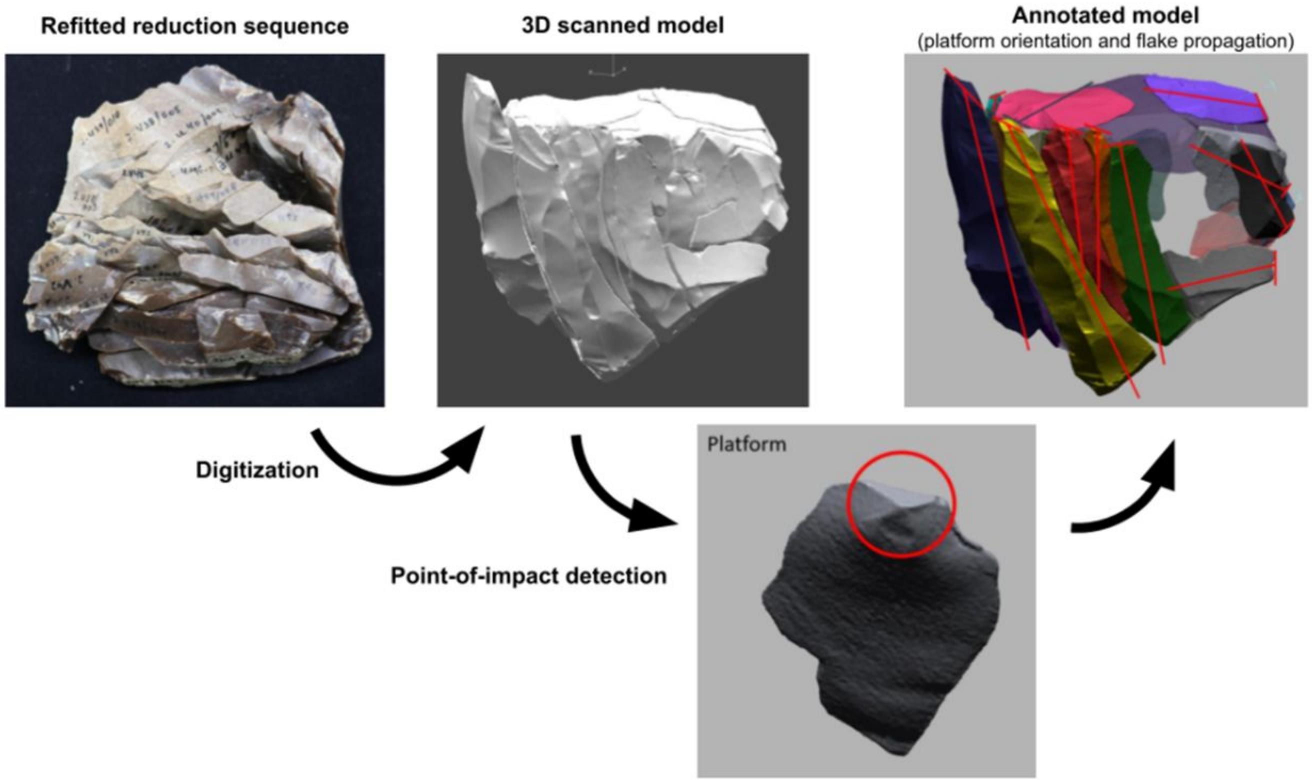
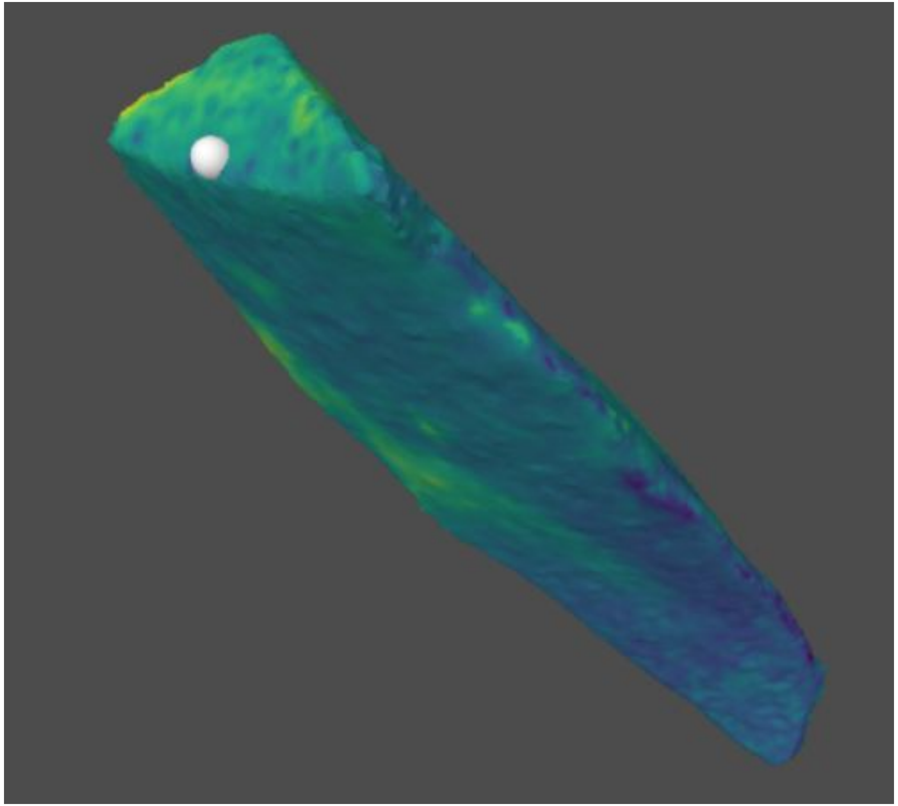
Geometric deep learning for quantitative analysis of stone tool reduction sequences

