MITSUBISHI ELECTRIC
Changes for the Better

DamageEst: Accurate Estimation of Damage for Repair using Additive Manufacturing

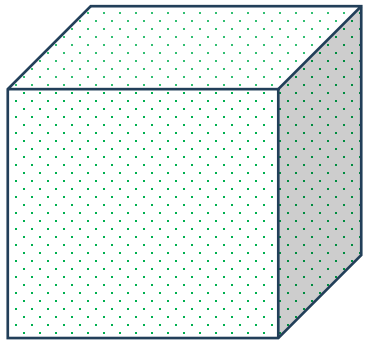
Patrick Gambill*, Devesh K. Jha, Bala Krishnamoorthy*, Arvind Raghunathan and William Yerazunis

*Washington State University

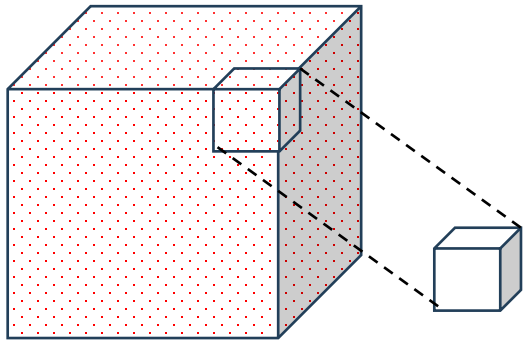
MITSUBISHI ELECTRIC RESEARCH LABORATORIES (MERL)
Cambridge, Massachusetts, USA
http://www.merl.com

Problem Description

Original printed part



Damaged part



Objectives:

- Estimate damage using observations from a vision sensor (a depth camera)
- Design tool path geometry to deposit metal to cure the damage while avoiding collisions

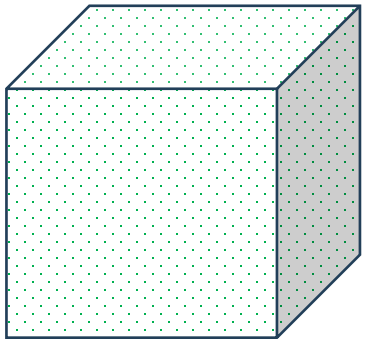
Assumptions:

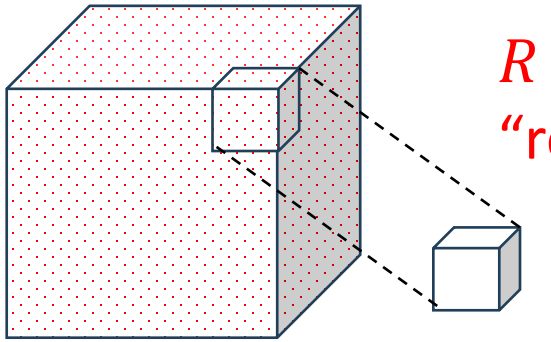
- We do **not** access to a CAD model of the damaged part
- We **do** have access to the CAD model of the original part

MITSUBISHI ELECTRIC
Changes for the Better

Problem Description

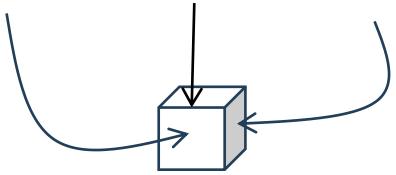
G – set of points in the "good part" point-cloud



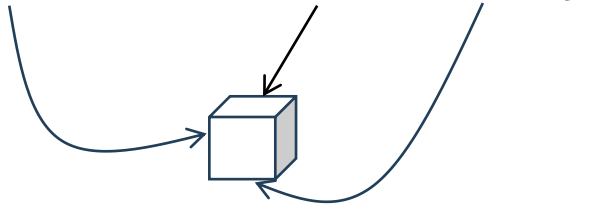


R – set of points in the "repair part" point-cloud

$$G - \mathrm{R} = \{p \mid p \in G, \notin R\}$$



$$\mathrm{R} - G = \{p \mid p \in R, \notin G\}$$



The points on the exposed surfaces of the damage are in $G - \mathrm{R}$

The points on the hidden surfaces of the damage are in $\mathrm{R} - G$

DamageEst

- Our goal is to find a valid, watertight mesh that fully encloses the damage volume and doesn't enclose the undamaged region.
- To do so, first reconstruct the inner damage surface and find its boundary.
- Next, "push" this boundary to the background mesh (which will enclose everything).
- "Split" the background mesh into two components along the "pushed" boundary.
- Finally, attach the reconstructed surface to a component of the "split" mesh to get an enclosing surface.

MITSUBISHI ELECTRIC
Changes for the Better

DamageEst

Assumptions

1. We have the original STL file and a point cloud generated from that STL. *
2. The damage volume is connected. (If it is not connected, we can repeat this method for each damage site).
3. The damage volume contains part of the surface of the original STL.

* The good STL point cloud intentionally includes high density sampling on every edge and explicitly contains every vertex in the STL. This leads to better alignment of clouds in the preprocessing steps.

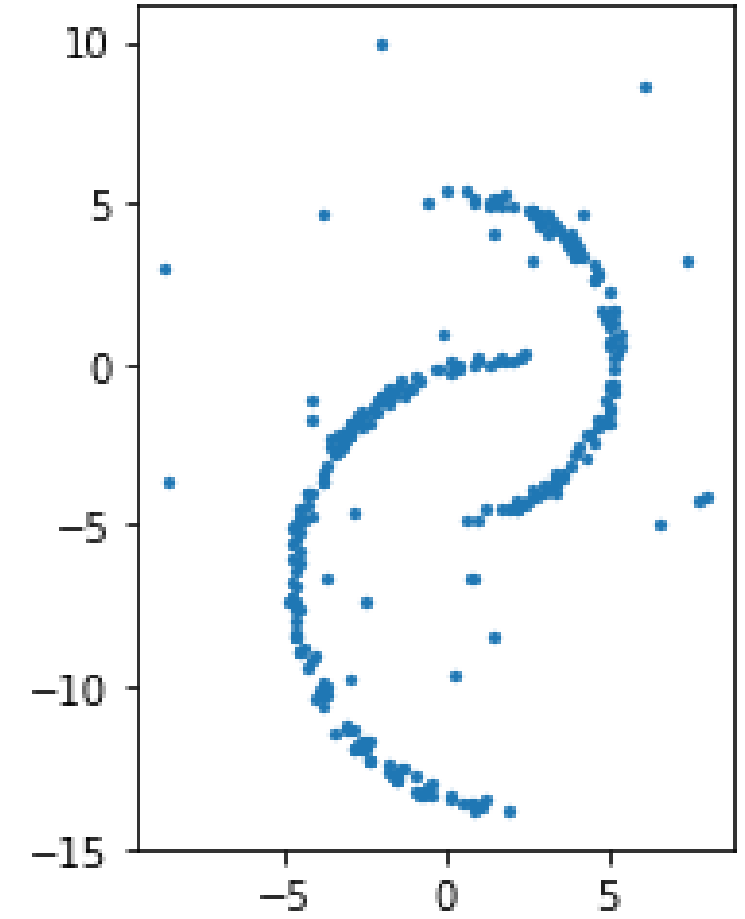
Aside on DBSCAN

In order to remove noise from the point cloud and to detect damage sites, we apply DBSCAN

To clean the scanned point cloud up, we use DBSCAN* (Density Based Spatial Clustering Of Applications with Noise). DBSCAN yields the cluster identifier of each cluster in the input (plus a "cluster" of noise points that don't belong to any cluster). DBSCAN doesn't need to know the number of clusters in advance.

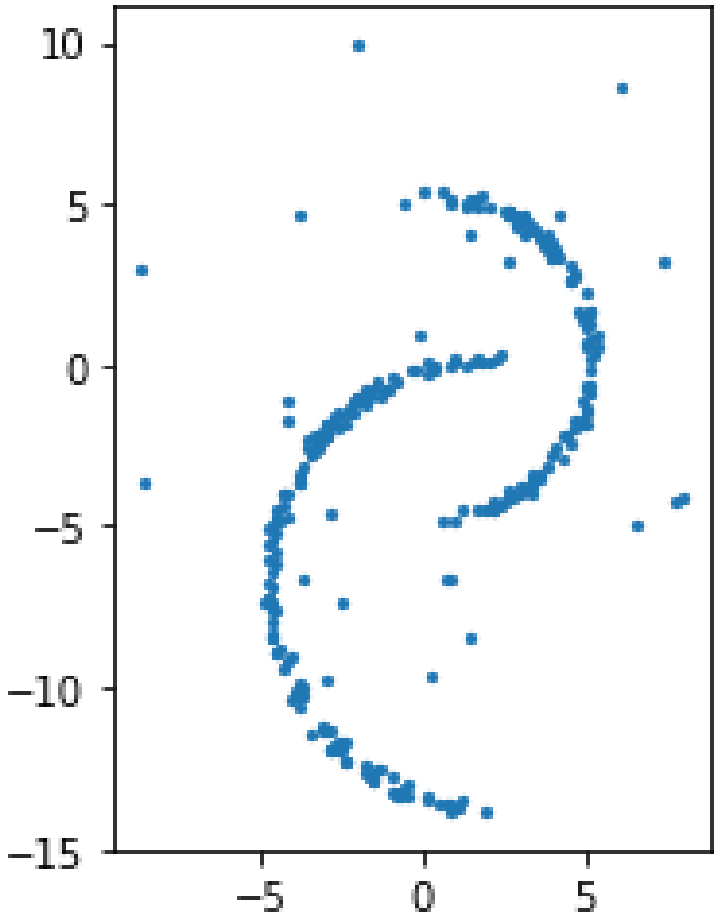
To illustrate the algorithm, consider the cloud of points on the right:

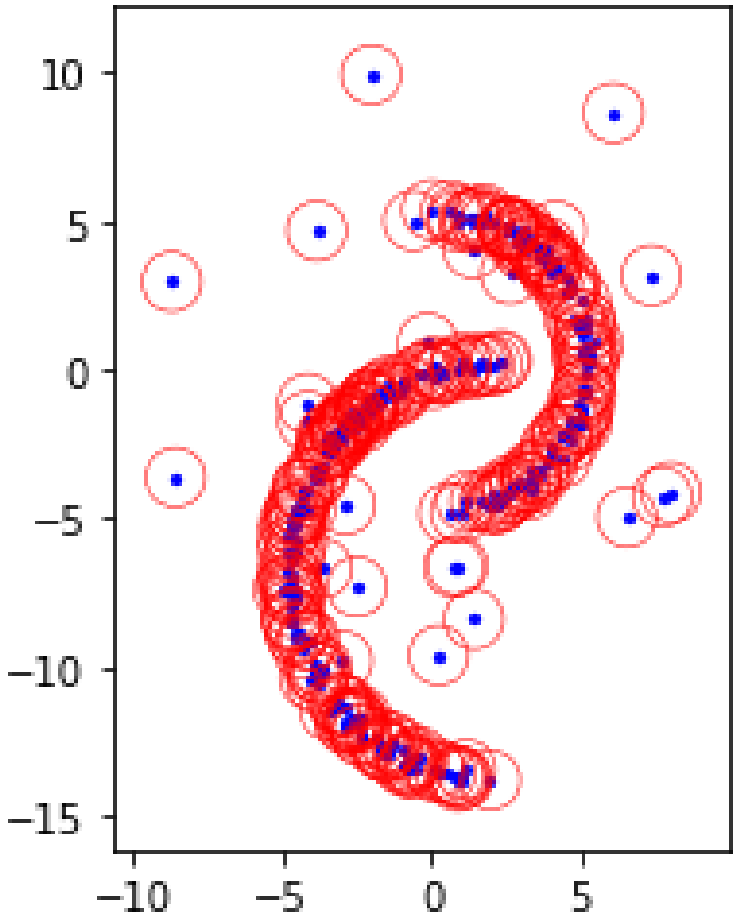
(*)"DBSCAN" has nothing to do with databases, and doesn't do anything like scanning. Near as we can figure it, it's just a cool acronym. It's also quite durable, having first been used in 1996.



Aside on DBSCAN

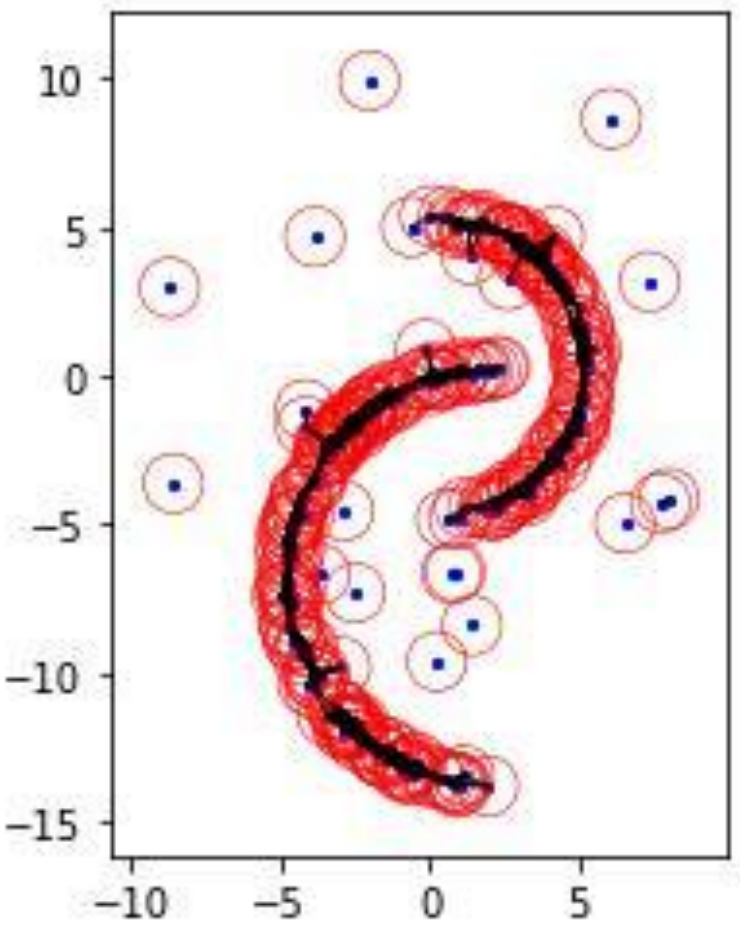
First, we create a ball of radius ϵ around each point in the cloud.

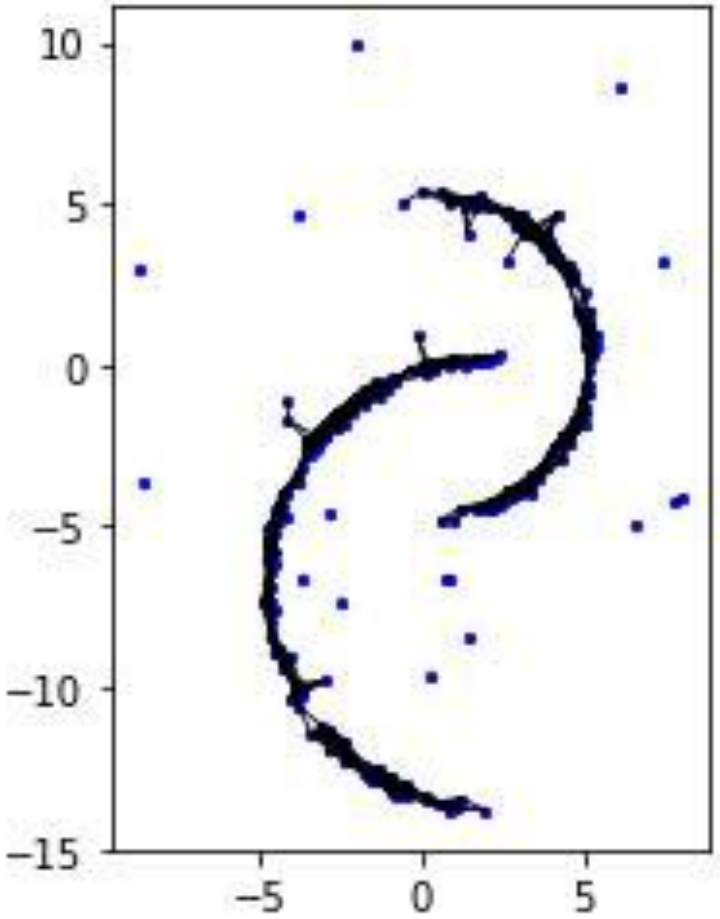




Aside on DBSCAN

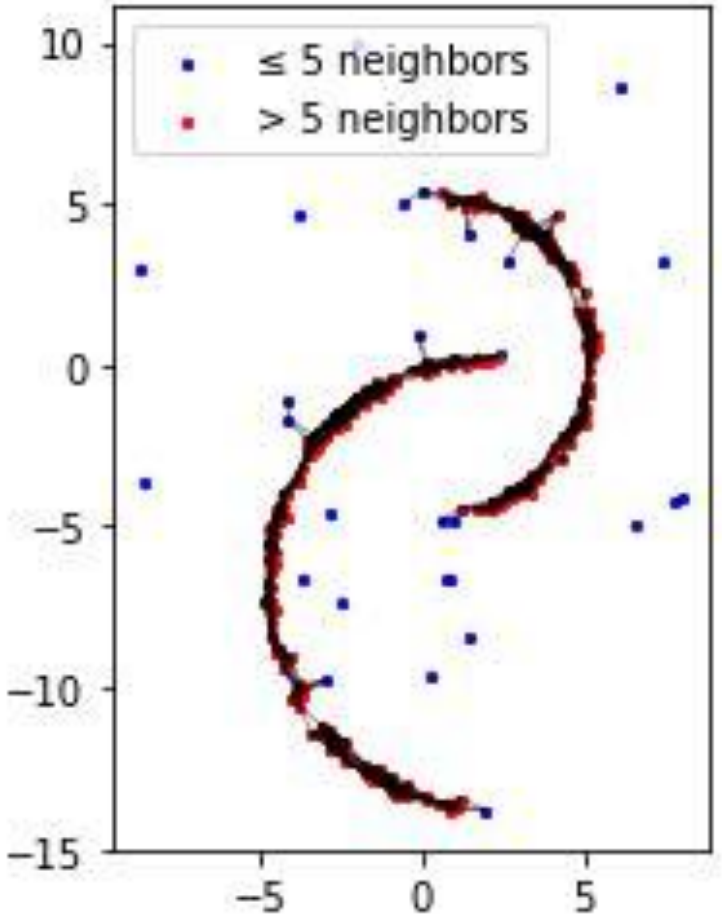
Next, we create the graph where each point is joined to its ϵ-neighbors.

