DeFi: Detection and Filling of Holes in Point Clouds Towards Restoration of Digitized Cultural Heritage Models

Ramesh Ashok Tabib Dikshit Hegde Tejas Anvekar Uma Mudenagudi
Center of Excellence in Visual Intelligence (CEVI), KLE Technological University,
Vidyanagar, Hubballi, Karnataka, India-580031
ramesh_t@kletech.ac.in, dikshithegde@gmail.com, anvekartejas@gmail.com, uma@kletech.ac.in

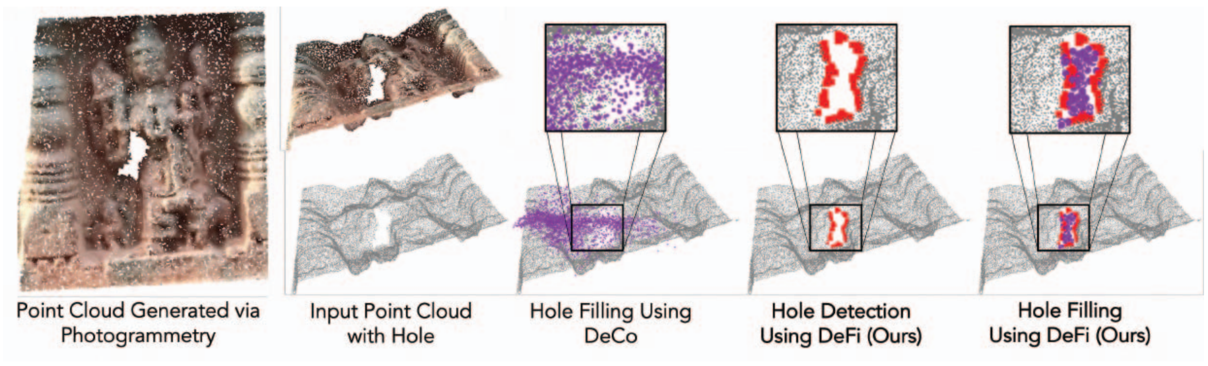

Point Cloud Generated via Photogrammetry
Input Point Cloud with Hole
Hole Filling Using DeCo
Hole Detection Using DeFi (Ours)
Hole Filling Using DeFi (Ours)

Figure 1: The figure highlights the potential of the proposed DeFi and shows the impact on hole detection and filling in the Heritage 3D models in comparison with the state-of-the-art technique DeCo. While DeCo struggles with hole filling due to its limited understanding of the underlying geometry and the precise location of hole boundaries. The proposed DeFi framework achieves superior hole filling due to prior knowledge of hole boundary and geometry, enabling it to achieve better results in specific areas (hole filling) of point cloud completion.

Abstract

In this paper, we propose DeFi: a novel perspective for hole detection and filling of a given deteriorated 3D point cloud towards digital preservation of cultural heritage sites. Preservation of heritage demands digitization as cultural heritage sites deteriorate due to natural calamities and human activities. Digital preservation promotes acquisition of 3D data using 3D sensor or Multi-view reconstruction. Unfortunately, 3D data acquisition finds challenges due to the limitations in sensor technology and inappropriate capture conditions, leading to formation of missing regions or holes in the acquired point cloud. To address this, we propose a pipeline consisting of detection of hole boundaries, and understanding the geometry of the hole boundaries to fill the region of the point cloud. Recent research on hole detection and filling fails to generalize on complex structures such as heritage sites, as they find challenges in differentiating between the hole boundary and non-hole boundary points. To address this, we propose to detect boundary points of point cloud and learn to classify them into "hole boundary" and "non-hole boundary" points. We generate a synthetic dataset based on ModelNet40 to learn the detection of hole boundaries. We demonstrate the results of the proposed pipeline on (i) ModelNet40 dataset, (ii) Heritage 3D models generated via photogrammetry, and compare the results with state-of-the-art methods.

1. Introduction

In this paper, we present a pipeline for detecting and filling holes in point clouds, with the aim of preserving and digitally presenting cultural heritage sites[13]. Cultural heritage sites are physical and intangible elements of a culture passed down through generations, and often deteriorate or get destroyed due to natural calamities and human activities. Preserving these sites is a major goal worldwide, and digital recreation using modern technology such as photographs, 3D scanners, and photogrammetry is one way to achieve this[5, 23, 26, 25]. The advancement of 3D data acquisition techniques and increased computational power

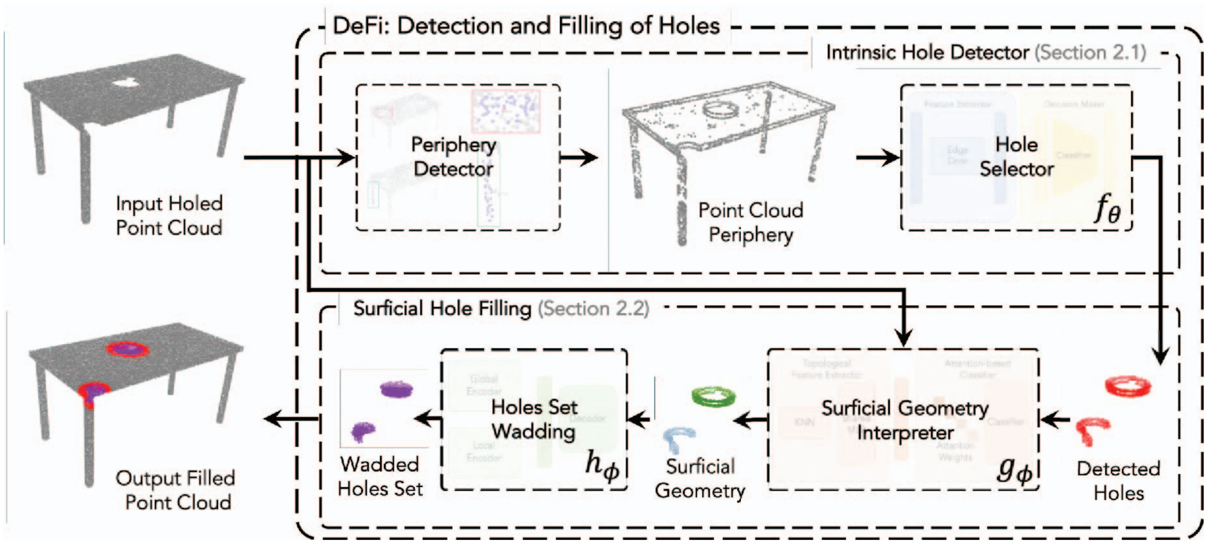


Figure 2: Illustration of proposed framework DeFi: a hole detection and filling of deteriorated point cloud.

