

The background is a dark blue-grey color. It is decorated with various geometric shapes in orange and white. There are circles of different sizes, some with dotted patterns inside. There are hexagons, some solid orange and some outlined in white. There are also triangles and lines. A horizontal dotted line is positioned above the title, and another one is below the team list. The title '3D Reconstruction Occupancy Networks' is written in a large, white, sans-serif font, centered on the slide.

3D Reconstruction Occupancy Networks

Team Noisy Pixels:

Shubham Dokania (2020701016)

Shanthika Naik (2020701013)

Sai Amrit Patnaik (2020701026)

Madhvi Panchal (2019201061)

01. Introduction &
Proposal

02. Project Expectations

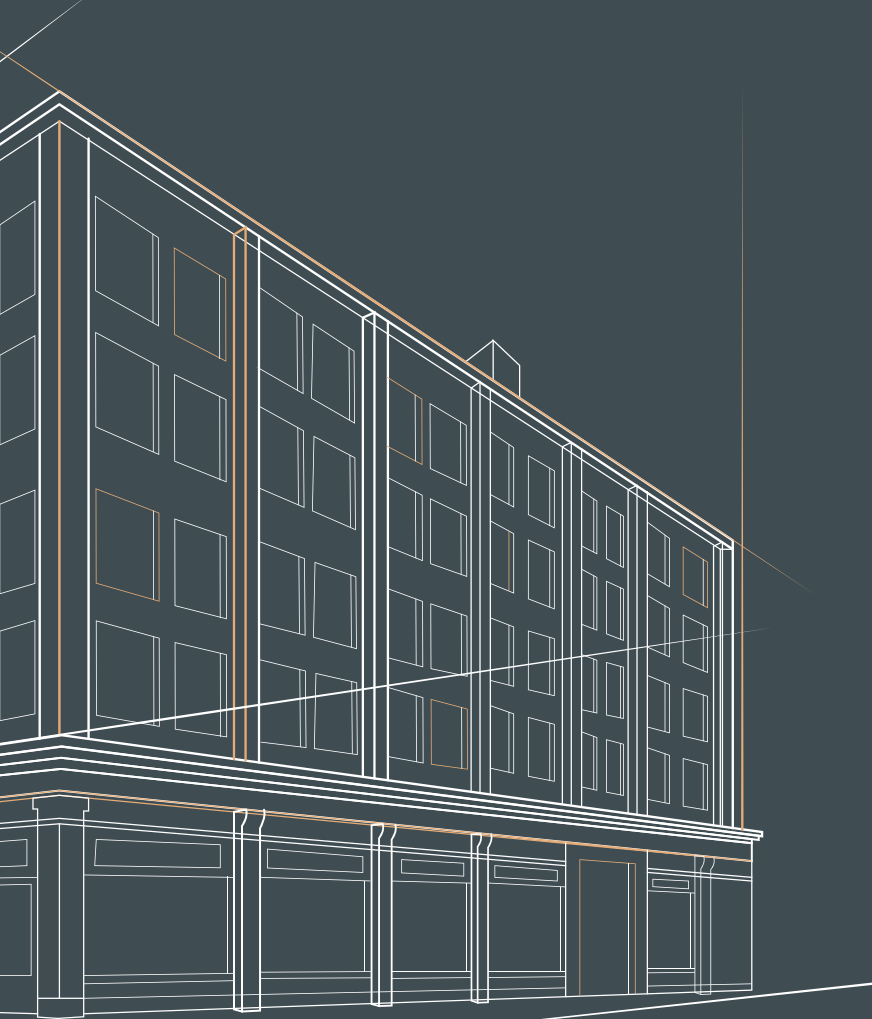
03. Mid-Evaluation
Progress

04. Future Work

05. References

06. Ending Notes



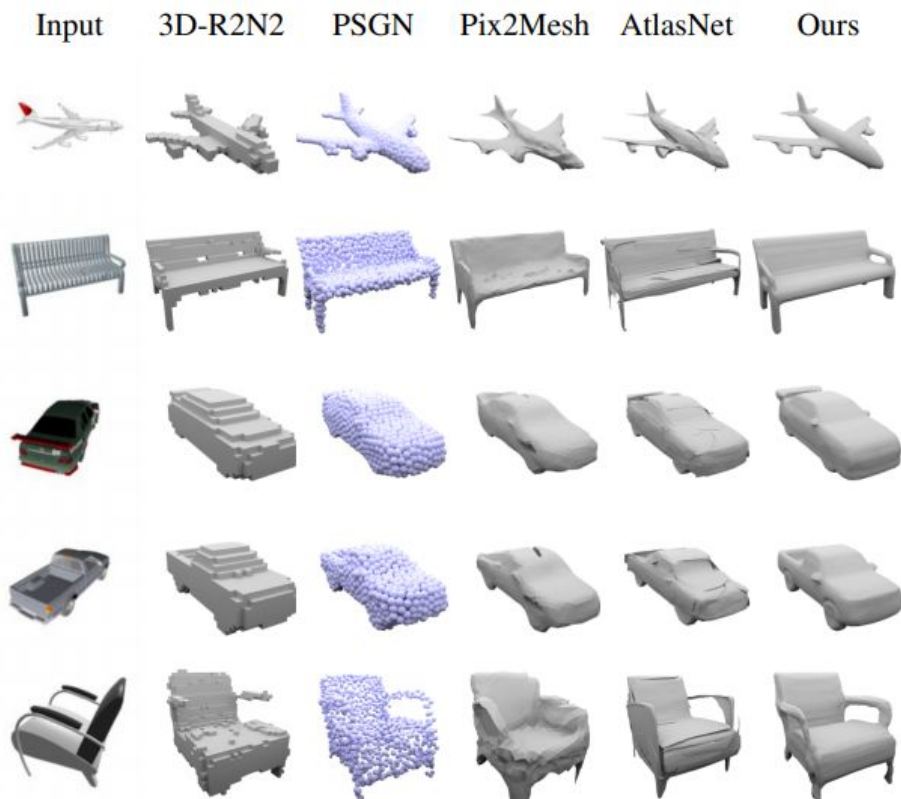


01

Introduction & Proposal



3D Reconstruction



Problem Statement

With the considerable pace of advancement in the field of deep learning and its applications, notable developments have been seen in the domain of 3D vision.

