Measuring corrosion levels of ductile iron pipes using scanned images

Mak, H.; Wong, B.; Kruithof, S.; Krys, D.; Liu, Z.

Publisher's version / Version de l'éditeur:
https://doi.org/10.4224/20374408
Research Report (National Research Council of Canada. Institute for Research in Construction), 2008-11-01

NRC·CNRC Institute for Research in Construction
CNRC·NRC Institut de recherche en construction

http://irc.nrc-cnrc.gc.ca

Measuring corrosion levels of ductile iron pipes using scanned images

RR-268

Mak, H.; Wong, B.; Kruithof, S.; Krys, D.; Liu, Z.

November 2008

National Research Council Canada
Conseil national de recherches Canada
Canada

Executive Summary

This report documents the findings and the results of an engineering exercise to develop a process to measure the corrosion depth of buried samples from their scanned images before and after burial. The recommended method consists of 5 steps which combine the power of a Konica Minolta scanner, PolyWorks and software developed in Matlab. PolyWorks is mainly used as a human interface for manipulating and aligning point cloud data sets. The use of PolyWorks reduces software development to a minimum. Software developed in Matlab is mainly for image transformation and corrosion depth/volume calculation.

Since the corrosion measurements are planned to span from 5 to 20 years, the proposed process is developed with long term viability in mind. Both the Minolta scanner and PolyWorks can be replaced with other systems providing similar functionalities. The interface files are ASCII based which is universally acceptable. All theories, algorithms and engineering principals involved are documented in sufficient details in the Appendix sections so that the software code can be re-implemented in any form possible if Matlab becomes obsolete.

The practicality of this process has been verified using three simulated laser-scanned images each of two samples at corrosion depths of 0.08 mm, 4 mm and 4 mm with +/-5% noise added. In this way, the recommended method and software developed in Matlab is also verified.

The calculation of the corrosion volume is one of the goals of this exercise. It is found that the corrosion volume derived from a single image scanning method is not a comparable quantity between any two corrosion measurements. This is because the scanning angle is uncontrollable when samples are scanned years apart. This varies the visibility of the boundaries of the sunken corrosion surface at each scan. Instead, it is proposed to use the mean corrosion depth, defined as the total corrosion volume per unit corrosion area, for this purpose. Multiple scanning of the sample will correct this problem, but it will involve more complicated image stitching operations in PolyWorks. This is left as a future improvement of the current method.

Logically, the accuracy of this method depends heavily on the surface treatment of the corroded surface. Surface dirt and loose particles have to be removed in a way that does not damage the actual underlying metal surface. Surface rusts that cannot be removed will introduce errors to this method.

In conclusion, the recommended method has measured the simulated corrosion depths with accuracies over 95%. It is an acceptable process.

Table of Contents

1 Background

1.1 The Project

The Buried Utilities Group, Urban Infrastructure Program, is conducting a project to assess the degree of corrosion of buried ductile iron pipes. The project uses several sensors, CORROSOMETER® Corrosion Probes, which are designed to estimate the corrosion level of a ductile iron pipe. The sensors contain a sample of ductile iron, which is exposed to the same elements as the target ductile iron pipe. The corrosion on this exposed area will reflect the degree of corrosion on the buried pipes. These sensors are designed to be read by a corrosometer, CORROSOMETER® Portable Instrument, which measures electrical resistance across the exposed ductile iron sample. As the sample corrodes there is less material so the resistance of the sample increases. The problem is that there seems to be little experimental data showing the effectiveness and characteristics of these specific sensors. So a series of tests have been set up to evaluate the characteristics of the sensors.

Initial tests include leaving these sensors in controlled environments beside ductile iron pipes. The problem with this is that the samples will take 5 to 20 years to corrode. So another set of tests has been developed that will use electrolysis to increase the corrosion rate of the samples. This will allow the samples to corrode in months rather than years.

To confirm that the electrolysis is a controlled environment and the results can be reproduced, a set of our own samples was manufactured that can be used to test the environment. These samples are just sections of ductile iron pipe that have an exposed surface that is the same surface area as the corrosion sensors. Once the results show that the electrolysis can be performed as to yield the same results each time, then the experiment will be used on the actual corrosion sensors.

In order to evaluate the amount of corrosion, a set of different test methods are needed. For the electrolysis test, three different techniques are used to gather data on the amount of metal loss. The first method is to compare the weight of the sample before and after the experiment. The difference in weight will reflect the total metal loss. The second technique will be to measure the number of electrons that were removed from the sample using electrolysis and then calculate that total weight loss. The problem with these techniques is that they will tell us how much metal loss there was, but there is no information to say where it was lost. This third technique is to use laser-scanned images to measure the corrosion rate. Dennis Krys has also developed a basic program in Matlab to measure the corrosion depth of any two laser-scanned images. This technique will indicate where the metal was lost as well as how much was lost.

1.2 Role for Center for Computer-assisted Construction Technologies (CCCT)

In this project, CCCT provides a supporting role to the Buried Utilities Group. The role of CCCT is to recommend an engineering process to provide a rough measurement on the corrosion level of the samples using laser-scanned images. The process has to use existing tools at CCCT. The corrosion level is described as the corrosion depths for the exposed surface of a sample. The corrosion depth across the complete area of the exposed surface, the distribution and the total corrosion volume must be calculated. CCCT has been provided with two samples (Sample A & Sample B as shown in Appendix C.1) to perform the study.

Based on this approach, CCCT has:

- Recommended a way to improve the accuracy of image alignment of the scanned images by incorporating abrupt changes to the otherwise gentle geometric shape of the samples. This is achieved by using a few pieces of Lego blocks to be installed in the surrounding area of the exposed surface (See Appendix C.1).
- Performed the scanning of the samples using the Konica Minolta 3D Digitizer VIVID 910.
- Developed a method to combine the image alignment capability of PolyWorks and image processing power in Matlab to derive the delta of any two scanned images.
- Verified the method by using three simulated corrosion scanned images of each sample with corrosion depths of 0.08 mm, 4 mm and 4 mm with noise.

The goal of this document is to record the methods, the results and the engineering principles / theories involved. As the methodology proposed in this project has a potential time span greater than 10 years, it is necessary to document all relevant information in very simple terms so that they can be referred to in future. The image files, due to their size, are not included in this document. They, together with the Matlab code, are stored electronically in the shared drives at the NRC London site. The files are backed-up periodically; their archives are kept for a long time and are available on request.

1.3 The Technology & Challenges

The challenge of this project is to find the offset distances (profile) of the corroded surface from its original exposed position. However, one cannot find this distance by simply subtracting the points in the scanned images of the corroded object and the original object, because each image sits in its own 3D space. The origin, the viewing angle (scan angle) and coverage area of any two scanned images may be different because of the individual setup.

The general procedure to calculate the offset distance(s) of two scanned images is as follows:

1. Align the two geometric shapes represented by their point cloud data sets by fitting local surfaces to them.
2. Lay a common grid to the corresponding surfaces and find out all the grid points on these surfaces.
3. Calculate the z-distance between corresponding grid points of the two sets of surfaces.
4. Analyze the distribution of the offset distances of the exposed area before and after corrosion which will reflect the nature of the corrosion.

The alignment (or some times called registration) of two cloud point data sets in Step 1 is still a hot research topic. Typically, the process is divided into two stages:

1. Crude pre-alignment: The intention of this operation is to roughly put the two images in the same orientation by translation and rotation. Many researchers are still working on this area to provide an automatic way to pre-align two point cloud data sets. Commercial tools, e.g. PolyWorks, usually use a semi-manual method, which require the operator to pick a set of identical points (landmarks) from the two images and then the software will perform the rough alignment[1].
2. Fine alignment: The goal of this operation is to move the two point clouds relative to each other so that the sum of the square distances between the two images is minimum (least square method). The golden standard process used in the field is the Iterative Closest Point (ICP) algorithm which is also implemented in PolyWorks.

Since the ICP will attempt to average the distance of the two images, the corroded surface will introduce errors in the fine alignment stage. Hence, the corroded surface has to be excluded in the image alignment step. Once the alignment is achieved, one needs to find out the transformation matrix that the alignment algorithm has used and re-apply it to the points of the corroded area.

The scanning operation of the original surfaces of the samples is very important in this method, because it produces the images to which all subsequent images of corroded samples have to be compared. As the method to calculate the corrosion depth developed by Dennis is based on the delta of the z-values of the grid points, the scanning has to be performed as close to the surface normal of the samples as possible. Since it is very difficult to be exact, the orientation of the scanned images needs to be determined and rotated accordingly, using image processing techniques so that their surface normals align with the z-axis before the corrosion depth can be calculated.

One intention of this project is to calculate the corrosion volume of the sample after a period of burial. To do this, one has to find the actual boundary of the corroded surface and the corrosion depth at each point. Unfortunately this may not be possible, especially for sunken surfaces, because the scanning angle is not controllable in our case. Some part of the actual boundary of the corroded surface may not be visible in the scan.

To overcome this challenge, it is proposed to use the mean corrosion depth of the corroded surface instead. The mean corrosion depth is the total corrosion volume per unit corrosion area. In this way, it is independent of the corroded surface boundary. To calculate the corrosion volume, it is proposed to pre-mark the exposed surface of the original sample using PolyWorks (Step 2)b), then use the boundary of this surface (a quadrilateral) to automatically determine the corrosion surface of the corroded sample.

The theory for calculating the image delta of two point cloud data sets is documented in Appendix A.1. The image rotation operation and its related theory are documented in

---

[1] Due to the fine spacing of the scan points, picking a landmark from the screen usually identifies an area with a small set of points and not a single point. Hence this can only be a rough alignment process.

Appendix A.3 and Appendix A.4. The algorithm to determine the boundary of the exposed area used in mean corrosion depth calculation is documented in Appendix A.5 and the corrosion volume calculation in Appendix A.2.

2 The Proposed Process

Figure 1 and Figure 2 show the complete operation flow of this process.

For the sample before corrosion (refer to Figure 1):

1) Scan the original sample using the Konica Minolta 3D Digitizer VIVID 910. The scanning should be done at the minimum distance (or maximum resolution) of the scanner. The Digitizer's line of sight (z-axis) should be aligned closely with the sample surface normal. This generates the image file IMO.
2) Load the image IMO into PolyWorks:
    a. Cut off the points in the exposed area of the sample and save it as image file IMO_hole.
    b. Cut off all the points of the IMO image leaving only the exposed area and save it as image file IMO_exposed.

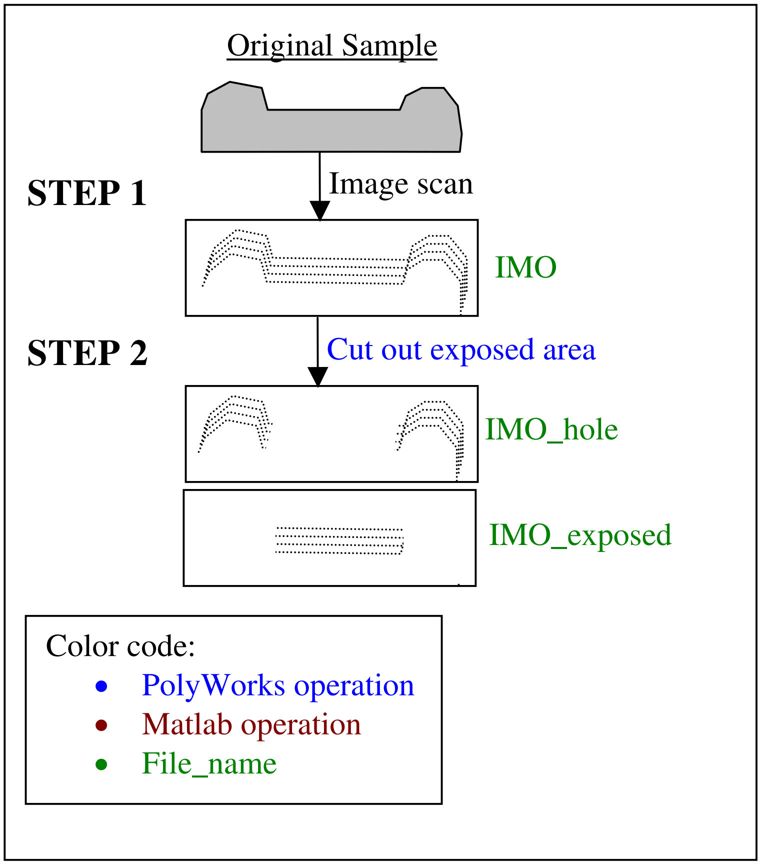


Figure 1: The proposed process for sample before corrosion

For the sample after corrosion (refer to Figure 2):

1) Scan the corroded sample using the Konica Minolta 3-D Digitizer VIVID 910. The scanning should be done at the minimum distance (or maximum resolution)

of the scanner. The Digitizer's line of sight (z-axis) should be aligned closely with the sample surface normal. This generates the image file IME.

2) Load the image IME into PolyWorks, cut off the points in the exposed area of the sample and save it as image file IME_hole.
3) Load both IMO_hole[2] and IME_hole into PolyWorks and use the point cloud alignment tool to align IME_hole to IMO_hole. Save the aligned IME_hole to file IME_hole_aligned.
4) Use Matlab to calculate the corrosion depth
    a. Edit the front part of main6 in Matlab to provide
        i. the input file directory and file names for IMO_exposed, IME_hole, IME_hole_aligned and IME, and
        ii. the output file directory and file name for IME_aligned, IMO_exposed_rotated1 and IME_exposed_rotated.
    b. Run main6 in Matlab
    c. Log down the analysis results displayed on the screen and the figure displaying the results of corrosion depth histogram and distribution.
5) Use PolyWorks to display the delta of IMO and IME_aligned and to inspect IMO_exposed_rotated1 and IME_exposed_rotated to check if the area used for corrosion calculation is properly set.

---

[2] Another approach is to load IMO in instead and cut out the exposed part to re-generate IMO_hole again.

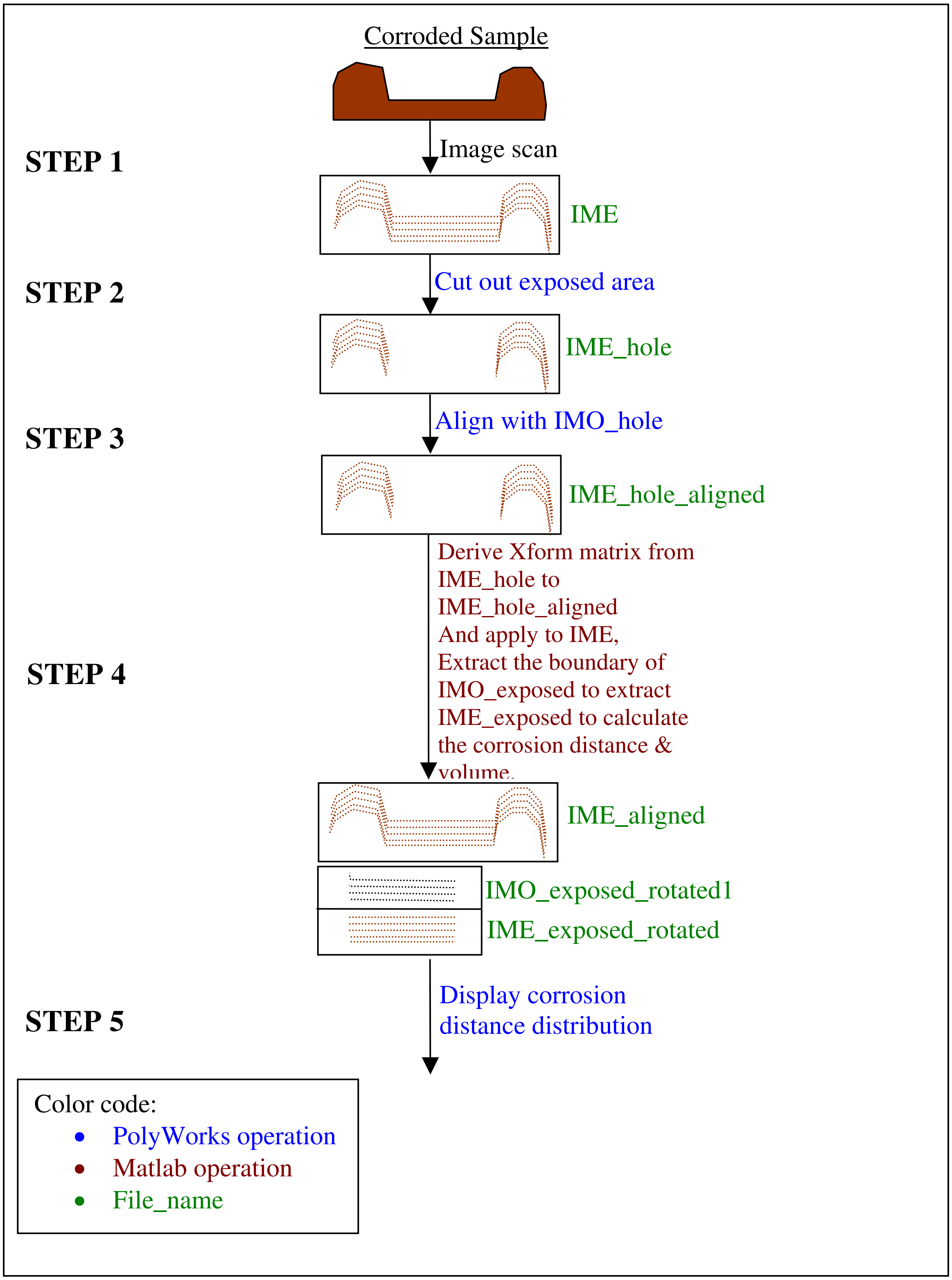


Figure 2 : The proposed process for sample after corrosion

3 Test Results

In order to test the process and all software tools developed, three simulated corrosion scanned images are created mathematically[3] to simulate the corrosion depths of 0.08 mm, 4 mm and 4 mm with 5% noise added respectively to the exposed area of Sample A and Sample B. This section documents the test results of the process using these simulation files.

3.1 Sample A

The data measured from Sample A:

- The surface normal of the exposed surface is found to be at 2.43^o to the z-axis. After re-orientation, the angle becomes 0.002^o.
- The vertices of the inner cropping polygon used in all corrosion calculations for Sample A are (-3.7963, 29.7675), (43.2575, 26.9300), (42.2943, 17.9787) and (-3.8680, 19.3511).

3.1.1 0.08 mm Simulated Corrosion Depth

The screen captures of the results for processing the 0.08 mm simulated scanned image are included in Appendix C.3 section C.3.1.

Here are the results obtained from the process:

- The min, median, max, and std of the corrosion depth are -0.014 mm, 0.083 mm, 0.163 mm and 0.088, respectively.
- The total corrosion volume measured within the cropping polygon is 37.06 mm^3.
- The total corrosion area within the cropping polygon is 449.27 mm^2.
- The mean corrosion depth = 0.082 mm.
- Method accuracy = $\left(1-\frac{0.082-0.08}{0.08}\right)\times 100\% = 97.5\%$

3.1.2 4 mm Simulated Corrosion Depth

The screen captures of the results for processing the 4 mm simulated scanned image are included in Appendix C.3 section C.3.1.

Here are the results obtained from the process:

- The min, median, max, and std of the corrosion depth are 3.799 mm, 4.032 mm, 4.269 mm and 0.052, respectively.
- The total corrosion volume measured within the cropping polygon is 1811.92 mm^3.
- The total corrosion area within the cropping polygon is 449.31 mm^2.
- The mean corrosion depth = 4.03 mm.

---

[3] The scanned image of Sample A is taken and all the points in the exposed area are isolated using PolyWorks. Matlab is used to (1) find the angle of the surface normal to the z-axis, (2) rotate the exposed area accordingly, (3) add 0.08 mm, 4 mm and 4 mm + 5% random noise, respectively to the z values of these points and then (4) rotate the exposed area back to generate the simulation scan files.

- Method accuracy = 99.25%

3.1.3 4 mm Simulated Corrosion Depth with +/- 5% Noise

The screen captures of the results for processing the 4 mm simulated scanned image are included in Appendix C.3 section C.3.1.

Here are the results obtained from the process:

- The min, median, max, and std of the corrosion depth are 3.681 mm, 4.050 mm, 4.415 mm and 0.105, respectively.
- The total corrosion volume measured within the cropping polygon is 1824.12 mm^3.
- The total corrosion area within the cropping polygon is 450.43 mm^2.
- The mean corrosion depth = 4.05 mm.
- Method accuracy = 98.75%

3.2 Sample B

The data measured from Sample B:

- The surface normal of the exposed surface is found to be at 6.64^o to the z-axis. After re-orientation, the angle becomes 0.004^o.
- The vertices of the cropping inner polygon used in all corrosion calculations for Sample B are (-14.2617, 80.5095), (35.0287, 78.0985), (35.3734, 68.0502) and (-14.3429, 70.2161).

3.2.1 0.08 mm Simulated Corrosion Depth

The screen captures of the results for processing the 0.08 mm simulated scanned image are included in Appendix C.3 section C.3.2.

Here are the results obtained from the process:

- The min, median, max, and std of the corrosion depth are -0.219 mm, 0.083 mm, 0.690 mm and 0.032, respectively.
- The total corrosion volume measured within the cropping polygon is 41.83 mm^3.
- The total corrosion area within the cropping polygon is 500.34 mm^2.
- The mean corrosion depth = 0.084 mm.
- Method accuracy = 95%

3.2.2 4 mm Simulated Corrosion Depth

The screen captures of the results for processing the 4 mm simulated scanned image are included in Appendix C.3 section C.3.2.

Here are the results obtained from the process:

- The min, median, max, and std of the corrosion depth are 2.241 mm, 3.997 mm, 4.689 mm and 0.074, respectively.
- The total corrosion volume measured within the cropping polygon is 200.17 mm^3.

- The total corrosion area within the cropping polygon is 500.34 mm^2.
- The mean corrosion depth = 3.998 mm.
- Method accuracy = 99.95%

3.2.3 4 mm Simulated Corrosion Depth with +/- 5% Noise

The screen captures of the results for processing the 4 mm simulated scanned image are included in Appendix C.3 section C.3.2.

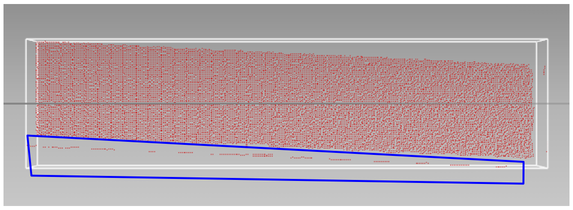
Here are the results obtained from the process:

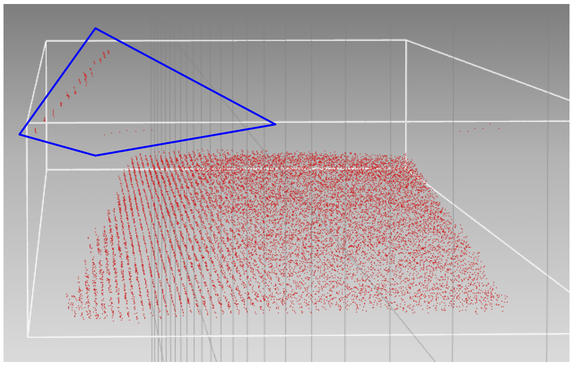
- The min, median, max, and std of the corrosion depth are 1.336 mm, 4.094 mm, 4.875 mm and 0.263, respectively.
- The total corrosion volume measured within the cropping polygon is 2034.11 mm^3.
- The total corrosion area within the cropping polygon is 500.34 mm^2.
- The mean corrosion depth = 4.065 mm.
- Method accuracy = 98.38%

4 Discussion

From the test results, it is found that:

1. The accuracy of this method depends heavily on the surface treatment of the buried sample prior to the measurement. One has to remove all dirt and loose particles (e.g. rust) from the metal surface without damaging the integrity of the underlying metal surface. When metal rusts, it expands in volume. The rust on the corroded surface will introduce errors in the mean corrosion depth calculated.
2. The corrosion volume derived from a single image scanning is not a comparable quantity between any two corrosion measurements. Instead the mean corrosion depth is a better means for this purpose. This is because the scanning angle of a corroded sample, performed in subsequent scans years apart, is not controllable. The resulting point cloud image of the corrosion area will differ between scans, especially for sunken surfaces, where there may be areas that are visible in one scan and not in another. Even if one applies image processing techniques to align the corroded surface with the original one and crop both with the same outline, the resulting corroded surface may still differ between scans. Since the corrosion volumes are calculated based on these corrosion areas, the corrosion volume thus calculated cannot compare one from the other. The mean corrosion depth, however, being the ratio of the corrosion volume per unit corrosion area, is independent of the corrosion area from one scan to another. It is a better measure to compare degree of corrosion with. A multiple scan approach may solve this problem. It is left as a future development of this method.
3. The test results have shown that this method can measure the simulated corrosion depth, even with noise introduced, above 95% accuracy. Better results are obtained for Sample A (97%, 99% and 99%) than Sample B (95%, 100% and 98%).
4. The histogram of each corrosion simulation test should theoretically consist of one value only, except the ones with normally distributed random noise added. The deviation from the theoretical value is due to several factors: the non-planar nature of the exposed surface, the closeness of the edges of the selected exposed area to the protected area and the way the corrosion depth is calculated, using the Matlab method.
    a. The non-planar nature of the exposed surface, especially Sample B, prevents a perfect alignment of the surface normal to the z-axis in every part of the surface. In creating the simulation files, the simulated corrosion depth is added to the z-values of all the points in the aligned exposed area of the original scanned image. However, due to the local surface normal variations, the result is not a straight offset movement of the exposed surface. This introduces variations in the image delta.
    b. In Sample B, the selected edges of the exposed area are very close to the protected area. In the tests for the simulated corrosion depth of 4 mm and 4mm with noises, combining with the effect of a slight misalignment of the exposed surfaces with holes before and after corrosion in Step 3 and the non-planar nature of the exposed surface, the inner quadrilateral has cropped a small part of the upper protected area. This produces a small





amount of corrosion depths from 2.0 to 3.5 mm as seen in the histogram of Sample B with 4.00 mm corrosion depth +/- 5% noise added (Appendix 0). The diagrams on the right show the different views of the cropped exposed surface area in this case. The points concerned are enclosed in the blue polygon. However, they do not affect the accuracy of the corrosion depth measurement. This problem can be corrected by selecting a smaller exposed area for Sample B. However, this is not done to show this is a possible outcome.

   c. The interpolation of the scan points into the common grid points using local cubic surfaces introduces variances into the intended result. The median values and the tight σ values of the corrosion depth in the 0.08 and 4 mm test cases have shown that the method works relatively well.

5. The Matlab code for analyzing the corrosion depth results are written to handle corrosion depth variations. The histogram, statistical data and 2D distribution display of the measured corrosion depth, should reveal valuable information as to the nature of the corrosion.
6. During the engineering exercise, normally distributed random noises have been introduced to the corroded surface of the simulated scanned images. Unfortunately, it occurs that if too high a noise level (e.g. +/- 25%) is introduced, PolyWorks will not be able to create an underlying surface to these scan points, hence the method developed will not work. Hence this exercise restricts itself to +/-5% as a maximum.
7. The long term applicability of this process, though it cannot be validated, has reasonable assurances. In years to come, the scanner can be replaced with a newer one as long as it can output the point cloud image in text file format. PolyWorks can be replaced by any packages having the same functionality to manipulate point cloud data. The Matlab code may not run because of a new language format or platform changes. However the theories and algorithms involved will remain the same. One can easily recode the program to suit the new language format required.
8. The current method provides only limited data on corrosion features, e.g. pits. It provides the minimum, maximum and distribution of the corrosion depth. One can recognize the pits on the surface using the corrosion depth distribution diagram. In future, it may be feasible to extract more information about pits or other corrosion features such as the location, size and depth.

5 Future Improvements

The investigation is by no means complete given the time constraint. There are possible future improvements of the current proposal.

The proposed method is based on a single scanned image of the samples. Unfortunately, the visibility of the boundaries of a sunken surface is not consistent for scans to be performed years apart. This is because the scanning angle is not controllable. At a different angle, the bottom edges of the sunken surface may not be visible from the scanner. To overcome this problem, a multiple scan operation can be used. Each sample will be scanned multiple times at different angles to ensure the entire sunken surface including the transition slops, is covered. PolyWorks can be used to stitch all these point cloud images together to create one single scanned image. This will replace the IMO and IME files in Step 1 of the process. In Step 4, the current Matlab software uses a smaller (10% reduction) inner polygon to calculate the corrosion volume. With the multiple scans approach, the Matlab software should be modified to use an enlarged outer and inner polygon for the corrosion volume calculation. The enlarged polygons are to ensure that the transition slops and bottom corrosion surface are fully covered for corrosion volume calculation. A possible enlargement factor can be $\alpha = 1.3$ and $\alpha = 1.1$ for the outer polygon and inner polygon respectively. The rest of the process will remain the same.

The current method avoids the determination of the actual boundaries of the sunken surface. Instead, it requires the operator to identify them in the original sample image (Step 2). An alternate approach will be to use image processing capabilities in Matlab to detect the actual irregular boundaries of the exposed surface and the transition slops. This approach will require multiple scanning of the exposed surface.

The current method uses PolyWorks to align the before and after corrosion images with the exposed area removed (IMO_hole and IME_hole). This is required because PolyWorks does not allow users to specify areas to ignore during the alignment process. One can find a different software tool to provide this capability or it can be implemented in Matlab. In this case, the entire process can be greatly simplified. The operator will just scan the sample and run the Matlab program to get the results.

The current method provides only limited data on corrosion features, e.g. pits. It provides the minimum, maximum and distribution of the corrosion depth. One can recognize the pits on the surface using the corrosion depth distribution diagram. In future, it may be feasible to extract more information about pits or other corrosion features such as the location, size and depth.

6 Conclusion

In this engineering exercise, a usable process has been developed to measure the corrosion levels of buried samples. This method is based on existing equipment at hand. It combines the powers of our Konica Minolta scanner, PolyWorks and Matlab. The proposed process has long term sustainability since the engineering theory is well documented. It can be easily modernized into a newer form making use of newer equipment and software tools at the time. Comparing corrosion volumes for each corrosion measurement is still a challenge yet to be resolved. However, the method developed has achieved accuracy over 95% in measuring the mean corrosion depth of the sample.

Appendix A Theory & Engineering Notes

A.1 Image delta of two point cloud data sets

A.1.1 The Concept

The key engineering problem of this evaluation is how to calculate the offset of the corroded surface scanned image from the original surface scanned image of the same buried sample. As the corrosion may take considerable time (e.g. 5 years) to occur, the two image scans have to be performed independently, some years apart. This makes it impossible to scan the samples in an identical setup. The scanned images will have different scan origins, 3D spaces and total number of scan points. The scan lines and points of one image will not align with the other. So the first step is to align the 3D space of the two scanned objects. Even though the two 3D spaces become the same, the scan lines and scan points are not aligned. Hence, the delta of the two scanned images cannot be calculated by finding the difference of the z values. The two surfaces have to be fitted between the points and the corresponding positions have to be found to calculate the image delta.

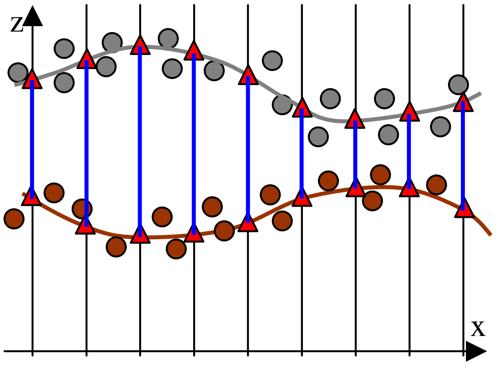


Figure 2: Image delta

The method[4] developed in this exercise is based on how 3D surfaces are represented in Matlab. A 2D regular grid is created in the XY plane of the two point cloud sets as shown in Figure 2. Figure 2 illustrates the concept on a 2D cross section of the 3D space on an X-Z plane. The grey and brown dots represent the point cloud data of the original surface and the corroded surface respectively. The grey and brown curves represent the local fitted curves (cross section of fitted surfaces) to the two point cloud data sets. The red triangles represent the interpolated points (z values) of the fitted curves at the x grid positions. The blue lines are the corrosion depth (image delta) that will be calculated.

