

# Densify Point Clouds Sprint 3

By Group A2\_9: Chris, Risa, and Brian

# Sprint 3 Tasks

- Continue researching papers and finding a paper that we can feasibly reproduce
  - Find frameworks that we can utilize from these papers
- Keep working on testing our development software
  - Test advanced tutorials on torch-3d
  - Install pytorch 3d and PCL locally
  - Install open3d and test tutorials
- Start working on theoretical ML module in pytorch 3d



# Accomplishments

- Our Team found an interesting paper that implemented their own deep learning model to classify sparse point clouds into denser objects.
- The paper also had the framework they developed under github called PointSDF



# Research Finding - PointSDF

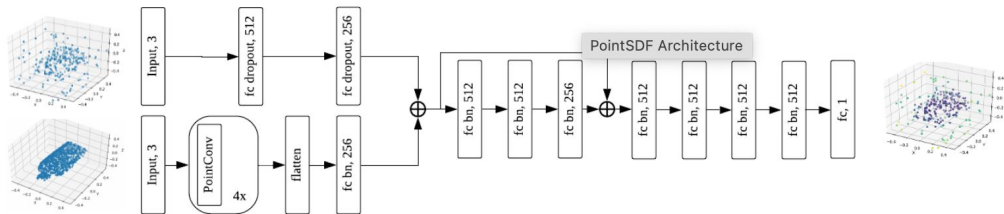
- **PointSDF - Signed Distance Function**

- Directly regresses signed distance functions from point clouds
- Providing geometrically rich input and output

- **Distance Prediction**

- Implicitly encode geometry by introducing the point cloud embedding from PointSDF

3D Object Reconstruction



ec602\_fall2022\_hw4\_wedding\_checker

NeuralPoints

Project

NeuralPoints

code

model

utils

README.md

External Libraries

Scratches and Consoles

Pull Requests

Structure

Favorites

README.md

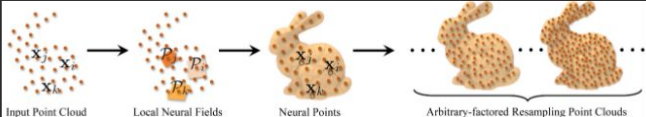
ec602\_fall2022\_hw4\_wedding\_checker.py

```
1 # Neural Points
2 【Code of CVPR 2022 paper】Neural Points: Point Cloud Repres
3
4 - Paper address: [https://arxiv.org/abs/2112.04148](https://
5 - Project webpage: [https://wanquanf.github.io/NeuralPoints.
6
7
8 ![avatar](./utils/Pipeline_v5.png)
9
10 ## Prerequisite Installation
11 The code has been tested on Ubuntu 18, with Python3.8, PyTor
12
13     conda create --name NePs
14
15     conda activate NePs
16
17     conda install pytorch=1.6.0 torchvision=0.7.0 cudatoolkit=
18
19     conda install -c conda-forge igl
20
21 Before running the code, you need to build the cuda&C++ exte
22
23     cd [ProjectPath]/model/model_for_supp/pointnet2
```

## Neural Points

【Code of CVPR 2022 paper】Neural Points: Point Cloud Representation with Neural Fields for Arbitrary Upsampling (CVPR 2022).

- Paper address: <https://arxiv.org/abs/2112.04148>
- Project webpage: <https://wanquanf.github.io/NeuralPoints.html>



The diagram shows the workflow of the Neural Points method. It starts with an 'Input Point Cloud' (a collection of points  $x_1, x_2, x_3, x_4$ ). This is processed by 'Local Neural Fields' (represented by orange spheres  $P_1, P_2$ ). The output is 'Neural Points' (a denser set of points  $\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4$ ). Finally, these are used for 'Arbitrary-factored Resampling Point Clouds', which are shown as two separate point clouds of a bunny shape.

