[PG07] Reconstruction of a 3D environment with Kinect

Matteo Zaramella[1,2]

Abstract

This project focuses on implementing data acquisition and matching using a Kinect depth camera, capable of providing RGB-D images. Multiple acquisitions are performed to capture the environment from different viewpoints, and these acquisitions are merged using Iterative Closest Point (ICP) to obtain a coherent 3D representation.

To improve the quality of the reconstructed environment, two denoising algorithms are developed: a 3D guided filter [1] and a denoising bilateral filter [2] adaptation for point clouds. The 3D guided filter leverages a guidance image to reduce noise while preserving essential geometric features. The denoising bilateral filter reduces noise in an image while preserving its edges and fine details.

The project aims to evaluate the performance of both denoising approaches in terms of noise reduction and preservation of fine details in the reconstructed 3D environment. By combining data acquisition, point cloud matching, and denoising techniques, this research seeks to enhance the accuracy and quality of 3D environment reconstruction for various applications, such as virtual reality, augmented reality, and robotics.

Keywords

LaTeX class, paper template, paper formatting, CEUR-WS

1. HIGHLIGHTS

5 bullet point to describe my work

- Acquisition of the images with Kinect to reconstruct 3d environment. Usage of Kinect and Kinect Studio;
- Representation and merging of the point clouds using Open3D library and ICP algorithm;
- Python implementation of 3D guided filter to denoise ;
- Python implementation of denoising bilateral filter adapted to point clouds slightly modified;
- Evaluation of the proposed methods with practical examples.

2. Introduction

The reconstruction of 3D environments from acquired sensor data is a challenging and crucial task in the fields of computer vision and robotics. This process involves several intricate steps, including data acquisition, point cloud matching, and denoising, which collectively form the foundation for creating reliable 3D representations. The accuracy of the final 3D representation heavily depend on how well these steps are executed. Noise is an inevitable factor that affects data acquired from sensors, making denoising an essential aspect of the reconstruction process.

This project delves into the realm of 3D environment reconstruction, aiming to enhance the accuracy of the reconstructed scenes through advanced denoising techniques. The primary objectives of this project are to implement data acquisition and matching techniques, then, are developed two denoising algorithms, namely a 3D guided filter [1] and a denoising bilateral filter [2], and evaluate their effectiveness in improving the reconstructed 3D environments.

For data acquisition, a Kinect version 2 RGB-D camera was employed due to its ability to provide RGB-D images, which contain both color and depth information. Here is utilized Kinect version 2, also known as Kinect for Xbox One, because it is an advanced version of its predecessor, Kinect version 1 (for Xbox 360), and was designed to provide more accurate and sophisticated interaction experiences. More in specific, the Kinect version 2 RGB-D camera, is a depth-sensing device developed by Microsoft [3], which has gained popularity due to its capability to provide RGB-D images. The key components of this Kinect are: the RGB camera which captures traditional color images and the depth sensor. The RGB camera consists of a high-resolution sensor that records color information from the visible spectrum, this allows the camera to capture standard color images in real-time. The depth sensor instead, operates based on the principle of Time-of-Flight (ToF). In this method, the Kinect emits short pulses of infrared light and measures the time it takes for the light to bounce back to the camera after hitting objects in the scene. Based on the time-of-flight measurement, the camera can calculate the depth information. The RGB and depth data captured by the Kinect camera, using Kinect Studio, can be combined to create a unified RGB-D image, where each pixel is associated with a color value and a corresponding depth value. Utilizing

Woodstock'22: Symposium on the irreproducible science, June 07–11, 2022, Woodstock, NY

zaramella.2025806@studenti.uniroma1.it (M. Zaramella)

this sensor, multiple acquisitions were captured to create a more comprehensive and detailed representation of the environment. To align these acquired point clouds and merge them coherently, the Iterative Closest Point (ICP) algorithm was utilized, which is a widely-used technique for point cloud registration. It aims to find the best rigid transformation (translation and rotation) that aligns one point cloud with another. ICP is an iterative process, and the transformation is refined in each iteration until a satisfactory alignment is achieved. The quality of the alignment depends on the quality of the initial guess, the presence of noise and outliers in the data, and the number of iterations performed.