The XY gridding of the point cloud is effectively a sub-sampling of the point cloud data. It reduces the computational points from 96K to a manageable size which is controllable by the resolution of the grid. The z-value delta is a simple matrix operation between the grid points. One important note is that the two scan point clouds may cover different areas because the two images are scanned with different scanning angles. So some grid points of one image may not have corresponding points in the other on which to find the delta. This is especially true around the perimeters of the two point cloud data. This has to be taken care of in the image delta calculation.

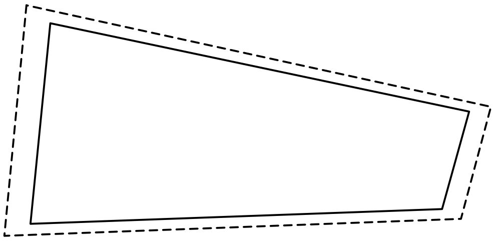
---

[4] This method is developed by Dennis and Zheng

A.1.2 The Implementation

Matlab provides a good number of existing functions to support the grid generation and interpolation of the point cloud data into a grid. The interpolating function allows users to select some simple surfaces to fit the local region of points. In the code, a cubic surface is used for this purpose.

The common grid for the two point cloud data sets is created with a size based on the minimum and maximum x and y values of the combined two data sets. The grid (z values) values of each point cloud sets are represented in an m-by-n matrix, corresponding to the x-y positions of the grid. Since the grid space is larger than or equal to the point cloud space, the z values of the uncovered area in the matrix are filled with NaN (not a number, a special value in Matlab). The corrosion depth is found by subtracting the two matrices. The resulting m-by-n matrix of the image delta is not totally numerical. It contains a large number of NaNs, especially around the edges. In the calculation of the corrosion volume, these values have to be skipped.



It is found that the surface fitting function in Matlab, griddata, may generate abnormalities at the region around the edge of the interpolated surface. Depending on the local surface gradient at the edge area, the resulting surface may rise up or fall down in the edge regions introducing erroneous results in the image delta calculation. To work around this problem, it is recommended to use a two steps approach:

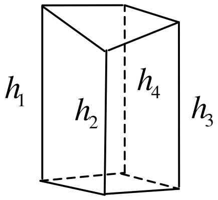
1. The image delta is calculated over the entire region (area enclosed in dotted lines in the figure) and
2. The result is interpreted within a reduced area in contour with the external boundaries (area enclosed in solid lines in the figure).

From the experimental results, it is found that a 10% margin will be sufficient for this purpose.

A.2 Corrosion Volume and Mean Depth Calculation

A.2.1 Calculation

In the corrosion depth calculation, a regular grid (square) is needed to sample the two fitted images surfaces. The corrosion depth at a grid point is calculated as the z-distance between the two surfaces at the corresponding grid sample points on the two surfaces. This creates a number of hexahedrons with square cross-sections. When the grid resolution is high, the corrosion volume can be approximated by the summing of the volumes of all these hexahedrons.



Take one of the hexahedron as example:
Δv_i=cross section area x average height

$$\Delta v_i = s^2 \times \frac{1}{4}\sum_{j=1}^{4} h_j \,,$$

where s is the side of the square cross-sectional area.
Hence, the total corrosion volume, $V_E \approx \sum_i \Delta v_i$,

And the total corrosion area, $A_E = n \times s^2$ where n=total number of hexahedrons in the image delta.

The mean corrosion depth, $\overline{h}_E = \frac{V_E}{A_E}$

A.2.2 Implementation

Unfortunately, even though the before and after corrosion scans can be perfectly aligned using image processing techniques, the two images may still have different coverage areas. This is because the two scan may be performed at slightly different scan angles and the resulting images may have scan points existing in one scan and not in another. The image delta thus derived from Appendix A.1 will have invalid values (NaNs) in some locations.

To find the total corrosion volume and the mean corrosion depth, one has to scan every hexahedron in the image delta, check for its validity (i.e. having 4 real heights) before summing its volume and area towards the total. The mean corrosion depth thus derived is of higher accuracy than the method documented in Version 1 of this document.

A.3 Fitting a plane to the image

A.3.1 The Theory

The image is a set of N points in 3D space $\begin{pmatrix} x_i & y_i & z_i \end{pmatrix}_N$. A plane in 3D space is defined by $Ax + By + Cz + D = 0$, in general. To fit a plane through the point cloud is to find the set of coefficients $\begin{pmatrix} A & B & C & D \end{pmatrix}$ that will provide the least error through all the N points. One should note that there is not a unique set of $\begin{pmatrix} A & B & C & D \end{pmatrix}$ for this plane. If all 4 numbers are divided with a constant, the plane remains the same. This problem is a classical linear[5] regression problem with multiple variables.

Let us rewrite the plane into a more useful form: $z = a_0 + a_1 x + a_2 y$
When a point i from the point cloud points is substituted into this plane:
$z_i = a_0 + a_1 x_i + a_2 y_i + e_i$, where e_i is the error.
$e_i = z_i - \left(a_0 + a_1 x_i + a_2 y_i\right)$
The optimal solution of the a's, in the least square sense, will be when

$$E = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left[z_i - \left(a_0 + a_1 x_i + a_2 y_i\right) \right]^2$$

is minimum, i.e. at $\frac{\partial E}{\partial a_0} = 0$, $\frac{\partial E}{\partial a_1} = 0$ and $\frac{\partial E}{\partial a_2} = 0$

Giving:

$$\frac{\partial E}{\partial a_0} = -2\sum_{i=1}^{N} \left(z_i - a_0 - a_1 x_i - a_2 y_i \right) = 0$$

$$\frac{\partial E}{\partial a_1} = -2\sum_{i=1}^{N} x_i \left(z_i - a_0 - a_1 x_i - a_2 y_i \right) = 0$$

$$\frac{\partial E}{\partial a_2} = -2\sum_{i=1}^{N} y_i \left(z_i - a_0 - a_1 x_i - a_2 y_i \right) = 0$$

Giving:

$$\begin{bmatrix} N & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \sum z_i \\ \sum x_i z_i \\ \sum y_i z_i \end{Bmatrix}^{6} \qquad (1.1)$$

---

[5] The term linear refers to A, B, C and D. It can have any non-linear combinations of x, y and z.

[6] In this document, $\left[A\right]$ represents a matrix and $\left\{A\right\}$ represents a vector.

Let $[P] = \begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & & \vdots \\ 1 & x_i & y_i \\ \vdots & & \vdots \\ 1 & x_N & y_n \end{bmatrix}$, then $[P]^T = \begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \\ x_1 & & x_i & & x_N \\ y_1 & \cdots & y_i & \cdots & y_N \end{bmatrix}$ and $\{A\} = \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix}$, $\{Z\} = \begin{Bmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_n \end{Bmatrix}$

Then equation (1.1), can be rewritten in matrix form as:

$$[P]^T [P]\{A\} = [P]^T \{Z\}$$

Solving for $\{A\}$:

$$\{A\} = \left([P]^T [P]\right)^{-1} \left([P]^T \{Z\}\right) \tag{1.2}$$

A.3.2 Calculation

Equation (1.2) is easily solved in Matlab by using the matrix inverse function:

$$\{A\} = \left([P]' * [P]\right) \backslash \left([P]' * \{Z\}\right)$$

The plane $\begin{pmatrix} A & B & C & D \end{pmatrix}$ will be $\begin{pmatrix} -a_1 & -a_2 & 1 & -a_0 \end{pmatrix}$

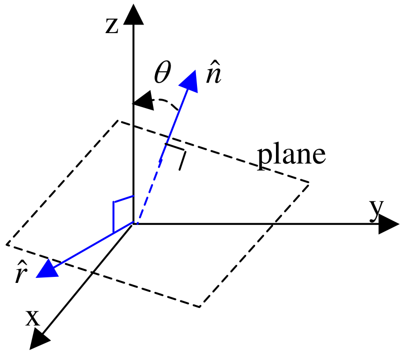
A.4 Rotating an image

A.4.1 Background

In this approach, the corrosion depth at each point on the exposed surface of the sample is found by subtracting the z value of the corresponding point from the original surface. However, this assumes that the image is scanned precisely in a vertical direction. This is impractical, in general. Hence, in order for the corrosion calculation to work, the image must be rotated so that the normal of the image surface aligns with the z-axis.

A.4.2 The Calculation

To find out the general orientation of the image, the simplest method is to fit a plane through the image and use the plane normal to determine the angle of rotation and the axis of rotation required.



Supposed a plane $Ax + By + Cz + D = 0$ is fitted through the point cloud image. A normal vector N is given by the vector $\{A \quad B \quad C\}^T$.

The unit normal to the plane is

$$\hat{n} = \frac{N}{|N|} = \begin{Bmatrix} A \\ B \\ C \end{Bmatrix} / \sqrt{A^2 + B^2 + C^2}$$

The angle of rotation θ is the angle made by $\hat{n}$ with the z-axis $\hat{z} = \{0 \quad 0 \quad 1\}^T$ and the axis of rotation $\hat{r}$ should be normal to the plane formed by $\hat{n}$ and $\hat{z}$ in a direction obeying the right hand rule. Hence:

$$\theta = \cos^{-1}\left(\hat{n} \bullet \hat{z}\right)$$

$$\hat{r} = \frac{\hat{n} \times \hat{z}}{|\hat{n} \times \hat{z}|}$$

The transformation matrix T to rotate a point $\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$ to a new position $\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix}$ through an angle θ with an axis of rotation $\hat{r} = \begin{Bmatrix} r_x \\ r_y \\ r_z \end{Bmatrix}$ is given by the following relationship:

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = T \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

where

$$T=\begin{bmatrix}(1-\cos\theta)r_x^2+\cos\theta & (1-\cos\theta)r_xr_y-r_z\sin\theta & (1-\cos\theta)r_xr_z+r_y\sin\theta\\(1-\cos\theta)r_yr_x+r_z\sin\theta & (1-\cos\theta)r_y^2+\cos\theta & (1-\cos\theta)r_yr_z-r_x\sin\theta\\(1-\cos\theta)r_zr_x-r_y\sin\theta & (1-\cos\theta)r_zr_y+r_x\sin\theta & (1-\cos\theta)r_z^2+\cos\theta\end{bmatrix}^{7}$$

Or,

$$T=Sym(\hat{r})(1-\cos\theta)+Skew(\hat{r})\sin\theta+I\cos\theta$$

where

$$Sym(\hat{r})=\hat{r}\times\hat{r}^T=\begin{bmatrix}r_x^2 & r_xr_y & r_xr_z\\r_xr_y & r_y^2 & r_yr_z\\r_xr_z & r_yr_z & r_z^2\end{bmatrix},\ Skew(\hat{r})=\begin{bmatrix}0 & -r_z & r_y\\r_z & 0 & -r_x\\-r_y & r_x & 0\end{bmatrix}$$

and I is the 3x3 identity matrix.

Hence for an image $IM=\begin{bmatrix}x_1 & y_1 & z_1\\\vdots & & \vdots\\x_i & y_i & z_i\\\vdots & & \vdots\\x_n & y_n & z_n\end{bmatrix}$, the rotated image:

$$IM_R=IM*T^T$$

---

[7] Reference: Josef Hosckek and Dieter Lasser, “Fundamentals of Computer Aided Geometric Design”,1993,A K Peters, ISBM:9781568810072

This can be checked by setting the rotation vector to be the x-axis, i.e. $r_x=1,\ r_y=r_z=0$, then

$[T_x]=\begin{bmatrix}0 & 0 & 0\\0 & \cos\theta & -\sin\theta\\0 & \sin\theta & \cos\theta\end{bmatrix}$, similarly $[T_y]=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\0 & 0 & 0\\-\sin\theta & 0 & \cos\theta\end{bmatrix}$ and

$[T_z]=\begin{bmatrix}\cos\theta & -\sin\theta & 0\\\sin\theta & \cos\theta & 0\\0 & 0 & 0\end{bmatrix}$

A.5 Finding the boundaries of the exposed area

In order for us to compare the corrosion volumes from one corrosion level to another for the same sample, the boundaries of the corroded surfaces in each corrosion volume calculation have to be the same. Doing it manually on PolyWorks does not provide this consistency. An automatic method has to be developed.

In this process, it is proposed to use the XY projection of the exposed surface of the un-corroded sample to be used as a fixed template. Once this template is extracted, one can use it to extract the corresponding surfaces from subsequent corroded samples. In this way, one will have consistent boundaries to calculate our corrosion volume. To do this, the boundaries of the exposed un-corroded surface have to be extracted. As this surface is obtained from PolyWorks by cutting all the rest of the sample image away, the resulting image points usually form a quadrilateral (a 4-sided polygon). This Appendix contains the engineering details on the algorithm developed in Matlab to extract this quadrilateral automatically.

A.5.1 Forming the pattern

In a MRI or CT scanned image, the 3D space of the image is filled with pixels with intensity values. Edge detection techniques are well developed for these types of images. In a point cloud image, however, there are no pixels and the spaces between points are not filled. Hence, to use the edge detection techniques, one has to create a pixel-like pattern from the point cloud data. To complicate the situation, the organization of the image points in the image file is also unknown to us. PolyWorks output them in some orders but is not consistent from point cloud file to another. All one can rely on are the x and y values of the points.

The solution proposed is to form a 2D grid and fit all the points into this grid according to their x and y values. The grid creates an array of pixels with values either 1 or 0. Pixels with one or more points located in their area are set to 1 otherwise 0. In this way, the order of the points in the data set is immaterial.

In order to form a solid pattern for edge detection, the resolution of this grid is very important. The resolution has to be greater or equal to the spacing of the scan lines in the image file. Otherwise, the resulting pattern will mainly consist of broken line segments (Figure 3). As the spacing of the scan lines can be obtained easily from PolyWorks, in viewing the point cloud data at close ranges, it is left to the user to enter this value into the grid forming process.

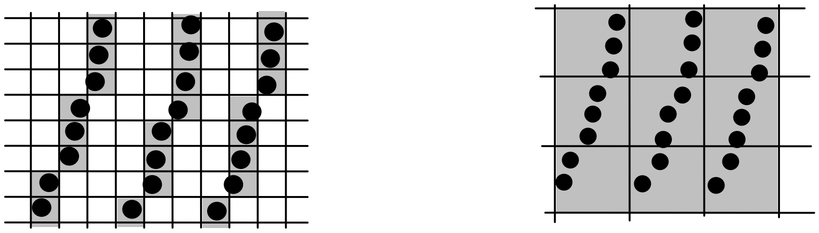


Figure 3: The effect on the patterns form by using different grid resolutions

A.5.2 Edge detection

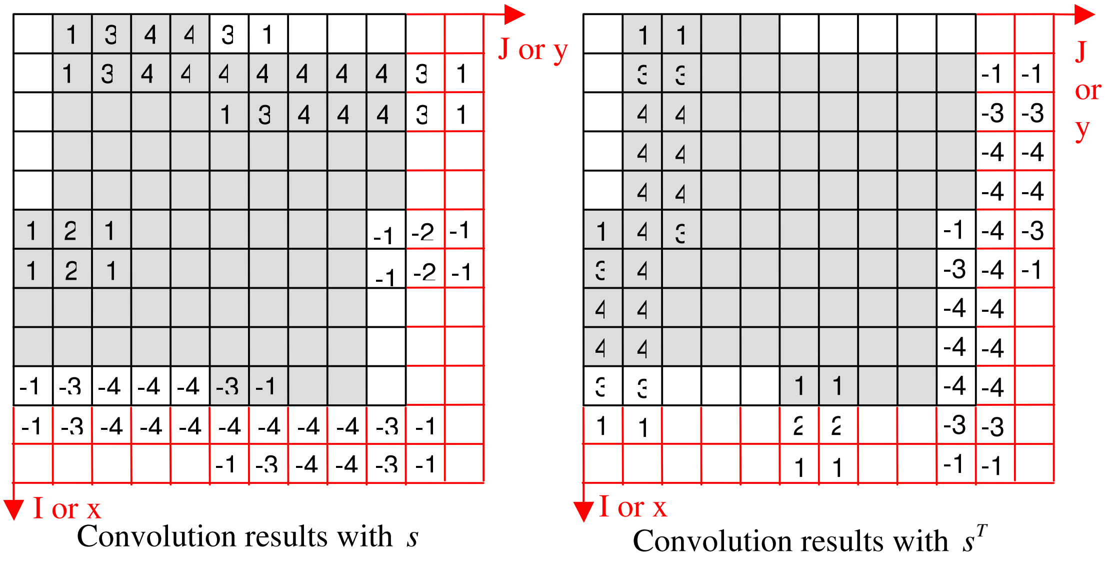
Broadly speaking, edge detection methods can be classified into two types: gradient and Laplace (change of gradient). In most of these methods, a mask (usually square) is used to convolute with the 2D image. The resulting array contains a pattern easy for edge extraction. The theory of convolution, contained in many text books, is beyond the discussion of this document. Herein, only the practical aspect of this method is of interest.

Many edge detection methods have been developed. We propose to use the Sobel method, which is simple to implement (as compared with the Canny method), insensitive to noises (as compared with Laplace methods) and produces good results in this situation.

The Sobel method is a gradient method. It uses a 3x3 mask to calculate the intensity gradient of the image. The mask it uses are:

$$s = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \textit{and} \qquad s^T = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Where s is used for horizontal edge detection and s^T, the transpose of s, for vertical edge detection. For an m-by-n image, the convolution result is an m+2 by n+2 array.



Convolution results with s

	1	3	4	4	3	1					
	1	3	4	4	4	4	4	4	4	3	1
					1	3	4	4	4	3	1
1	2	1							-1	-2	-1
1	2	1							-1	-2	-1
-1	-3	-4	-4	-4	-3	-1					
-1	-3	-4	-4	-4	-4	-4	-4	-4	-3	-1	
					-1	-3	-4	-4	-3	-1	

Convolution results with s^T

	1	1									
	3	3								-1	-1
	4	4								-3	-3
	4	4								-4	-4
	4	4								-4	-4
1	4	3							-1	-4	-3
3	4								-3	-4	-1
4	4								-4	-4	
4	4								-4	-4	
3	3				1	1			-4	-4	
1	1				2	2			-3	-3	
					1	1			-1	-1	

Figure 4: Convolution results of a 10x10 binary pattern with the Sobel edge detectors

Figure 4 uses a 10x10 pattern to illustrate the results of the Sobel edge detectors. The grey pattern represents a 10x10 binary image of a 4-sided polygon with edges not parallel to the I- and J- axis. The grey cells have values 1 and all the rest 0. The convolution result of each edge detection operation is a 12x12 sparse array. The additional elements of this array are shown in red and the values of the non-zero elements are shown.

For the horizontal edge detection, it is notable that:

- All cells of value greater than, or equal to 3, represent the location of the top edge (the west edge in a proper XY plane plot).
- All cells of value less than, or equal to -3, represent the location of the bottom edge (the east edge in a proper XY plane plot). However, the I values of these cells are offset by 2.
- The right-most cell of each edge may be outside the 10x10 image frame; hence they should be ignored in the edge detection process.

For the vertical edge detection, here are some important notes:

- All cells of value greater than, or equal to 3, represent the location of the left edge (the south edge in a normal XY plane plot).
- All cells of value less than or equal to -3 represent the location of the right edge (the north edge in a normal XY plane plot). However, the J values of these cells are offset by 2.
- The bottom cell of each edge may be outside the 10x10 image frame; hence they should be ignored in the edge detection process.

In general, the Sobel method produces thicker edges (2 cells width in our example) as compared with other methods, e.g. Canny. However, it is a one step method. The thicker edges do not cause a problem, because it is the points within these cells that are used for the straight line fitting. A thicker edge only means the resulting straight line will have more points to average with and its location will be slightly within the boundary of the original image. This should be an advantage.

A.5.3 Boundary polygon determination

After extracting the 4 groups of edge cells from the pattern, the remaining work is to establish the 4 boundary edges of the image. A simple least-square based curve fitting function in Matlab (polyfit) is used. The 4 boundary edges are assumed to be of the form: $y = ax + c$ for the horizontal lines and $x = by + c$ for the vertical lines[8].

Once the 4 boundary edges are found, it is simple to determine the corners, $\begin{pmatrix} x_{ij} & y_{ij} \end{pmatrix}$, of the boundary polygon by solving the intersection points of any 2 boundary edges i and j from the 4 straight line equations found:

$$\begin{bmatrix} a_i & b_i \\ a_j & b_j \end{bmatrix} \begin{Bmatrix} x_{ij} \\ y_{ij} \end{Bmatrix} = \begin{Bmatrix} c_i \\ c_j \end{Bmatrix}, \text{ where i, j=1 to 4 and i} \neq \text{j,}$$

solving

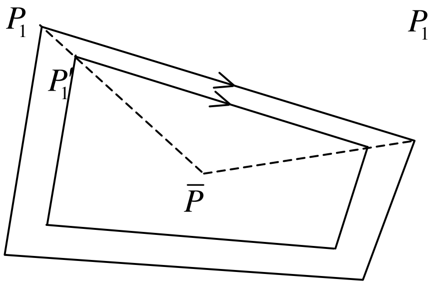
$$\begin{Bmatrix} x_{ij} \\ y_{ij} \end{Bmatrix} = \begin{bmatrix} a_i & b_i \\ a_j & b_j \end{bmatrix}^{-1} \begin{Bmatrix} c_i \\ c_j \end{Bmatrix}$$

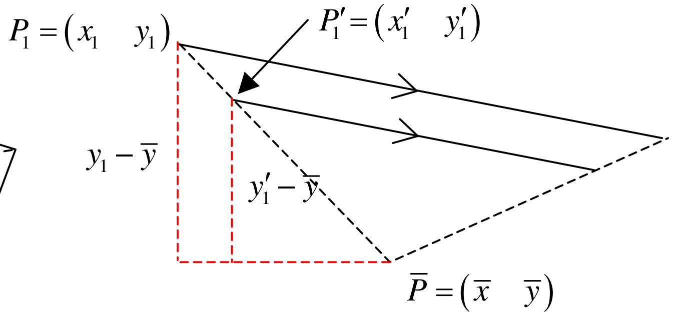
---

[8] It is found that the Matlab function, polyfit, does not work well for lines close to vertical in the XY plane, hence the fitting order is reversed instead.

A.5.4 Contour polygon calculation with constant scale factor

From section A.1.2, it is found that the surface interpolation of the point cloud image will introduce edge effects. An inner contour polygon is required to filter out erroneous results from the image delta. To determine the vertices of this inner polygon, one can use the following calculations.





$P_1 = (x_1 \quad y_1)$
$P_1' = (x_1' \quad y_1')$
$y_1 - \overline{y}$
$y_1' - \overline{y}$
$\overline{P} = (\overline{x} \quad \overline{y})$

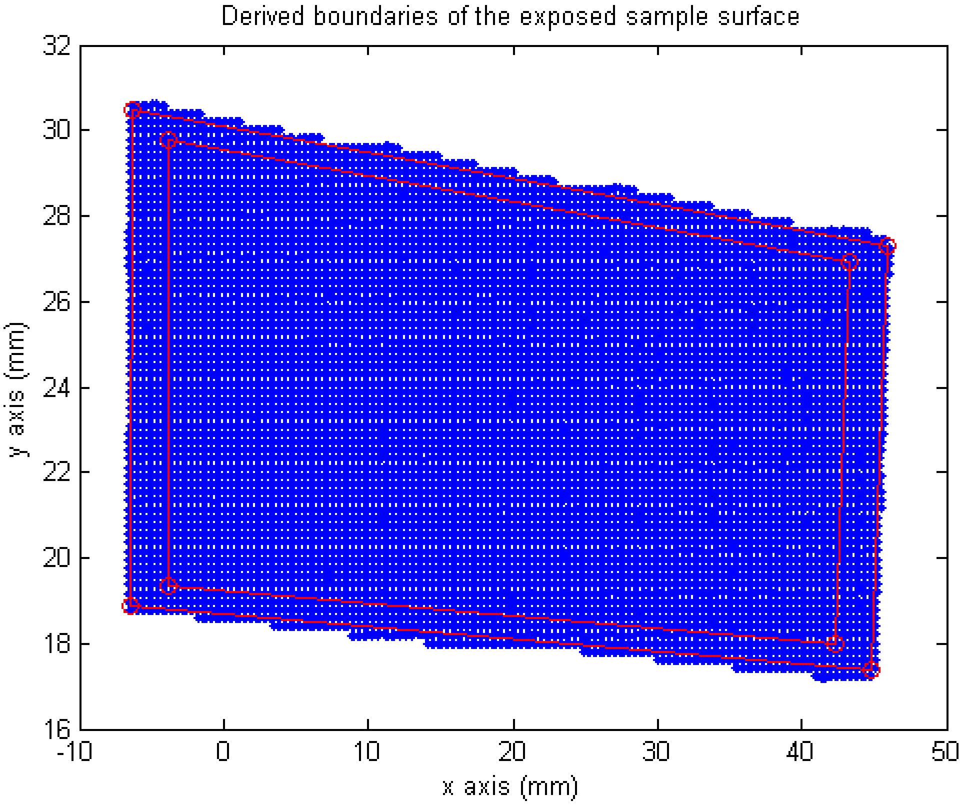
Take any point within the polygon, e.g. the centroid $\overline{P}$, then

$$\overline{P} = (\overline{x} \quad \overline{y}) = \frac{1}{4}\sum_{i}^{4}(x_i \quad y_i)$$

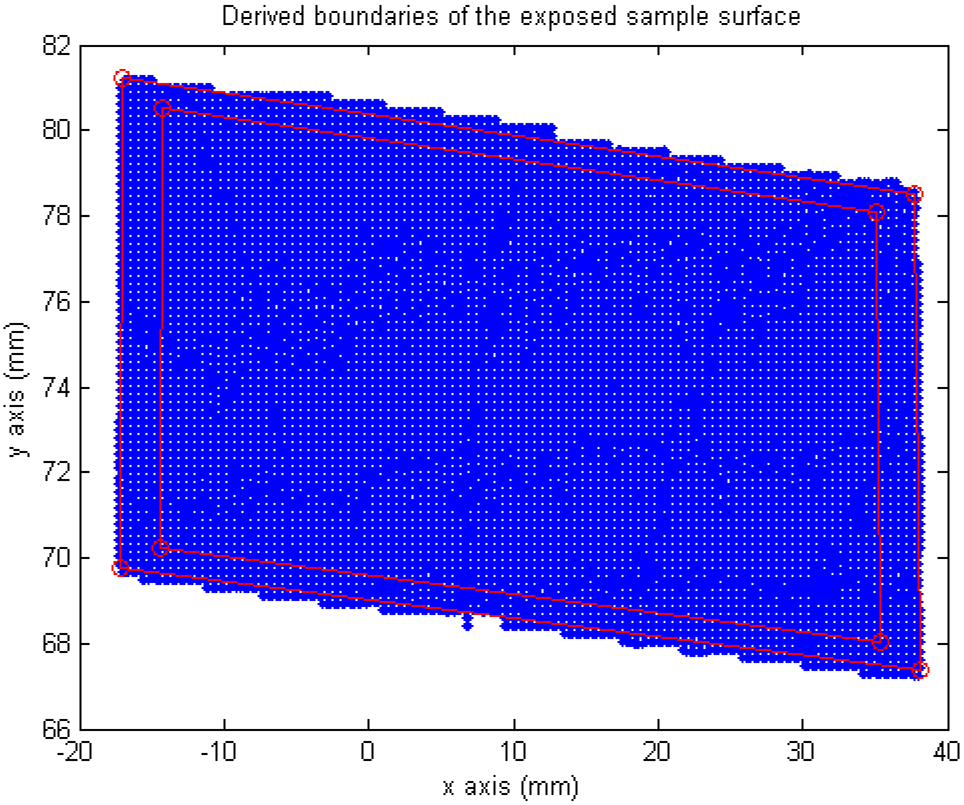
If $\frac{P_i'\overline{P}}{P_i\overline{P}} = \alpha$, the scale factor, then $\frac{x_i' - \overline{x}}{x_1 - \overline{x}} = \frac{y_i' - \overline{y}}{y_1 - \overline{y}} = \alpha$, giving

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \alpha \begin{pmatrix} x_i \\ y_i \end{pmatrix} + (1-\alpha)\begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}$$, for i=1 to 4.

The following figure shows the boundary polygon obtained for the exposed surface of sample A with $\alpha = 0.1$



The following figure shows the boundary polygon obtained for the exposed surface of sample B with $\alpha = 0.1$


Derived boundaries of the exposed sample surface
y axis (mm)
x axis (mm)
82
80
78
76
74
72
70
68
66
-20
-10
0
10
20
30
40

Appendix B Image Processing Software

B.1 Matlab Code

The Matlab code main6 contains the common code for the image processing and corrosion volume calculation for each corroded sample. The operator is expected to modify the front portion of the code to point to the input and output file directories and the file names required.

