Kickoff Presentation

Roee Ben Shlomo
Yevgeniy Men

We'll be starting our project by making adversarial examples with an attack that's generated by optimization methods, as suggested by Prof. Bronstein *et al.* in "Generating Adversarial Surfaces via Band-Limited Perturbations".
We'll be training the network with the FAUST dataset, and it'll be based on PointNet architecture.
We first decomoposed our program to small tasks as follows:

1. Prepare the PointNet architecture (perhaps with PyTorch Lightning) together with the FAUST dataset.
   This includes finding a proper way to load the FAUST dataset, as the .ply format is new for us.
   Make sure the net properly classifies our dataset.

2. Firstly, we build an untargeted attack that is not band-limited, i.e iterate through Eq. (13) :
   $\boldsymbol{X'}^{(i)} = \Pi_{\boldsymbol{X},\varepsilon}\left(\boldsymbol{X'}^{(i-1)} + \alpha \mathrm{sign}\left(\nabla L\left(\boldsymbol{X'}^{(i)}, y\right)\right)\right)$.
   We initiate the process with $\boldsymbol{X'}^{(0)} = \boldsymbol{X}$ and set L as the cross-entropy loss between the probability output of the classifier and the ground-truth class y.
   α, as specificed by the article, will be set to 0.3ρ and $\varepsilon = 3\rho$, where ρ is the median edge length of $\boldsymbol{X}$.
   As specificed by the article, the gradient will be computed via automatic differentiation in autograd.
   In order to implement the iterations we'll build a modular approach, hence:

3. Create the clipping function $\Pi_{\boldsymbol{X},\varepsilon}$. It should act vertex wise.

4. Create the loss function $L\left(\boldsymbol{X}, y\right) = -\log p\left(y|\boldsymbol{X}\right)$

5. Lastly, we can build a method to show to results.
   To do so, we can plot the original figure, and to the side of it plot our adversarial attack with a visualization of the per-point absolute distortion of the mean curvature from the original figure, encoded as a heatmap, just like in the paper.
   For our initial purposes this is optional, as we will probably be able to see the differences with regular side-by-side plotting.

This should result in misclassification as desired, although the result won't be smooth.
Now, we want to improve our untargeted attack by adding band-limitation that'll smooth out the result.

1. Build $\boldsymbol{\Phi} \in \mathbb{R}^{n\times k}$,the matrix that contains the first k Laplacian eigenvectors of $\mathbf{X}$ (the sample we're attacking).

2. Build $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the matrix of area elements.