## Prerequisite Installation

The code has been tested on Ubuntu 18, with Python3.8, PyTorch 1.6 and Cuda 10.2:

```
conda create --name NePs
conda activate NePs
conda install pytorch=1.6.0 torchvision=0.7.0 cudatoolkit=10.2 -c pytorch
conda install -c conda-forge igl
```

## libigl

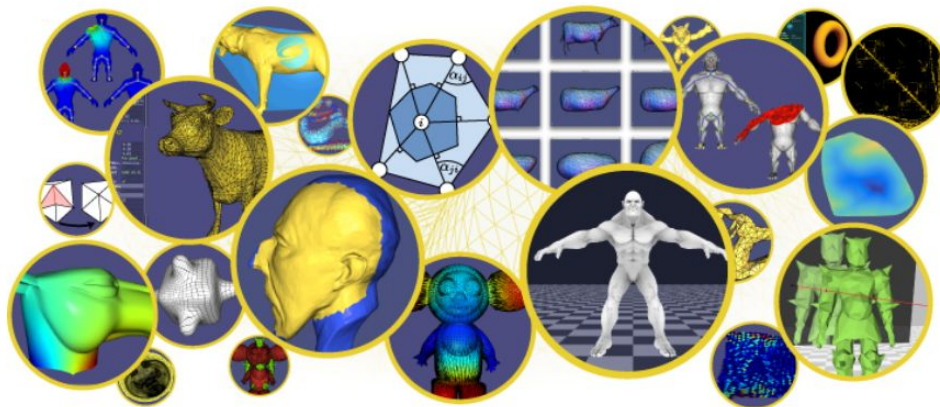
[Home](#)[Tutorial](#)[Python Bindings](#)[Compilation](#)[Contributing](#)[Misc](#)[FAQ](#)[About](#)

&gt;

&gt;

&gt;

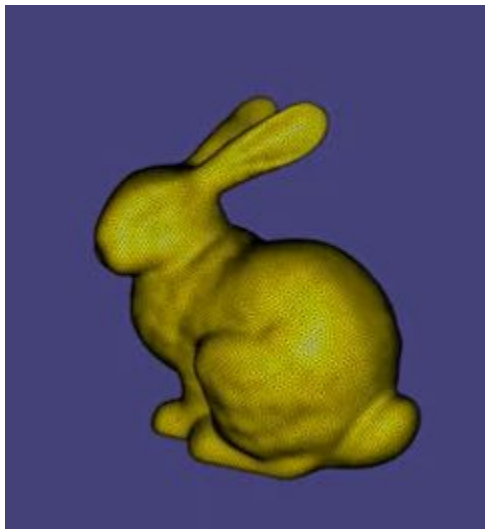
&gt;

 Build passing

libigl is a simple C++ geometry processing library. We have a wide functionality including construction of sparse discrete differential geometry operators and finite-elements matrices

**Table of contents**[Short Video Introduction](#)[Tutorial](#)[libigl Example Project](#)[Coding Guidelines and Tips](#)[Installation](#)[Dependencies](#)[Optional Dependencies](#)[Downloading Dependencies](#)[GCC and the Optional CGAL  
Dependency](#)[OpenMP and Windows](#)[Download](#)[Known Issues](#)[Unit Testing](#)[How to Contribute](#)[License](#)[Citation](#)[Projects/Universities using libigl](#)[Contact](#)

- - -



```
libigl-course — bash — 84x25
git clone https://github.com/libigl/libigl.git
Cloning into 'libigl'...
remote: Enumerating objects: 159, done.
remote: Counting objects: 100% (159/159), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 38540 (delta 87), reused 105 (delta 55), pack-reused 38381
Receiving objects: 100% (38540/38540), 10.08 MiB | 8.57 MiB/s, done.
Resolving deltas: 100% (24014/24014), done.
vi demo.cpp
clang++ -std=c++11 -I libigl/include/ -I eigen/ -o demo demo.cpp
./demo bunny.obj
vi demo.cpp
clang++ -std=c++11 -I libigl/include/ -I eigen/ -o demo demo.cpp
./demo bunny.obj bunny.ply
viewmesh bunny.ply
```