B.1.1 Sample A

Simulated Corrosion Depth 0.08 mm

```
function main6
% This function calculates the depth of erosion of a buried sample by
% calculating the image difference of the original surface and the eroded
% surface after some time of burial.
% This function carries the following steps:
%
% Notes:
% 1.    All image files are cloud point files. The positions of the cloud
%       are represented in the file as 3 columnes of numbers corresponding
%       to the x, y and z values of the points respectively. The numbers
%       are represented in ASCII form with 6 places of decimals. This is
%       the format that PolyWork use to output the point clouds.

%History:
%2008-09-09 created from main5_part1 and main5_part2
%2008-09-22 remove sub-function format
%2008-09-23 in find_image_boundaries, polyfit will fail when the resulting
%           line is close to verticle. Need to reverse x and y to get a
%           proper fit
%2008-09-26 add polygon area calculation
%2008-10-01 1.  change the accuracy of erosion volume calculation to improve
%               its accuracy.
%           2.  add dynamic range for displaying ed_2D
%2008-10-15 Use outside and inside polygons for croping and result
%           interpretation respectively to
%           1.  cut down edge distortion introduced by surface interpolation
%           2.  avoid including portion of the protected surface due to the
%               non-planar nature of the corroded surface making the
%               surface normal only a rough estimation.
%2008-10-16 Modify main & rotate_image to crop the rotated Imo using the
%           polygon instead of Imo_exp. This is more correct.

    clear all;
    close all;

    %---------------------------
    % Program parameters setup section
    %---------------------------
    % Input files
    fpath = ['..' filesep '..' filesep];        %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];         %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-0.08' filesep];   %input file directory for eroded
sample
    %The original image of the sample before erosion
    fn_imo = [fpathi1 '200808111515 Sample A - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200809241146 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after erosion
    fn_ime = [fpathi2 '200809241503 EST 0.08 Step 1 - Start Pos.txt'];
    %The image of the sample after erosion with eroded part cut off
    fn_imeh = [fpathi2 '200809241510 EST 0.08 Step 2.txt'];
    %The aligned image of the eroded sample with eroded part cut off
    fn_imeha = [fpathi2 '200809241548 EST 0.08 Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-0.08' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full eroded image aligned with the original image
    fn_imea = [fpathod 'EST 0.08 Step 4.txt'];
    %The eroded exposed surface after rotation to align the surface normal with z-axis
    fn_ime_exp_r = [fpathod 'EST 0.08 Step 4-Exp R.txt'];
    %The original exposed surface after rotation and cropped
    fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];

    %The resolution required for edge detection: scan line spacing distance
    S_LINE_DIST = 0.2;  % distance in mm
    %The resolution for the image processing
    RESOLUTION = 0.05;  % distance in mm

    %----------------------------
    % main program
    %----------------------------

    % read all images
    Imo=read_image(fn_imo);
    Ime=read_image(fn_ime);
    Imeh=read_image(fn_imeh);
    Imeha=read_image(fn_imeha);
    Imo_exp=read_image(fn_imo_exp);

    % Calculate the transformation matrix to apply to Imeh to get Imeha
    % Note: The transformation must be a 3x3 matrix
    T=Imeh\Imeha;
    % Transform Ime so that it will align with Imo
    Imea=Ime*T;
    % Save image to file for display
    save_image(fn_imea,Imea);

    %Orientate images so that their surface normal are parallel to the z-axis
    [Imo_exp_r,Imo_r,Imea_r]=rotate_images(Imo_exp,Imo,Imea);

    % Find the boundaries of the original exposed surface projected onto
    % the XY plane. The boundaries are defined as a 4-sided polygon. Two
    % polygons are determined. The outer one is for image cropping and the
    % inside one is for image delta results filtering.
    [poly_out,poly_in]=find_image_boundaries(Imo_exp_r,S_LINE_DIST);
    % Use the polygons to isolate the image area of the original exposed
    % surface and the eroded sample for erosion depth calculation
    Imo_exp_r=crop_image(Imo_r,poly_out);
    Ime_exp_r=crop_image(Imea_r,poly_out);

    % Save all files for validation
    save_image(fn_imo_exp_r,Imo_exp_r);
    save_image(fn_ime_exp_r,Ime_exp_r);

    % Calculate the degree of erosion and display results
    ed2D=image_delta(Imo_exp_r,Ime_exp_r,RESOLUTION,poly_in);   %the erosion distribution
in 2D
    display_results(ed2D,RESOLUTION,poly_in);
return

%-----------------------------------
% Helper Functions
%-----------------------------------
function im=read_image(fname);
    fid=fopen(fname);
    im=fscanf(fid,'%g %g %g',[3 inf]);
    im = im';
    fclose(fid);
return

function save_image(fname,im)
    dlmwrite(fname,im,'delimiter',' ','newline','pc','precision','%.6f');
return

function im_diff=image_delta(im1,im2,res,poly)
```

```
    %This function calculates the z-distance delta of im1 and im2 and
    %im1 is considered as the reference image.
    %res is the resolution of the grid
    %poly is the polygon which is the boundary for valid results
    %Method: It fits a surface each to im1 and im2 and project a grid
    %at resolution res on both surfaces and use the grid points to
    %sample the surfaces for delta. At the end, it filters out all results
    %outside the polygon given.

    % find bounding box and grid for the data
    xyzmin=min([im1;im2]);xyzmax=max([im1;im2]);
    xmin=xyzmin(1);ymin=xyzmin(2);
    xmax=xyzmax(1);ymax=xyzmax(2);
    xlin=linspace(xmin,xmax,(xmax-xmin)/res + 1);
    ylin=linspace(ymin,ymax,(ymax-ymin)/res + 1);

    % setup a common grid for both images
    [Xgrid,Ygrid]=meshgrid(xlin,ylin);

    % fit the image into the grid
    X1=im1(:,1);Y1=im1(:,2);Z1=im1(:,3);
    X2=im2(:,1);Y2=im2(:,2);Z2=im2(:,3);
    Gim1=griddata(X1,Y1,Z1,Xgrid,Ygrid,'cubic');
    Gim2=griddata(X2,Y2,Z2,Xgrid,Ygrid,'cubic');

    % find the difference with the common parts of the two images
    [y_size1,x_size1] = size(Gim1);
    [y_size2,x_size2] = size(Gim2);
    x_size = min([x_size1, x_size2]);
    y_size = min([y_size1, y_size2]);
    im_diff = Gim1(1:y_size,1:x_size) - Gim2(1:y_size,1:x_size);

    % filter out results outside the polygon
    % convert polygon into integer indices into im_diff
    polyIJ=round((poly - ones(size(poly,1),1)*[xmin ymin])/res)+1;
    % generate all the indices of im_diff
    [XI,YI]=meshgrid(1:x_size,1:y_size);XI=XI(:);YI=YI(:);
    % find all indices inside or on the polygon
    [IN,ON]=inpolygon(XI,YI,polyIJ(:,1),polyIJ(:,2));
    OUT=~(IN | ON); % all indices outside the polygon
    % delete all results outside the polygon
    im_diff(sub2ind(size(im_diff),YI(OUT),XI(OUT)))=NaN;
return

function [im1r,im2r,im3r]=rotate_images(im1,im2,im3)
    %This function finds the plane that best fits im1 and rotate both
    %im1, im2 and im3 so that the normal of this plane is pointing in the
    %z-axis.

    %Find the 3D plane that best fits im1 using least square method.
    %The plane is defined as Ax+By+Cz+D=0 that passes through im1,
    ABCD=fit_plane(im1);

    %Calculate the axis of rotation
    %The normal vector to plane Ax+By+Cz+D=0 is (A,B,C)
    Plane_Unit_Norm=ABCD(1:3)/norm(ABCD(1:3));
    %Angle between Z axis and Plane_Unit_Norm
    z_axis=[0,0,1]';
    theta=acos(dot(Plane_Unit_Norm,z_axis));
    %Rotation axis is normal to the plane formed by the z_axis and
    %Plane_Unit_Norm
    R_axis=cross(Plane_Unit_Norm,z_axis);
    RU_axis=R_axis/norm(R_axis); %unit vector of R_axis

    %Create the rotation matrix
    a=RU_axis(1);b=RU_axis(2);c=RU_axis(3);
    sym=RU_axis*RU_axis';skew=[0,-c,b;c,0,-a;-b,a,0];
    Xform=sym*(1-cos(theta))+skew*sin(theta)+eye(3)*cos(theta);
    Xform=Xform';    %rotate the matrix to suit our image format

    %Perform the rotation
    im1r=im1*Xform;
    im2r=im2*Xform;
    im3r=im3*Xform;

    %debug code
    display(['Before orientation, theta = ',num2str(theta*180/pi),' degree']);
    abcd=fit_plane(im1r);
    PUN=abcd(1:3)/norm(abcd(1:3));
    phi=acos(dot(PUN,z_axis));
    display(['After orientation, theta = ',num2str(phi*180/pi),' degree']);
return

function plane=fit_plane(im)
    %This function fits a plane through the image point of cloud using
    %least square method. The plane is desribed by the equation
    %z=a0+a1x+a2y. plane=[-a1 -a2 1 -a0]'
    %Method:
    %This is a multiple linear regression problem to find the a's.
    %If P=[ones X Y] and A=[a0 a1 a2]', then the least square error
    %of plane A will occur at P'*P*A=P'*Z we can solve A.
    n=size(im,1);   %n=number of points in the point cloud image
    P=[ones(n,1) im(:,1) im(:,2)];
    A=(P'*P)\(P'*im(:,3));
    plane=[-A(2);-A(3);1;-A(1)];
return

function [polygon_out, polygon_in]=find_image_boundaries(im,res)
    %This function finds the 4-sided polygon containing im on the XY
    %plane. The projected image of im on the XY plane has to be bounded
    %by 4 straigth lines. res is the resolution required for this
    %operation. res has to be large enough so that when we fit im to
    %the grid formed by res, the resulting pattern will have solid
    %edges in most of the places.
    %This funciton creats an XY grid using res and fits the XY
    %projections of im into this grid. It uses the Sobel edge operator
    %to detect the 4 edges. It then fits 4 lines through the edge
    %points of im and returns the intersections of these 4 lines.
    % 20081014: The polygon is shrunk by 5% to cut down edge effects.

    % project im onto the XY plane
    im_XY=im(:,1:2);
    % fit the project image points into a grid
    xymin=min(im_XY);xymax=max(im_XY);
    dim=uint32(ceil((xymax-xymin)/res))+1; %dim=MxN of the grid
    % calculate the(i,j)cell each point of im_XY to fit into the grid
    im_IJ=uint32(floor((im_XY-ones(size(im_XY,1),1)*xymin)/res))+1;    %index starts from
1
    im_ind=sub2ind(dim,im_IJ(:,1),im_IJ(:,2));
    % fit into the grid
    grid=zeros(dim);
    grid(im_ind)=1;

    % Detect the 4 edges in the grid
    s=[1 2 1;0 0 0;-1 -2 -1];   %Sobel operator
    H=conv2(grid,s);V=conv2(grid,s');   % H&V markes the horizontal and vertical edges
    for i=1:4
        switch i    % north, east, south & west are ref to the XY plane
            case 1  % west edge detection
                [I,J]=find(H>3);
                % ensure J will not be > dim(2)
                idx=find(J<=dim(2)-3);I=I(idx);J=J(idx);
            case 2  % north edge detection
                [I,J]=find(V<-3);
                % compensate offset of Sobel operator
                J=J-2;
                % ensure I will not be be greater than dim(1)
                idx=find(I<=dim(1)-3);I=I(idx);J=J(idx);
            case 3  % east edge detection
                [I,J]=find(H<-3);
                % compensate offset of Sobel operator
                I=I-2;
                % ensure J will not be > dim(2)
                idx=find(J<=dim(2)-3);I=I(idx);J=J(idx);
            case 4  % south edge detection
                [I,J]=find(V>3);
                % ensure I will not be be greater than dim(1)
                idx=find(I<=dim(1)-3);I=I(idx);J=J(idx);
        end
        % find all points belonging to this edge
        pts=im_XY(ismember(im_IJ,[I,J],'rows'),:);
        % fit a straight line through these points
        switch i
            case {1,3}  %east & west edge
                %reverse x and y to get better fit
                line=polyfit(pts(:,2),pts(:,1),1);
                edge(i,:)=[1 -line(1) line(2)];
            case {2,4}  %north & south edge
                line=polyfit(pts(:,1),pts(:,2),1);
                edge(i,:)=[-line(1) 1 line(2)];
        end
    end

    % find the 4 corner points of this boundary
    for i=1:4
        j=i+1;
        if j>4
            j=1;
        end
        a=[edge(i,1);edge(j,1)];
        b=[edge(i,2);edge(j,2)];
        c=[edge(i,3);edge(j,3)];
        polygon_out(i,:)=[a b]\c;
    end
    % shrink the polygon by 10%
    xymean=ones(size(polygon_out,1),1)*mean(polygon_out);r=0.1;
    polygon_in=xymean*r+polygon_out*(1-r);
    % close the polygons
    polygon_out=[polygon_out;polygon_out(1,:)];
    polygon_in=[polygon_in;polygon_in(1,:)];

    % debug code
%     figure;plot(im(:,1),im(:,2),'b.',polygon_out(:,1),polygon_out(:,2),'r-
o',polygon_in(:,1),polygon_in(:,2),'r-o');
%     xlabel('x axis (mm)');ylabel('y axis (mm)');
%     title('Derived boundaries of the exposed sample surface');
return

function imo=crop_image(imi,polygon)
    %This function returns all the points from imi that is within the
    %polygon supplied. The polygon has to be on the XY plane.
    %imi is the input image
    %imo is the cropped output image

    % find the XY projection of imi
    X=imi(:,1);Y=imi(:,2);
    % setup the polygon
    pX=polygon(:,1);pY=polygon(:,2);
    % find all the points inside the polygon
    [IN,ON]=inpolygon(X,Y,pX,pY);
    % find all the image points within the polygon
    imo=imi(IN|ON,:);
%     % debug code
%     figure;
%     plot(polygon(:,1),polygon(:,2),'-ro',imo(:,1),imo(:,2),'b.');
return

function display_results(ed_2D,res,polygon)
    % This function displays the erosion depth in the following forms:
    % 1.    2D image with color coding the erosion depth
    % 2.    the histogram of the erosion depth
    % 3.    the statistical distribution of the erosion depth
    % 4.    the croping polygon for erosion calculation
    % 5.    the erosion volume, effective erosion area and average erosion
    %       depth
    % The input parameter ed_2D is a m-by-n matrix representing the
    % erosion depth values in the X-Y plane.
    % res is the resolution that ed_2d is evaluated upon.
    % poly contains the vertices of the polygon where the erosion
    % volume is based upon.
    % calculate the range for displaying ed_2D
    hmax=max(max(ed_2D));hmin=min(min(ed_2D));hrange=hmax-hmin;
    hmax=hmax+hrange/10;hmin=hmin-hrange/10;    % extend the range a bit
    figure;imagesc(ed_2D,[hmin,hmax]);    %display the erosion image
    title('Corrosion depth distibution on exposed surface (mm)');
    xlabel(['x-axis (unit = ' num2str(res) ' mm)']);
    ylabel(['y-axis (unit = ' num2str(res) ' mm)']);

    % Analyze the erosion level & display the statistics
    ed1D=ed_2D(:);
    % imdelta_2D contains a large number of NaN in its content. This
    % is because the two images may not fully overlap each other and
    % the unoverlapped area generates NaNs in ed2D
    % Remove all NaN elements from ed1D
    ed1D=ed1D(find(~isnan(ed1D)));
    figure;hist(ed1D,1000);grid on; %display the erosion distribution
    title('Corrosion depth histogram');xlabel('Corrosion depth (mm)');ylabel('Counts');
    display(['corrosion [min,median,max]= [',...
        num2str([min(ed1D),median(ed1D),max(ed1D)]),...
        '] in mm']);
    display(['erosion std = ',num2str(std(ed1D))]);
    % Erosion volume from step 5 results
    [ER_vol,ER_area] = calculate_erosion_volume(ed_2D,res);
    display(['Estimated corrosion volume = ',num2str(ER_vol), ' mm^3']);

    display(['Corrosion area = ',num2str(ER_area), ' mm^2']);
    display(['Estimated corrosion depth = ',num2str(ER_vol/ER_area), 'mm']);
    poly=polygon(1:end-1,:);   %remove the end point
    display('Vertices of polygon for corrosion depth calculation :');
    poly
return

function [vol,area]=calculate_erosion_volume(ed_2D,res)
    %This function calulates the erosion volume and effective erosion area.
    %ed_2D is the erosion depths in the x-y plane
    %res is the resolution the erosion depth is calculated upon. Unit in mm
    %This function sums the volume and area of all valid hexahedrons in
    %ed_2D. A valid hexahedron is one with all 4 valid erosion depths (not NaN).

    [M,N]=size(ed_2D);
    sum_h=0;ncells=0;     %ncells is the number of squares that have valid 4 sides
    for i=1:M-1
        for j=1:N-1
            h=ed_2D(i:i+1,j:j+1);h=h(:);
            if all(~isnan(h))
                % This is a valid hexahedron
                sum_h=sum_h+sum(h)/4;
                ncells=ncells+1;
            end
        end
    end

    area=ncells*res^2;
    vol=sum_h*res^2;
return
```

Simulated Corrosion Depth 4.00 mm

```
    % Input files
    fpath = ['..' filesep '..' filesep];        %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];         %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-4.00' filesep];   %input file directory for corroded
sample
    %The original image of the sample before corrosion
    fn_imo = [fpathi1 '200808111515 Sample A - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200809241146 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after corrosion
    fn_ime = [fpathi2 '200809251048 EST 4.00 Step 1 - Start Pos.txt'];
    %The image of the sample after corrosion with corroded part cut off
    fn_imeh = [fpathi2 '200809251101 EST 4.00 Step 2.txt'];
    %The aligned image of the corroded sample with corroded part cut off
    fn_imeha = [fpathi2 '200809251128 EST 4.00 Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-4.00' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full corroded image aligned with the original image
    fn_imea = [fpathod 'EST 4.00 Step 4.txt'];
    %The corroded exposed surface after rotation to align the surface normal with z-axis
    fn_ime_exp_r = [fpathod 'EST 4.00 Step 4-Exp R.txt'];
    %The original exposed surface after rotation and cropped
    fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];
```

Simulated Corrosion Depth 4.00 mm with noise

```
    % Input files
    fpath = ['..' filesep '..' filesep];         %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];          %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-4.00 wn' filesep];   %input file directory for corroded
sample
    %The original image of the sample before corrosion
    fn_imo = [fpathi1 '200808111515 Sample A - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200809241146 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after corrosion
    fn_ime = [fpathi2 '200809251149 EST 4.00 wn Step 1 - Start Pos.txt'];
    %The image of the sample after corrosion with corroded part cut off
    fn_imeh = [fpathi2 '200809251157 EST 4.00 wn Step 2.txt'];
    %The aligned image of the corroded sample with corroded part cut off
    fn_imeha = [fpathi2 '200809251215 EST 4.00 wn Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-4.00 wn' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full corroded image aligned with the original image
    fn_imea = [fpathod 'EST 4.00 wn Step 4.txt'];
    %The corroded exposed surface after rotation to align the surface normal with z-axis
    fn_ime_exp_r = [fpathod 'EST 4.00 wn Step 4-Exp R.txt'];
    %The original exposed surface after rotation and cropped
    fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];
```

B.1.2 Sample B

Simulated Corrosion Depth 0.08 mm

```
    % Input files
    fpath = ['..' filesep '..' filesep];         %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];          %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-0.08' filesep];   %input file directory for eroded
sample
    %The original image of the sample before erosion
    fn_imo = [fpathi1 '200810081809 Sample B - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200810081917 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after erosion
    fn_ime = [fpathi2 '200810091454 EST 0.08 Step 1 - Start Pos.txt'];
    %The image of the sample after erosion with eroded part cut off
    fn_imeh = [fpathi2 '200810091507 EST 0.08 Step 2.txt'];
    %The aligned image of the eroded sample with eroded part cut off
    fn_imeha = [fpathi2 '200810091538 EST 0.08 Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-0.08' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full eroded image aligned with the original image
    fn_imea = [fpathod 'EST 0.08 Step 4.txt'];
    %The eroded exposed surface after rotation to align the surface normal with z-axis
    fn_ime_exp_r = [fpathod 'EST 0.08 Step 4-Exp R.txt'];
    %The original exposed surface after rotation and cropped
    fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];
```

Simulated Corrosion Depth 4.00 mm

```
    % Input files
    fpath = ['..' filesep '..' filesep];        %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];         %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-4.00' filesep];   %input file directory for eroded
sample
    %The original image of the sample before erosion
    fn_imo = [fpathi1 '200810081809 Sample B - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200810081917 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after erosion
    fn_ime = [fpathi2 '200810091614 EST 4.00 Step 1 - Start Pos.txt'];
    %The image of the sample after erosion with eroded part cut off
    fn_imeh = [fpathi2 '200810091621 EST 4.00 Step 2.txt'];
    %The aligned image of the eroded sample with eroded part cut off
    fn_imeha = [fpathi2 '200810091642 EST 4.00 Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-4.00' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full eroded image aligned with the original image
    fn_imea = [fpathod 'EST 4.00 Step 4.txt'];
    %The eroded exposed surface after rotation to align the surface normal with z-axis
    fn_ime_exp_r = [fpathod 'EST 4.00 Step 4-Exp R.txt'];
    %The original exposed surface after rotation and cropped
    fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];
```

Simulated Corrosion Depth 4.00 mm with noise

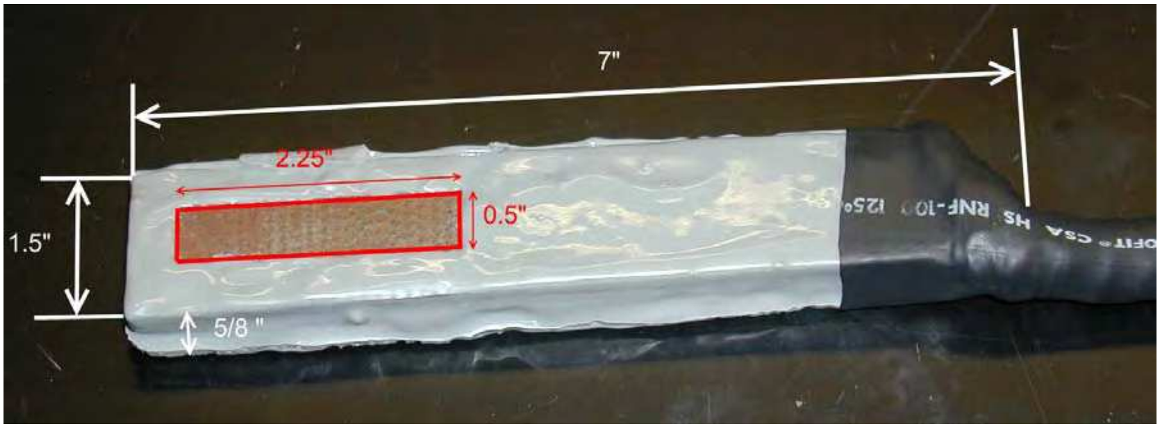
```
    % Input files
    fpath = ['..' filesep '..' filesep];        %file directories common path
    fpathi1 = [fpath 'Take 4' filesep];         %input file directory for original sample
    fpathi2 = [fpath 'Take 4 modZ-4.00 wn' filesep];   %input file directory for eroded
sample
    %The original image of the sample before erosion
    fn_imo = [fpathi1 '200810081809 Sample B - Point Cloud - Take 4.txt'];
    %The exposed area of the original surface
    fn_imo_exp = [fpathi1 '200810081917 Take 4 Step 2 - Exposed.txt'];
    %The image of the sample after erosion
    fn_ime = [fpathi2 '200810091708 EST 4.00 wn Step 1 - Start Pos.txt'];
    %The image of the sample after erosion with eroded part cut off
    fn_imeh = [fpathi2 '200810091715 EST 4.00 wn Step 2.txt'];
    %The aligned image of the eroded sample with eroded part cut off
    fn_imeha = [fpathi2 '200810091734 EST 4.00 wn Step 3 - After.txt'];

    %Output files for display and validation
    fpatho = [fpath 'Take 4 modZ-4.00 wn' filesep];    %output file directory
    today=datestr(date,26);today=today(find(today~='/'));   %remove '/' from today
    fpathod = [fpatho today ' '];    %output file directory + today
    %The full eroded image aligned with the original image
```

```
fn_imea = [fpathod 'EST 4.00 wn Step 4.txt'];
%The eroded exposed surface after rotation to align the surface normal with z-axis
fn_ime_exp_r = [fpathod 'EST 4.00 wn Step 4-Exp R.txt'];
%The original exposed surface after rotation and cropped
fn_imo_exp_r = [fpathod 'EST Take 4 Step 4-Exp R.txt'];
```

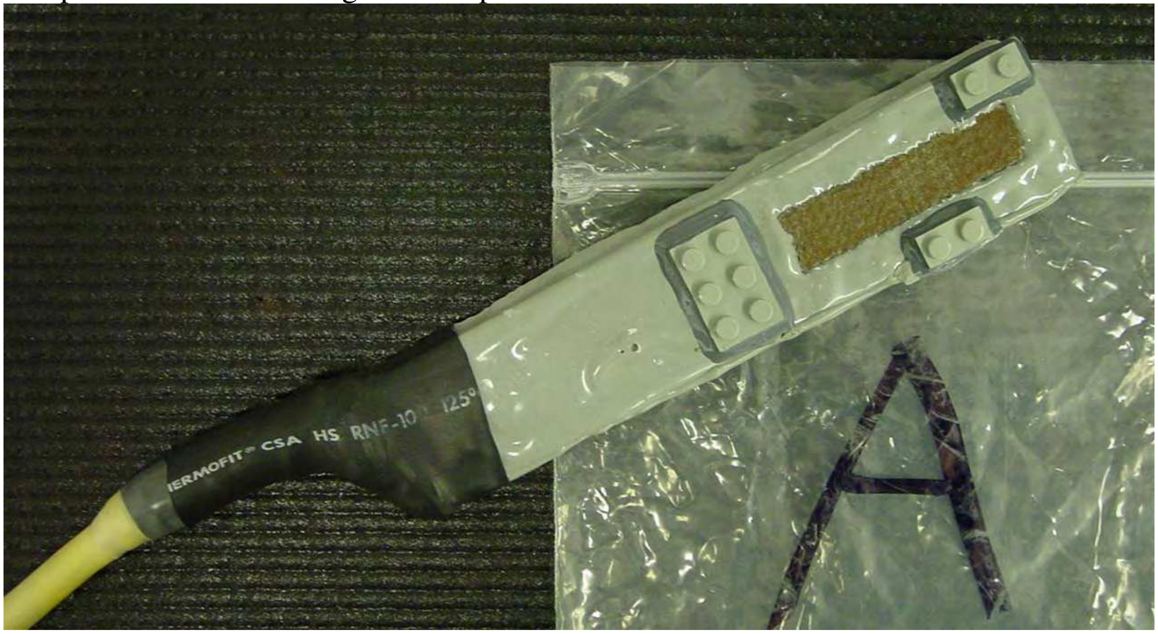
Appendix C Test Results

C.1 Images of Sample A & B

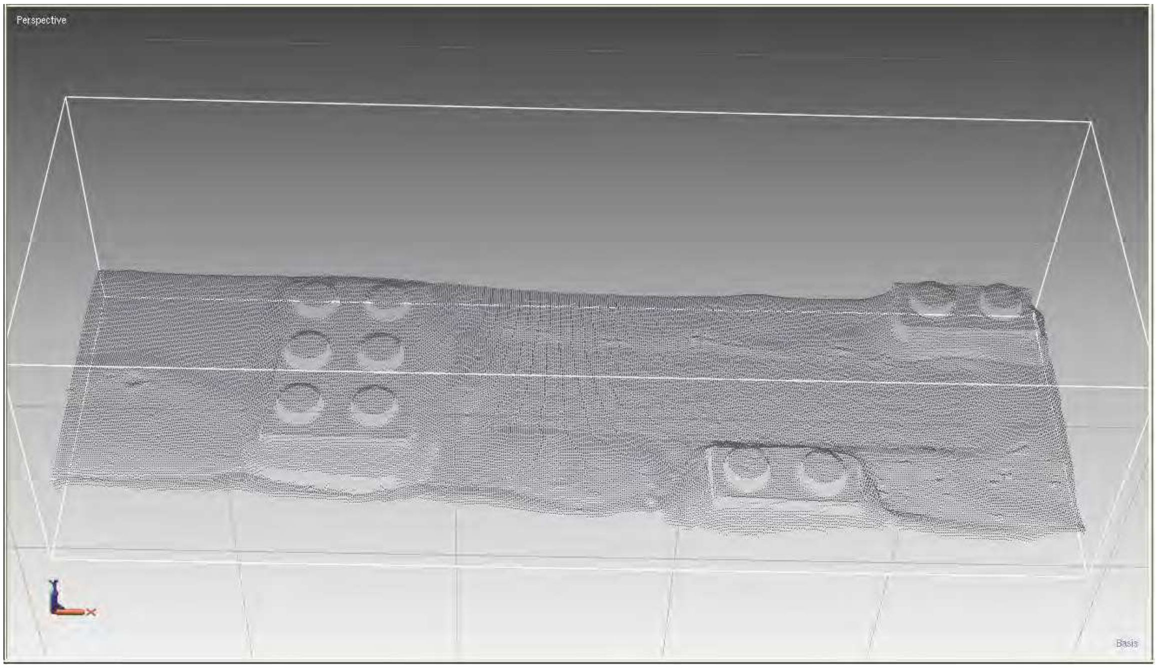
C.1.1 Sample A



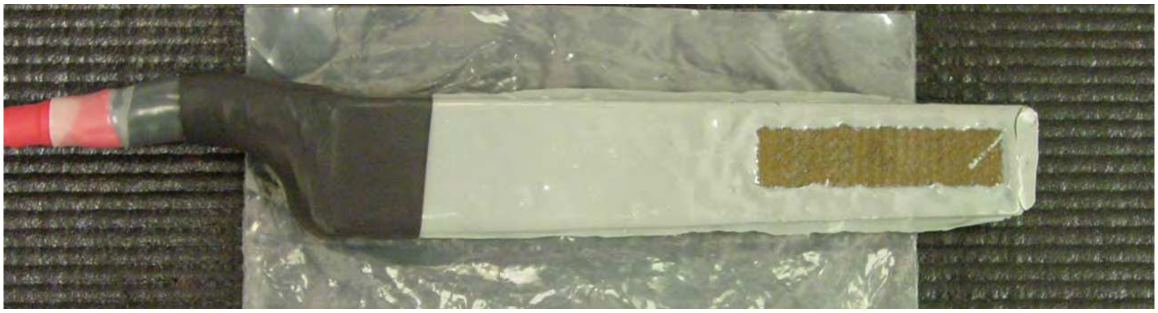
Sample A with surrounding surfaces profiles modified:



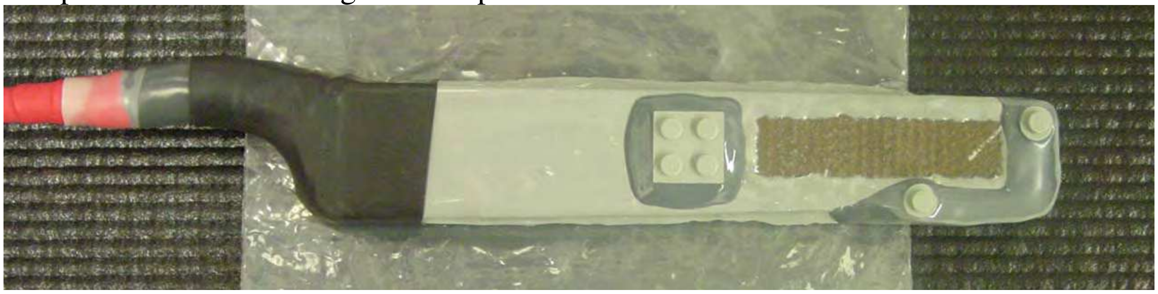
Point cloud view:



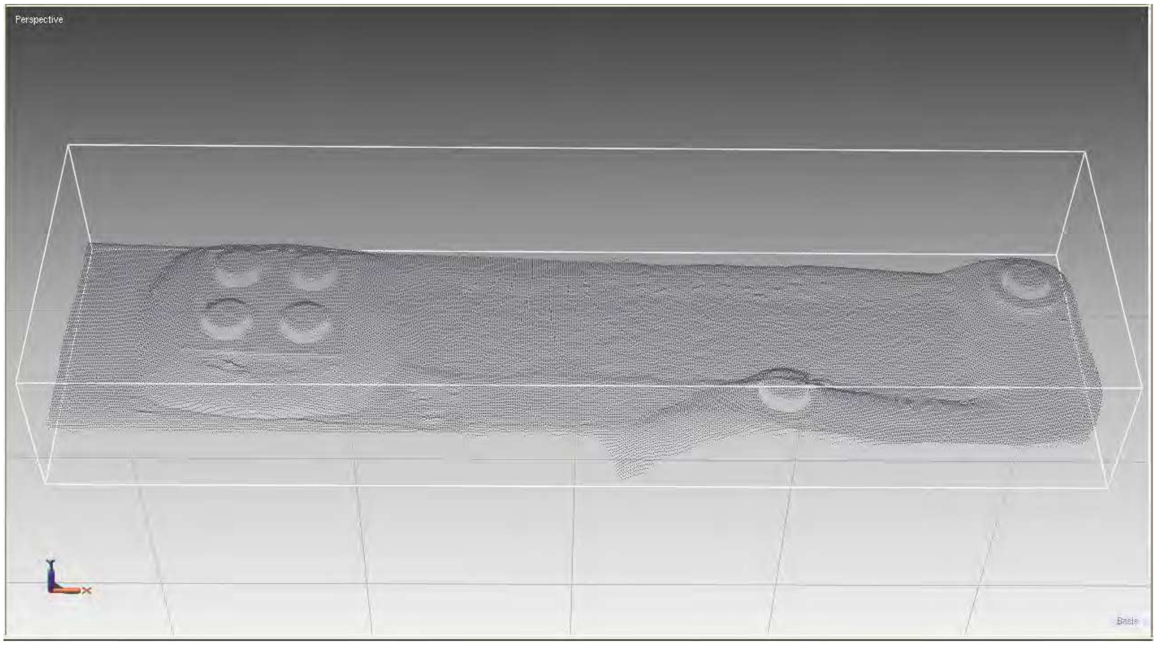
C.1.2 Sample B:



Sample B with surrounding surfaces profiles modified:



