POLITECNICO
MILANO 1863

Reference based defect detection using Foveated and classical NL-means

Professor:

Boracchi Giacomo

Student:

Bertarini Andrea

1. Introduction

Defect detection is an important problem, especially in industrial application. One wants to discriminate between good and bad products automatically, since manual detection is error prone and not accurate. Moreover, it helps us to define the error rate of a product during the process, this information is helpful because can be used to tune the production parameters in order to improve the quality of the final product.

The method we are going to study has been proposed by M. Zontak in [1], in particular is a reference based method, the detection method works with two images, a clean image without defect called reference(ref) and one with a possible defect called source (src). The algorithm is based on NL-means, a pixel in the src is re-estimated as a weighted average of pixel in the ref. Weight are found by applying a similarity measure based on the patch distance. In [1] as patch distance is proposed the windowed quadratic one, we introduce also the foveated-distance as proposed by Foi, Boracchi in [2], which may improve the performance the algorithm. In [2] is shown that foveated-distance outperform windowed-distance in term of quality reconstruction. In fact, we are going to determine if exist a relation between the algorithm performance and the quality reconstruction.

The method works with every image and it does not need image registration, however best results are obtained in case of a frequent pattern. For example, in [1] it is applied to detect anomaly in silicon wafer where there are many copies of the same electrical components.

2. Background

We model a noisy image as

$$v(i) = v_0(i) + \eta(i) \text{ (1)}$$

Where v_0 is the noise free image and η is an additive Gaussian white noise.

2.1. NL-means

NL-means is a de-noising algorithm proposed by Buades, Coll, and Morel in [3]. NL-means stands for Non-Local because, in principle, the algorithm uses all the pixels in the image. In practice due to a high computational cost, the algorithm considers only a small search neighborhood around each pixel, however this restriction does not affect the results.

2.1.1. NL-means formulation

The estimated value of a pixel is

$$\hat{v}(i) = \sum_{j \in Ni} w(i,j) * u(j) \text{ (2)}$$

Which means that the value of a pixel is recomputed as a weighted mean of all pixels belonging to Ni, which is the neighborhood we mentioned before. Note that the dimension of Ni affects the computational complexity of the algorithm.

Instead of considering a single pixel, the algorithm considers a squared patch to compute the weights, resorting to a patch similarity measure: the more similar two pixels, the larger the weight. Since we want to give more importance to the center of the patch, it is multiplied by a kernel that smooth out the borders, a common kernel used is the Gaussian kernel, but in principle, any kernel having maximum at the center works as well.

$$d = \left|\left|k * zi - k * zj\right|\right|_2^2 \text{(3)}$$

$$w(i,j) = e^{\frac{d}{h^2}} \text{(4)}$$

Equation (3) is defined as windowed-distance, where zi and zj are the patches for pixels i and j respectively, k is the kernel and h is a filtering parameter controlling the weight decay, it should be equal to the standard deviation of the noise in the noised image. Equation (4) represents how the weights are computed.

$$w(i,j) = 1 \text{ and } 0 \leq w(i,j) \leq 1 \text{ (6)}$$

$$Z(i) = \sum_{j \in Ni} w(i,j) \qquad \text{(7)}$$

$$w(i,j) = \frac{w(i,j)}{Z(i)} \quad (8)$$

Finally, we have to normalize the weight (equation (6) should be true) with equations (7) and (8).

2.2. Foveation

Foi, Boracchi in [2], has proposed a new similarity distance inspired by a peculiarity of the Human Visual System (HVS). Our visual acuity is maximal at the center of the retina, also called fovea, this is a consequence of the inhomogeneous distribution of ganglion cells and cone cells in the retina. Ganglion and cone cells are concentrated in the fovea, they decrease towards the periphery and so the acuity does: it reaches the maximum in the center and decreases toward the periphery.

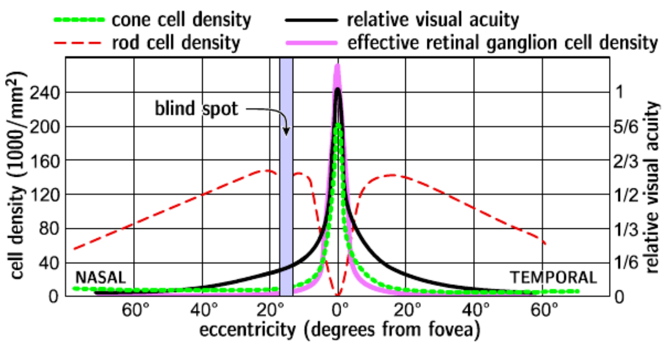


Fig.1) Graph showing the distribution of the cell in the retina

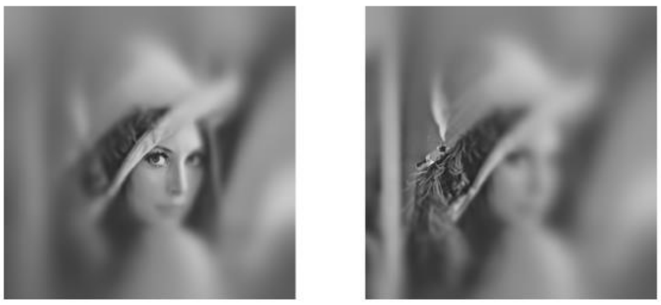


Fig.2) Foveated image of Lena at two different fixation point

Retinal images are sharp at the center, or fixation point, and becomes progressively blurred as the distance from the center increase. To reproduce this effect, we adopt a series kernels with increasing standard deviation, which are convolved with the image to get progressively blurred images. Finally, via F, the foveation operator, we can get the foveated patch for a pixel.

