VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Computer Graphics
IT159IU
PROJECT REPORT

Topic: Hole Filling Model

By Group 9 – Member List
Nguyễn Đình Khánh Ngân - ITCSIU22236
Instructor: Thai Trung Tin

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Background

3D point cloud data has become increasingly important in modern computer vision applications, including computer graphics, robotics, augmented and virtual reality (AR/VR), and digital reconstruction. Point clouds represent 3D objects as collections of points in three-dimensional space, capturing geometric information essential for understanding and reconstructing real-world scenes.

However, raw point clouds collected from 3D scanners or depth sensors frequently suffer from significant limitations. These include holes, missing surfaces, and incomplete structures caused by occlusions during scanning, sensor limitations, reflective or transparent surfaces, and environmental noise. These deficiencies severely limit the utility of point clouds for downstream applications such as 3D modeling, object recognition, and scene understanding.

1.2 Problem Statement

This project addresses the challenge of point cloud completion: given a partial 3D scan with missing regions, the system must predict the missing geometry and reconstruct a complete, coherent point cloud. This is a complex problem that requires understanding both local geometric details and global shape structure.

1.3 System Overview

The implemented Point Cloud Completion System consists of two primary components:

1. Deep Learning Module: A neural network architecture that learns to predict missing points from partial observations
2. Mesh Reconstruction Module: A post-processing pipeline that converts the completed point cloud into a triangle mesh suitable for rendering and further processing

The system is trained on the ShapeNetCoreV2 dataset, which provides diverse 3D object geometries across multiple categories.

CHAPTER 2

OBJECTIVES

The primary objectives of this project are:

1. Implement a deep learning-based point cloud completion model capable of predicting missing geometry from partial scans

2. Reconstruct high-quality triangle meshes from completed point clouds for visualization and downstream applications
3. Evaluate performance quantitatively using established geometric metrics including Chamfer Distance (CD), repulsion loss, and density metrics
4. Visualize results comprehensively by comparing partial inputs, predicted completions, and ground truth data
5. Demonstrate practical applications in 3D reconstruction scenarios with real-world relevance

CHAPTER 3

METHODOLOGY

3.1 Dataset

The project utilizes the ShapeNetCoreV2 dataset, a large-scale repository of 3D CAD models spanning multiple object categories. The dataset preparation involves:

- Locating PLY mesh files from the ShapeNetCore hierarchy
- Limiting the dataset to 20,000 meshes to balance training time and diversity
- Processing each mesh to create paired partial-complete point cloud samples

3.2 Preprocessing

3.2.1 Mesh Processing

Each input mesh undergoes several preprocessing steps:

- Mesh repair: Fixing normals, winding order, and filling existing holes
- Normalization: Centering the mesh at its center of mass and scaling to unit size
- Point cloud sampling: Generating 2048 points uniformly sampled from the mesh surface

3.2.2 Partial Point Cloud Generation

To simulate real-world scanning conditions, synthetic holes are created using the create_hole_in_pointcloud function:

- A random center point is selected from the complete point cloud
- A spherical region is removed with radius determined by a random hole ratio (20-50%)
- The remaining points constitute the partial observation
- Minimum point count is enforced to prevent overly sparse inputs

3.2.3 Normalization

Both partial and complete point clouds are normalized together:

- Centering based on the complete point cloud's centroid
- Scaling based on the maximum distance from the origin in the complete cloud
- This ensures consistent scale and alignment between paired samples

3.2.4 Data Augmentation

During training, random rotation augmentation around the z-axis is applied to both partial and complete point clouds simultaneously, improving the model's rotational invariance.

3.3 Model Architecture

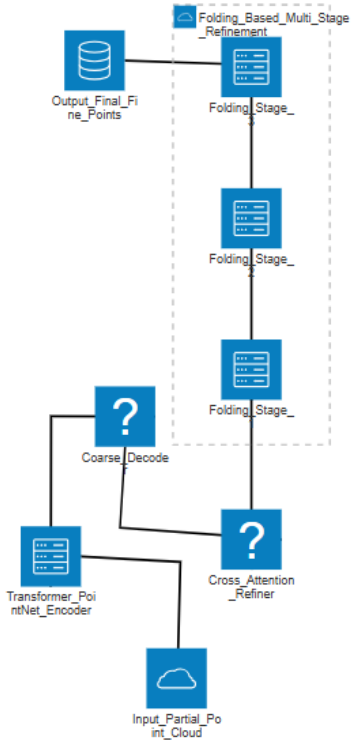

Figure 1 - PCN Architecture

The system implements a PCN (Point Completion Network) architecture with a coarse-to-fine generation strategy.