The Problem



Refitted reduction sequence
3D scanned model
Annotated model
(platform orientation and flake propagation)
Digitization
Point-of-impact detection
Platform

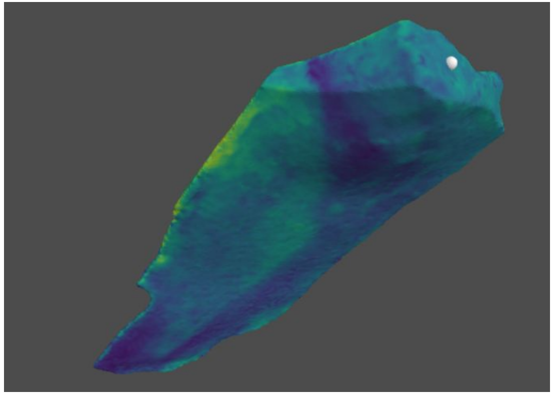
The Problem



Surface Segmentation

Surface Segmentation

We want to identify larger super faces of the flakes.



Therefore we perform surface segmentation on a given flake.

Our Approach Clustering

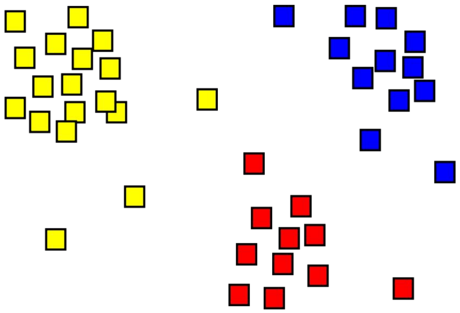
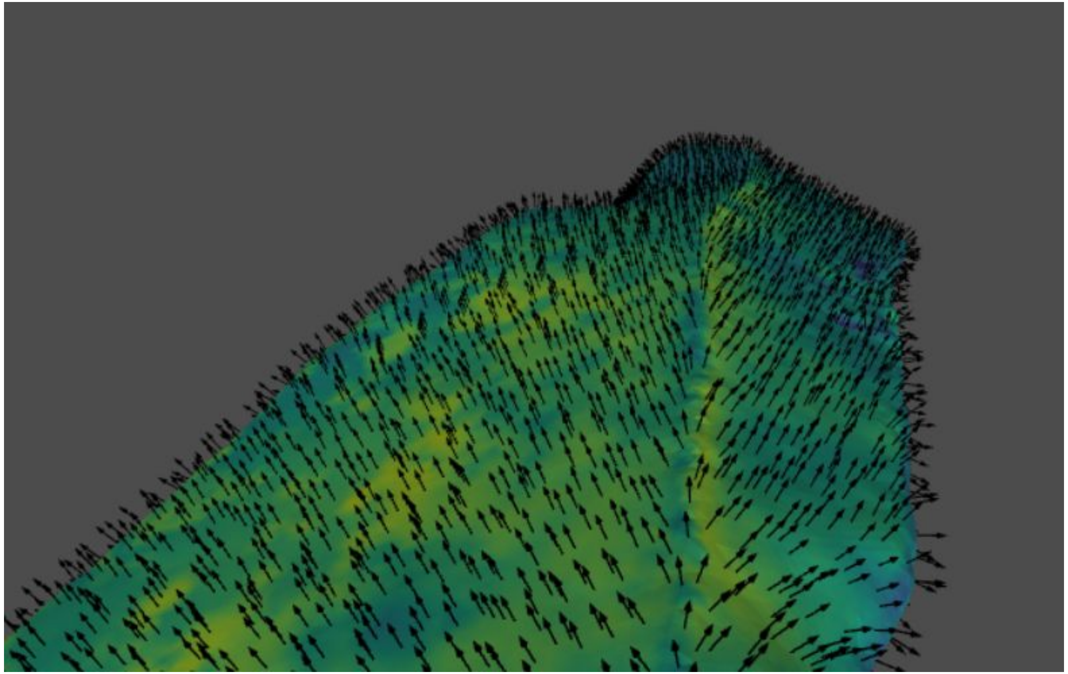