To better understand how F works we can imagine to have a stack of image (Scale space Layers), which is the result of the convolution between the image and the kernels. At the bottom of the stack, there is the sharp image while at the top we have the most blurred one. F can be imagined as a cone: the vertex of the cone overlap with the fixation point in the bottom image, and the patch is composed by those pixels of the images intersecting the cone

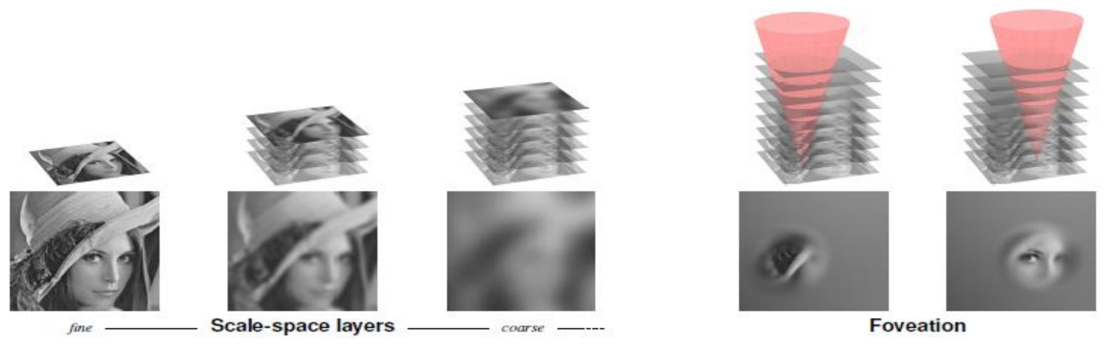


Fig.3) Scale space layers is the result of the convolution between the image and a set of kernels.
The red cone is representing the selection of pixels belonging to a foveated patch

2.2.1. Foveated Distance

We define the foveated distance as follows

$$d^{FOV}\ (i,j) = \left\|F[v,i] - F[v,j]\right\|_2^2 = \left\|z_i - z_j\right\|_2^2 \ (9)$$

This distance can replace equation (3) in the NL-means. F is the foveation operator, which given an input image and a fixation point x gives as output a foveated patch. In [2] it has been also proposed a constrained design for the foveation operator F, this allow in case of structural similarity to have the same result as windowed-distance. The constrained design starts from a gaussian kernel and for every unique value composing it create another kernel for the foveation operator

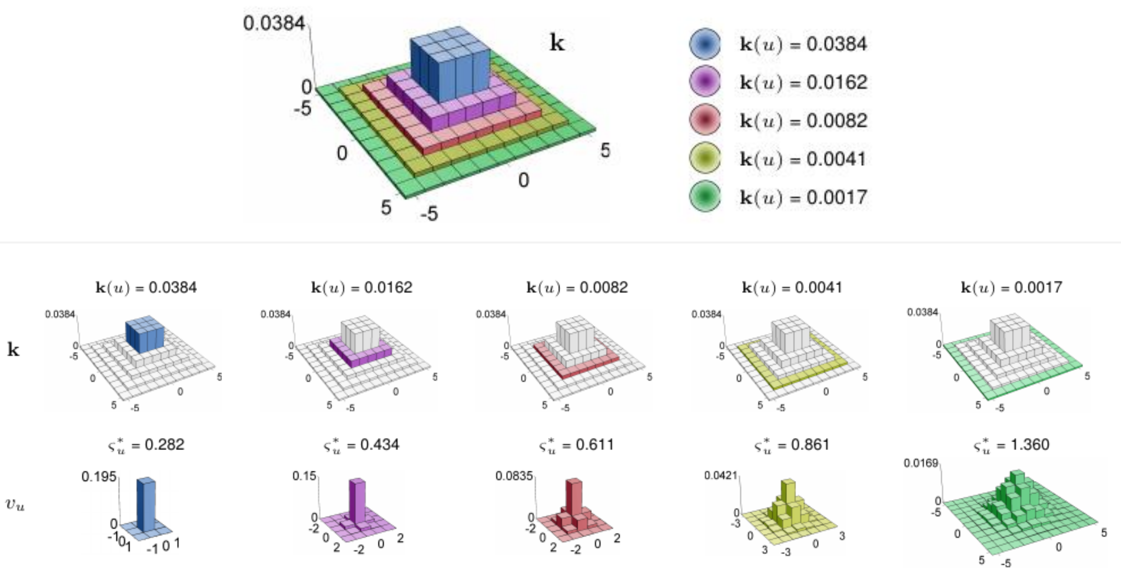


Fig.4) Example with a 11x11 kernel

3. Reference Based Defect Detection

The method proposed by M. Zontak is a reference-based method based on NL-means. As we have seen in section 2.1, NL-means try to reconstruct a pixel as a weighted average of other pixel in its neighborhood. In this case, the search neighborhood moves from the src to the ref: a pixel of the src is reconstructed using pixels in the ref.

$$\hat{v}_{src}(s) = \frac{1}{w(s,s')} * \sum_{s' \in Ns} w(s,s') * v_{ref}(s') \quad (10)$$

3.1. The algorithm

```
1.  {$s$ - src pixel index,$v_{src}$ - source image $\hat{v}_{src}$- reconstructed source image}
2.  for all $s$ ∈ f do{
3.  $z_s$ ⇐ extract patch from $v_{src}$
4.  {$s'$ - ref pixel index, $v_{ref}$ - reference image, $Ns$- search region neighborhood of $s$}
5.  for all $s'$ ∈ $Ns$ do {
6.              $z^i_{s'}$ ⇐ extract patch from $v_{ref}$
7.              $W_i$ ⇐ )
8.              $i \Leftarrow i + 1$
9.              $S_w \Leftarrow \sum_i W_i$
10. }
11. if $S_w$ = 0 {
12.             for all $i$ do{
13.             $W_i$ ⇐ 0
14.             }
15. else{
16.         for all $i$ do{
17.                 $W_i \Leftarrow W_i / S_w$
18.                 }
19.         $\hat{z}_s \Leftarrow \sum_i W_i * z^i_{s'}$ {source image patch estimation using reference neighboring patches}
20.         $D(s) \Leftarrow || \hat{z}_s - z_s ||_2$ {difference image value at pixel s calculation}
21.         $\hat{v}_{src}(s) \Leftarrow \sum_i W_i * v_{ref}$ {source image pixel estimation}
22.     }
23.   }
24. }
```

Alg. 1) Algorithm proposed by M. Zontak et al in [1]

In practice, the algorithm implements equations (2) -(5) of NL-means algorithm.

