Online Bivariate Outlier Detection in Final Test Using Kernel Density Estimation

H.C.M. Bossers, J.L. Hurink, G.J.M. Smit
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
h.c.m.bossers@utwente.nl

Abstract—**In parametric IC testing, outlier detection is applied to filter out potential unreliable devices. Most outlier detection methods are used in an offline setting and hence are not applicable to Final Test, where immediate pass/fail decisions are required. Therefore, we developed a new bivariate online outlier detection method that is applicable to Final Test without making assumptions about a specific form of relations between two test parameters. An acceptance region is constructed using kernel density estimation. We use a grid discretization in order to enable a fast outlier decision. After each accepted device the grid is updated, hence the method is able to adapt to shifting measurements.**

I. Introduction

In parametric integrated circuit (IC) testing, measurements usually need to be within certain specification limits; devices that are not within these limits will be rejected. However, these limits are usually quite wide, since they need to cope with lot-to-lot and part-to-part variation [1]. Therefore, outlier detection methods are used in order to detect deviating behavior in the measurements; the outliers. An outlier is defined as a measurement which differs significantly from an expected pattern of behavior, even if the measurement is within its specification limits [2]. The rationale for outlier detection is that deviating behavior is probably caused by some latent defect and entails reliability risks. This is supported by empirical evidence [3]. For a more extensive discussion of outlier detection in IC testing see, for example, [4].

In this paper, we specifically focus on *bivariate online* outlier detection, i.e. online outlier detection in a combination of two test parameters. In the next section we elaborate on this in more detail. *Online* means that we have to apply outlier detection during testing and perform outlier detection each time new measurements become available; this in contrast to (offline) post-processing. Until now, most outlier detection methods (both in semiconductor testing and elsewhere) are post-processing methods; see, for example, [5]. Post-processing is possible in Wafer Test where after testing, each die still can be identified based on its coordinates on the wafer. However, in Final Test (also called package test) this identification is usually not possible. Even if it is possible, the filtering of the detected outliers from the tested lot requires an additional process step in the testing process, which is undesirable. Therefore, we specifically focus on online outlier detection which is applicable to Final Test. The online element leads to the requirement that methods should be extremely fast, since otherwise testing time increases, which may be expensive. However, due to the handling and testing time, there is some time before the arrival of the next device. As a consequence, only the outlier decision needs to be made very rapidly. Afterwards, there is some time (dependent on the handling and testing time) which can be used for updating of limits and statistical analysis, before the next decision has to be taken.

Our main contribution is the development of an algorithm which can be used to perform online bivariate outlier detection and is applicable at Final Test, without any assumptions about the relation between the parameters. Furthermore, our method is able to adapt to shifting data. The basis of the method is a rolling horizon approach: it uses test data of a certain number of previous good (non-outlier) devices to construct a rejection region for the upcoming device. Note that we do not consider the test pair selection, for some guidelines we refer to [6].

II. Multivariate Outlier Detection

In a previous paper [7] we have introduced a *univariate online* outlier detection method. However, univariate methods are only able to detect unexpected measurements within a single test. Due to all kinds of relations between tests, it may occur that only some combinations of values are very strange, whereas the values on itself are not deviating in their own dimension. The most simple example of this phenomena occurs if two tests have a strong linear relation. In this case, it is easy to form combinations of values that are normal in their own dimension, but if combined deviate significantly from the linear relation. In [6], this linear relation is exploited for outlier detection. However, there also exist all kinds of nonlinear relations. Therefore, we now focus on bivariate outlier detection that is suitable for arbitrary relations between two test parameters. Most of the used concepts can be extended to higher dimensions, but then very fast the statistical curse of dimensionality shows up. This means that in order to make reliable statistical estimations the number of required observations grows very fast. However, this is not very practical for online outlier detection, since using a very large horizon means both long initialization time and less (slower) adaptivity to measurement shifts. In the following subsections we elaborate on the concepts of distance and scaling. Furthermore, we discuss a method which can be used for outlier detection: kernel density estimation (KDE). This method is very suitable for our problem, since it does not make any assumptions about the relations in the data.