Image from https://en.wikipedia.org/wiki/Cluster_analysis

We cluster the points of the mesh in order to divide the object into surfaces

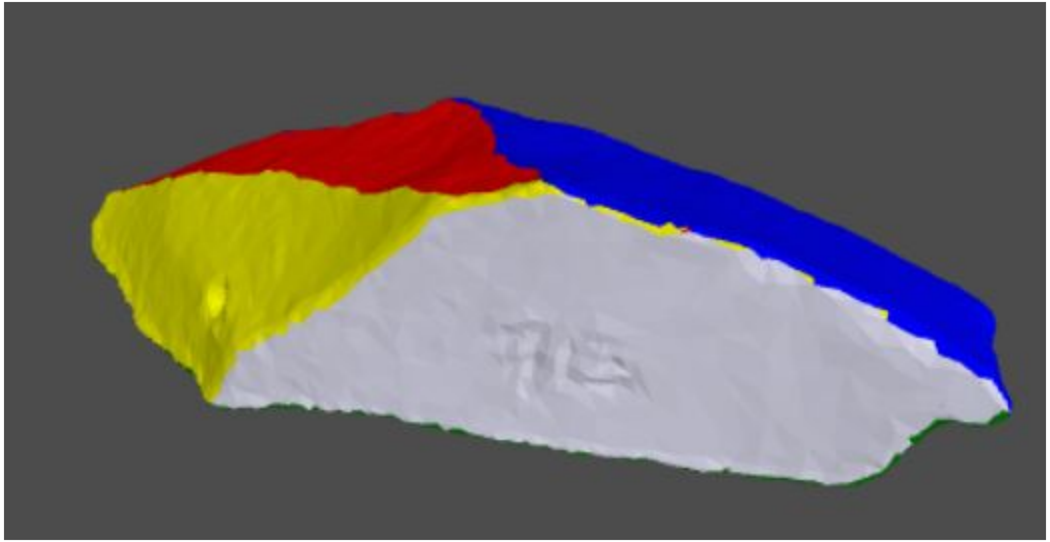
K-Means on the Surface Normals

A first simple Idea is to perform K-Means Clustering on the point normals.



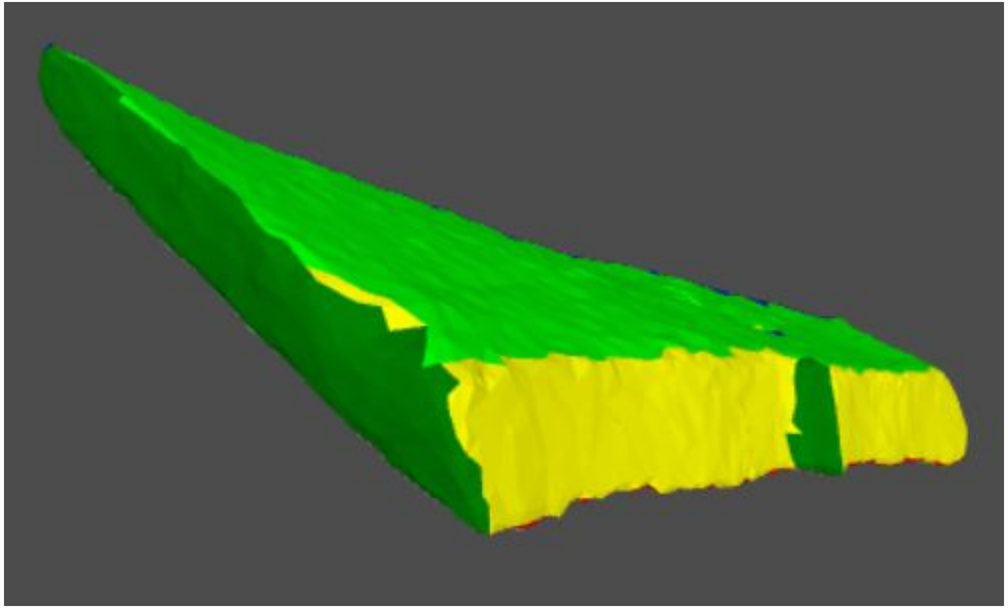
K-Means on the Surface Normals

With this very simple (and computationally cheap) approach we achieve the following result.



K-Means on the Surface Normals

But this simplicity comes at a cost: we lose the local information.

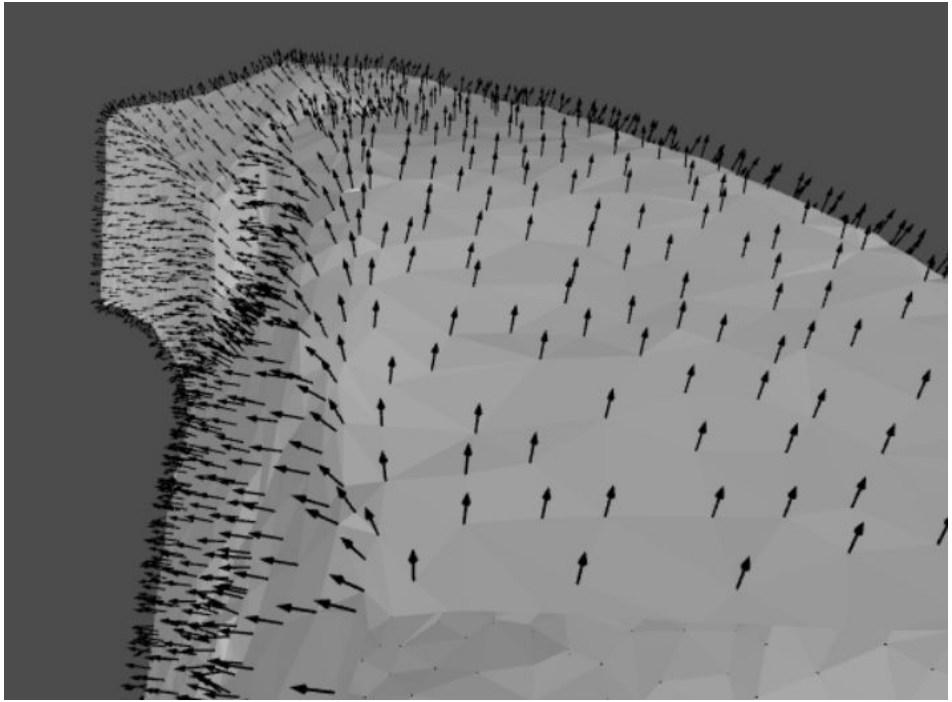


Spectral Clustering

In order to incorporate spatial locality we use spectral clustering.