have made it possible to digitally recreate, reconstruct, and render 3D models of cultural heritage sites[6, 9]. Many of these sites have been damaged or are in a state of deterioration due to natural weathering, disasters, and wars. The government has shown interest in recreation and restoration of these sites, which paves the way for 3D reconstruction and modeling. Several challenges exist in the 3D reconstruction pipeline, as presented by[6, 3, 29].

Photogrammetry is a technology that uses images and scans to accurately measure and build 3D models of real-world objects or scene. However, reconstructed models often have missing regions or holes, noise[10], low density regions[14, 17], due to various factors such as occlusion, physical properties of the scanned surface, repeated information[32, 25],missing pieces, limited accessibility, and external intervention[6, 9, 11, 12]. Filling these holes is important for better representation of 3D models as shown in Figure 1.

Recent literature has focused on various methods for filling holes in point clouds, including interpolation of new points[19], and generating missing parts[21, 20, 22]. However, these methods have their limitations. Interpolating new points through encoding and decoding the partial/holed point cloud as a latent representation leads to reconstructing the entire point cloud rather than just filling the missing parts, resulting in generic reconstructions that lack specific details[33]. Generating missing parts may be more likely to generate the missing parts but is not generalized to holed point clouds[1]. Interpolating points inside the boundary using higher order polynomial fit, triangulation, or surface fitting faces challenges depending on the size of the missing regions and requires perfect hole boundaries[18, 28]. Towards this, we propose a learning-based filling algorithm that takes into account the geometric information[2, 27, 8, 7] of the detected hole boundaries. Our method aims to generate the missing points in the regions using the specific properties of the input instance, rather than reconstructing the entire point cloud. By utilizing the detected hole boundaries, we can focus on local properties of each sample rather than just the global geometry, resulting in more accurate and specific reconstructions. Our method is expected to provide a better solution for filling holes in point clouds towards the digital preservation and presentation of cultural heritage sites.

Most existing non-learning-based hole boundary detection methods rely on manually setting threshold parameters, which is a tedious task[4]. To automate this process, authors in [24] automate this process by per point classification using PointNet[15] architecture. The model finds challenges to detect the hole boundaries as there is imbalance between hole boundary and non hole boundary. The proposed pipeline focuses on building a learning-based filling algorithm with the geometric information of the detected hole boundaries. The pipeline comprises two main stages: hole boundary detection and hole filling as shown in Figure 2.

Towards this, the main contributions of the paper are,

- We propose DeFi: a pipeline for detection and filling of holes towards improved presentation of point clouds.
  - We introduce a Periphery Detector to aid detection of hole boundaries for complex structures, where distinguishing between hole boundary and non-hole boundary points finds challenges.

- We propose Surficial Geometry Interpreter to derive geometric features (point cloud decomposition) of the point cloud towards assisting hole filling.

- We propose to generate synthetic dataset based on ModelNet40 to facilitate learning detection and filling of holes.
- We propose to demonstrate effectiveness of the proposed methodology on both ModelNet40[31] and Heritage 3D models.

In Section 2, we discuss the proposed pipeline for detection and filling of missing regions. We discuss the results and effect on filling of missing regions in Section 3 and conclude in Section 4.

2. DeFi: Hole Detection and Filling

We propose DeFi: a novel perspective for hole detection and filling of a given deteriorated 3D point cloud as shown in Figure 2. The DeFi incorporates mainly two modules: (i) Intrinsic Hole Detector, and (ii) Surficial Hole Filling. We define point cloud $P = \{p_1, p_2, ..., p_n\}$ where $p_i \in \mathbb{R}^3$ where we consider three input features x, y, z for processing.

2.1. Intrinsic Hole Detector (IHD)

The detection of hole boundaries in point cloud is subtle process, as the underlying surface of point cloud is undefined. Recent advancements in deep learning techniques detect boundaries of point cloud by leveraging local surface information[16, 30]. However, in hole detection problem the hole boundary points sets $\mathcal{H} \ll$ non-hole boundary points set $\mathcal{N}$, such that $\mathcal{H} \bigcup \mathcal{N} = P$ this demands a robust training strategy for hole boundary detection. Towards this, we propose Intrinsic Hole Detector (IHD), a two-stage approach for hole boundary detection in point clouds, and include (a) Periphery Detector: primarily to detect boundaries of the point cloud by mitigating the aforementioned challenge, (b) Hole Selector: to detect the hole boundaries from identified point cloud boundaries.

- **Periphery Detector**: Unstructured characteristic of the point cloud makes it challenging to understand the boundary points. One approach to detect point cloud boundaries is by finding the distance d between a query point q and centroid μ of local neighbourhood point given by K-NN graph[30], the distance d_i between query point and the centroid is calculated using the equation:

$$d_i = ||q_i - \mu_i||_2^2 \tag{1}$$

where $||\cdot||_2^2$ is euclidean distance.

