Published in Towards Data Science

Rishab Sharma Follow
Jan 15, 2021 · 12 min read · Listen

Save

STATE OF AI

Deep Learning for 3D Synthesis

Transformers vs Convolutional Deep Networks for synthesizing 3D Data

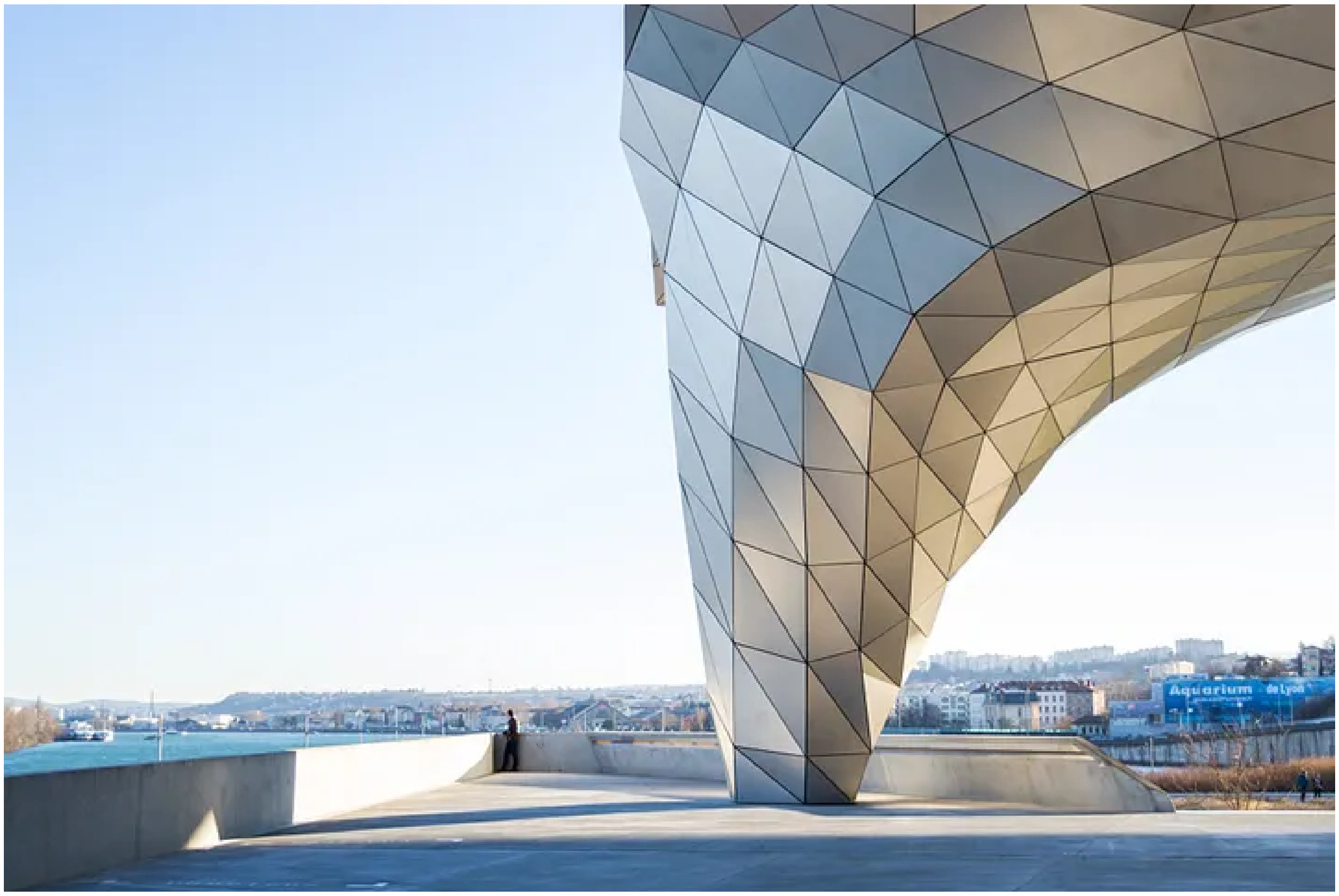
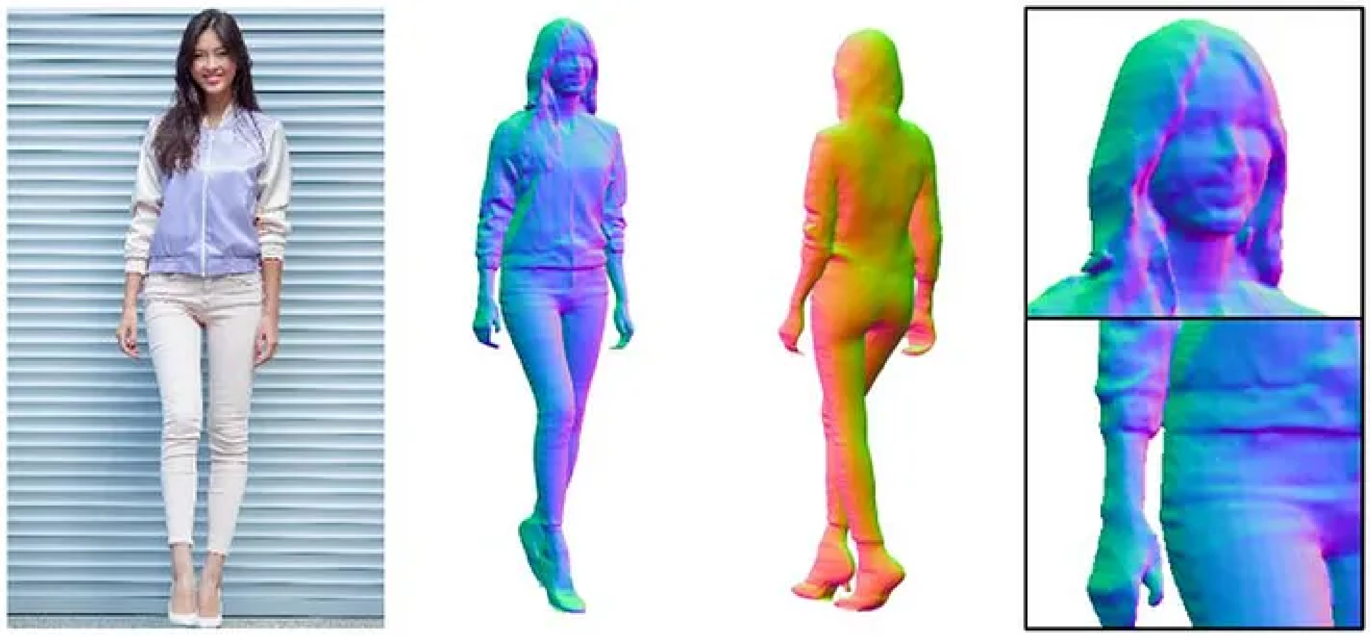


