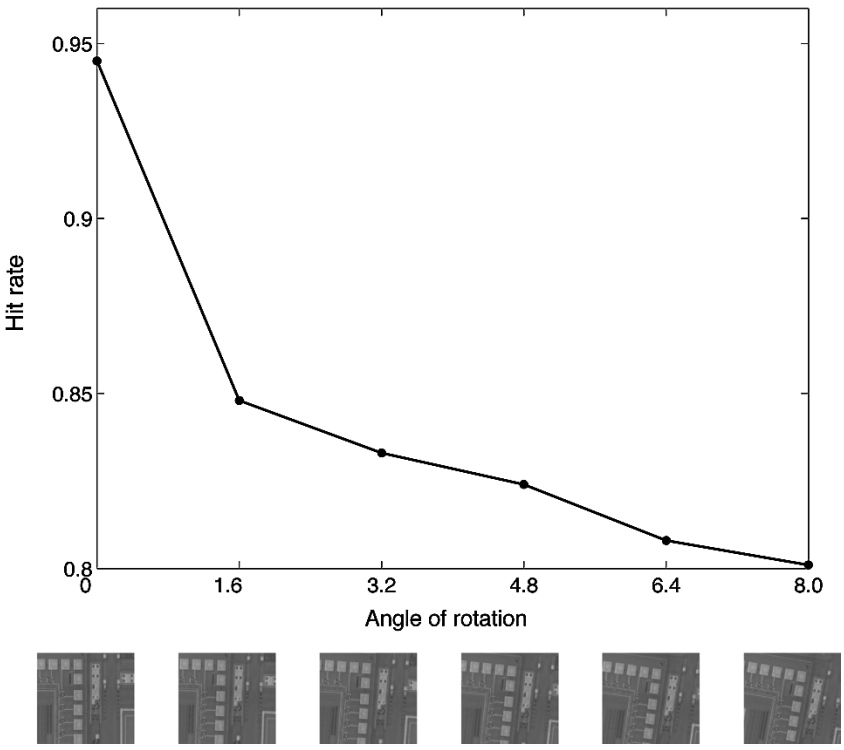

Fig. 21. Average navigation results performed on systematically distorted images simulating real-world rotations. Images under the graph are the examples corresponding to rotations by 0–8°, respectively.

to perform the voting in parallel thereby reducing the navigation time, on one hand, and eliminating the need for huge storage, on the other, since the hash table would reside on multiple computers.

ACKNOWLEDGMENT

The authors would like to thank KLA-Tencor Israel for a donation of the 5200XP metrology tool and for support in installation and training.