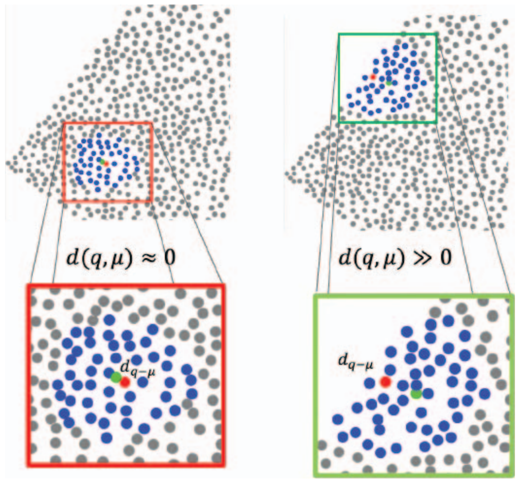


Figure 3: Illustrates the notion of proposed Periphery Detector where we consider magnitude of entropy between a query point q and centroid μ of local neighbourhood estimated via euclidean distance $d(q, \mu)$. The highlighted region with Red box depicts classification of query point as a point on manifold as $d(q, \mu) \approx 0$. In contrast the highlighted region with Green box depicts classification of query point as a point on periphery as $d(q, \mu) \gg 0$ facilitating detection and filling of deteriorated point cloud.

The classification of a query point into a boundary or non-boundary point depends on the distance between the point and the centroid point of its nearest neighbors using the condition:

$$b_i = \begin{cases} 1 & d_i \gg 0, \\ 0 & else. \end{cases} \tag{2}$$

where, b_i is classified output of Periphery Detector, 1 indicates boundary point and 0 indicates non-boundary point.

Algorithm 1: Point Cloud Periphery Detector

```
   Input: Point Cloud → $P$
   // $B, N_{in}, 3$
   Output: Boundaries of Point Cloud ← $b$
   // $B, N_{out}, 3$
1  Initialize $K$, $N_{in}$
2  for $i$ in $N_{in}$ do
3      $idx_{knn}$ = KNN($q_i$, $K$)
4      $\mu_i$ = mean( gather operation( $q_i$, $idx_{knn}$ ) )
5      $d_i$ = Calculate distance ( $q_i$, $idx_{knn}$)
6
7  $idx$ = $topK(d, N_{out}, idx)$
8  $b$ = gather operation($P$, $idx$)
```

In a similar manner, the distance d_i is estimated for all

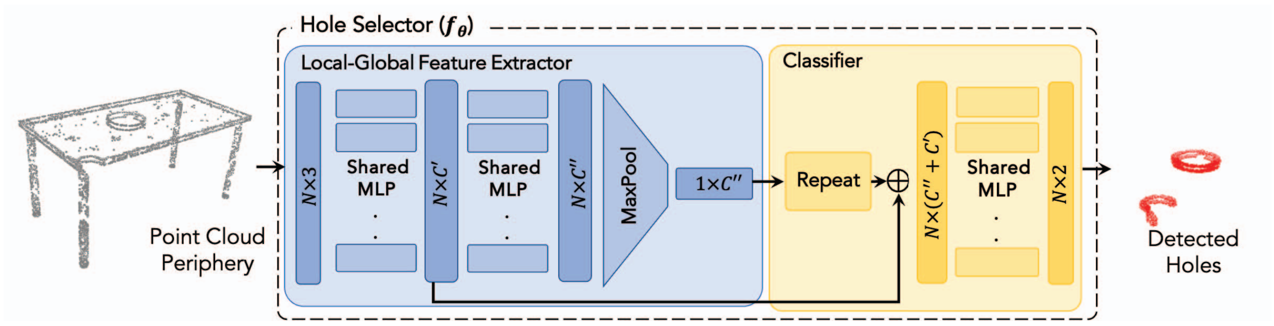


Figure 4: Proposed Hole Selector module, consumes peripheries of a point cloud to yield hole sets. We advocate to use a PointNet style backbone for performing a binary semantic segmentation task on Hole Boundary points vs Non-Hole Boundary.

points in the point cloud, and each point is classified as a boundary or non-boundary point using the Algorithm 1. The effect of K-nearest neighbour search in Periphery Detector is illustrated in Figure 3. Periphery Detector facilitates the overall performance of hole detection by significantly reducing the search space. We infer, detecting holes within boundary points is a simpler task when compared to detecting in entire point cloud.

- **Hole Selector**: The Hole Selector f_θ is a learning-based algorithm parameterized by weights θ that classifies each point in a point cloud into either a hole boundary or a non-hole boundary. To achieve this, each point is treated as an independent point and undergoes the same encoding module in the form of a Shared MLP[15]/Convolution[30]. A shared MLP is introduced to extract features from a point and use a symmetric function to extract the global permutation invariant features of the point cloud, facilitating to understand the overall structure of the point cloud. These global features are merged with local features, as shown in Figure 4, to provide a better understanding of the global context for each local feature. The extracted point cloud boundaries are then passed through the Hole Selector to differentiate between the hole boundary and non-hole boundary

Towards tuning of hole boundary detection by optimizing the weights of Hole Selector, we propose to use Cross Entropy Loss as a per point classification loss function given by,

$$\mathcal{L}_{classifier}(y_i, \hat{y}_i) = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} y_i \log \hat{y}_i \qquad (3)$$

where, b_i is the ground-truth information on hole boundary point or non hole boundary point, $\hat{b_i}$ is the predicted class of the particular point P_i from point cloud boundaries and N_{out} be the number of points in point cloud boundaries.

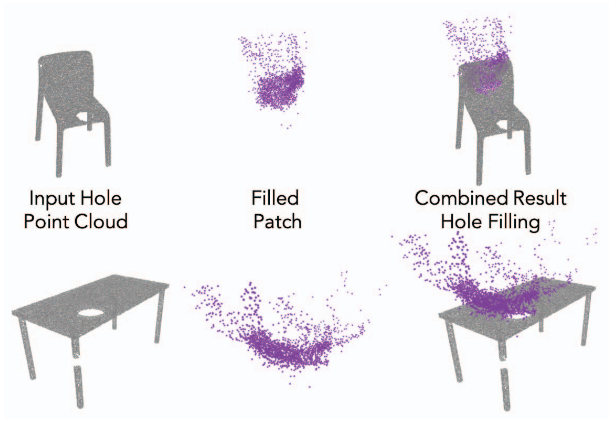


Figure 5: Results of hole filling using DeCo[1] without considering hole boundary information. We observe, the model fails to fill the hole due to unavailability of hole boundary information.