Photo by Diogo Nunes on Unsplash

Introduction to 3D Data

It's a consensus that synthesizing 3D data from a single perspective is a fundamental human vision functionality which is extremely challenging for computer vision algorithms. But recent advancements in 3D acquisition technology have taken a great leap after the increased availability and affordability of 3D sensors like LiDARs, RGB-D cameras (RealSense, Kinect) and 3D scanners. Unlike the widely available 2D data, 3D data is rich in scale and geometry information, thus provides an opportunity for better environment understanding for machines. However, the availability of 3D data is relatively lower along with a higher acquisition cost as compared to 2D data. Therefore, recently many deep learning approaches have been proposed to synthesize 3D data from the available 2D data without relying on any 3D sensors. But before we dive into these approaches, we should understand the format in which the 3D data is handled.



3D Synthesis from 2D Input

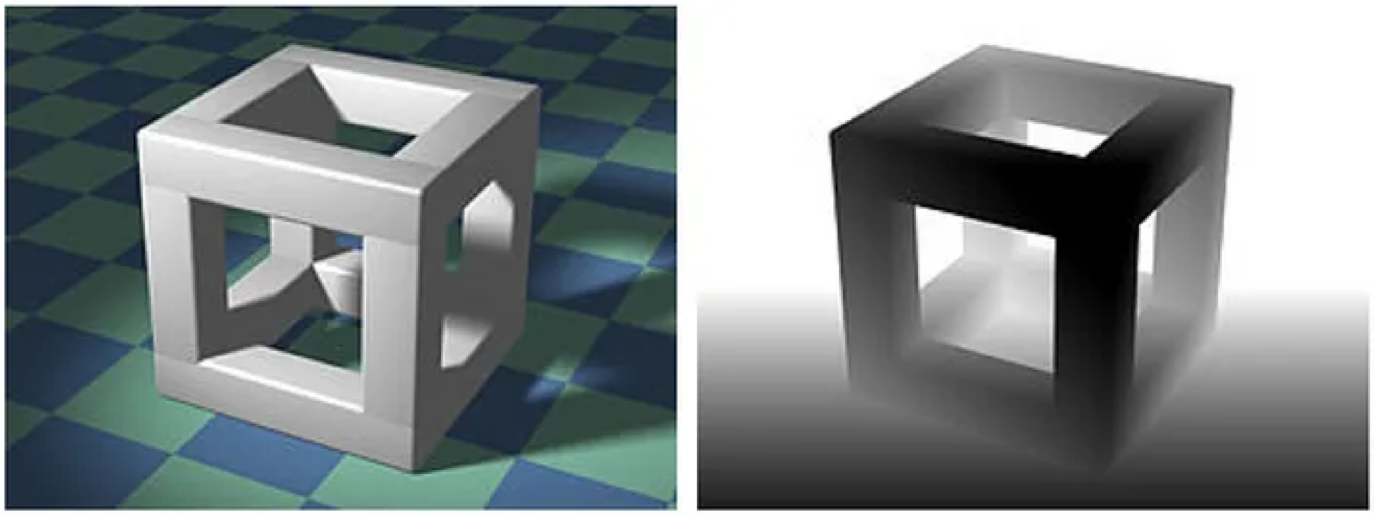
The synthesized three-dimensional data can be represented with different formats based on the final utility of the data. Some commonly used formats are:

1. Depth images

2. Point clouds

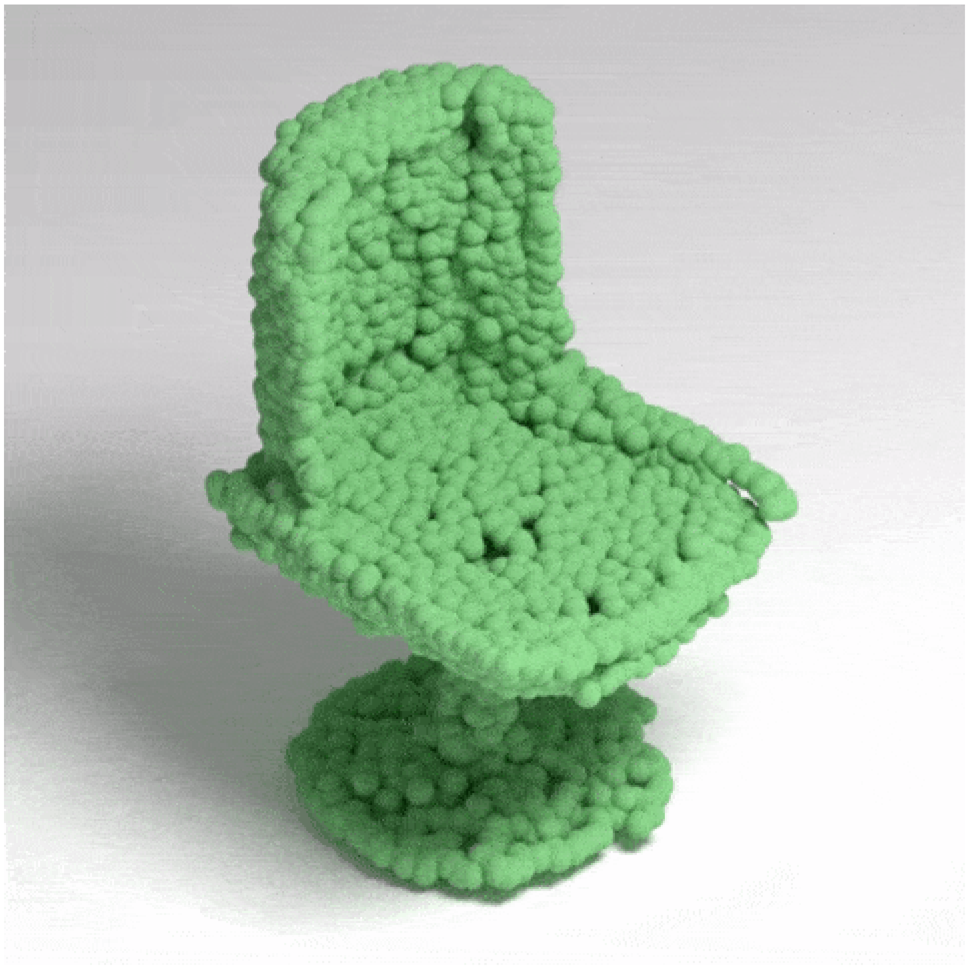
3. Voxels
4. Meshes

Depth images contain the depth values of a scene in the form of distance from the camera in meters for each pixel in the image frame. This depth information from a scene hold immense value for many tasks like self-driving cars, augmented reality, robotics, etc. Such information is very useful for tasks like enabling motion parallax as the camera scans over a still scene and animation in virtual cameras, but when emphasizing on a particular object in the scene for 3D modelling, this information becomes insufficient and inaccurate at the present state of the art.



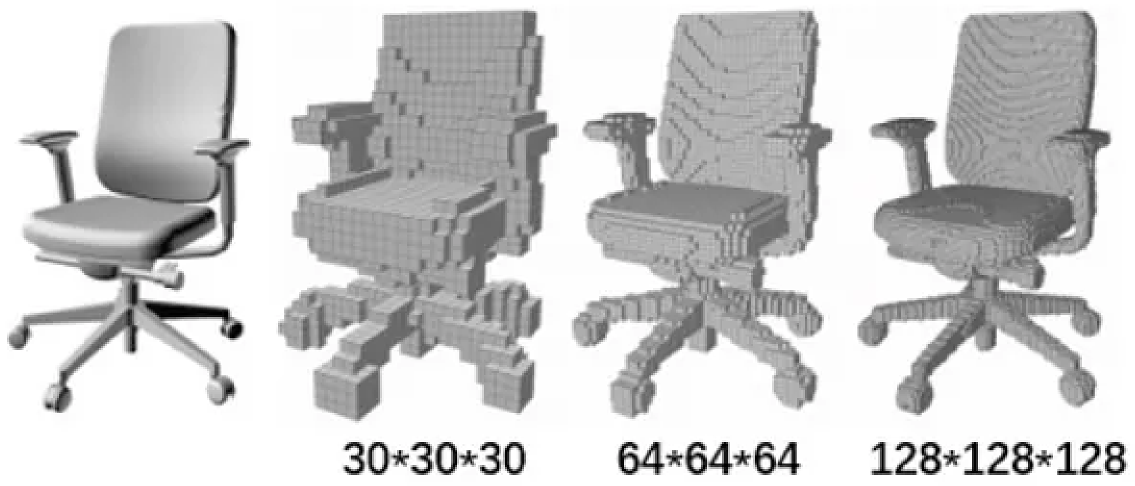
Depth Image

A Point cloud is a collection of three-dimensional points distributed in a 3D space. Each of these 3D points has a deterministic position denoted by a certain (x, y, z) coordinate along with other attributes like RGB colour values. Unlike depth images, point cloud representation preserves more high-quality geometric information of the three-dimensional space without any discretization. However, the point cloud representation has no local connections between points, thus leading to a very large degree of freedom and high-dimensionality that makes it more difficult to synthesize accurately.