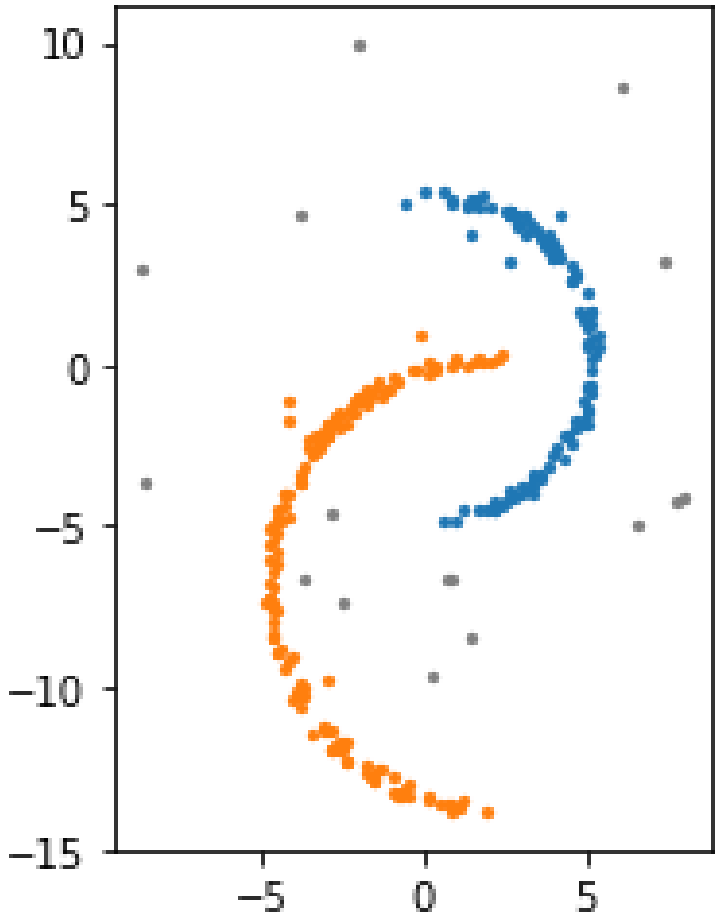




Aside on DBSCAN

Each node with a degree greater than k is a core vertex. Each component containing a core vertex is a cluster.

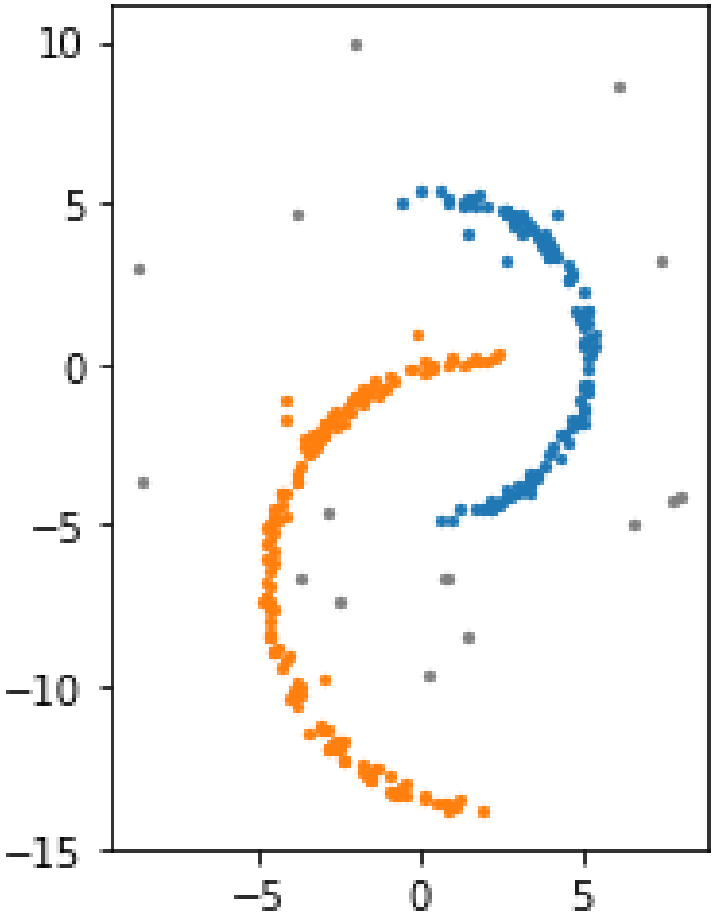




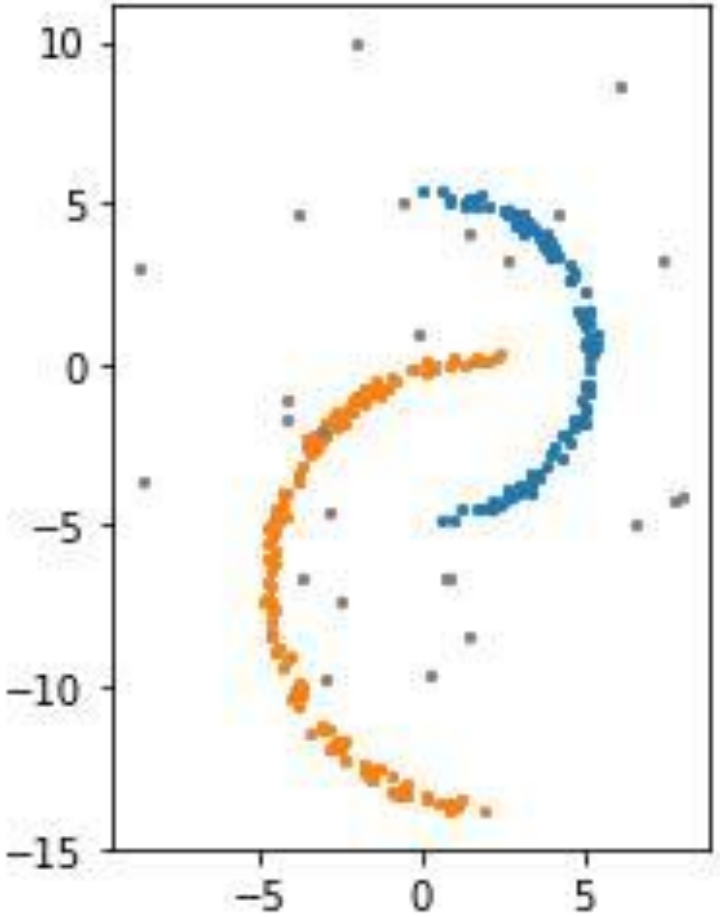
Aside on DBSCAN

DBSCAN can filter noise and can find non-linear clusters!

Clusters from DBSCAN

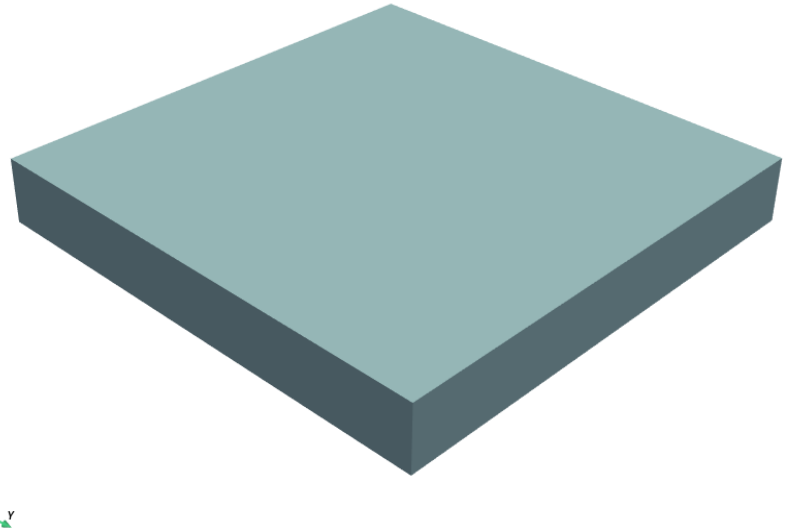


Created Clusters

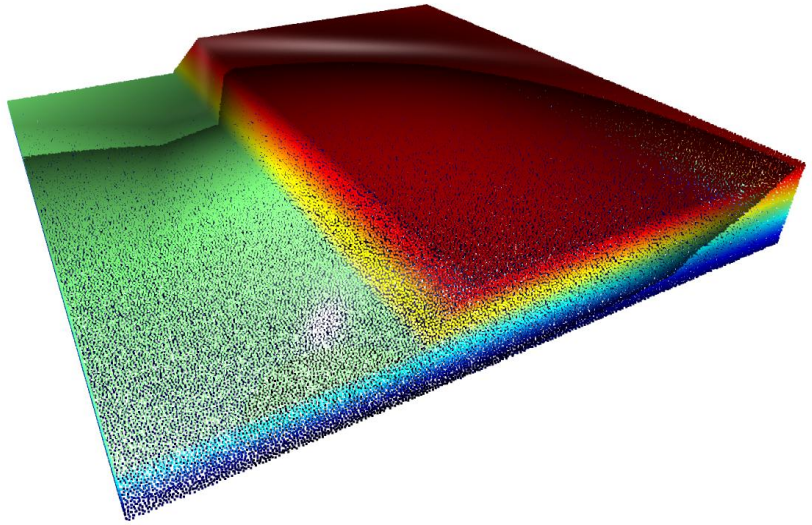


DamageEst

Undamaged Part – a noiseless, watertight, triangulated, damage free CAD model.



Damaged Part Scan - a noisy, porous, real-world mess of a point cloud.



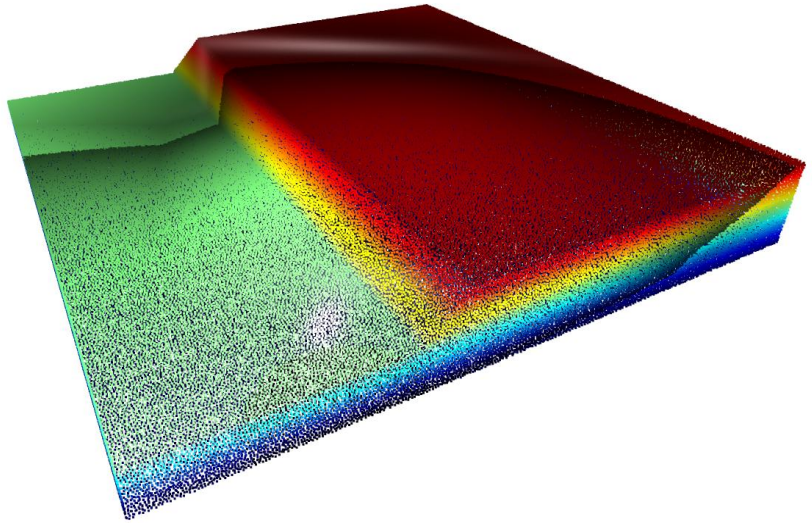
DamageEst

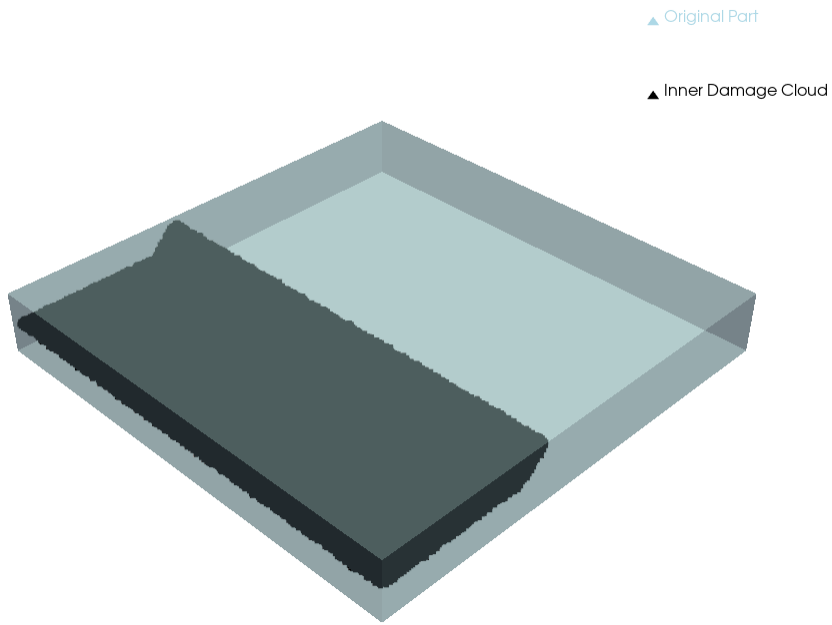
Preprocessing

1. Align the point cloud scan with the original part's STL (We need to be working in the same coordinate system when comparing parts).

2. Find the points in the damage scan that are far from the surface of the original part.

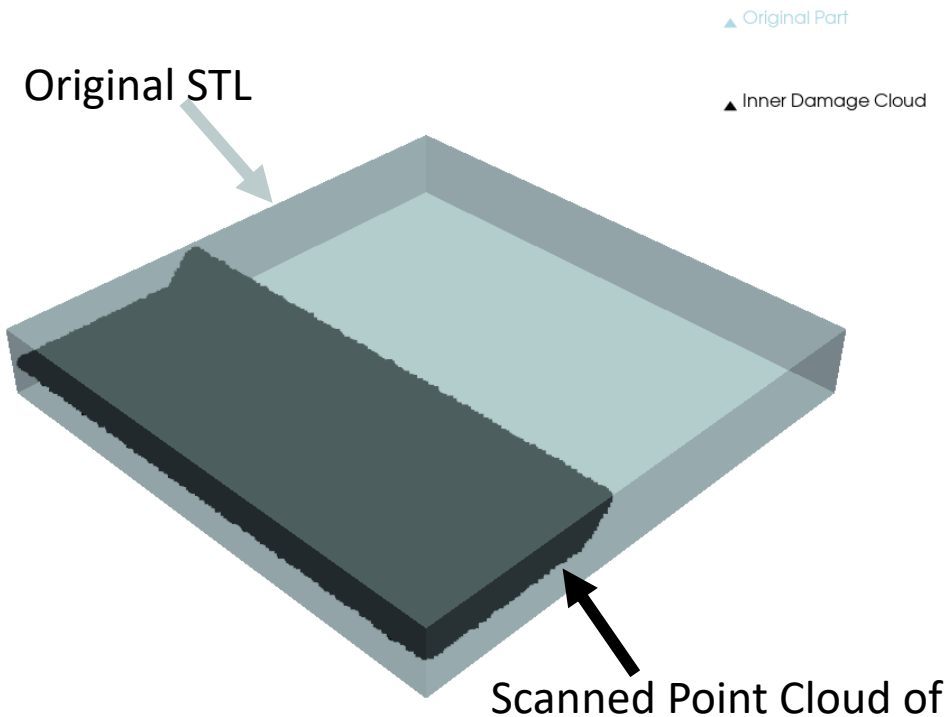
3. Cluster these points using DBSCAN to remove noise and to get each individual damage site. We can then apply DamageEst to each damage site.