2.2. Surficial Hole Filling

Surficial Hole filling is a process of interpolating/generating $\mathcal{M}$ point with hole points prior $\mathcal{H}$ such that $\mathcal{M} \bigcup \mathcal{H}$ retains/extrapolates Surficial geometry of hole points. Learning-based extrapolation/completion method DeCo[1] fails to extrapolate/complete the points as shown in Figure 5. We infer, point cloud completion demands two types of information: (a) identifying the regions that needs completion, and (b) utilizing geometric information about the boundaries to extrapolate the points. However, current state-of-the-art methods fail to extrapolate the points based on the underlying geometry of the detected hole boundary. Towards this, we extract geometric features of the hole boundary points using the Surficial Geometry Interpreter (g_ϕ) and fill the holes using the Holes Set Wadding module (h_ϕ), as shown in Figure 2.

- **Surficial Geometry Interpreter**: We advocate to employ ABD-Net[7] as a Surficial geometry interpreter

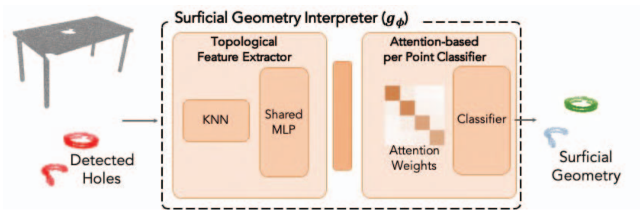

Surficial Geometry Interpreter (g_ϕ)
Topological Feature Extractor
KNN
Shared MLP
Attention-based per Point Classifier
Attention Weights
Classifier
Detected Holes
Surficial Geometry

Figure 6: Our proposed framework features a Surficial Geometry Interpreter module, which is designed to identify the primitive decomposition of hole boundary points within a point cloud. This is accomplished by leveraging a Topological feature extractor to gain a deeper understanding of the surface geometry, and an attention-based per-point classifier to map the Topological features to one of four primitive decompositions: planar, spherical, conical, or cylindrical.

g_ϕ to comprehend an object's geometry. The fundamental objective here is to discern the geometrical properties of a 3D point cloud through point-wise classification into one of four primitive shapes: planar, spherical, conical, or cylindrical. However comprehending the geometry of each point can be laborious, and demands understanding the geometry in conjunction with its neighborhood. To accomplish this, a K-Nearest Neighbor technique is applied for a given query point, and topological features are derived accordingly. An attention mechanism is subsequently employed to facilitate the architecture in easily distinguishing the geometry based on the extracted topological features, as illustrated in Figure 6. These features further aid in Surficial hole filling by extrapolating the topology of the hole boundary.

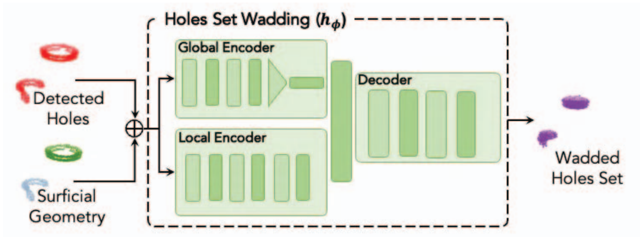


Figure 7: Proposed Holes Set Wadding module consumes Detected Holes and Surficial geometry priors of these holes towards filling of these holes. The h_ϕ module features a global encoder and a local neighborhood-based encoder, which work together to facilitate the decoder in gaining a better understanding of the geometry, proximity, and location of the holes that need to be filled. To perform hole filling, we leverage DeCo[1], which serves as our decoder and enables efficient and effective hole filling within the point cloud.

- **Holes Set Wadding**: Towards hole segment filling, it is crucial to understand the global and local structure based on the geometric signature. To achieve this, we model h_ϕ two parallel encoders: a local encoder and a global encoder, as shown in Figure 7. The local encoder is responsible for understanding the local topology, while the global encoder captures the global information. These two representations are then combined and processed through a decoder that extrapolates the points between the boundary to fill the holes.

Towards extracting geometric features of hole boundaries, we propose to train Surficial Geometry Interpreter using Eq.3. Towards filling of hole boundaries, we optimize the weights of Hole Set Wadding module by *Chamfer Distance* (C_{dis}) as a reconstruction loss. Chamfer distance is the measure of squared distance between each points of one point cloud with nearest point from another point cloud given by,

$$C_{dis}(G,\hat{Y}) = \sum_{g\in G} \min_{y\in\hat{Y}} ||g-y||_2^2 + \sum_{y\in\hat{Y}} \min_{g\in G} ||g-y||_2^2 \quad (4)$$

here, g is a point in ground-truth G, and y is a point in filled segment $\hat{Y}$ (generated through Hole Set Wadding module).

3. Results and Discussions

In this section, we provide a brief description on the dataset considered for experimentation, the experimental setup of the proposed methodology, and demonstrate the results of proposed methodology in comparison with state-of-the-art methods[24, 1].

3.1. Dataset

We synthetically introduce missing regions considering ModelNet40[31] as a base dataset to facilitate detection, and filling of holes. We employ Traceparts dataset to train Surficial Geometry Interpreter.

- **Synthetic Data Generation**: Due to unavailability of missing region dataset and the corresponding ground-truth, we synthetically generate the missing regions using ModelNet40[31] dataset. We consider a random query point (q) and apply K-Nearest Neighbour search on the point cloud to generate the missing regions. The r points nearer to the query point (q) are deleted to create a hole and the remaining ($K-r$) points are labeled as hole boundary. ModelNet40 dataset consists of CAD models of 40 categories. These CAD models are sampled to 4096 points to form a point cloud. We generate 40K samples including train and validation set.