After the execution of the data acquisition and the point cloud matching parts, the point cloud is passed to the denoising algorithms. The first denoising approach explored in this project is the 3D guided filter [1]. This filter leverages guidance information from the input data to smooth and refine the noisy 3D point clouds effectively. The key advantage of the 3D guided filter denoising approach is its ability to achieve significant noise reduction while retaining important details and edges in the point cloud. It is particularly useful when dealing with noisy depth data from 3D sensors like in this case.

The second denoising approach investigated is a denoising bilateral filter. This approach In applies a Gaussian function that gives more weight to nearby pixels and less weight to pixels that are farther away. This means that pixels close to each other in space contribute more to the resulting pixel value.

Throughout the course of this project, the performance of these denoising algorithms will be thoroughly assessed and compared. Their respective strengths and weaknesses will be analyzed, considering factors such as denoising quality, computational efficiency, and robustness to varying noise levels and environmental conditions.

The outcomes of this research hold great potential for various applications, including robotics, augmented reality, and virtual reality, where accurate 3D environment representation is critical for seamless interaction and immersive experiences. Moreover, the insights gained from this project contribute to the broader field of computer vision and signal processing, where denoising techniques play a vital role in enhancing the quality of acquired data.

In the following sections, we will delve into the methodology used for data acquisition, point cloud matching, and the implementation of the denoising algorithms. The results part will contain the evaluation metrics, followed by a comprehensive analysis of the experiments done and the values obtained. Finally, the conclusion will summarize the findings and discuss potential future directions for further improvement and research in this domain.

3. Related Works

There are many works related to point cloud denoising, first of all the Bilateral filtering-based methods are employed for image denoising and smoothing while preserving edges. Initially introduced by Tomasi et al. [4], bilateral filtering is a non-linear technique that considers both spatial position and pixel value of neighboring elements. Digne et al. [2] extended bilateral filtering on meshes, initially introduced by Fleishman et al. [5], by considering spatial and normal distances. In addition to point position and normal, Zhang et al. [6] incorporated point color and designed a bilateral filter for denoising point clouds. Another type of denoising method is The guided filter, proposed by He et al. [7], it is an explicit image filter that acts as an edge-preserving smoothing operator. It assumes a local linear model between the guidance image and the filter output. Zheng et al. [8] extended the guidance normal filter for mesh normal smoothing [9] to point clouds using a multi-normal strategy. They estimated multi-normals for each feature point by partitioning its k-nearest neighbors (k-NN) into smooth patches, with each patch corresponding to one normal. They then applied the guided filter to the normal field to obtain a piece-wise smooth result. Similarly, Liu et al. [10] adopted the multi-normal strategy and presented a feature-preserving framework for noise-free point cloud recovery. They used an anisotropic second-order regularization method to restore the point normal field from the noisy input.

Moving on from Guided filter there are Graph-based point cloud denoising methods that treat the point cloud as a graph signal and utilize graph filters for denoising. Each point is considered a node, connected through edges to its k-NN neighbors. Duan et al. [11] constructed a k-NN graph based on Euclidean distance to capture local and global geometric structures. They proposed a weighted multi-projection (WMP) denoising algorithm, where each point was projected to its neighbors' tangent planes, and the denoised point was obtained by averaging the multiple projections. Irfan et al. [12] exploited the correlation between geometry and color attributes and generated a k-NN graph based on both color similarity and geometry proximity. They used graph-based convex optimization to denoise the point cloud. Another interesting work is the Moving Robust Principal Components Analysis (MRPCA) by Mattei et al. [13], which preserves sharp features by using weighted minimization of point deviations from the local reference plane. They modeled the point cloud as a collection of overlapping 2D subspaces and pooled independent estimates for a single point to yield a collaborative estimate. Leal et al. [14] proposed a similar approach, combining l1-median filtering with sparse l1 regularization for reconstructing and smoothing point clouds. They utilized sparsity in both

data fitting and the prior term and performed normal denoising and point position updating iteratively.