Unlike Images, there is no well defined, structured representation method in 3D which is both computationally and memory efficient, but allows arbitrary topology to represent high-resolution geometry.

However, considering functional space, we can map the 3D space to an implicit function, for which a non-linear classifier may exist.

The task we wish to solve here is that of 3D reconstruction from a single view image, i.e. given an image of an object (without camera pose), we want to retrieve the 3D geometry of the object and reconstruct it in 3D space.

. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .

Goals

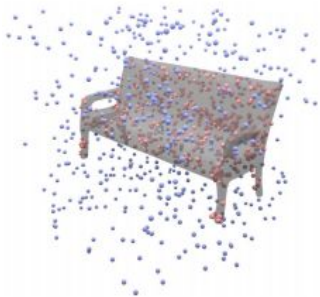
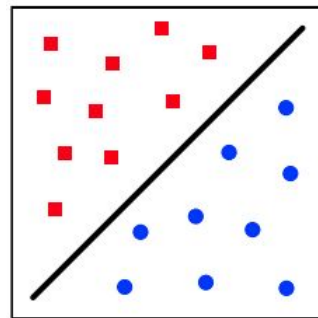
- In this project the 3D surface is implicitly learnt as a continuous decision boundary of a non-linear classifier instead of learning to predicting an explicit voxelized representation at a fixed resolution.
- A neural network learns the entire occupancy function that can be evaluated at any arbitrary resolution.
- At inference time, the mesh is extracted from the learnt occupancy function by using a Multi-resolution IsoSurface Extraction(MISE) technique and Marching cubes.
- The final mesh is refined using a 2 step process using Fast-Quadric-Mesh-Simplification algorithm and first and second order (i.e., gradient) information.

Occupancy Networks

- Model 3D surface as a decision boundary of a non-linear classifier.
- Returns an Occupancy probability of a 3D point conditioned on an input image.

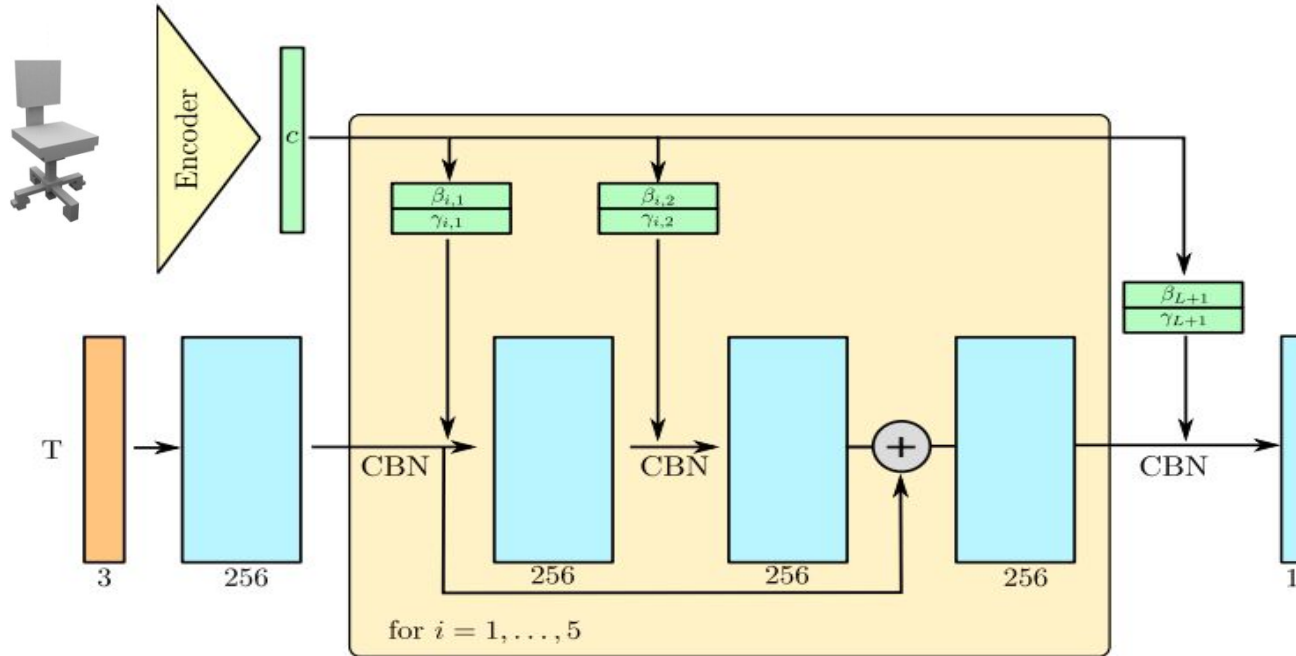
$$f_{\theta} : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1]$$

3D Condition Occupancy
Location (eg, Image) Probability



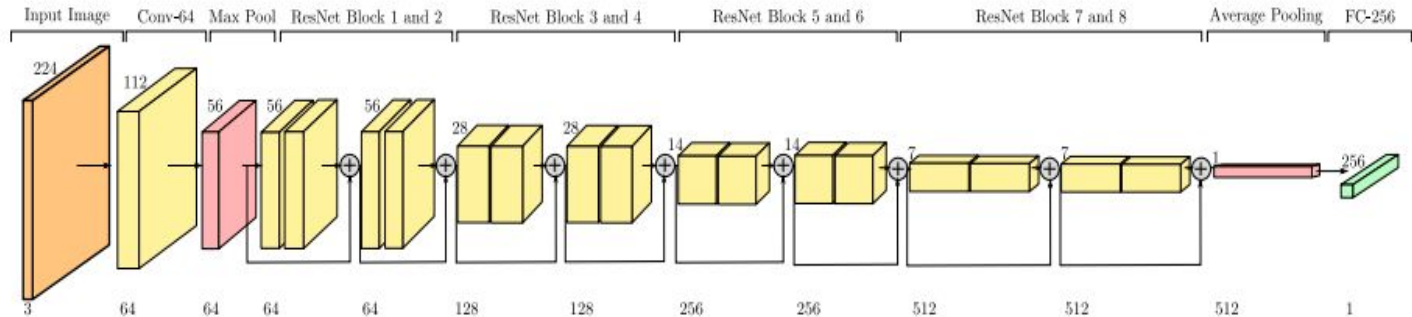
- Model 3D surface as a decision boundary of a non-linear classifier.
- Returns an Occupancy probability of a 3D point conditioned on an input image.