The algorithm tests two hypotheses:

$$H0: \hat{v}_{src}(i) \rightarrow v_{src}(i) \Rightarrow \mathrm{i} \in \mathrm{A} \quad (11.a)$$

$$H1: \hat{v}_{src}(i) \rightarrow 0 \Rightarrow \mathrm{i} \notin \mathrm{A} \quad (11.b)$$

H0 is true when the pixel i can be possibly reconstructed therefore it does not belong to A, the region of anomalous pixels, otherwise it cannot be reconstructed, goes to zero and belongs to A. In the algorithm lines from (11) to (18) compute the normalization of the weights; if the sum of them is zero then $H1$ is true else $H0$ is true. In practice it is not possible to have $S_w = 0$, so a threshold is used (for example S_w<e^{-15}).

In the introduction we said that this method works very well with repeated pattern images, this is true because in that situation the algorithm could find many similar pixel, so we will have a better reconstruction, thus anomalies can be better spotted. Moreover, we have some other advantages, for example, we do not need to register the images if we assume that the images are not too much misaligned or an image is rotated, we can also keep a small search window and so the computational time is not large.

4. Our implementation

Our solution built upon the structure of the method proposed by M. Zontak in [1]. From that method, we can define two main procedures: the reconstruction and the defect detection. In the reconstruction phase what we can redefine is how patch distance is measured, in particular we adopt two solutions. The first solution is the windowed-distance of the classic NL-means as already done in [1], since we know that it works well in de-noising application and we want to test it. While the second one is the foveated-distance (proposed in [2]), since we know it outperform the windowed-distance in de-noising application, so we are confident that even in this kind of application it will works well.

The second procedure is the defect detection, which we redefine completely. In particular, we resort to a patch-wise difference between the original src and the reconstructed one, as in the following equation.

$$Anomaly(s) = ||\hat{z}_s - z_s||_2^2 \quad (12)$$

Where $\hat{z}_s$ is the patch extracted from the reconstructed image and z_s is the one from the original src. Finally, the intermediate result from equation (12) is passed to a post-processing phase to find anomalous pixels.

4.1. NL-implementation

```
1.  {$s$ - src pixel index, $v_{src}$ - source image, $\hat{v}_{src}$- reconstructed source image}
2.  $k$ ⇐construct gaussian kernel
3.  for all $s$ ∊ $v_{src}$ do {
4.  $z_s$ ⇐ extract patch from $v_{src}$
5.  {$s'$ - ref pixel index, $v_{ref}$ - reference image, $Ns$- search region neighborhood of $s$}
6.  averaged_pixel ⇐ 0
7.  $S_w$ ⇐ 0
8.  for all $s'$ ∊ $Ns$ do {
9.          $z_s^i$⇐ extract patch from $v_{ref}$
10.         $W_i$ ⇐ )
11.         $S_w$ ⇐ $S_w$ + $W_i$
12.         averaged_pixel ⇐ averaged_pixel + $W_i$* $v_{ref}(i)$
13.     }
14. $\hat{v}_{src}$(s) ⇐ averaged_pixel/$S_w$ {source image pixel estimation}
15.  }
16.
17.
18. for $s$ ∊ $v_{src}$ do {
19.  $\hat{z}_s$⇐ extract patch from $\hat{v}_{src}$
20. $z_s$⇐ extract patch from  $v_{src}$
21. Anomaly(s) ⇐ $||\hat{z}_s - z_s||_2^2${difference image value at pixel s calculation}
22. }
```

Alg. 2)

4.2 Foveation-implementation

```
1.  {$s$ - src pixel index, $v_{src}$- source image, $\hat{v}_{src}$- reconstructed source image, ref_patch - patches of the reference}
2.  {$s'$ - ref pixel index, $v_{ref}$ - reference image, src_patch - patches of the source}
3.  $K$ ⇐construct kernel {a number of kernel is created}
4.  for all $k \in K$ do {
5.          $F \Leftarrow v_{src} \times k$ {vector of convolved $v_{src}$}
6.          $F_{ref} \Leftarrow v_{ref} \times k$ {vector of convolved $v_{ref}$}
7.  }
8.  src_patch ⇐ extract patches from $F$
9.  ref_patch ⇐ extract patches from $F_{ref}$
10. for all $z_s \in$ src_patch do {
11. averaged_pixel ⇐ 0
12. $S_w \Leftarrow 0$
13. for all $z_s^i \in Ns$ do {
14.         $W_i \Leftarrow exp(d^{FOV}(z_s, z_s^i) / h^2)$
15.         $S_w \Leftarrow S_w + W_i$
16.         averaged_pixel ⇐ averaged_pixel + $W_i * v_{ref} * (r)$
17.     }
18. $\hat{v}_{src}$(s) ⇐ averaged_pixel/$S_w$ {source image pixel estimation}
19. }
20.
21. for $s \in v_{src}$ do {
22. $\hat{z}_s \Leftarrow$ extract patch from $\hat{v}_{src}$
23. $z_s \Leftarrow$ extract patch from $v_{src}$
24. Anomaly(s) ⇐ $||\hat{z}_s - z_s||_2^2$ {difference image value at pixel s calculation}
25. }
```

Alg. 3)

There are just two differences with the NL-implementation, the first is how patches are extracted and it works as explained in section 2.2, lines 4-7 create two stacks of images for src and ref, lines 8 -9 represent the intersection between the stacks and the cones. The second is how the weights are computed, instead of windowed-distance, foveated-distance is used.

4.3 Post-Processing

Once we have the result of the algorithm, which corresponds to equation (12), implemented by lines 18-22 in the NL-implementation and in lines 21-25 in the foveation-implementation, we pass it to the post-processing algorithm. It is a thresholding followed by a majority voting. The threshold is a high value such that

$$Anomaly(s) < \zeta \quad s \notin A \text{(13.a)}$$

$$Anomaly(s) \geq \zeta \quad s \in A \text{(13.b)}$$

Equations (13.a) and (13.b) corresponds to tests (11.a) and (11.b) respectively. Then the majority voting is applied, it works on a logical image, the result of equation (14).

$$temp(s) = Anomaly(s) \geq \zeta \quad (14)$$

Now we define A(s) as the number of anomalous pixel around s and N(s) the number of normal pixel around s, considering a dimension as large as the patch dimension previously used. The result will be a logical image.

$$final(s) = 0 \leftrightarrow A(s) \leq N(s) \text{(15.a)}$$

$$final(s) = 1 \leftrightarrow A(s) > N(s) \text{(15.b)}$$

5. Experiments

The training set is composed by 26 150x150 images with a regular pattern, where we manually inserted anomalies. Both the methods have been tested with different white noise realization in order to study what happen changing the level of noise; in particular, we have five different noise realizations with increasing σ: 10, 20, 30, 50, and 70.