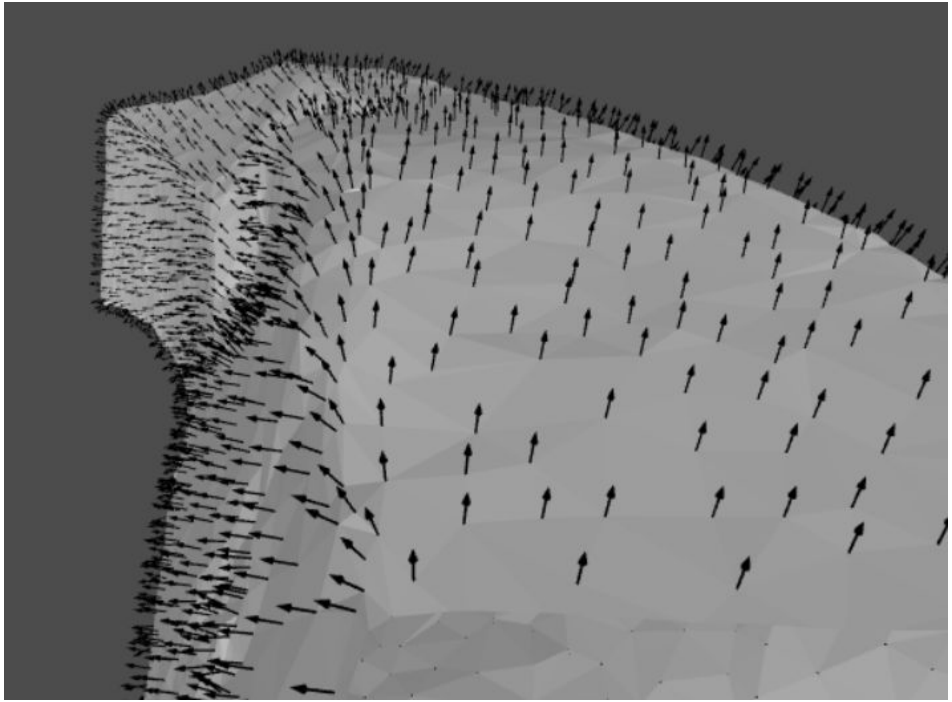
Spectral Clustering

We first build a graph given by the mesh structure.



Spectral Clustering

The edges are weighted by the angle between two normals.