Network Architecture



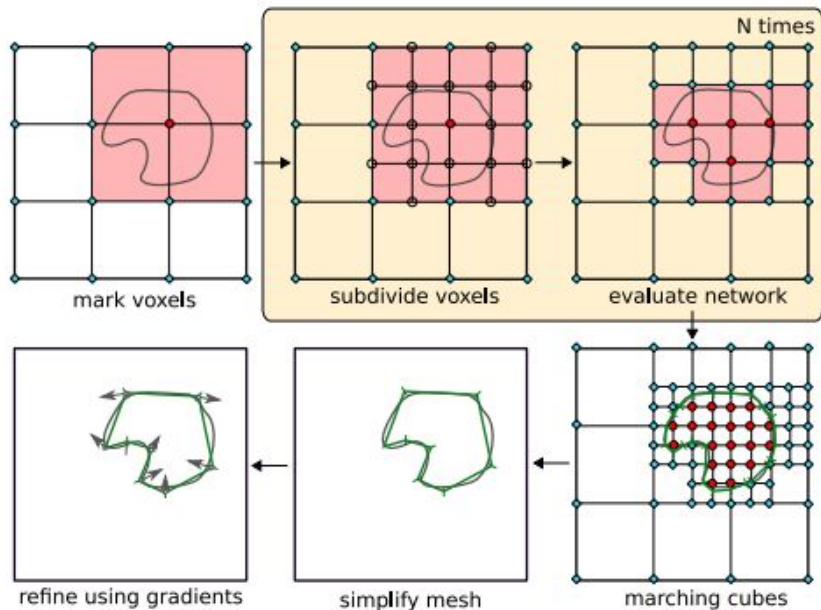
Occupancy Networks

- Occupancy network consists of a fully connected neural network with 5 ResNet blocks with the input conditioned it using Conditional Batch Normalization.
- The network takes as input a set of uniformly sampled 3D points.
- Image encoder encodes the input image to a feature vector.
- The extracted feature vector for the image is fed to the Occupancy Net using a Conditional Batch Normalization operation.
- Image Encoder is a ResNet 18 network with the final layer returning a 256D feature vector of the input image.



Iso-surface Extraction

We use the MISE (Multi-resolution Iso-Surface Extraction) method proposed in the paper; this is a hierarchical method which incrementally builds an octree.



We first mark all points at a given resolution which have already been evaluated as either occupied (red circles) or unoccupied (cyan diamonds).

We then determine all voxels that have both occupied and unoccupied corners and mark them as active (light red) and subdivide them into 8 sub-voxels each.

Next, we evaluate all new grid points (empty circles) that have been introduced by the subdivision.

The previous two steps are repeated until the desired output resolution is reached. Finally we extract the mesh and refine, as explained further.




Mesh extraction



- Once the desired resolution is reached, we use the Marching Cube algorithm to extract an approximate isosurface :

$$\{p \in \mathbb{R}^3 \mid f_{\theta}(p, x) = \tau\}.$$

- The mesh obtained is further simplified using Fast-Quadric-Mesh-Simplification algorithm.
- For this purpose, random points are sampled from each face of output and minimizing the loss :

$$\sum_{k=1}^K (f_{\theta}(p_k, x) - \tau)^2 + \lambda \left\| \frac{\nabla_p f_{\theta}(p_k, x)}{\|\nabla_p f_{\theta}(p_k, x)\|} - n(p_k) \right\|^2$$


Expected Results

Fig. Example results on different objects from occupancy networks, along with post processing.

(right): Example input of a chair, with image from the back and it's reconstructed output.

(down): Example input of a bench and its corresponding 3D reconstruction output.



