Computer Vision - 2025

Week #09. 3D Image Processing

Lectures by Alexei Kornaev [1,2,3]
Practical sessions by Kirill Yakovlev [2]

[1]AI Institute, Innopolis University (IU), Innopolis
[2]Robotics & CV Master's Program, IU, Innopolis
[3]Dept. of M^2R, Orel State University, Orel
[4]RC for AI, National RC for Oncology, Moscow

March 17, 2025

Agenda

QA - Session

Questions:

- Does the third dimension (time or depth) matter?
- Is it difficult to transfer a few video frames into a 3D image?
- Are 3D CNNs applicable for videos or 3D images processing?
- Does the third dimension (time or depth) matter in classification, or in segmentation, or in OD?
- How to solve a problem of multiple static object detection using video data (e.g. a person with a camera revises a car or a flat)?
- 3D CNNs or 2D CNNs + LSTM, or 3D Transformers?

Section 1. Introduction

Does 3D Matter?

The Third Dimension in Computer Vision

- **2D vs 3D**. Traditional 2D vision works on flat images, while 3D vision processes volumetric data (e.g., depth, time, or spatial structure).
- **Why 3D?** Many real-world problems require understanding spatial relationships, depth, or temporal dynamics.

A Few More Questions

- How do we reconstruct and process 3D images effectively?
- Does depth/time improve classification, segmentation, or object detection, or overfits them?

Section 2. 3D Image Reconstruction

Let’s Refer to HuggingFace

Please Check Unit 8 of the HF Community Computer Vision Course

and the Following Paragraphs:

- Representations for 3D Data
- Novel View Synthesis
- Neural Radiance Fields (NeRFs)

Video Frames to 3D: Rendering with a Smartphone Application

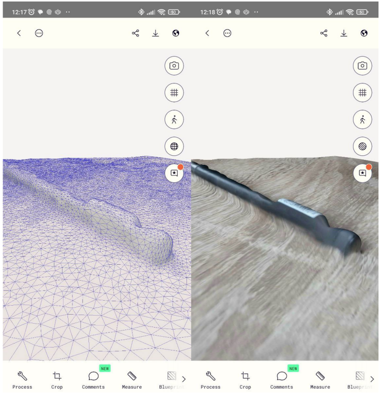

Figure: 3D model reconstructed from a video (about 20 frames): surface mesh (left) reconstructed pen (right).

Section 3. Image Processing with 3D CNNs

Recap (2D): Convolution Applied to an Image

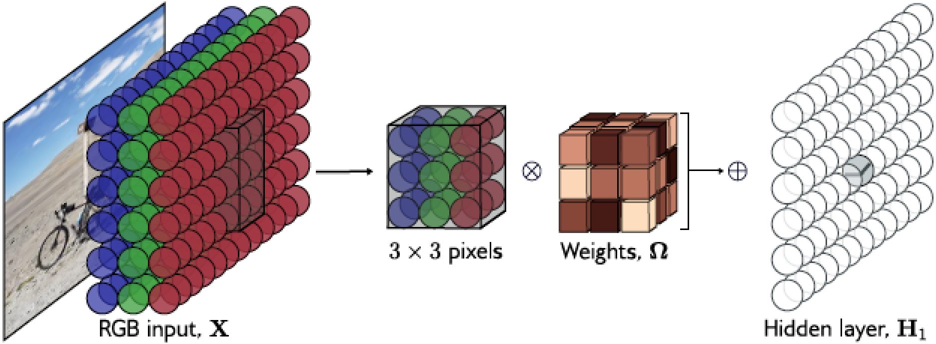

Figure: The image is treated as a 2D input with three channels corresponding to the red, green, and blue components. With a 3×3 kernel, each pre-activation in the first hidden layer is computed by pointwise multiplying the $3 \times 3 \times 3$ kernel weights with the 3×3 RGB image patch centered at the same position, summing, and adding the bias. To calculate all the pre-activations in the hidden layer, we âslideâ the kernel over the image in both horizontal and vertical directions. The output is a 2D layer of hidden units. To create multiple output channels, we would repeat this process with multiple kernels, resulting in a 3D tensor of hidden units at hidden layer H_1 [Prince, 2023]

