Inspection detection

Amichay Feldman

May 24, 2020

This report is intended to decribe my code and the accompanying assumptions.

1 Assumptions

1. Input images are always grayscale.
2. The Inspections have significant visually difference and also have minimum area (was defined in config to be 40 pixels)
3. There is just single position of perfect alignment between reference and inspection images.
4. All hyperparameters are defined on *config.ini* file.

2 Code

Inspection detector was written as class format. Just *process* method is public in order to prevent the user from using spesific private method which depends on process order, So user should call just *process* method for run the detector.
Class instance gets two pathes of reference and inspected images as are defined in the config.
I implemented the code with classical image processing and computer vision algorithms and used OpenCV and Skimage libraries .

Note: I tuned the threshold according to the three given images. In order to produce more robust detector which will able to produce good results on generally cases, more data is needed.

Algorithm order:

1. **Align images:** Use Bilateral filter in order to blur all non-edges areas. Then compute images correlation with FFT method (which designed to calculate convolution result). Maximum response value of correlation result is the shift for the best alignment. Finally, inspected image is shifted and mask is done for the area in reference image which is parallel to zeros area of inspected that caused by the shifting.
2. **Compute differences mask:** Using Bilateral filter for noise blurring, XOR for computing pixel-weise differences, simple binarization on diff map to remain just "strong" differences (as I assumed, inspection has significant difference from parallel pixels in reference image) and morphologies (filter small noise differences by repeat erosion many times and reconstruct remain differences by dilation) in order to produce diff map.
3. **Compute similarities:** Use connected-components method in order to segment blobs of diff map. For each blob I check if blob area is larger than the minimum, compute histogram for tight bbox that surrounds the blob (I ignored all values of [0-10] because I don't want to take into account background pixels inside the bbox) and compare histograms by correlation score (I also tried many method like Kullback-Leibler, Bhattacharyya, SSIM, etc.). Finally, if blobs has small correlation, I remained the adjusted pixel in diff map as 1 (indicates inspection), otherwise change to 0.
4. **Reverse shift:** In order to produce indication map for inspected map, reverse shift with same values from step 1 is mandatory.
5. **Save diff map:**: Save diff map as an image.

3 Results

The results are attached bellow. Each example results in separate line. For each example I attached the origin Inspected image, the pixelwise map, the new masked reference and the shifted inspection image.

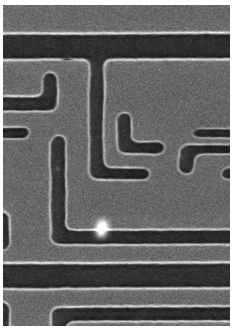

Figure 1: Inspected A

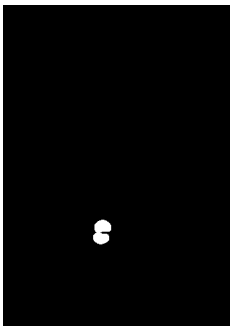

Figure 2: Indication map A

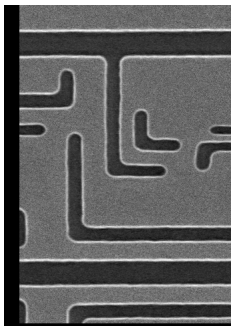

Figure 3: New reference A

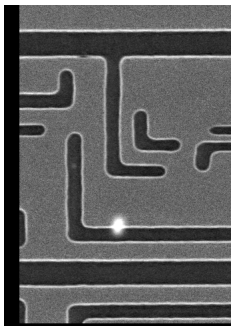

Figure 4: Shifted inspected A

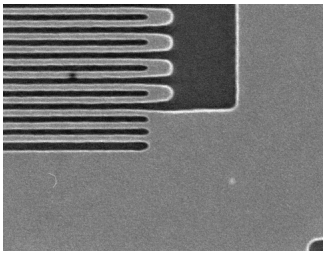

Figure 5: Inspected B

Figure 6: Indication map B

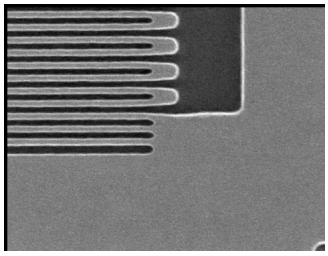

Figure 7: New reference B

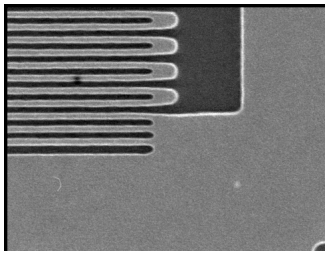

Figure 8: Shifted inspected B

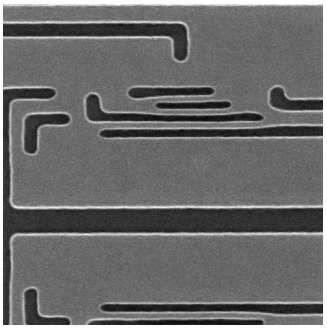

Figure 9: Inspected C

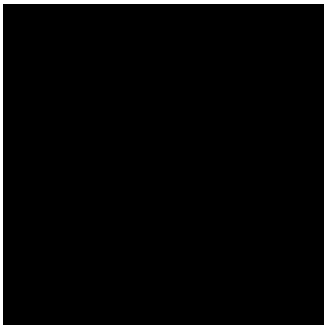

Figure 10: Indication map C

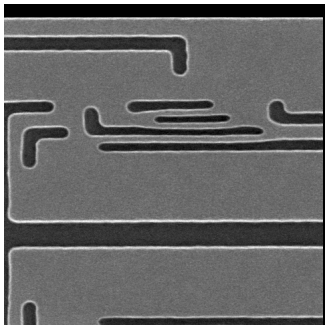

Figure 11: New reference C

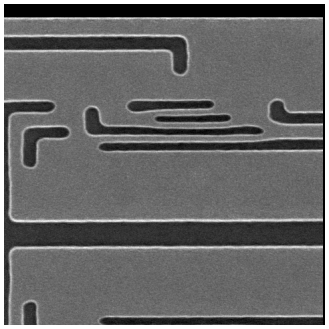

Figure 12: Shifted inspected C