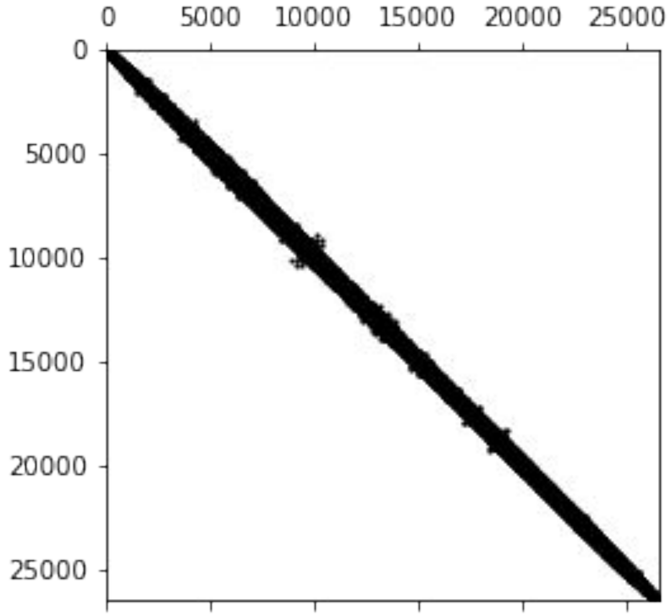
$\exp(-\delta\varphi)$

φ=angle between vertex normals

δ= weighting factor

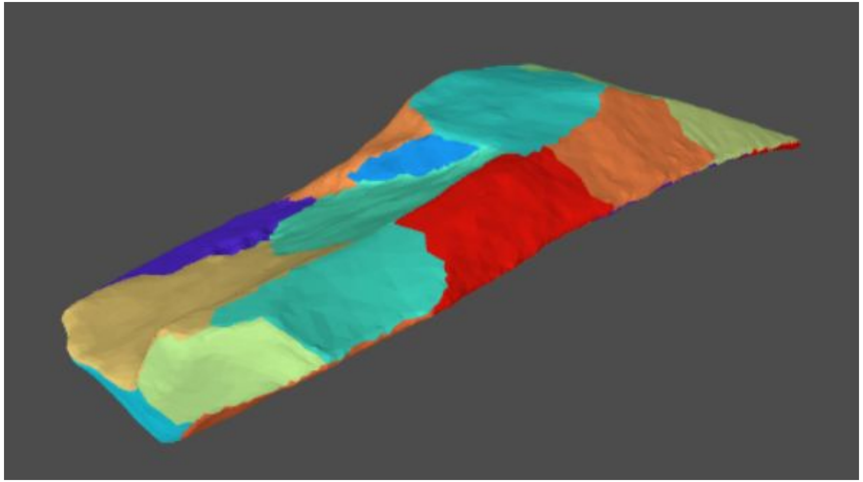
Spectral Clustering

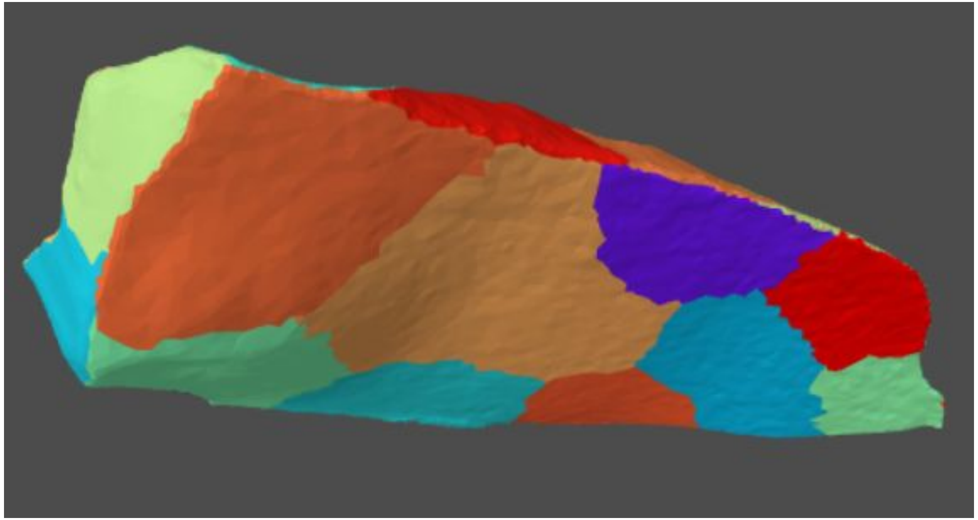
This constructs a sparse affinity matrix which is then used for the spectral clustering.



Spectral Clustering

The results of spectral clustering with a larger number of clusters.



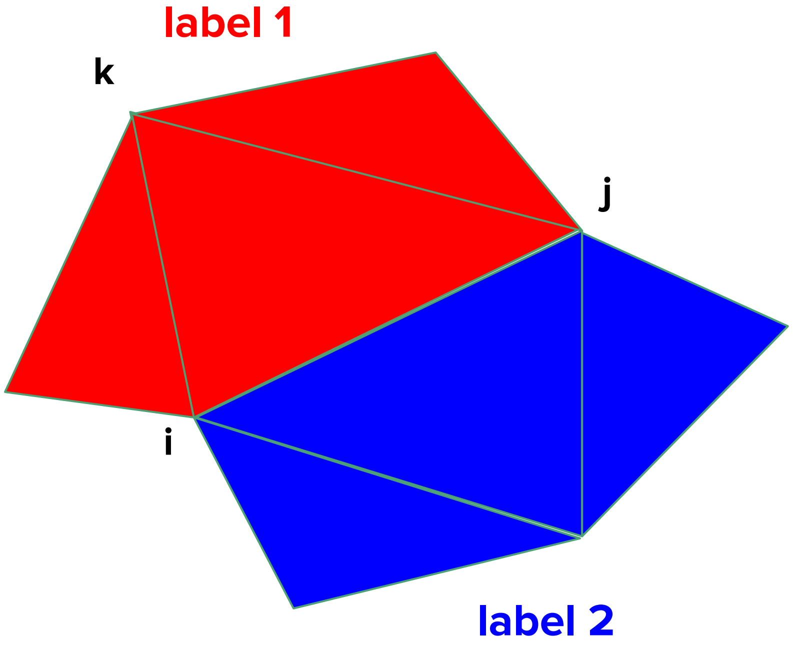


Merging Strategies

1) Finding adjacent supersurfaces
   - finding differently labeled surfaces along all vertices

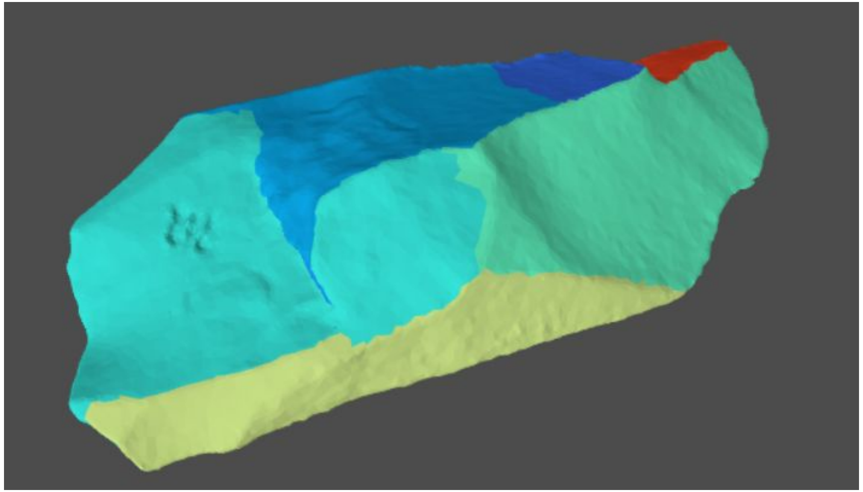
   Combine adjacent supersurfaces with normal vector difference < threshold