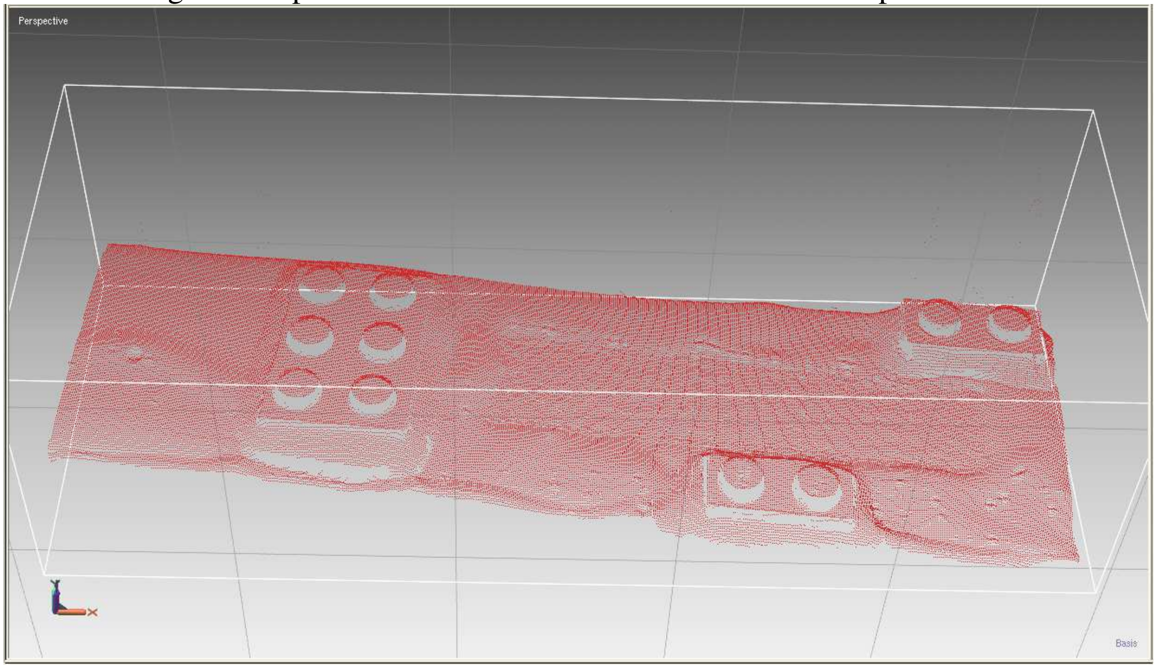
Point cloud view:


Perspective
x
Basic

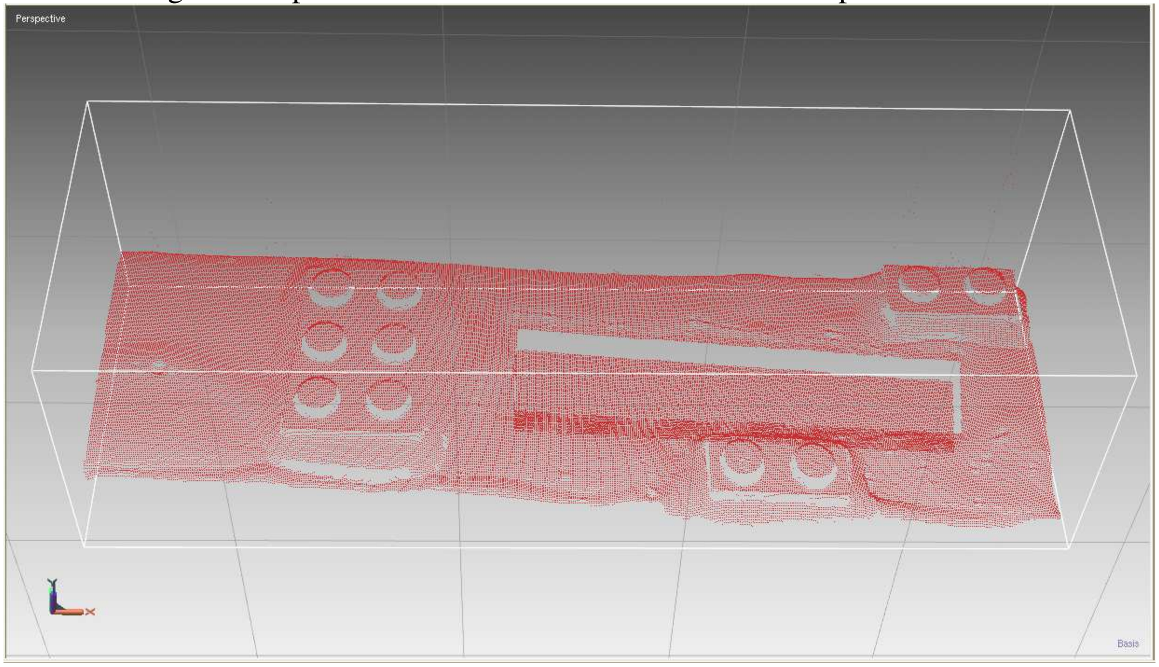
C.2 Images of Simulated Corrosion Samples

C.2.1 Sample A

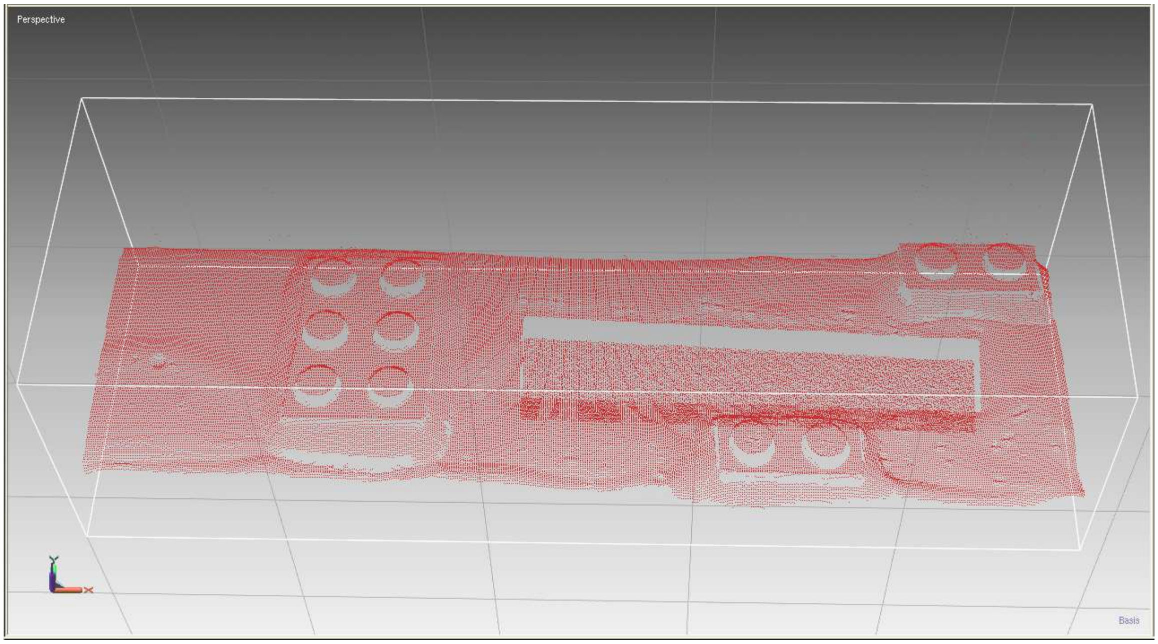
Scanned image of sample with 0.08 mm simulated corrosion of Sample A



Scanned image of sample with 4 mm simulated corrosion of Sample A.

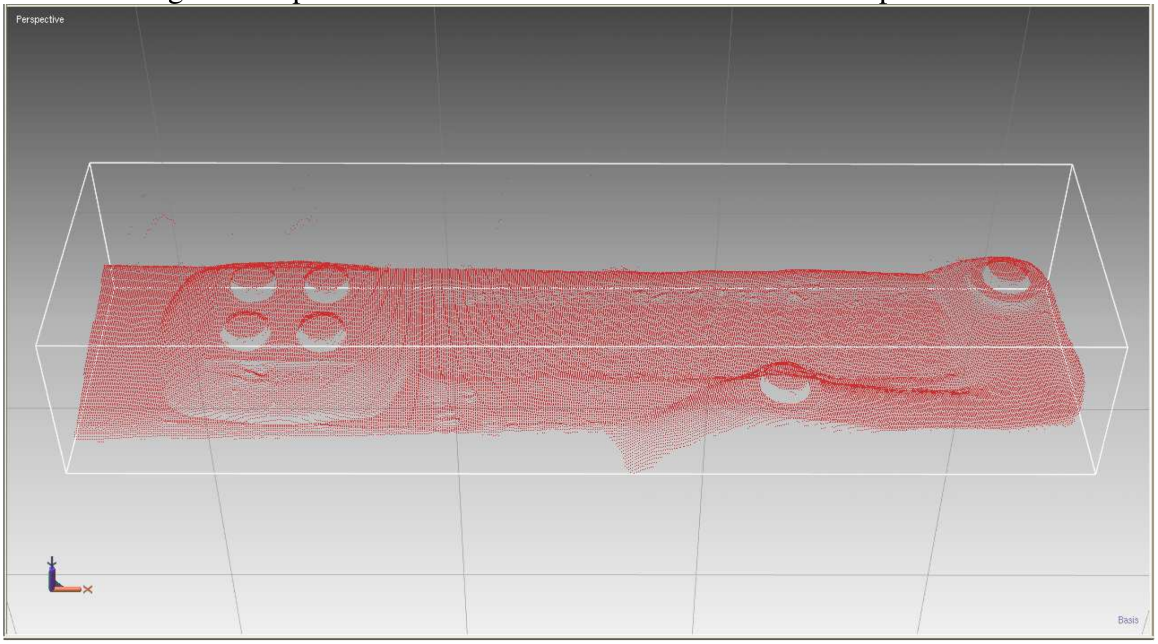


Scanned image of sample with 4 mm simulated corrosion of Sample A with 5% noise added.

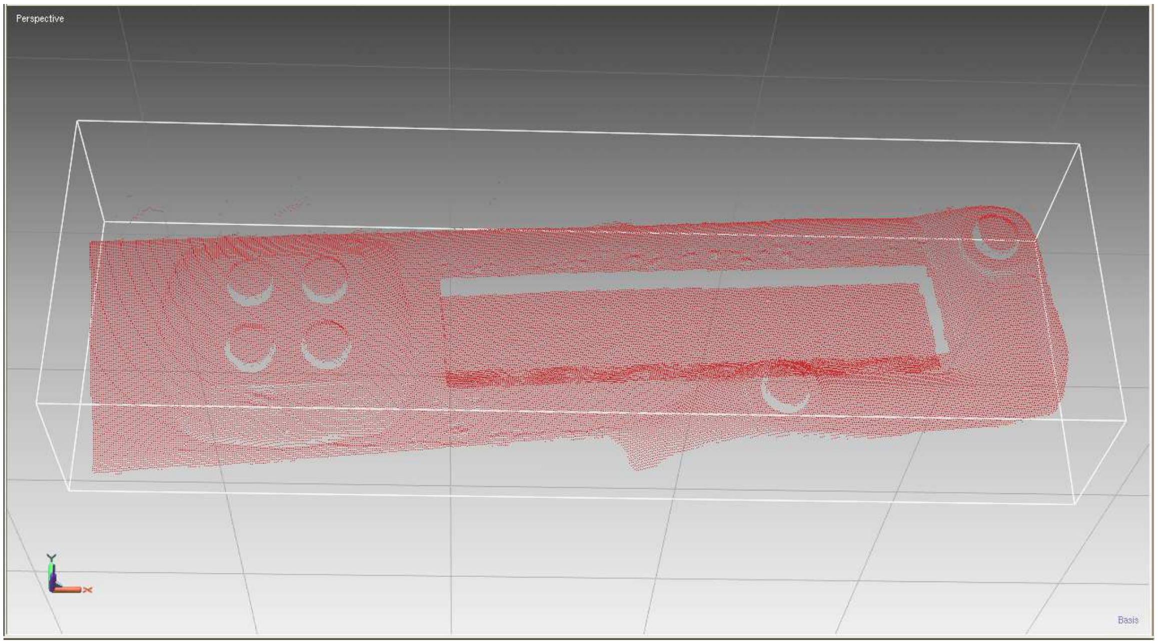


C.2.2 Sample B

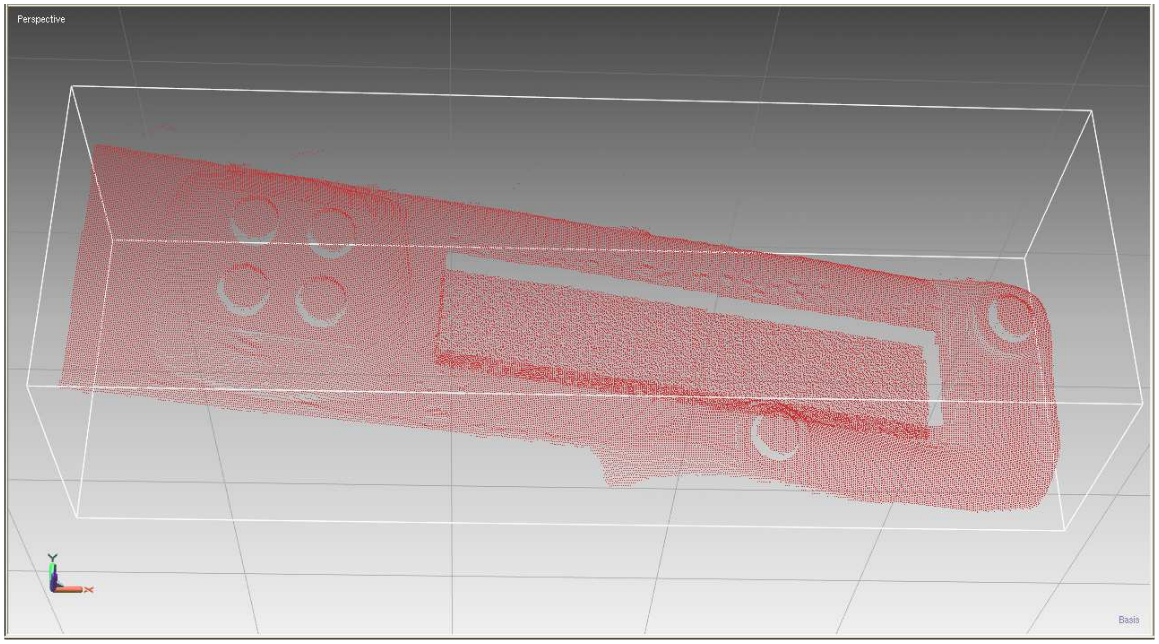
Scanned image of sample with 0.08 mm simulated corrosion of Sample B



Scanned image of sample with 4 mm simulated corrosion of Sample B.



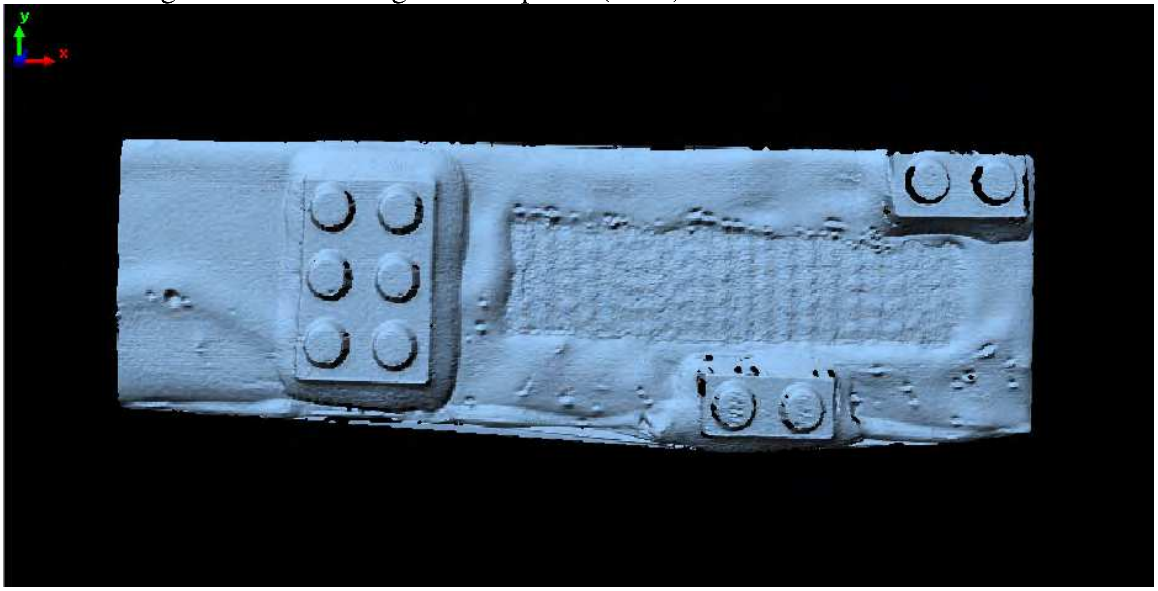
Scanned image of sample with 4 mm simulated corrosion of Sample B with 5% noise added.



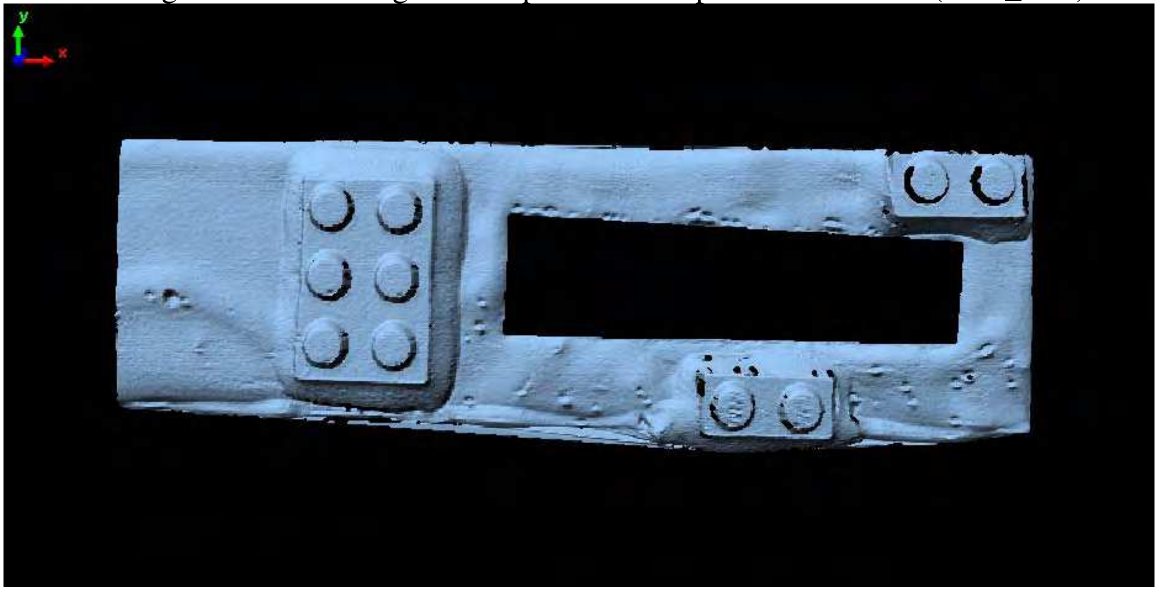
C.3 Operation results from test files

C.3.1 Sample A

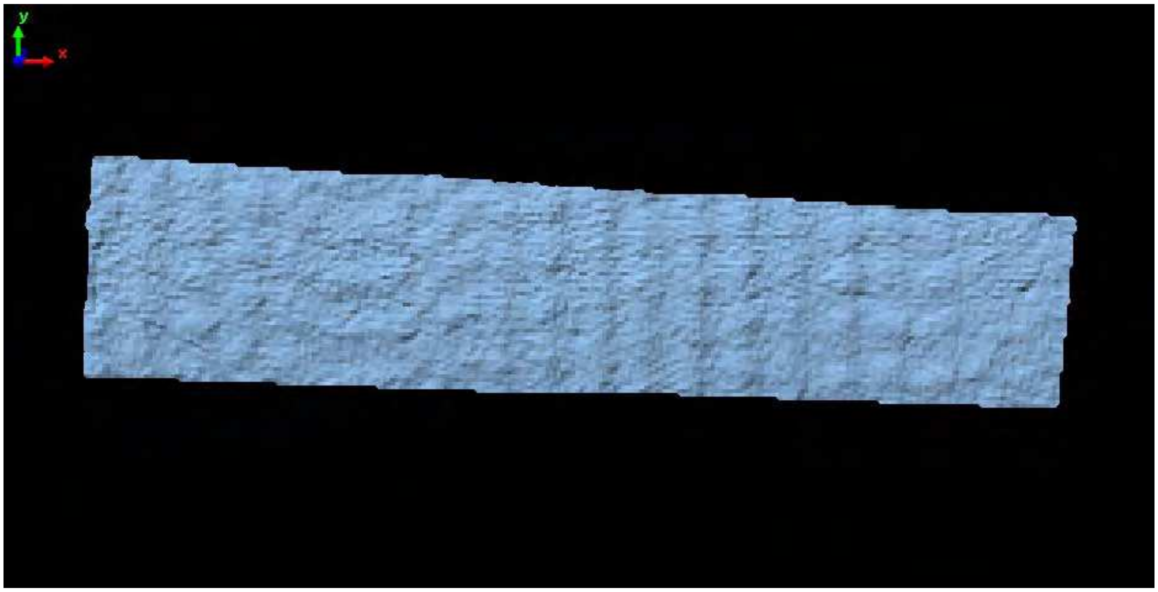
STEP 1. Original scanned image of Sample A (IMO)



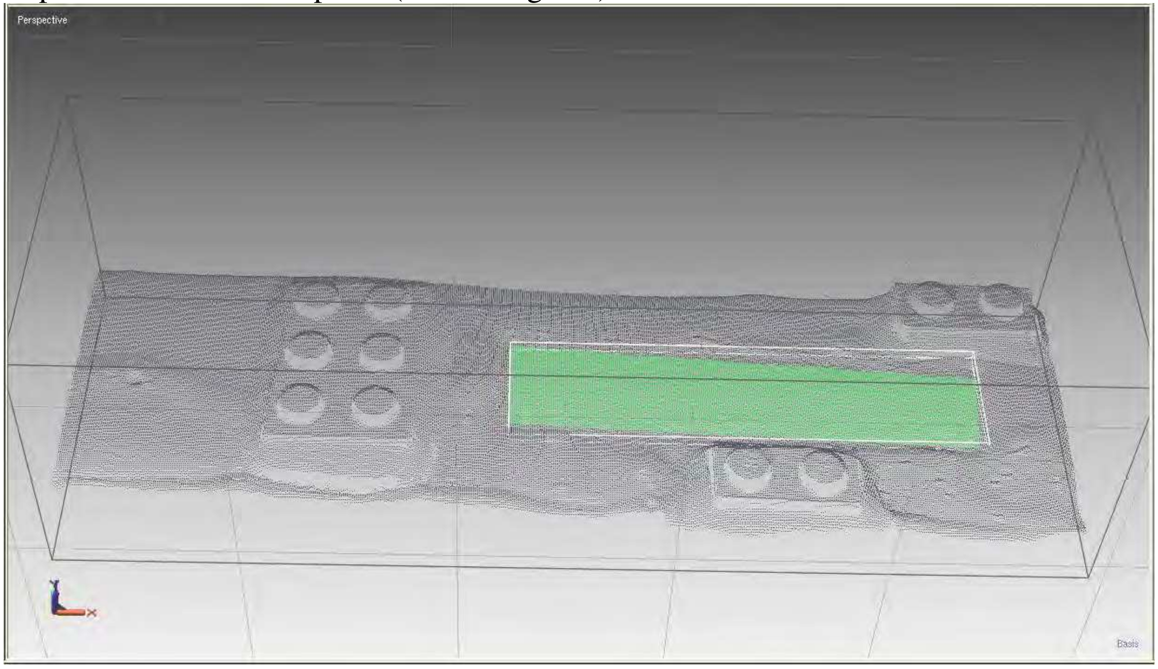
STEP 2: Original scanned image of Sample A with exposed area cut off (IMO_hole)



STEP 2: Exposed area of original scanned image of Sample A (IMO_exposed)

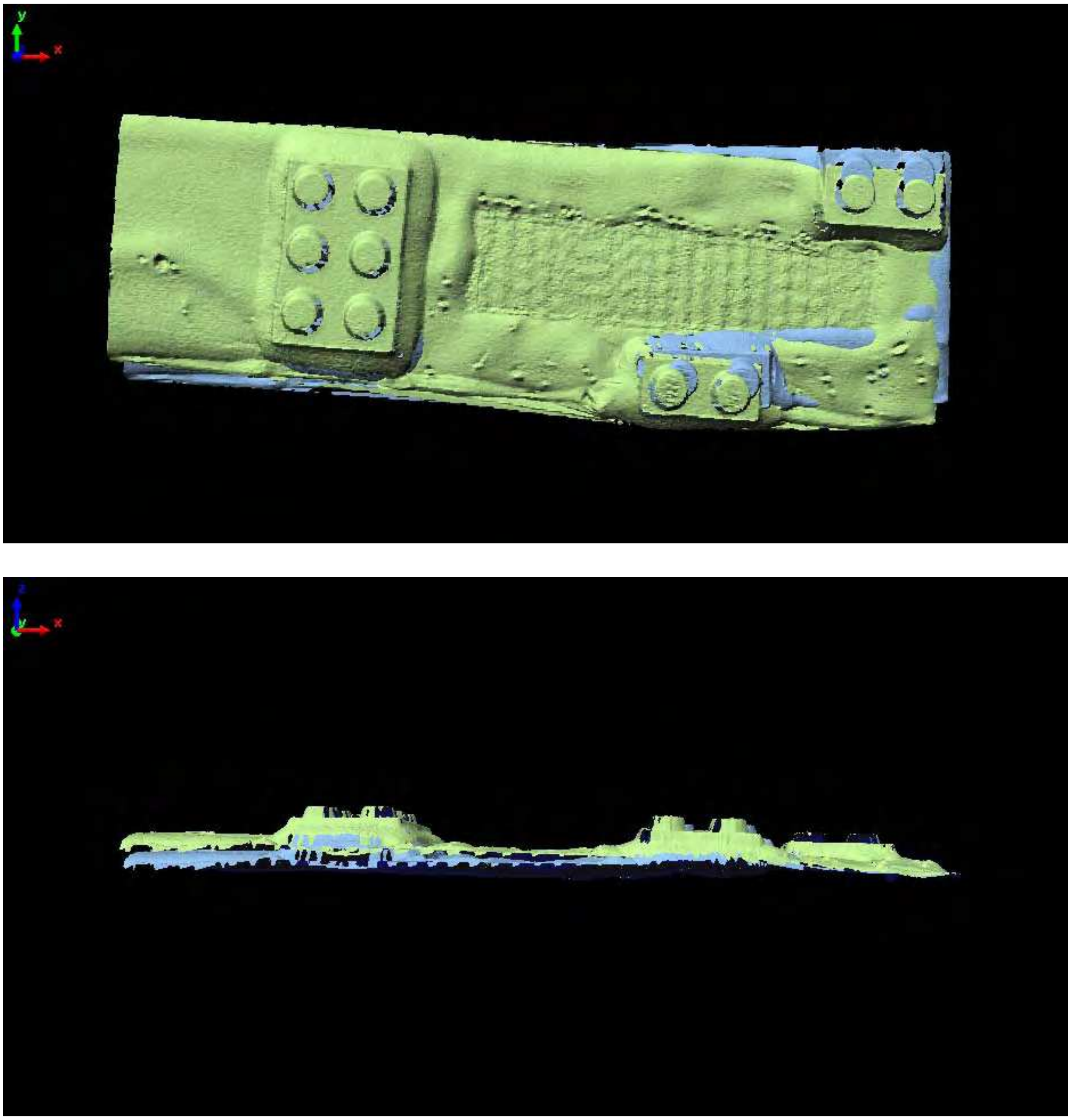


Exposed surface of Sample A (shown in green):

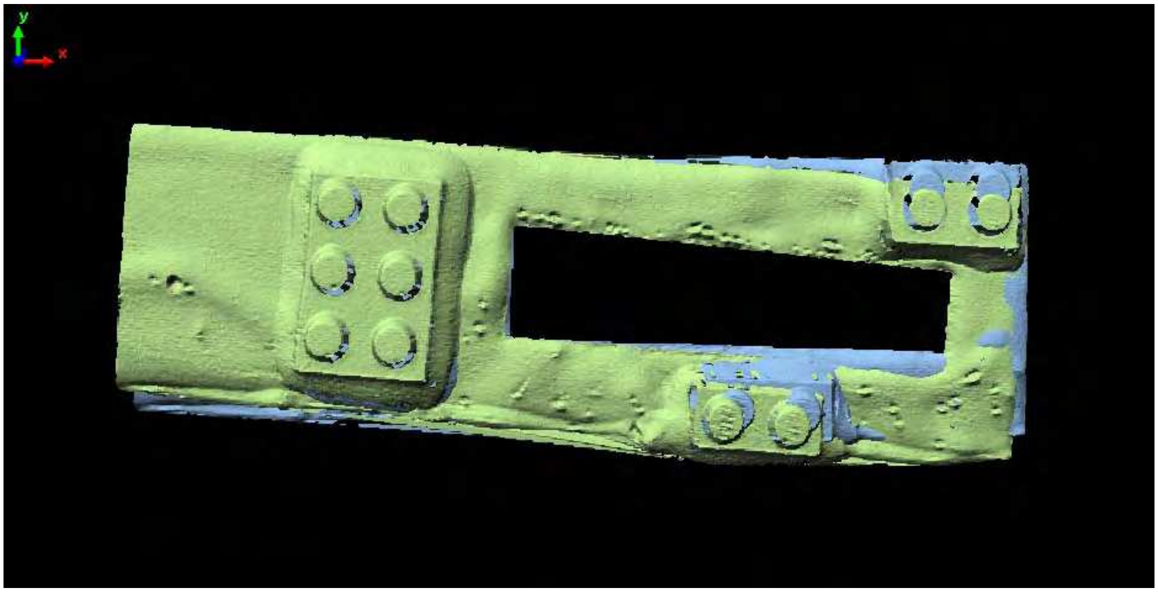


Test Results for the 0.08 mm Process

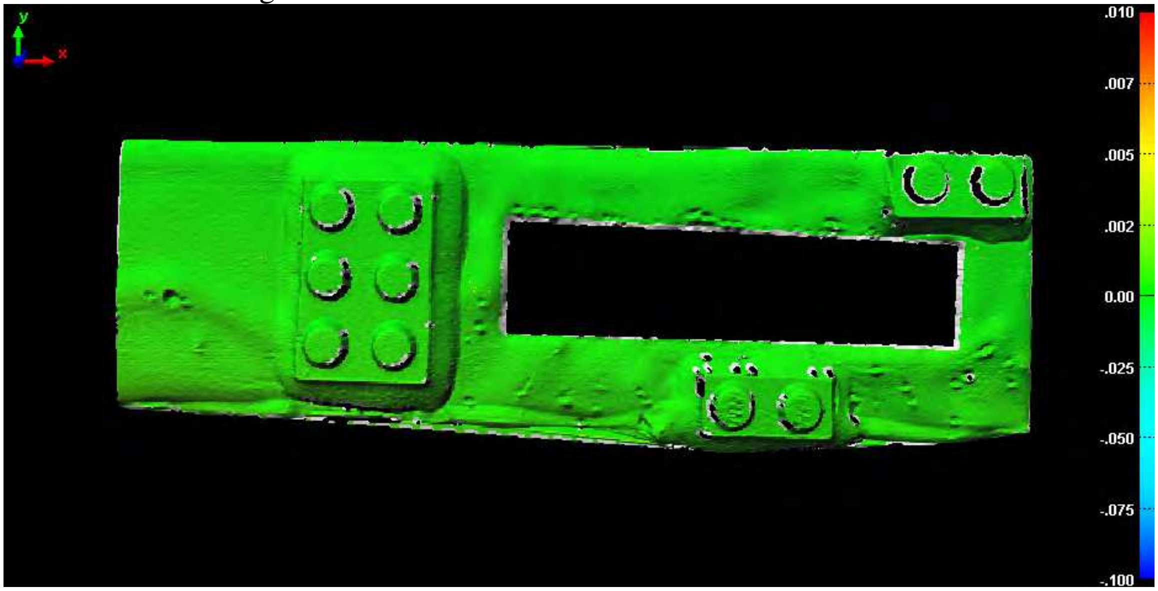
STEP 1. Original scanned image of Sample A is at the bottom colored in light blue and the corroded sample is at top colored in light green. (Note: The two images are not aligned)



STEP 2: The exposed areas are removed individually from original scan of Sample A and the corroded sample. (Note: The two images are not aligned)