- **Traceparts**[7]: dataset consists a total of 16157 mechanical component models, along with primitive

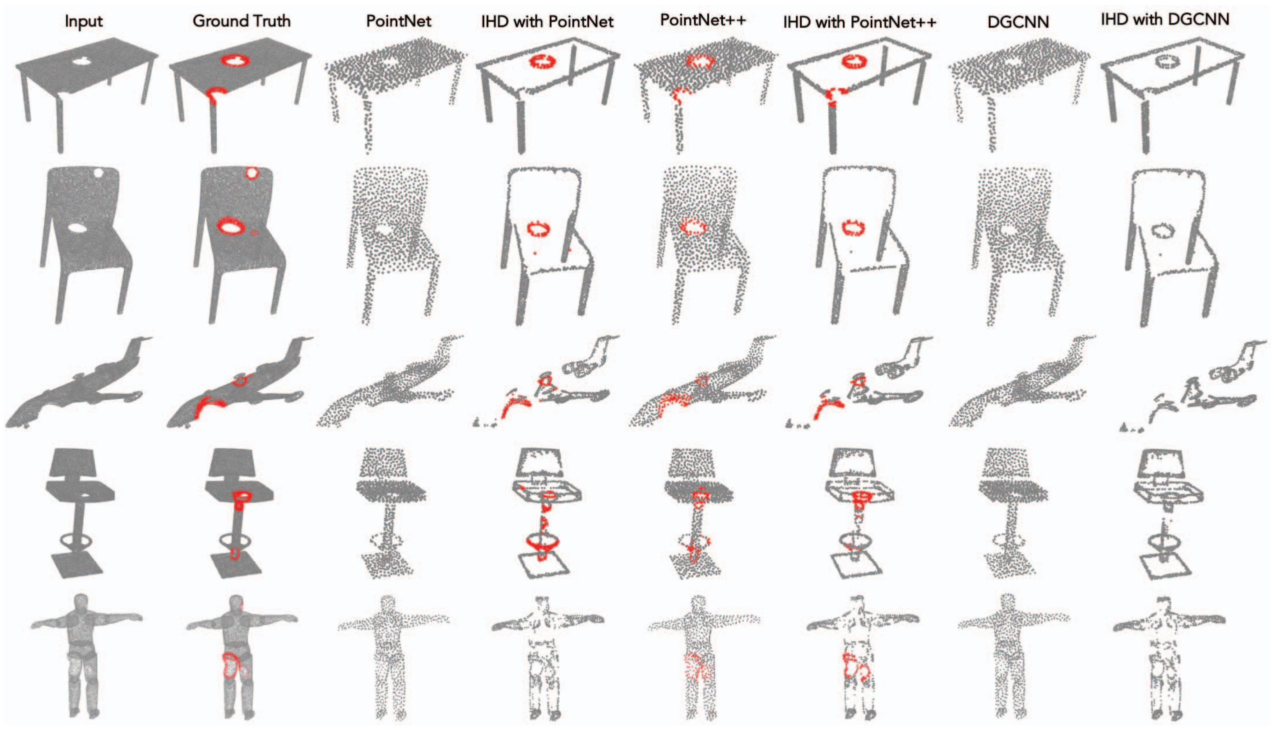


Figure 8: Visualization of hole detection results on ModelNet40 objects. First column are the considered input point cloud, Second column are the ground-truth point cloud, and Third to Eighth column are the hold detection results using PointNet[15], PointNet++[16], DGCNN[30] and their IHD variants.

shapes information labeled as planar, spherical, cylindrical, and conical. The dataset is divided into 12984 training samples and 3173 testing samples.

3.2. Experimental Setup

In this section, we discuss experimental setup used in DeFi, and provide a detailed description on Intrinsic Hole Detector and Surficial Hole Filling module used in DeFi.

- **Intrinsic Hole Detector**: Initial we extract the boundary points of the given point cloud using Periphery detector as shown in Algorithm 1. This helps to balance the ratio of the number of points belonging to missing region boundary points verses non missing region boundary points. We train Hole selector f_θ module using the extracted boundary points and classify the boundary points into missing region boundary points and non missing region boundary points. We use PointNet[15], PointNet++[16], and DGCNN[30] as a backbone architecture for Hole selector. We train Hole Selector f_θ with initial random weights θ for 100 epochs on synthetically generated data with 1024 points as a input, and with learning rate of $10e^{-3}$ using Adam optimizer.
- **Surficial Hole Filling**: uses the detected boundary points as clues and estimates the underlying geometry. We estimate the basic geometry of the detected hole boundaries through Surficial Geometry interpreter (g_ϕ) with ABD-Net[7] as the backbone. We train ABD-Net on Traceparts dataset for 50 epochs and classify the geometry of hole boundary points into four basic primitives namely Planar, Spherical, Conical, and Cylindrical. Classified geometric signatures of the hole boundary points are considered as a clue for completion/hole filling using Holes Set Wadding with DeCo architecture[1] as the backbone. We use the hyper-parameter settings similarly to the authors of DeCo and train the filling/completion architecture h_ϕ for 250 epochs.

3.3. Results

In this section, we demonstrate the results of proposed methodology DeFi on hole detection and filling of point cloud using synthetically generated dataset. Due to limited availability of literature on learning-based hole detection techniques, we compare the results with[24] and show the corresponding segmentation IoU scores in Table 1. Authors in[24] use PointNet[15] as backbone architecture for hole detection. Through experiments, we infer unbalanced distribution of hole and non-hole boundary points makes hole boundary detection a challenging task. As discussed in Section 2, Periphery detector helps us to mitigate the challenge by confining the search space.

Table 1: Comparison of hole boundary detection with PointNet, and PointNet++ with and without our Periphery Detector. The term **mIoU** denotes the mean Intersection over Union (IoU) of both hole and non-hole segments. In this context, the **Hole IoU** corresponds to the IoU calculated specifically for the hole segment. Higher the IoU, better the results. Higher values are represented in **Bold**. Topmost values are represented in " . " (blue).

	mIoU	**Hole IoU**
PointNet[15](2016)	0.4530	0.0172
Learning-based Hole Detection[24](2020)	0.4530	0.0172
IHD with PointNet(Ours)	**0.5555**	**0.3234**
PointNet++[16](2017)	0.8239	0.6899
IHD with PointNet++(Ours)	**0.8956**	**0.8361**

We visually represent the results of proposed methodology in Figure 8. We infer, the vanilla PointNet[15] fails to detect holes, as illustrated in Figure 8. In contrast, the Intrinsic Hole Detector (IHD) variant of PointNet is capable of detecting holes, and is further supported by our quantitative analysis presented in Table 1. We observe a significant improvement of **10%** in mIoU and **31%** in Hole IoU compared to vanilla PointNet on Hole Detection. We report a similar finding for PointNet++[16] and its IHD variant, where we achieve a **7%** improvement in mIoU and **14%** in Hole IoU, as depicted in Figure 8. This is due to the fact that IHD reduces the search space of hole detection by limiting the point cloud to just the periphery points.