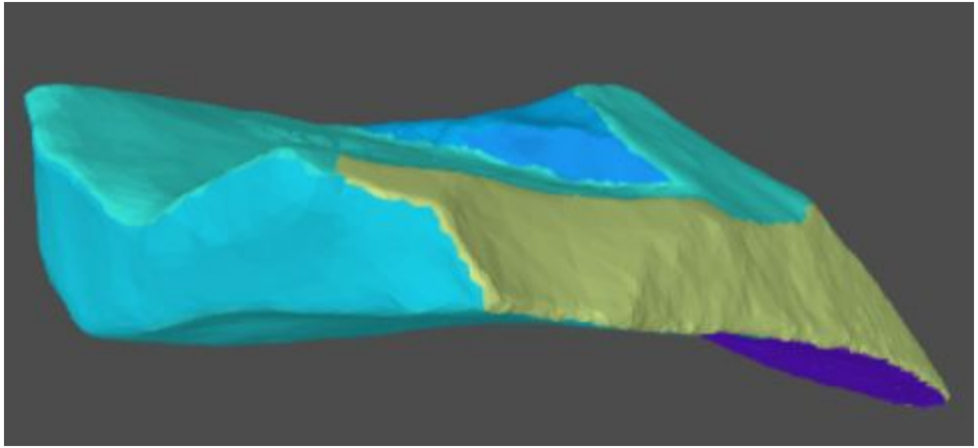
2) Merge supersurfaces if prior k-means Approximation predicts it



Spectral Clustering

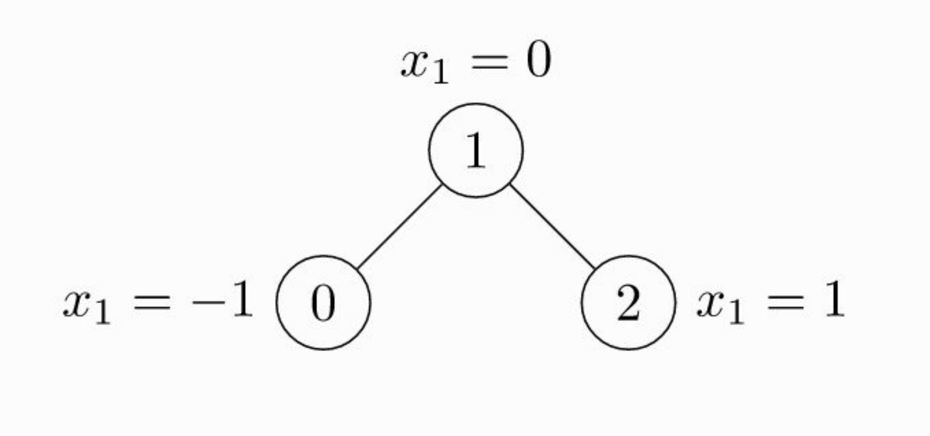
After merging we end up with fewer clusters.





Geometric Deep Learning

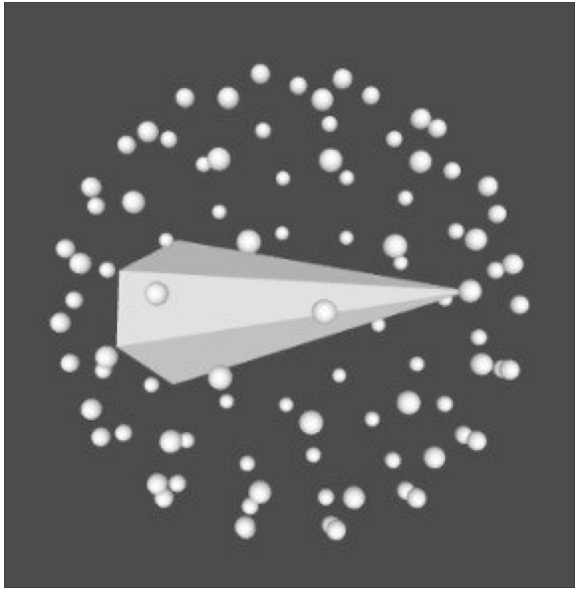
Graph Neural Networks



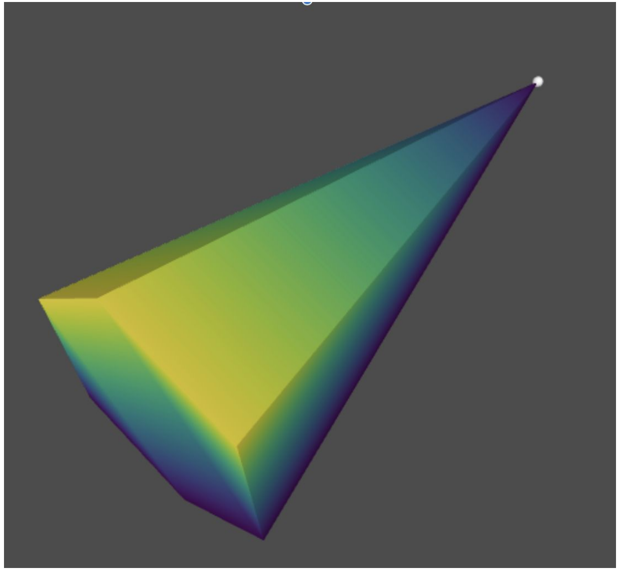
We use the Point-Pair-Feature convolution layer (PyTorch Geometric)

