# Neural Highlighting of Affordance Regions

TA: Paolo Rabino (paolo.rabino@polito.it)

## TASK OVERVIEW

reference paper: 3D highlighter

This project aims to explore and familiarize with novel neural field advancements applied to the affordance detection task.

The goal of the project is to build a label-free pipeline for affordance generation leveraging pre-trained language vision models.
You will use the architecture proposed in *3D Highlighter: Localizing Regions on 3D Shapes via Text Description* and adapt it for generating affordance labels on real-world point clouds.

Literature:

Before starting the project take time to familiarize yourself with the literature on Vision Language models, Neural Fields, and Task Affordance.

## Neural Fields

Neural fields are an enticing new technique used for encoding continuous signals into a compact smooth representation parametrized in part or fully by a neural network, they work by taking in input a set of coordinates and generating as output the value of the field at the specified coordinates. Thus the network itself becomes a representation of the desired signal. For general applications of Neural Fields and an overview of the paradigm, you can read [2].

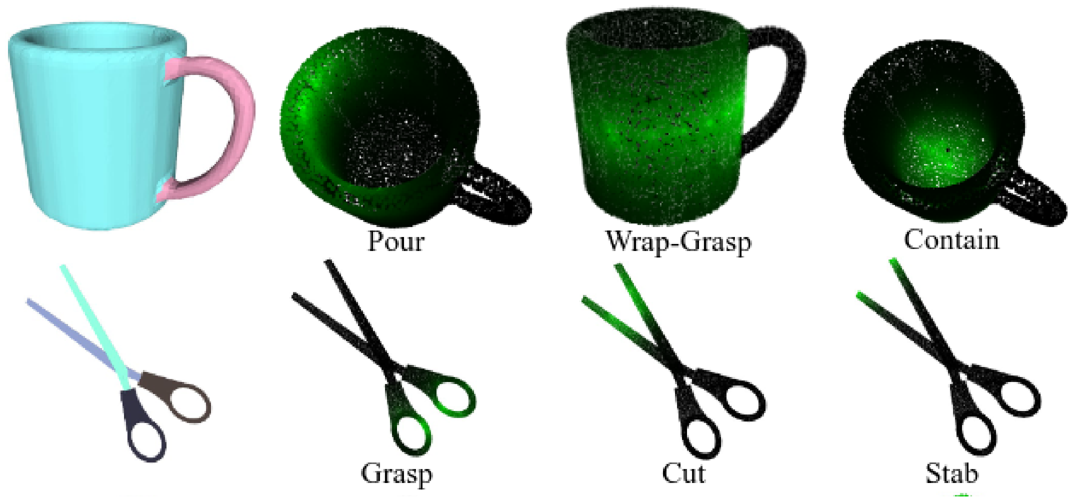
## Contrastive Vision Language Model

CLIP [3] has unlocked various applications by providing a bridge between visual data and language. Read the paper that introduced it and familiarize yourself with the concept of contrastive vision language models.
Training such models requires huge amounts of data, unfortunately in the field of 3D deep learning there aren’t datasets with as much diversity and quantity as needed. The solution that is usually adopted is to bypass the lack of data by casting the problem to the 2D vision domain. Differentiable rendering techniques enable the creation of 2D renderings starting from 3D data while retaining the differentiability of the whole pipeline.

In this project you will use the same trick, we will render our generated outputs to 2D images enabling the use of CLIP without additional pre-training.

### Object Affordances

Object affordance regions are a useful concept for defining the human-object interaction. An affordance region in the context of objects is defined as the object part that is most useful for achieving a specific task. For example, a chair has an affordance region that includes sitting, standing on, or using it as a makeshift ladder. Predicting affordance regions is very useful for improving robotic applications that require fine manipulation skills. For a more detailed description on affordance regions and their applications read [4]



## PROJECT GOALS

### First Part

The initial goal of this project is to allow you to get familiar with the neural highlighter architecture. First, you should thoroughly study the reference paper [1] and any other paper that you may find among its references and that could be useful to get a better insight on the task.

Start from the base repo here.
The main file *notebook.ipynb* contains a skeleton for the code base, the mesh rendering part is already implemented. The parts that you should implement yourself are marked in comments and NotImplemented exceptions, in particular, you should implement:

- **CLIP loading**, use the open-clip module and load the model from the API, implement the possibility to choose from the different available models

- **Neural Highlighting Model**, the model is a stack of **Linear, ReLU, LayerNorm** modules, implement it yourself in the NeuralHighlighter class, and implement it with a variable number of layers

- **Clip Loss**, the model is trained using cosine similarity between the rendered images and the text prompt.

$$\mathcal{L} = 1 - \frac{x_{\text{img}} \cdot x_{\text{text}}}{||x_{\text{img}}|| \cdot ||x_{\text{text}}||}$$

Choose a couple of meshes from the example meshes contained in the folder and a small set of prompts from the example prompts reported in the paper.

Validate your choice of architecture by comparing the speed of convergence of the CLIP score, the final CLIP score, and the visual quality of the obtained samples.

Due to the generative nature of the model, rigorous quantitative evaluation is difficult, therefore visualizing often the results is important to assess the performance of the model.

When reporting your results report both the CLIP score and a sampled visualization of your results commenting on how and why the results differ from previous experiments.

The hyperparameters you should explore are:

- **learning rate**
- **Network depth**
- **Number of sampled views**: the mesh renderer will generate a fixed number of views, does the network converge faster when increasing the number of views? Is it worth the additional computational overhead?
- **Augmentations**: the original paper uses several augmentations to enhance the rendered views, try this strategy yourself and see if it enhances the convergence ability of the model

Remember that the clip score is comparable only between the same sample and the same textual prompt.