In the end, motivated by the remarkable achievements of deep learning, novel deep-learning-based approaches have emerged for point cloud denoising. These methods learn mappings from noisy inputs to their corresponding ground-truth representations during an offline training stage. Subsequently, during runtime, they can be automatically applied to new instances exhibiting similar geometric and noise characteristics as the models used for training. One pioneering method, PointNet [15], directly processes point clouds by independently learning features for each point through shared Multi-Layer Perceptrons (MLPs). To capture local structural information between points, Qi et al. [16] introduced PointNet++, a hierarchical network that captures fine geometric structures from each point's neighborhood. Recent advancements in point cloud denoising have been based on PointNet, owing to its simplicity and powerful representation capabilities. Inspired by PointNet, Guerrero et al. [17] devised a local variant called PCPNet, which estimates local 3D shape properties in point clouds. PCPNet exhibits superior results for shape details and is also applicable to denoise point clouds. The PCPNet utilizes a set of k nonlinear functions within local patch neighborhoods, yielding a k-dimensional feature vector per patch to regress various local features. Building upon the denoising architecture of PCPNet, Rakotosaona et al. [18] introduced PointCleanNet, a two-stage data-driven denoising architecture featuring a local outlier detection network and a denoising network. The local outlier detection network employs a PCPNet-based architecture to detect and remove outliers. Riccardo et al. [19] presented PointProNet, the first deep learning method for local point cloud processing with a fully differentiable, CNN-based architecture. Similar to [20], their approach utilizes local patches, where each patch of geometry around a point is represented as an oriented 2D heightmap that stores distances to sample points in the neighborhood along a given direction. The main objective is to learn a local mapping that accurately and densely samples the underlying surface in the consolidated version of the extracted points from the local patch. Similarly, Lu et al. [21] developed a CNN-based feature-preserving normal estimation framework based on 2D heightmaps for point cloud denoising. During training, they represent each point and its neighbors as a 2D heightmap through a simple projection approach based on PCA. They subsequently classify points into feature and non-feature points using a classification network. Pistilli et al. [22] introduced GPDNet, a Graph-convolutional Point Denoising Network, which denoises point clouds based on graph-convolutional layers. Graph convolution is a generalization of convolution applicable to data defined over the nodes of a general graph rather than a grid. A part from the presented deeplearning methods, the Pointfilter [23], proposed by Zhang et al., adopts a typical encoder-decoder architecture for point cloud denoising. Raw neighboring points of each noisy point serve as input, and the network regresses a displacement vector to restore the noisy point to its ground truth position. PCA is utilized for alignment, and the aligned patch is fed into the neural network. The encoder comprises feature extractors and a collector, with PointNet [15] serving as the backbone in the feature extractors. In the realm of image denoising, methods such as Noise2Noise [24], Noise2Self [25], and Noise2Void [26] have demonstrated unsupervised denoising capabilities without requiring clean data. However, only a limited number of related works have been reported for point cloud denoising in the existing literature.

4. System Design

The system comprises three distinct components, namely: data acquisition (pertaining to the Kinect module), point cloud registration (for the purpose of integrating diverse acquisitions), and denoising algorithms (employed to attenuate noise in the acquired point cloud).

4.1. Data acquisition

The data acquisition phase assumes paramount importance as it facilitates the acquisition of data with minimal noise interference. Therefore, ensuring a tranquil and well-illuminated environment, alongside appropriately positioning and halting the depth camera, is of utmost significance.

Following the identification of an optimal location and positioning, the data capture process employing the Kinect depth camera was executed using the Kinect Studio software. Developed by Microsoft [3], Kinect Studio offers an intuitive interface for the operation of the Kinect depth camera. It grants the capability to record both RGB and depth images, providing a real-time preview of the captured data.

4.2. Point Cloud Matching

For the point cloud matching task is used the Iterative Closest Point algorithm (ICP), from the Open3D library [27]. It is a widely used algorithm for aligning two or more point clouds in 3D space to find the best transformation that minimizes the distance between corresponding points in the clouds. It is commonly used for point cloud registration, object recognition, environment reconstruction and other computer vision tasks. The primary objective of ICP is to determine the optimal rigid transformation involving translation and rotation that aligns a given source point cloud to a target point cloud.

The algorithm iteratively refines this transformation until convergence is achieved.