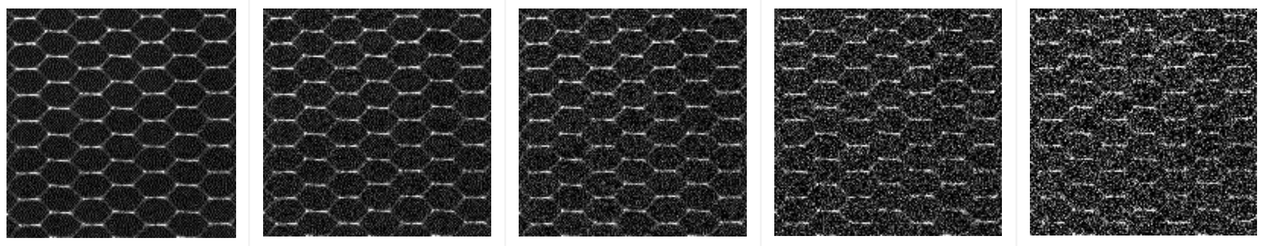


Fig.5) An example image from the training set, just to show how noise affect the image.
From left to right we have increasing value of σ from 10 to 70.

5.1. ROC

Now we are going to show the performance of our implementation, using ROC curves and their AUC for every one of the five realization of the noise and in order to optimize the process we used the optimal parameter as found in [2], which are:

	NL-means Optimal Parameter		Foveation Optimal Parameter	
σ	Patch Size	Search window size	Patch Size	Search window size
10	5x5	11x11	7x7	17x17
20	11x11	9x9	11x11	17x17
30	11x11	9x9	13x13	17x17
50	13x13	11x11	17x17	17x17
70	14x14	13x13	19x19	17x17

Tab 1) Optimal parameter table

ROC are computed for every image for each noise realization we are considering, then are averaged. To obtain a ROC curve we started from the intermediate result of equation (12) and by varying the threshold, ζ we get the curve. Then, it is possible to compute the AUC as the integral of the ROC by using the trapezoidal method.

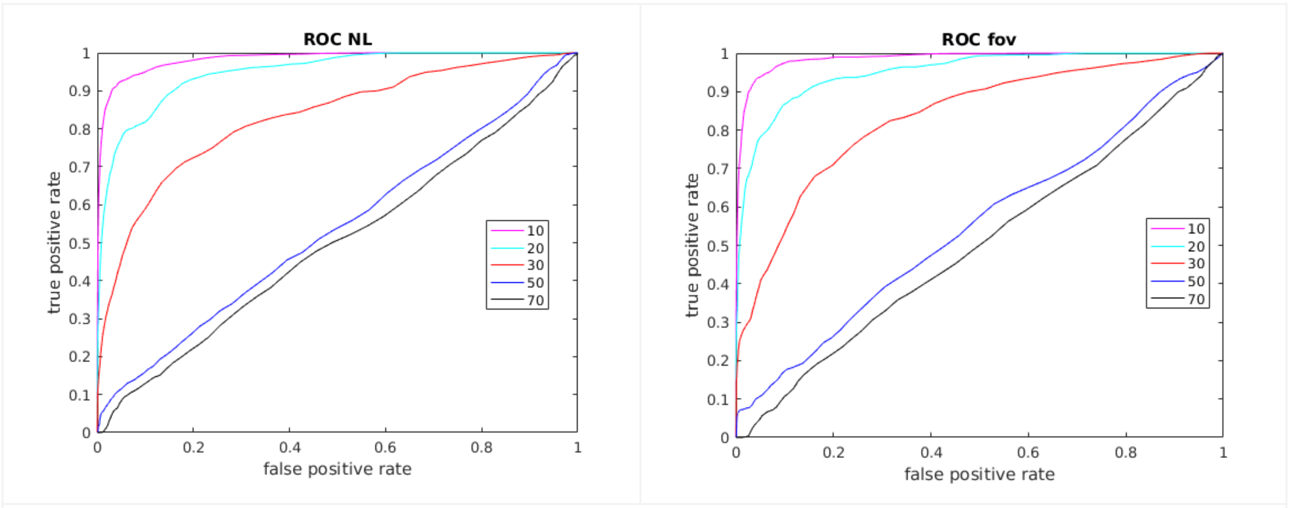


Fig.6) The resulting ROC are an average of the resulting ROC of every single image

σ	10	20	30	50	70	AVERAGE
AUC-NL	0.9871	0.9453	0.8210	0.5359	0.4894	0,75574
AUC-fov	0.9907	0.9494	0.8163	0.5559	0.4978	0,76264

Tab 2) Resulting AUC table

5.2. Setting the anomaly detection Threshold

To find the threshold we consider the results of the algorithm before the post-processing, in practice we took the result of equation (12). From these intermediate results, we removed values corresponding to anomalous pixels, from the remaining values we picked up the threshold as the 99th percentile. The result is the value used in equation (14), which is ζ