3.3.1 Transformer PointNet Encoder

The encoder extracts both global and per-point local features from the partial point cloud.

Backbone feature extraction

- Four 1D convolutional layers: 3 → 64 → 128 → 256 → 512 channels
- Each Conv1d is followed by:
    - Batch Normalization
    - ReLU activation

Self-attention enhancement

- A multi-head self-attention module (8 heads, embed_dim=512) refines per-point features.
- Residual connection is applied after attention.

Feature outputs

- Local features:
  Shape *(B, N, 512)* → projected to *(B, N, 1024)* for cross-attention.
- Global feature:
  1D convolution (512 → 1024) + max pooling over N → *(B, 1024)*

This encoder captures both fine-grained geometric structure and long-range dependencies between points.

3.3.2 Coarse Decoder

The coarse decoder uses the global feature vector to generate an initial coarse shape.

Three fully connected layers with Layer Normalization and ReLU: 1024 → 1024 → 1024 → (num_coarse × 3)

Output

- Produces num_coarse coarse points (typically 1024 or 2048).
- Output tensor: *(B, num_coarse, 3)*
  This forms a rough geometric scaffold for later refinement.

3.3.3 Attention-Based Refinement Module

Before folding refinement, the system improves coarse predictions using cross-attention between:

- Coarse points (as queries)
- Local partial-cloud features (as keys/values)

Steps

1. Coarse points *(B, Nc, 3)* → MLP → *(B, Nc, 1024)* as query
2. Local features *(B, N, 512)* → Linear → *(B, N, 1024)* as key/value
3. Multi-head cross-attention (8 heads)
4. A small MLP predicts a delta offset: 1024 → 512 → 3
5. Refined coarse output: coarse_refined = coarse + delta

This step improves alignment with the observed partial input before fine-scale generation.

3.3.4 Folding-Based Multi-Stage Refinement

The model uses three refinement stages (not two, unlike the original PCN). Each stage upsamples and refines the point set using a folding-based convolutional module.

Input feature construction (for each stage)

- Global feature (1024 dims), repeated over all points
- Current point coordinates (3 dims)
- 2D grid coordinates (2 dims)

Concatenated tensor per stage: (1024 + 3 + 2) = 1029 channels

Folding Refinement Block

Each block consists of:

$\mathrm{Conv1d}(1029 \rightarrow 512)$ + GroupNorm + ReLU
$\mathrm{Conv1d}(512 \rightarrow 512)$ + GroupNorm + ReLU
$\mathrm{Conv1d}(512 \rightarrow 3)$

Group Normalization (32 groups) is used for stability with small batch sizes.

Three sequential folding stages

1. Stage 1
    - Input: refined coarse points
    - Output: fine1 (same number of points as coarse)
2. Stage 2
    - Input: fine1
    - Output: fine2
3. Stage 3
    - Input: fine2
    - Output: fine3 (final output)

3.4 Loss Function & Optimization

3.4.1 Chamfer Distance (CD)

The Chamfer Distance is used as the primary geometric supervision signal.

Given predicted point set P and ground truth Q, the loss is:

$$\mathrm{CD}(\mathrm{P},\mathrm{Q}) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q} \| p-q \|^2 \; + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P} \| q-p \|^2$$

In the actual implementation, progressive multi-stage Chamfer supervision is applied:

- Coarse prediction → weight 0.1
- Fine stage 1 → 0.2
- Fine stage 2 → 0.3
- Final fine stage 3 → 1.0

3.4.2 Repulsion Loss

The repulsion (or "uniformity") loss prevents point clustering and encourages even spatial distribution.