ICP takes as input a source point cloud (Q), representing the points to be aligned, a target point cloud (P), which serves as the reference to which the source must be aligned, and initial guess for the transformation T). Subsequently, it seeks to establish corresponding points between the source and target point clouds. For efficiency, the data is organized in KD-trees (using the Open3d library), enabling ICP to efficiently find the nearest neighbors in the target point cloud for each point in the source cloud.

Once correspondences (K) is established, ICP aims to minimize the error between corresponding points by utilizing the Euclidean distance as the error metric. The primary objective is to minimize the sum of squared distances between the corresponding points:

$$\mathrm{E(T)} = \sum_{\mathrm{(p,q)} \in K} \|\mathrm{p} - \mathrm{Tq}\|^2$$

Using the estimated correspondence, ICP computes a new optimal transformation that minimizes the error between the two point clouds. This new transformation is applied to the source point cloud, and the process is iteratively repeated until convergence is reached. Convergence is considered to be achieved either when the change in the transformation parameters falls below a predefined threshold of 50 or when the maximum number of iterations (200) is reached.

4.3. Denoising algorithms

Guided 3D filter The 3D filter introduced herein represents an adaptation of the 3D Guided filter initially proposed by Han et al. [1]. In contrast to the previous work by Han et al., which primarily focused on denoising individual objects or specific geometrical shapes, this current undertaking employs the filter to attenuate noise across an entire environmental context.

The proposed approach is characterized by its straightforward and rudimentary implementation, serving as a foundational framework to enhance comprehension of the ensuing outcomes. The algorithm, as delineated in the pseudocode 1, commences by receiving two inputs: a point cloud denoted as P that necessitates denoising, and a parameter denoted as e representing the filtering effect. Subsequently, for each point within the point cloud, the algorithm calculates the adjacent neighbors, computes the mean point, and derives two integral parameters, a_i and b_i. The parameter a_i is computed by averaging the squared magnitudes of the neighboring points and subtracting the squared magnitude of the mean point. This value is then divided by the sum of itself and the filtering effect parameter e. Conversely, the second parameter, b_i, is determined by subtracting the product of the mean point and the previously calculated a_i from the mean point itself. Both a_i and b_i contribute to the transformation of the current noisy point p_i into a new denoised point $P_i^{'}$.

Algorithm 1 3D Guided filter

Require: P = point cloud to denoise with N point
e = filter effect parameter
for $i \in N$ **do**
$\{p_{ij}\}$ = neighbors of p_i
$k = |\{p_{ij}\}|$
$\hat{p}_i = \frac{1}{k} \sum_j p_{ij}$
$a_i = \frac{(\frac{1}{k} \sum_j p_{ij} * p_{ij} - * \hat{p}_i * \hat{p}_i)}{(\frac{1}{k} \sum_j p_{ij} * p_{ij} - * \hat{p}_i * \hat{p}_i) + e}$
$b_i = \hat{p}_i - \hat{p}_i * a_i$
$p_i^{'} = a_i * p_i + b_i$
end for

Denoising Bilateral filter I have adapted the renowned Bilateral filter, originally introduced by Fleishman et al. [5], for deployment as a denoising filter in the context of environment reconstruction. This filter relies on two pivotal parameters: σ_c and σ_s, where σ_c represents the radius of the neighborhood surrounding vertex v_i, equivalent to the distance between v_i and its farthest neighbor. Conversely, σ_s denotes the standard deviation of distances between a given vertex and its adjacent vertices. The primary modification, respect to the original one, involves the option to either provide these values as pre-defined parameters (to speed up the process) or compute them based on the input point cloud. More in specific, σ_c is calculated with:

$$\sigma_c = \max_{n \in \text{neighbors}} \left(\min_{p \in \text{tri.simplices}[n]} (\|v - \text{points}[p]\|) \right)$$

Instead the standard deviation σ_s is calculated as:

$$\sigma_s = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (t_i - \text{average\_offset})^2}$$

Where n is the number of neighbors and t_i are the individual offset values.

The filter takes as input a 3D point cloud $\mathcal{M}$ with vertices V and faces F, wherein each vertex $v_i \in V$ possesses both a position $\mathbf{p}_i$ and a surface normal $\mathbf{n}_i$. Additionally, the number of iterations for applying the filter, and optionally, the parameters σ_c and σ_s can be specified. The central operation hinges on the bilateral filter function $B(v_i)$, which refines each vertex by considering the weighted contributions of its neighboring vertices. These weights are determined by two factors: the spatial distance between vertex positions and the similarity in surface normals.