Input
image



Output
mesh



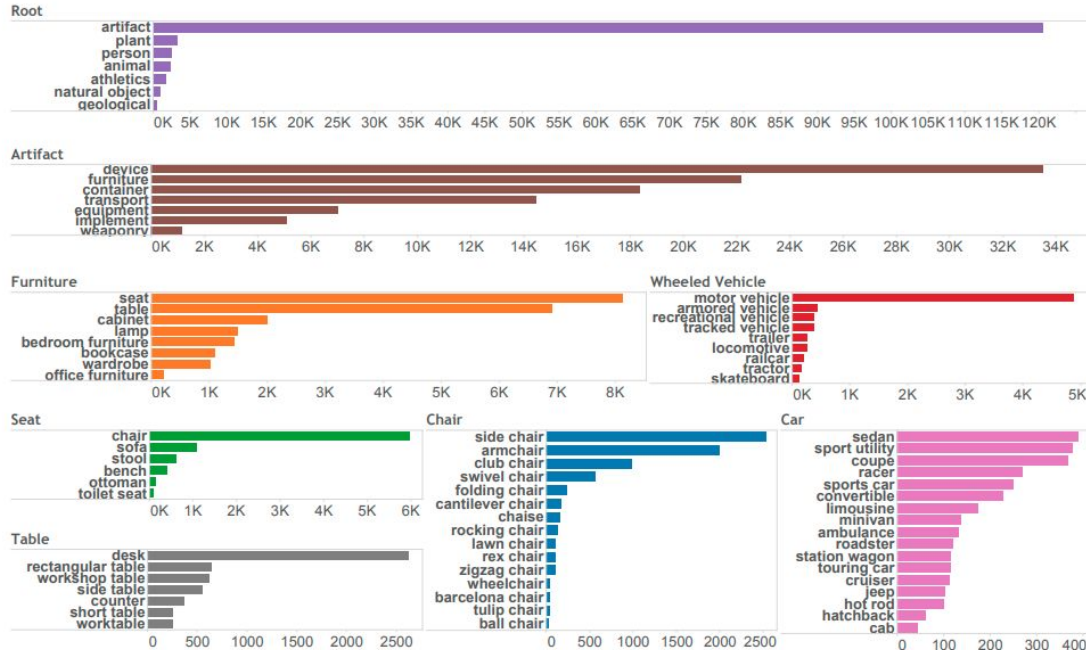
Input
image

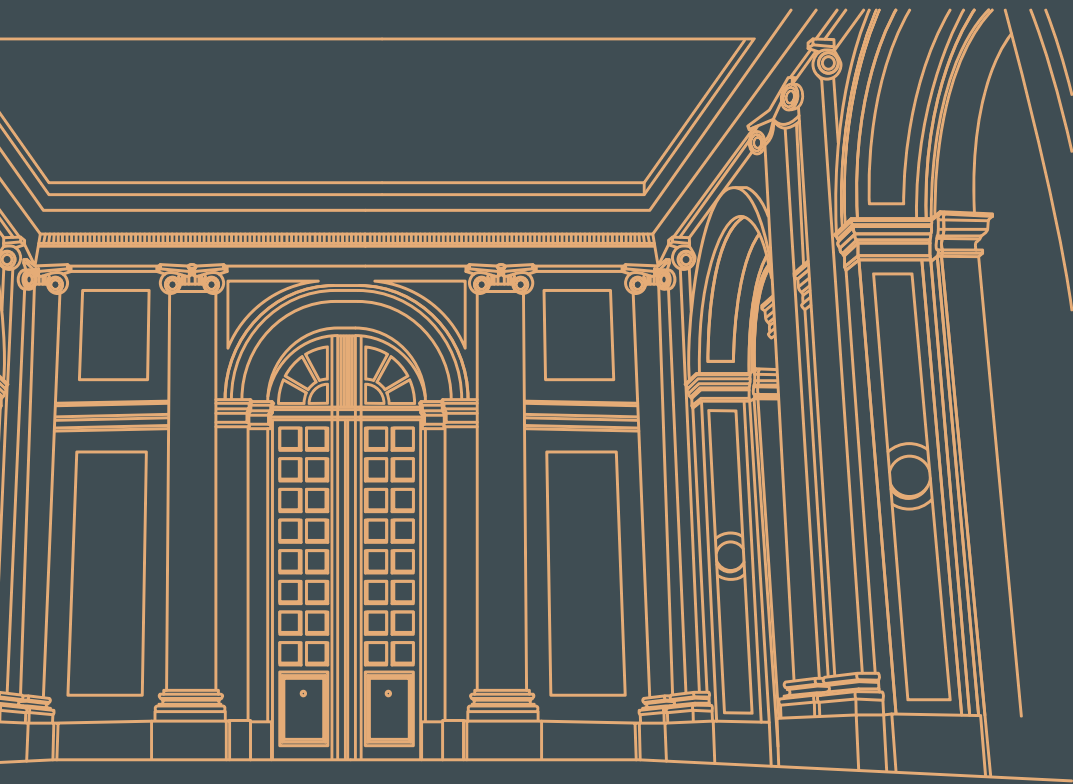


Output
mesh

Dataset

- The dataset used in this paper is the *ShapeNet* dataset.
 - ShapeNet is a richly-annotated, large-scale repository of shapes represented by 3D CAD models of objects. It is the ImageNet equivalent for 3D data.
 - ShapeNet contains 3D models from a multitude of semantic categories and organizes them under the WordNet taxonomy.
 - ShapeNet has indexed more than 3,000,000 models, 220,000 models out of which are classified into 3,135 categories (WordNet synsets).





02

Project Expectations






Expected Deliverables



The overall expectation from this project is an implementation of the occupancy network approach for 3D reconstruction. To outline the tasks in milestones, we shall follow the below deliverables:

1. Analysis of the ShapeNet Dataset relevant to 3D reconstruction.
 2. PyTorch implementation of Occupancy Network for Single View Reconstruction.
 3. Visualization of point clouds as inside/outside the surface as output of network.
 4. Prepare demo for 3D mesh reconstruction using mesh simplification as available in open-source mesh-fusion library.
 5. Compare performance metrics such as Chamfer distance, IoU, and Normal consistency with similar methods (available implementations).
 6. Release all code as open-source and with proper instructions.
- 



Expected Timeline



- **Week 1-2**

- Read the paper thoroughly and get detailed understanding of the idea.
- Prepare detailed proposal and download shapenet dataset on servers.
- Initial analysis on dataset


- **Week 3-4**

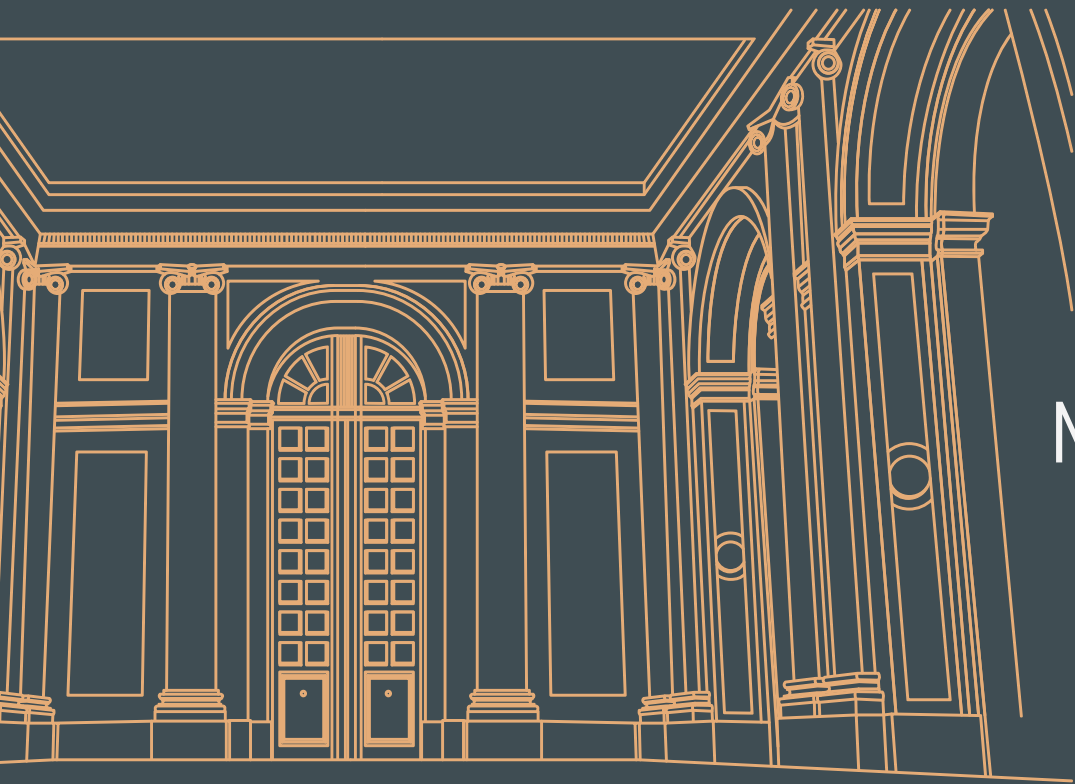
- Prepare preprocessed dataset from instructions of paper.
- Make visualizations of point cloud data and example GT meshes.
- Apply additional processing on dataset and prepare PyTorch dataloaders.
- Prepare encoder models and setup structure for entire experiments.