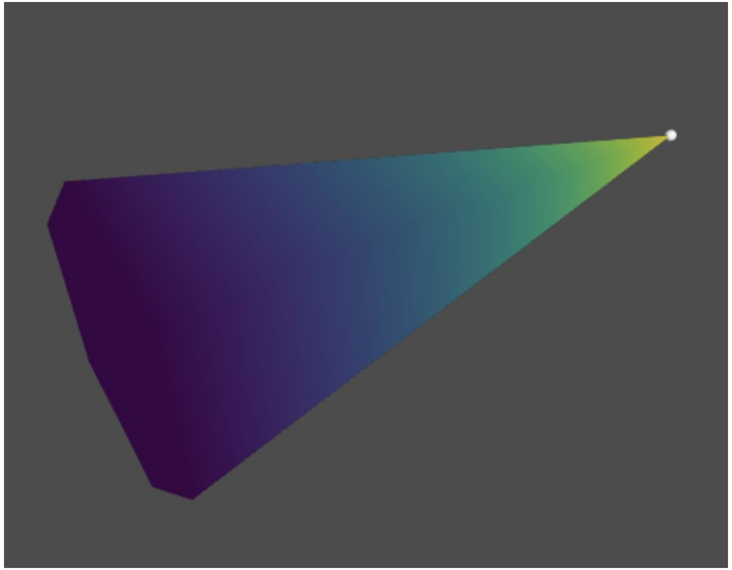
$$\mathbf{x}_i' = \gamma_{\mathbf{\Theta}} \left(\max_{j \in \mathcal{N}(i) \cup \{ i \}} h_{\mathbf{\Theta}} (\mathbf{x}_j, \| \mathbf{d_{j,i}} \|, \angle(\mathbf{n}_i, \mathbf{d_{j,i}}), \angle(\mathbf{n}_j, \mathbf{d_{j,i}}), \angle(\mathbf{n}_i, \mathbf{n}_j) \right)$$