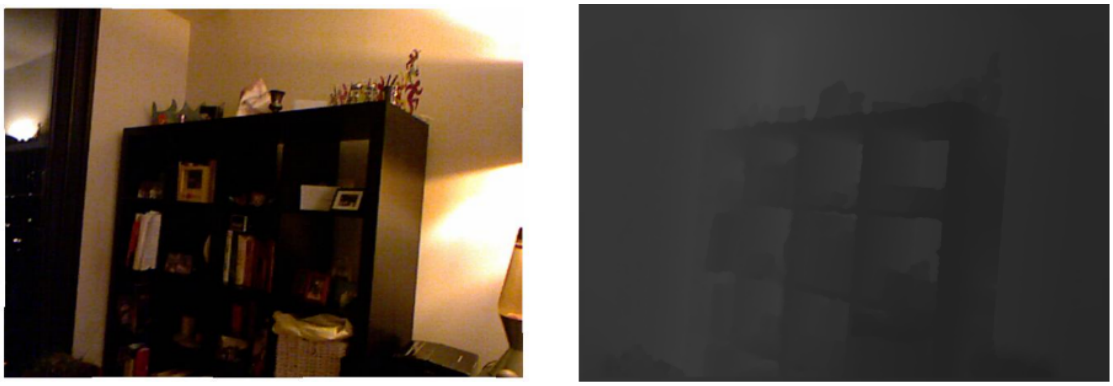

Figure 1: Example of rgb image (left) and depth image (right), from NYU dataset [28]

Mathematically, the bilateral filter function is expressed as:

$$B(v_i) = \frac{1}{W_i} \sum_{j=1}^{|V|} w_{ij} \cdot f(\mathbf{p}_j) \cdot \mathbf{n}_j$$

Here, w_{ij} represents the weight assigned to vertex v_j based on distance and similarity metrics, and $f(\mathbf{p}_j)$ encapsulates the geometric attributes associated with vertex v_j.

The entire denoising process is carried out through iterative refinement, updating vertex positions and normals based on the computed bilateral filter. The result is a denoised point cloud $\mathcal{M}_{\text{denoised}}$ that retains its critical geometric details while eliminating unwanted noise.

Figure 2: Example of merged and down sampled point cloud, from the acquisitions done with Kinect

5. Results

For the evaluation process of the two denoising methods, I employed distinct datasets. Firstly, a subset of the NYU Dataset [28] comprising randomly selected environmental images was utilized to ensure the attainment of valid and reliable results. Additionally, images were acquired using the Kinect depth camera. In both instances, each acquisition yielded an RGB image (.jpg in the case of the NYU dataset and .png in the Kinect acquisition) as well as a corresponding depth image (.png in both cases) Figure 1. Subsequently, each pair of images was amalgamated to form a unified point cloud containing comprehensive information. The point clouds originated from the same scene, were further merged through the application of the Iterative Point Cloud (ICP) algorithm. Following the amalgamation of all information into a singular extensive point cloud, I applied downsampling using the voxel downsampling function of Open3D [27], employing a size of 0.000000005. This procedure served to retain crucial points while simultaneously reducing dimensions to expedite processing Figure 2.

Two distinct experiments were conducted. The first involved visual assessments employing data obtained from the Kinect depth camera. This entailed a comparative analysis of the denoising outcomes produced by the 3D Gaussian filter and the Denoising Bilateral filter Figure 3. The second experiment entailed augmenting the data from the NYU dataset with Gaussian noise. Subsequently, the noisy input was subjected to the denoising filters. Following the acquisition of denoised outputs, a comparative analysis was performed with the original point cloud (prior to the application of Gaussian noise).

In this second case, for the assessment of the obtained results, I employed three distinct metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the

Figure 3: The screenshots of the *Why don't you speak* App for smartphone (from left to right Home page, Create the deepfake, Know your statue and Add your audio).