- **Week 5-6**

- Write complete encoder-decoder architecture in PyTorch.
- Prepare loss functions and final training pipeline.
- Wrap the training pipeline in pytorch-lightning for better code structure.
- Train initial model in dataset subset as dry-run. Report/Save accuracies and training artifacts.

- **Week 7-8**

- Prepare graph encoder for point cloud input to the model. For the task of high resolution mesh generation from object point clouds. **(additional proposal)**
 - Prepare final post-processing pipeline for mesh creation with trained occupancy network using marching cubes and smoothing proposed in the paper.
 - Code and report metrics such as chamfer distance and Earth mover distance for the generated results.
 - Provide visualizations of occupancy, generated meshes, and original data.
 - Compare with meshes generated with possession and ball-pivot algorithm of Open3D. **(additional proposal)**
- 



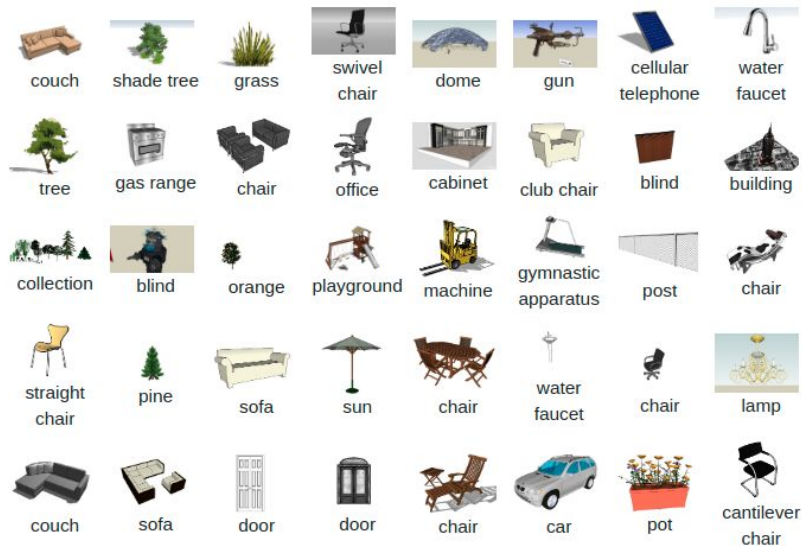
03

Mid-Evaluation Progress



ShapeNet Dataset

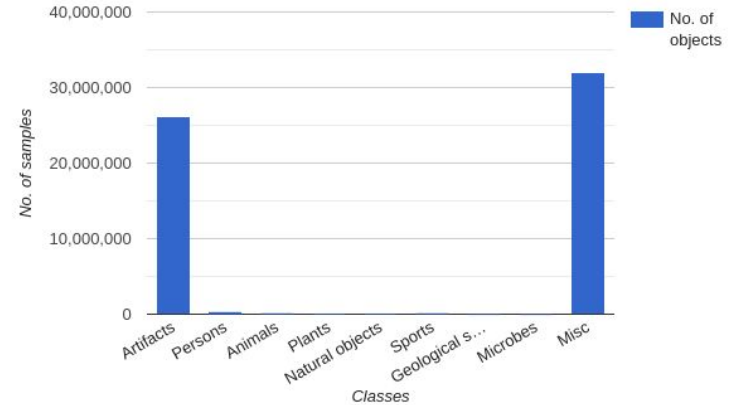
- Large, information-rich repository of 3D models.
- Rich set of annotations like geometric attributes (upright and front orientation vectors), parts and keypoints, shape symmetries (reflection plane, other rotational symmetries), and scale of object in real world units.
- Currently the shapenet dataset has 58906411 objects which are categorized into classes according to wordnet taxonomy.
- They are roughly arranged into 218403 synsets according to wordnet synsets.



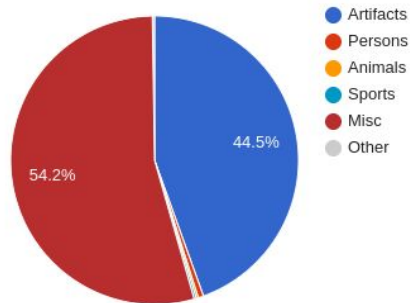
Data Distribution

Data is distributed into these high level super sets and there are further hierarchical sub classes for each class :

- Artifacts - (2,62,04,893)
- Persons, individuals, people - (3,52,561)
- Animals - (1,42,761)
- Plants - (63,311)
- Natural objects - (71,059)
- Sports - (1,41,764)
- Geological structures - (10,549)
- Microbes - (1,167)
- Misc - (3,19,18,346)



Data Distribution




- The figure in brackets indicates number of objects in each class. We can clearly see a lot of data imbalance for some classes (Artifacts and Misc classes) as compared to other classes.
- In each class, there also exists class imbalance between various sub classes.