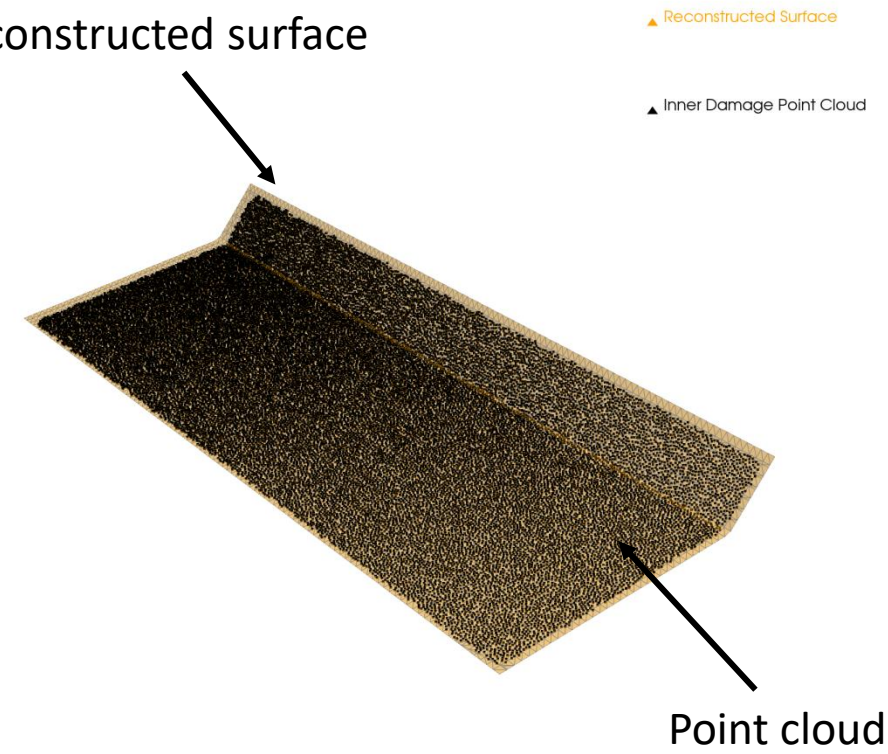


Our plan of Action:

1. **Get Inner Damage Surface Points via Hausdorff Distance from the known good STL-derived point cloud**

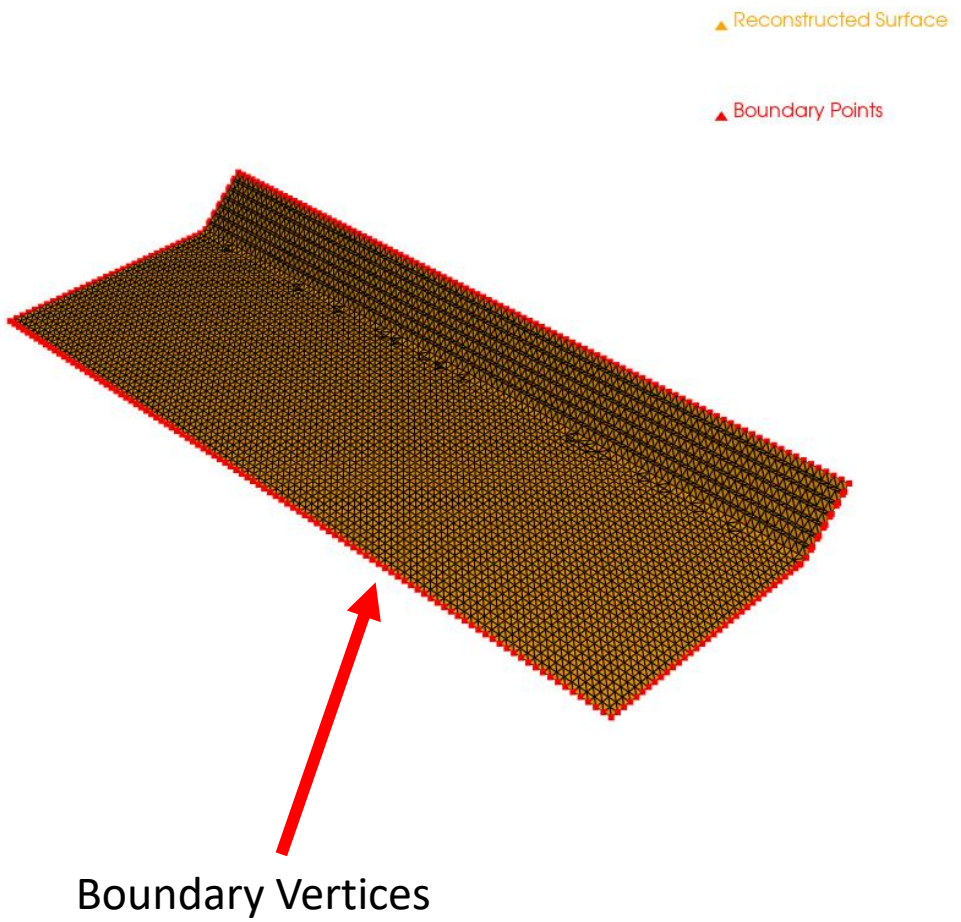


2. **Reconstruct Inner Damage Surface via PyVista's "reconstruct_surface" library**

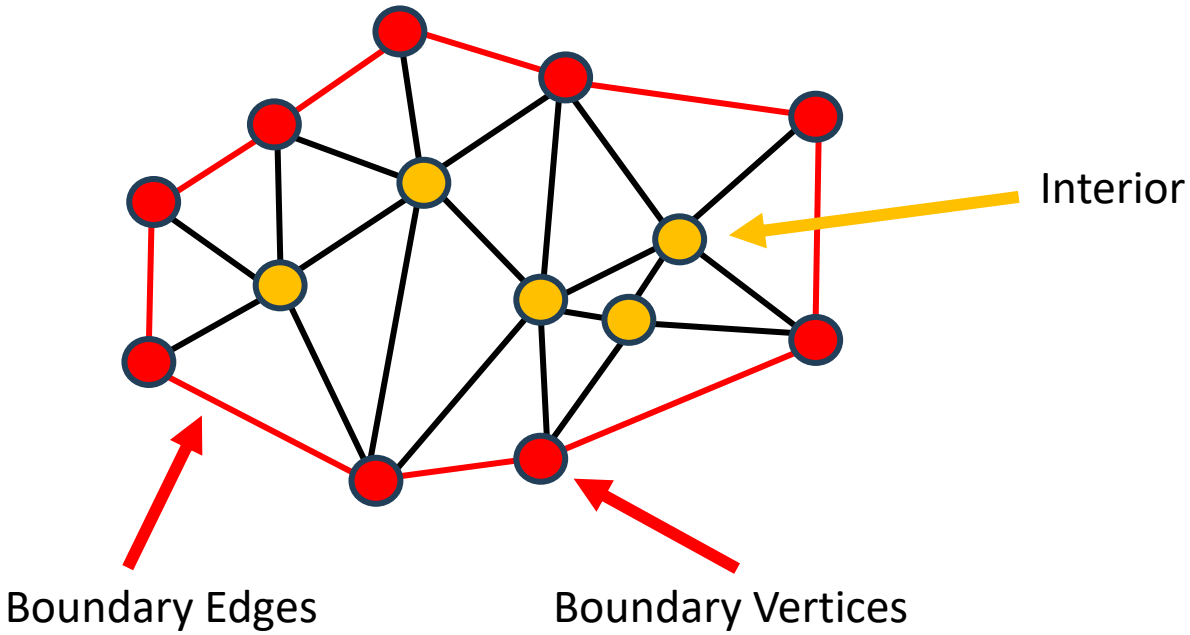


(Pyvista: reconstruct_surface)

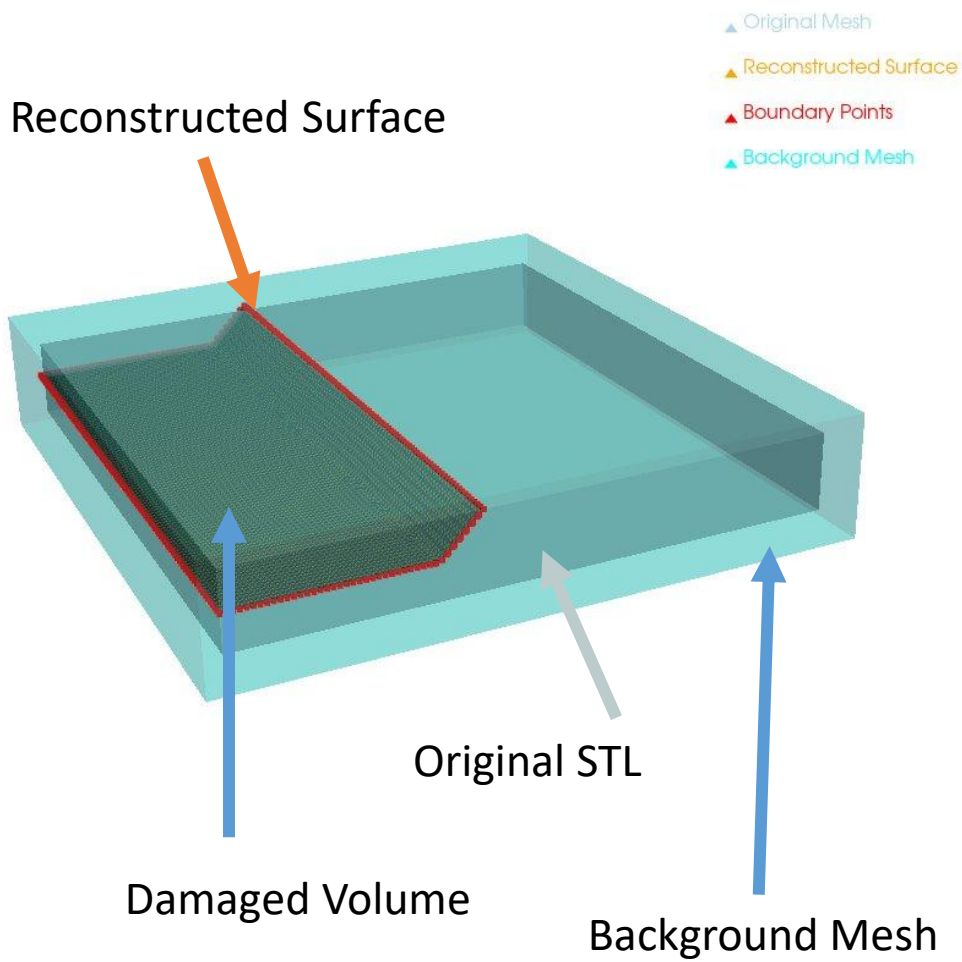
3. Find Surface Boundary



- We find the boundary of the surface by finding which edges in the mesh are bound only one triangle.
- The vertices contained in these boundary edges will eventually be "pushed" to the background mesh.



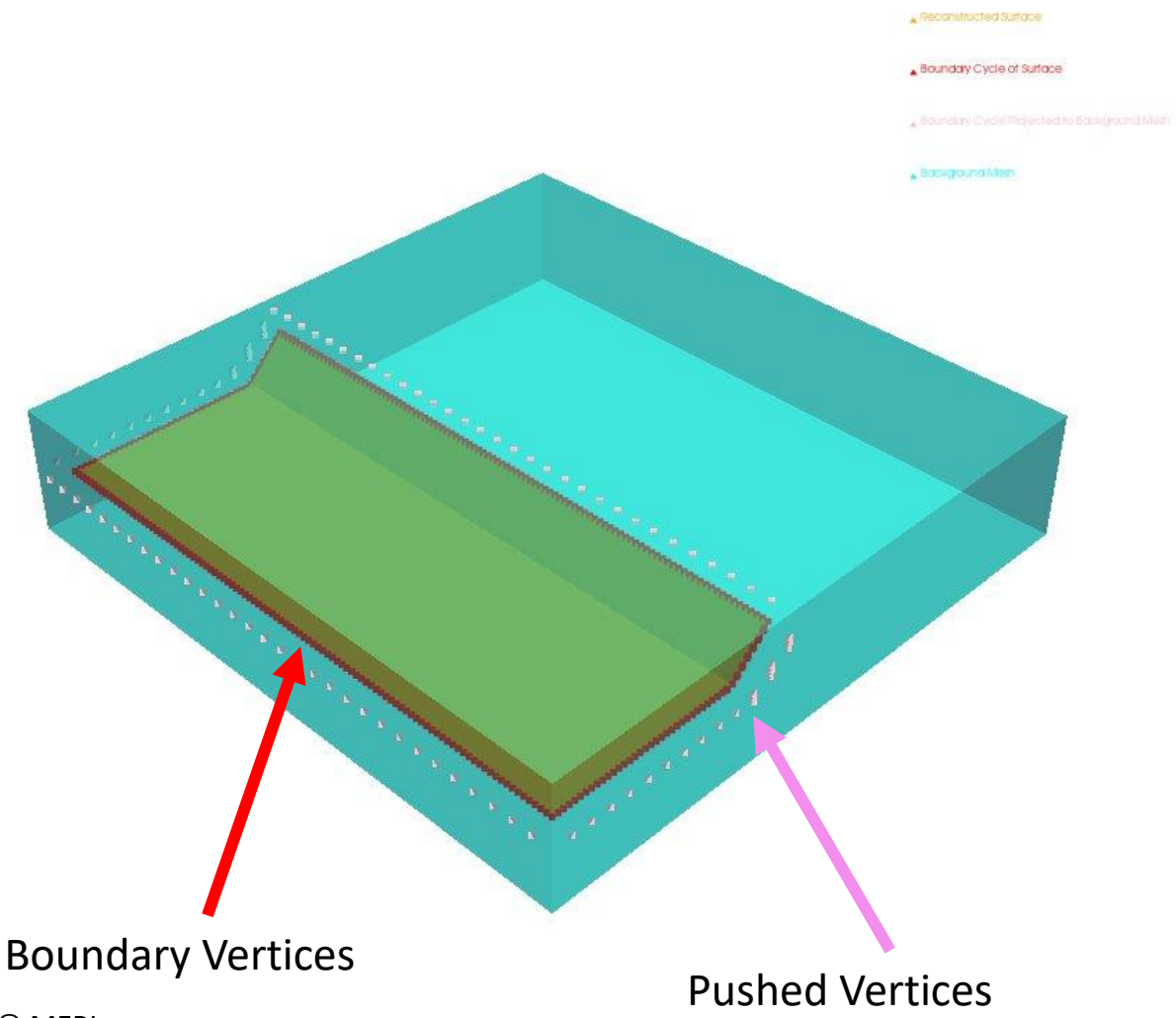
4. Create Background Mesh



- The background mesh should enclose the mesh from the original STL and the reconstructed surface. This is to ensure there is no self-intersection.
- This background mesh is where the "pushout" and "split" will ultimately occur.
- Think of the background mesh as a “neutral ground” that both the original (precise) CAD model generated cloud and the (noisy) scanned damaged-part cloud can be pushed onto, and then navigated easily on an equal (and relatively fine grained) basis.

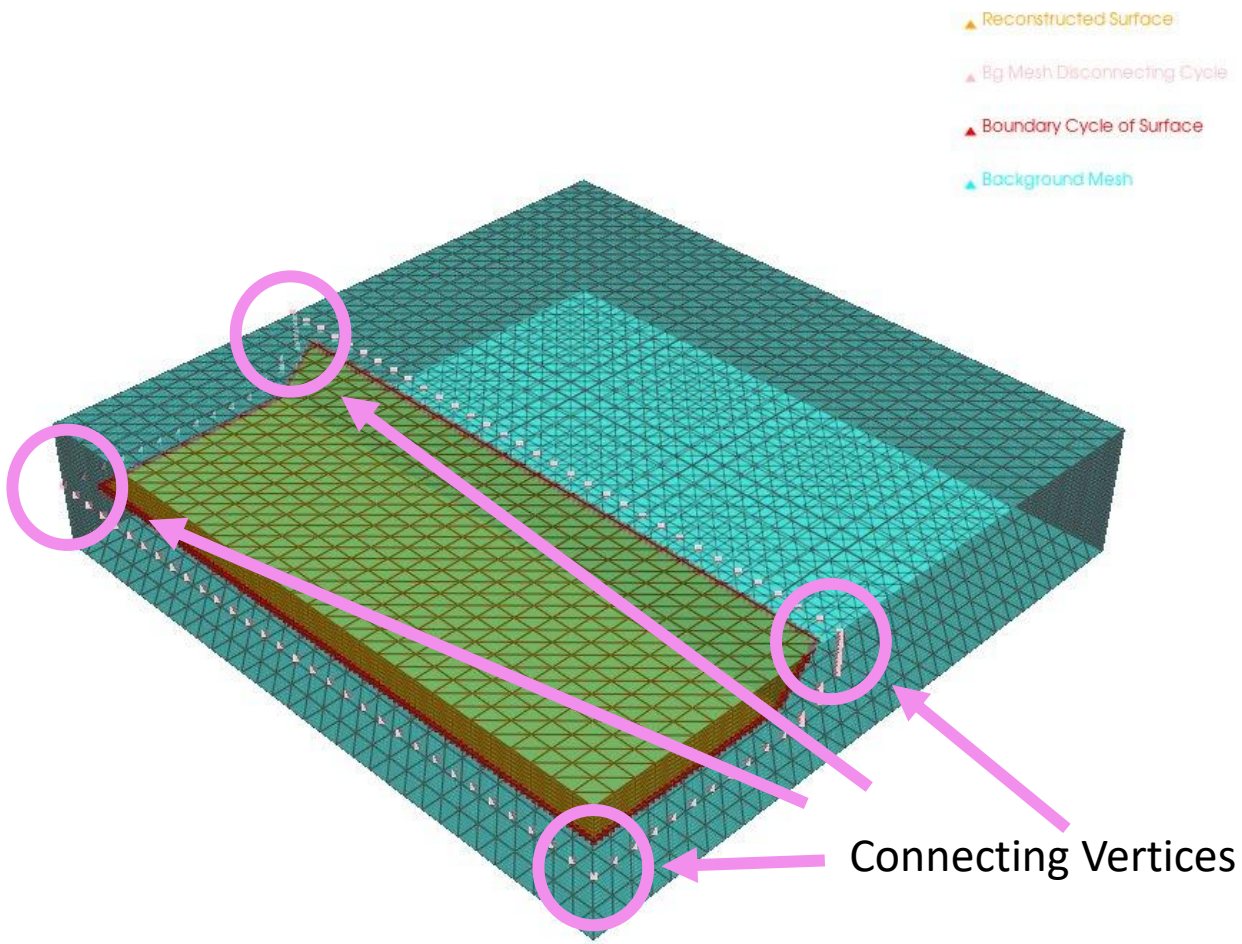
DamageEst

5. Push Boundary Vertices



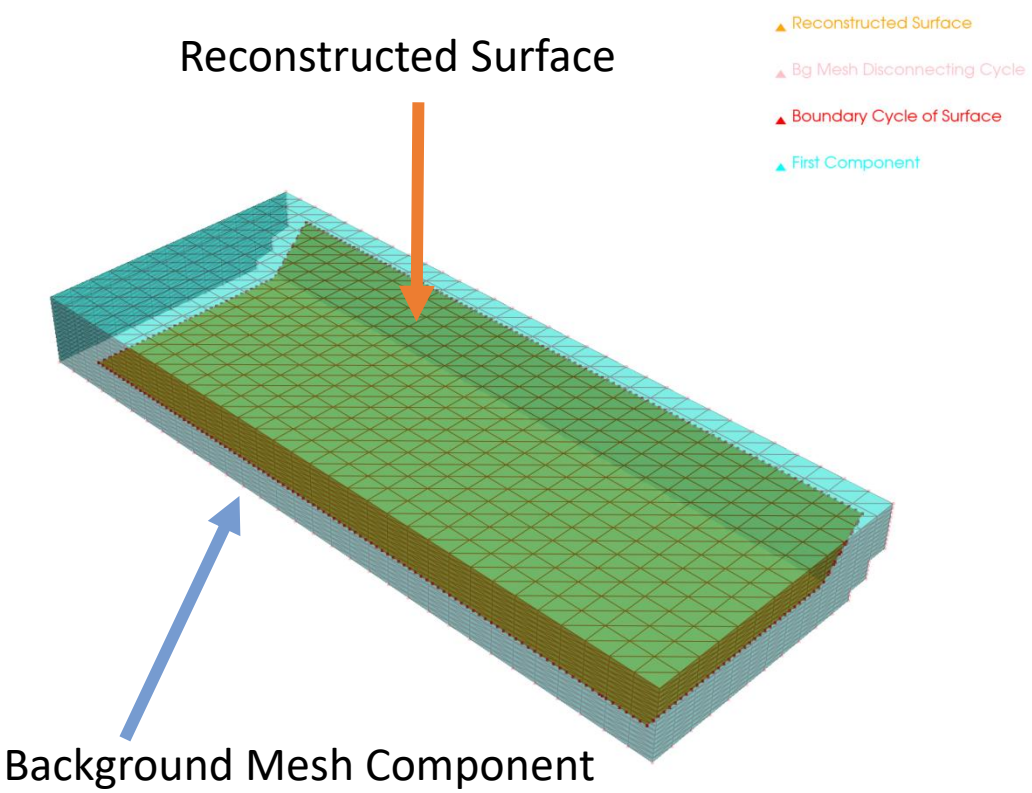
- Map each of the boundary vertices to the closest vertex in the background mesh.
- This is the "pushout" step.