F-Score. More in details, Mean Absolute Error (MAE) is used to quantify the average difference in position between corresponding points in two point clouds. For each pair of corresponding points p_{ref} and p_{target}, the absolute error is computed as the Euclidean distance between them:

$$\text{Absolute Error} = ||p_{\text{ref}} - p_{\text{target}}||$$

The Mean Absolute Error is then calculated as the average of these absolute errors:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} ||p_{\text{ref}_i} - p_{\text{target}_i}||$$

where n is the total number of corresponding points. A smaller value of MAE indicates a better alignment between the two point clouds.

Root Mean Square Error (RMSE) is a commonly used metric for quantifying the average discrepancy between corresponding points in two point clouds. It's particularly useful in point cloud evaluation, especially when you want to understand the overall accuracy of a registration or reconstruction process. The RMSE is obtained by taking the square root of the MSE:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

A smaller value of RMSE indicates a better alignment between the two point clouds.

The F-score is used to measure the overlap between two point clouds, particularly when dealing with segmentation tasks. It is a metric that combines precision and recall to provide a single value that summarizes the quality of the correspondence between two point clouds. Precision is the ratio of true positive points to the total number of points classified as positive:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall is the ratio of true positive points to the total number of actual positive points:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The F-score is then calculated as the harmonic mean of precision and recall:

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F-score ranges from 0 to 1, with higher values indicating better correspondence between the point clouds. It strikes a balance between precision and recall, making it a useful metric for evaluating segmentation performance. The table [TABLE] displays the outcomes achieved on the one of the Bedroom subsets of the NYU dataset. Specifically, I incorporated four images (for each RGB and depth components) as acquisitions, a choice driven by computational constraints. Additionally, a Gaussian noise with a standard deviation of 0.00001 was introduced.

6. Conclusion

elaborate on the results and discuss their implications

References

[1] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, Guided 3d point cloud filtering, Multimedia Tools and Applications 77 (2018) 17397 – 17411. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85032034079&doi=10.1007%2fs11042-017-5310-9&partnerID=40&md5=f0073d14d06b0eca0bf7b61a96aeae33. doi:10.1007/s11042-017-5310-9, cited by: 22.

[2] J. Digne, C. de Franchis, The Bilateral Filter for Point Clouds, Image Processing On Line 7 (2017) 278 – 287. URL: https://hal.science/hal-01636966. doi:10.5201/ipol.2017.179.

[3] Microsoft company, 1975. URL: https://www.microsoft.com.

[4] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271) (1998) 839–846. URL: https://api.semanticscholar.org/CorpusID:14308539.

[5] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, ACM Trans. Graph. 22 (2003) 950–953. URL: https://doi.org/10.1145/882262.882368. doi:10.1145/882262.882368.

[6] Z. Feng, Z. Chao, Y. Huamin, Z. Lin, Point cloud denoising with principal component analysis and a novel bilateral filter., Traitement du Signal 36 (2019) 393 – 398. URL: https://login.ezproxy.uniroma1.it/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=141469443&lang=it&site=eds-live&scope=site.

[7] K. He, J. Sun, X. Tang, Guided image filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 1397–1409. doi:10.1109/TPAMI.2012.213.

[8] Y. Zheng, G. Li, S. Wu, Y. Liu, Y. Gao, Guided point cloud denoising via sharp feature skeletons, The Visual Computer 33 (2017) 1–11. doi:10.1007/s00371-017-1391-8.

[9] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, B. Guo, Rolling guidance normal filter for geometric processing, ACM Trans. Graph. 34 (2015). URL: https://doi.org/10.1145/2816795.2818068. doi:10.1145/2816795.2818068.

[10] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, Z. Xie, A feature-preserving framework for point cloud denoising, Computer-Aided Design 127 (2020) 102857. URL: https://www.sciencedirect.com/science/article/pii/S0010448520300506. doi:https://doi.org/10.1016/j.cad.2020.102857.

[11] C. Duan, S. Chen, J. Kovačević, Weighted multi-projection: 3d point cloud denoising with estimated tangent planes, 2018. arXiv:1807.00253.

[12] M. A. Irfan, E. Magli, Exploiting color for graph-based 3d point cloud denoising, Journal of Visual Communication and Image Representation 75 (2021). URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099775572&doi=10.1016%2fj.jvcir.2021.103027&partnerID=40&md5=3f72bfaf44cc7a4c9882ba20eea0e33c. doi:10.1016/j.jvcir.2021.103027, cited by: 6.