Data Pre-Processing




- We use the (32^3) voxelization and image renderings as used by original paper.
 - In order to determine if a point lies in the interior of a mesh (e.g., for measuring IoU), the meshes need to be watertight. We therefore use the code provided by *Stutz et al.*, which performs TSDF-fusion on random depth renderings of the object, to create watertight versions of the meshes
 - We have centered and rescaled all meshes so that they are aligned with the voxelizations from. We transform the meshes so that the 3D bounding box of the mesh is centered at 0 and its longest edge has a length of 1.
 - We then sample 100k points in the unit cube centered at 0 with an additional small padding of 0.05 on both sides and determine if the points lie inside or outside the watertight mesh.
 - We save both the positions of the 100k points and their occupancies to a file.
- 



Data Processing




- Furthermore, we also prepare the dataset into HDF5 files for each object.
 - Each HDF5 file contains information regarding only one object:
 - The set of images for the corresponding object.
 - Camera matrices for each image (intrinsic and extrinsic).
 - Original point clouds along with surface normals.
 - Occupancy point cloud (100k) and ground truth information.
 - During training, we subsample 2048 points from this set (with replacement) as training data.
 - HDF5 format provides significant advantages compared to storing raw data, such as dynamic loading and easier data partitioning for storage in servers.
(Also, this lets us run our demos on google colab)
 - For the subset, we randomly sample roughly 10% of the entire dataset, while keeping same number of objects per class. This allows to reduce any class imbalance during the prototyping stage.
- 



Data Visualization



- For data sanity tests, we also prepare data visualizations and notice some visual properties of the dataset.
 - For the visualization, we use the original data point cloud first, along with surface normals and the RGB image.
 - For an example, mesh we use the ball-pivot algorithm from the Open3D library and demonstrate the results in the next slide.
 - The surface normals are visualized as RGB colors for each of the directions, this lets us understand the transition of the surface smoothness and bends on the surface, as can be seen for the examples:
 - Consistent colors on planar regions (ex: car and sofa), and gradients of colors on curvatures as direction changes (ex: airplane body).
 - Furthermore, we notice the holes in the generated meshes. This is because the Open3D's ball-pivoting algorithm passes through big gaps in the point cloud and hence shows the holes on the surface. An alternative is the poisson surface fitting algorithm, or marching cubes (as we shall be using for our post-processing steps).
- 

Data Visualization

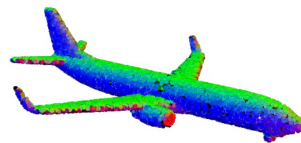
Point cloud



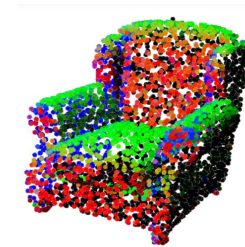
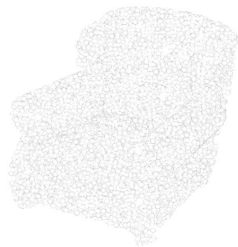
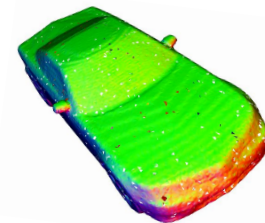
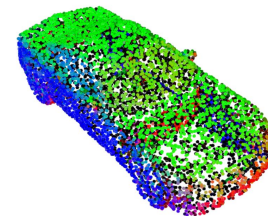
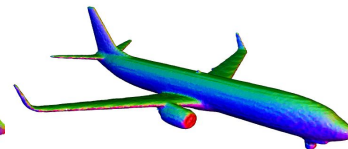
Image



point cloud
(surface normals)



Mesh






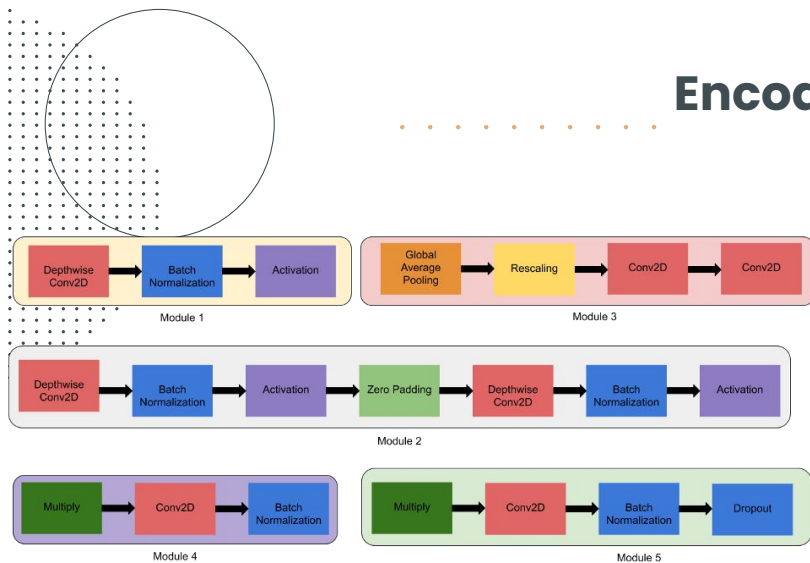
Encoder-Decoder Network



- We use a ResNet-18 architecture, pretrained on the ImageNet with last FC layer adjusted to project the features to a 256-d embedding c . We generate a feature vector from this network and combine with 2048 xyz points.
- The feature vector is passed through a series of ResNet-blocks, each of which applies a conditional batch Normalization (CBN) and an activation, before being fed into a final FC block with another CBN layer. A skip-operation is performed for information passing with the previous layer.
- Finally, this vector is passed through a last block and transformed down to a scalar value representing the probability of occupancy.
- CBN Implementation: We pass the conditional encoding c from the encoder through two FC layers to obtain 256-d vectors $\beta(c)$ and $\gamma(c)$. We then normalize the 256-d input feature vector f_{in} using first and second order moments, multiply the output with $\gamma(c)$ and add the bias term $\beta(c)$ as follows:

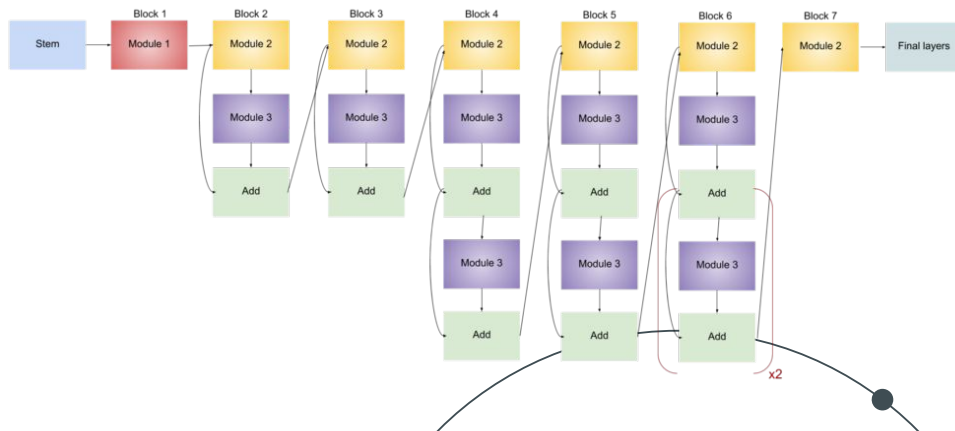
$$f_{out} = \gamma(c) \frac{f_{in} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta(c)$$