6. Connect Pushed Vertices



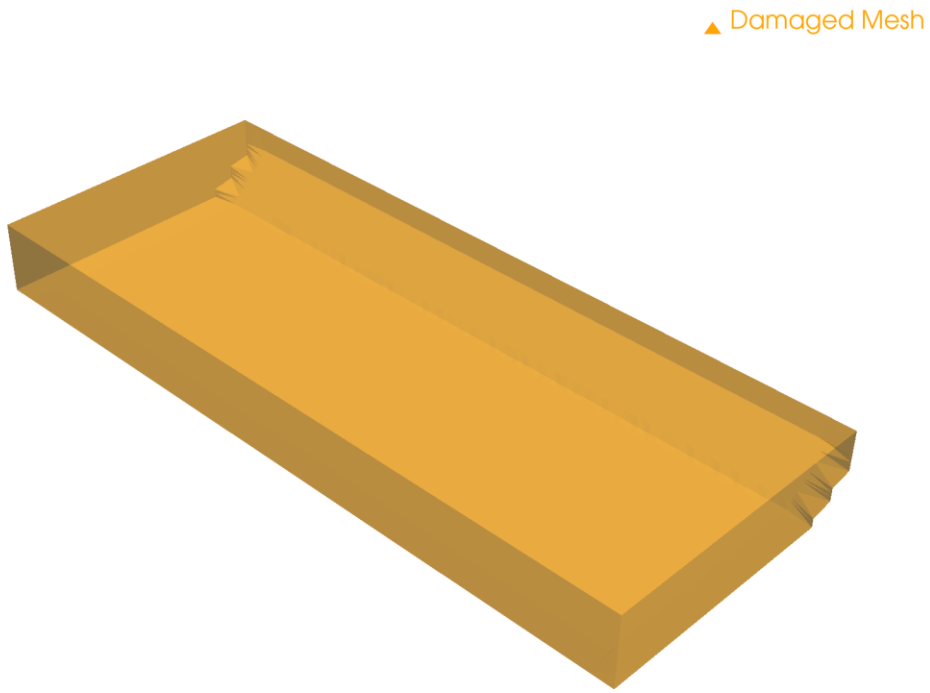
- Now, we connect the pushed vertices to get a cycle. To achieve this, we take two adjacent background vertices and connect their "pushouts" by finding the shortest path between them on the background mesh.
- This connected cycle is where we "split" the background mesh.

7. Split Background Mesh



- When we split the background mesh, one component will correspond to the damaged volume and the other will correspond to the undamaged volume.
- The component corresponding to the damaged volume is shown with the reconstructed surface.

8. Connect Surfaces



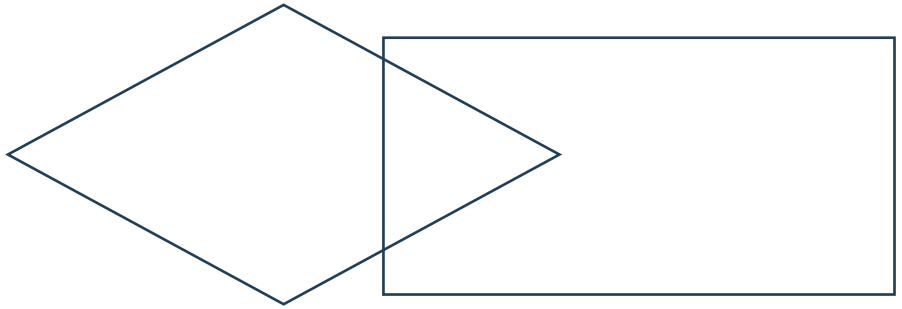
- Now we connect the component of the background mesh and the reconstructed surface to get the damaged mesh. To do so, we create an edge between each boundary vertex and it's corresponding "pushed" vertex. Then we fill in the triangles using pymeshfix.

- The damaged mesh will enclose the damage volume.

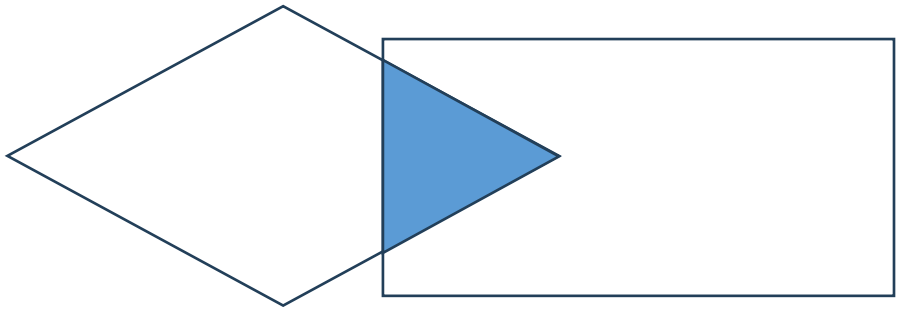
MITSUBISHI ELECTRIC
Changes for the Better

DamageEst

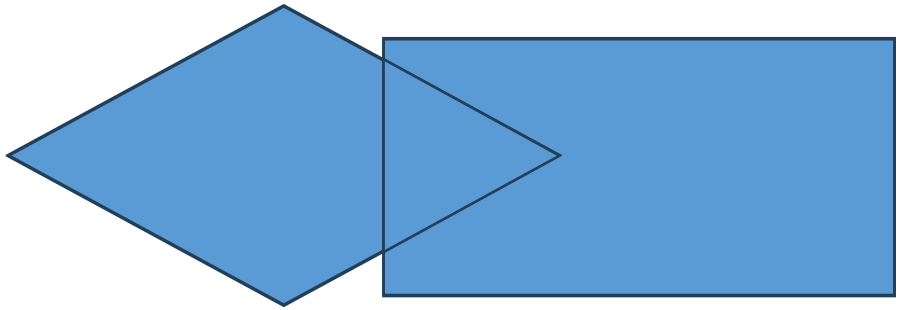
Boolean Operations



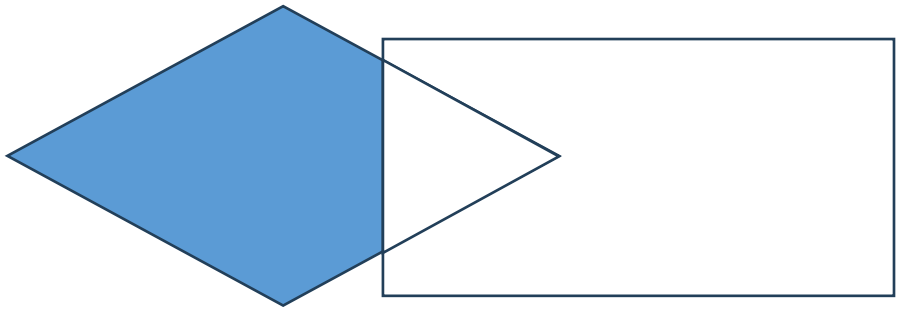
A B



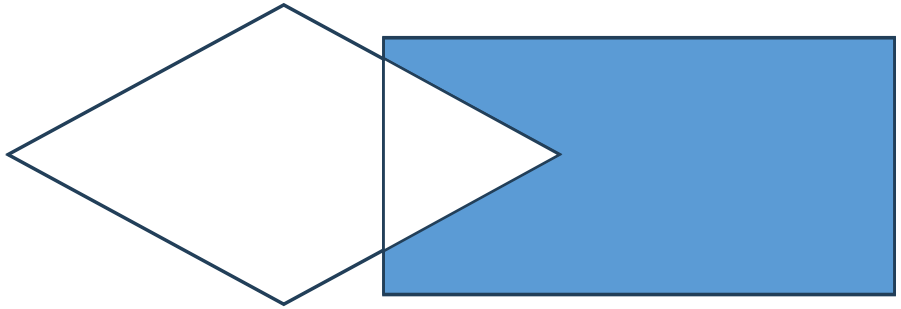
$A \cap B$



$A \cup B$

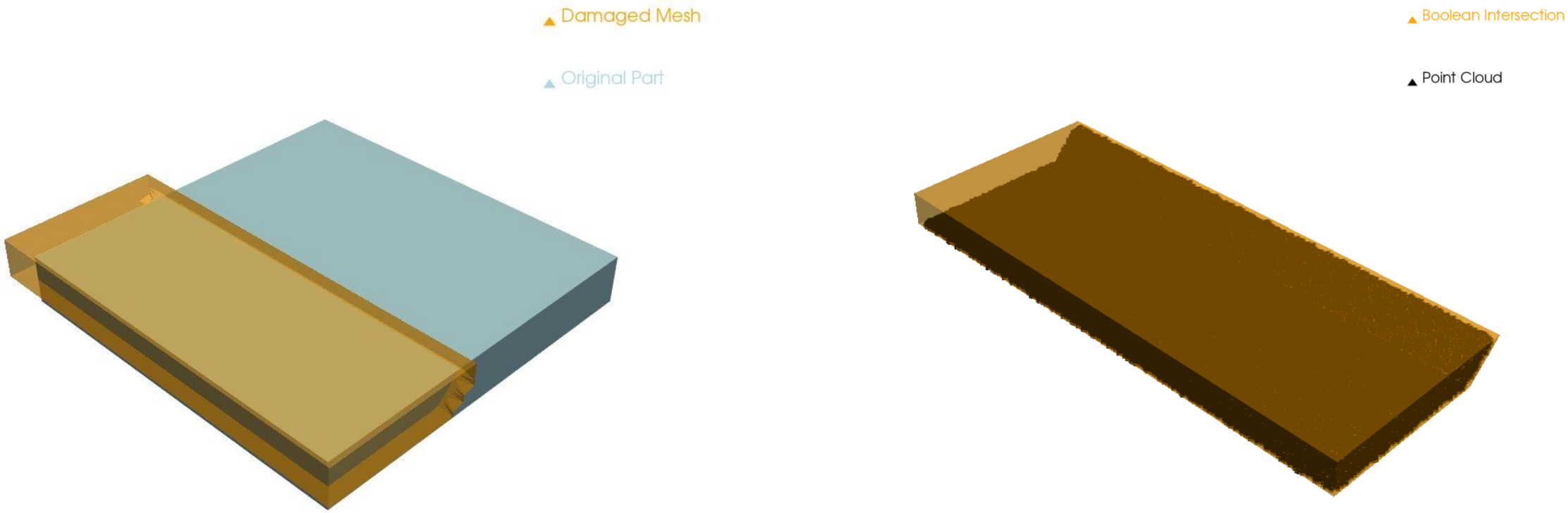


$A - B$



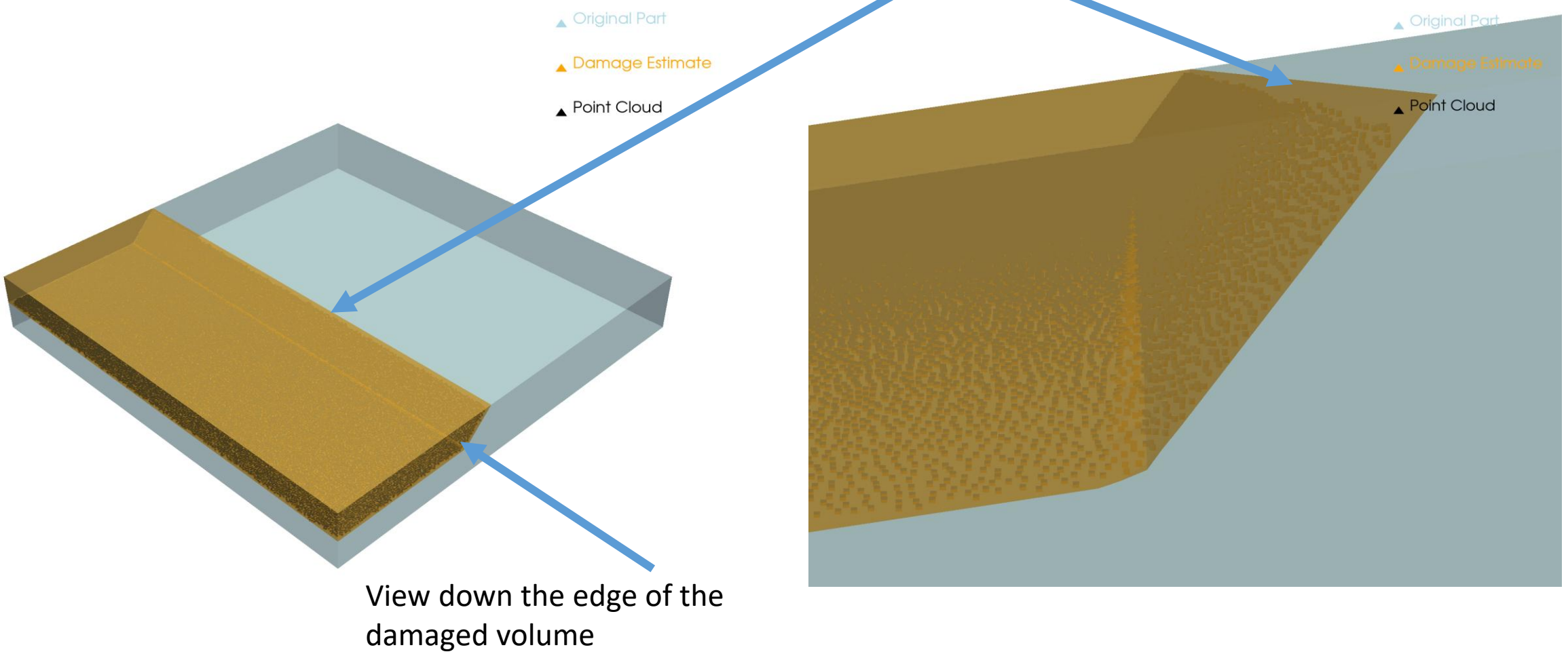
$B - A$

9. Take Boolean Intersection (Find Overlapping Volume)



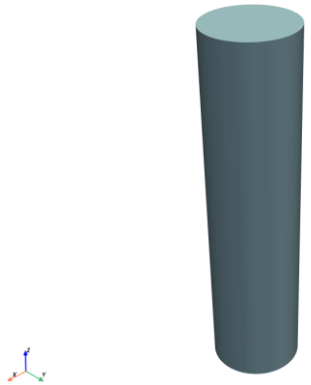
DamageEst

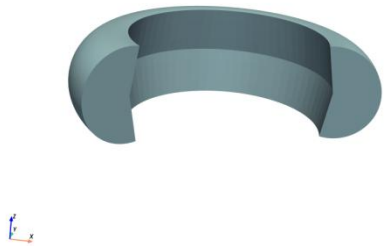
Result



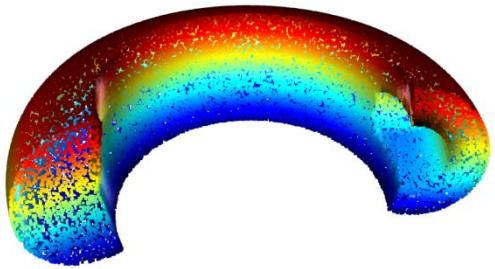
But, does it actually work?