Fig.2 CVPR 2022 Neural Points

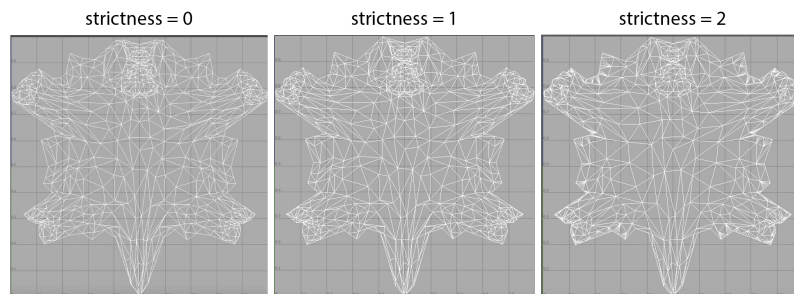
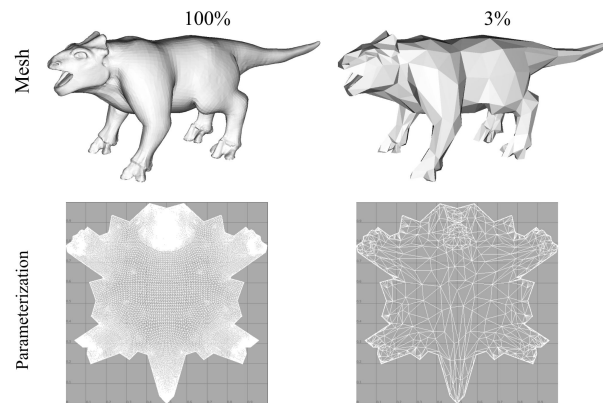
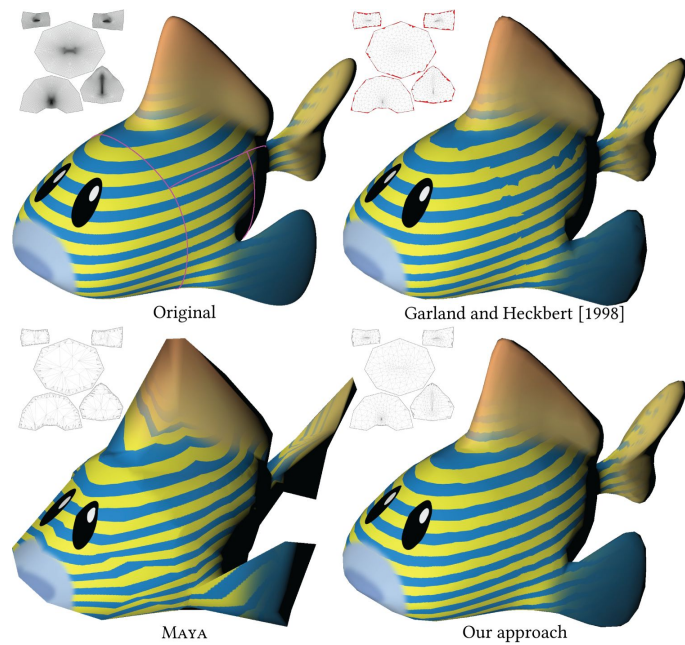


Fig.3 Libigl



# Research Finding

- **Supapixel Method**
  - Only performs a decent result under certain environmental conditions
  - Cannot present all the possible variations in an image, may deliver wrong/missing spots
- **Markov Random Field(MRF)**
  - Pixelwise optimization
  - Pair up the neighbor points/pixels to compute a certain 3D surface
- **Semantic Labeling**
  - Mapping elements on the image such as color and texture to a 3D plane normal
- **Multiple Segmentation Method**
  - Obtaining a higher accuracy by analyze the advantages of each single elements

# Set Backs

- We had serious trouble trying to install torch-3d and pytorch 3d using the SCC
  - Our team worked in tandem with IT but we didn't have much luck getting these libraries to function as intended
    - The closest we got was getting pytorch 3d to use CPU
- We also had trouble installing pytorch 3d locally
  - I tried to install Pytorch 3d locally on Linux but it didn't work out due to graphic driver issues
    - Im currently in the process of installing pytorch 3d using a conda environment



# Sprint 4 Tasks

- Install PointSDF locally and test their framework
  - Have a demo showcasing the PointSDF framework
- Continue Paper Research
  - Diagram a model for our ideas on how to optimize the frameworks ML component
- Continue trying to get pytorch 3d working
  - Work with Osama to check if the SCC has correctly installed the software and test example
  - Continue trying to install locally
  - (Reach goal) Have a demo showcasing pytorch 3d