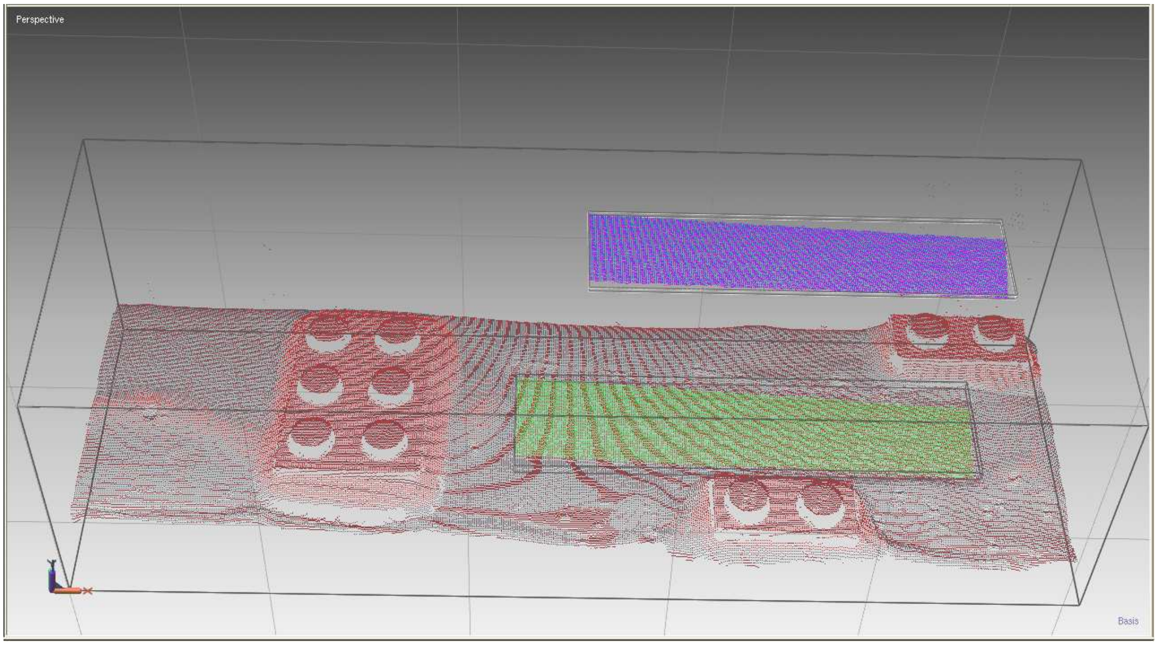


STEP 3: Align the two images with the exposed area removed. The color code shows the delta of the two images

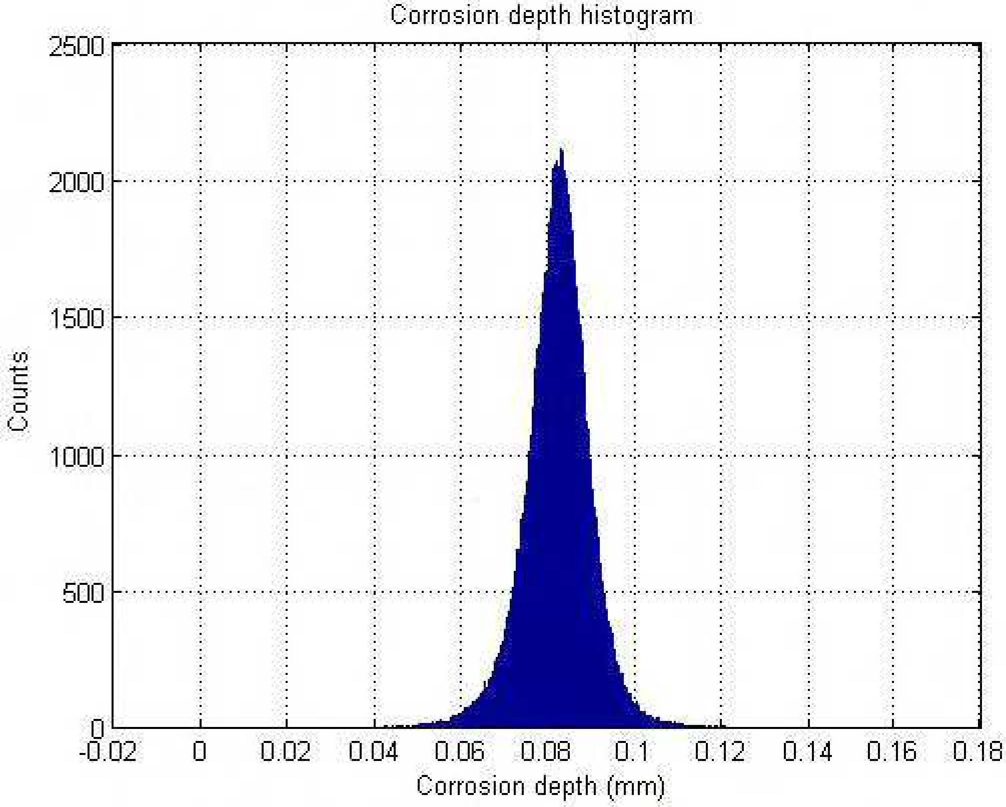


STEP 4: The corroded sample was aligned with the original Sample A using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

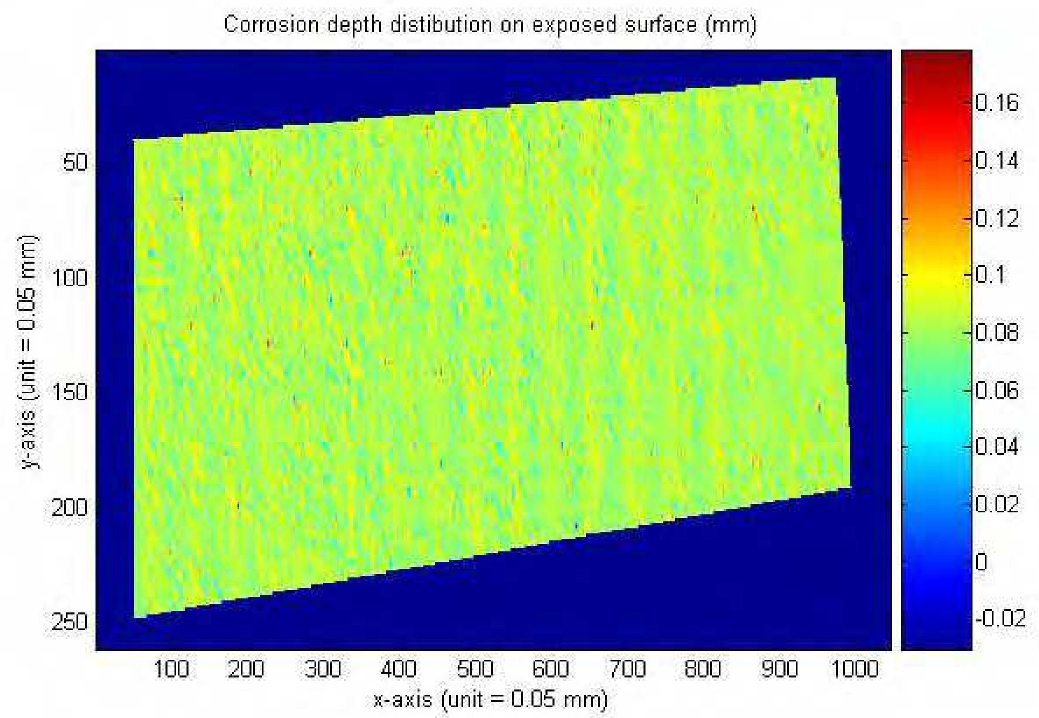
1. original image of Sample A (shown in black)
2. aligned image of corroded Sample A (shown in red)
3. the exposed surface of Sample A in its original position (shown in green)
4. the orientated exposed surface of Sample A with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample A orientated accordingly (shown in pink, unfortunately, it is only 0.08mm below and is not clearly visible)



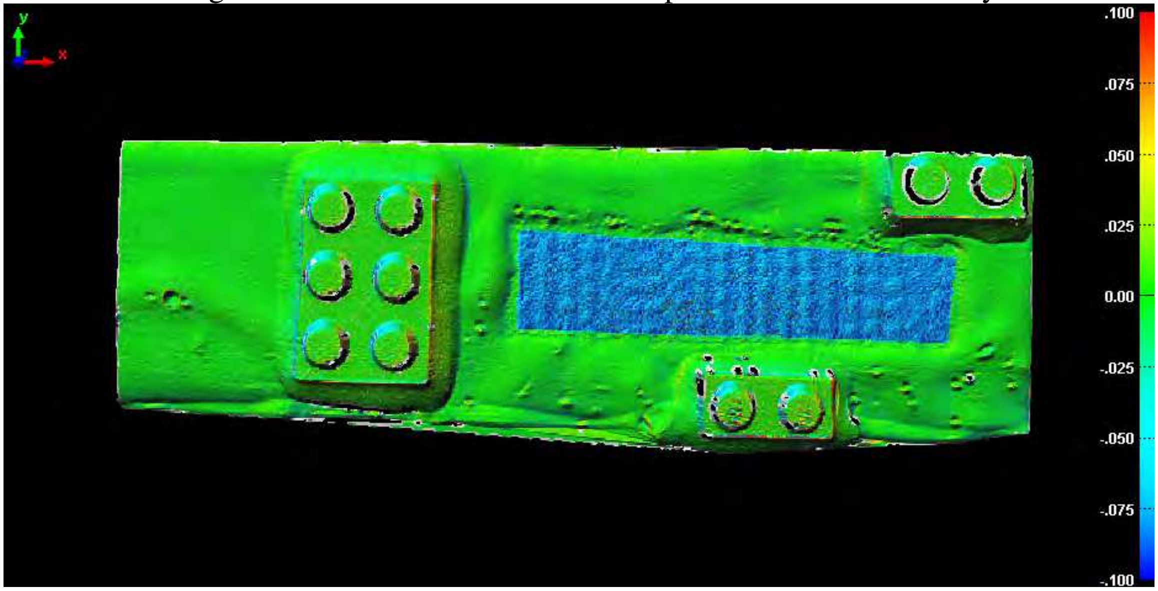
The histogram of the corrosion depth on the corrosion surface:

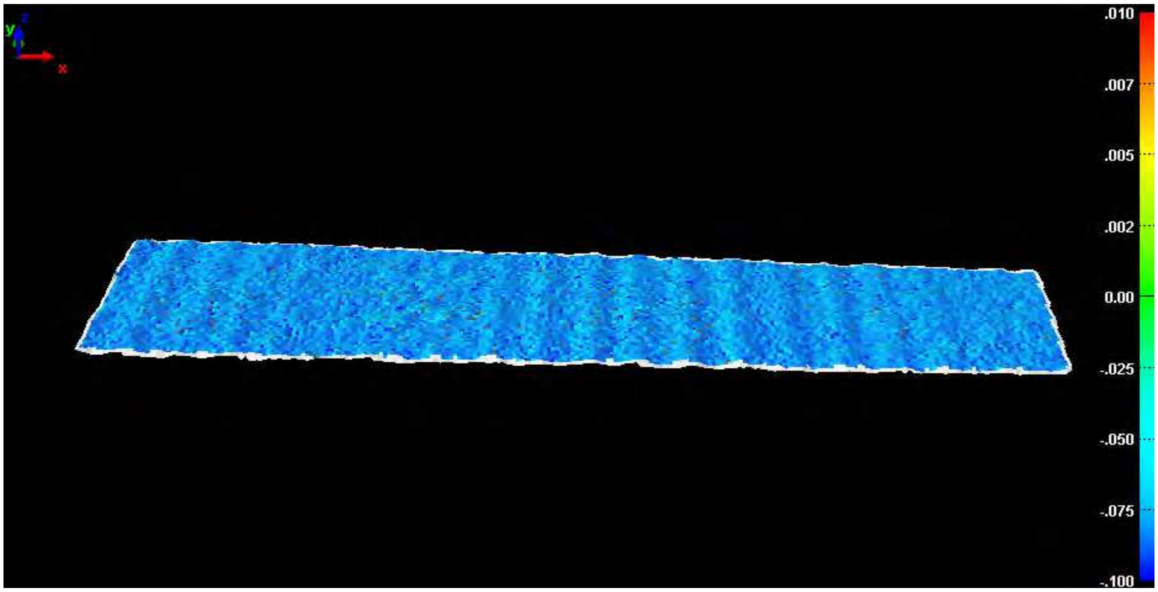


The 2D distribution of the corrosion depth on the corrosion surface:



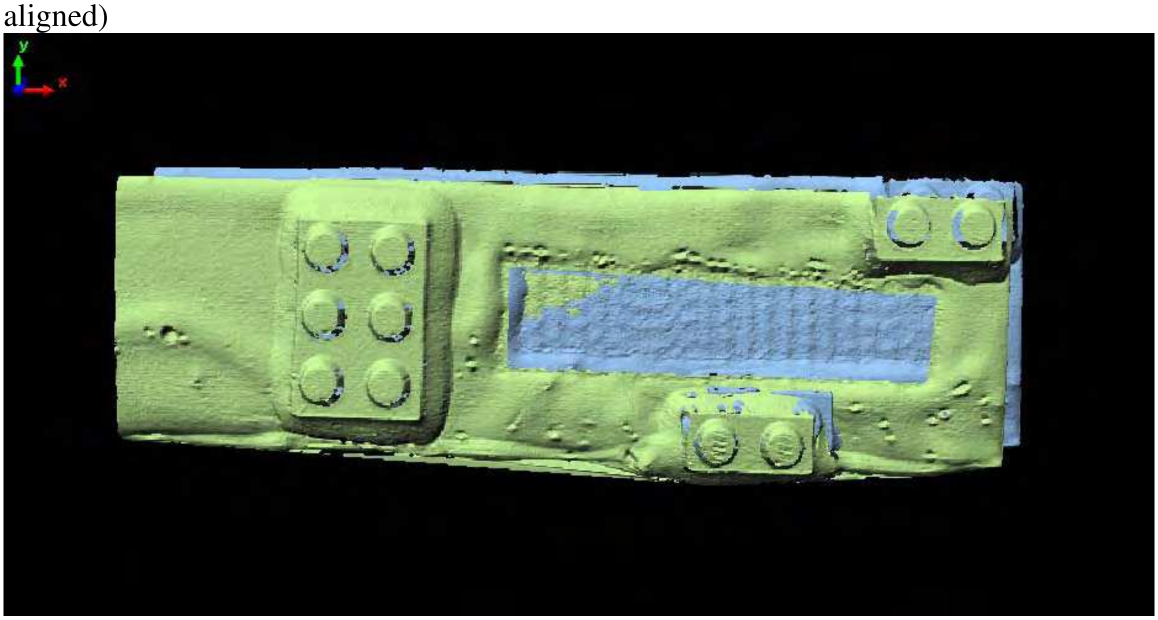
STEP 5: The alignment results and the corrosion depth distribution from PolyWorks

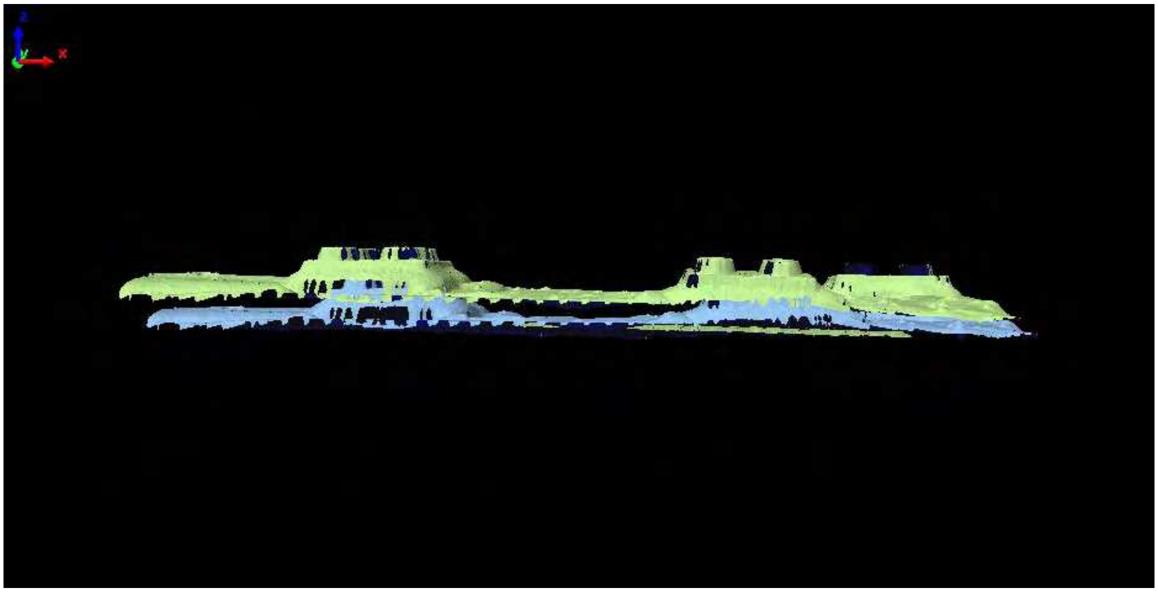




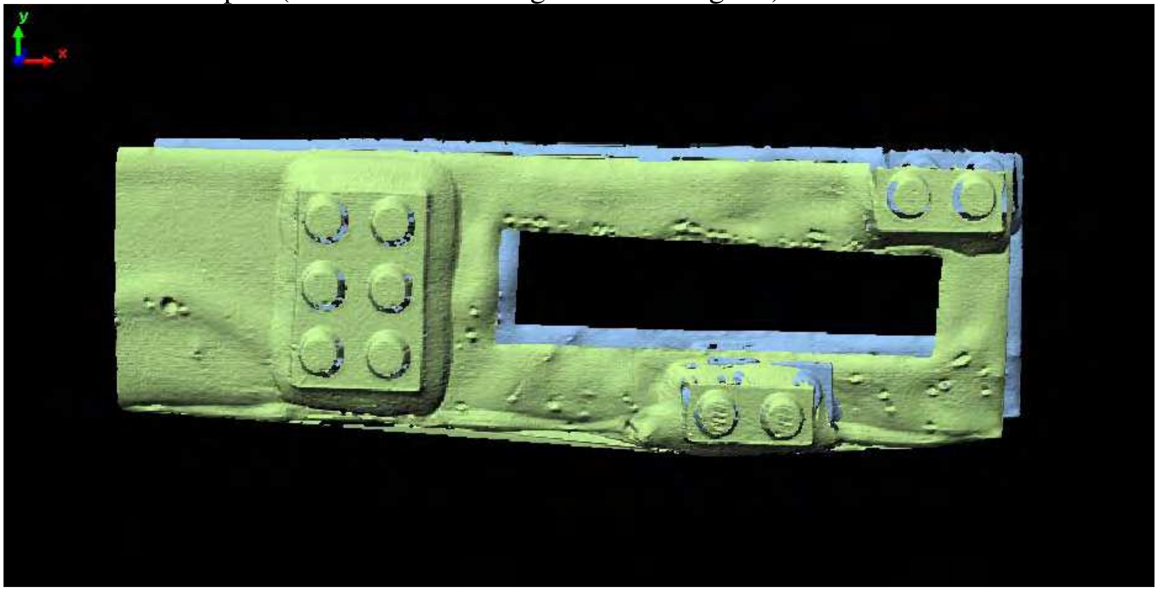
Test Results for the 4 mm Process

STEP 1. Original scanned image of Sample A is at the bottom, colored in light blue and the corroded sample is at the front, colored in light green. (Note: The two images are not aligned)

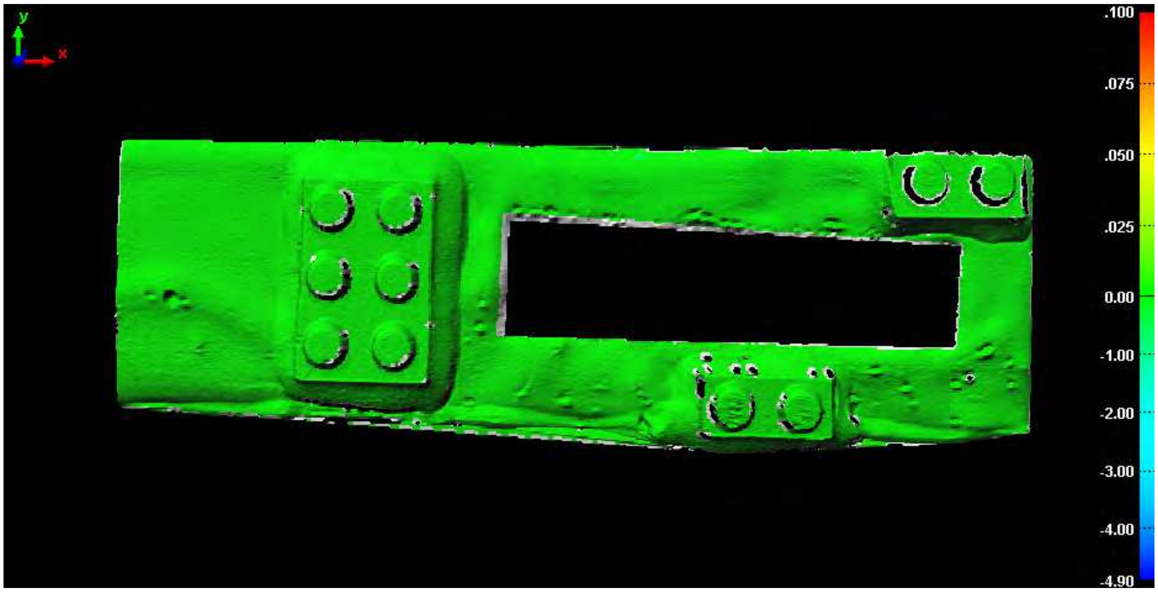




STEP 2: The exposed areas are removed individually from original scan of Sample A and the corroded sample. (Note: The two images are not aligned)

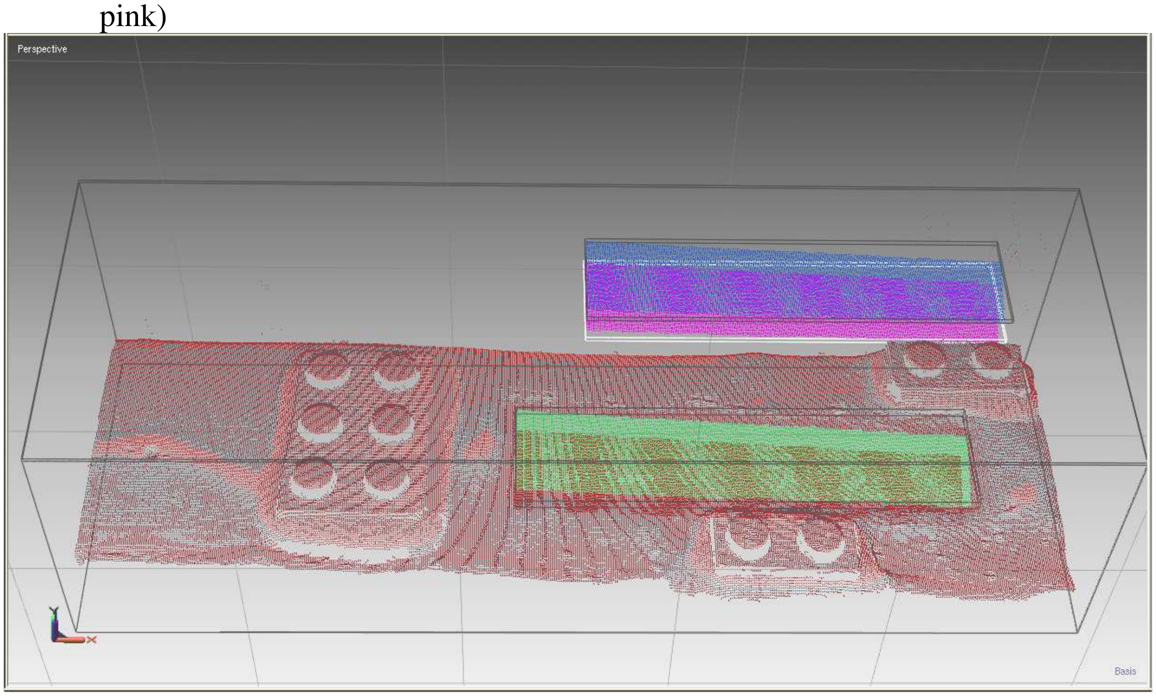


STEP 3: Align the two images with the exposed area removed. The color code shows the delta of the two images

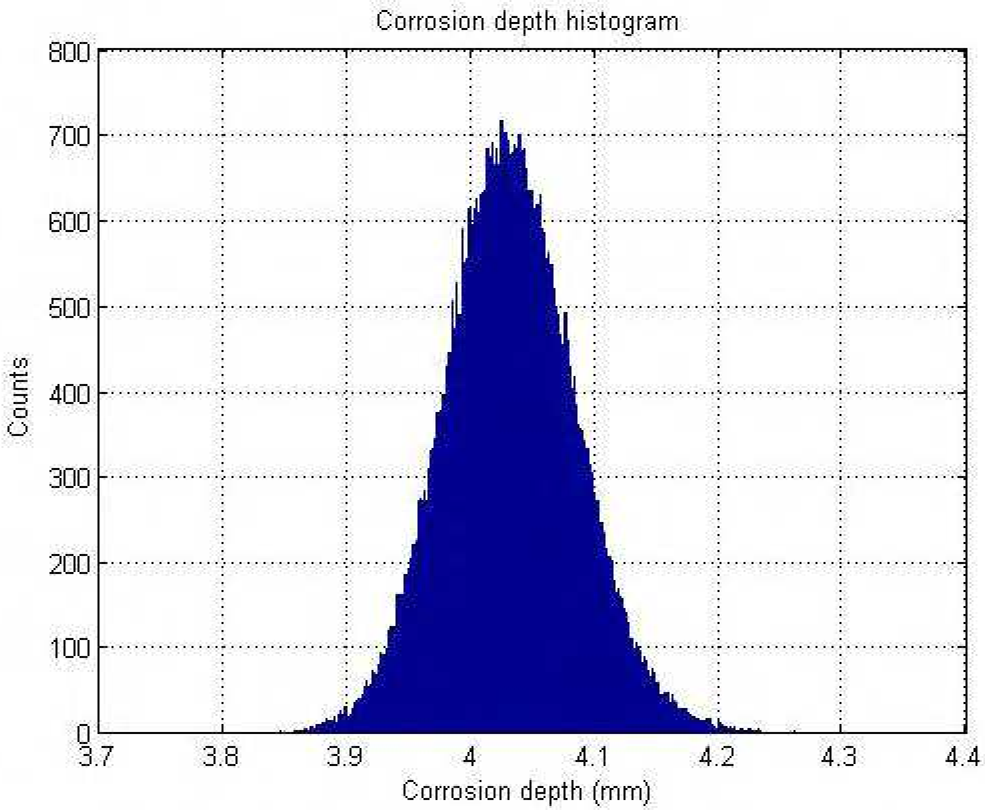


STEP 4: The corroded sample was aligned with the original Sample A using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

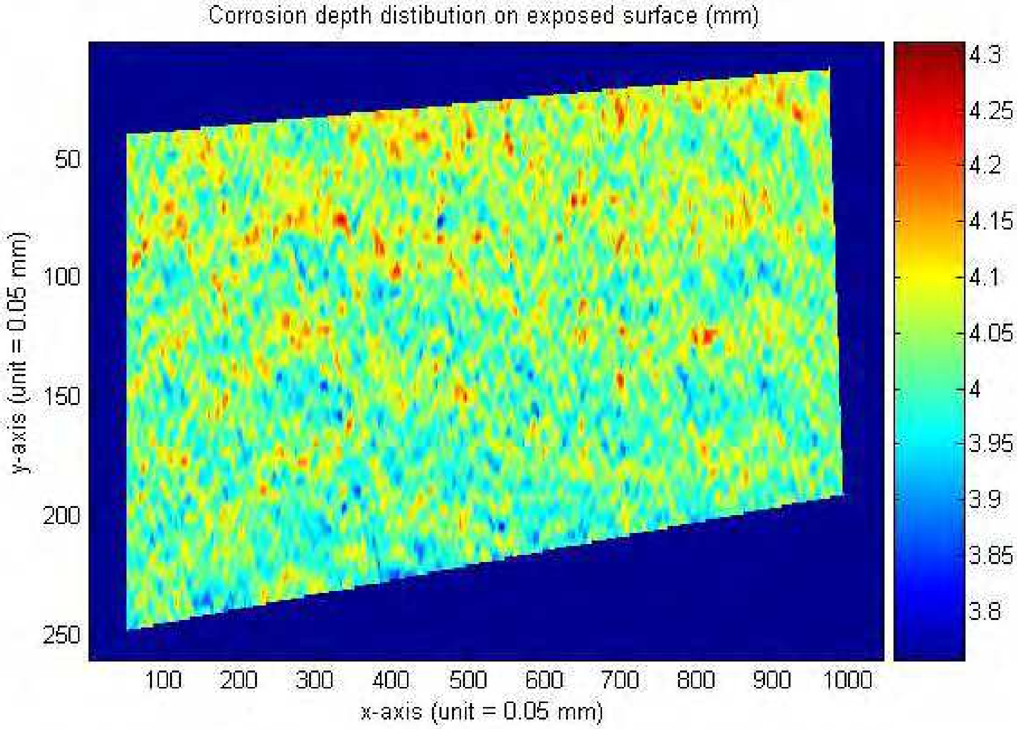
1. original image of Sample A (shown in black)
2. aligned image of corroded Sample A (shown in red)
3. the exposed surface of Sample A in its original position (shown in green)
4. the orientated exposed surface of Sample A with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample A orientated accordingly (shown in pink)



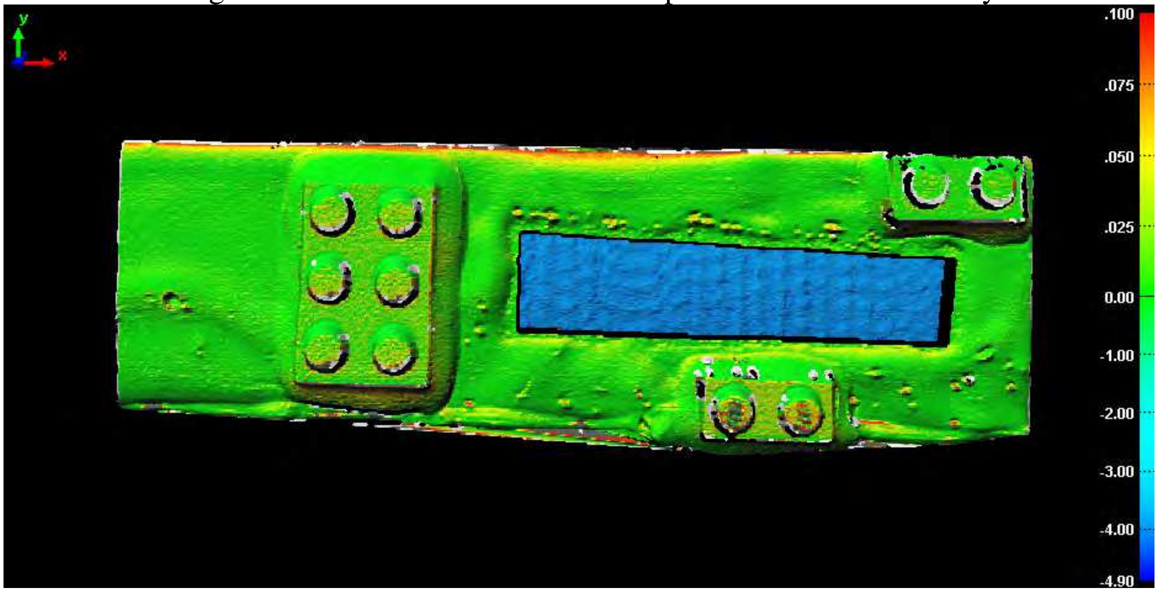
The histogram of the corrosion depth on the corrosion surface:

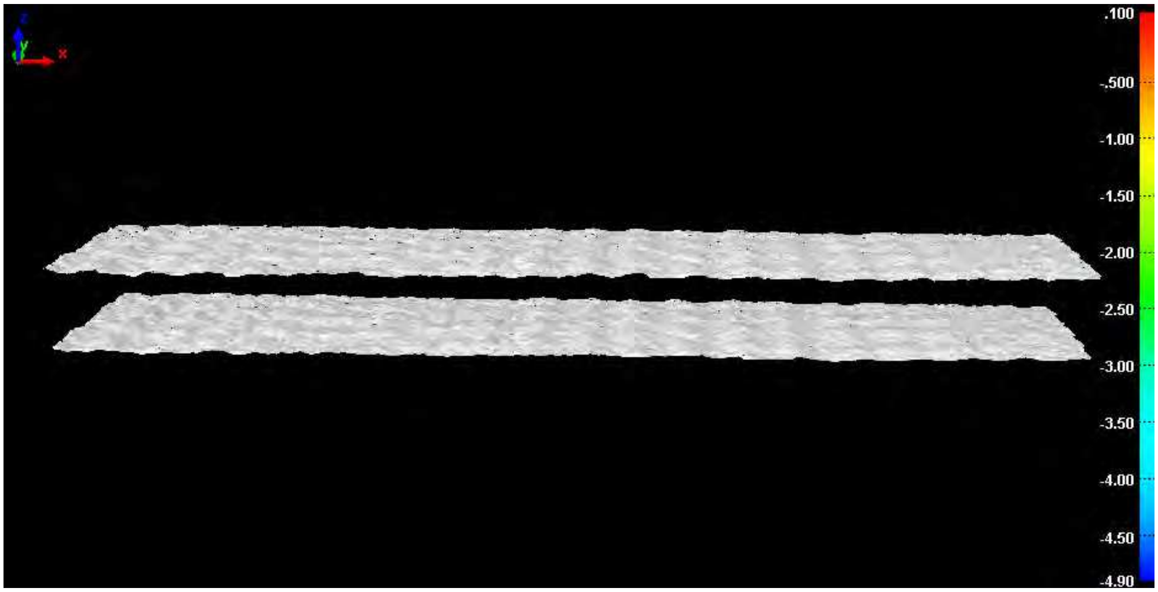


The 2D distribution of the corrosion depth on the corrosion surface:



STEP 5: The alignment results and the corrosion depth distribution from PolyWorks

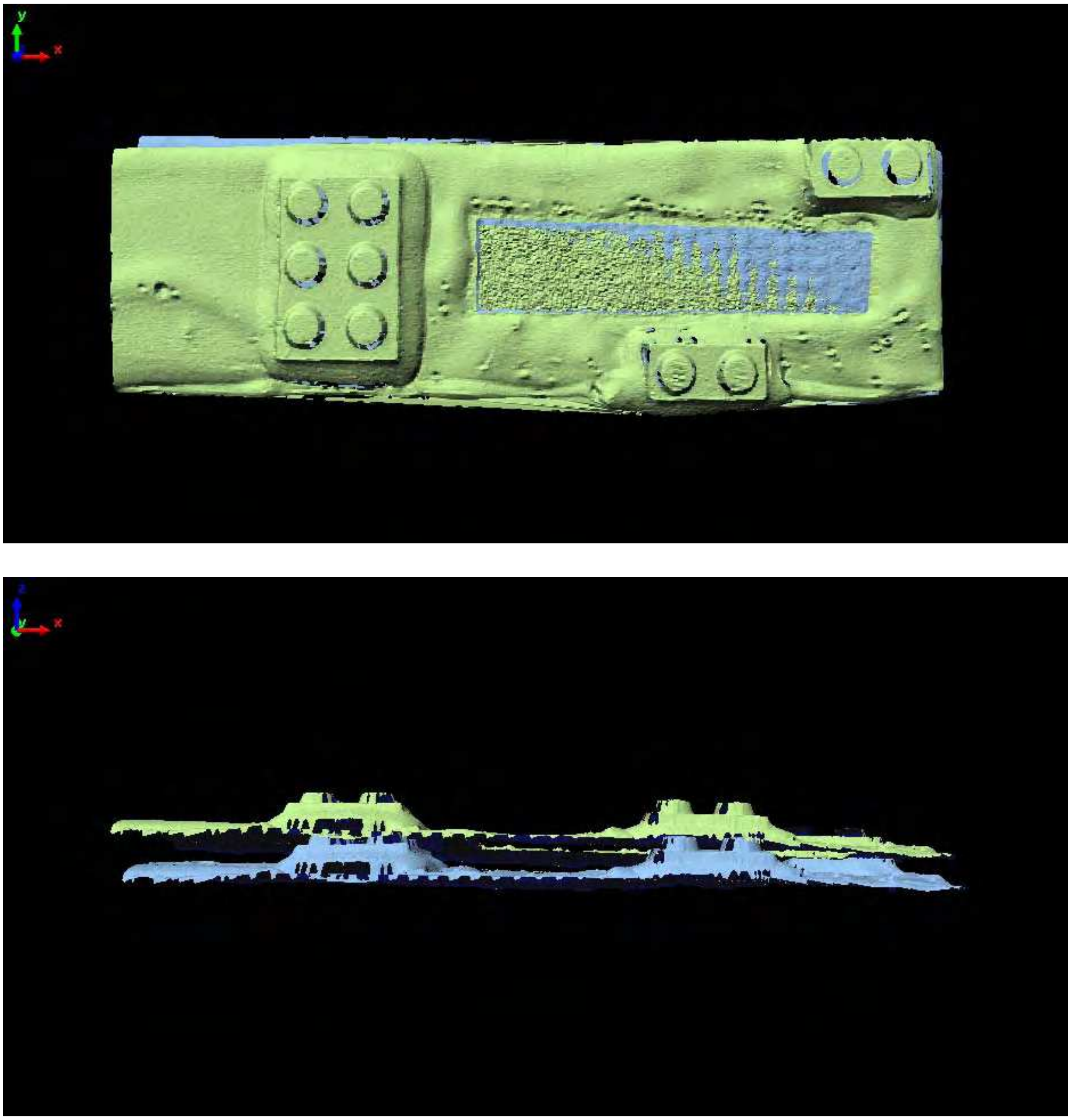




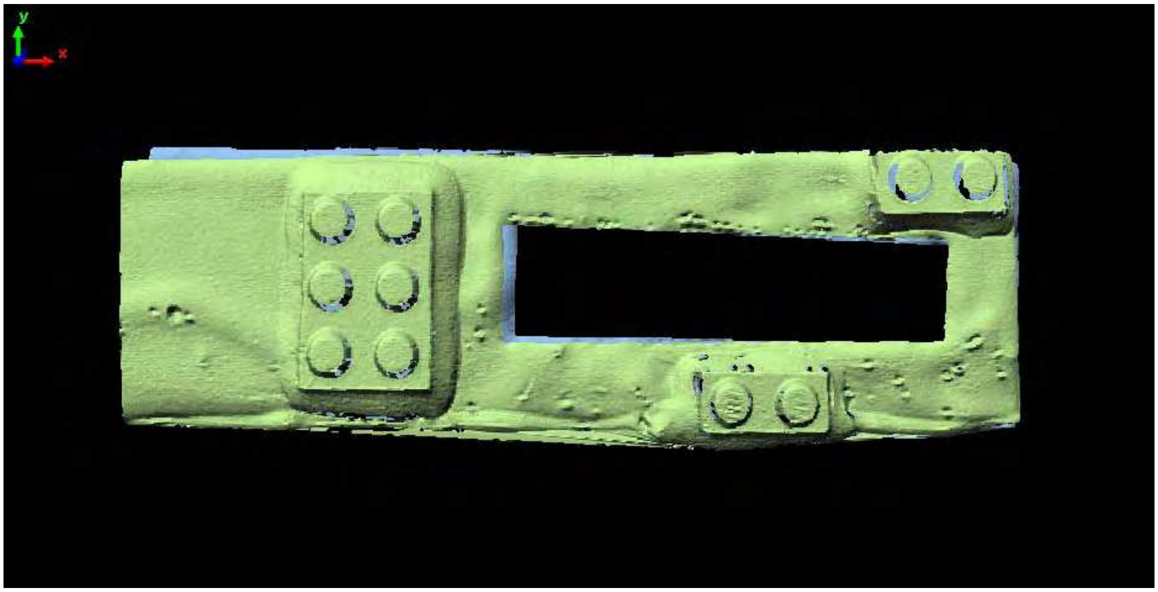
(Note: PolyWorks cannot create the delta of the two surfaces)

Test Results for the 4 mm with 5% noise

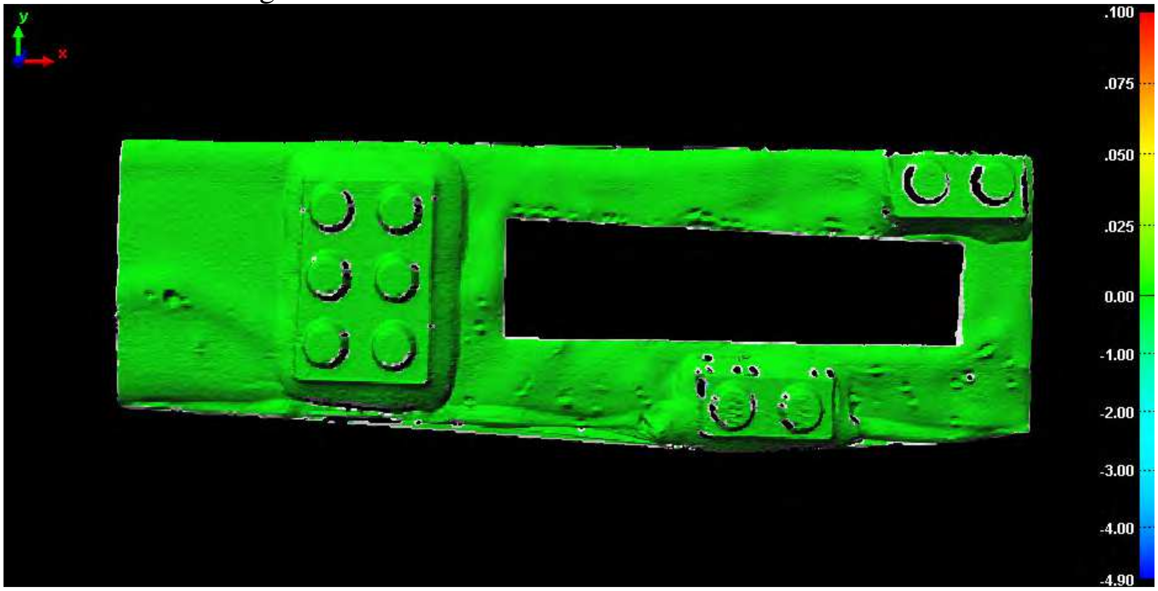
STEP 1. Original scanned image of Sample A is at the bottom colored in light blue and the corroded sample is at front colored in light green. (Note: The two images are not aligned)



STEP 2: The exposed areas are removed individually from original scan of Sample A and the corroded sample. (Note: The two images are not aligned)

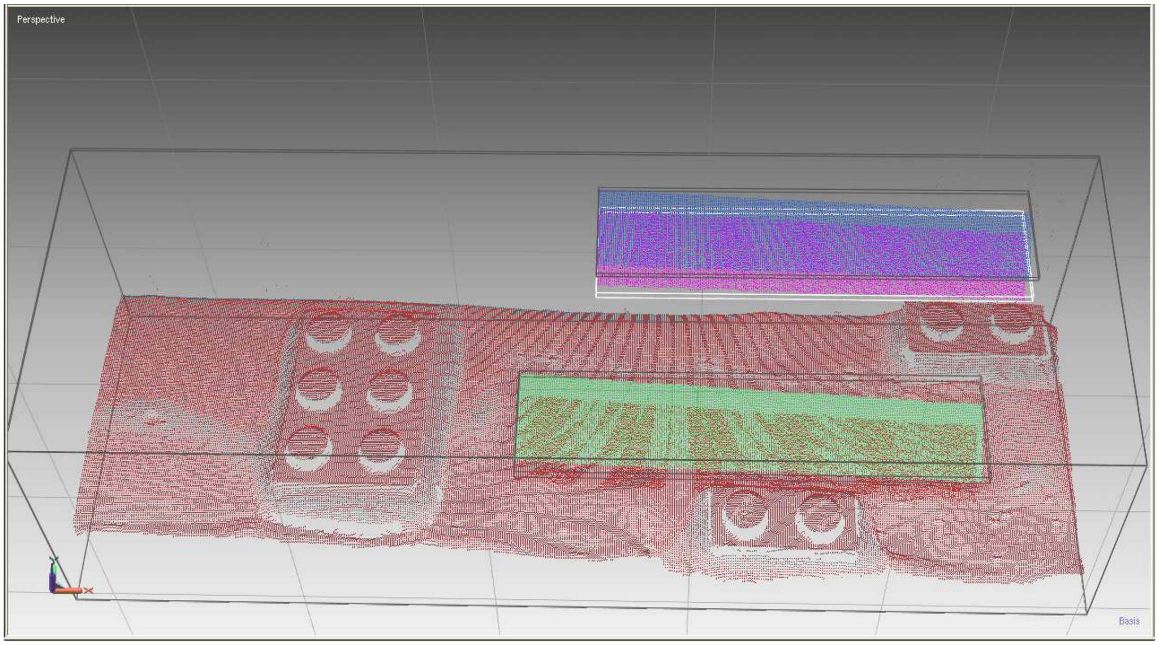


STEP 3: Align the two images with exposed area removed. The color code shows the delta of the two images

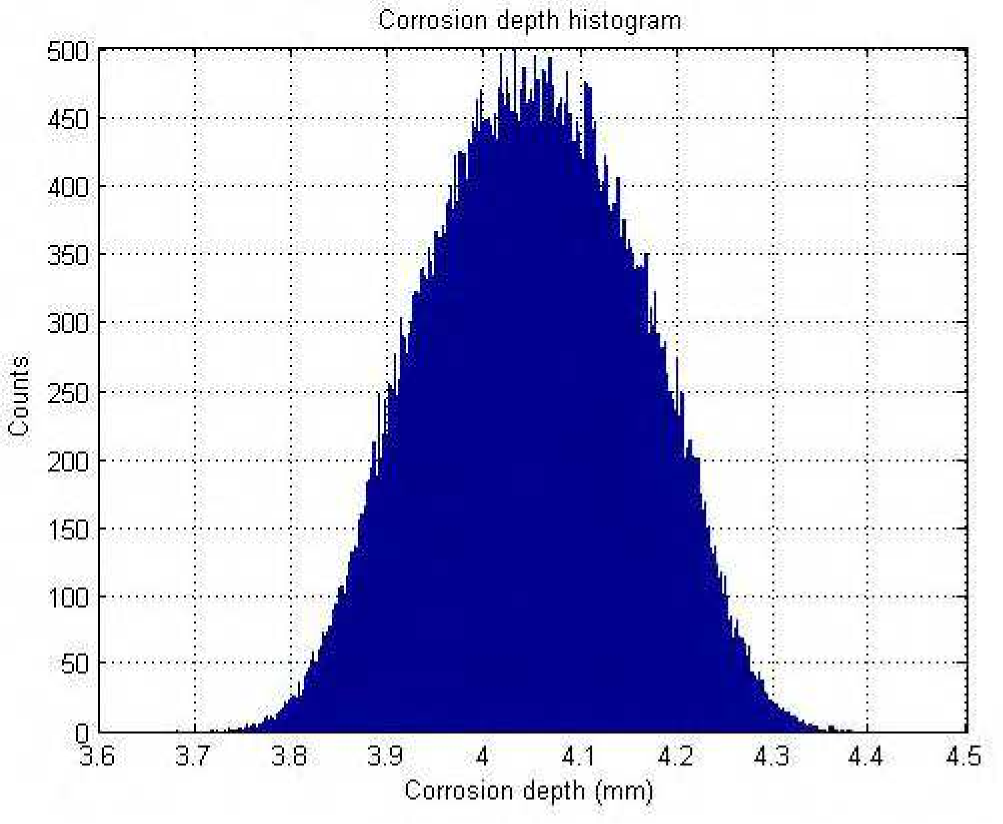


STEP 4: The corroded sample was aligned with the original Sample A using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

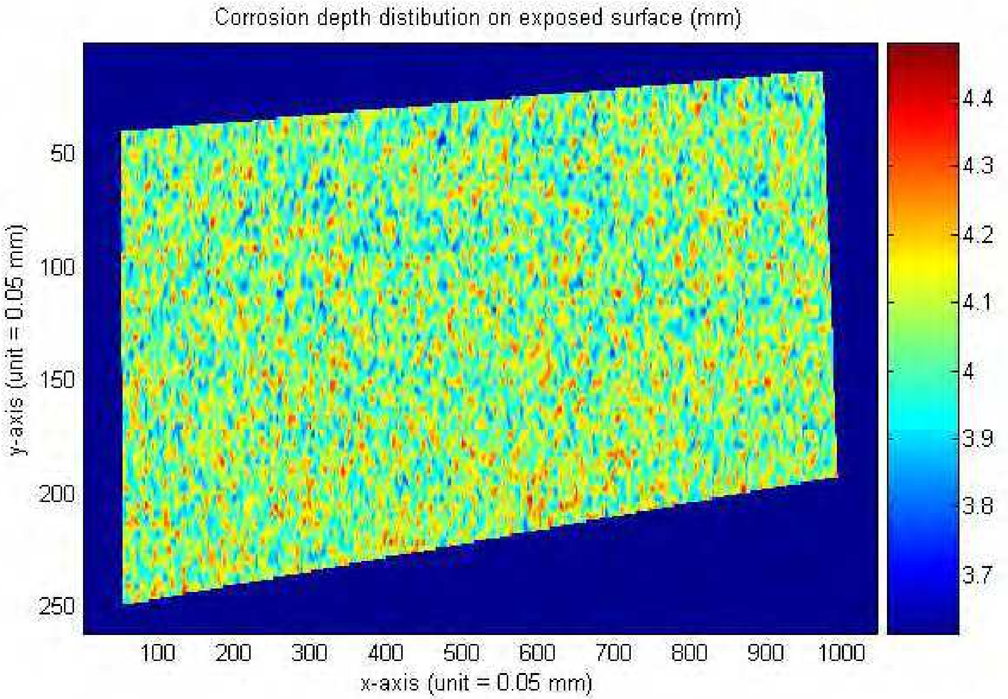
1. original image of Sample A (shown in black)
2. aligned image of corroded Sample A (shown in red)
3. the exposed surface of Sample A in its original position (shown in green)
4. the orientated exposed surface of Sample A with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample A orientated accordingly (shown in pink)



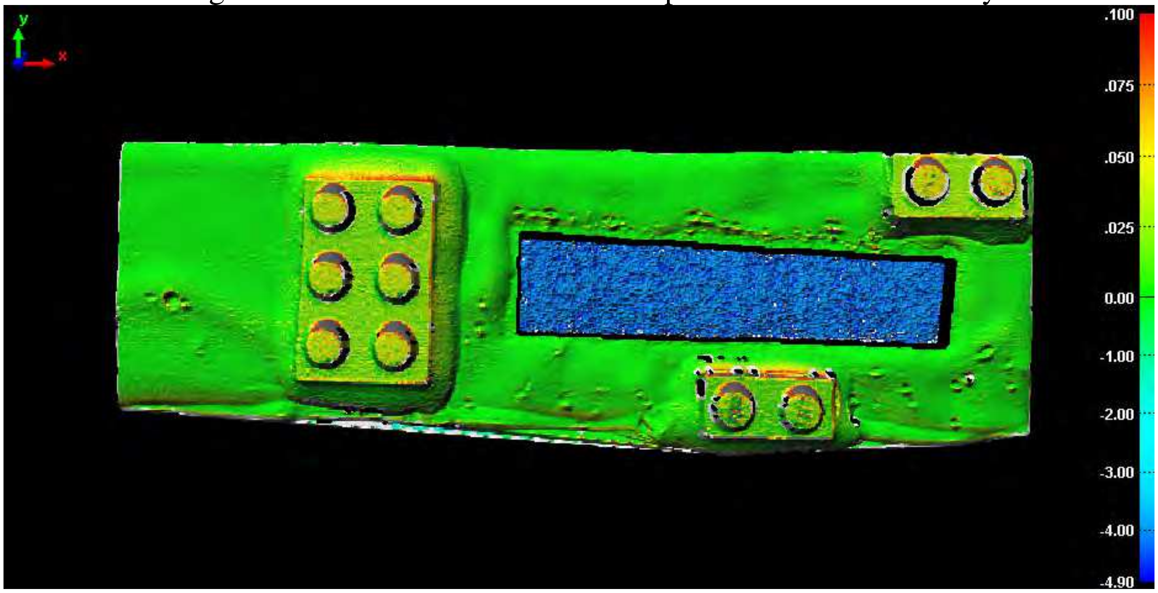
The histogram of the corrosion depth on the corrosion surface:

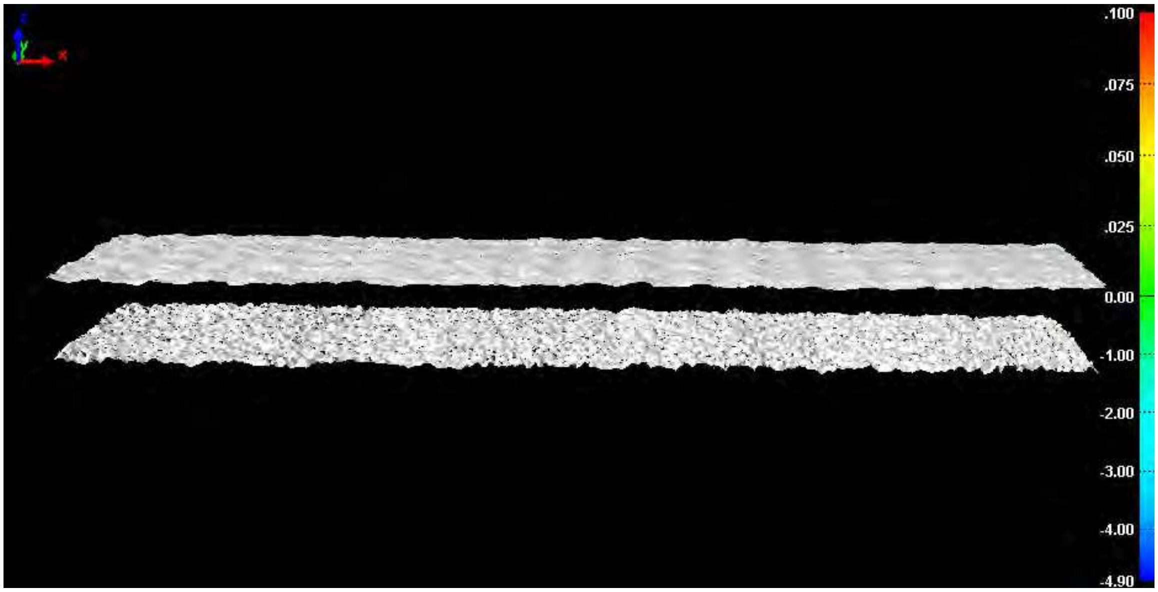


The 2D distribution of the corrosion depth on the corrosion surface:



STEP 5: The alignment results and the corrosion depth distribution from PolyWorks

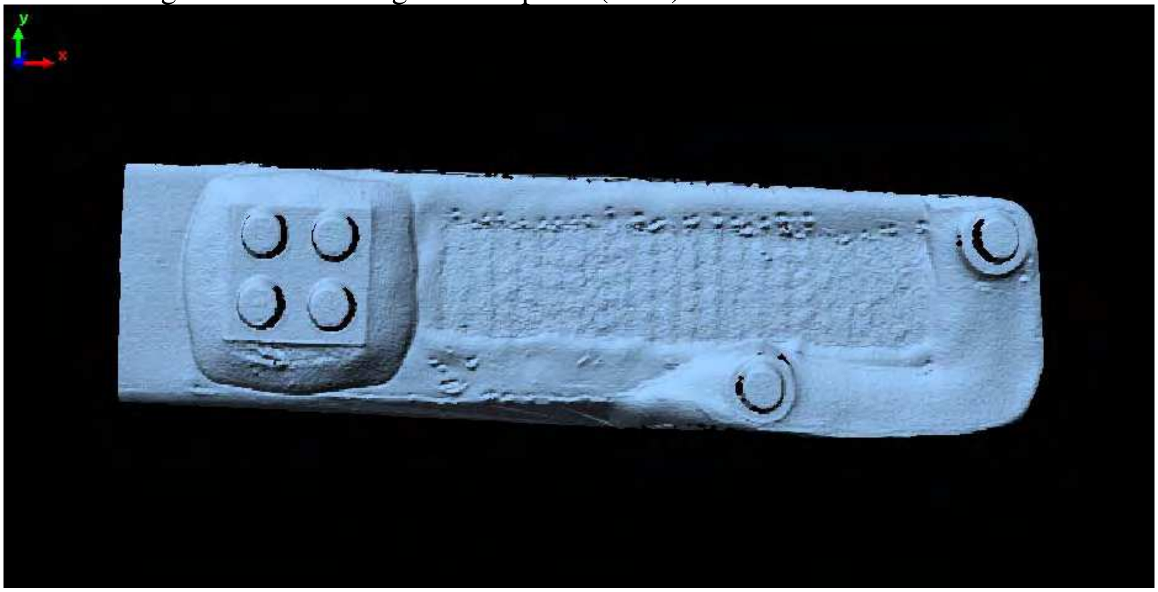




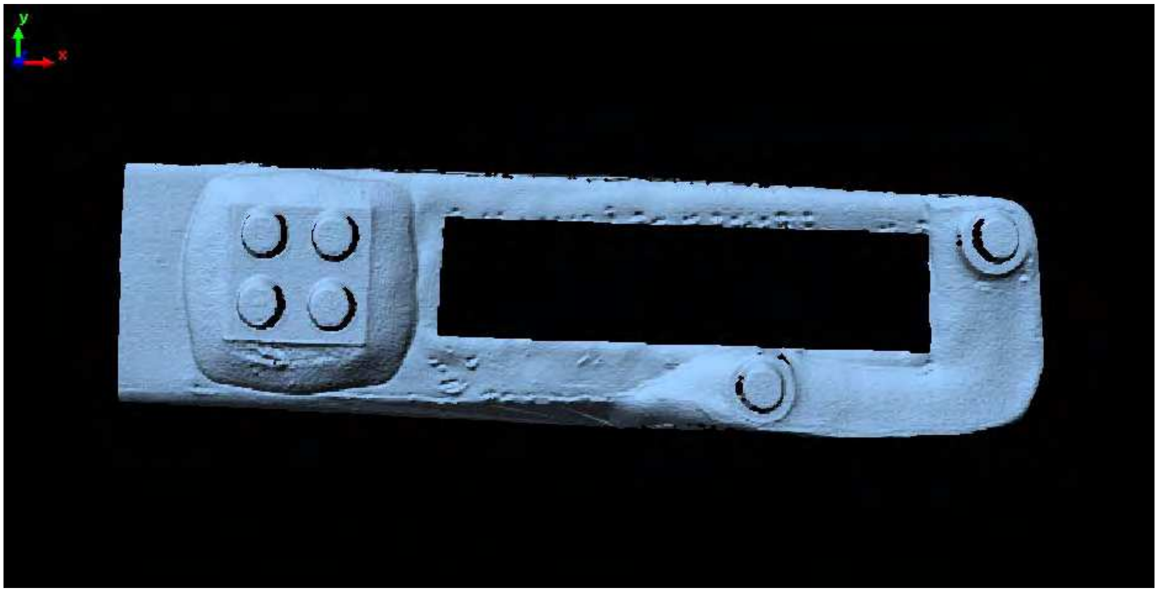
(Note: PolyWorks cannot create the delta of the two surfaces)

C.3.2 Sample B

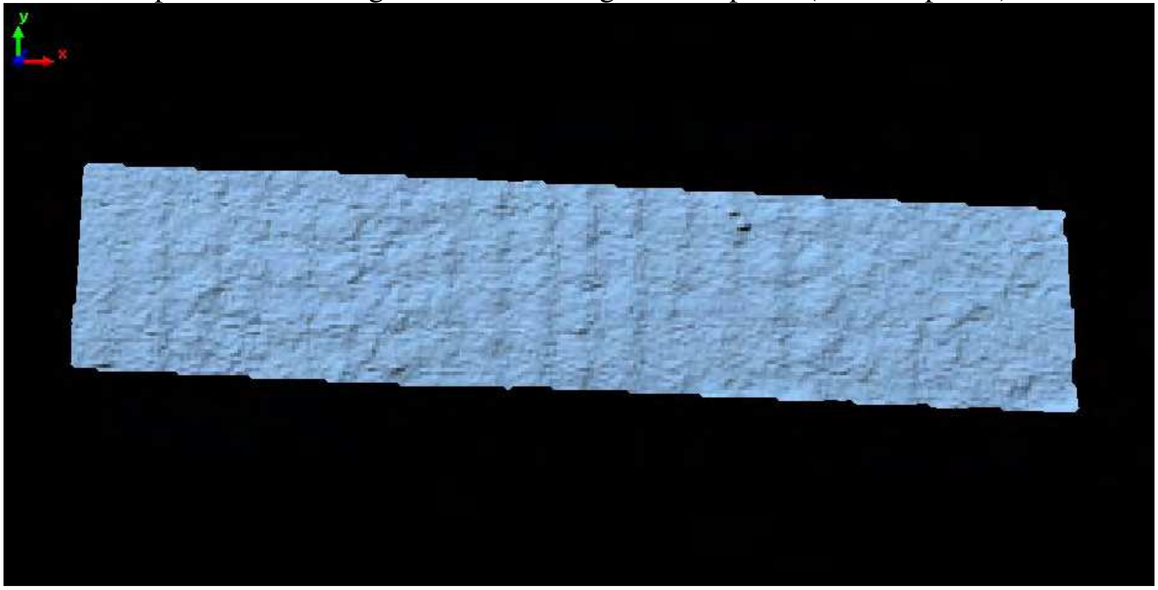
STEP 1. Original scanned image of Sample B (IMO)



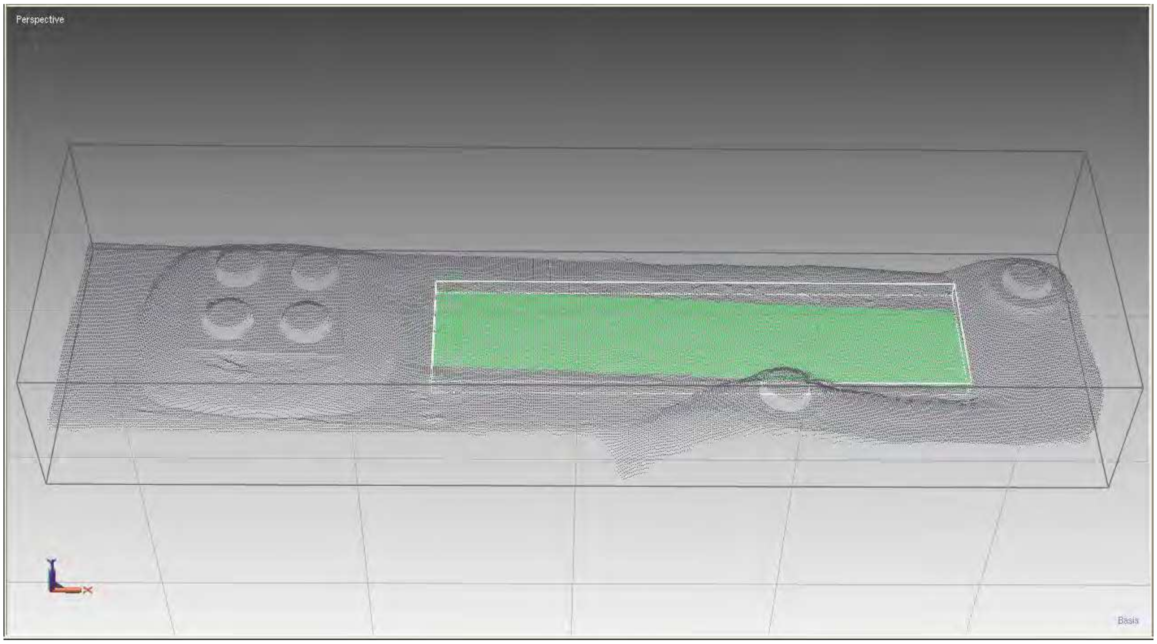
STEP 2: Original scanned image of Sample B with exposed area cut off (IMO_hole)