A. *Distance*

The key in multivariate outlier detection is the concept of distance or dissimilarity between two (d-dimensional) observations x and y, or between an observation and some 'center of mass' $\bar{x}$. In the univariate case we can simply take the (absolute) difference of the observations. In order to obtain a scale-invariant distance, this difference is divided by the standard deviation. However, in the multivariate case we have d-dimensional vectors and distance can be calculated in several ways. Some well known distance functions are the Euclidean distance and Mahanalobis distance. The Euclidean distance between two vectors x and y is defined as:

$$\text{dist}_E(x,y) = \sqrt{(x-y)^T(x-y)} \tag{1}$$

However, this distance is extremely sensitive to scale differences in the dimensions, since the deviation in each dimension is equally weighted. Furthermore, it ignores correlations in the data, so each deviation is equally weighted, regardless whether it is a deviation which could be expected on basis of correlation or a deviation which is opposite to the expected correlation. As a consequence, the dataset should be more or less equally scaled in each dimension. Furthermore, the outlier detection method itself should take correlations into account.

Mahanalobis distance is based on correlations and is scale-invariant, but requires a covariance matrix S of the data. It is defined as:

$$\text{dist}_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)} \tag{2}$$

Note that the Mahanalobis distance reduces to the Euclidean distance if S is the identity matrix, i.e. there are no correlations and the data is equally scaled. The key of Mahanalobis distance lays in the re-scaling of deviations in each dimension by use of the covariance matrix. This sounds very useful for the multivariate outlier setting, but we have to make some remarks. The first is regarding the robustness: as in the case with standard deviation in the univariate case, also the covariance matrix is very sensitive to outliers [8]. Robust alternatives can be computed but are computationally more expensive. Secondly, we should remark that the covariance matrix is based on linear dependence, so non-linear relations are usually not well captured by the covariance. This can be clarified with a simple example: let X be a standard normally distributed random variable and let $Y = X^2$. Then the covariance between those variables is 0, but Y is completely determined by X. So Mahanalobis distance does not seem suitable for our problem. We have decided to use Euclidean distance and to cope with the scaling problem and data dependencies within our method.

B. *Scaling*

Scaling is necessary in datasets with unequally scaled variables. The most well known technique is autoscaling or standardization. Suppose we have n datapoints in d-dimensions: $x_{ij} : i = 1, \dots n, j = 1, \dots d$. Let $\bar{x_j}$ denote the mean and s_j the standard deviation of the j^th dimension. Then the autoscaled data z_{ij} is given by:

$$z_{ij} = \frac{x_{ij} - \bar{x_j}}{s_j} \tag{3}$$

The use of mean and standard deviation immediately shows the sensitivity to outliers. So probably other scaling techniques are more appropriate. For example robust scaling, where the median and MAD (Median Absolute Deviation around the median) are used instead of mean and standard deviation. We refer to [9] for a more extensive discussion of different scaling methods. Until now we neglected the fact that the scaling method has to work online, since we need the scaling for online outlier detection. Therefore, we propose to use the univariate method presented in [7]. This online method essentially computes the standardized distance from the (robust) mean and rejects a device if the distance exceeds a certain threshold. Furthermore, this method protects itself against outliers by using robust statistics in the initialization phase, i.e. the start-up of the method. So we can use this standardized distance in order to obtain equally scaled variables.

C. *Kernel Density Estimation*

Data dependencies can be taken into account by using a density-based approach. The aim is to estimate a density, which can be used to distinguish between low and high density areas. One way to do this is to assume some parametric family of distributions (like the normal distribution) and estimate its parameters. However, real-life data usually does not come from the assumed distribution, so it might be better to use only the data to determine the shape of the distribution, without further assumptions about the class of distributions. This can be done with Kernel Density Estimation (KDE). The main advantage of KDE is that it does not make any assumption about the shape or linear relations between the different variables. KDE can be used for outlier detection, see for example [10].