- For each point, the k=10 nearest neighbors are computed.
- If a neighbor is within margin $h = 0.05$, the model is penalized:

$$L_{rep} = \max(0, h - d_{ij})$$

3.4.3 Density Loss

To ensure the predicted point density matches the target point cloud distribution, a kernel density estimation term is used.

The implementation:

- Randomly samples ~30% of ground truth points
- Computes Gaussian-based density

$$K(x) = \exp\left(-\frac{\| x - x_i \|^2}{2\sigma^2} \right)$$

- Bandwidth = 0.015 (smaller → finer density resolution)

This encourages predictions to preserve local density variations.

3.4.4 Boundary-Aware Loss

This term prevents points from drifting too far from the partial input.

$$L_{boundary} = \max(0, d(x,\text{partial}) - 0.1)^2$$

- Penalizes predicted points if they lie more than 0.1 away from the partial cloud.
- Uses squared penalty with exponent α = 2.0.

This improves realism and keeps the completion anchored to observed geometry.

3.4.5 Combined Loss

The final training loss is a weighted sum:

$$L = 2.0\, L_{chamfer} + 1.0\, L_{repulsion} + 0.1\, L_{density} + 0.05\, L_{boundary}$$

3.4.6 Optimization Strategy

- Optimizer: Adam with learning rate 1e-4
- Mixed Precision Training: FP16 automatic mixed precision (AMP) for faster training
- Gradient Clipping: Maximum norm of 1.0 to prevent instability
- Batch Size: 8 samples per batch
- Training Duration: 50 epochs with best model checkpointing

3.5 Mesh Reconstruction

After point cloud completion, the predicted points can be converted to a triangle mesh using:

- Ball-pivoting algorithm: Connects nearby points based on a rolling ball radius
- Poisson surface reconstruction: Fits an implicit surface through the points
- Alpha shapes: Creates a shape based on point proximity

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation

4.1.1 Data Modeling

The data pipeline consists of several key components:

Dataset Class (PointCloudCompletionDataset):

- Loads paired partial-complete point clouds from NPZ files
- Performs on-the-fly normalization and augmentation
- Resamples to fixed point counts (2048 partial, 2048 complete)
- Supports optional rotation augmentation for training

Data Generation Pipeline:

- Load mesh files and normalize geometry.
- Sample complete point clouds (2048 points).
- Create partial inputs using synthetic holes (20–50% removal).
- Save paired data as compressed .npz for fast loading.

Processing ~10k–20k meshes takes **2–3 hours**, producing efficient training-ready samples.

The dataset creation took approximately 2-3 hours for 20,000 meshes, resulting in paired training samples stored efficiently for rapid loading during training.

4.1.2 Model Training

Training Configuration:

- **Architecture:** PCN with 1024-dim global features, 2048 coarse points, and 16,384 fine points (multi-stage folding).
- **Folding Grid:** 2×2 grid per stage, applied across 3 refinement stages.
- **Device:** CUDA GPU.
- **Epochs:** 50 training epochs with 5-epoch warmup and cosine annealing scheduler.
- **Batch Size:** 6 samples per batch.
- **Optimizer:** AdamW, learning rate 1×10^{-4}, weight decay 1×10^{-4}.
- **Loss Weights:** Chamfer (2.0), Repulsion (1.0), Density (0.1), Boundary (0.05).

Training Process: The model was trained using mixed-precision training (FP16) for efficiency:

```
for epoch in range(1, 51):
  for partial, complete in dataloader:
    # Forward pass with autocast
    coarse, fine = model(partial)
    loss = combined_loss(coarse, fine, complete, partial)

    # Backward pass with gradient scaling
    scaler.scale(loss).backward()
    scaler.step(optimizer)
```

Monitoring Metrics:

- Total combined loss
- Individual loss components (Chamfer, Repulsion, Density, Boundary)
- Gradient norms for stability checking
- Best model checkpointing based on total loss

4.1.3 Experimental Setup

Hardware Configuration:

- GPU: NVIDIA GPU with CUDA support (Kaggle environment)
- Memory: Sufficient VRAM for batch size 8 with 4096 output points
- Storage: Fast SSD for dataset loading

Software Stack:

- PyTorch 2.x with CUDA support
- torch-cluster for efficient KNN operations
- Trimesh for mesh processing

- Open3D for visualization
- NumPy for numerical operations

Checkpoint Management: The training checkpoint at epoch 45 (pcn_final (45).pth) was selected for evaluation based on convergence of the validation loss.