Dummy data

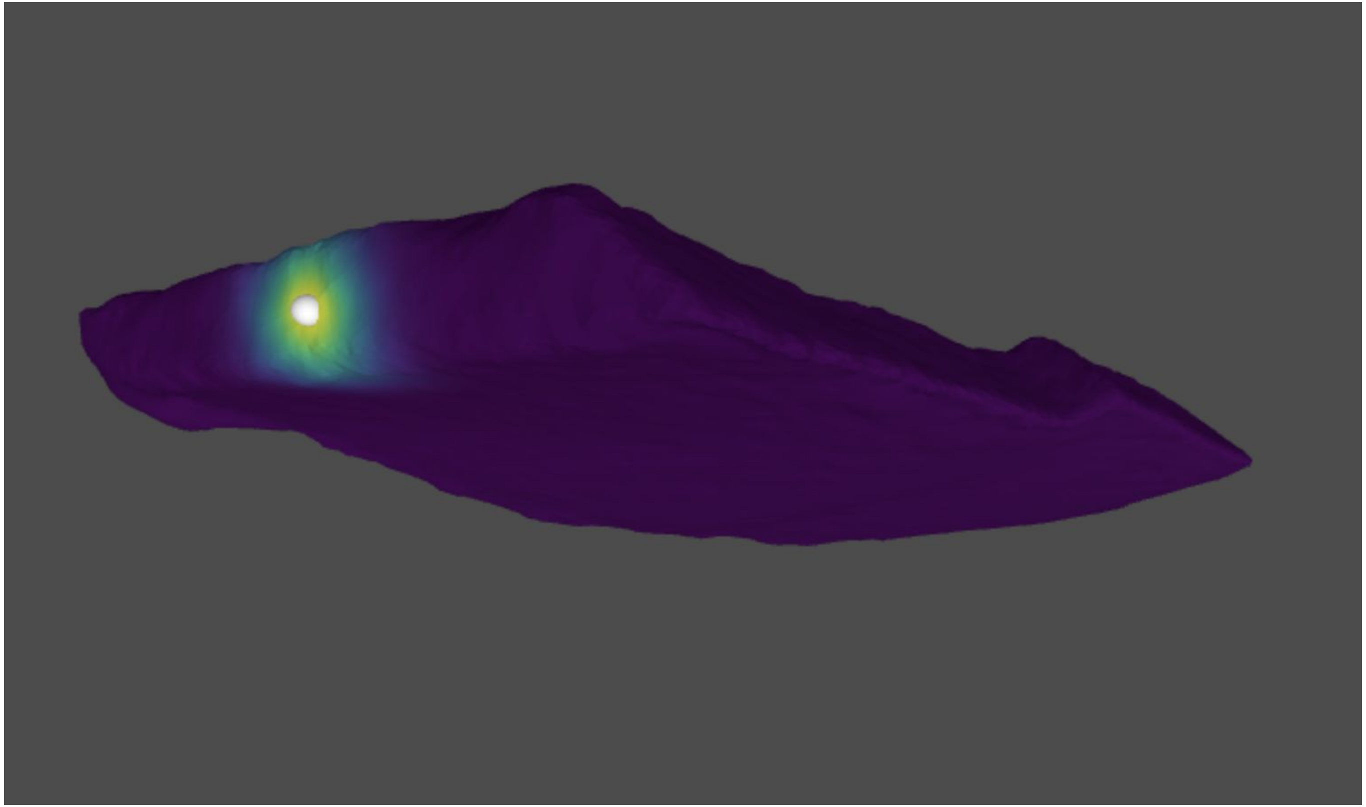


Dummy data

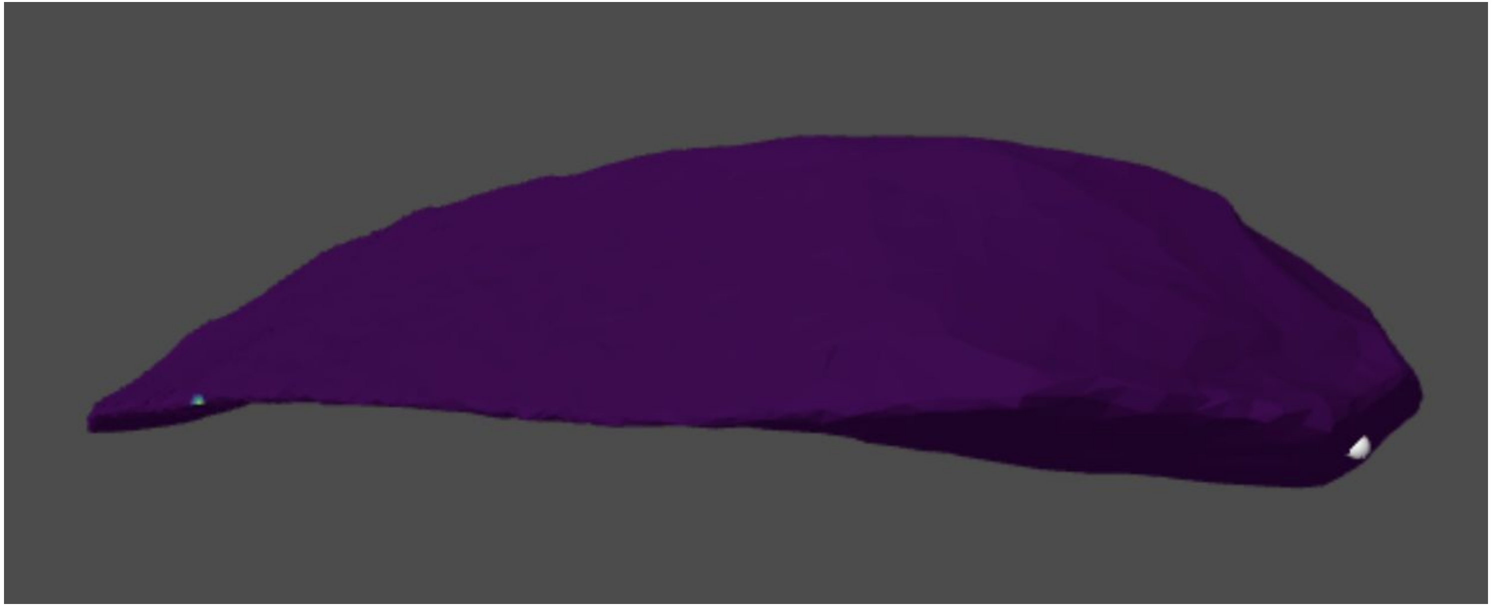




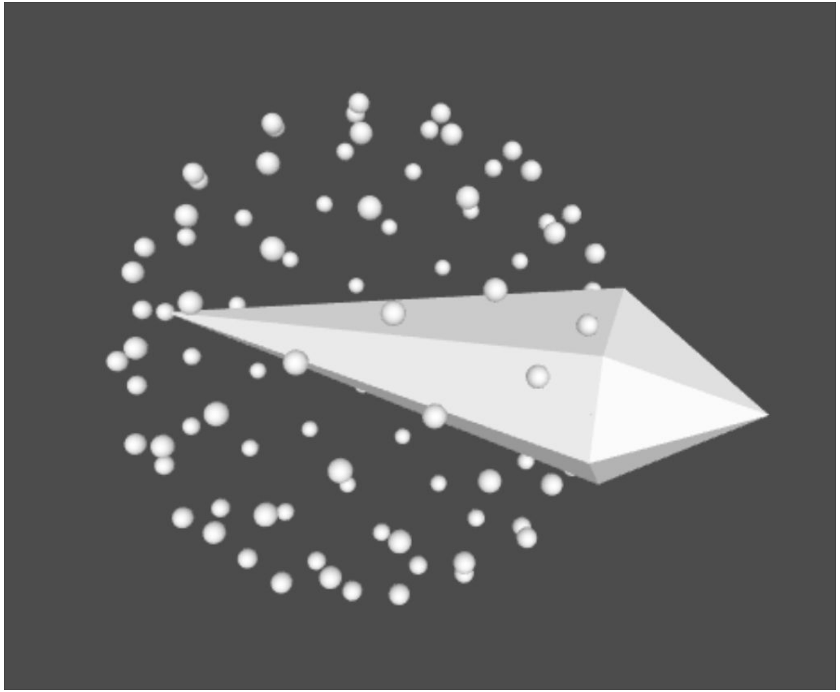
Smoothing out the data



After training...



Experiments on Artificial Data



Experiment on Artificial Data