KDE basically means that each observation is given an equal probability mass of $1/n$, which is spread out in a certain region around the observation. This region is defined by the kernel function and the bandwidth parameter. The kernel function determines the shape of the region and the bandwidth parameter controls the size of the region (note that the total mass is the same for each observation). Then a density estimate for a certain point x can be obtained by aggregating the values of all individual kernels on that point. Mathematically, $\hat{f}(x)$, a density estimate for x, looks as follows:

$$\hat{f}(x|X_1, \dots, X_n) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{x - X_i}{h}\right)$$

where $X_1, \dots, X_n$ are given d-dimensional observations, h is the bandwidth parameter and K a kernel function. If $\hat{f}(x)$ is smaller than a certain threshold, x lies in a very low density region and it can be characterized as an outlier. In the 2-dimensional case, this can be interpreted as follows: each observation has an influence within a circle around it with radius h, the bandwidth parameter. The values in this circle are determined by the kernel function which has a maximum at the observation itself and decreases to zero by moving from the center until the (euclidean) distance to the center is equal to h. A density estimate for x can be obtained by adding the kernel values of x for each observation X_i. In Figure 1 an example is shown. The black dots indicate the (2-dimensional) observations, the surface represents the resulting density estimate from these six observations. Because of the low number of observations, the effect of a single observation is clearly visible as a 'bubble' centered around an observation. Furthermore, when two (or more) observations are close to each other, the bubbles intersect and the values of the kernels in the intersection area are added.

Note that basic KDE, as described above, requires equally scaled variables, since the Euclidean distance is used. Scale differences can be taken into account within KDE by using variable bandwidths for each dimension [11]. However, for online outlier detection we prefer to use the basic KDE since using variable bandwidths means that the update of the density estimation after each measurement becomes more complex and computationally demanding.

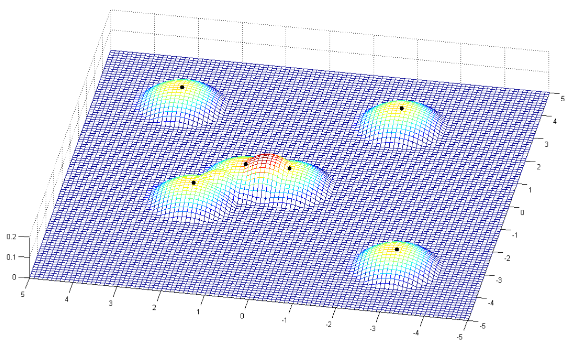

Figure 1. Example Kernels

III. Online Kernel Outlier Detection

In the previous sections, we discussed the kernel density estimation as an offline outlier detection method. However, for Final Test we need an online method. Thus, in this section we transform the kernel density method to an online version.

As in the univariate case, a rolling horizon is used. So each time a decision is based on the previous n observations. In our description, we mainly focus on the 2-dimensional case. Most concepts can easily be generalized to higher dimensions, however the required number of observations for reliable density estimations grows very fast [11], as the space in which observations can occur, grows exponentially. The use of Euclidean distance in the kernel functions implies that more or less equally scaled dimensions are required. Therefore, we use the univariate rolling horizon method described in [7] to standardize the measurements. Furthermore, this method detects and removes the gross outliers in each dimension. After this preprocessing, we apply multivariate outlier detection on the standardized measurements. A schematic overview of our method is presented in Figure 2. In the next subsection, we describe the (online) bivariate outlier detection method in more detail.

1) Measurements $x_i = [x_{i1}\ x_{i2}]$ become available
2) Apply Univariate Outlier Detection on x_{i1} and x_{i2} separately (based on previous n_1 observations)
3) If both x_{i1} and x_{i2} are accepted: return standardized values $y_i = [y_{i1}\ y_{i2}]$
4) Apply Bivariate Outlier Detection on y_i (based on previous n_2 observations)

Figure 2. Online Outlier Detection Method

A. *Online Bivariate Outlier Detection*

For the bivariate outlier detection we use a rolling horizon approach: based on the previous n non-outlier observations we decide if the next observation is an outlier or not. Note that in our case, the observations are pairs of measurements of two test parameters. The method consists of two phases: the initialization phase and the rolling horizon phase. We first introduce the method, and discuss afterwards how to deal with the initialization phase.