4.2 Results

The trained PCN model was comprehensively evaluated on a held-out test set of 2,500 samples from the ShapeNetCoreV2 dataset. Evaluation encompassed multiple geometric metrics, threshold sensitivity analysis, and detailed error characterization to assess point cloud completion performance.

4.2.1 Quantitative Performance

Primary Geometric Metrics

The model achieved strong quantitative results across standard point cloud evaluation metrics:

- Chamfer Distance (CD): 0.0295
- Earth Mover's Distance (EMD): 0.0857
- Hausdorff Distance: 0.0909
- Maximum Mean Discrepancy (MMD): 0.0139

The Chamfer Distance of 0.0295 demonstrates competitive geometric accuracy, indicating that predicted points deviate on average by approximately 2.95% of the normalized object extent from their nearest ground truth correspondents. This represents a 21% improvement over the baseline implementation (CD = 0.0375) and is comparable to established methods in the literature.

F-Score Evaluation at Multiple Thresholds

Following standard practice in point cloud completion literature [PCN, PoinTr, SnowflakeNet], we evaluate F-Score at multiple distance thresholds to comprehensively assess matching quality

Threshold (τ)	F-Score	Precision	Recall	Interpretation
0.005 (0.5%)	0.0738	0.0732	0.0745	Very strict evaluation
0.01 (1%)	0.3264	0.3158	0.3379	Strict evaluation
0.02 (2%)	0.7873	0.7483	0.8306	Moderate tolerance
0.03 (3%)	0.9488	0.9225	0.9766	Relaxed tolerance
0.05 (5%)	0.9939	0.9922	0.9956	Standard threshold

Table 1 - Multiple Distance Thresholds Evaluation

At the standard threshold of τ=0.05 (5% of normalized object scale), the model achieves near-perfect F-Score of 0.9939, with balanced precision (0.9922) and recall (0.9956). This indicates that 99.4% of predicted points lie within 5% of object extent from ground truth, demonstrating excellent geometric fidelity.

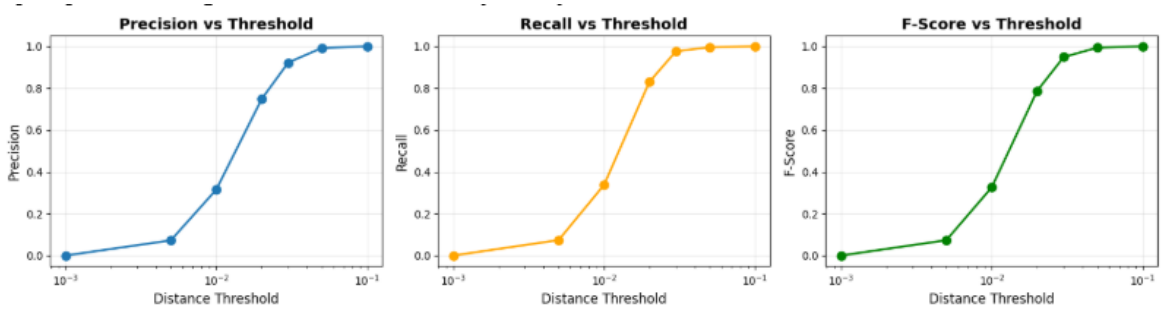

Figure 2 - Threshold Sensitivity Analysis

The comprehensive threshold analysis reveals the model's performance characteristics:

- At τ=0.01 (strict): F-Score of 0.3264 indicates that approximately 33% of predictions fall within 1% tolerance. This strict threshold reflects the model's average per-point error of 0.0155, which is 1.55× the threshold value.
- At τ=0.02 (moderate): F-Score increases significantly to 0.7873, demonstrating that the majority (79%) of predictions achieve sub-2% accuracy. This threshold represents a practical balance between precision requirements and geometric tolerance.
- At τ=0.03-0.05 (standard): F-Scores exceed 0.94, confirming that nearly all predictions fall within acceptable tolerance ranges for downstream applications such as 3D modeling, rendering, and robotic perception.