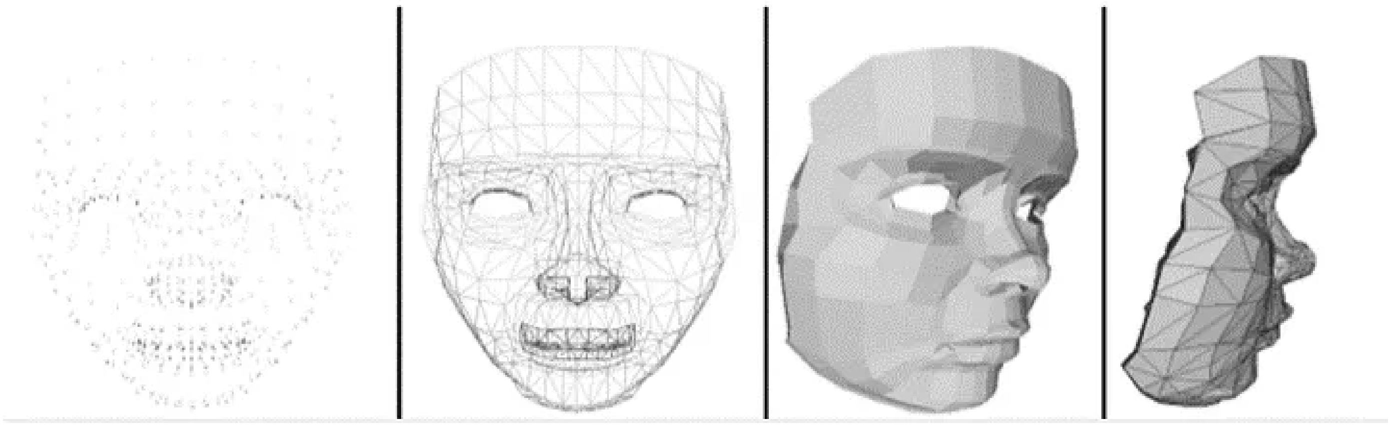
Point Cloud Representation

A Voxel or a volumetric pixel is the direct extension of a spatial-grid pixel into the volume-grid voxel. In simple words, a voxel is just a pixel in a three-dimensional space. The relative locality of each voxel together defines the unique structure of the volumetric data. A voxel can be regarded as a quantized point cloud with a fixed size. However, for 3D modelling, voxel representations are too sparse and show a tradeoff between details and computational resources, which make it more infeasible for the same.



Voxel Representation

A Polygon Mesh is a collection of edges, vertices and faces that together defines the shape and volume of a polyhedral object. The convex polygon faces of the mesh join together to approximate a geometric surface. Similar to a voxel, a mesh can also be regarded as a three-dimensional point cloud set sampled from a set of continuous surfaces (relatively lower complexity). Mesh faces can be triangular (triangle mesh), quadrilateral (quad mesh) or a convex polygon (n-gon mesh). Approaching a more realistic representation, a mesh can also consist of polygons with holes or concave polygons depending on generalisability of the representation. However, unlike voxels and point cloud that lose important surface details and are non-trivial to reconstruct a surface model, a mesh is more desirable for many real applications. Therefore considering the above points, a polygon mesh seems to be more realistic and better representation to be synthesized as compared to other formats.



Polygon Mesh Representation

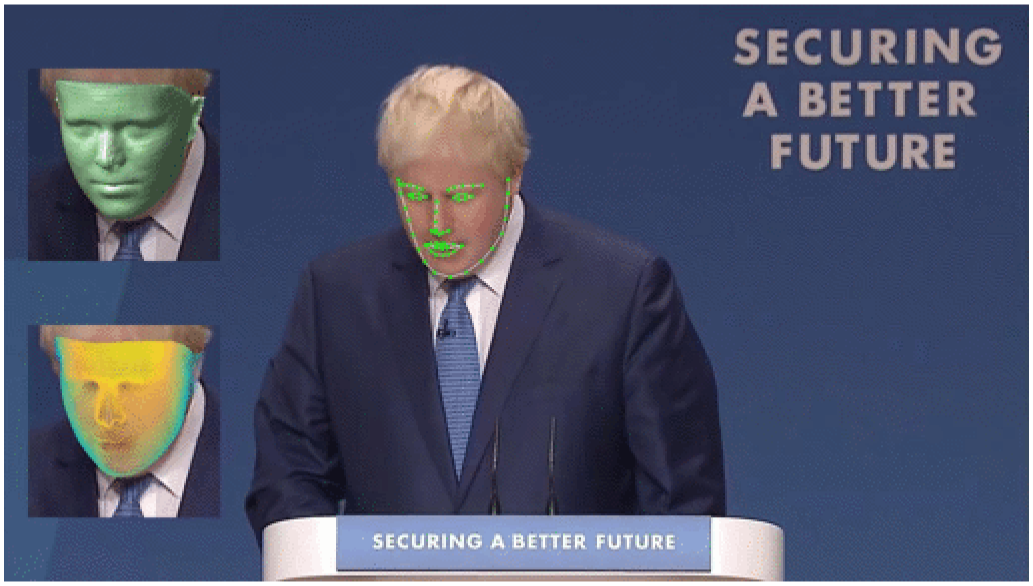
In this blog, we will discuss three types of approaches that can be used to synthesize 3D data from 2D data. Out of these three approaches, one approach is based on transformer-based architecture, whereas the other two are based on autoencoder and graph-based convolutional neural network respectively. The key difference between these two methodologies is that unlike the later, transformers based deep networks relies entirely on an attention mechanism to draw global dependencies between input and output.

Convolutional Neural Networks

In this section, we will discuss two types of recently proposed approaches that use autoencoder and graph-based convolutional neural networks to synthesize 3D data.

