Stanford University

Wafer Topographical Map Interpolation

Luis Castaneda
Stanford University SPCD
luismicas@stanford.edu

Abstract

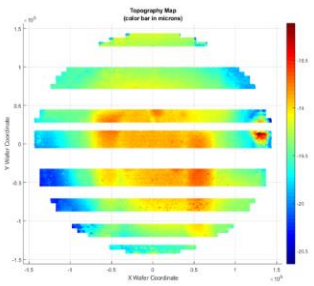
I propose a novel application of deep partial convolutional networks to take topographical maps with missing data areas and interpolate with meaningful context derived from surrounding data. I will be applying techniques from "Inpainting" algorithms, as well as enhancing the results with Single Image Super Resolution (SISR) algorithms. The goal of this project will be to show that physical data collection on system can be reduced, while maintaining a high fidelity to the ground truth mapping.

Introduction

In the world of data storage, 3D NAND is at the forefront of memory technology, offering higher bit densities, faster write speeds, and lower power consumption. However, this architecture brings with it a myriad of new challenges. To help manufacturers, meet yield targets, metrology equipment vendors need to be able to inspect wafers implementing 3D pattern processes, and feedback information on key defects of interest to manufacturers. The manufacturer can then use this information to correct their process and improve yields.

One of the biggest problems for equipment manufacturers has been keeping the top surface of the wafer in focus. Most wafer inspection systems rely on an optical autofocus subsystem to keep the wafer in focus for the system sensor. However, certain autofocus system (AF) architectures are susceptible to process variation which can cause a delta between the AF measured best focus, and the inspection systems best focus. This can lead to a loss of sensitivity, which reduces the effectiveness of the inspection system.

A method which has been devised to work around this issue is to create a topographical map prior to inspection of the wafer. However, this is a time-consuming process that can increase the total inspection time by up to a factor of two. Empirically, it has been found that not all the wafer needs to be mapped, and that interpolation of data points can help reduce the wafer mapping time.



(Figure 1. Sample Wafer Topography Map)

In this project, I will take existing full wafer maps, remove regions of data, and then interpolate the missing data to show that we can effectively reduce the data collection time, and rely on this interpolation method to improve customer throughput. Additionally, monitoring output layers of the network could provide additional process change information over time.

Related Work

In-painting has become a staple of image processing. This is a task for which CNN's are uniquely suited. There have been a number of

methods for in-painting typically applied to photographs, or other images. Some of the earlier works focused on fixed fill areas, such as “Context Encoders” [1] which would only work to fill a 64x64 pixel box in the center of the image. Other methods such as “Generative Inpainting” [2] could handle irregular holes of missing data, however this method typically requires a larger training set than what was available.

The architecture developed by Guilin L, et al. in Image Inpainting for Irregular Holes Using Partial Convolutions had the advantages that it could handle larger missing areas, and once trained should produce a filled image in a single pass. There are also of course non-learning methods such as patch matching which employ a statistical matching algorithm [6].

Challenges

One of the main challenges of this project will be the lack of varying data sets. Due to the manufacturer's tendency to protect their data, only a limited number of maps will be available for training and testing. The data set will come from a few dozen mappings, on a handful of different wafers types, collected over past weeks at several customer sites.

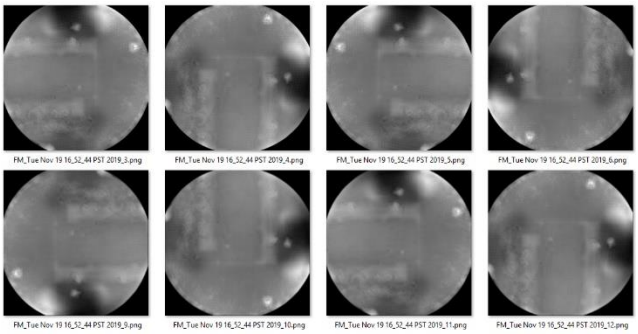
Additionally, the original data set is rather larger at about 12MB per map which would be difficult and slow to work with. However, there is a significant amount of superfluous information, and some redundancies which could be filtered out. The data sets will be reduced to (x, y, z) triplets. Converting these triplets into an image of 512x512, maintains a minimum amount of fidelity needed for the final application. Given a standard 300mm radius wafer each pixel will effectively map a 586umx586um area of the map. This would be enough for a typical application, but by upscaling post the interpolation step, we can ensure we can at least meet the Nyquist sampling criteria.