Furthermore, we observe both vanilla DGCNN and its IHD variant fail significantly in detecting holes. This is because edge conv module in DGCNN may consider to build semantic relationship between two or more holes, leading to a drop in hole detection performance. Overall, our results demonstrate the IHD variant on existing state-of-the-art point cloud semantic segmentation algorithms performs better than their vanilla counterparts in detecting holes, indicating the effectiveness of our proposed approach.

Humans typically perceive the overall structure of an object in 3D by understanding or extracting the boundary points, and the proposed Periphery Detector performs similar to human perception. The Periphery Detector algorithm can facilitate the understanding of the overall structure/boundary of an object in 3D, as demonstrated in Figure 9. Additionally, this algorithm opens the door for many applications, such as classification of point clouds and understanding key points through heat waves and kernels. In this paper, we utilize this algorithm to extract all the boundaries from the point cloud, which aids in differentiating between hole boundaries and non-hole boundaries.

We report the performance of our proposed surficial hole filling module with and without surficial geometry interpreter in Figure 10. One can conclude that our proposed Surficial geometry interpreter is capable of understanding the underlying geometry which indeed facilitates Hole filling unlike vanilla DeCo[1] which fail comprehensively to extrapolate the hole boundary points. We also infer that our proposed method yields noisy hole filling output may be due to the fact that our methodology is not an end-to-end trained model on a specific task. Although our study has some limitations, we believe that it provides valuable insights into the hole filling process in point clouds. We anticipate that our results will stimulate further investigation to overcome these limitations and promote the creation of more reliable and efficient approaches for filling holes in point clouds.

We demonstrate the detection and filling of hole through our proposed methodology on heritage 3D models in Figure 1 and Figure 11. We observe, there is discontinutiy

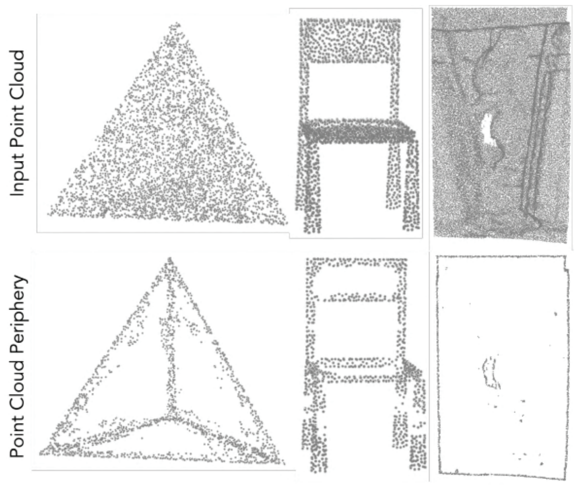
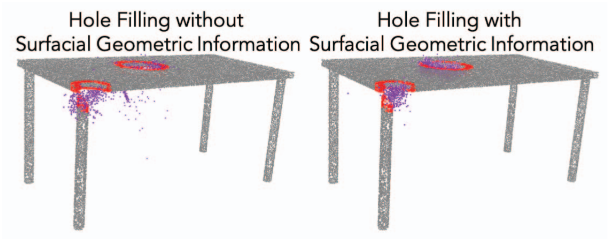


Figure 9: Visualization of Periphery Detector on exemplars such as tetrahedron, chair, and a Heritage 3D model. We infer, detected boundaries of the point clouds facilitate the deep learning algorithms to analyse the object. First row shows the input to the Periphery Detector and second row shows the extracted boundaries through Periphery Detector.


Hole Filling without Surfacial Geometric Information
Hole Filling with Surfacial Geometric Information
Figure 10: We visualize the result of hole filling with and without proposed Surficial Geometric Interpreter. Right figure show supremacy in hole filling due incorporation of geometry priors unlike the one without in Left.