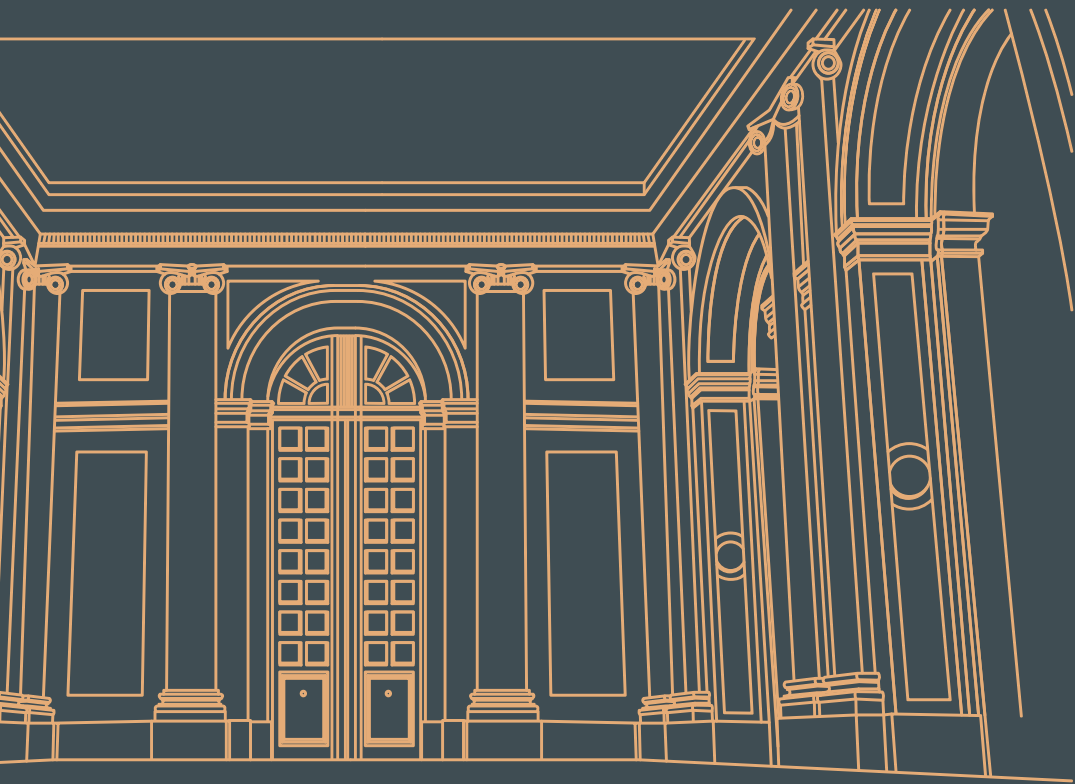
Encoders (Contd.)



Several aspects of the EfficientNet architecture make it a favorable choice for our experiments, as we also plan to show in a short ablation study over the full set of experiments.

We also experiment with EfficientNets as our encoder networks considering their success for image classification. We extract the 1280-d feature vector and process it to be 128-d or 256-d for further usage. The decoder is similar to the one described previously.





04


Future Steps

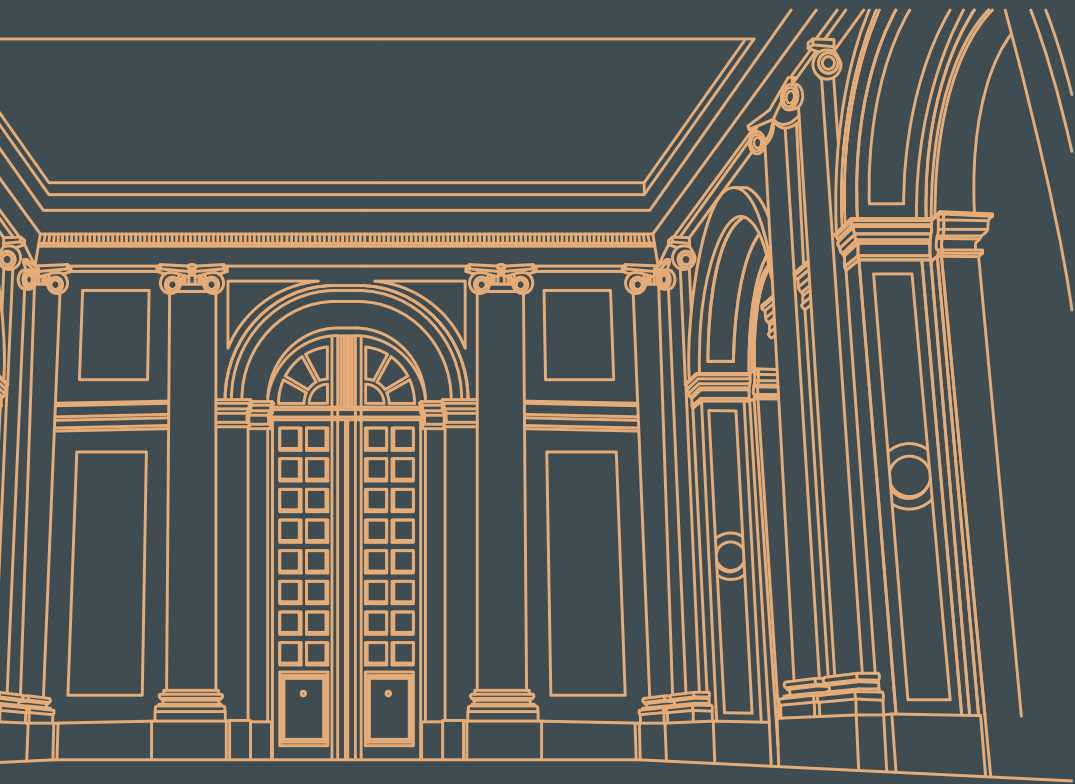




Next Steps



- As per the tasks outlined in the expected deliverables and the timeline, the next tasks comprise of preparing the end-to-end training pipeline for occupancy networks on the data subset (10% of entire dataset).
 - Additionally, to organise the code structure better and allow for more flexibility, we shall wrap the training scripts in pytorch-lightning format which would allow for smoother transitions across multiple devices/machines.
 - We shall prepare an ablation study on comparison of different encoder methods, architecture choices and effects of hyper-parameters on the results.
 - Additionally, an optional experiment with point cloud systems is also planned, where the input to the encode is a point cloud (sparse or incomplete). Network learns the object properties in such a way that we can generate the object in higher resolution as point cloud or prepare the object mesh from the same method as SVR (Single View Reconstruction).
- 