STEP 2: Exposed area of original scanned image of Sample B (IMO_exposed)

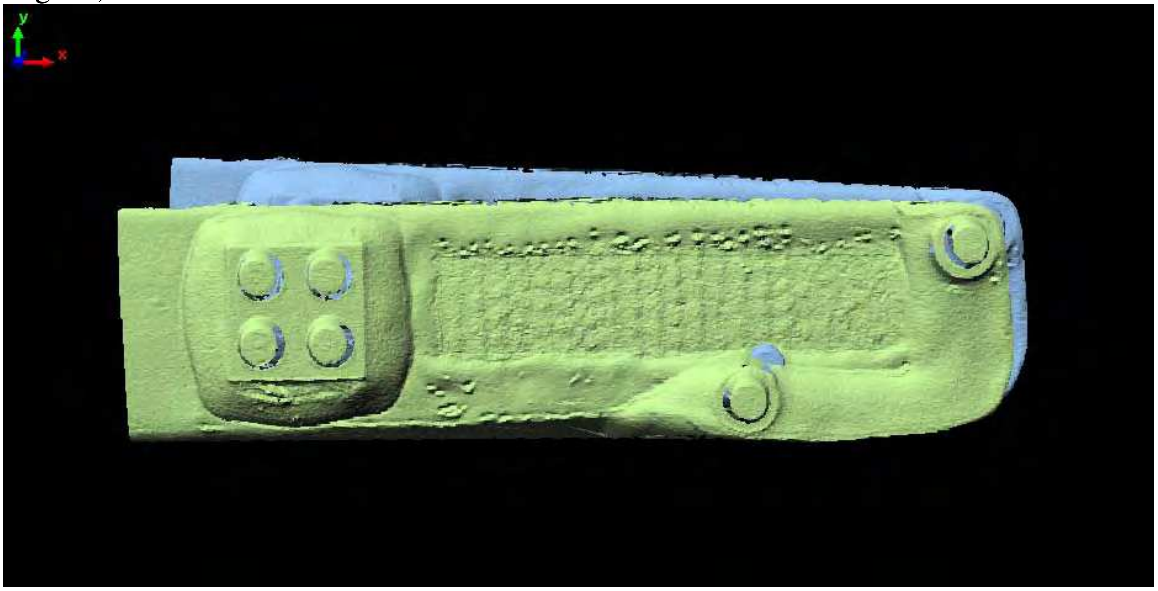


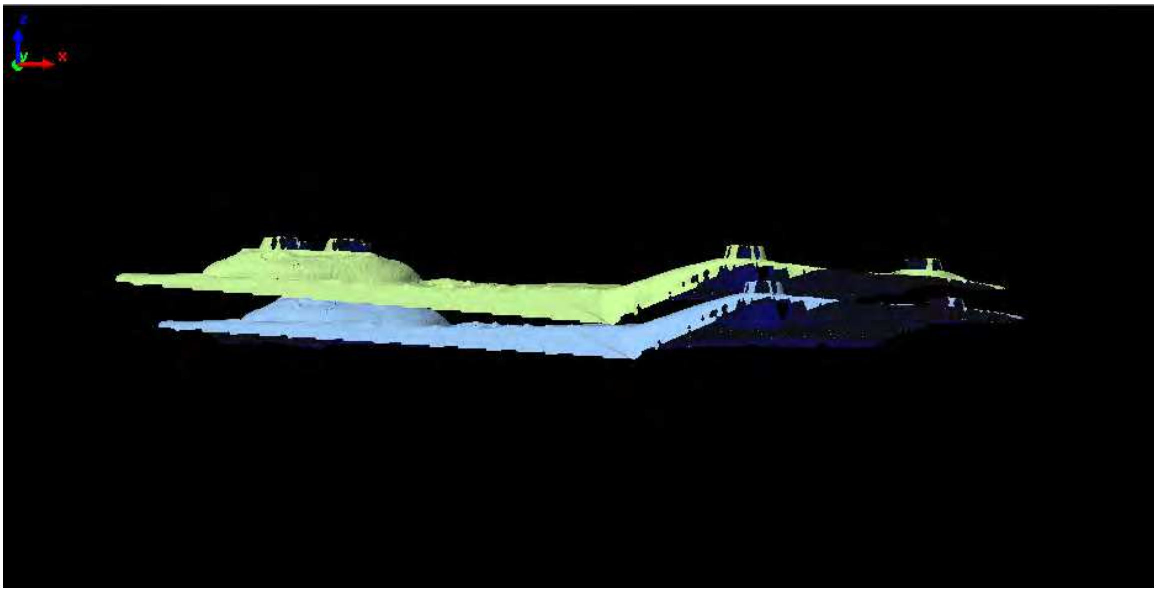
Exposed surface of Sample B (shown in green):



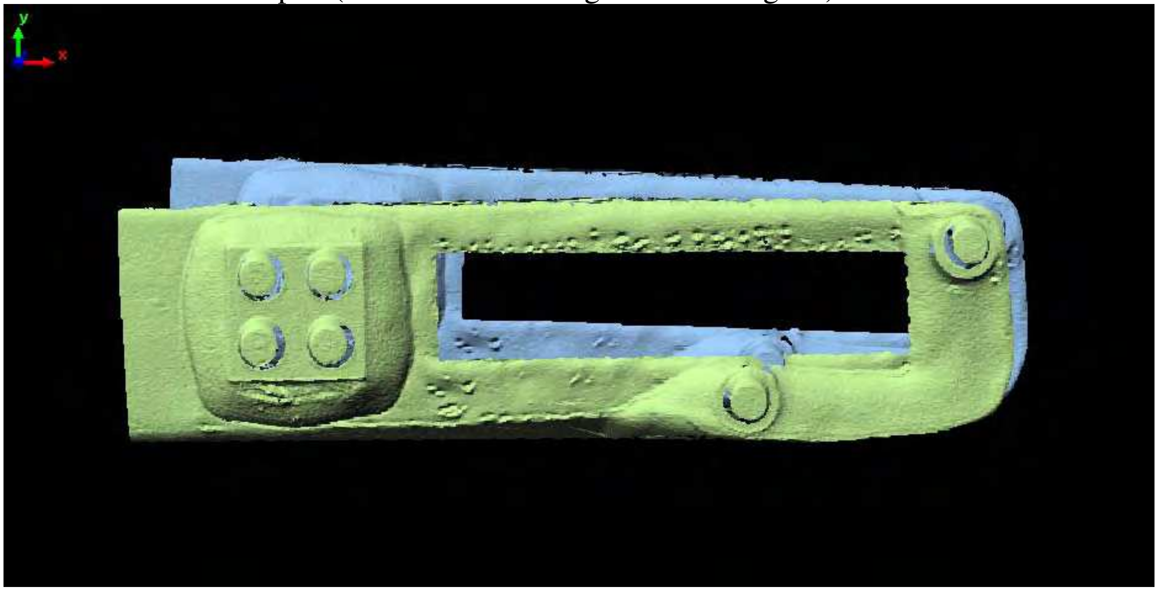
Test Results for the 0.08 mm Process

STEP 1. Original scanned image of Sample B is at the bottom, colored in light blue and the corroded sample is at the top, colored in light green. (Note: The two images are not aligned)

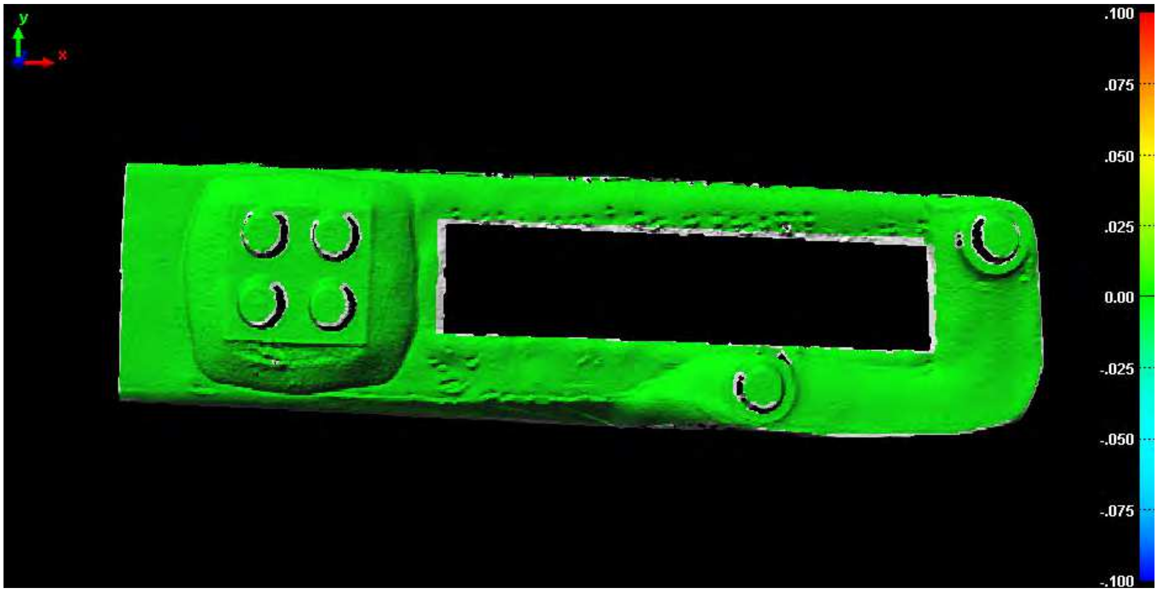




STEP 2: The exposed areas are removed individually from the original scan of Sample B and the corroded sample. (Note: The two images are not aligned)

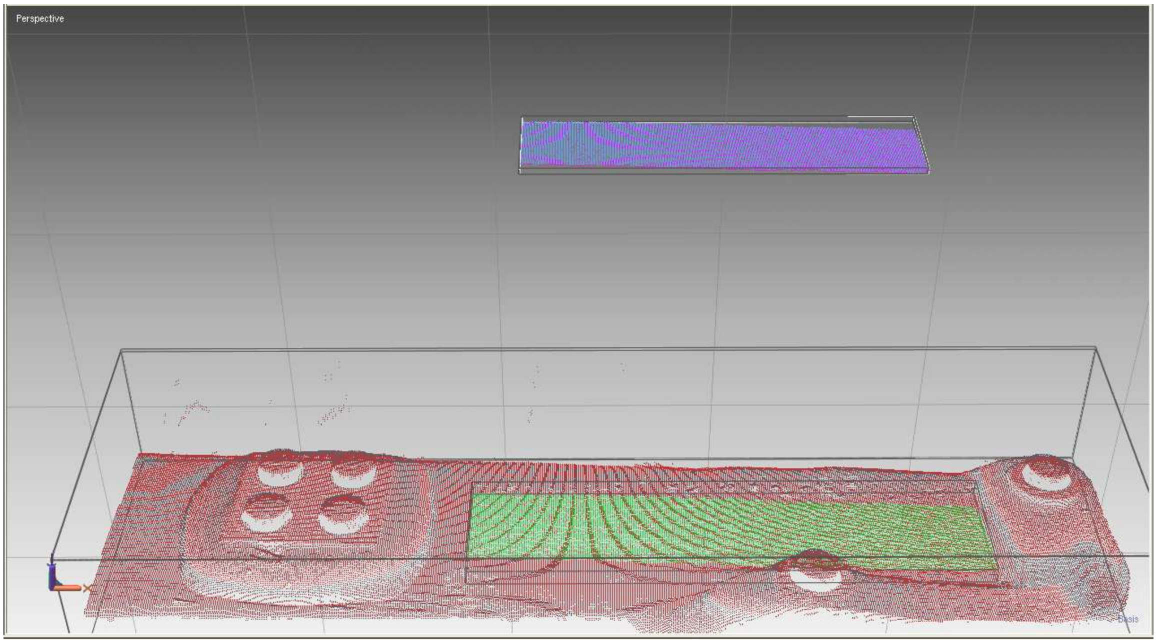


STEP 3: Align the two images with exposed area removed. The color code shows the delta of the two images

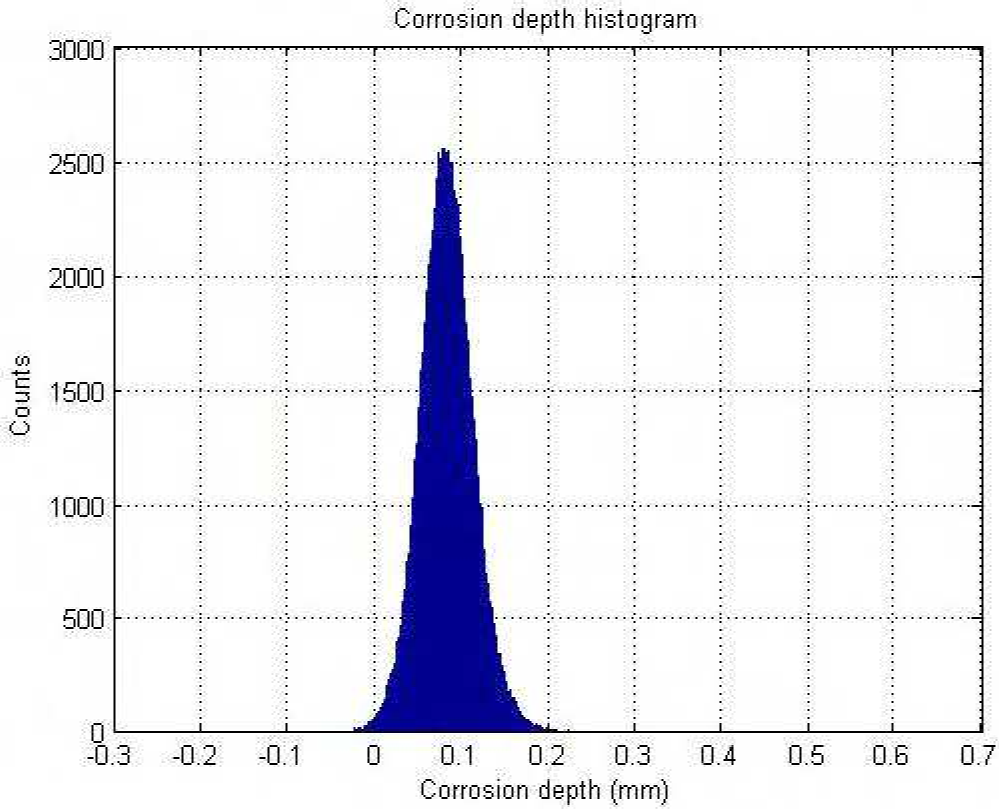


STEP 4: The corroded sample was aligned with the original Sample B using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

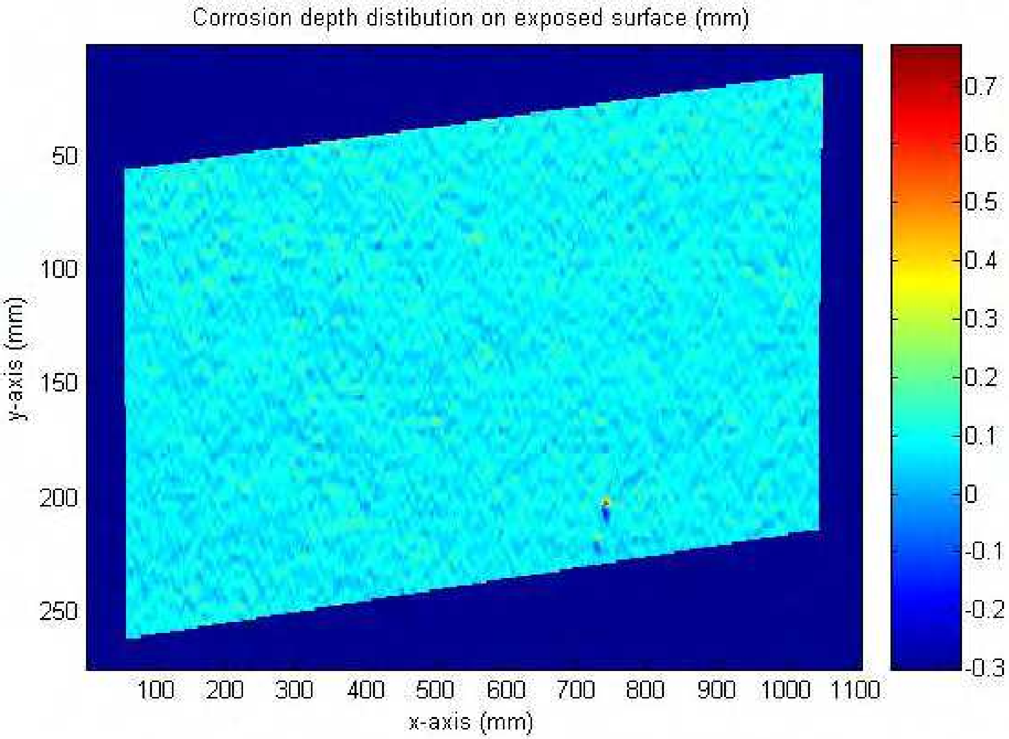
1. original image of Sample B (shown in black)
2. aligned image of corroded Sample B (shown in red)
3. the exposed surface of Sample B in its original position (shown in green)
4. the orientated exposed surface of Sample B with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample B orientated accordingly (shown in pink. Unfortunately, it is only 0.08mm below and is not clearly visible)



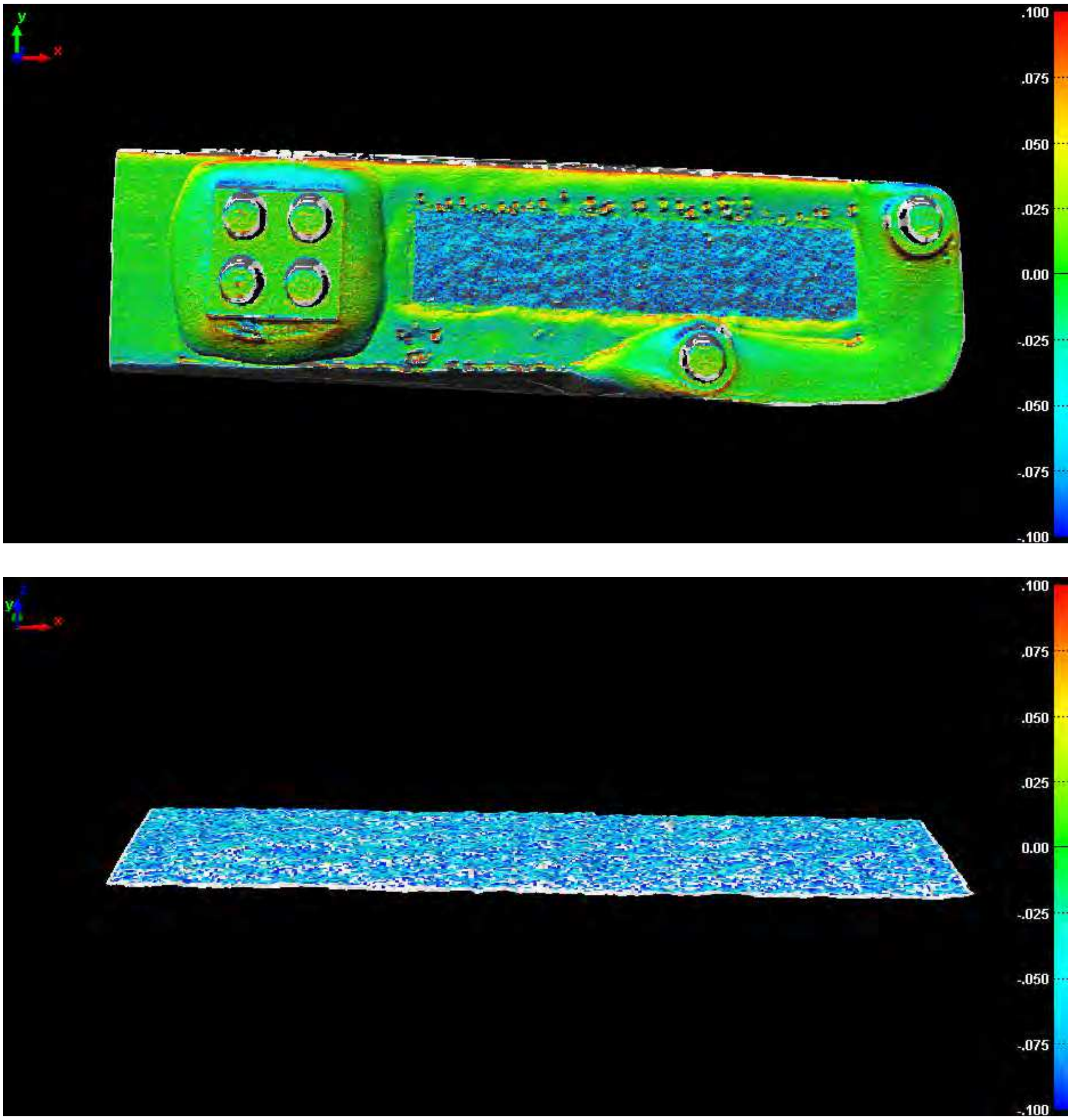
The histogram of the corrosion depth on the corrosion surface:



The 2D distribution of the corrosion depth on the corrosion surface:

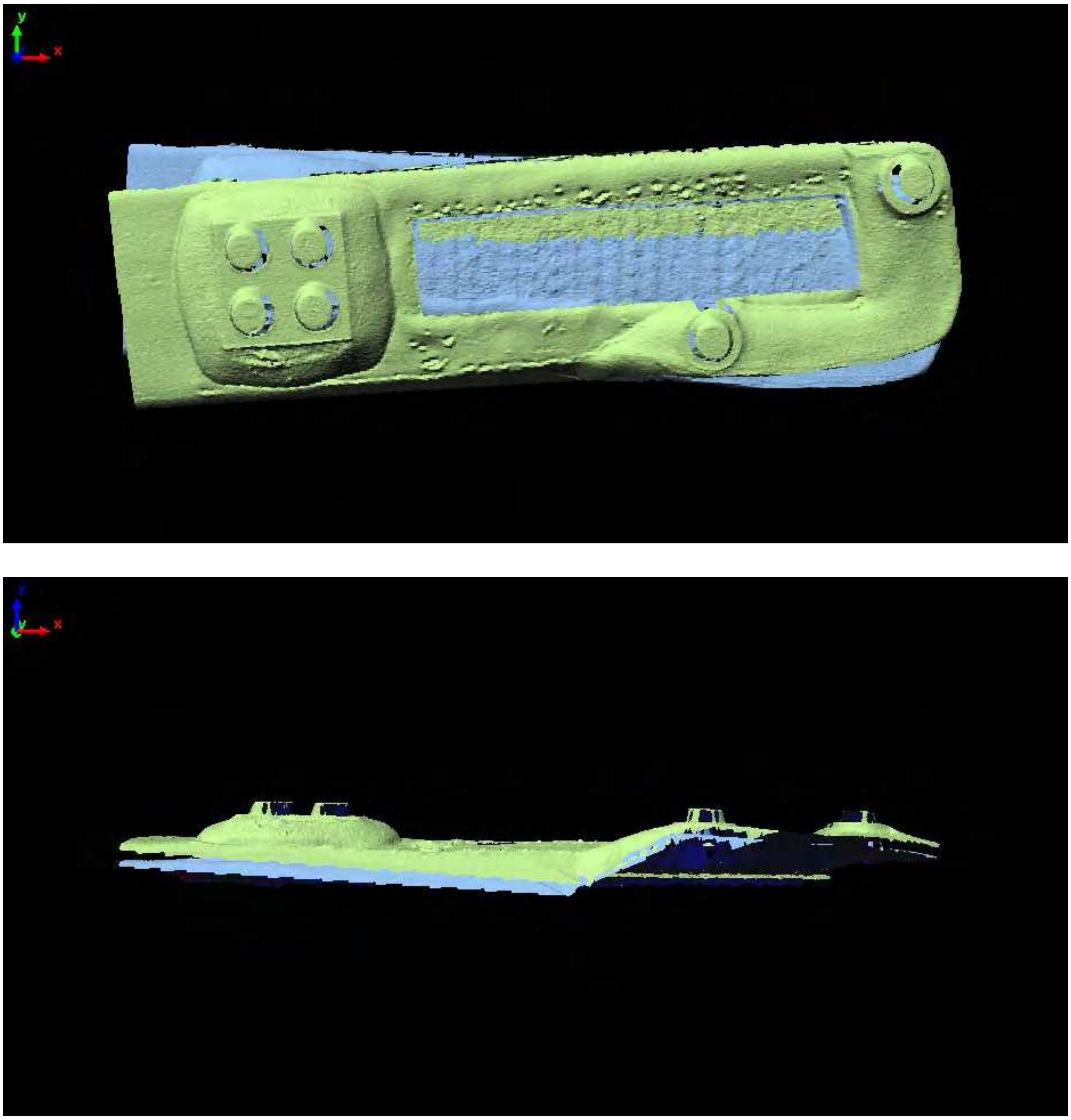


STEP 5: The alignment results and the corrosion depth distribution from PolyWorks

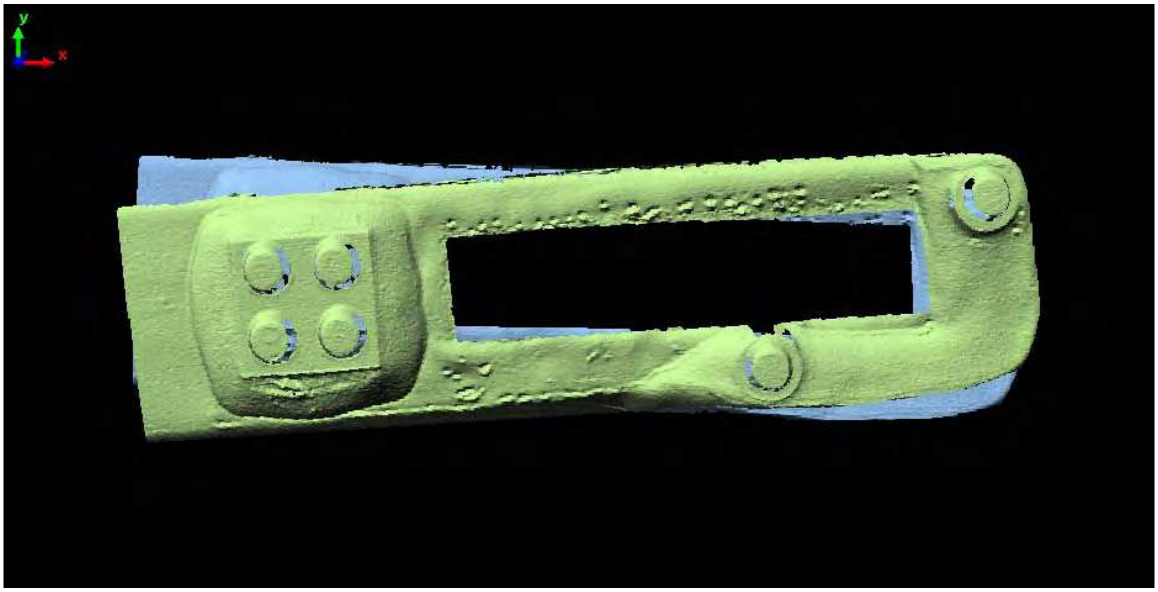


Test Results for the 4 mm Process

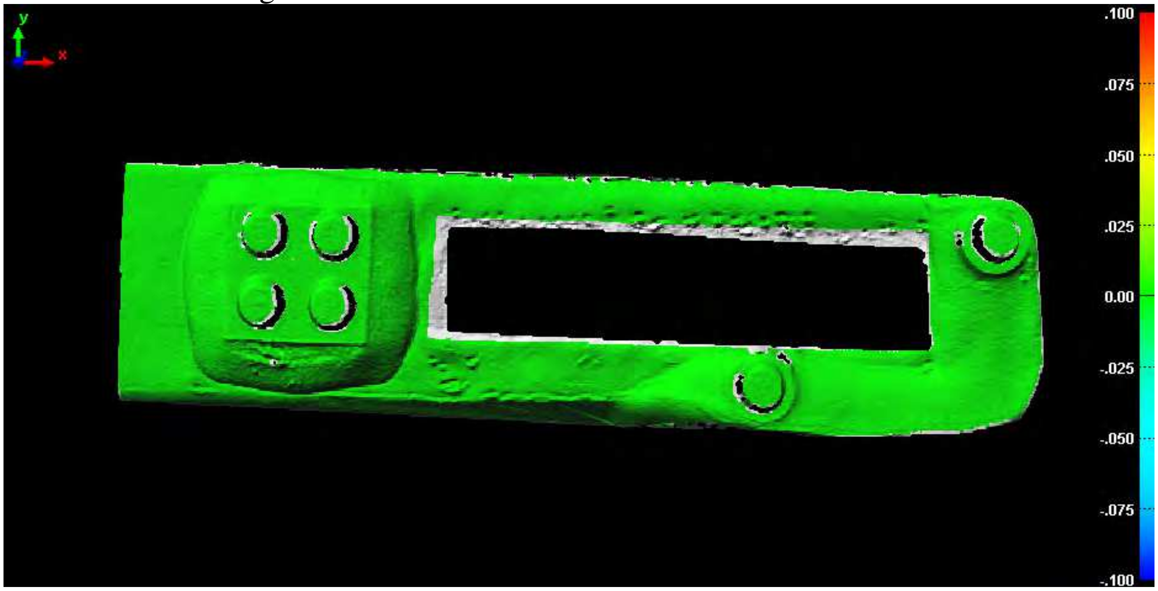
STEP 1. The original scanned image of Sample B is at the bottom, colored in light blue and the corroded sample is at the front colored, in light green. (Note: The two images are not aligned)



STEP 2: The exposed areas are removed individually from the original scan of Sample B and the corroded sample. (Note: The two images are not aligned)

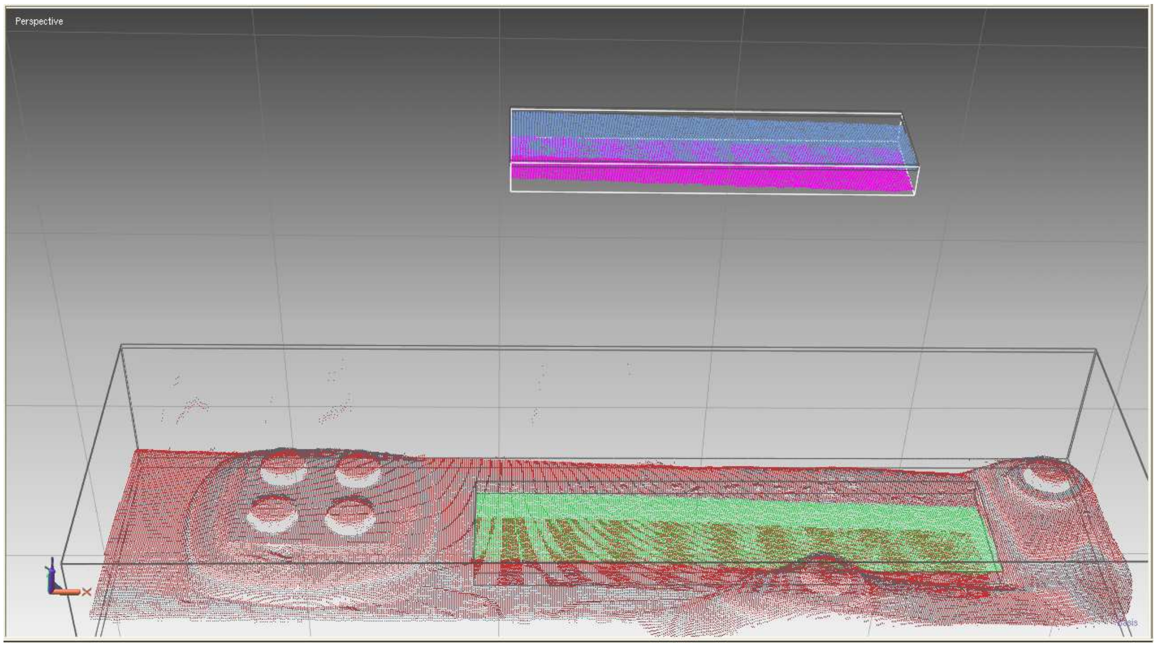


STEP 3: Align the two images with the exposed area removed. The color code shows the delta of the two images.