Undamaged Parts





Damaged Part Scan



(hint to self – pass around samples now)

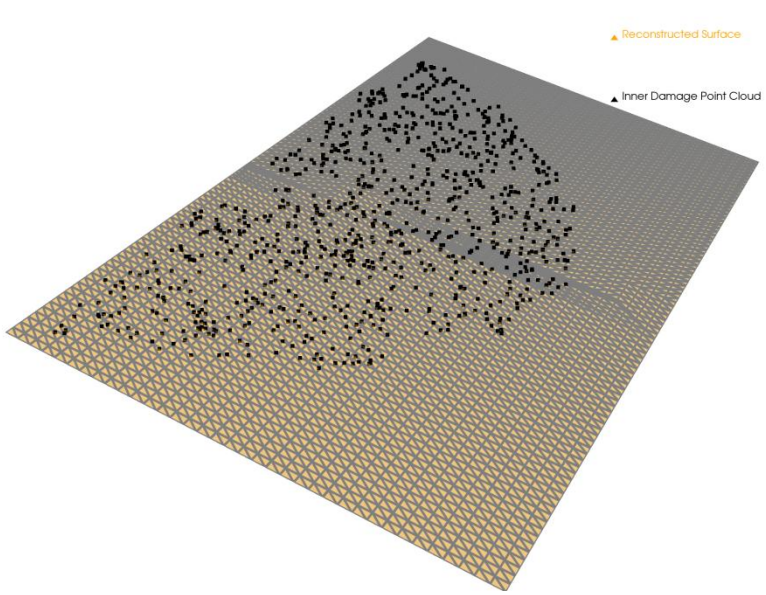
DamageEst

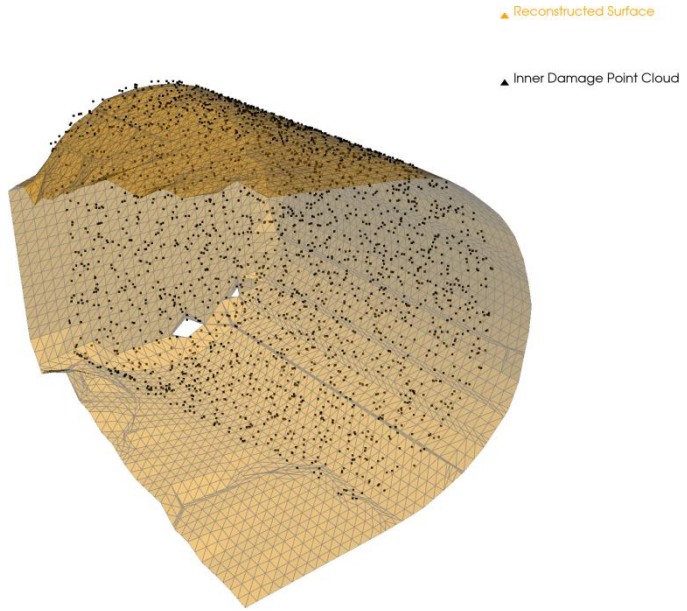
1. Get Inner Damage Surface Points

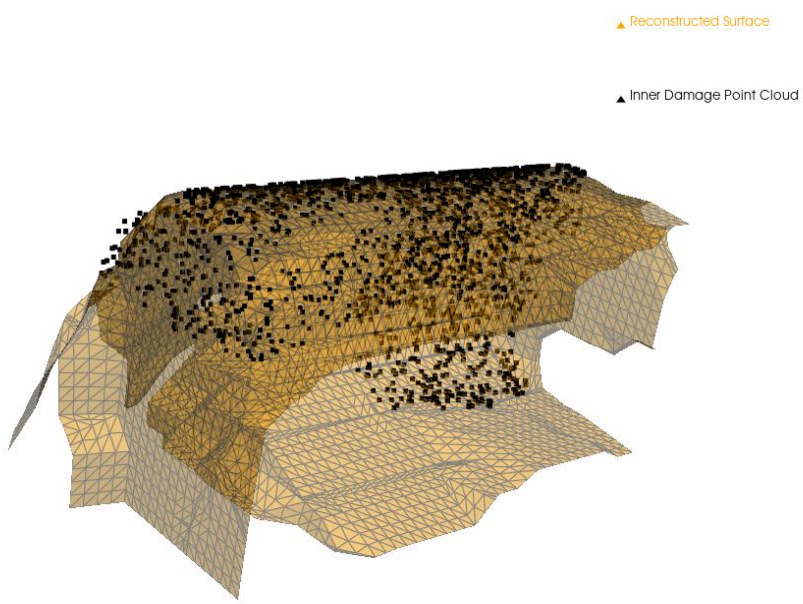
MITSUBISHI ELECTRIC
Changes for the Better

DamageEst

2. Reconstruct Inner Damage Surface



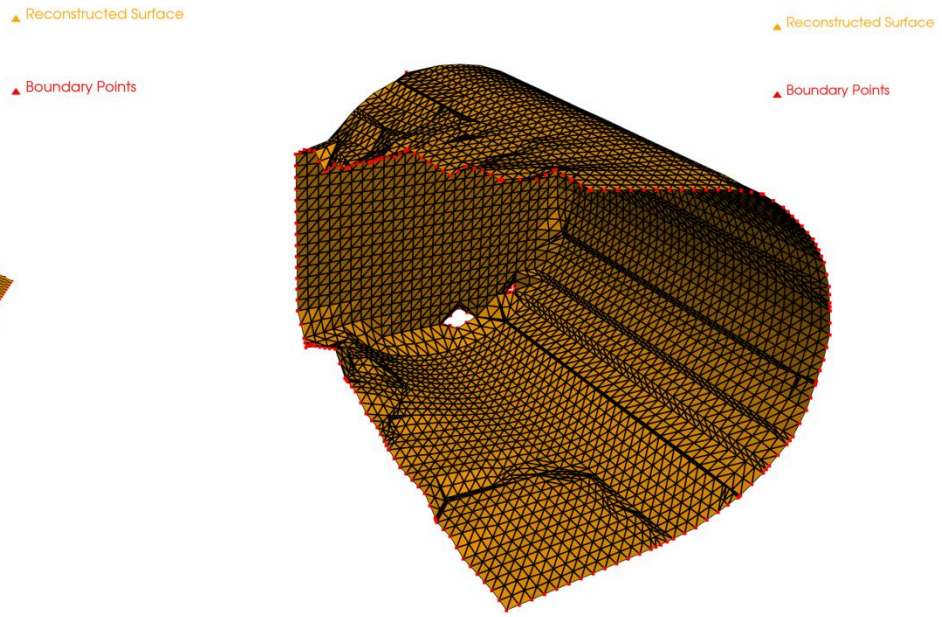


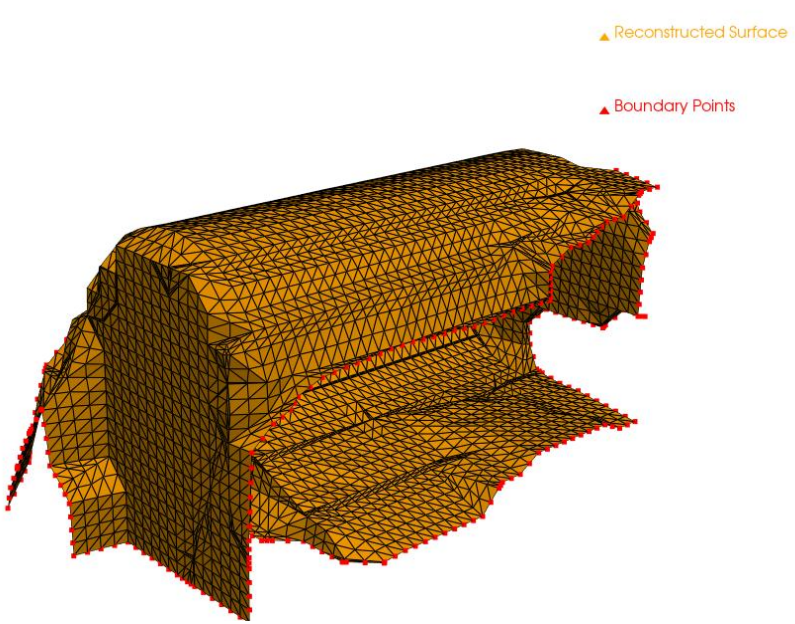


MITSUBISHI ELECTRIC
Changes for the Better

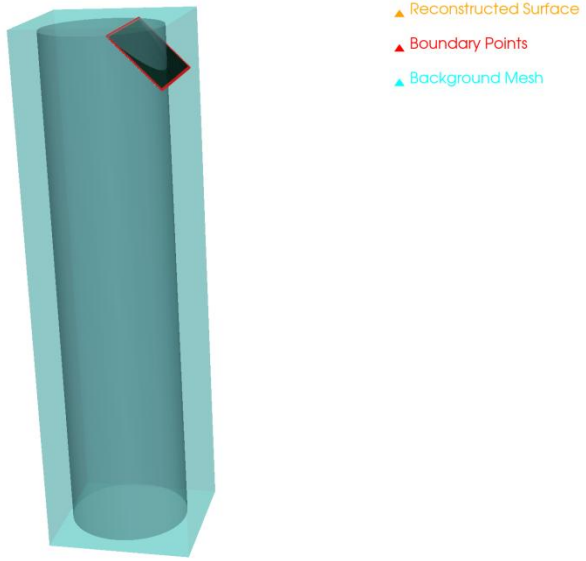
DamageEst

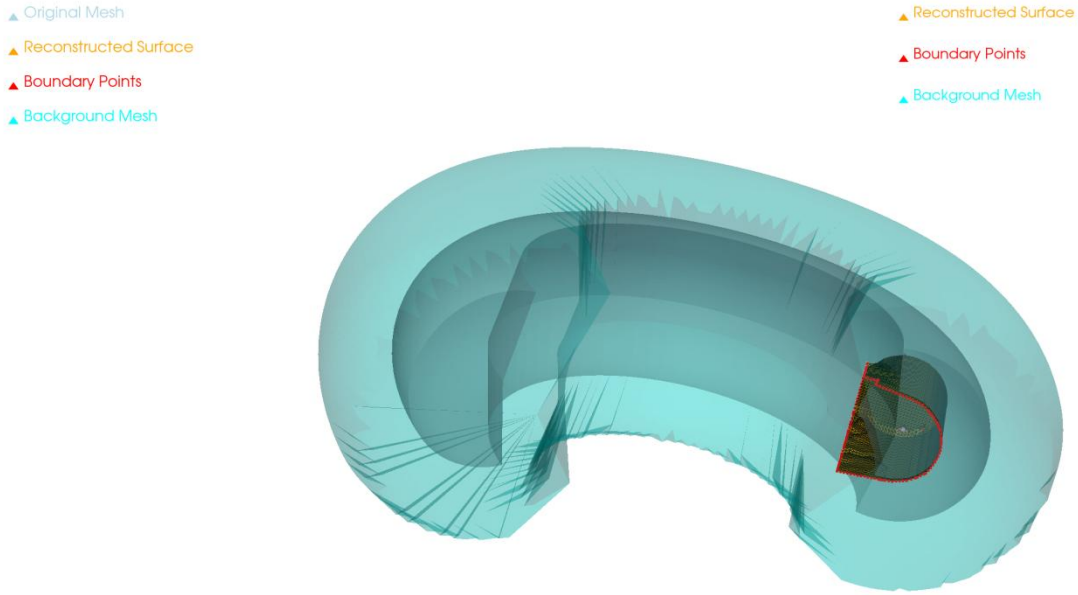
3. Find the Boundary of the Damage Surface

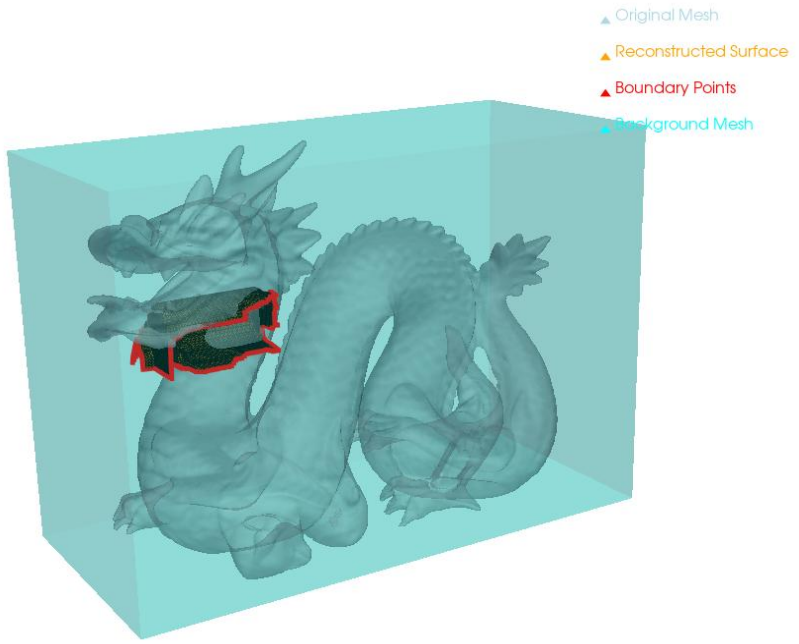




4. Create the Background Mesh



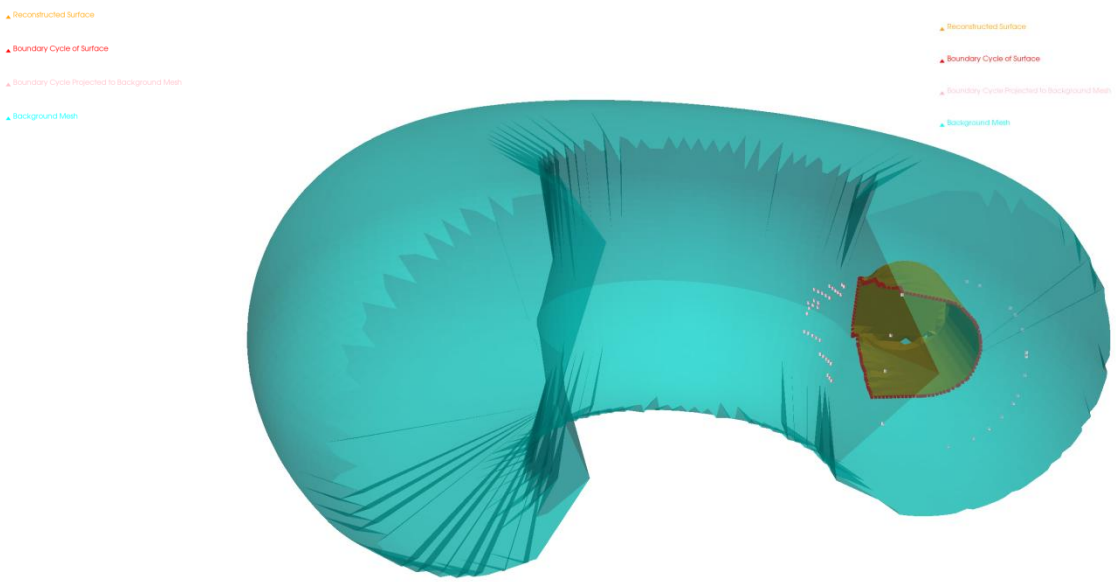


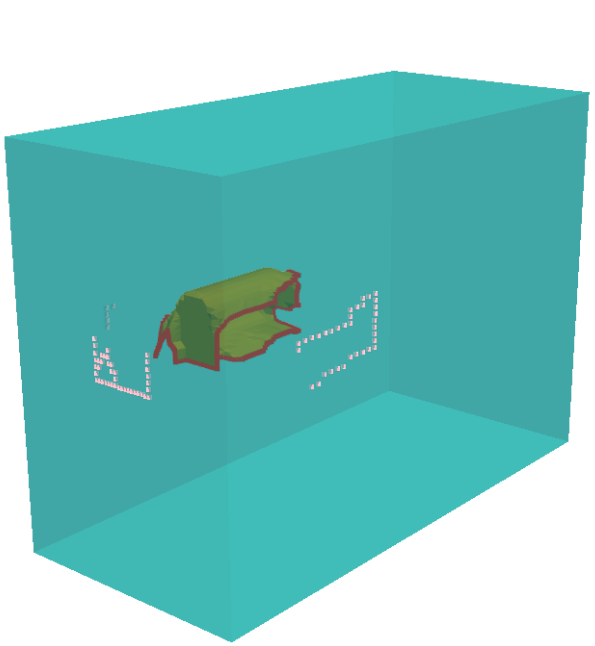


MITSUBISHI ELECTRIC
Changes for the Better

DamageEst

5. Push Boundary to the Background Mesh

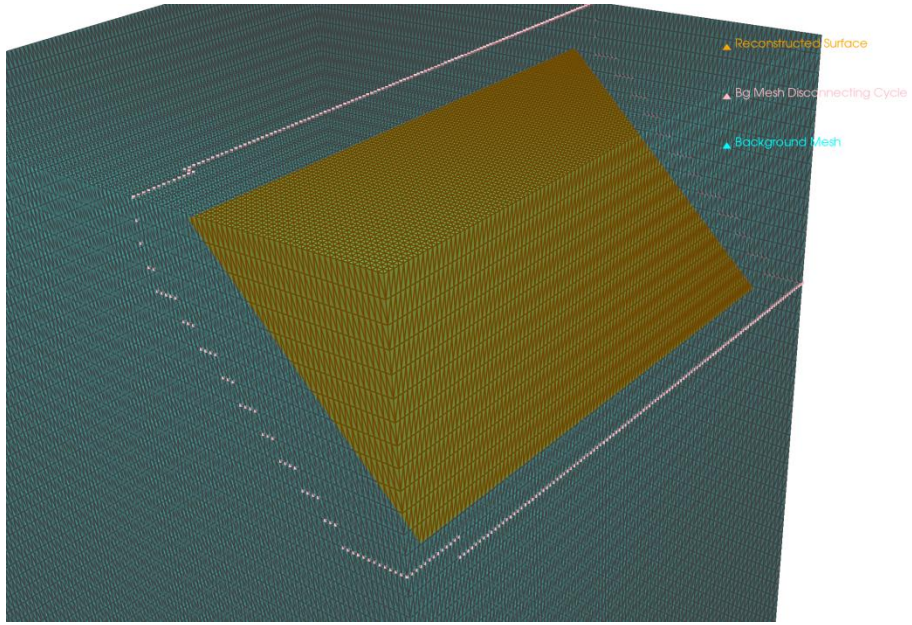


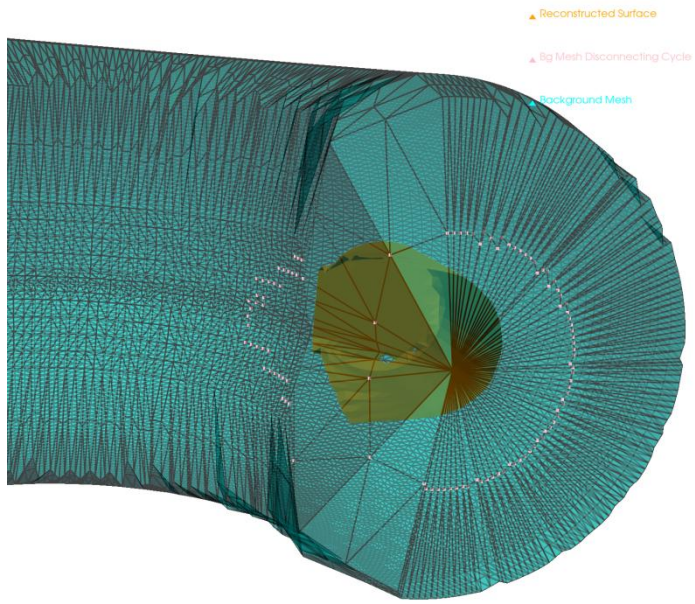


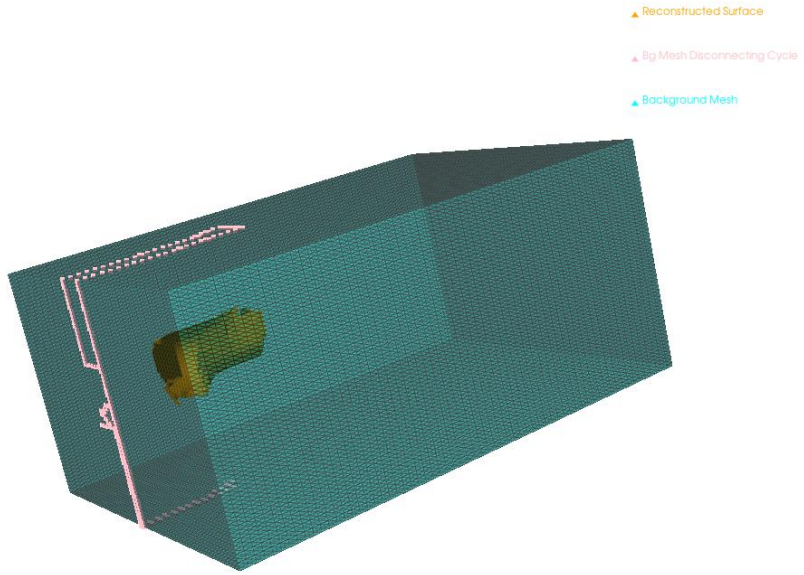
MITSUBISHI ELECTRIC
Changes for the Better