In each iteration of the rolling horizon phase, we have an horizon $H = \{Y_1, \ldots, Y_n\}$ of n standardized observations Y_j, and a new standardized observation y_i on which we have to decide if it is an outlier or not. A straightforward approach would be to compute the kernel density estimate of y_i:

$$f(y_i|H) = \frac{1}{nh^2} \sum_{j \in H} K\left(\frac{y_i - Y_j}{h}\right) \tag{4}$$

and characterize y_i as outlier if $f(y_i|H)$ is smaller than a certain threshold T. A disadvantage of this approach is that evaluation (4) takes some time, hence the outlier decision cannot be taken very fast. Ideally, we are prepared for taking a fast decision, after which update computations may be performed while the testing process continues until the next device is tested. To achieve this, we need to store density estimates of the horizon. In kernel density estimates, the effect of a single observation can easily be added or subtracted, so updating is very easy. However, it is impossible to store density estimates for all points, since the data comes from a continuous distribution. Hence, a discretization is required such that we assign regions to a limited number of points, equally spread among the (2-dimensional) continuous measurement space. Note that this space is bounded, since the univariate outlier detection method rejects measurements with an (absolute) standardized value larger than a certain tolerance level k. So the measurement space consists of all points in $[-k, k] * [-k, k]$. A discretization can be obtained by constructing a grid of horizontal and vertical lines each having a distance of δ in between. The intersections of these lines are used as points in the discretized (storage) space DS. Each continuous observation is assigned to the nearest point in the discretized space by rounding.

The level of discretization can be controlled by the distance δ between the lines in the grid. The number of points in DS is in the order of $O(k^2/\delta^2)$. Furthermore, for updating the density estimate, only the area around the new observation is influenced, which is determined by the bandwidth h and δ. The number of points which need to be updated each iteration is in the order of $O(h^2/\delta^2)$. So a low value for δ results in a large number of points, which means a good approximation, but also more computation time and more demanding memory requirements. An overview of the method is presented in Figure 3. The index j is used for the oldest measurement in the Horizon, N is the total number of devices in the test run and T is the threshold for deciding if an observation is an outlier or not.

```
Initialization:
  Construct Baseline Horizon $H = \{Z_1, \ldots, Z_n\}$
  Compute $f(x|H)\ \forall x \in DS$
  set $i := n$, $j := 0$

Repeat
   $i = i + 1$
   $z_i$ = discretize($y_i$)
   Look up: $f(z_i|H)$
   If $f(z_i|H) \leq T$
      then outlier detected;
   Else
      update density estimate
      $j = (j \mod n) + 1$
      $Z_j = z_i$
Until $i = N$
```

Figure 3. Multivariate Rolling Horizon Method

Note that this discretization is an approximation to the exact density shown in formula (4), since we approximate the density of a new observation with the density of the nearest discretized point. However, we have moved almost all required calculations after the outlier decision moment. Only the discretization of the new observation and lookup of corresponding density needs to be done before the outlier decision can be made. Furthermore, the required calculation time decreases if δ increases. If δ is chosen not too small compared to h, the total computation time per observation (both before and after the outlier decision) is smaller than using the exact formula (4). This is because for each new observation, the evaluation of (4) requires n (bivariate) kernel evaluations. In the discretization we can compute the kernel beforehand, during the update process only the center of the kernel changes, so we can simply add/subtract the (pre-computed) kernel values for all points in the neighborhood.

B. Parameters

The presented method is influenced by four parameters: discretization level δ, bandwidth h, threshold T and horizon size n. Furthermore, the type of kernel K can also be seen as a parameter, but the kernel type is usually not very important and the most convenient type can be chosen [11]. Since our final purpose is outlier detection, it is enough if we can distinguish low density points from high density points. Hence, we can use integer valued kernels in order to save memory and computation time. Therefore, we use a modified boxcar kernel: all points within a distance h from the center get value 1, but points within distance $0.5h$ or $0.25h$, get value 2 or 3 respectively. So we have an integer valued kernel where points closer to the center have a higher value. The value of δ has a large impact, on the one hand on approximation performance, and on the other hand on computation time and memory requirements. The bandwidth h and horizon size n are connected, if n increases, h can be decreased. The horizon size needs to be large enough to obtain reliable density estimates, but small enough for practical use. There are some rules for computing the optimal bandwidth [11]. However, these rules only provide optimal bandwidths if the density which needs to be estimated is multivariate normal. Furthermore, our purpose is outlier detection, thus only for the low density regions we need reliable density estimates. The density estimates for the dense regions are not very important, as long as there is a clear distinction between high and low density regions. Therefore, we recommend to do experiments to obtain good settings for the parameters. The threshold T needs to be set based on the desired strictness of the method, but also the kernel type and the bandwidth need to be taken into account.