The data sets will also have to be preprocessed with various steps including: fitting to a standard mesh grid, normalizing, and discretizing. A discretization of the topography into 0-255 step will maintain enough resolution given the typical range of focus deviation seen in typical maps. (typical topography range is less than 6um, resulting in a resolution of ~23nm which is sufficient for the final application of the map.

Data Augmentation

Given the symmetrical nature of the wafer maps, several data augmentation techniques could be leveraged, including rotation, mirroring, and blurring. Specifically, the original images were rotated by 90^0, 180^0, and 270^0. These augmentation techniques are valid use cases as customer may choose to inspect the wafers in any of those orientations. Additionally, die typically have rectangular structures, and these rotations maintain the context of that pattern.

Blurring of the images is analogous to a lack of sensitivity in the mapping, which is also a reasonable approximation of issues that can arise from real-time mapping on mechanical systems. In this case the “blurring” would be analogous to a delay in z-stage response, as the z-coordinate information is being directly converted to pixel intensity. (Blurring was produced by convolving the images with a 4x4 filter)

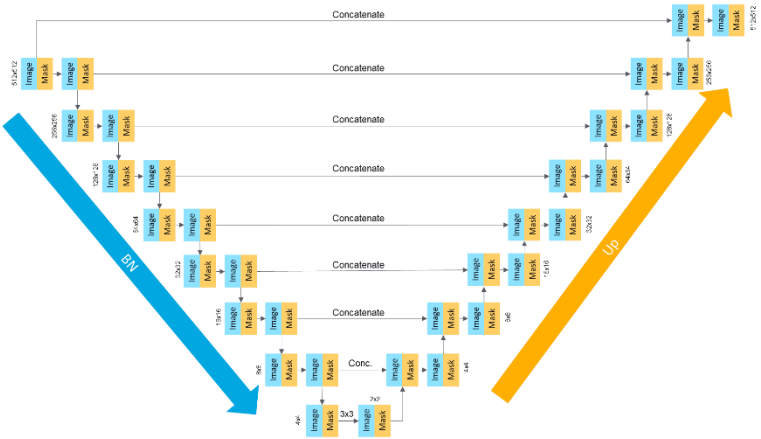


(Figure 2: Enhanced Data Set Examples)

Methods 1 - Interpolation Through Partial Convolutions

For this project I have elected to apply a technique developed by Guilin Liu, et. Al. from the NVIDIA corporation [4]. This is a modification of the standard UNet architecture but using partial convolutions and a mask updating step.

Total params: 32,876,128
Trainable params: 32,865,248
Non-trainable params: 10,880



(Figure 3: Architecture of Partial Conv)

The number of layers were not modified since the data was pre-processed in such a way that it was compatible with the existing architecture. However, the learning rate of 0.0002 was increased to 0.0005 given the fact that the areas were relatively larger, but the context was simpler to encode given the lower spatial frequencies of the maps. The loss function, which is a combination of 6 sub-loss functions, was also modified to more heavily weight the perceptual context given the relatively large mask patches. (See appendix for individual loss functions)

$$L_{total} = L_{valid} + 6\,L_{hole} + 0.05\,L_{percept} + 120\left(L_{style_{out}} + L_{style_{comp}}\right) + 0.1\,L_{tv}$$

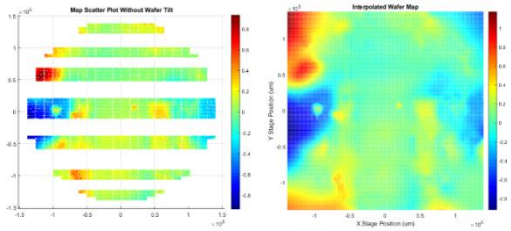
This architecture will recurrently update a mask as well as the image through each layer, reducing the size of the mask until its removed. Per [5], the imaged is updated by the following convolutional filter:

$$x' = \begin{cases} \boldsymbol{W}^T(\boldsymbol{X}\odot\boldsymbol{M})\dfrac{sum(\mathbf{1})}{sum(\boldsymbol{M})} + b, if\ sum(\boldsymbol{M}) > 0 \\ 0, \qquad\qquad Otherwise \end{cases}$$

Where ⊙ is an element-wise multiplication, and 1 has shape M, but all elements = 1. Additionally, the mask is updated by the following expression:

$$m' = \begin{cases} 1, & if\ sum(\boldsymbol{M}) > 0 \\ 0, & otherwise \end{cases}$$