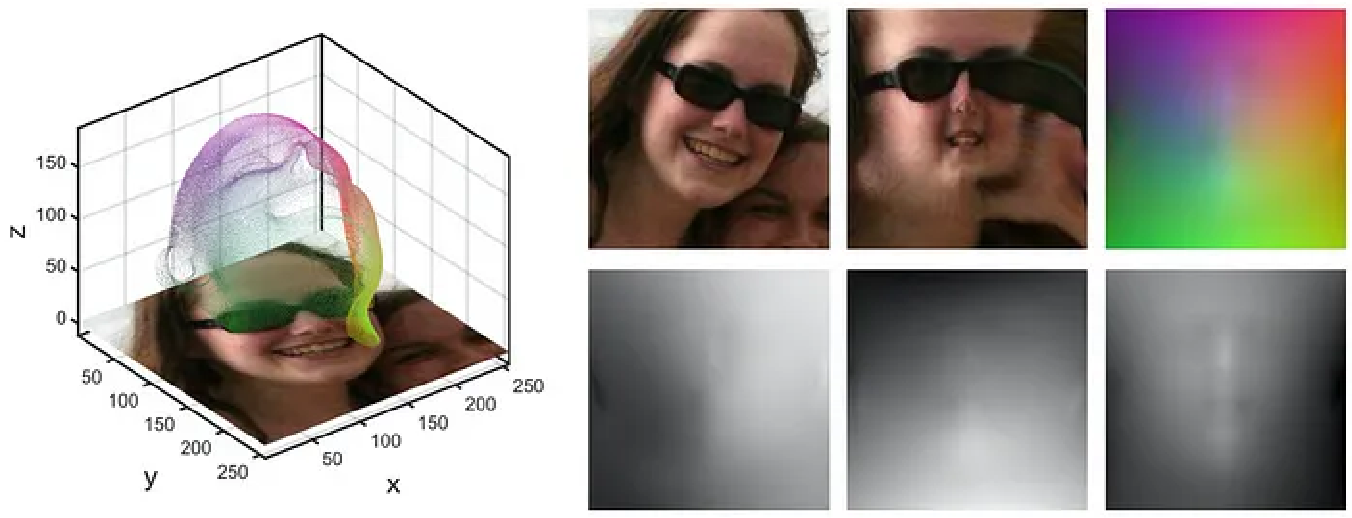
Autoencoder-based Convolutional Neural Networks

For understanding this approach, we would take the case of 3D face reconstruction and face alignment using an autoencoder network.



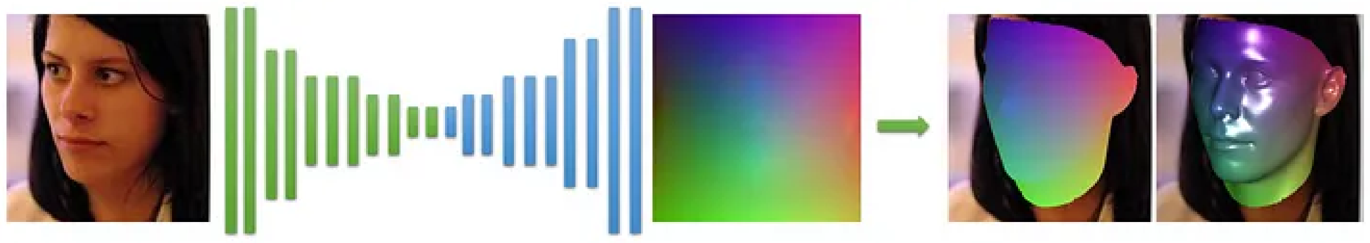
Face Reconstruction and Face Alignment

Autoencoders use a convolutional network to reduce the dimensionality of the input 2D image into a latent space, then use this latent space representation to reconstruct the original 3D data format. Many pieces of research have used autoencoders (encoder-decoder based architecture) to estimate 3D facial morphable model coefficients and model warping functions. The objective of these researches is mainly to use these 3D model warping functions to restore the corresponding 3D shape from a single RGB image, thus providing a dense 3D face alignment and reconstruction output at the same time. However, the performance of such methods is restricted by the limitation of the 3D representation space defined by face model templates. Similarly, approaches like Volumetric Regression Network (VRN) [1] use full convolutional layer architectures to estimate 3D binary volume as the discretized version of point clouds. However, most of the output points correspond to the wasteful non-surface points, and also this discretization limits the resolution of the output representation. Therefore, better research to discuss would be a network like Position Map Regression Network (PRN)[2] that jointly predict dense alignment and reconstruct the 3D face shape using a *UV position and texture map.*



The illustration of UV position and texture map

PRN constructs a 2D representation of the 3D facial structure in the form of a UV position map. A UV position is a 2D image that records the 3D facial coordinates of a facial point cloud. The map also attaches the semantic feature of the 3D coordinates at each position in the representation. In simple words, a UV map is a 2D representation of 3D data that records the 3D coordinates of all the points in a UV space. UV space and UV position maps are being used by researchers frequently in the domain of computer graphics to parameterize a 3D space into a 2D image plane. Moving to the network architecture of the PRN, the network takes an input RGB image and transfers the 2D image information into the UV position map using a simple encoder-decoder structure (autoencoder). The autoencoder fits this transfer functions with the use of ten downsampling residual blocks and seventeen upsampling transposed convolutional blocks to finally predict a 256×256×3 UV position map. The learned UV position map helps in directly regressing the final 3D structure and semantic feature for the 3D face to be synthesized. For training networks like PRN, we just need datasets like the 2D image to 3D point cloud mapping, which makes this approach more usable as the network's output format is not restricted by a specific 3D template or 3D morphable model linear space.