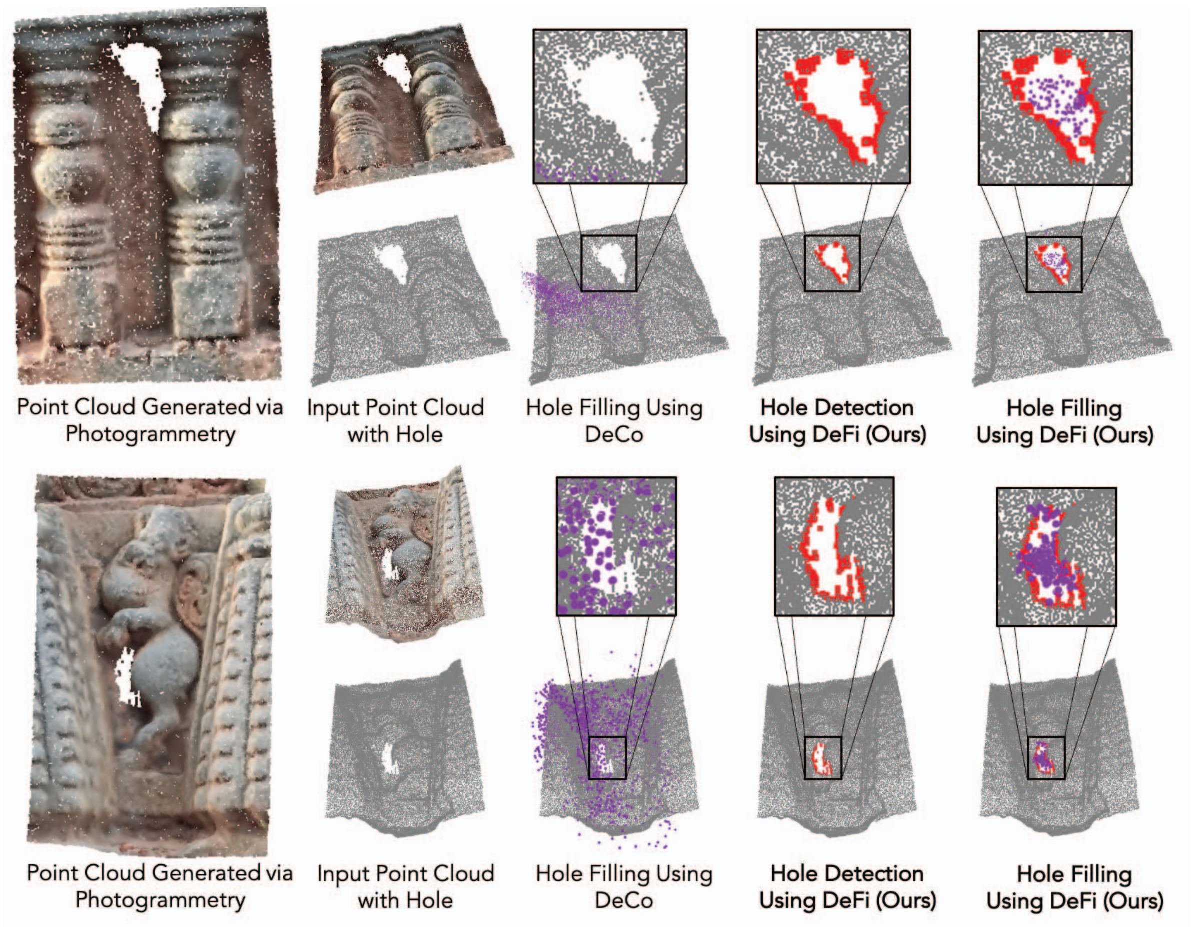

Point Cloud Generated via Photogrammetry
Input Point Cloud with Hole
Hole Filling Using DeCo
Hole Detection Using DeFi (Ours)
Hole Filling Using DeFi (Ours)
Point Cloud Generated via Photogrammetry
Input Point Cloud with Hole
Hole Filling Using DeCo
Hole Detection Using DeFi (Ours)
Hole Filling Using DeFi (Ours)

Figure 11: Visual comparison of our proposed methodology DeFi for detection and filling of holes on Heritage data with DeCo[1]. Red points are hole boundary detected points and Purple points are the filled/generated points.

in the detected hole boundaries due to the complex structures/curvature of the point cloud. We observe the filled points do not deviate from the hole region when compared with Vanilla DeCo. The complex structure/curvature of the hole boundary, makes the surface hole filling uneven resulting in cluster of points in a particular area within a hole region as shown in Figure 11.

4. Conclusions

In this paper, we have proposed a pipeline for the detection and filling of holes in point clouds to aid in the digital preservation and presentation of cultural heritage sites. We have proposed a method for detecting hole boundaries by balancing the uneven distribution of point clouds, and have shown that our proposed approach achieves significant improvements in IoU on hole boundary region detection when trained on PointNet, PointNet++, and DGCNN. Additionally, we have demonstrated that our proposed ABD-Net approach is effective in filling holes by leveraging geometric signatures and achieving a 0.2 decrease in Chamfer distance when trained on Deco. We have demonstrated the results of proposed methodology on 3D acquired heritage point clouds, highlighting the potential of our proposed pipeline for digital preservation and presentation of cultural heritage sites.

5. Acknowledgement

This work is partly carried out under the AICTE- RPS project “Shape Representation, Reconstruction and Rendering of 3D Models” (8-247/ RIFD/ RPS (POLICY-r)/2OLB-19), Department of Science and Technology (DST) through the ICPS programme - Indian Heritage in Digital Space for the project “Crowd-Sourcing” (DST/ ICPS/ IHDS/ 2018 (General)), and Department of Science and Technology (DST) through ICPS programme - Indian Heritage in Digital Space for the project “Digital Poompuhar” (DST/ ICPS/ Digital Poompuhar/ 2017 (General)).

References

[1] Antonio Alliegro, Diego Valsesia, Giulia Fracastoro, Enrico Magli, and Tatiana Tommasi. Denoise and Contrast for Category Agnostic Shape Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4629–4638, June 2021.

[2] Tejas Anvekar, Ramesh Ashok Tabib, Dikshit Hegde, and Uma Mudengudi. VG-VAE: A Venatus Geometry Point-Cloud Variational Auto-Encoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2978–2985, 2022.

[3] Fausto Bernardini and Holly Rushmeier. The 3D Model Acquisition Pipeline. *Comput. Graph. Forum*, 21:149–172, 06 2002.

[4] Pavel Chalmoviansky and Bert Jüttler. Filling Holes in Point Clouds. pages 196–212, 01 2003.

[5] Dikshit Hegde, Tejas Anvekar, Ramesh Ashok Tabib, and Uma Mudengudi. DA-AE: Disparity-Alleviation Auto-Encoder Towards Categorization of Heritage Images for Aggrandized 3D Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5093–5100, June 2022.

[6] Katsushi Ikeuchi, Takeshi Oishi, Jun Takamatsu, Ryusuke Sagawa, Atsushi Nakazawa, Ryo Kurazume, Ko Nishino, Mawo Kamakura, and Yasuhide Okamoto. ”the great buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects”. *International Journal of Computer Vision*, 75(1):189–208, Oct. 2007. Funding Information: This research is sponsored, in part, by JST under Ikeuchi Crest program, and, in part, by Ministry of Education under Leading Project. The Bayon temple in Cambodia was digitized with the cooperation of the Japanese Government Team for Safeguarding Angkor (JSA).

[7] Siddharth Katageri, Shashidhar V Kudari, Akshaykumar Gunari, Ramesh Ashok Tabib, and Uma Mudenagudi. ABD-Net: Attention Based Decomposition Network for 3D Point Cloud Decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2049–2057, October 2021.