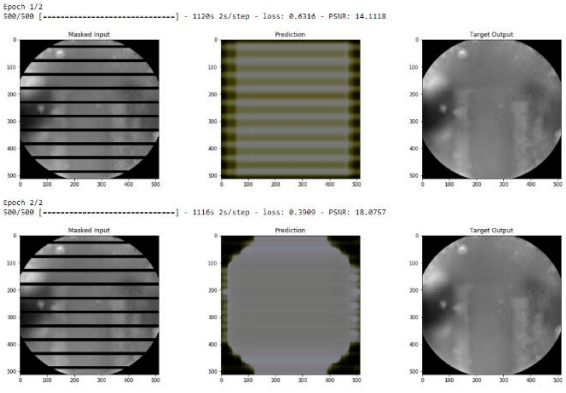
To apply this network, I first converted the maps to images, and resized them to work with the existing network. The original data consisted of column vectors with x stage position, y stage position, and z height data. Because of the way the data was collected, there were areas of high density, and low density. I used a grid fit gradient method to fit the available data and interpolate between the missing data. This produced a uniform set of triplets.



(Figure 4: Raw Data and Pre-process Interpolation)

I then discretized the z height data into pixel intensity values ranging from 0-255. The data from this channel was then copied twice to create a 3-channel image with an 8-bit pixel depth. The network, which is based off the VGG-16 CNN, was originally trained on a subset of the ImageNet library. I then trained again with my 360 enhanced data set, with a batch size of 4. Leaving 40 images for validation, and 40 images for test. I then generated a mask to identify the areas which should be interpolated. Given the small data set, additional training repeats were performed and there was likely some overfitting of the data. This could be mitigated with a larger data set if this method is rolled out.

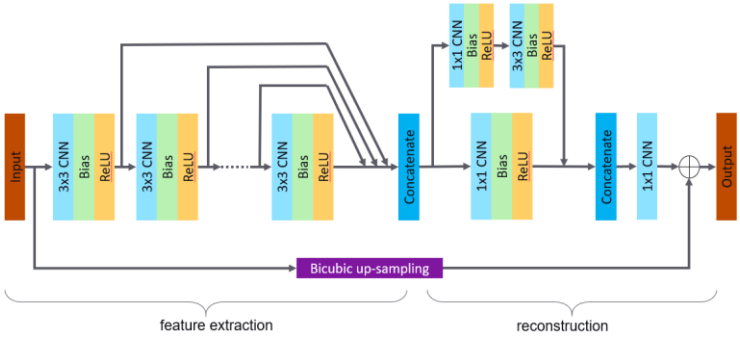
The CNN then took the context from the unmasked areas to fill into the mask areas per the previous description. A more detailed post processing was done on a handful of generated images, with comparisons of the interpolated map areas to the ground truth raw map data.



(Figure 5: Mask, Prediction, Target)

Methods 2 – Post Processing Image Upscaling

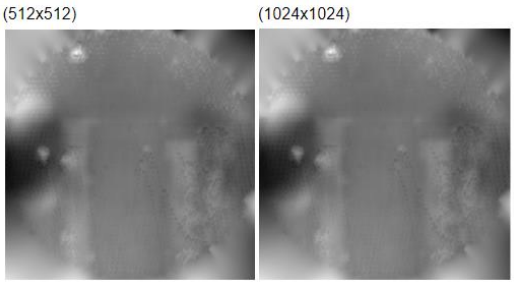
As a post processing step, I implemented variation of the CNN model described in the paper "Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network" by Jin Yamanaka, et. Al. With an addition pre-processing step in before running the image through the network.



(Figure 6: Upscaling Network)

The main goal of this step was to attempt to increase the contrast of the filled in context, as well as to upscale the image to provide at least a 2x resolution over the standard AF sampling rate. This architecture is a bit dated, and simple, but more than enough in extracting the feature set's an providing an upscaled image with improved contrast.

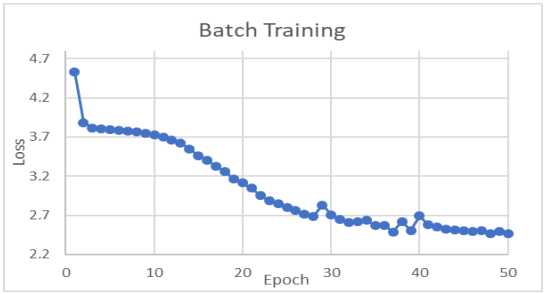
The loss function is a simple mean squared error, with the addition of the L2 norms for regularization. No change was made to the learning rate of 0.002. With this increase image size, the resultant pixel size translated to 293um at the for a 300mm wafer, which is 2x smaller than the AF spot size. When translated back this should provide more than enough resolution for the system to track without the pixilation being a limiting factor.