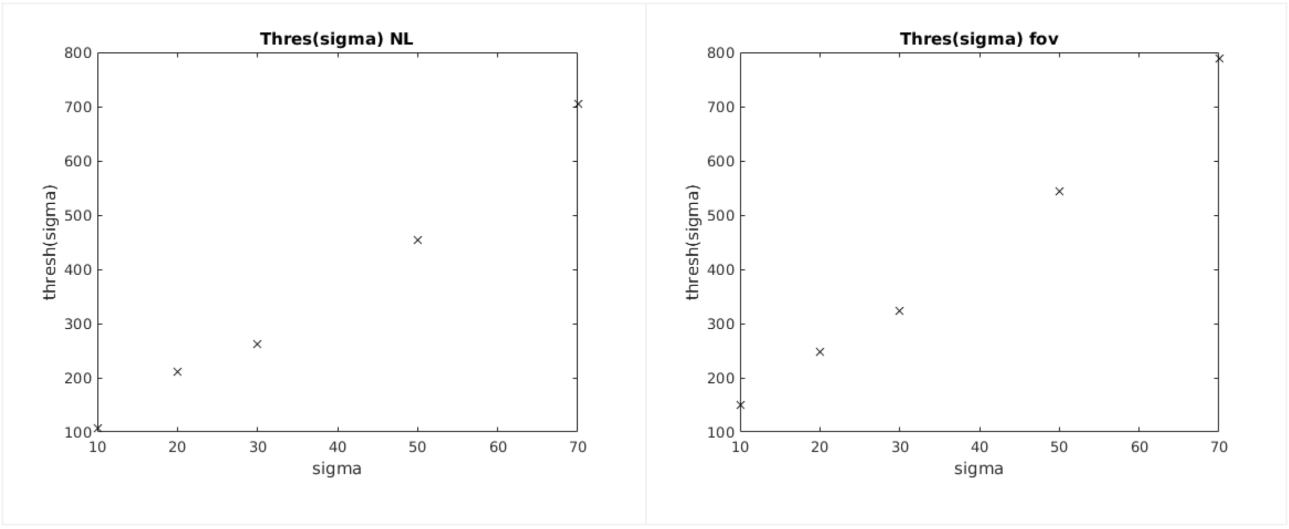


Fig.7) Threshold value in function of σ, note that high σ corresponds to high threshold value

σ	10	20	30	50	70
Threshold-NL	102.7658	188.6992	237.2043	429.2821	669.0157
Threshold-fov	141.7342	218.9689	276.8137	488.9769	710.5433

5.3. AUC vs PSNR

Once an image has been reconstructed, it is possible to compute the PSNR (peak-signal-to-noise-ratio) used as a quality measure for the reconstruction of the src. Then we can plot for every image the detection performance against the quality measure as in Fig. 8. To better understands the result, in figure 9 we plotted the mean relation.

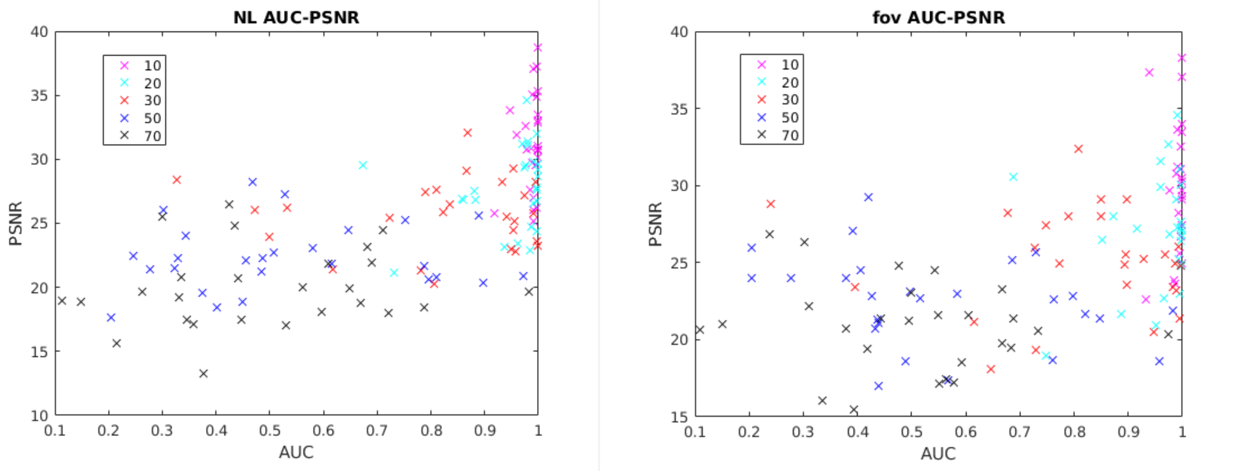


Fig.8) Relation between AUC and PSNR

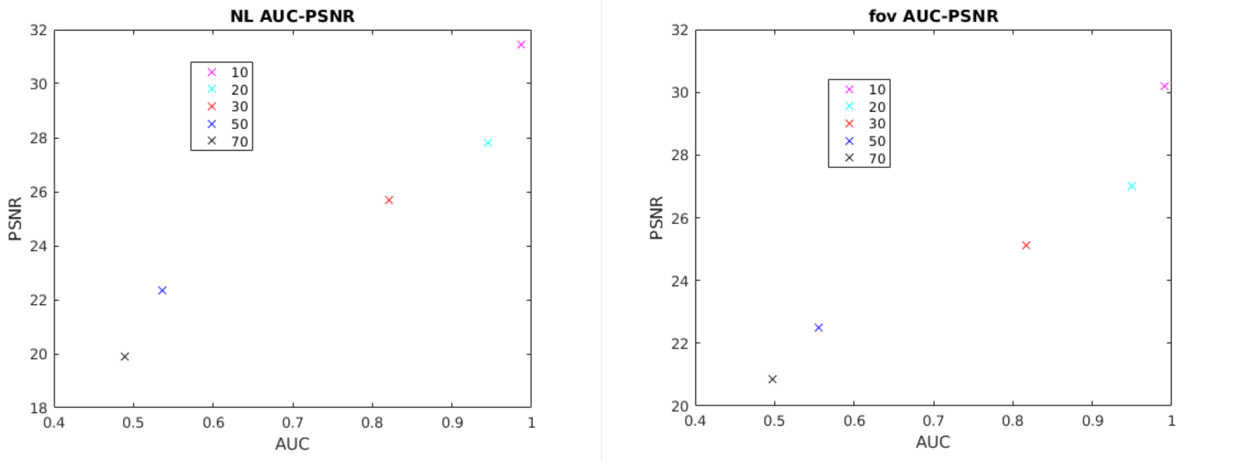
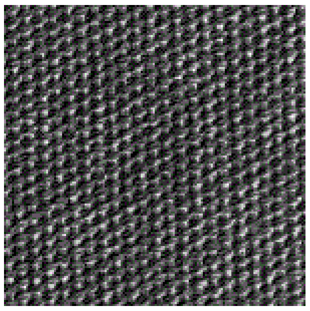


Fig.9) Mean relation between AUC and PSNR

5.4. Result on Test Images

For each test image and for each noise level, we run the algorithm as explained in section 4 so we have a reconstruction phase in which we apply both the patch similarity measure we have seen, windowed-distance and foveated-distance. Finally, in the detection phase, we will use the thresholds found before as in section 5.2. Remember that the result is the output of equation (15), therefore a logical image, where $final(i,j) = 1$ means anomaly.