C. Initialization Phase

Until know, we ignored the initialization phase. That means that all devices in the baseline escape outlier detection. One solution could be to analyze afterwards if the baseline contains outliers. If yes, then the complete baseline should be retested in order to find the outliers. However, in case of a lot size of 2500 and a baseline of 250, this would result in 10% more test time. So it is worthwile to incorporate outlier detection also during the initialization. For the first devices, online analysis is only possible if data of previous lot can be used. After a few devices, some information of the current lot is available which can be used for (rough) outlier detection. An example of outlier detection in the baseline is presented in [7], where we consider the univariate case. There we use robust statistics, in order to avoid the influence of outliers and start outlier detection early, for example after 30 or 50 devices. However, we have to protect against possible outliers, therefore we have to be more conservative and take a higher threshold. In this way, we can distinguish between normal devices and the 'still unsure' devices. Usually, all or almost all of the 'still unsure' devices also turn out to be normal devices afterwards, but at the decision moment there was not enough evidence for this. Depending on the required conservativeness and variation in the baseline, we can already classify about 70-90 % as normal during the initialization phase. This means that in case a retest of the baseline is required, we only have to retest about 10-30 % of the baseline.

IV. Experiments

In this section we present the results of some initial experiments with the presented method using real-world data. Furthermore, we show the effects of the different steps in the method. We performed three types of comparisons, first we show the effects of the discretization compared to the exact formula (4). Then we compare our online outlier detection method with offline outlier detection (also using KDE) on scaled data. The first two comparisons are made based on the scaled data, in the third comparison we show the results of our online outlier detection method on original data. This can vary because the measurements are online scaled, hence the scaling can differ along the process.

A. Discretization Effects

We start with the comparison of our discretization method and the exact online density computation (formula 4) for different discretization levels ranging from 0.1 to 0.75. Lower levels are not practical since then computation time and memory will drastically increase; higher levels are not expected to deliver good approximations. In Figure 4 we plotted the computed densities of both methods on test data of a combination of two tests. However, since our aim is outlier detection we only looked at the the low density regions; there we need reliable density estimation. The horizontal and vertical line indicate the 2^{nd} percentiles in both directions. The x^{th} percentile is the value of the density below which x percent of the observations fall. As a consequence, all observations below the horizontal line or to the left of the vertical line lie in the lowest density regions. This gives an indication which devices would be rejected by both methods if the objective was to reject all devices below the 2^{nd} percentile. All observations in the upper left and lower right quadrant would then be treated differently, so these quadrants should be as empty as possible. As can be seen in Figure 4 we can obtain a quite good approximation with $\delta = 0.1$. A discretization level of 0.25 also performs reasonably well, although there occur some minor differences in both methods. The higher levels show a degrading performance, but for $\delta = 0.5$ the linear pattern is still clearly visible, especially for the lowest density points which are most relevant. Note that we have chosen a rather high percentile, in practice the desired rejection rate is usually lower.