DamageEst

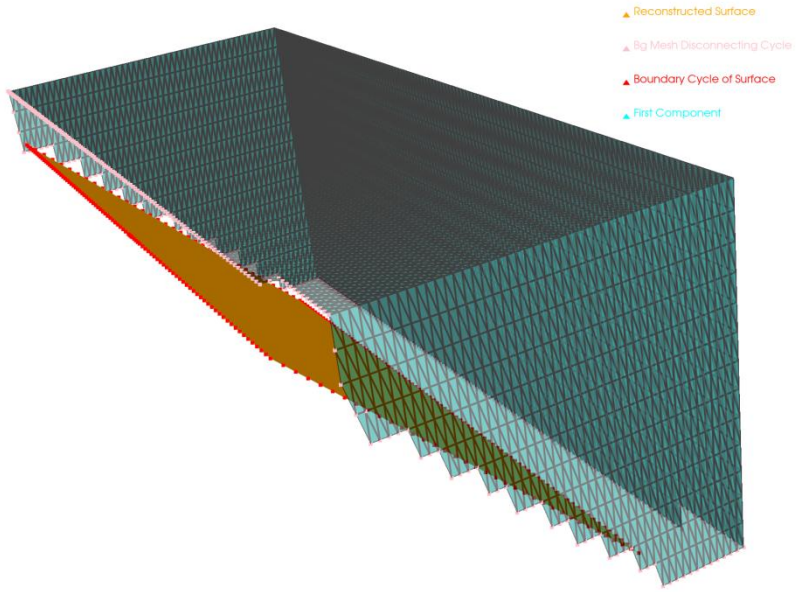
6. Connect Pushed Cycle

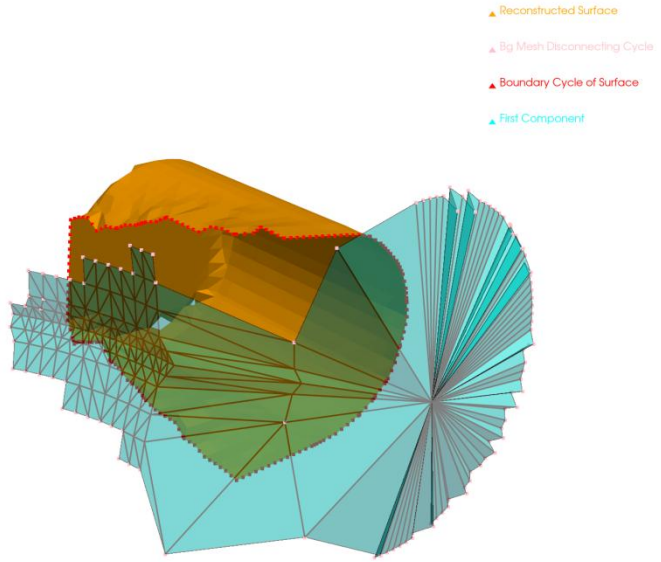


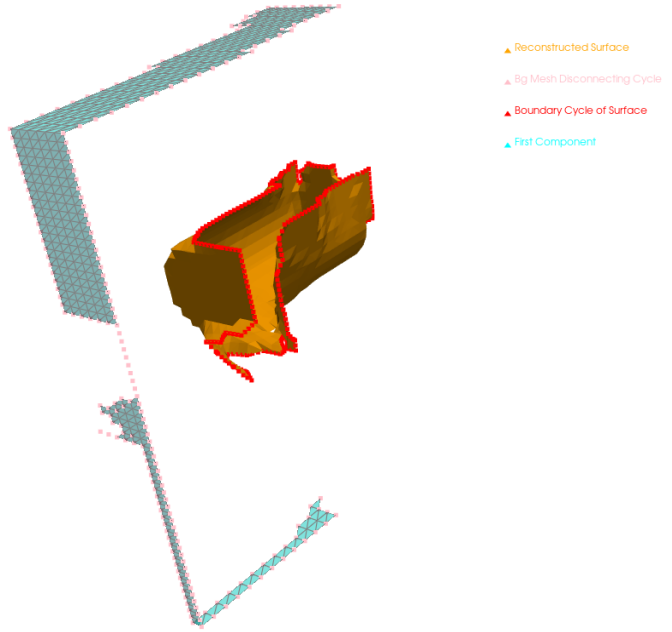




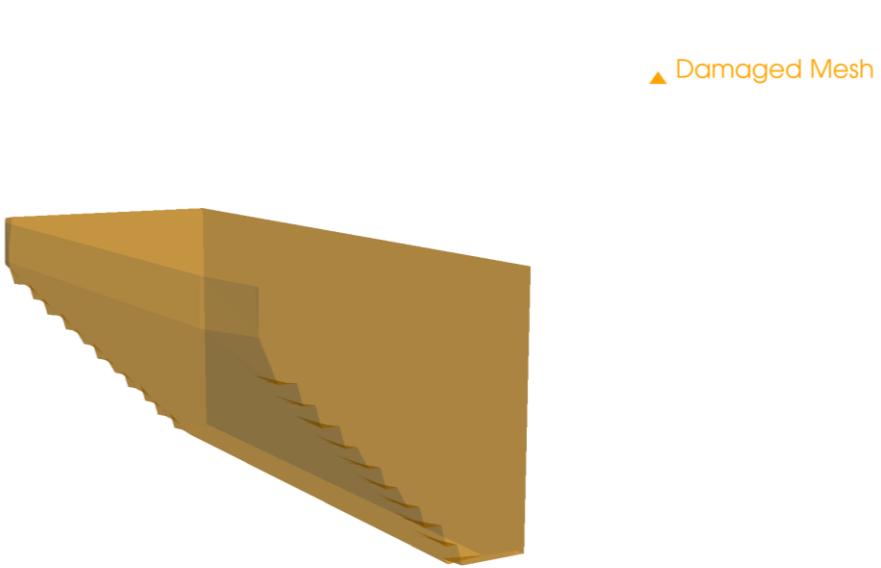
7. Split the Background Mesh

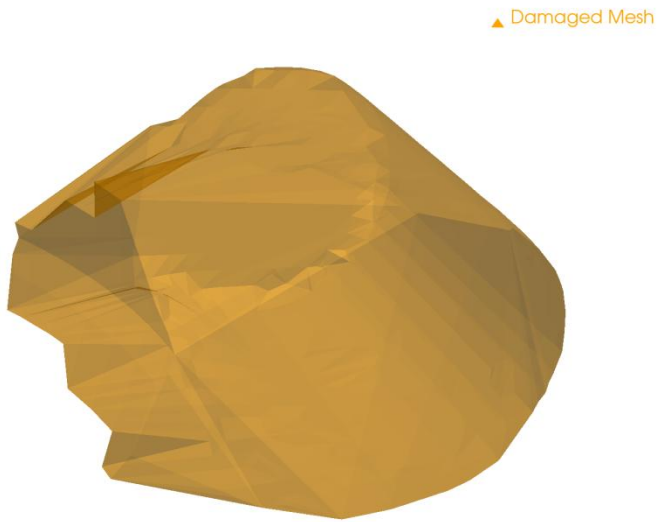


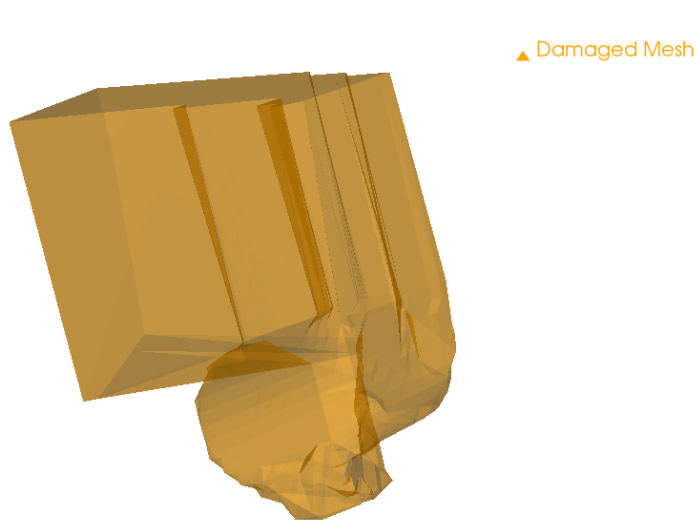




8. Connect to the Damage Surface



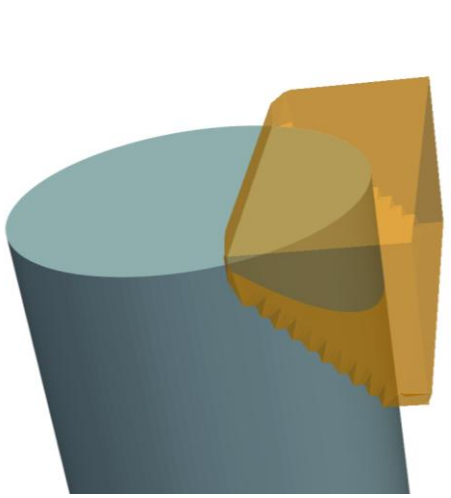


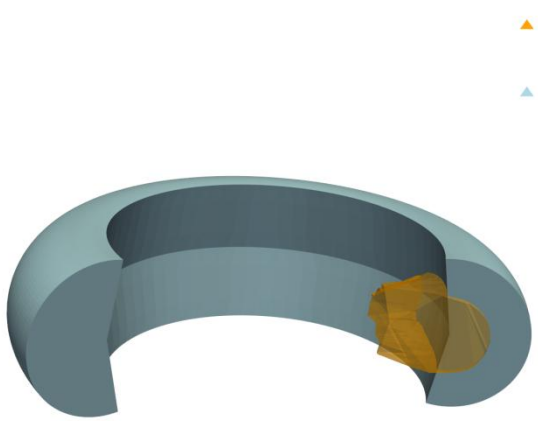


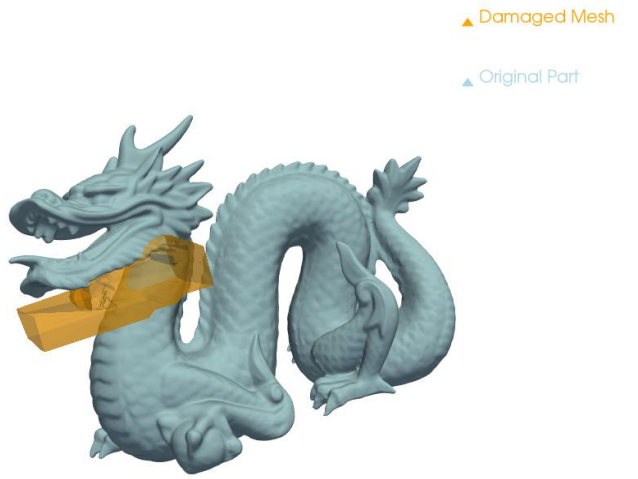
MITSUBISHI ELECTRIC
Changes for the Better

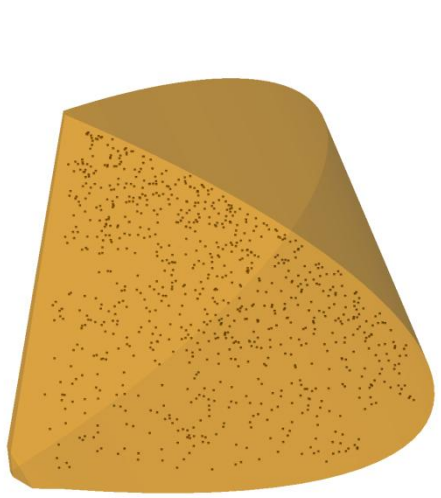
DamageEst

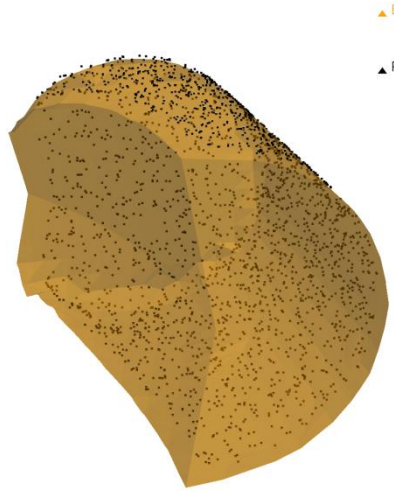
9. Take Boolean Intersection (find overlapping volume)