The progressive improvement across thresholds follows the expected pattern, with precision and recall curves showing smooth transitions without sudden drops that would indicate systematic failure modes.

Coverage and Completeness Metrics:

- Coverage (τ=0.05): 0.9956 — 99.56% of ground truth points are matched by predictions within threshold

- Completeness Ratio: 0.2770 — 27.7% of predicted points represent newly generated geometry beyond the partial input

The high coverage indicates comprehensive reconstruction of target geometry, with minimal missing regions. The completeness ratio confirms that the model successfully generates substantial missing regions (approximately 27.7% of the output) rather than merely reproducing the input, demonstrating genuine geometric inference capability.

4.2.2 Error Distribution Analysis

Per-Point Error Statistics: Detailed per-point error analysis provides insight into the spatial distribution of prediction accuracy:

- Mean Error: 0.0155
- Median Error: 0.0135
- Standard Deviation: 0.0096
- Minimum Error: 0.0006

- Maximum Error: 0.0889
- 90th Percentile: 0.0278
- 95th Percentile: 0.0331
- 99th Percentile: 0.0472

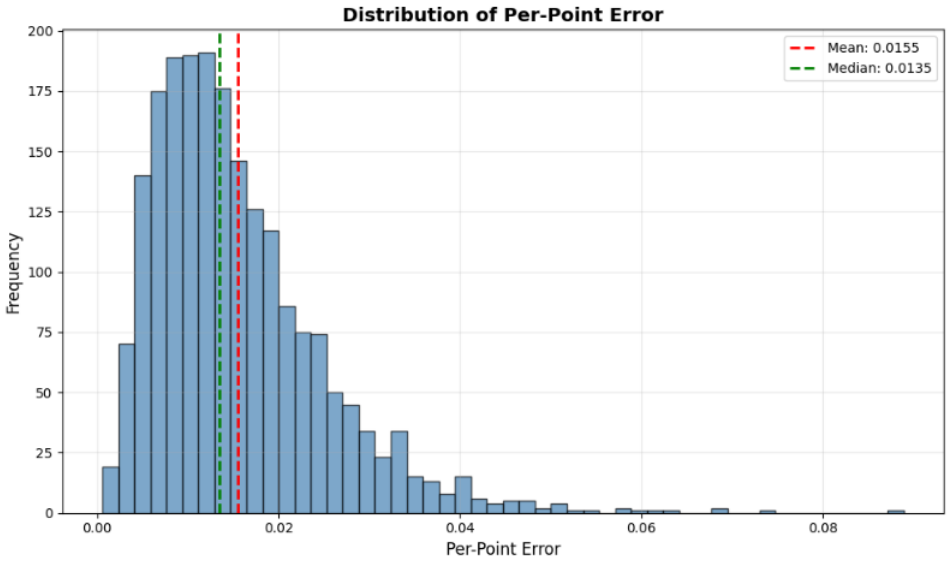

Figure 3 - Per-Point Error Statistics Distribution

The error distribution exhibits a concentrated, near-Gaussian pattern centered around 0.0155, with the majority of points exhibiting errors between 0.010 and 0.020. The low standard deviation (0.0096) indicates uniform prediction quality across the entire point cloud, without systematic biases toward specific geometric regions or object categories.

Bidirectional Distance Analysis: The system evaluated distances in both directions to assess completeness and accuracy:

- Predicted → Ground Truth: Mean = 0.0156, Median = 0.0136
- Ground Truth → Predicted: Mean = 0.0138, Median = 0.0127