B. Online Effects

In this section we compare the results of the presented online method versus an offline outlier detection method. The offline outlier detection method is executed on all (scaled) observations at the same time, so this can only be done if all measurements are available. For the online method we used a horizon size $n = 250$ and discretization parameter $\delta = 0.1$, since this value provides very accurate approximation,

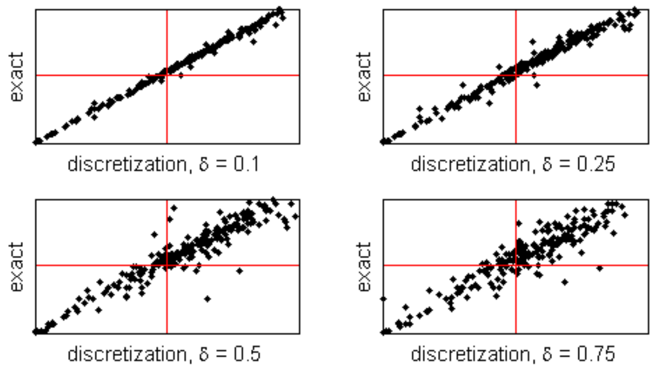

Figure 4. Approximation Results (lower 2%-percentile)

so differences in the results are most likely the result of differences between online and offline outlier detection. As offline outlier detection method we also use KDE, but with adaptive bandwidth in order to obtain smoother density estimation in the tails. As a result, observations in the low density regions get a higher bandwidth and are more flattened. For more detailed information about adaptive bandwidth estimators we refer to [12] and [11].

In Figure 5, a combination of two test parameters is plotted of 4600 devices. The gray points are the normal points, the black points are detected as outlier both online and offline. The circled points are only detected online, the points with a plus sign are only detected offline. As can be seen the results of methods coincide on the majority of points, however there are three only offline outlier and five only online outliers.

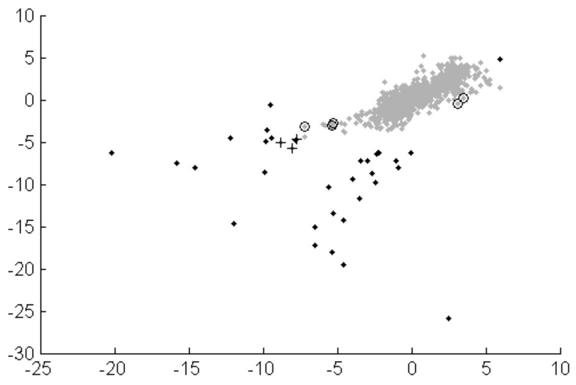

Figure 5. Online (o) vs. Offline (+)

The same pattern can be seen in Table I. This table shows a comparison of online versus offline with different bandwidth and threshold settings on 40,000 devices. Note that each parameter setting results in a different rejection percentage. For a fair comparison we look at the same percentage offline outliers, by taking those observations with the lowest density such that the number of online and offline outliers is the same for each parameter setting. The offline suspects are all observations within twice the rejection percentage, so the suspects are those observations which are also in the lower density regions. As can be seen is the number of outliers the highest with a low bandwidth and decreases if the bandwidth increases. Therefore, we used a higher threshold for $h = 2.5$. Furthermore, a large part of the online outliers is also offline outlier, and from the remaining part only a small fraction is offline regarded as normal. On the other hand, the number of offline outliers which are not detected is still relatively high: around 10% of the total number of outliers. However, from yield perspective this is not neccessarily bad: it is better to detect most outliers without much overkill, than detecting all outliers with large overkill.

(h,T)	(1.5,1)	(2,1)	(2.5,2)	(2.5,3)
online outlier, offline outlier	317	250	241	279
online outlier, offline suspect	47	34	29	23
online outlier, offline normal	5	1	1	1
offline outliers not detected	52	35	30	24

Table I
PARAMETER COMPARISON

C. Results in Original Data

In this section we analyze the results of our online outlier detection method in the original data, i.e. without scaling. In Figure 6, four plots are shown with data points of two tests. The online detected outliers are circled. The first plot is on an aggregated dataset of three wafers, the other three plots only contain data of the individual wafers. In the aggregate plot, it seems if there are some online outliers detected which are quite in the middle of a dense cloud of points. However, in the individual wafer plots there are no outliers within the dense cloud of points (except for one in wafer 19). Apparently, there has occurred a shift in the measurements. This shows that outlier detection should not be executed on a very aggregate level, if possible. Otherwise, the variance is likely to increase which will have a negative effect on outlier detection possibilities. See for example [5], for a paper about the importance of variance reduction for outlier detection.