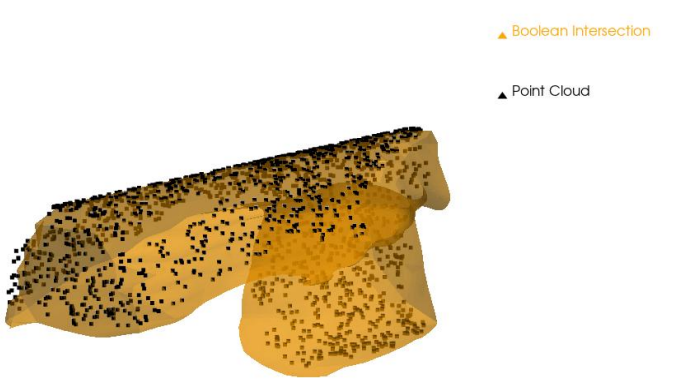






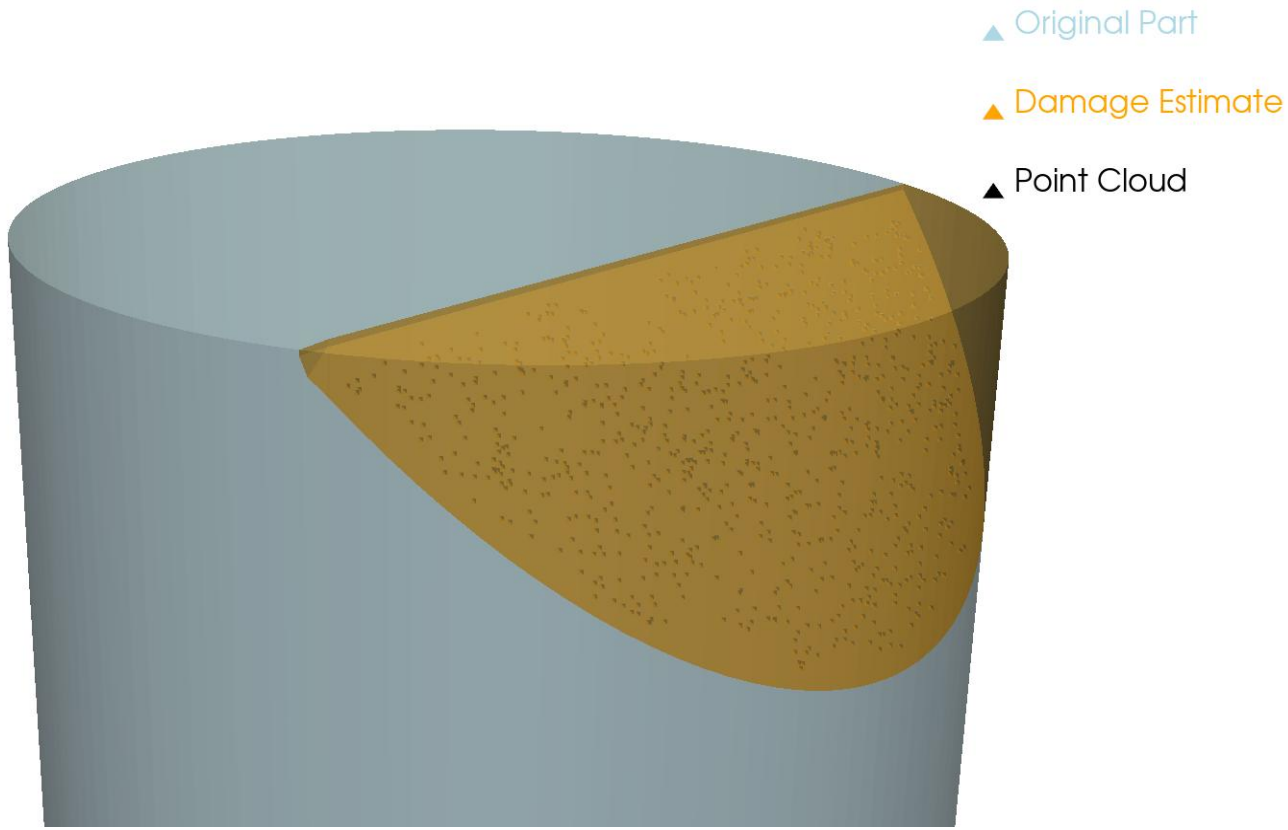






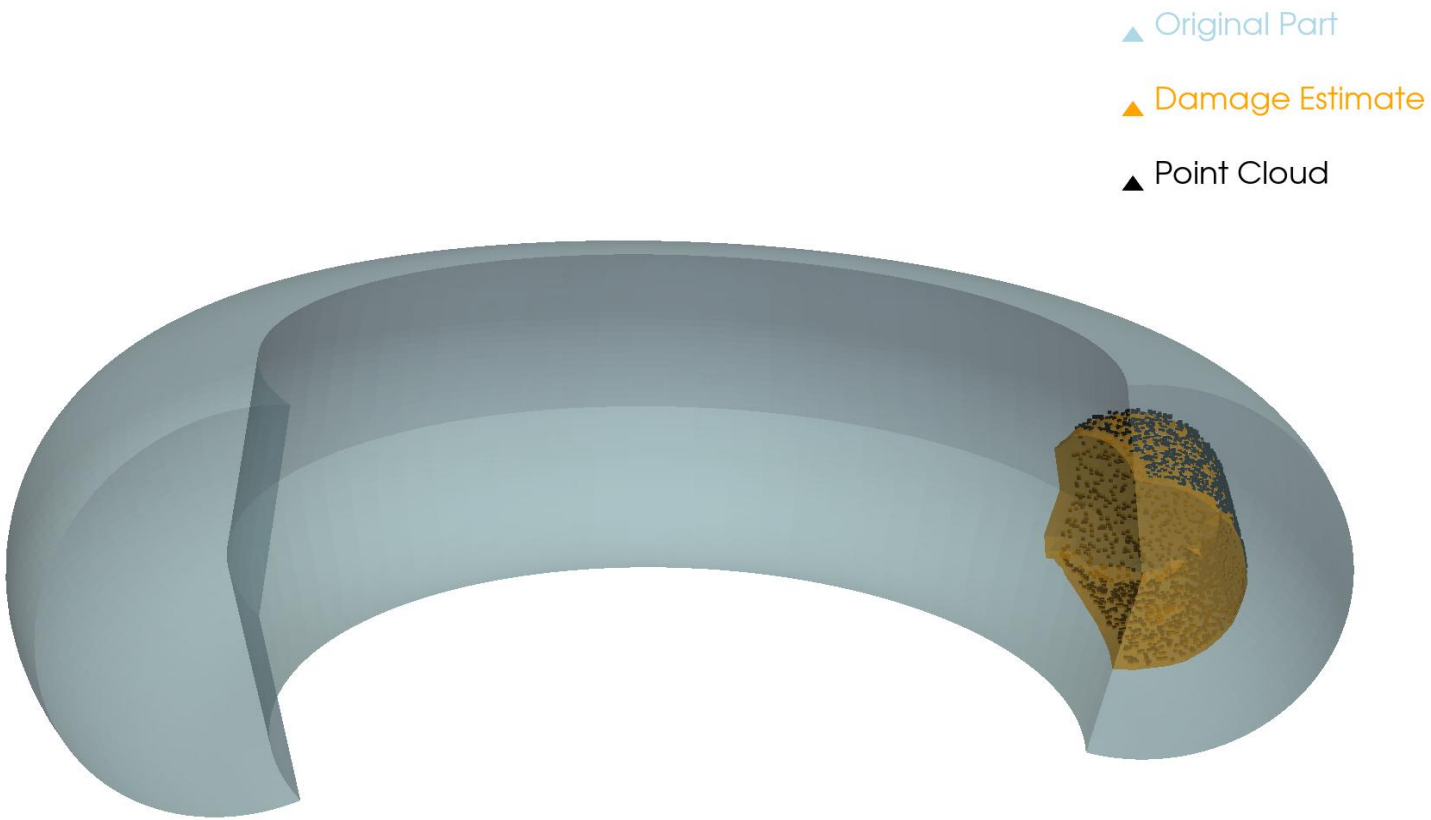
DamageEst

Results



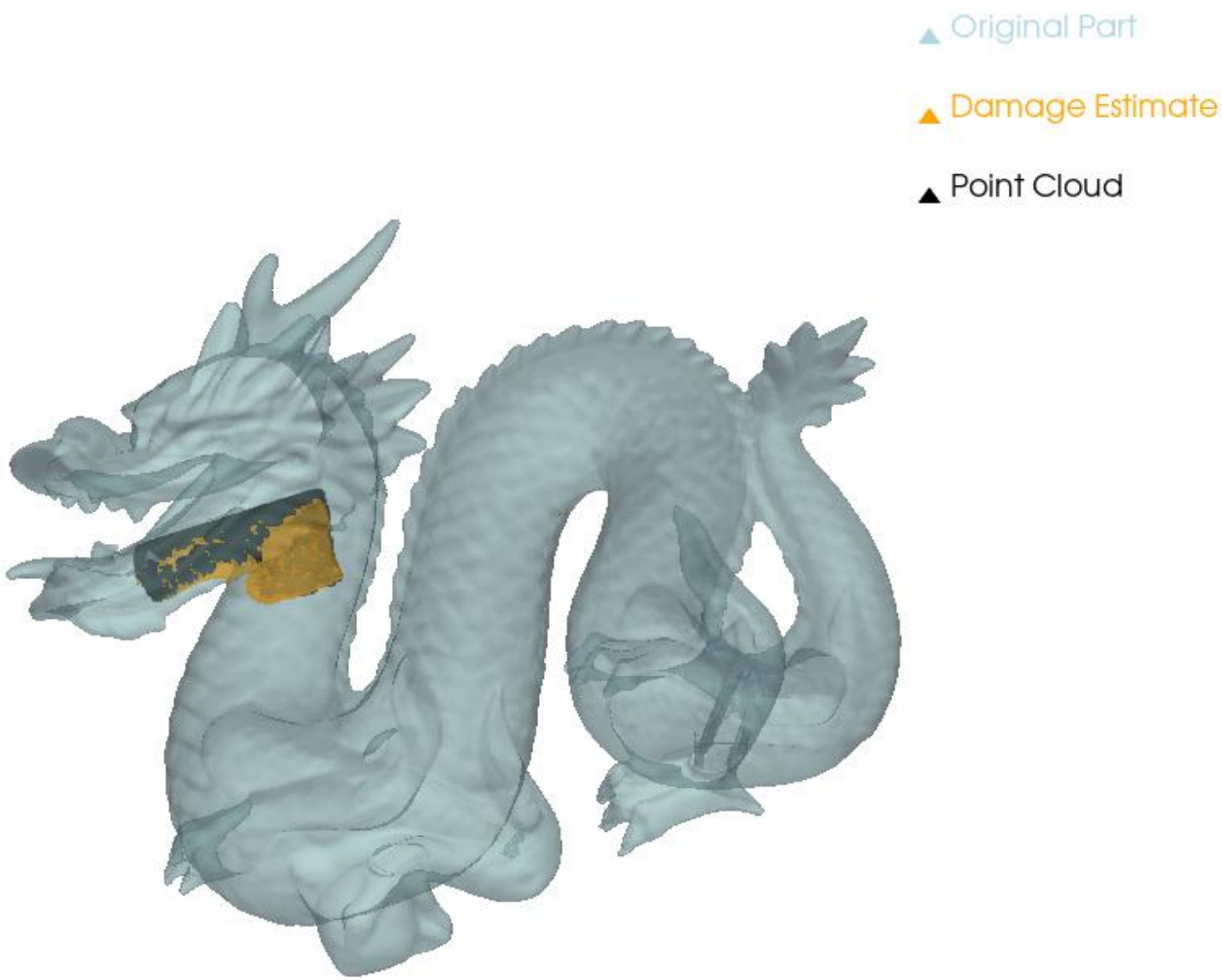
Pushout/Split Method

Results



Pushout/Split Method

Results



MITSUBISHI ELECTRIC
Changes for the Better

Comparison to State of the Art (Method 1, “Li et al”)

- Li’s method compares a point cloud scan of a damaged part to an undamaged part.
- Li’s method takes far longer to run then ours.
- Li’s method is iterative, ours isn't.

An integrated approach of reverse engineering aided remanufacturing process for worn components

Lingling Li[a], Congbo Li[a,*], Ying Tang[b], Yanbin Du[c]

[a] State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China
[b] Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028, USA
[c] Chongqing Key Laboratory of Manufacturing Equipment Mechanism Design and Control, Chongqing Technology and Business University, Chongqing, China

CrossMark

Comparison of Methods				
Algorithm	**Part**	**Number of Points**	**Number of Iterations**	**Runtime (s)**
Damage Est	Torus	50,000	1	12
Damage Est	Dragon	100,000	1	7
Standard ICP	Model P	30,000	54	2415
Modified ICP	Model P	30,000	30	1713

MITSUBISHI ELECTRIC
Changes for the Better

Comparison to State of the Art (Method 1, “Li et al”)

- Li’s method assumes that only "small" damage depths can be repaired due to the maximum depth of damage constraint.
- Our method can estimate large missing components without considering a maximum depth constraint.
- Li’s method detects additive and subtractive estimates at the same step. Ours considers these estimates separately.

a) Maximum depth of damage

With registration result of two models, the distance (damage depth) between corresponding points of the two models and the maximum distance (maximum damage depth H) between corresponding points can be determined as shown in Fig. 2. The dotted line represents the outline of original non-effective part and the circles represent the points on the original outline. The solid line shows the outline of the damage parts and the cross points represent the points on the damage outline.

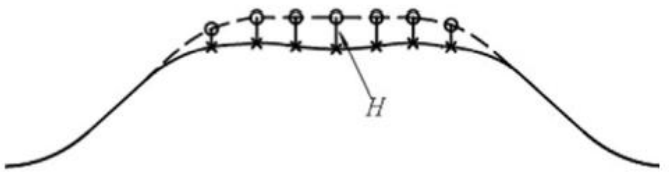


Fig. 2. Schematic diagram of the damage depth.

MITSUBISHI ELECTRIC
Changes for the Better

Comparison to State of the Art (Method 1 “Li et al”)

- Li’s method classifies the vertices as defective, maps any undamaged vertices to the nearest undamaged vertex in the original part, then attempts to recreate the entire surface of both the damaged and undamaged clouds. Finally, they take the Boolean operations.

- We use the background mesh to help counter noise and inaccuracies in the point cloud scans. In addition, this helps with inaccuracies in the reconstruction process.

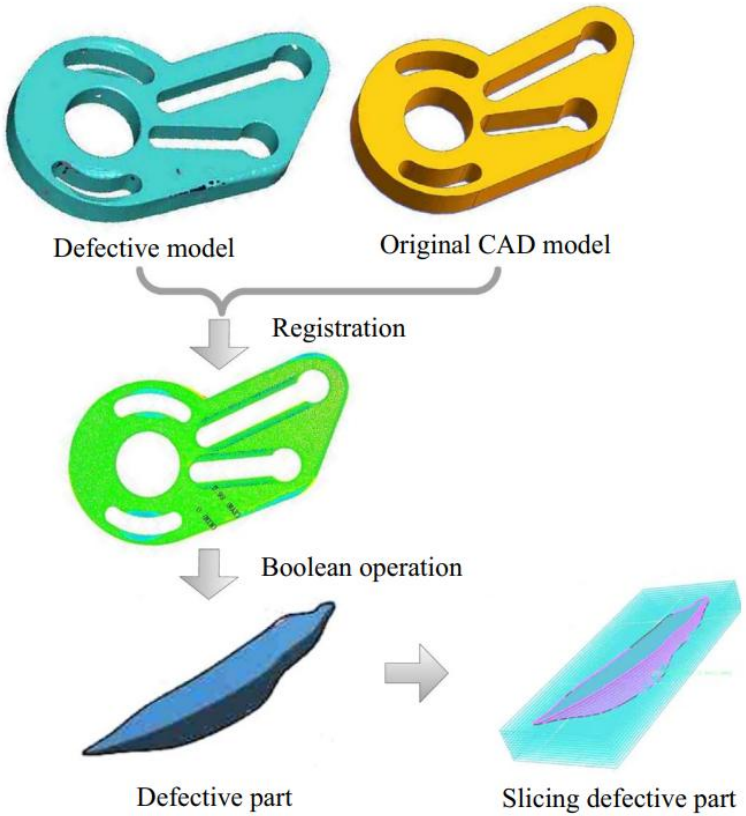


Fig. 25. Additive restoration simulation of CGAP.

MITSUBISHI ELECTRIC
Changes for the Better

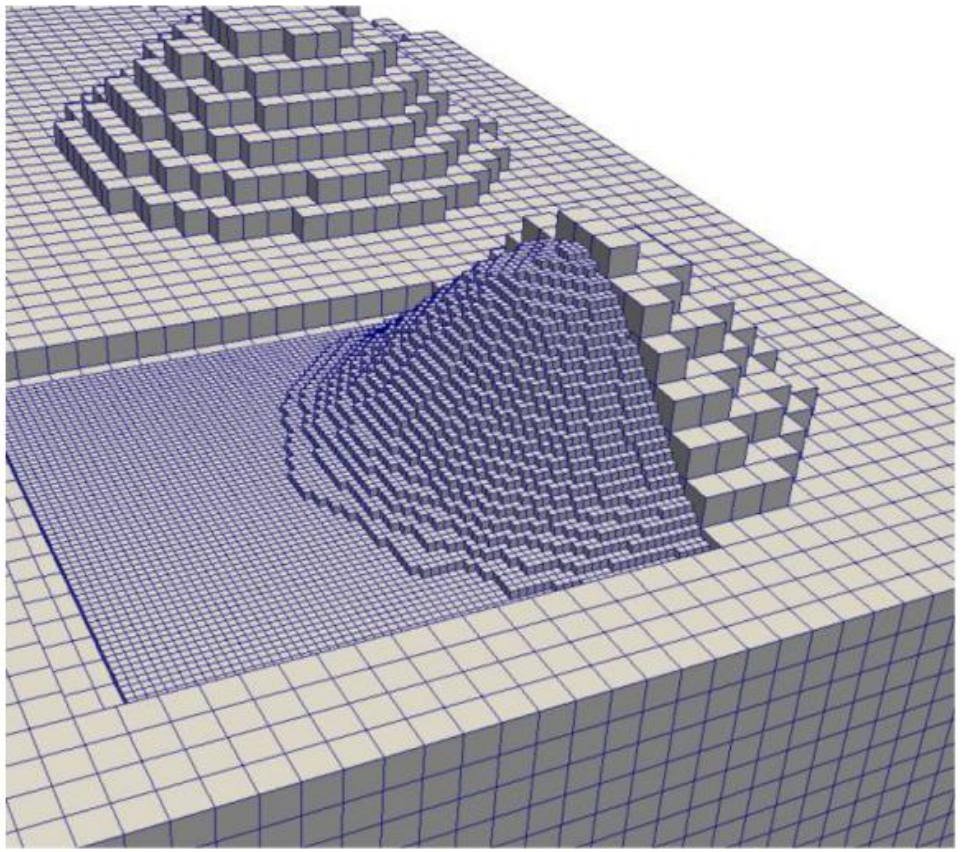
Comparison to State of the Art (Method 2, “Perini et al”)

Additive manufacturing for repairing: from damage identification and modeling to DLD

Matteo Perini and Paolo Bosetti
Department of Industrial Engineering, University of Trento, Trento, Italy, and
Nicolae Balc
Department of Manufacturing Engineering, Technical University of Cluj-Napoca, Cluj-Napoca, Romania

- Perini’s method uses an octree to estimate the damage volumes. They estimate the damage volume and original part as cubic blocks.
- We only consider the surfaces of the damaged part and the original part.