5.4.1 Test1



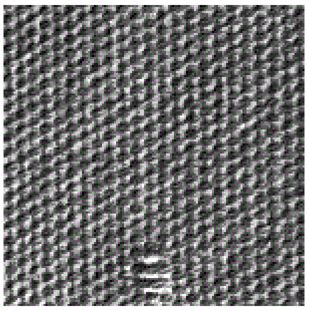


Fig.10) On the left the ref image, on the right the src

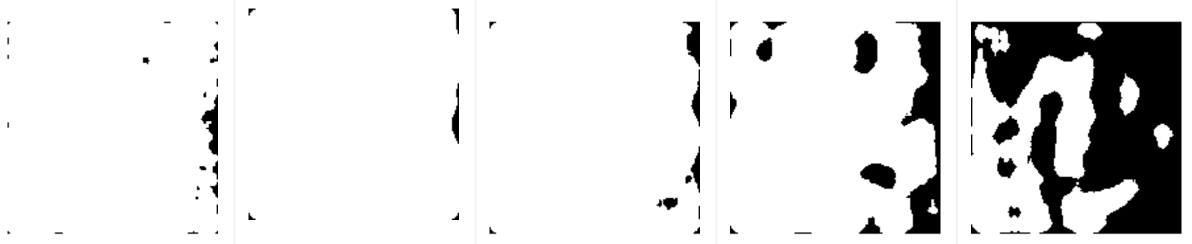


Fig.11) From left to right we have increasing value of σ from 10 to 70.
Top row show the result of foveation-implementation, while the bottom one show the NL-implementation
The white region corresponds to the detected anomalous region.

5.4.2 Test2

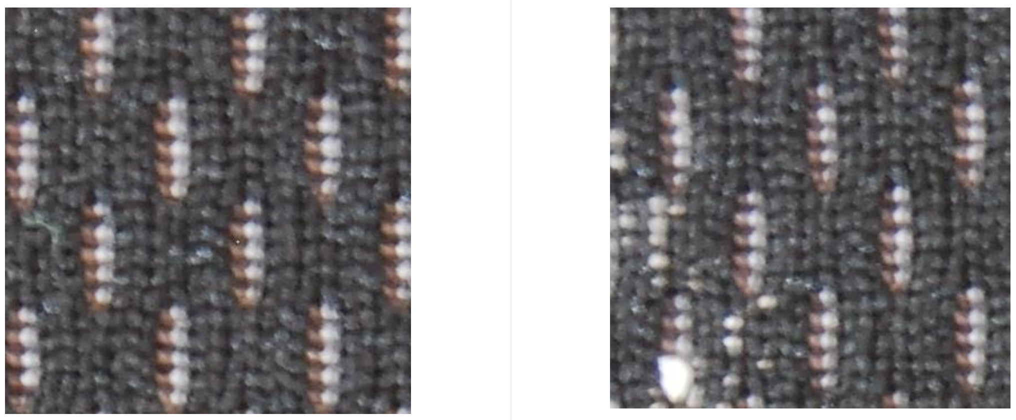
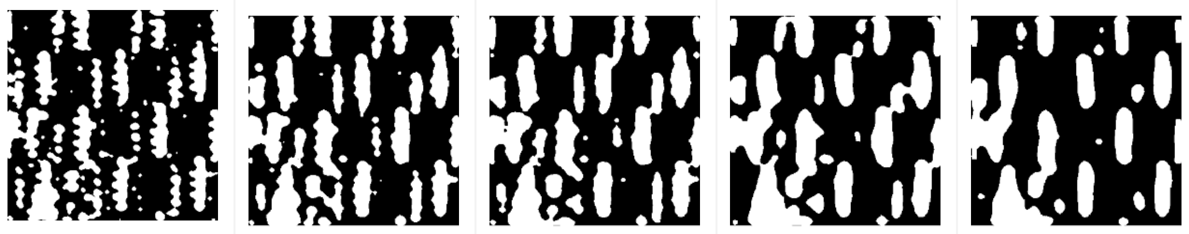


Fig.12) On the left the ref image, on the right the src



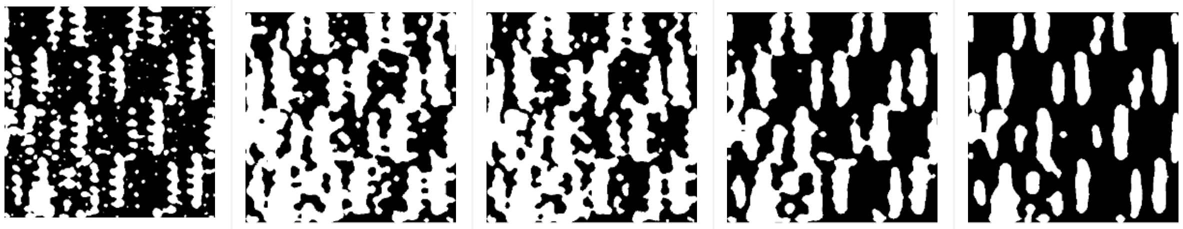


Fig.12) From left to right we have increasing value of σ from 10 to 70.
Top row show the result of foveation-implementation, while the bottom one show the NL-implementation
The white region corresponds to the detected anomalous region.