[13] E. Mattei, A. Castrodad, Point cloud denoising via moving rpca, Computer Graphics Forum 36 (2017) 123–137. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13068. doi:https://doi.org/10.1111/cgf.13068.

[14] E. Leal, G. Sanchez-Torres, J. W. Branch, Sparse regularization-based approach for point cloud denoising and sharp features enhancement, Sensors 20 (2020) 3206. URL: http://dx.doi.org/10.3390/s20113206. doi:10.3390/s20113206.

[15] R. Charles, S. Hao, M. Kaichun, J. Leonidas, Pointnet: Deep learning on point sets for 3d classification and segmentation. stanford: Stanford university (2017).

[16] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. arXiv:1706.02413.

[17] P. Guerrero, Y. Kleiman, M. Ovsjanikov, N. J. Mitra, Pcpnet learning local shape properties from raw point clouds, Computer Graphics Forum 37 (2018) 75 – 85. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85047834256&doi=10.1111%2fcgf.13343&partnerID=40&md5=033295762eb5f5b40c50e18fef0ab5c8. doi:10.1111/cgf.13343, cited by: 182; All Open Access, Green Open Access.

[18] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, M. Ovsjanikov, Pointcleannet: Learning to denoise and remove outliers from dense point clouds, Computer Graphics Forum 39 (2020) 185 – 203. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85068262115&doi=10.1111%2fcgf.13753&partnerID=40&md5=f837b907c7d283732cc0fa4a38409bcf. doi:10.1111/cgf.13753, cited by: 113; All Open Access, Green Open Access.

[19] R. Roveri, A. Cengiz Öztireli, I. Pandele, M. Gross, Pointpronets: Consolidation of point clouds with convolutional neural networks, Computer Graphics Forum 37 (2018) 87 – 99. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85051490490&doi=10.1111%2fcgf.13344&partnerID=40&md5=76fbf2a1ebb0aa3d21dc0ad1d9e1345a. doi:10.1111/cgf.13344, cited by: 52.

[20] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, P.-A. Heng, Ec-

net: an edge-aware point set consolidation network, 2018. arXiv:1807.06010.

[21] D. Lu, D. Zhang, Q. Zhao, X. Lu, X. Shi, A critical factor for quantifying proteins in unmodified gold nanoparticles-based aptasensing: The effect of ph, Chemosensors 8 (2020) 1 – 12. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85093667635&doi=10.3390%2fchemosensors8040098&partnerID=40&md5=4034fb6d8b69a0539b842fb7480bdceb. doi:10.3390/chemosensors8040098, cited by: 3; All Open Access, Gold Open Access.

[22] F. Pistilli, G. Fracastoro, D. Valsesia, E. Magli, Learning robust graph-convolutional representations for point cloud denoising, IEEE Journal on Selected Topics in Signal Processing 15 (2021) 402 – 414. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85098758705&doi=10.1109%2fJSTSP.2020.3047471&partnerID=40&md5=d13984623e311c300858ab4e97377771. doi:10.1109/JSTSP.2020.3047471, cited by: 12.

[23] D. Zhang, X. Lu, H. Qin, Y. He, Pointfilter: Point cloud filtering via encoder-decoder modeling, IEEE Transactions on Visualization and Computer Graphics 27 (2021) 2015 – 2027. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85100442580&doi=10.1109%2fTVCG.2020.3027069&partnerID=40&md5=5147a9e3a55eee46c9fe1c01bb5ca88c. doi:10.1109/TVCG.2020.3027069, cited by: 30; All Open Access, Bronze Open Access, Green Open Access.

[24] V. Naumova, K. Schnass, Dictionary learning from incomplete data for efficient image restoration, 2017.

[25] J. Batson, L. Royer, Noise2self: Blind denoising by self-supervision, 2019. arXiv:1901.11365.

[26] A. Krull, T.-O. Buchholz, F. Jug, Noise2void - learning denoising from single noisy images, 2019. arXiv:1811.10980.

[27] Open3d library, 2018. URL: http://www.open3d.org/docs/release/index.html.

[28] Nyu dataset v2, 2012. URL: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.