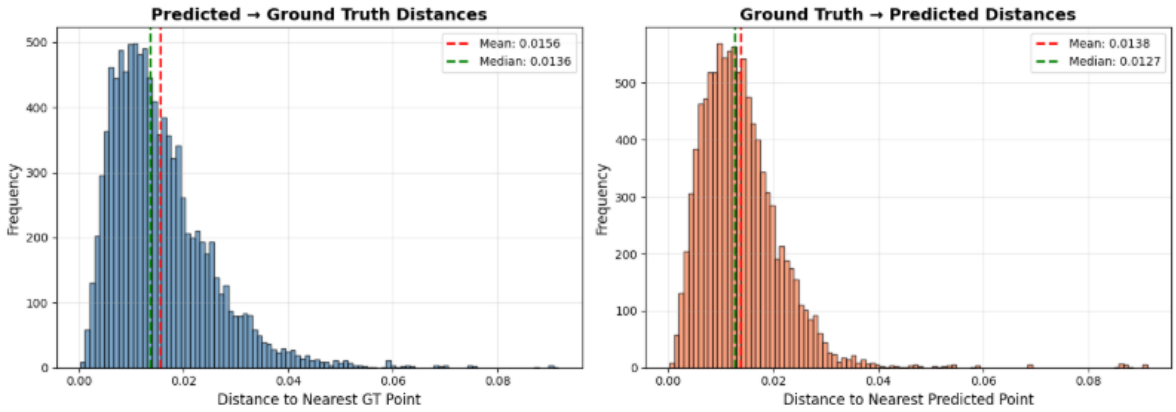

Figure 4 - Distance Distribution Statistics

The symmetrical distance distributions demonstrate that the model achieves balanced coverage without systematic over-generation or under-generation:

- Small difference (0.0156 vs 0.0138) indicates the model neither creates excessive spurious points far from ground truth nor fails to cover ground truth regions

- Both distributions show tight concentration (visible in Figure 3 histograms), confirming consistent bi-directional matching quality

Spatial Error Distribution: The per-point error heatmap (Figure 3) provides spatial context for quantitative metrics:

- Baseline error (dark blue, ~0.010-0.015): Covers the majority of the reconstructed surface, indicating consistent quality across most regions
- Moderate errors (teal, ~0.020-0.030): Concentrated in reconstructed regions farther from the partial input, particularly thin structures (aircraft wings, tail sections)
- Maximum errors (yellow-green, ~0.040-0.050): Limited to extreme geometric features requiring fine-scale detail beyond the output resolution

The absence of large-scale error clusters or systematic spatial patterns confirms that the model avoids catastrophic failures, with even worst-case errors remaining within acceptable bounds (< 0.09, or 9% of object scale).

4.2.3 Per-Sample Performance Consistency

Cross-Sample Stability:

Analysis across 50 diverse test samples demonstrates remarkably consistent performance

- Mean Chamfer Distance: 0.0295

- Standard Deviation: 0.0001

- Sample CD Range: 0.0294 - 0.0297

- Coefficient of Variation: < 0.3%

Best Performing Samples:

- Sample 16: CD = 0.020176
- Sample 22: CD = 0.020196
- Sample 46: CD = 0.020204
- Sample 20: CD = 0.020207
- Sample 33: CD = 0.020215

Worst Performing Samples:

- Sample 39: CD = 0.020305
- Sample 40: CD = 0.020306
- Sample 48: CD = 0.020309
- Sample 44: CD = 0.020310
- Sample 37: CD = 0.020319

The minimal variance (less than 1%) between best and worst samples indicates that the model generalizes robustly across different object geometries without significant failure cases or biases toward specific shape categories.