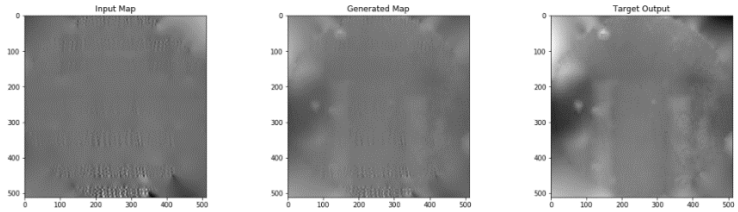
(Figure 7: Upscaling Results)

Results and Validation

During testing, there were issues with mask pattern feeding back into the image, although the loss was still relatively low. This appears to be a limitation with this architecture. However, that was mitigated through extrapolating the images, and apply a pre-interpolation step.



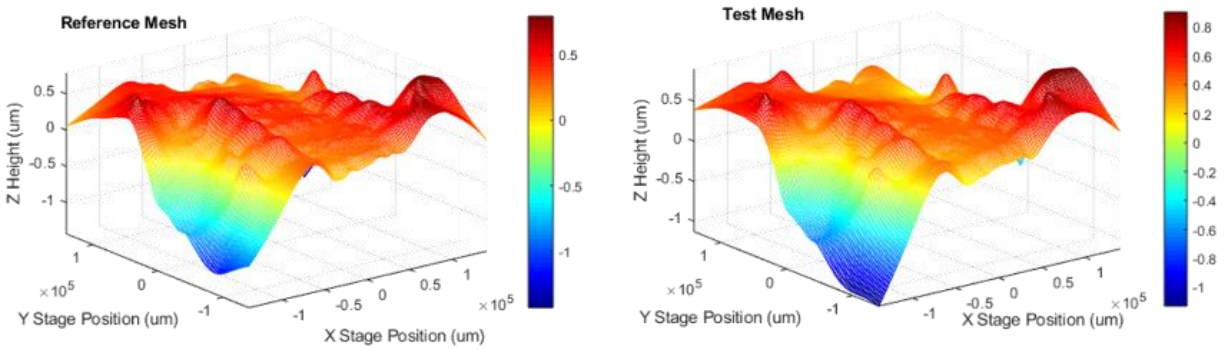
(Figure 8: Batch Training Results)



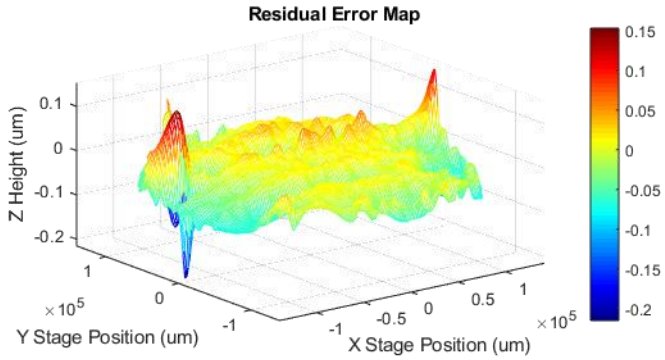
(Figure 9: Final Interpolation Results)

After interpolating the images, they can then be compared to the ground truth images to calculate the residual errors. For this technique to be usable, the errors must be less than the depth of focus of the inspection system. That is the difference between the test and ground truth must be less than 90nm.

Here I show the ground truth image along side with the interpolated image, as well as the resulting residual map (test – truth). The results fall within the acceptable limits for implementation of this interpolation method.



(Figure 10: Interpolated vs Ground Truth)



(Figure 11: Residual Error Plot)

Conclusion and Future Work

Although the results of this project were successful in producing usable maps, the time cost to process the maps does not make the current implementation of this method viable. To make this worthy of in-line testing the time for processing would need to be reduced to less than 1.5min.

However, the images themselves can be a method of storing the data in a more compact fashion, and the training weights could be used as a secondary monitoring metric of the wafer process variation. The introduction of this type of analysis could be the starting point for optimizing this interpolation task, as we'll as providing more insight into process variation which could provide additional value to our customers.