Recap (2D): Padding, Pooling, and Striding

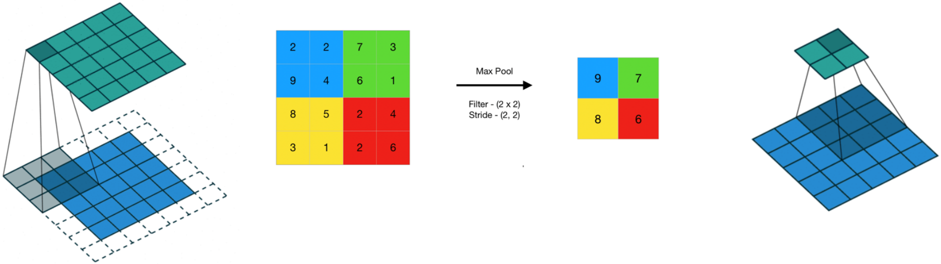

Figure: Padding (left), Pooling (middle), and Striding (right). CNNs by Neurohive.

Convolution Operations: Dimensions & Tensor Ranks

Table 1: Input/Output Tensor Sizes

	Input Size	Filter Size	Output Size
1D	$N \times C_{in} \times L$	$C_{out} \times C_{in} \times k$	$N \times C_{out} \times \left\lfloor \frac{L+2p-k}{s} + 1 \right\rfloor$
2D	$N \times C_{in} \times H \times W$	$C_{out} \times C_{in} \times k \times k$	$N \times C_{out} \times \left\lfloor \frac{H+2p-k}{s} + 1 \right\rfloor \times \left\lfloor \frac{W+2p-k}{s} + 1 \right\rfloor$
3D	$N \times C_{in} \times D \times H \times W$	$C_{out} \times C_{in} \times k \times k \times k$	$N \times C_{out} \times \left\lfloor \frac{D+2p-k}{s} + 1 \right\rfloor \times \left\lfloor \frac{H+2p-k}{s} + 1 \right\rfloor \times \left\lfloor \frac{W+2p-k}{s} + 1 \right\rfloor$

Table 2: Tensor Ranks & Convolution Directions

Conv Type	Input Rank	Filter Rank	Convolution Directions
1D	3	3	1 (length)
2D	4	4	2 (height, width)
3D	5	5	3 (depth, height, width)

Where N is the batch size, C is the number of channels, L is the (signal) length, H, W, D are the height, width, and depth, respectively, k is the kernel size, p is padding, and s is stride

3D CNN: Mathematical Formulation

3D Convolution Operation

Extends 2D convolution to volumetric data. For input tensor $\mathcal{X} \in \mathbb{R}^{C \times D \times H \times W}$:

$$\mathcal{Y}^{(l)}_{d,h,w} = \sum_{c=0}^{C-1} \sum_{i=0}^{k_d-1} \sum_{j=0}^{k_h-1} \sum_{k=0}^{k_w-1} \mathcal{W}^{(l)}_{c,i,j,k} \cdot \mathcal{X}^{(l-1)}_{c,d+i,h+j,w+k} + b^{(l)}$$

where $\mathcal{W} \in \mathbb{R}^{C \times k_d \times k_h \times k_w}$ is 3D kernel Maturana and Scherer [2015].

Key Advantages:

- Captures spatial-temporal features
- Preserves 3D structure
- Robust to viewpoint changes

Architectural Innovations in 3D CNNs

VoxNet ?

- Input: 32x32x32 occupancy grid
- Architecture: - Conv3D(32, 5x5x5) - MaxPool(2x2x2) - FC(128) $\rightarrow$ FC(n_classes)
- Application: Real-time object recognition

O-CNN Wang et al. [2017]

- Octree-based sparse convolution
- Adaptive depth partitioning
- Memory efficiency: $\mathcal{O}(N \log N)$ vs dense $\mathcal{O}(N^3)$
- Handles high-res 3D data (up to 512^3)

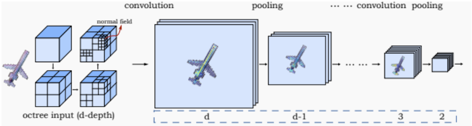

Figure: Octree structure in O-CNN Wang et al. [2017]