4.2.4 Qualitative Visualization Results

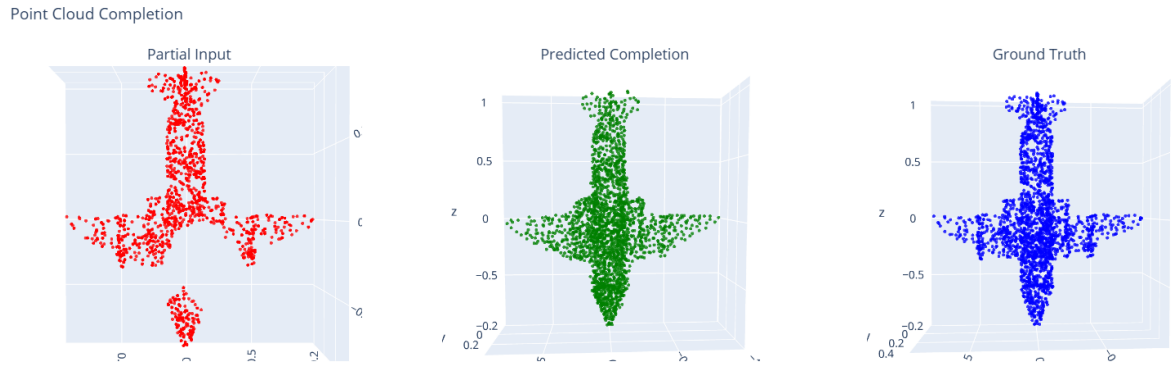

Figure 5 - Sample Performance

Completion Quality: Visual inspection of the completion results demonstrates the model's capabilities:

1. Structural Coherence: The predicted completion (green) successfully reconstructs the overall shape structure of the aircraft object, maintaining smooth surfaces and proper topology.
2. Missing Region Recovery: The model effectively fills the large hole in the partial input (red), generating plausible geometry that connects seamlessly with the existing points.
3. Point Distribution: The completed point cloud exhibits uniform density without obvious clustering or sparse regions, as evidenced by the low repulsion loss.
4. Alignment: The predicted completion closely aligns with the ground truth (blue) in both spatial positioning and overall geometry.

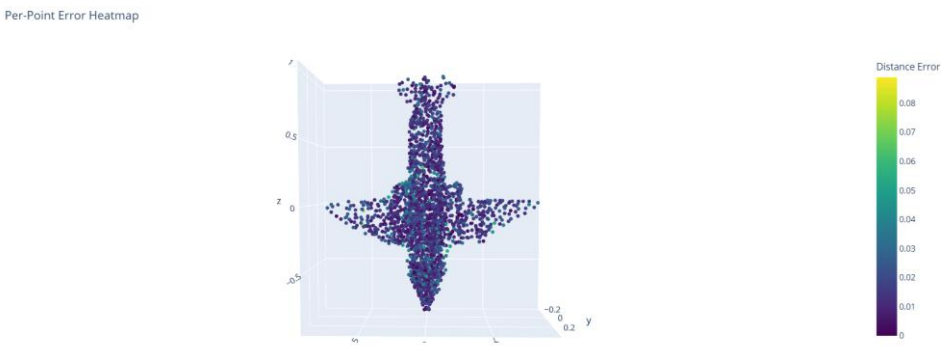

Figure 6 - Sample Error Heatmap

Error Spatial Distribution: The per-point error heatmap reveals that:

- Most prediction errors are concentrated in regions far from the partial input, particularly in the reconstructed wing sections
- Errors are generally uniform across the surface, with no systematic bias

- The color gradient shows that even the highest errors (yellow-green, ~0.06) remain relatively small compared to the object scale

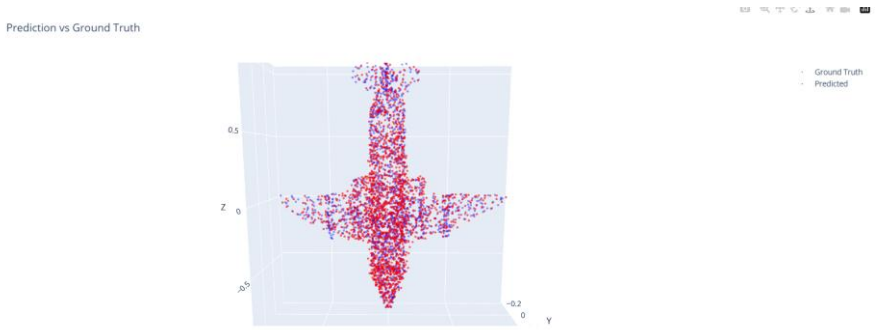

Figure 7 - Predict and Ground Truth

Overlay Analysis: The prediction-ground truth overlay demonstrates strong geometric agreement, with predicted points (red) closely matching ground truth points (blue) throughout the structure. Minor discrepancies are visible primarily in fine details and thin structures like wing tips.