5.4.3 Test3

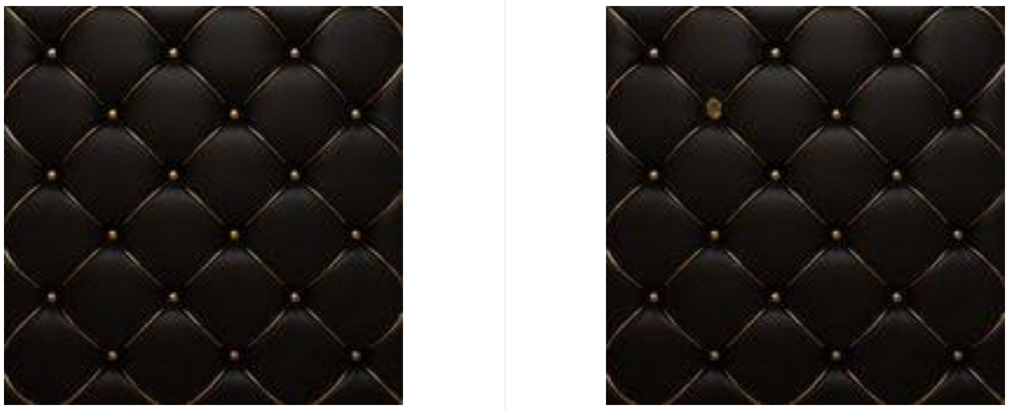


Fig.13) On the left the ref image, on the right the src

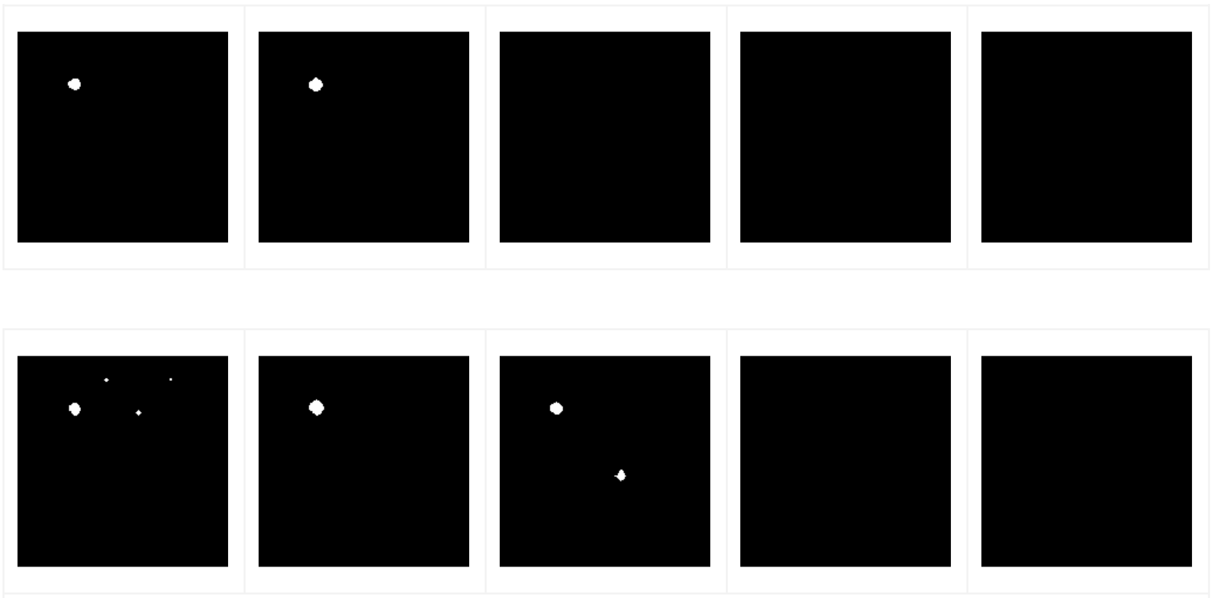


Fig.14) From left to right we have increasing value of σ from 10 to 70.
Top row shows the result of foveation-implementation, while the bottom one shows the NL-implementation
The white region corresponds to the detected anomalous region.

6. Conclusions

Both method we have presented achieve good performance, considering the average value 0.75574 and 0.76264 for NL-implementation and foveation-implementation respectively. In general, we achieve better result for lower σ, looking at the ROC curve (figure 6) and their AUC we note that we have the best results for σ = 10, while for higher value like 50 or 70 the detection is basically random ($AUC \leq 0.5$).

What we can see from the graphs in the figures 8 and 9 is that there is no a clear relation between the AUC and PSNR, in particular for the foveation implementation. However, if we look only at the mean relation, in figure 9, we can see somehow a relation, high AUC imply high PSNR and obviously in the relation we have to take into account also the noise. In fact, we obtain better results in case of low sigma.

In test3, the results are as expected, fine for lower value of sigma, in both the implementation the defect is detected, otherwise not. The results are consistent with the ROC. While for Test1 and Test2 the detection does not work. In test1, the problem is that the threshold is too low, almost all the pixels are detected as anomalous. While in Test2 the problem is different, not only anomalous pixels are detected, but also pixels corresponding to withe dots, here the problem may be the search window or the patch dimension.

In test1 as we have said the problem is that the threshold is too low. In fact, if we look at the raw data before the post-processing (result of equation 12) in figure 15, we can clearly see the anomaly even for σ equal to 70. Therefore, a possible improvement for the algorithm could be in the post-processing phase or in the threshold selection. For example, the threshold strongly depends on the sigma, but it may also exist a relation with the other parameters, which are the window and the patch dimension, so one can automatically select the right value. Those two parameter are also important because performance of the algorithm depends also on them. In [1] it has been shown that we get better performance if the anomaly cover most of the patch are. While increasing the search window up to a certain point might increase the performance, for example in figure 16, there are the result of increasing the search radius in test2, but we also increase the computational time since it depends on the image, patch and search window dimensions. Therefore, to improve the algorithm it is also possible to work on the complexity or implement a GPU version, which may decrease drastically the computational time.

The algorithm as implemented does not need image registration, meaning that we do not need to align the images, since we are exploiting NL-means. However, there could be problem if the images are rotated; in this case, we are not sure about the performance. In order to overcome this problem, it is possible to adopt rotational invariant solutions.