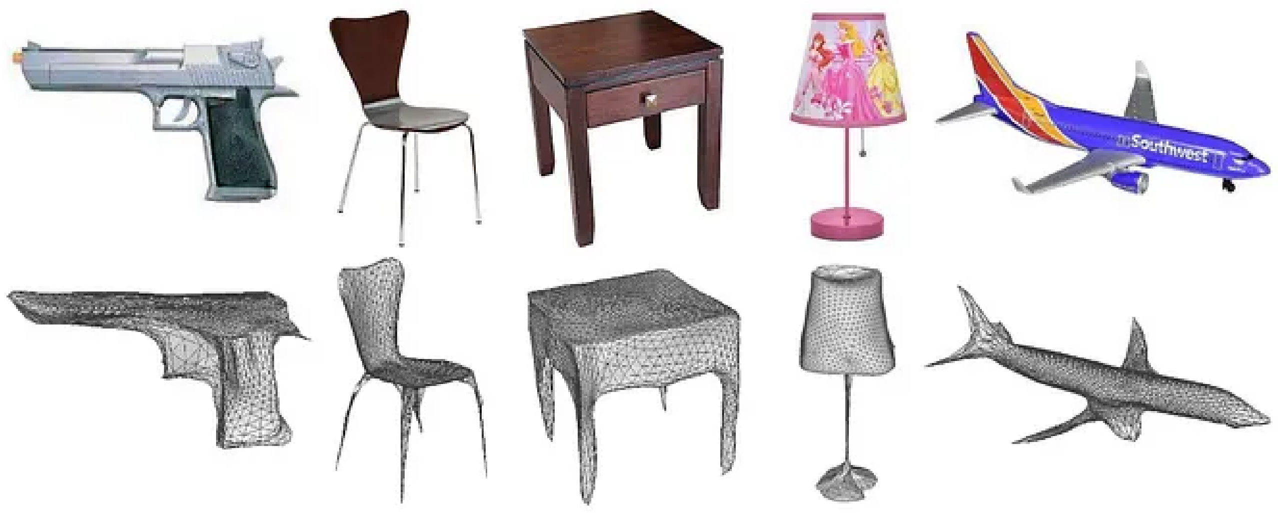


The architecture of PRN

Therefore considering the simplicity and effectiveness of the approach, PRN seems to be one of the best choices to synthesize 3D data from a 2D image for their network output format.

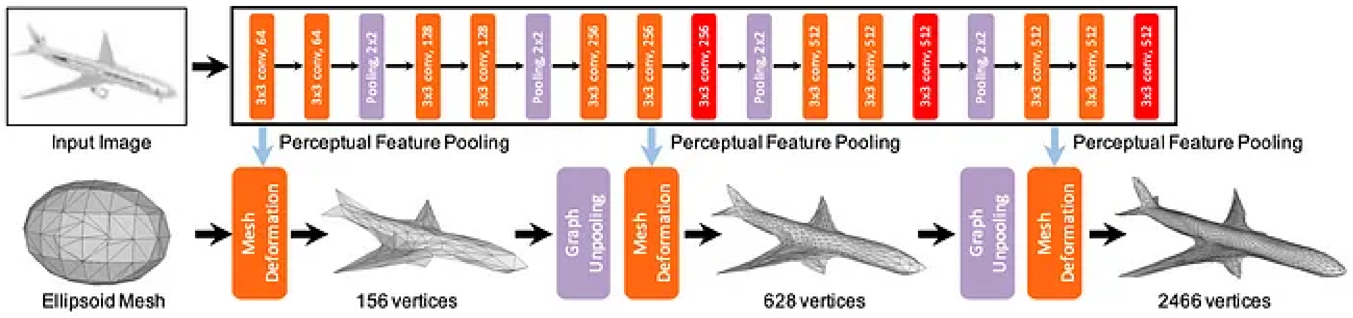
Graph-based Convolutional Neural Networks

As seen in the previous section, most of the conventional autoencoder based deep learning approaches have leveraged the point cloud and voxel data formats to synthesize 3D data. The main reason for the specific emphasis on these two data formats is the restriction imposed by the prevalent grid-based network architectures. However, a point cloud and a voxel representation have their own drawbacks as discussed in the first section. Therefore to avoid the drawbacks of a voxel or a point cloud representation, many pieces of research have moved on to synthesizing 3D polygon mesh data, which is a more desirable format from the application point of view. Some of the best architecture design approaches that work very well on synthesizing mesh data include graph-based convolutional neural networks. In this section, we will take the example of the approach proposed by Wang et. al [3] (Pixel2Mesh).



Reconstruction of Real-World images using Pixel2Mesh

Pixel2Mesh is a graph-based end-to-end deep learning framework that takes a single RGB colour image as input and transfers the 2D image to a 3D mesh model in a more desirable camera coordinate format. The graph-based convolutional neural network extracts and leverages the perceptual features in the 2D image to produce a 3D mesh by progressively deforming an ellipsoid until it reaches the semantically correct and optimised geometry. The adopted strategy is a coarse-to-fine methodology that makes the ellipsoid deformation procedure geometrically smooth and stable. The authors also define various mesh-related loss functions that help the network to capture more properties that guarantee a physically and visually appealing 3D geometrical result.