3D CNN Applications & Trade-offs

Key Applications

- Robotics: VoxNet for object grasping ?
- Medical Imaging: 3D tumor segmentation
- Autonomous Driving: LiDAR processing

Memory-Computation Trade-off

$$\text{Complexity} = \underbrace{C_{\text{in}} \times C_{\text{out}} \times k_d \times k_h \times k_w}_{\text{3D Kernel Parameters}} \times \underbrace{D \times H \times W}_{\text{Feature Map Size}}$$

- VoxNet: 32^3 grids $\rightarrow$ 32MB/scan
- O-CNN: Reduces memory by 70% Wang et al. [2017]

Key Insight

3D CNNs $\succ$ 2D when spatial relationships are critical, but require careful memory management

3D Transformers: Beyond 2D Attention

Core Idea

- Extend the **self-attention mechanism** to 3D data (point clouds, voxels, meshes).
- Model relationships between all parts of a 3D object/scene.
- No fixed grid assumption (unlike CNNs).

Key Questions

- How does self-attention work in 3D space?
- Are Transformers better than CNNs for 3D tasks?
- How to handle computational complexity?

How 3D Transformers Work

Key Components

- **Tokenization**: Convert 3D data (points/voxels) into tokens.
- **Positional Encoding**: Inject 3D coordinates (e.g., (x, y, z)).
- **Multi-Head Attention**: Compute interactions between tokens.

Self-Attention Formula

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

- Q, K, V: Queries, Keys, Values (projections of input tokens).
- d: Dimension of token embeddings.

Challenge

Quadratic complexity $O(N^2)$ for N tokens. Solutions: Sparse attention, window-based attention.

Voxel Transformer for 3D Object Detection

We present Voxel Transformer (VoTr), a novel and effective voxel-based Transformer backbone for 3D object detection from point clouds. Conventional 3D convolutional backbones in voxel-based 3D detectors cannot efficiently capture large context information, which is crucial for object recognition and localization, owing to the limited receptive fields. In this paper, we resolve the problem by introducing a Transformer-based architecture that enables long-range relationships between voxels by self-attention. Given the fact that non-empty voxels are naturally sparse but numerous, directly applying standard Transformer on voxels is non-trivial. To this end, we propose the sparse voxel module and the submanifold voxel module, which can operate on the empty and non-empty voxel positions effectively. To further enlarge the attention range while maintaining comparable computational overhead to the convolutional counterparts, we propose two attention mechanisms for multi-head attention in those two modules: Local Attention and Dilated Attention, and we further propose Fast Voxel Query to accelerate the querying process in multi-head attention. VoTr contains a series of sparse and submanifold voxel modules and can be applied in most voxel-based detectors. Our proposed VoTr shows consistent improvement over the convolutional baselines while maintaining computational efficiency on the KITTI dataset and the Waymo Open dataset. Mao et al. [2021].

Section 4. Conclusion & Discussion

3D CNNs vs 3D Transformers

Feature	**3D CNNs**	**3D Transformers**
Inductive Bias	Strong (local spatial patterns)	Weak (learns patterns from data)
Global Context	Limited	Excellent
Scalability	Small to medium datasets	Large datasets
Compute Efficiency	Efficient	Expensive
Data Requirements	Works well with small datasets	Needs large datasets
Use Cases	Medical imaging, robotics	Autonomous driving, 3D generation

Table: Comparison of 3D CNNs and 3D Transformers

Key Takeaways:

- Use **3D CNNs** for small datasets and local feature extraction.
- Use **3D Transformers** for large datasets and global context.
- Consider **hybrid models** for state-of-the-art performance.

Bibliography

Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection, 2021. URL https://arxiv.org/abs/2109.02497.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In **Ieee International Conference on Intelligent Robots and Systems**, pages 922–928, 2015.

Simon J.D. Prince. **Understanding Deep Learning**. The MIT Press, 2023. URL http://udlbook.com.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: octree-based convolutional neural networks for 3d shape analysis. **ACM Transactions on Graphics**, 36(4):1â11, July 2017. ISSN 1557-7368. doi: 10.1145/3072959.3073608. URL http://dx.doi.org/10.1145/3072959.3073608.