There are several optimizations that remain to be implemented. For example, the UNet used for inpainting can be reduced significantly given that only the original data consists of 1 channel and that the context is simple and periodic. Additionally, the upscaling could be integrated into the same network. These modifications could help to close the gap in implementing this process, and I hope to continue this work over the coming months.

Appendix

Complete Loss Function (See pg. 6, 7 of [3] for details)

$$L_{hole} = \frac{1}{N_{Igt}} \left\| (1-M) \odot (I_{out} - I_{gt}) \right\|_1 \quad , \quad L_{valid} = \frac{1}{N_{Igt}} \left\| (M) \odot (I_{out} - I_{gt}) \right\|_1$$

$$\mathcal{L}_{perceptual} = \sum_{p=0}^{P-1} \frac{\|\Psi_p^{\mathbf{I}_{out}} - \Psi_p^{\mathbf{I}_{gt}}\|_1}{N_{\Psi_p^{\mathbf{I}_{gt}}}} + \sum_{p=0}^{P-1} \frac{\|\Psi_p^{\mathbf{I}_{comp}} - \Psi_p^{\mathbf{I}_{gt}}\|_1}{N_{\Psi_p^{\mathbf{I}_{gt}}}}$$

$$\mathcal{L}_{style_{out}} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \left|\left| K_p \left(\left(\Psi_p^{\mathbf{I}_{out}}\right)^{\intercal} \left(\Psi_p^{\mathbf{I}_{out}}\right) - \left(\Psi_p^{\mathbf{I}_{gt}}\right)^{\intercal} \left(\Psi_p^{\mathbf{I}_{gt}}\right) \right) \right|\right|_1$$

$$\mathcal{L}_{style_{comp}} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \left|\left| K_p \left(\left(\Psi_p^{\mathbf{I}_{comp}}\right)^{\intercal} \left(\Psi_p^{\mathbf{I}_{comp}}\right) - \left(\Psi_p^{\mathbf{I}_{gt}}\right)^{\intercal} \left(\Psi_p^{\mathbf{I}_{gt}}\right) \right) \right|\right|_1$$

$$\mathcal{L}_{tv} = \sum_{(i,j)\in R,(i,j+1)\in R} \frac{\|\mathbf{I}_{comp}^{i,j+1} - \mathbf{I}_{comp}^{i,j}\|_1}{N_{\mathbf{I}_{comp}}} + \sum_{(i,j)\in R,(i+1,j)\in R} \frac{\|\mathbf{I}_{comp}^{i+1,j} - \mathbf{I}_{comp}^{i,j}\|_1}{N_{\mathbf{I}_{comp}}}$$

where, $N_{\mathbf{I}_{comp}}$ is the number of elements in $\mathbf{I}_{comp}$.

The total loss $\mathcal{L}_{total}$ is the combination of all the above loss functions.

$$\mathcal{L}_{total} = \mathcal{L}_{valid} + 6\mathcal{L}_{hole} + 0.05\mathcal{L}_{perceptual} + 120(\mathcal{L}_{style_{out}} + \mathcal{L}_{style_{comp}}) + 0.1\mathcal{L}_{tv}$$

Contributions and Acknowledgements

I'd like to thank the TA's of CS230 for their assistance and guidance, as well as the Coursera deeplearning.ai team. As well as special thanks to Sarah C. for her personal guidance throughout the project. Much of the coding was modified from Mathias Gruber's repository, which had an excellent step by step outline of the process with easy to read code implementation.

References

[1] Pathak, D. et. al. "Context encoders: Feature learning by inpainting", https://arxiv.org/pdf/1604.07379.pdf

[2] Raymond A. Yeh, et al. "Semantic Image Inpainting with Deep Generative Models", https://arxiv.org/pdf/1607.07539.pdf

[3] Guilin Liu, et. al. "Image Inpainting for Irregular Holes Using Partial Convolutions", https://arxiv.org/pdf/1804.07723.pdf

[4] Mathias Gruber "Partial Convolutions for Image Inpainting using Keras", https://github.com/MathiasGruber/PConv-Keras

[5] Jin Yamanaka, et. al. "Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network", https://arxiv.org/ftp/arxiv/papers/1707/1707.05425.pdf

[6] Connelly Barnes, et.al. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing", https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/patchmatch.pdf

[7] Github depository: https://github.com/luismicas/CS230Project