REFERENCES

[1] V. Sankaran, C. M. Weber, and K. W. Tobin, "Inspection in semiconductor manufacturing," *Websters Encyclopedia of Electrical and Electronic Engineering*, vol. 10, pp. 242–262, 1999.
[2] International Technology Roadmap for Semiconductors, S. I. Association. (2001). *http//public.itrs.net/Files/2001ITRS/Home.htm* [Online]
[3] International Technology Roadmap for Semiconductors Update——, (2002). *http//public.itrs.net/Files/2002Update/Home.pdf* [Online]
[4] International Technology Roadmap for Semiconductors——, (2003). *http//public.itrs.net/Files/2003ITRS/Home2003.htm* [Online]
[5] S. Melikian, "Geometric searching improves machine vision," *Lasers Optronics*, vol. 18, no. 7, p. 13, July 1999.
[6] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two-dimensional, model-based, boundary matching using footprints," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 38–55, 1986.
[7] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Affine invariant model-based object recognition," *IEEE Trans. Robotics Automat.*, vol. 6, pp. 578–589, Oct. 1990.
[8] ——, "On recognition of 3-d objects from 2-d images," in *Proc. IEEE Int. Conf. Robotics Automation*, vol. 3, Philadelphia, PA, Apr. 1988, pp. 1407–1413.
[9] ——, "Object recognition by affine invariant matching," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Ann Arbor, MI, June 1988, pp. 335–344.
[10] H. J. Wolfson, "Model-based object recognition by geometric hashing," in *Proc. Eur. Conf. Computer Vision*, Antibes, France, Apr. 1990, pp. 526–536.
[11] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 237–267, Feb. 2002.
[12] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Proce. 2nd Int. Conf. Computer Vision*, Tampa, FL, June 1988, pp. 238–249.
[13] J. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves," *Int. J. Robotics Res.*, vol. 6, no. 2, pp. 29–44, 1987.
[14] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, "From volumes to views an approach to 3-d object recognition," *CVGIP: Image Understanding*, vol. 55, no. 2, pp. 130–154, March 1992.
[15] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, Manchester, June 1988, pp. 147–151.
[16] M. V. Kreveld, M. Overmars, O. Schwarzkopf, and M. V. K. Mark de Berg, *Computational Geometry*, 2nd ed. Berlin, Germany: Springer Verlag, Feb. 2000, pp. 147–163.
[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[18] L. Balmelli, J. Kovacevic, and M. Vetterli, "Quadtree for embedded surface visualization: constraints and efficient data structures," in *Proc. IEEE Int. Conf.Image Processing*, vol. 2, Oct. 1999, pp. 487–491.
[19] H. J. Wolfson and I. Rigoutsos, "Geometric hashing: An overview," *IEEE Computational Sci. Eng.*, vol. 13, pp. 10–21, 1997.
[20] I. Rigoutsos and R. Hummel, "Robust similarity invariant matching in the presence of noise," in *Proc. 8th Israeli Conf. Artificial Intelligence Computer Vision*, Tel Aviv, Israel, Dec. 1991.
[21] W. E. L. Grimson, D. P. Huttenlocher, and D. W. Jacobs, "A study of affine matching with bounded sensor error," *Int. J. Computer Vision*, vol. 13, no. 1, pp. 7–32, 1994.
[22] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

Michael Lifshits received the B.A. degree (with honors) in computer science, from the Technion-Israel Institute of Technology, Haifa, in 2000. He is pursuing the M.S. degree in computer science at the same university.

From 1998 to 2000, he worked at GE Ultrasound, Israel, and in 2000 at Quantum, Snap Division, U.S. His research interests include machine vision and pattern recognition.

Roman Goldenberg received the B.A. (*summa cum laude*) and Ph.D. degrees in computer science from the Technion—Israel Institute of Technology, Haifa, in 1995 and 2003, respectively.

From 1994 to 1996 and in 1999, he was with the IBM Research Lab, Haifa. Currently, he is a Research Fellow at the Technion R&D Foundation and the Samuel Neeman Institute for Advanced Studies in Science and Technology, Haifa. His research interests include video analysis, tracking, motion-based recognition, PDE methods for image processing, and medical imaging.

Ehud Rivlin received the B.Sc. and M.Sc. degrees in computer science and the M.B.A. degree from the Hebrew University, Jerusalem, Israel, and the Ph.D. degree from the University of Maryland, College Park.

Currently, he is an Associate Professor in the Computer Science Department, Technion–Israel Institute of Technology, Haifa. His current research interests include machine vision and robot navigation.

Michael Rudzsky received the Ph.D. degree in physics and mathematics from the Institute of Space Research, Moscow, U.S.S.R., in 1980.

He worked in the Scientific and Industrial Association for Space Research in Baku, Azerbajan, until 1990. Since 1991, he has been a Research Fellow in the Physics Department and since 1995 at the Computer Science Department, Technion–Israel Institute of Technology, Haifa. His current research interests include computer vision, pattern recognition, and compression of images.

Mike Adel received the B.Sc. degree in physics from the University of New South Wales, Sydney, Australia, in 1983 and the D.Sc. degree in solid state physics from the Techion–Israel Institute of Technology, Haifa, in 1989.

Since graduation, he has held positions as an Industrial Physicist in areas ranging from electro-optical remote sensing, biomedical diagnostics, and more recently semiconductor metrology. Since 1999, he has been employed by KLA-Tencor, Migdal Ha'emek, Israel, where he currently holds the position of Director of Advanced Development for the Optical Metrology Division. He is also an Adjunct Lecturer to the Technion Industrial Engineering faculty where he teaches product development in the MBA program.