Figure 8 Octree depth can be increased for specific sub-volumes



MITSUBISHI ELECTRIC
Changes for the Better

Comparison to State of the Art (Method 2, Perini et al)

- For less precise estimates, Perini's method works well. As precision increases, the computation scales exponentially (note the first plot is logarithmic).
- This scaling limits the precision for larger parts if the runtime is to be kept reasonable. This is true even for simpler meshes.
- The plots on the right are only for building the octree. Their implementation of the Boolean difference is also an exponential algorithm. (But the paper claims their implementation only takes a few seconds to run in practice)

Figure 7 Time used to compute the octree of a mesh changing the number of grid subdivisions

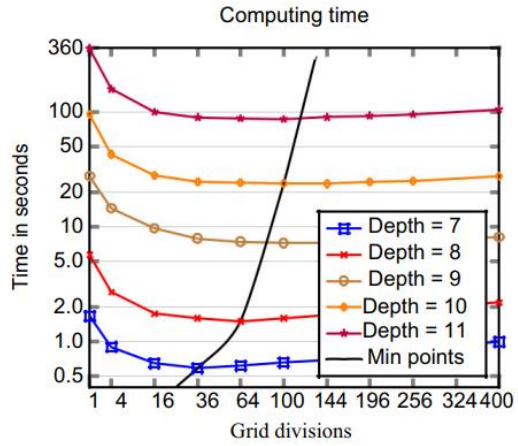
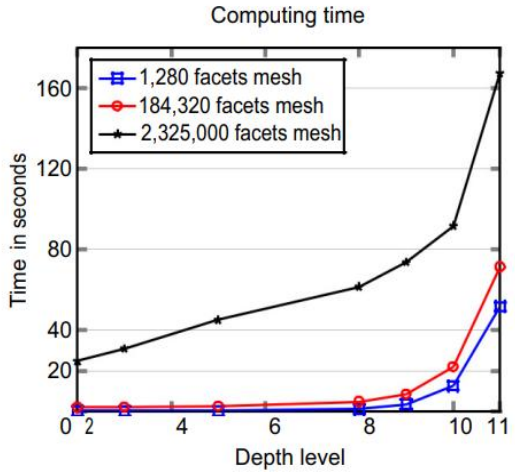


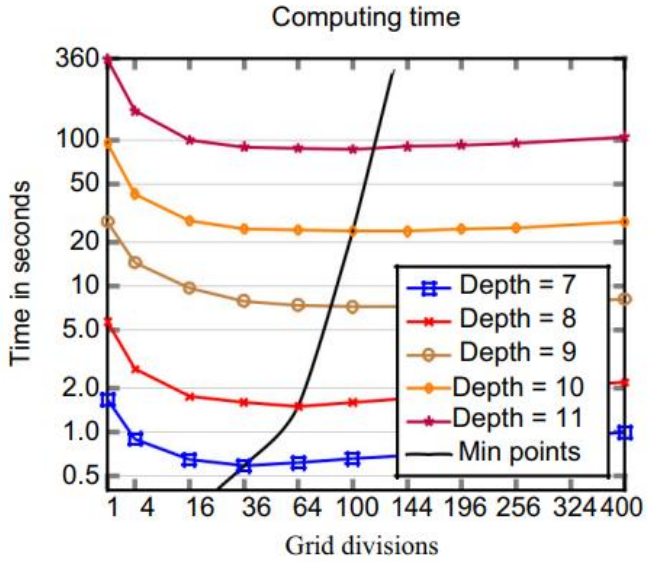
Figure 11 Time used to build the octree of three different meshes changing the depth level reached

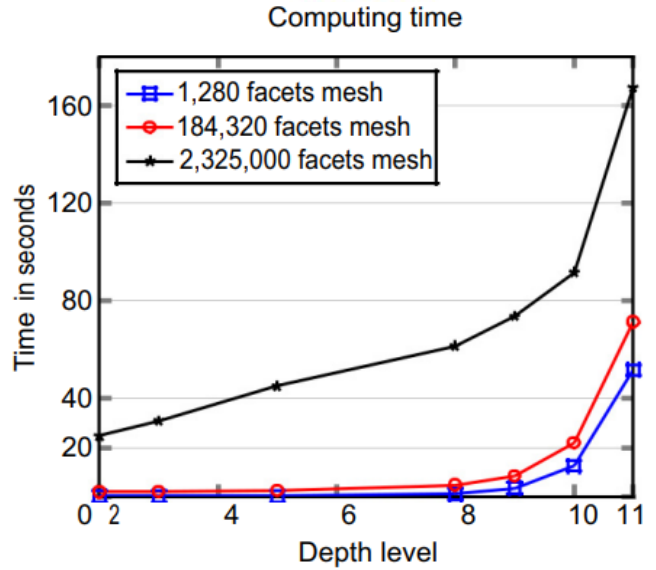


MITSUBISHI ELECTRIC
Changes for the Better

Comparison to State of the Art (Method 2, Perini et al)

- Our method has the maximum allowable precision (up to the point cloud scan's resolution and the float precision).
- Our entire pipeline works on meshes with similar complexity to their meshes, but without the runtime required for their method at high precision.
- Our bottleneck comes from the number of intersections of the background mesh and original STL, rather than the number of triangles.

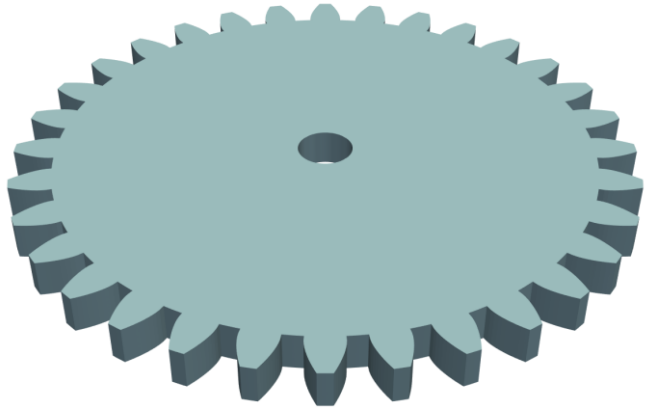




MITSUBISHI ELECTRIC
Changes for the Better

Comparison to Method 2 (Gear)

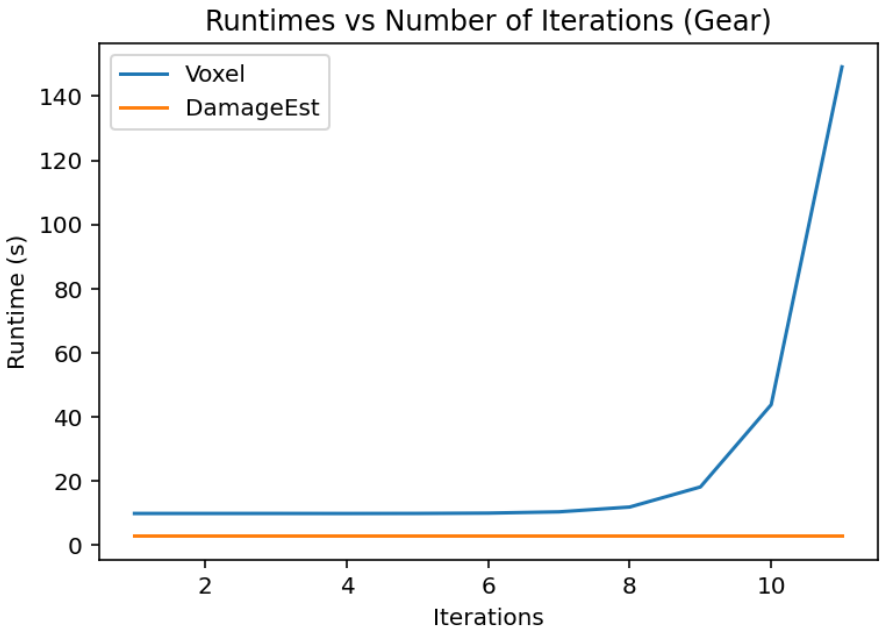
Comparison of Methods on Gear Damage Estimation	
Method	**Runtime (averaged over 10 runs)**
Ours (“Damage Est”)	2.9 s
Voxel (Depth 7)	10.4 s
Voxel (Depth 9)	18.2 s
Voxel (Depth 11)	149.1 s

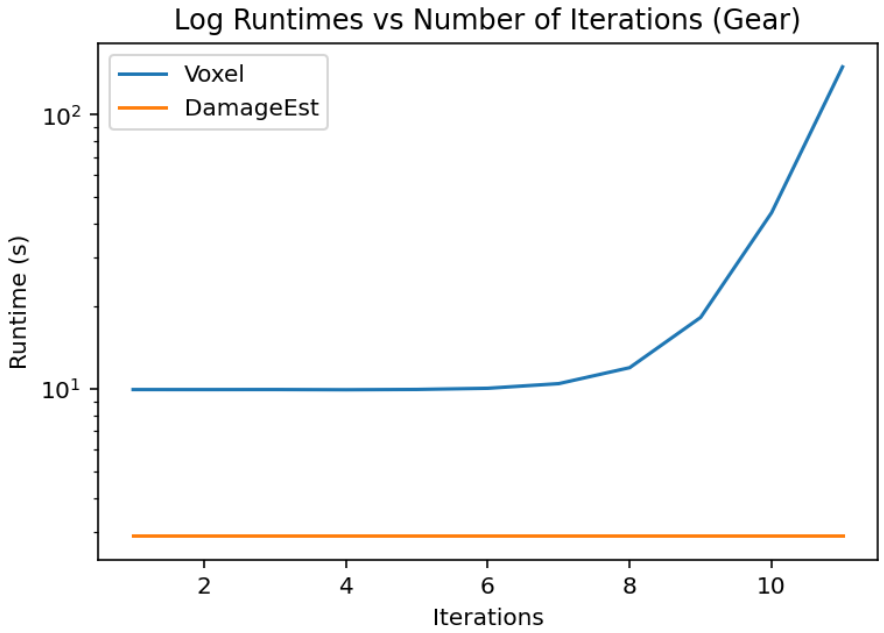


Gear

Damaged Point Cloud Scan

Comparison to Method 2 (Gear)

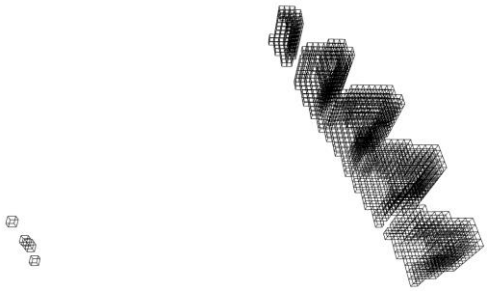




MITSUBISHI ELECTRIC
Changes for the Better

Comparison to Method 2 (Gear)

Damage Est



Voxel (Depth 7)

Voxel (Depth 9)

Voxel (Depth 11)

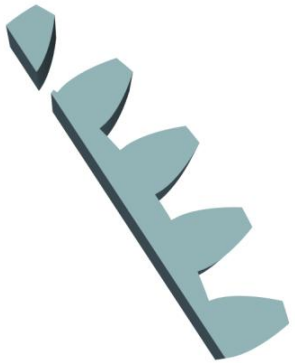
MITSUBISHI ELECTRIC
Changes for the Better

Comparison to Method 2 (Gear)

- For the voxel method, it is necessary to set a threshold when determining if a voxel is contained in the damage volume. High thresholds lead to underestimation, low thresholds lead to artifacts on sharp corners of the original STL. These may require post processing.

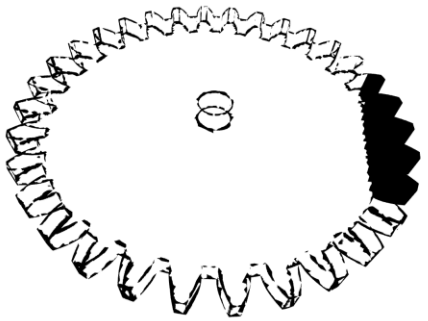
- DamageEst leads to very few artifacts and preserves sharp features in the original STL.

Damage Est



Voxel (Depth 11, High Threshold)

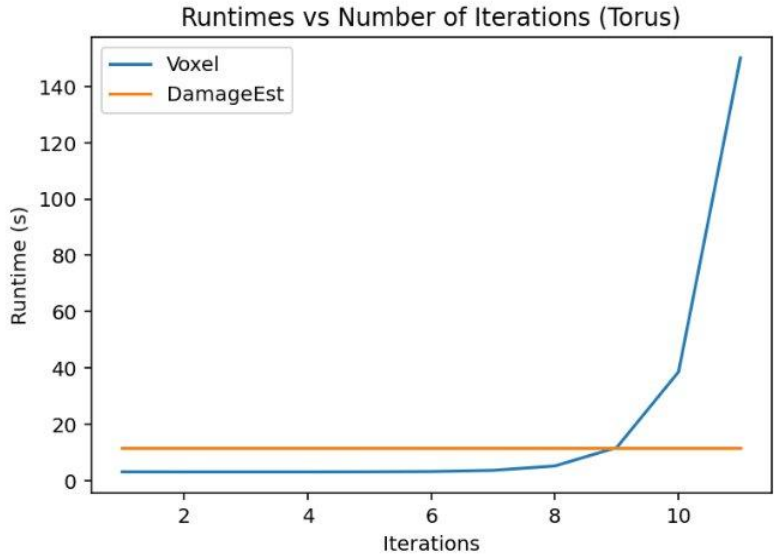
Voxel (Depth 11, Low Threshold)

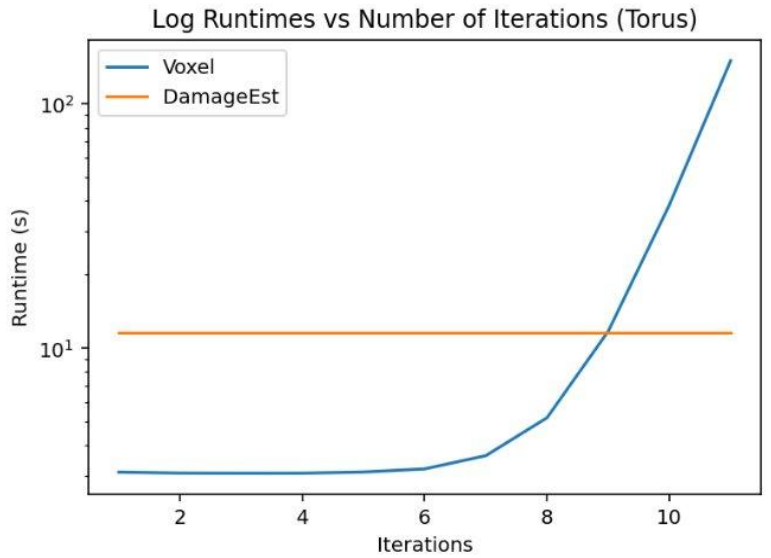


MITSUBISHI ELECTRIC
Changes for the Better

Comparison to Method 2 (Perini et al fixing the Torus)

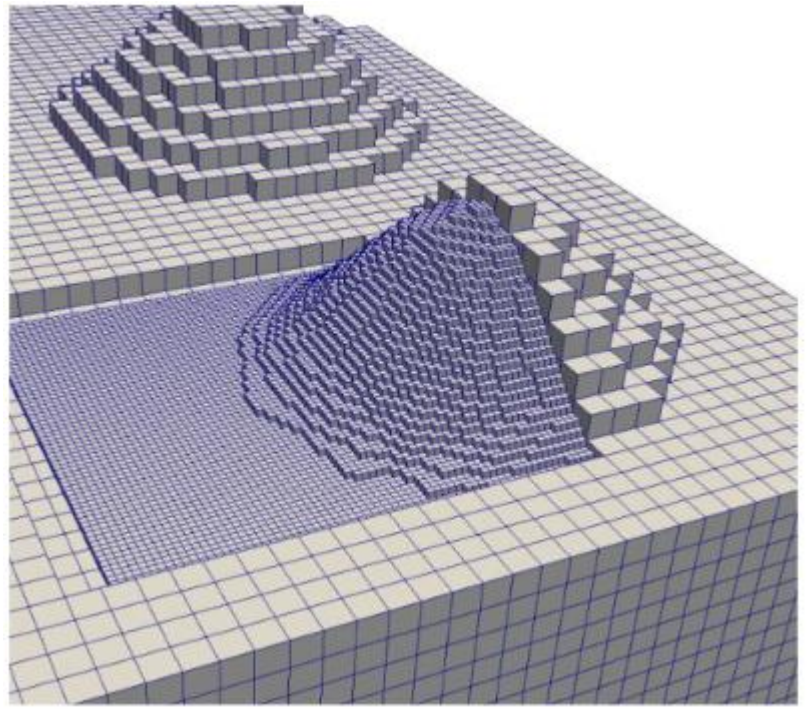
- We compared the runtimes DamageEst to the voxel method for a second (Torus) part. This is the same torus part we showed in our previous presentation.
- The runtimes are about the same at 9 iterations of the voxel method.
- Even with more complicated parts (involving more complicated Boolean operations), DamageEst gives high quality reconstructions without exponential scaling.





Existing Best Method

- The existing best method (Perini et al) uses voxels to estimate the damaged and undamaged regions of the part.
- This method is exponential in both memory and time, but yields accurate estimates in reasonable runtimes for small parts.
- Even with adaptive octree, curved surfaces and boundaries can increase runtimes in practice.
- With larger octree depths, the estimate becomes more accurate.



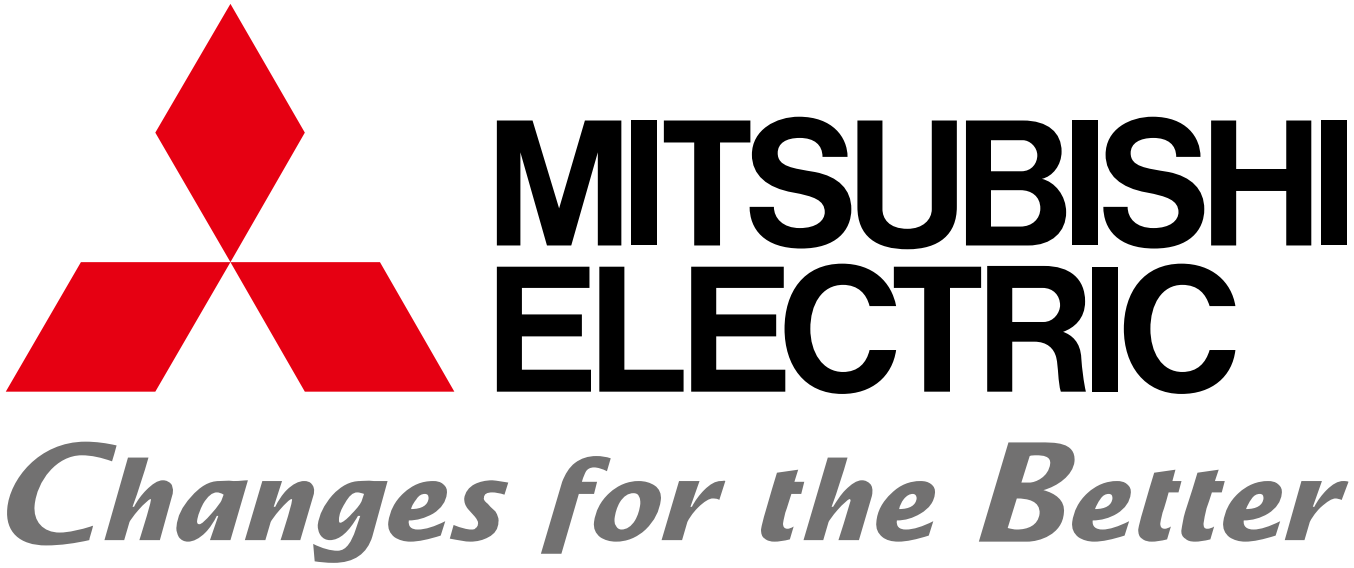
Perini 2020, Figure 8, Diagram, Rapid Prototyping Journal, Vol 26, Number 5, pg 930

MITSUBISHI ELECTRIC
Changes for the Better

Thank you I!

Questions ?

Discussion ?


MITSUBISHI
ELECTRIC
Changes for the Better