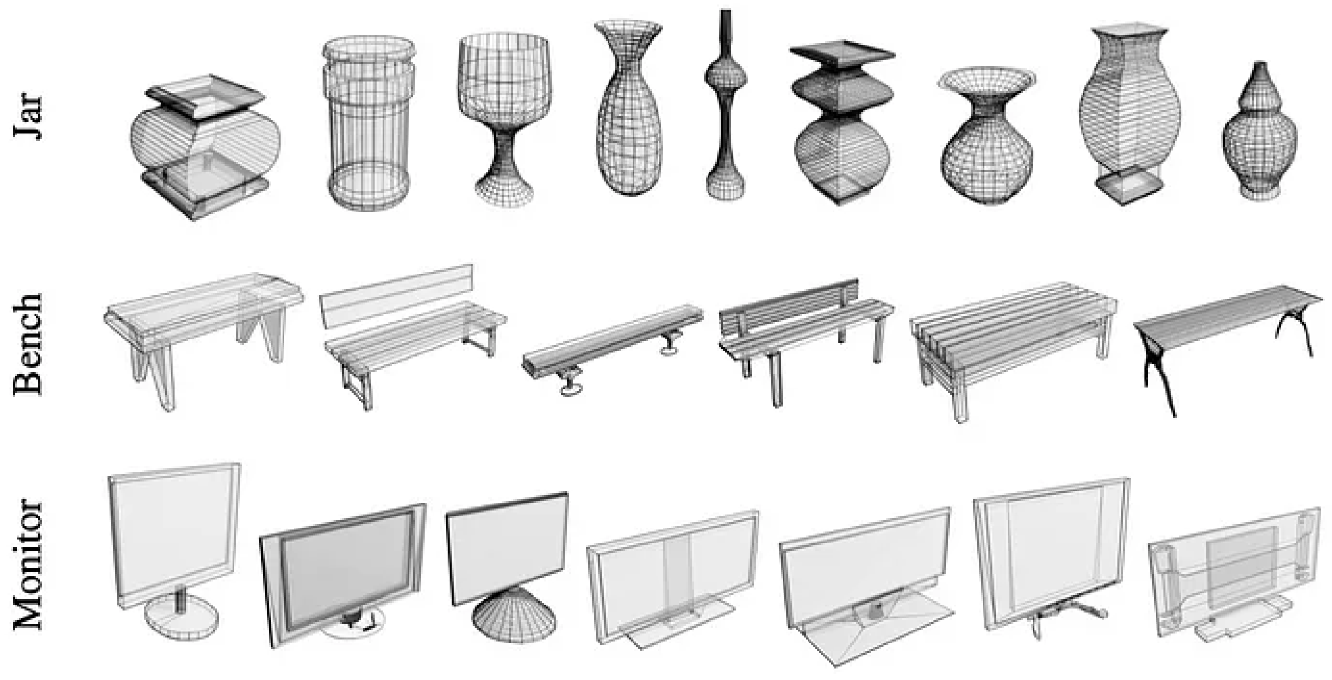


The Image feature network and cascaded mesh deformation network

The architecture of Pixel2Mesh mainly comprises of a cascaded mesh deformation network and an image feature network. The image feature network is responsible for perceptual feature extraction from the input 2D image and progressively passes these feature to the graph-based cascaded mesh deformation network to progressively deform the ellipsoid mesh's geometry into the 3D mesh of the target object. The graph convolution network of the mesh deformation network contains three deformation blocks along with two intermediate graph-unspooling layers. The deformation blocks progressively process the input graph of the mesh model, whereas the intermediate graph-unspooling layers progressively increase the graph vertices to increase the graph's information holding capacity, while still maintaining the triangular mesh formation of the data. Apart from the architectural details, one of the key advantages of Pixel2Mesh's graph-based architecture is the ability to perform simultaneous shape analysis similar to the conventional charting-based approaches that directly target the surface manifolds for the convolution operation. This approach acts as a bridge between the charting-based methods and 3D reconstruction methods by merging the two natural representation (graph and surface manifold) of meshed objects.

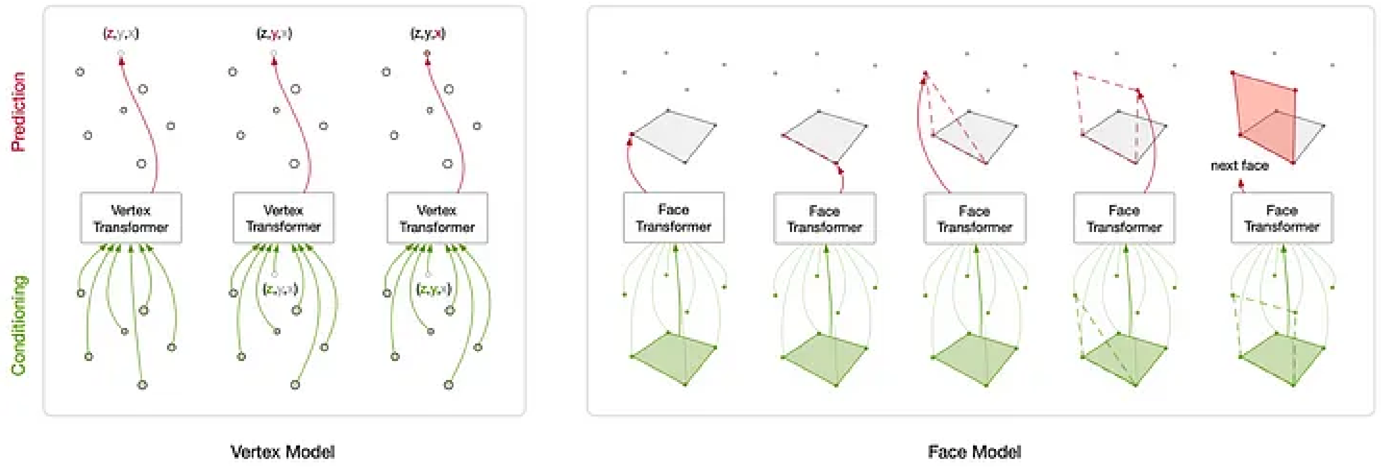
Transformer-based Deep Architecture

Convolutional neural networks are widely used for computer vision tasks because of their end-to-end ability to learn to perform the task directly from data without the requirement of any hand-designed visual feature. However, while the architecture design of CNNs is computationally demanding, the task of 3D synthetics make this computation more intense and opens a wide scope of computational optimisation and efficiency improvement. Looking forward to the next generation of neural architectures, transformers are the best family of scalable vision models that are not only domain agnostic but also computationally efficient and optimized. Moreover, recent researches have shown that transforms have achieved state-of-the-art results on many computer vision-based tasks. To understand the working of transformers in 3D data synthesis, we will take the example of Polygen by Deepmind [4].