05

References

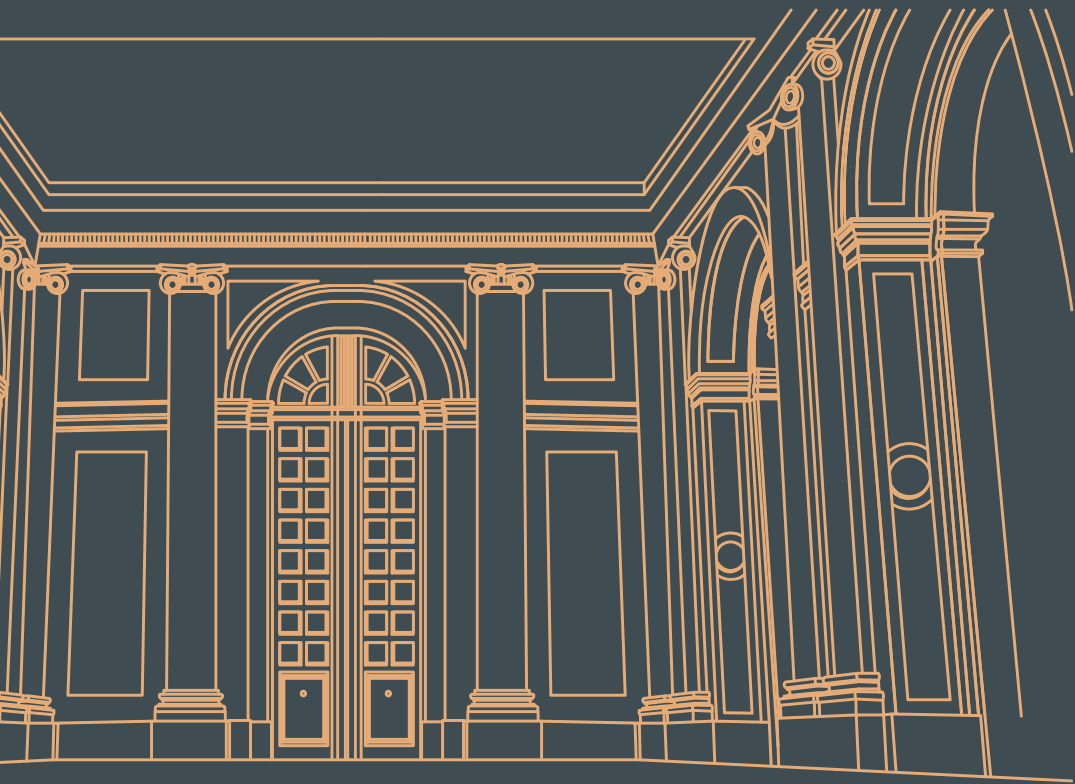




References



1. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. and Geiger, A., 2019. **Occupancy networks: Learning 3d reconstruction in function space**. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4460–4470).
2. Vakalopoulou, M., Chassagnon, G., Bus, N., Marini, R., Zacharaki, E.I., Revel, M.P. and Paragios, N., 2018, September. **AtlasNet: multi-atlas non-linear deep networks for medical image segmentation**. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 658–666). Springer, Cham.
3. Bartsch, M., Weiland, T. and Witting, M., 1996. **Generation of 3D isosurfaces by means of the marching cube algorithm**. *IEEE transactions on magnetics*, 32(3), pp.1469–1472.
4. M. Garland and P. S. Heckbert. **Simplifying surfaces with color and texture using quadric error metrics**. In Visualization'98. Proceedings, pages 263–269. IEEE, 1998.
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. and Xiao, J., 2015. **Shapenet: An information-rich 3d model repository**. *arXiv preprint arXiv:1512.03012*.
6. H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. **Modulating early visual processing by language**. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.



06

Endnotes

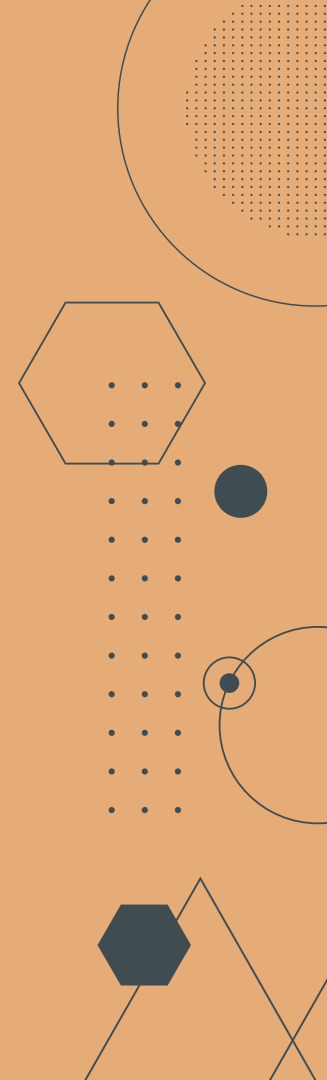


A vertical orange sidebar on the left side of the slide. It contains various geometric elements: a large circle with a small dark dot inside, a square with a diagonal line and a dotted pattern, a thick dark diagonal line, a small dark circle, and a series of dots arranged in a grid pattern.

Thanks!

...

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

A vertical orange sidebar on the right side of the slide. It contains various geometric elements: a large circle with a dotted pattern inside, a hexagon with a dotted pattern inside, a small dark circle, a series of dots arranged in a grid pattern, a small circle with a dark dot inside, and a hexagon with a dark fill.