END OF PROJECT FOR DAAI COURSE

## Second Part

Neural highlighter as proposed is limited to only work on meshes, while it is a useful data type most common 3D real-world scans are collected in the form of point clouds. Point clouds in general are much more flexible and can originate from a wide variety of sources, they can be sampled from meshes but they can also be collected from RGB-D cameras and Lidar sensors.

The goal of this part is to adapt the previously implemented pipeline to work on point cloud data, you are free to choose the approach you best see fit for the task, some possible approaches are:

1. Use a **differentiable point cloud renderer** instead of a differentiable mesh renderer, this will generate images where each point is represented as a **ball in 3d space**
   https://pytorch3d.org/tutorials/render_colored_points,
2. **Approximate a mesh** from the 3D point cloud
   http://www.open3d.org/docs/latest/tutorial/Advanced/surface_reconstruction.html
3. Convert the point-cloud to voxels and then **voxel meshes** using kaolin ops conversions
   https://kaolin.readthedocs.io/en/latest/modules/kaolin.ops.conversions.html

Test the results either using the point-clouds from the next project part or by sampling them from the provided example meshes using functions such as this

## Third Part

Use the previously defined pipeline to generate affordance information for the AffordanceNet benchmark [4].
The goal is to test how viable the pipeline you built for generating affordance labels without any kind of direct supervision.
You should focus on common household items and labels that deal with hand-object interaction e.g. Grasp Push Wrap etc.

Define two small subsets of samples from the training set as your validation and test sets, and focus on a few types of annotations that you think are best suited for the model. Define a testing pipeline using the provided GTs, use the **mIOU** metric to assess how good is your model at finding the affordance regions. Depending on your chosen projection strategy there could be some steps needed to compare the given GTs to the generated samples.

Define CLIP prompts for each of the possible selected affordance labels and test the segmentation accuracy using the ground truth labels of the benchmark. Explore and compare different prompting strategies for generating the affordance highlighting, depending on the affordance label and the type of chosen rendering there could be different optimal prompts.

### Extensions

In this part of the project, you can propose your contribution to expand the proposed pipeline, if you have interesting ideas you are free to implement them. You should at least implement one extension, possible extensions are:

1. Use Other models for the language supervision part, you can try to use models that work natively on 3D data without the need for differentiable rendering techniques like OpenShape [5] or advanced 2d to 3D clip models like PointClip [6, 7]

2. Expand your evaluation for the part segmentation task, try to define a similar benchmarking suit from the part-net dataset, and repeat the previously proposed evaluation

3. If you have a 3D camera at your disposal (Some modern iPhones or iPads have a lidar sensor integrated) you can also try to collect a few scans of real-world household objects and use them to test your model

4. Try if different training augmentations produce more reliable training dynamics and better results, you could try adding a background to the rendering and see if it improves over a regular mono-color background

5. Try different positional encoding try using classical Nerf positional embedding or others you can find in the literature

6. Implement your idea

## At The End

- Deliver PyTorch scripts for all the required steps.
- Write a complete PDF report (in paper style). The report should contain a brief introduction, a related work section, a methodological section for describing the algorithm that you're going to use, an experimental section with all the results and discussions, and a final brief conclusion.
- At the final exam, you will have to give a brief presentation regarding your project, using a slide format. You don't have to deliver the slides with the project, you can just show up at the exam with them.

## Example Of Questions you should be able to answer at the end of the project

1. What are object affordances?
2. What are the various types of 3D representations commonly used in DL?
3. What is a neural field?
4. How does CLIP work?
5. What is differentiable rendering and why is it useful?

## References

[1] Decatur, Dale, Itai Lang and Rana Hanocka. “3D Highlighter: Localizing Regions on 3D Shapes via Text Descriptions.” *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022): 20930-20939.

[2] Xie, Yiheng, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann and Srinath Sridhar. “Neural Fields in Visual Computing and Beyond.” *Computer Graphics Forum* 41 (2021): n. pag.

[3] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and Ilya Sutskever. “Learning Transferable Visual Models From Natural Language Supervision.” *International Conference on Machine Learning* (2021).

[4] Deng, Sheng, Xun Xu, Chaozheng Wu, Ke Chen and Kui Jia. “3D AffordanceNet: A Benchmark for Visual Object Affordance Understanding.” *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021): 1778-1787.

[5] Liu, Minghua, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, H. Cai, Fatih Murat Porikli and Hao Su. “OpenShape: Scaling Up 3D Shape Representation Towards Open-World Understanding.” *ArXiv* abs/2305.10764 (2023): n. pag.

[6] Zhang, Renrui, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Jiao Qiao, Peng Gao and Hongsheng Li. “PointCLIP: Point Cloud Understanding by CLIP.” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021): 8542-8552.

[7] Zhou, Kaiyang, Jingkang Yang, Chen Change Loy and Ziwei Liu. “Learning to Prompt for Vision-Language Models.” *International Journal of Computer Vision* 130 (2021): 2337 - 2348.

—-

Michel, Oscar, Roi Bar-On, Richard Liu, Sagie Benaim and Rana Hanocka. “Text2Mesh: Text-Driven Neural Stylization for Meshes.” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021): 13482-13492.

Sitzmann, Vincent, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell and Gordon Wetzstein. “Implicit Neural Representations with Periodic Activation Functions.” *ArXiv* abs/2006.09661 (2020): n. pag.

Li, Boyi, Kilian Q. Weinberger, Serge J. Belongie, Vladlen Koltun and René Ranftl. “Language-driven Semantic Segmentation.” *ArXiv* abs/2201.03546 (2022): n. pag.