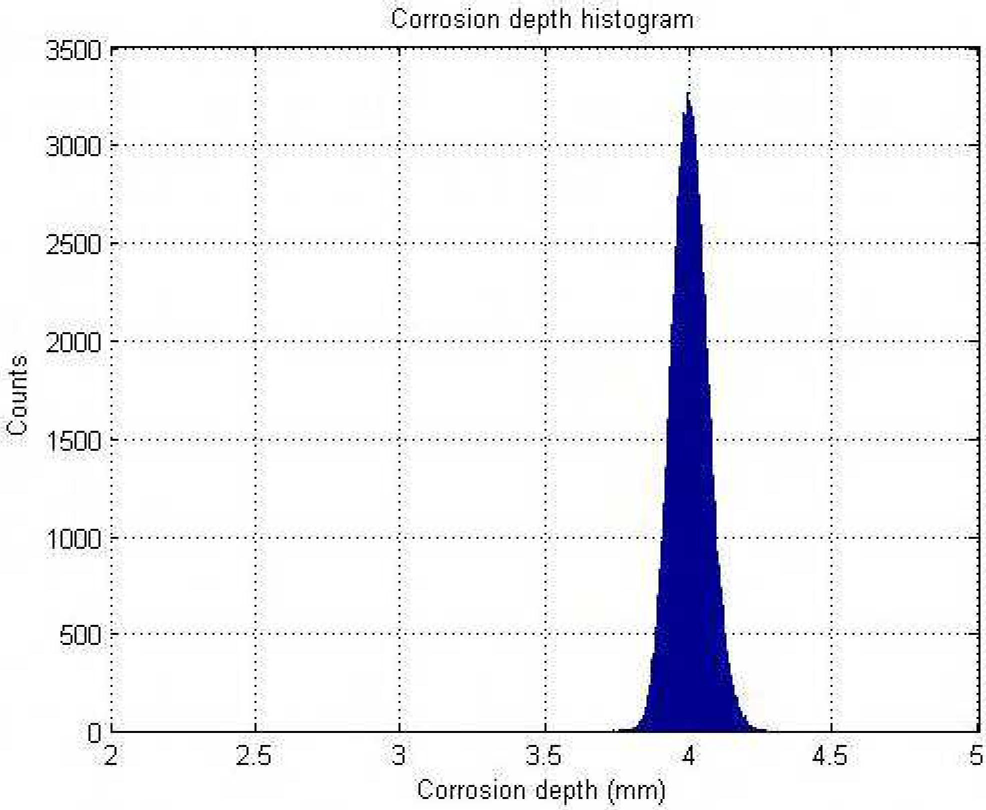


STEP 4: The corroded sample was aligned with the original Sample B using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

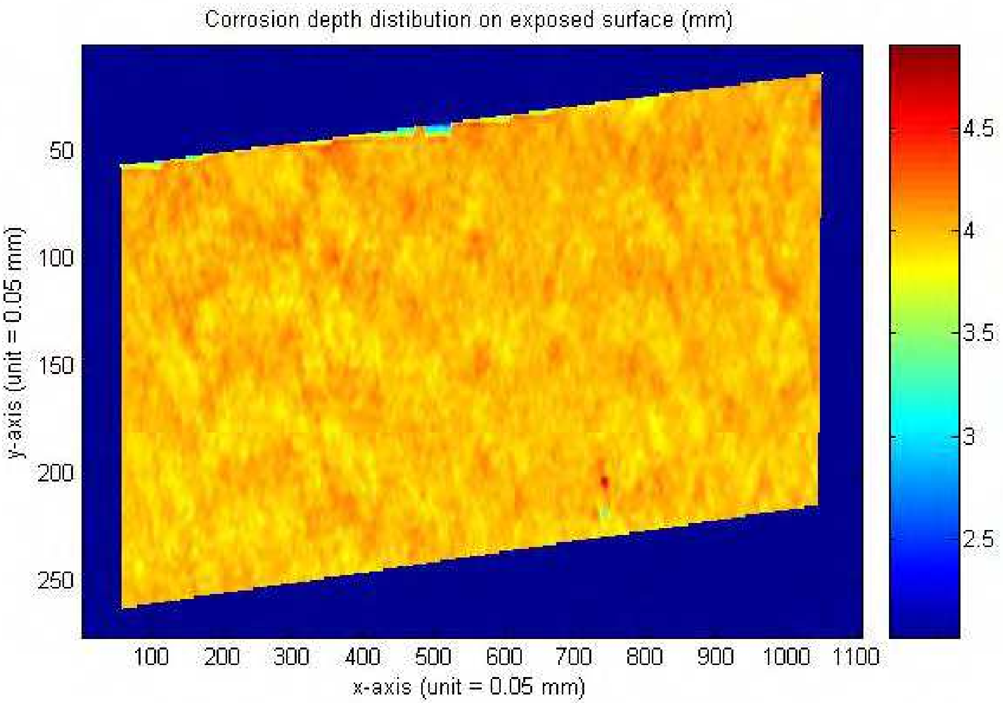
1. original image of Sample A (shown in black)
2. aligned image of corroded Sample A (shown in red)
3. the exposed surface of Sample A in its original position (shown in green)
4. the orientated exposed surface of Sample A with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample A orientated accordingly (shown in pink)



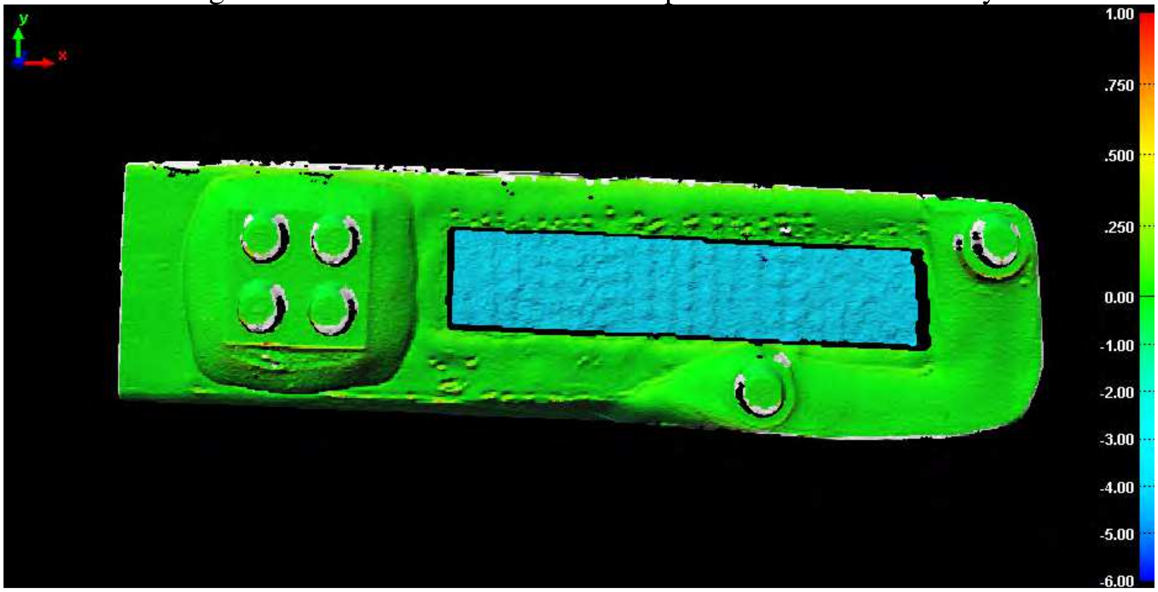
The histogram of the corrosion depth on the corrosion surface:

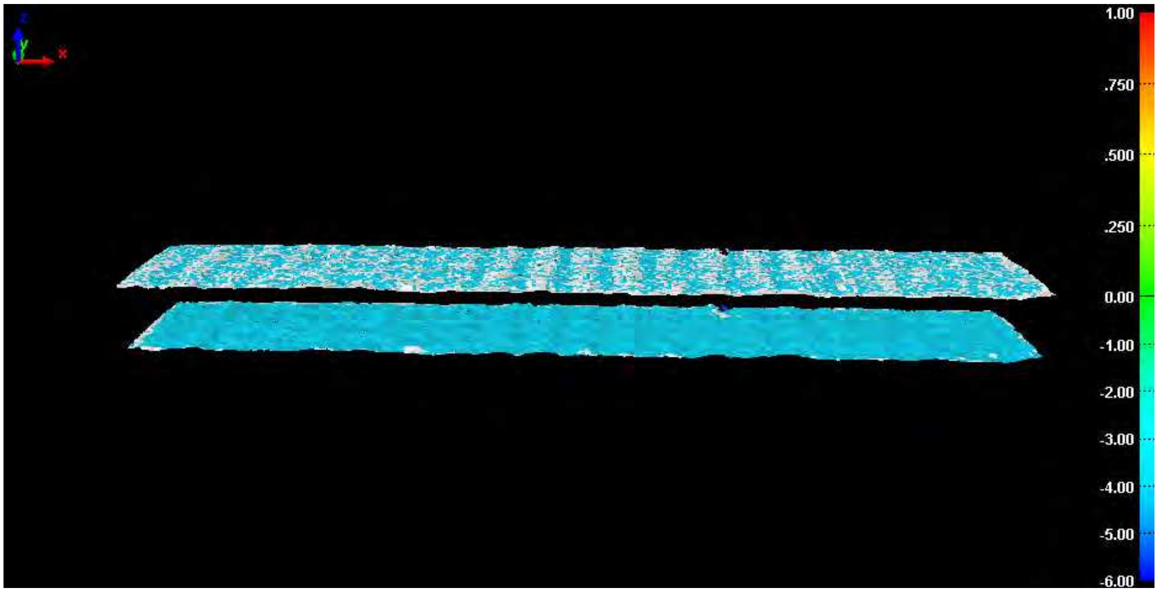


The 2D distribution of the corrosion depth on the corrosion surface:



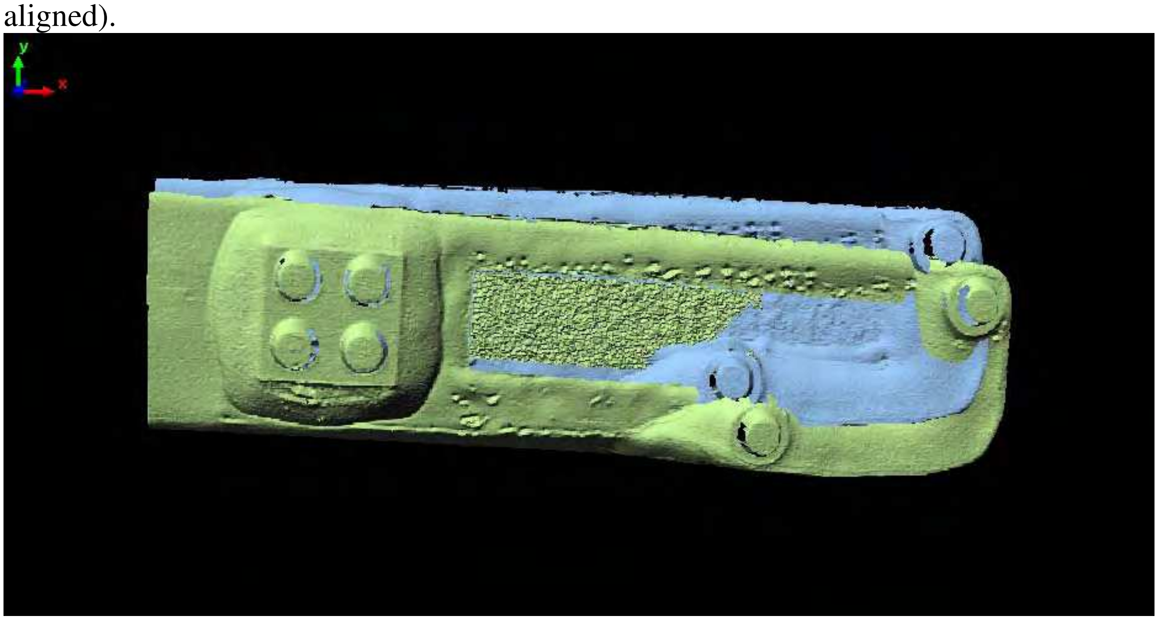
STEP 5: The alignment results and the corrosion depth distribution from PolyWorks

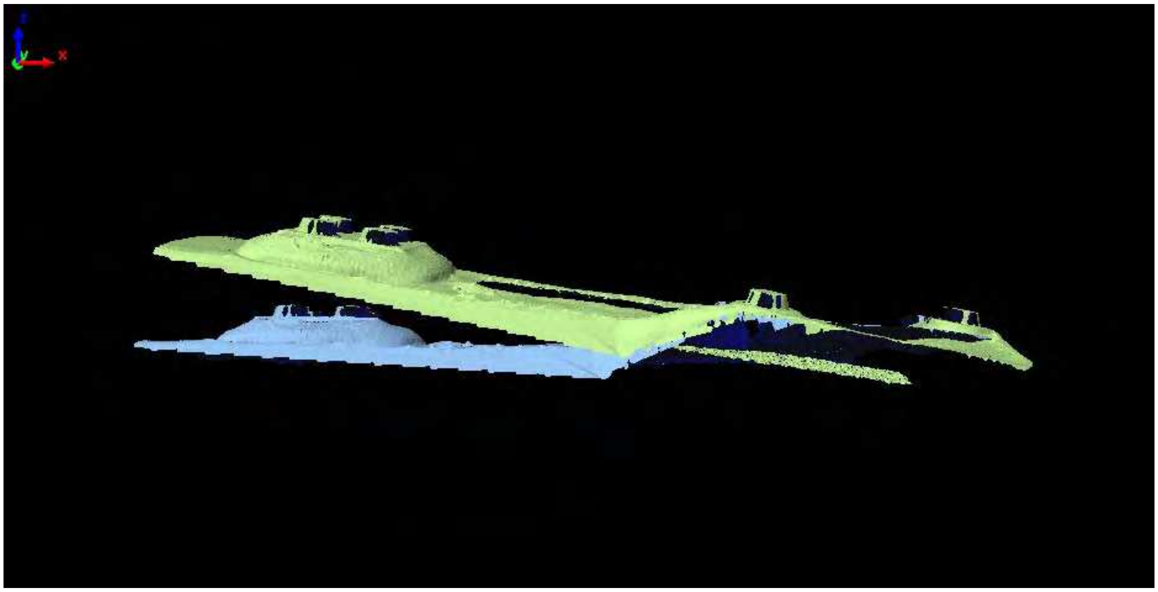




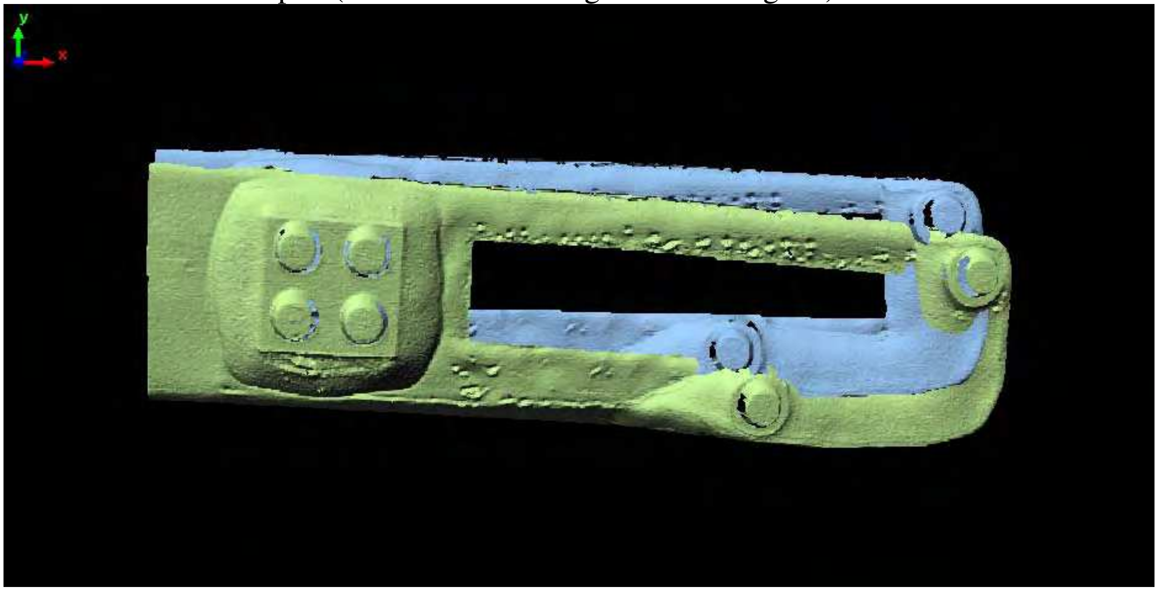
Test Results for the 4 mm with 5% noise

STEP 1. Original scanned image of Sample B is at the bottom colored in light blue and the corroded sample is at the front colored in light green. (Note: The two images are not aligned).

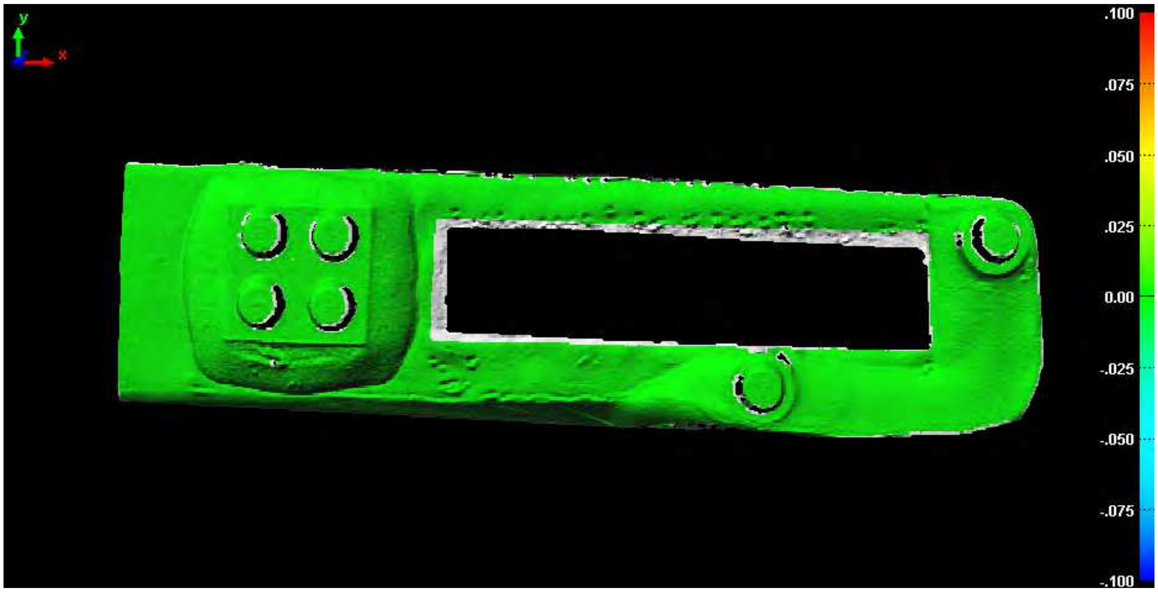




STEP 2: The exposed areas are removed individually from the original scan of Sample B and the corroded sample. (Note: The two images are not aligned).

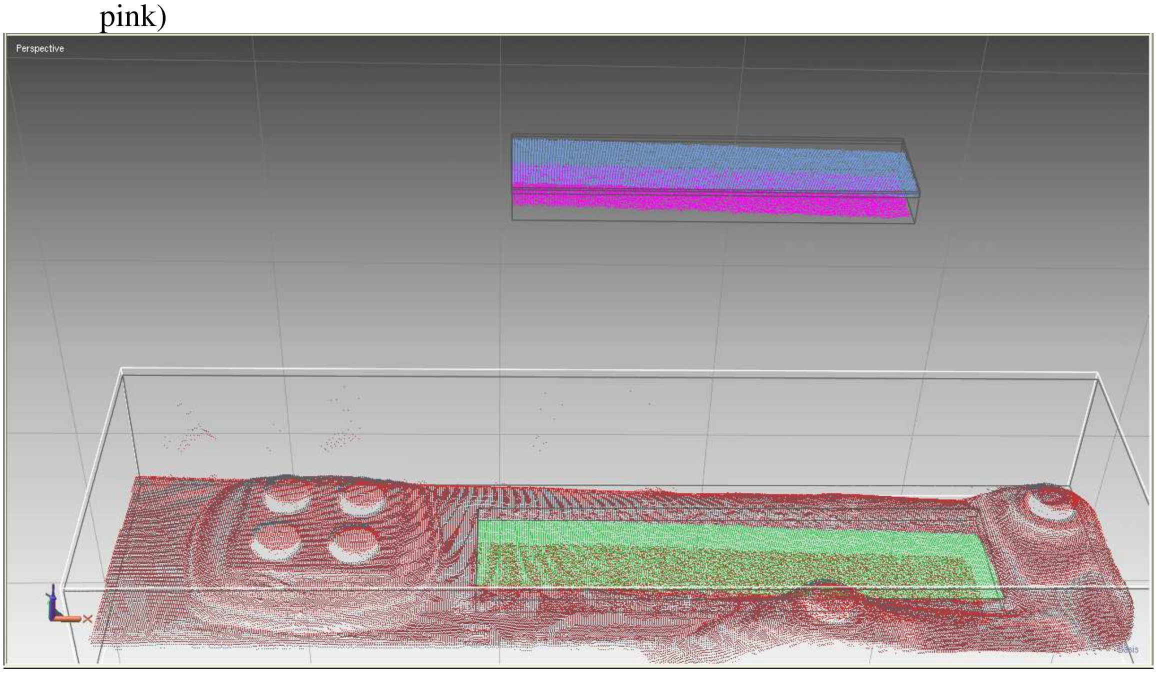


STEP 3: Align the two images with the exposed area removed. The color code shows the delta of the two images.

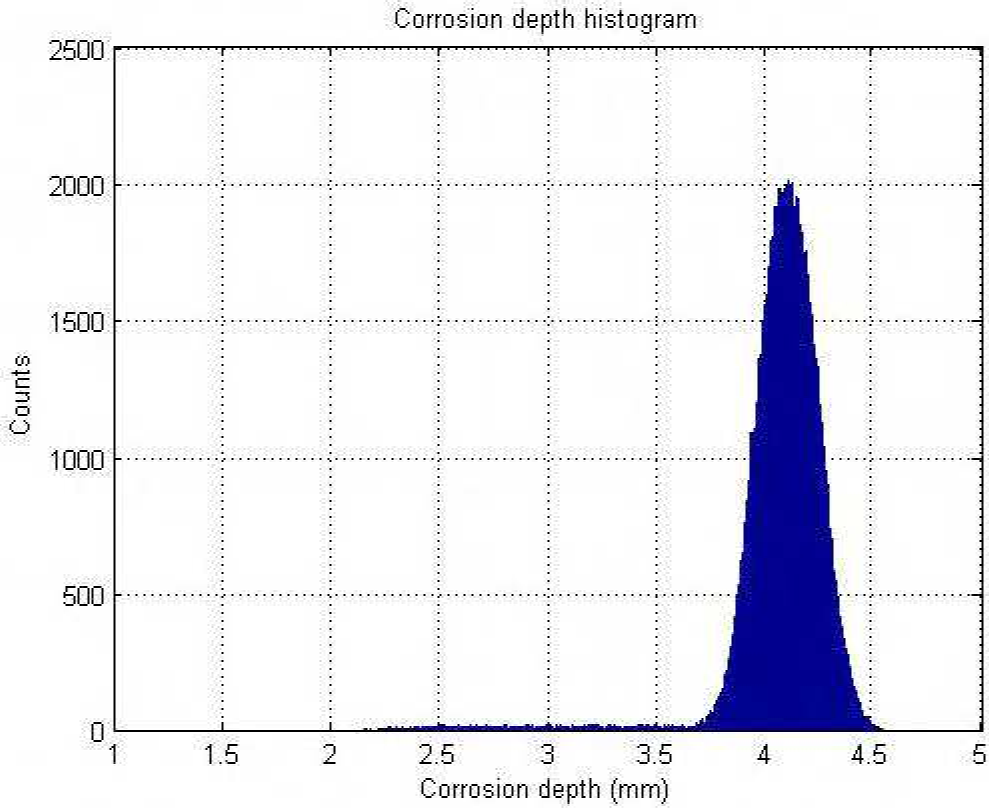


STEP 4: The corroded sample was aligned with the original Sample B using Matlab. The picture shows the following point cloud images installed under the same coordinate system:

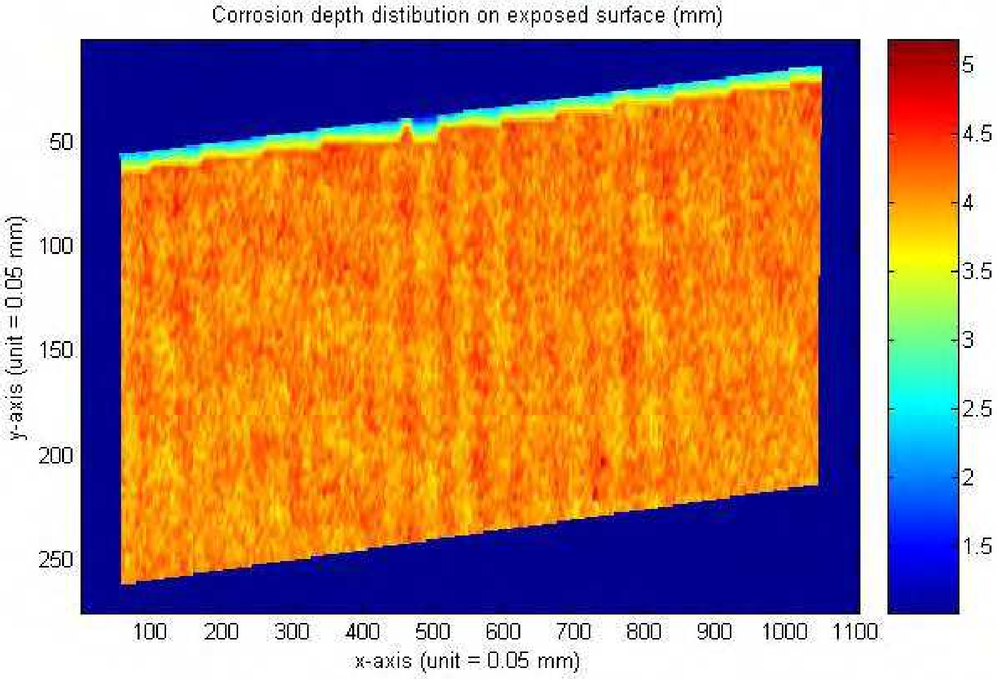
1. original image of Sample A (shown in black)
2. aligned image of corroded Sample A (shown in red)
3. the exposed surface of Sample A in its original position (shown in green)
4. the orientated exposed surface of Sample A with its surface normal aligned to z-axis (shown in blue)
5. the corresponding parts of corroded Sample A orientated accordingly (shown in pink)



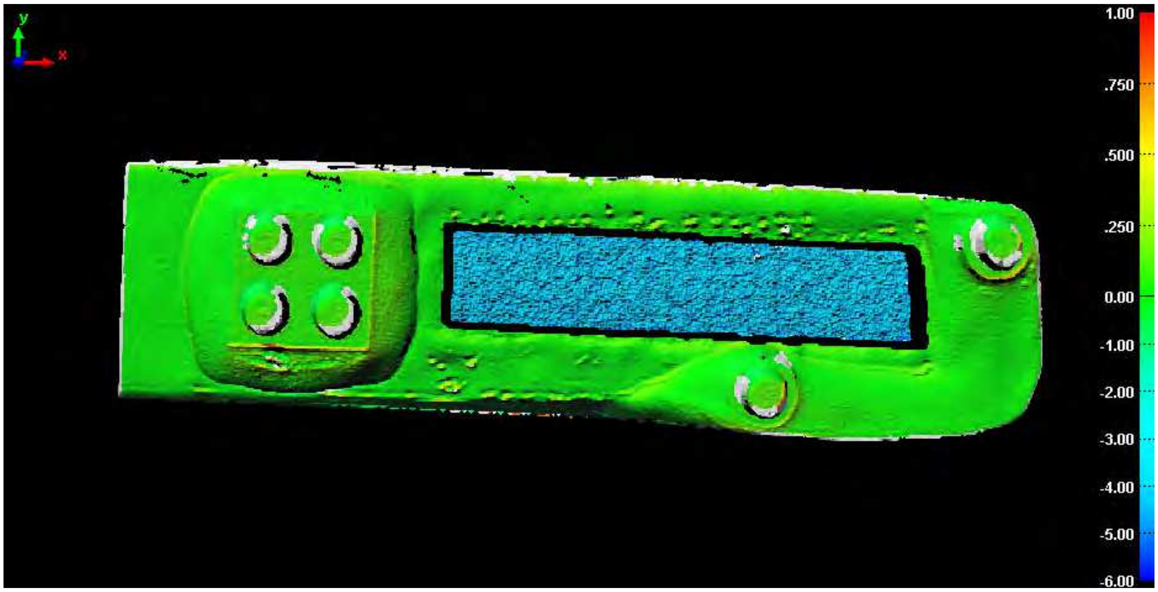
The histogram of the corrosion depth on the corrosion surface:

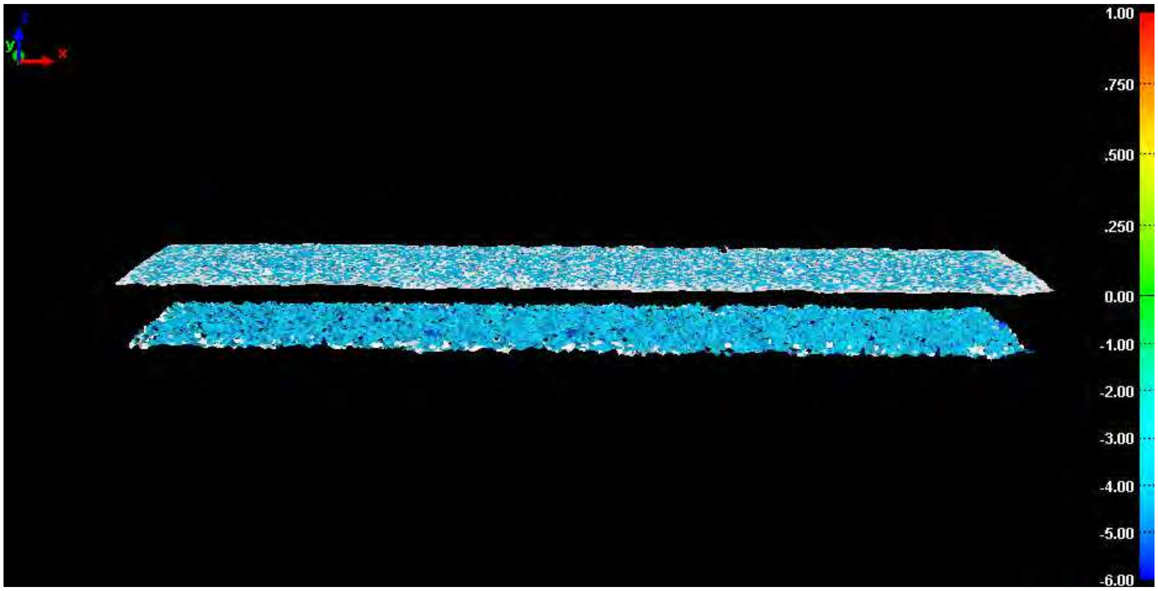


The 2D distribution of the corrosion depth on the corrosion surface:



STEP 5: The alignment results and the corrosion depth distribution from PolyWorks


y
x
1.00
.750
.500
.250
0.00
-1.00
-2.00
-3.00
-4.00
-5.00
-6.00


z
y
x
1.00
.750
.500
.250
0.00
-1.00
-2.00
-3.00
-4.00
-5.00
-6.00