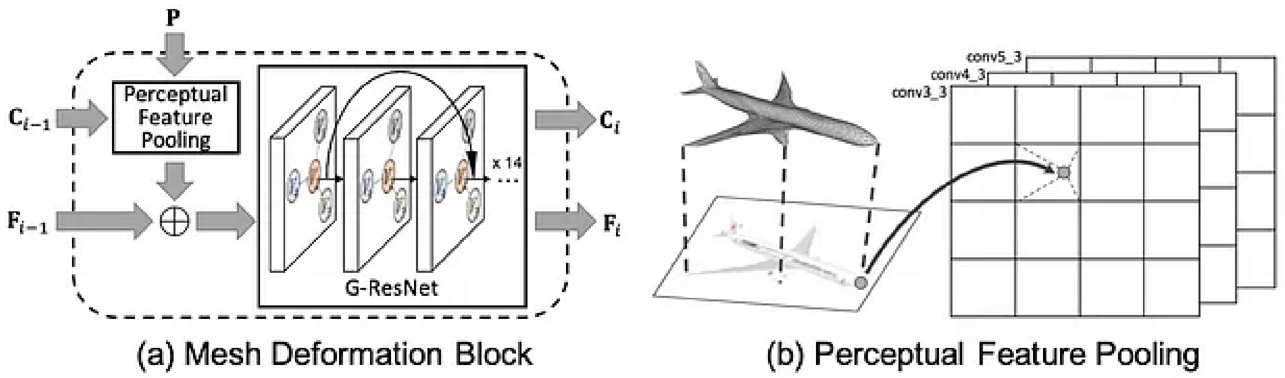
Samples generated by PolyGen

Polygen is an approach that models the n-gon 3D mesh directly by predicting the mesh faces and vertices sequentially using a transformer-based architecture. The model design is such that it can condition a range of input (object classes, voxels and 2D images) and probabilistically produce outputs that capture uncertainty in ambiguous scenarios. The network consists of the vertex model and the face model. The vertex model is a masked transformer decoder that unconditionally expresses a distribution over vertex sequences to model the mesh vertices. Whereas the face model is a pointer network-based transformer that conditionally expresses a distribution over variable-length input vertex sequences to model the mesh faces. Therefore in simple words, the goal of the two transformer models is to estimate a distribution over 3D meshes by first generating the mesh vertices and then use these vertices to generate the mesh faces.



Transformer-based Vertex and Face Model of Polygen

The transformer architectures used in Polygen is inspired by sequential models like WaveNet, PixelRNN and pointer networks. The work also derives its significant inspiration from Polygon-RNN (segmentation using polygons), whereas the vertex model is similar to the bespoke self-attention architecture of PointGrow [8] that models 3D point clouds using an autoregressive decomposition. As compared to the order autoregressive models, PointGrow has a shallower self-attention architecture that predicts discrete coordinate distribution using the self-attention mechanism by operating on a fixed-length point cloud input. Therefore Polygen can be considered as a well-balanced ensemble of some of the best ideas combined through the means of a novel transformer-based network design.


P
Perceptual Feature Pooling
C_{i-1}
F_{i-1}
C_i
F_i
x 14
G-ResNet
conv5_3
conv4_3
conv3_3
(a) Mesh Deformation Block
(b) Perceptual Feature Pooling

Mesh Deformation Block and Perceptual Feature Pooling Operation

One of the key features of Polygen is the ability of conditioning the output based on the input context (context example: 2D image, object class). To achieve this conditional nature, the input flow of the vertex and face model is altered to incorporate the context. For input formats like 2D images and voxels, the input is first encoded using a domain-appropriate encoder to retrieve a contextual embedding for the transformer decoder to perform cross-attention into the embedding sequence. Whereas for input formats like object classes, a pre-learned class embedding is projected to a vector that is added to the intermediate transformer predicted representation after the self-attention layer located in each network block. This is simply possible because of the generic nature of the vertex model that uses a simple, expressive and high modelling capacity transformer-decoder architecture which allows the network to model data from various domains. The transformer uses its ability of efficient information aggregation to capture the strong non-local dependencies present in the mesh vertices and the object geometry.

Conclusion

In this blog, we discussed two main types of 3D synthesis approaches i.e convolutional deep nets and transformer-based deep nets. The transformers being of a newer generation of networks are designed in a more computationally efficient and optimized manner, thus can be considered a step ahead of the convention convolutional nets. However, approaching a real-time inferencing scenario, transformers still have a long way to go for them to catch up to the approach we discussed in the autoencoder section that is comparatively light-weight and fast to infer. Nevertheless, transformers hold a huge scope of research and their attention mechanism's ability to efficiently aggregate information and extract global dependencies between input and output makes them even more promising.

Source

My blogs are a reflection of what I worked on and simply convey my understanding of these topics. My interpretation of deep learning can be different from that of yours, but my interpretation can only be as inerrant as I am.

References

[1] Jackson, A.S., Bulat, A., Argyriou, V., Tzimiropoulos, G., Jackson, A.S., Bulat, A., Argyriou, V., Tzimiropoulos, G.: Large pose 3d face reconstruction from a single image via direct volumetric CNN regression. In: International Conference on Computer Vision. (2017)

[2] Joint 3D Face Reconstruction and Dense Alignment with Position Map Regression Network

[3] Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images (ECCV2018)

[4] PolyGen: An Autoregressive Generative Model of 3D Meshes

[5] Sun, Y., Wang, Y., Liu, Z., Siegel, J. E., and Sarma, S. E. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *Winter Conference on Applications of Computer Vision*, 2020

Deep Learning Artificial Intelligence Machine Learning Transformers Convolution Neural Net