V. CONCLUSIONS

In this paper, we have presented a new method to deal with bivariate online outlier detection. The method is based on kernel density estimation, but in order to make fast outlier decisions we use a discretization such that previous density estimates can be stored. The level of discretization can be used to control the accuracy on one hand and required memory and update computation time on the other hand. Initial experiments show that our method is able to detect outliers, and there are only minor differences with offline detection. More experiments are required to determine the effectiveness of our method. This also can give more insight into favorable parameter settings. Further research is required to determine the effects of outlier detection on dppm values, that is, investigate which outliers are indeed potential failures.

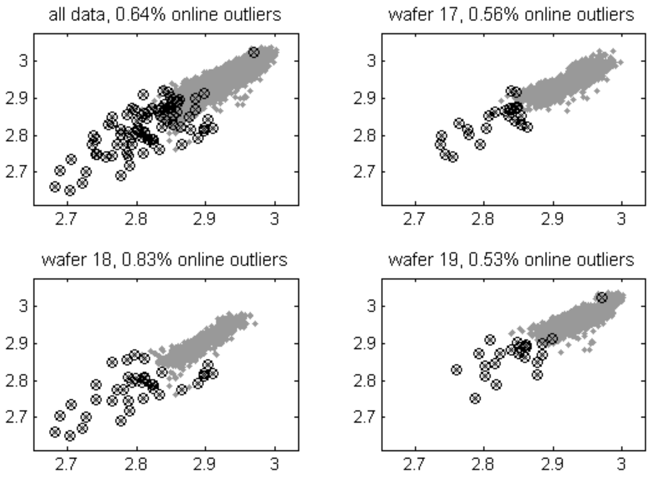

Figure 6. Online Outliers in Real Data

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support of AgentschapNL and the province Overijssel (Pieken in de Delta). Furthermore, the authors would like to thank Patrick Zwegers from Salland Engineering and the DATA reviewers for their valuable comments and feedback.

REFERENCES

[1] J. L. Roehr, "Measurement Ratio Testing for Improved Quality and Outlier Detection," in *Proceedings of the International Test Conference*, IEEE, 2007.

[2] P. O'Neill, "Statistical Test: A New Paradigm to Improve Test Effectiveness and Efficiency," in *Proceedings of the International Test Conference*, pp. 1–10, IEEE, 2007.

[3] R. Madge, M. Rehani, and K. Cota, "Statistical Post-Processing at Wafersort - An Alternative to Burn-in and a Manufacturable Solution to Test Limit Setting for Sub-micron Technologies," in *Proceedings of the VLSI Test Symposium*, pp. 69–74, IEEE, 2002.

[4] S. S. Sabade, *Integrated Circuit Outlier Identification by Multiple Parameter Correlation*. PhD thesis, Texas AM University, 2006.

[5] W. R. Daasch and R. Madge, "Variance Reduction and Outliers: Statistical Analysis of Semiconductor Test Data," in *Proceedings of the International Test Conference*, IEEE, 2005.

[6] L. Fang, M. Lemnawar, and Y. Xing, "Cost Effective Outliers Screening with Moving Limits and Correlation Testing for Analogue IC's," in *Proceedings of the International Test Conference*, IEEE, 2006.

[7] H. C. M. Bossers, J. L. Hurink, and G. J. M. Smit, "Online Univariate Outlier Detection in Final Test: A Robust Rolling Horizon Approach," in *Proceedings of the European Test Symposium*, IEEE, 2011.

[8] P. Rousseeuw and B. van Zomeren, "Unmasking Multivariate Outliers and Leverage Points," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 633–639, 1990.

[9] L. H. Chang, R. J. Pell, and M. B. Seasholtz, "Exploring process data with the use of robust outlier detection algorithms," *Journal of Process Control*, vol. 13, pp. 437–449, 2003.

[10] L. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier Detection with Kernel Density Functions," in *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 61–75, IEEE, 2007.

[11] B. Silverman, *Density Estimation for Statistics and Data Analyis*. Chapman and Hall, 1986.

[12] H. Stratigopoulos, S. Mir, E. Acar, and S. Ozev, "Defect Filter for Alternate Test," in *Proceedings of the European Test Symposium*, IEEE, 2009.