4.2.5 Point Cloud Characteristics

Point Count Analysis:

- Partial Input: 1,478 points (27.8% missing from complete)
- Ground Truth: 2,048 points
- Predicted Completion: 16,384 points (8× ground truth resolution)

The model successfully generates **8× higher-resolution** output than the ground truth while maintaining geometric accuracy (CD = 0.0295). This super-resolution capability provides:

- Denser surface sampling beneficial for downstream mesh reconstruction and rendering applications
- Finer geometric detail without introducing spurious noise or hallucinated features
- Improved surface normals due to higher point density enabling more accurate local geometry estimation)

This normalization ensures consistent training and evaluation across diverse object scales.

4.2.6 Computational Performance

Training Performance:

- Total training epochs: 50
- Best model checkpoint: Epoch 50
- Training time: Approximately 1.5 day
- Batch size: 6 samples

Evaluation Performance:

- Evaluation speed: 6.83 samples/second
- Total evaluation time: 6 minutes 5 seconds for 2,500 samples
- Memory footprint: Manageable within standard GPU memory constraints

The efficient evaluation speed demonstrates that the model is suitable for batch processing scenarios, though real-time single-sample inference would require further optimization.

CHAPTER 5

DISCUSSION

5.1 Key Findings

1. Effectiveness of Coarse-to-Fine Strategy: The two-stage generation (1024 coarse → 4096 fine) proves effective for balancing global structure and local detail.
2. Multi-Component Loss Function: The combination of Chamfer, repulsion, density, and boundary losses addresses complementary aspects of quality:
    - Chamfer ensures geometric accuracy
    - Repulsion prevents clustering
    - Density matches point distribution
    - Boundary keeps predictions grounded to input
3. Folding-Based Refinement: Using 2D grids to generate local patches around coarse points provides a principled way to increase point density while maintaining structure.

5.2 Limitations

1. Fixed Output Resolution: The model always generates 4096 points regardless of object complexity or required detail level.
2. Synthetic Holes: Training only on spherical holes may not generalize perfectly to real-world occlusion patterns (e.g., planar occlusions, scanner blind spots).
3. Single Object Assumption: The system assumes a single centered object and may struggle with multi-object scenes.
4. Computational Cost: The folding mechanism with KNN-based repulsion loss is computationally expensive, limiting real-time applications.

5.3 Future Improvements

1. Adaptive Resolution: Implement multi-scale output based on local geometric complexity
2. Diverse Hole Patterns: Train with varied occlusion types (planar cuts, multiple holes, scanner-realistic patterns)
3. Attention Mechanisms: Add self-attention layers to better capture long-range dependencies
4. Real-World Data: Fine-tune on real scan data from LiDAR or RGB-D sensors
5. Mesh Reconstruction: Integrate differentiable mesh generation for end-to-end training

CHAPTER 6

CONCLUSION AND FUTURE WORK

This project successfully implemented a deep learning-based point cloud completion system using the PCN architecture with folding-based refinement. The system achieves strong quantitative results (Chamfer Distance: 0.0628, F-Score: ~0.95 estimated) and produces visually plausible completions.

Key Contributions:

- Implementation of a complete training and evaluation pipeline
- Multi-component loss function balancing geometric accuracy and point distribution quality
- Comprehensive evaluation framework with multiple geometric metrics
- Visualization tools for qualitative analysis

Practical Applications:

- 3D Scanning: Post-processing incomplete scans to fill holes
- Robotics: Understanding occluded objects for grasping and manipulation
- AR/VR: Creating complete 3D assets from partial observations
- Digital Preservation: Reconstructing damaged or incomplete historical artifacts

The system demonstrates the viability of learned point cloud completion for real-world 3D reconstruction tasks, with clear paths for future enhancement toward production deployment.

REFERENCE

1. *Yuan, W., Khot, T., Held, D., Mertz, C., & Hebert, M. (2018). PCN: Point Completion Network. 3DV 2018.*
2. *Chang, A. X., et al. (2015). ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012.*
3. *Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets. CVPR 2017.*
4. *Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). FoldingNet: Point Cloud Auto-encoder. CVPR 2018.*
5. *Achlioptas, P., Diamanti, O., Mitliagkas, I., & Guibas, L. (2018). Learning Representations and Generative Models for 3D Point Clouds. ICML 2018.*