[8] Siddharth Katageri, Sameer Kulmi, Ramesh Ashok Tabib, and Uma Mudenagudi. PointDCCNet: 3D Object Categorization Network Using Point Cloud Decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2200–2208, June 2021.

[9] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, Duane Fulk, and Cyberware Inc. The Digital Michelangelo Project: 3D Scanning of Large Statues. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 1, 03 2001.

[10] Shitong Luo and Wei Hu. Differentiable manifold reconstruction for point cloud denoising. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1330–1338, 2020.

[11] Anupama Mallik, Santanu Chaudhury, Vijay Chandru, and Sharada Srinivasan. *Digital Hampi: preserving Indian cultural heritage*. Springer, 2017.

[12] Helen C Miles, Andrew T Wilson, Frédéric Labrosse, Bernard Tiddeman, Seren Griffiths, Ben Edwards, Katharina Moller, Raimund Karl, and Jonathan C Roberts. Crowd-sourced digitisation of cultural heritage assets. In *2014 International Conference on Cyberworlds*, pages 361–368. IEEE, 2014.

[13] Uma Mudenagudi, Syed Altaf Ganihar, Shreyas Joshi, Shankar Setty, G Rahul, Somashekhar Dhotrad, Meera Natampally, and Prem Kalra. Realistic Walkthrough of Cultural Heritage Sites-Hampi. In *Asian Conference on Computer Vision*, pages 554–566. Springer, 2014.

[14] Shanthika Naik, Uma Mudenagudi, Ramesh Tabib, and Adarsh Jamadandi. Featurenet: Upsampling of point cloud and it's associated features. In *SIGGRAPH Asia 2020 Posters*, pages 1–2. 2020.

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[16] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *CoRR*, abs/1706.02413, 2017.

[17] T Santoshkumar, Deepti Hegde, Ramesh Ashok Tabib, and Uma Mudenagudi. Refining sfm reconstructed models of indian heritage sites. In *SIGGRAPH Asia 2020 Posters*, pages 1–2. 2020.

[18] Shankar Setty, Syed Altaf Ganihar, and Uma Mudenagudi. Framework for 3D object hole filling,. In *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*.

[19] Shankar Setty, Syed Altaf Ganihar, and Uma Mudenagudi. Framework for 3D object hole filling. In *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4. IEEE, 2015.

[20] Shankar Setty and Uma Mudenagudi. Example-based 3d inpainting of point clouds using metric tensor and christoffel symbols. *Machine Vision and Applications*, 29:329–343, 2018.

[21] Shankar Setty and Uma Mudenagudi. Region of interest-based 3D inpainting of cultural heritage artifacts. *Journal on Computing and Cultural Heritage (JOCCH)*, 11(2):1–21, 2018.

[22] Shankar Setty, Himanshu Shekhar, and Uma Mudenagudi. Region of interest (roi) based 3d inpainting. In *SIGGRAPH ASIA 2016 Posters*, pages 1–2. 2016.

[23] Ramesh Ashok Tabib, Dikshit D Hegde, T Santoshkumar, Srikar H I, Mutturaj Harage, Chaitra Desia, Ujwala Patil, and Uma Mudenagudi. Deep Features for Categorization of Heritage Images Towards 3D Reconstruction. *Procedia Computer Science*, 171:483–490, 2020. Third International Conference on Computing and Network Communications (CoCoNet'19).

[24] Ramesh Ashok Tabib, Yashaswini V. Jadhav, Swathi Tegginkeri, Kiran Gani, Chaitra Desai, Ujwala Patil, and

Uma Mudenagudi. Learning-Based Hole Detection in 3D Point Cloud Towards Hole Filling. *Procedia Computer Science*, 171:475–482, 2020. Third International Conference on Computing and Network Communications (CoCoNet'19).

[25] Ramesh Ashok Tabib, Sujaykumar Kulkarni, Abhay Kagalkar, Vaishnavi Hurakadli, Abhijeet Ganapule, Rohan Raju Dhanakshirur, and Uma Mudenagudi. Deep learning-based filtering of images for 3d reconstruction of heritage sites. *Digital Techniques for Heritage Presentation and Preservation*, pages 147–156, 2021.

[26] Ramesh Ashok Tabib, T Santoshkumar, Varad Pradhu, Ujwala Patil, and Uma Mudenagudi. Categorization and selection of crowdsourced images towards 3d reconstruction of heritage sites. *Digital Techniques for Heritage Presentation and Preservation*, pages 133–146, 2021.

[27] Ramesh Ashok Tabib, Nitishkumar Upasi, Tejas Anvekar, Dikshit Hegde, and Uma Mudenagudi. IPD-Net: SO (3) Invariant Primitive Decompositional Network for 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2735–2743, 2023.

[28] Vishwanath S Teggihalli, Ramesh Ashok Tabib, Adarsh Jamadandi, and Uma Mudenagudi. A polynomial surface fit algorithm for filling holes in point cloud data. In *Pattern Recognition and Machine Intelligence: 8th International Conference, PReMI 2019, Tezpur, India, December 17-20, 2019, Proceedings, Part I*, pages 515–522. Springer, 2019.

[29] Alexandre Vrubel, Olga Bellon, and Luciano Silva. A 3D reconstruction pipeline for digital preservation. pages 2687 – 2694, 07 2009.

[30] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[31] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets for 2.5D Object Recognition and Next-Best-View Prediction. *CoRR*, abs/1406.5670, 2014.

[32] Smita C Yadavannavar, Varad Vinod Prabhu, Ramesh Ashok Tabib, Ujwala Patil, and Uma Mudengudi. Evidence Based Image Selection for 3D Reconstruction. In *Computer Vision, Pattern Recognition, Image Processing, and Graphics: 7th National Conference, NCVPRIPG 2019, Hubballi, India, December 22–24, 2019, Revised Selected Papers 7*, pages 53–63. Springer, 2020.

[33] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point Completion Network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737, 2018.