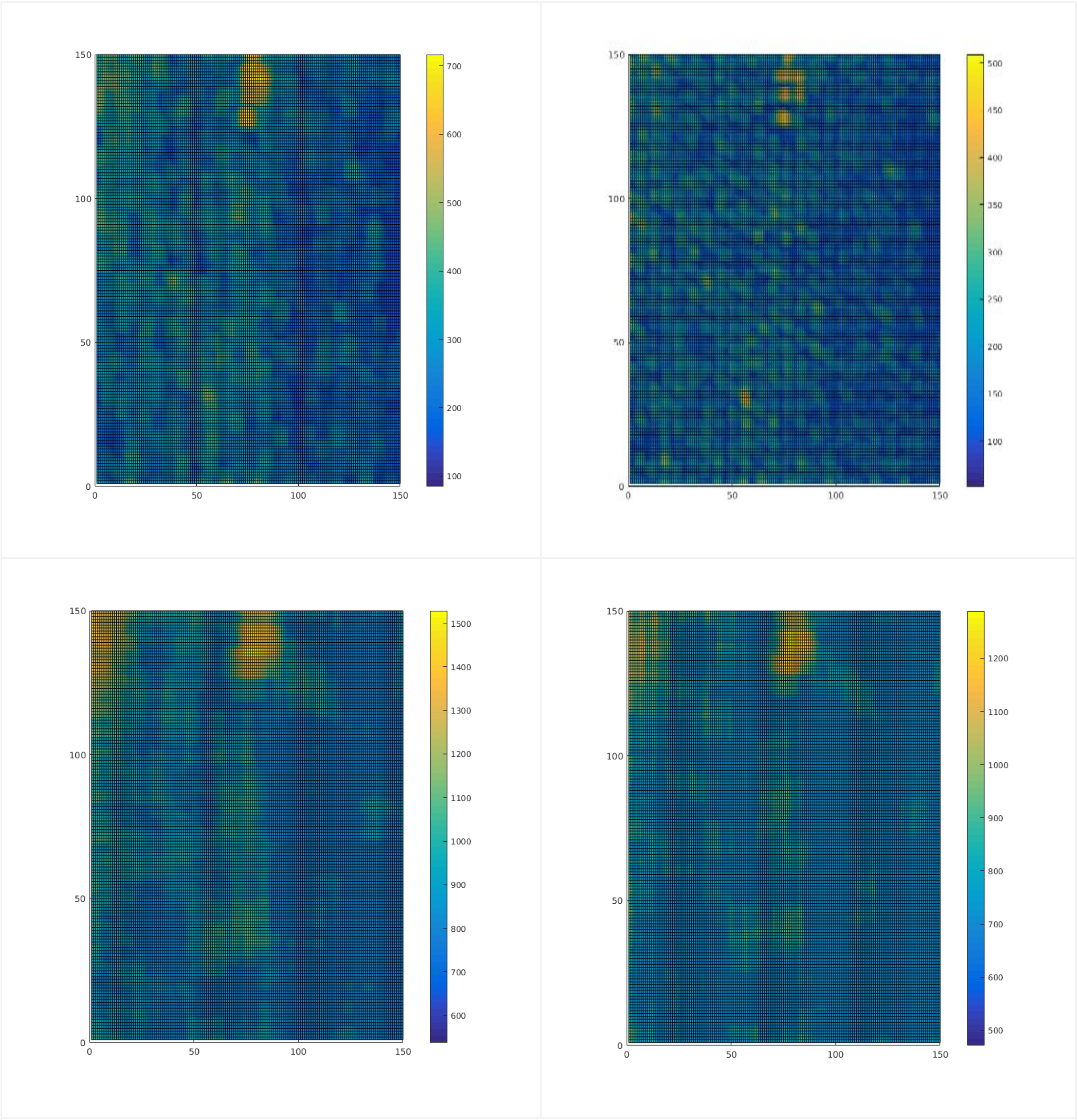


Fig.15) On the left Anomaly image for foveation, on the right NL for test1; top row σ equal to 10, bottom 70. The value on the bar represent the patch-wise difference.

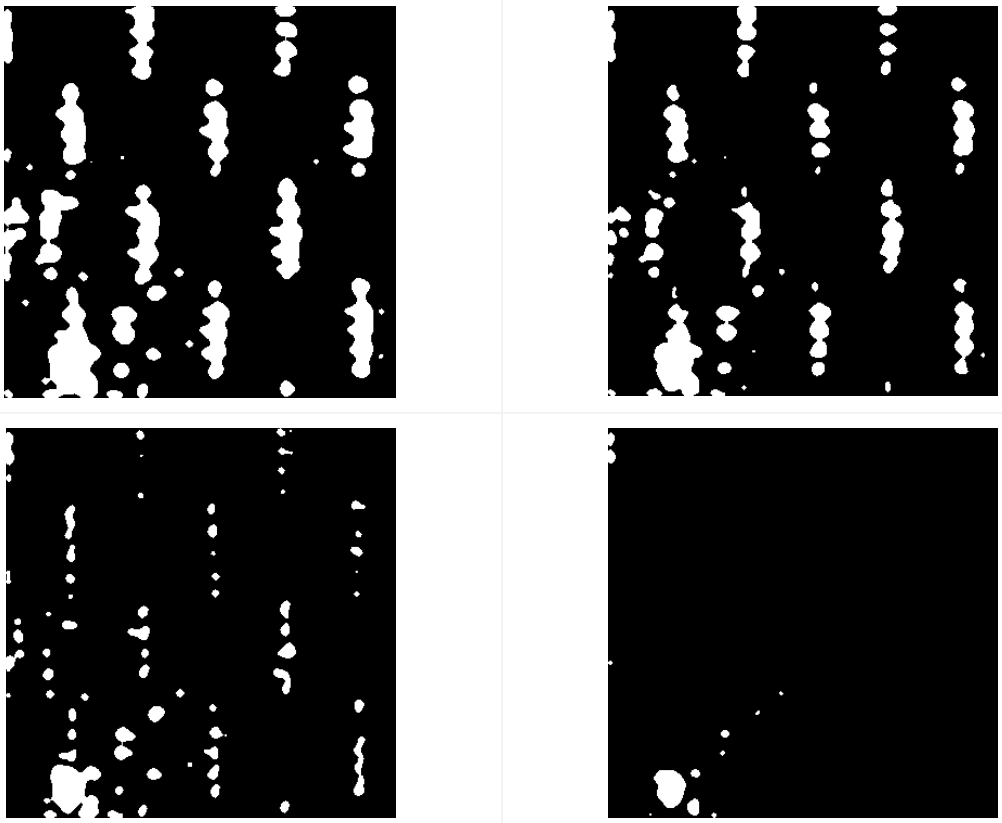


Fig.16) On the left Anomaly image for foveation, on the right NL; top row search window is 51x51, while bottom row is 101x101

7. Reference

[1]Maria Zontak · Israel Cohen: Defect detection in patterned wafers using anisotropic kernels.

[2] Alessandro Foi · Giacomo Boracchi: Foveated Nonlocal Self-Similarity.

[3]Antoni Buades · Bartomeu Coll · Jean-Michel Morel: A non-local algorithm for image de-noising.