applied sciences

MDPI

Article

An Improved Point Cloud Upsampling Algorithm for X-ray Diffraction on Thermal Coatings of Aeroengine Blades

Wenhan Zhao [1,2,†], Wen Wen [1,†], Ke Liu [3,†], Yan Zhang [2], Qisheng Wang [1,*], Guangzhi Yin [3], Bo Sun [1,3,*], Ying Zhang [3] and Xingyu Gao [3,*]

[1] Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China; 15650390060@163.com (W.Z.); wenwen@sinap.ac.cn (W.W.)
[2] Logistics Engineering College, Shanghai Maritime University, Shanghai 201306, China; zhangyan@shmtu.edu.cn
[3] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201204, China; liuk@sari.ac.cn (K.L.); yingz@sari.ac.cn (G.Y.); zhangying@sari.ac.cn (Y.Z.)
* Correspondence: wangqisheng@sinap.ac.cn (Q.W.); sunb@sari.ac.cn (B.S.); gaoxy@sari.ac.cn (X.G.)
† These authors contributed equally to this work.

Abstract: X-ray diffraction can non-destructively reveal microstructure information, including stress distribution on thermal coatings of aeroengine blades. In order to accurately pinpoint the detection position and precisely set the measurement geometry, a 3D camera is adopted to obtain the point cloud data on the blade surface and perform on-site modeling. Due to hardware limitations, the resolution of raw point clouds is insufficient. The point cloud needs to be upsampled. However, the current upsampling algorithm is greatly affected by noise and it is easy to generate too many outliers, which affects the quality of the generated point cloud. In this paper, a generative adversarial point cloud upsampling model is designed, which achieves better noise immunity by introducing dense graph convolution blocks in the discriminator. Additionally, filters are used to further process the noisy data before using the deep learning model. An evaluation of the network and a demonstration of the experiment show the effectivity of the new algorithm.

Keywords: point cloud upsampling; aeroengine blades; deep learning; generative adversarial network

Citation: Zhao, W.; Wen, W.; Liu, K.; Zhang, Y.; Wang, Q.; Yin, G.; Sun, B.; Zhang, Y.; Gao, X. An Improved Point Cloud Upsampling Algorithm for X-ray Diffraction on Thermal Coatings of Aeroengine Blades. *Appl. Sci.* **2022**, *12*, 6807. https://doi.org/10.3390/app12136807

Academic Editor: Byung-Gyu Kim

Received: 23 March 2022
Accepted: 9 May 2022
Published: 5 July 2022

1. Introduction

High-pressure turbine blades are the core hot-end components of aeroengines. Detecting the coating surface of blades with complex shapes and cross-sections, and obtaining surface microstructure information, including stress distribution, are of great significance for guiding the preparation, subsequent processing, and service of turbine blades [1]. X-ray diffraction is one of the most widely used and most accurate characterization methods for studying the microstructure of materials. In particular, high-intensity synchrotron radiation X-rays can perform high-sensitive, high-spatial-resolution, and high-energy resolution analysis to obtain essential structural information such as the internal crystal structure, stress–strain, phase transition, and defects of the thermal barrier coating [2,3].

To determine the coordinates of the measured point and the X-ray incident angle, point cloud data are adopted to perform 3D modeling at the beamline BL14B1 [4] in the Shanghai Synchrotron Radiation Facility (SSRF), which is shown in Figure 1. Point cloud data of the blade's surface are collected with a 3D camera. In the 3D model, the XYZ coordinate information and the X-ray incidence angle are defined and the target point to be detected is selected. Then, the normal direction of that point is calculated. According to the mapping of the 3D model and the real samples, the blade mounted on the six-axis motorized platform is driven to the desired position. Finally, the diffraction data are collected to analyze the stress information.

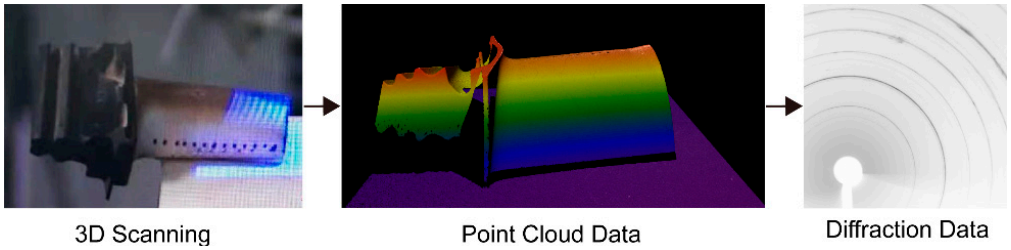

Figure 1. Point cloud used in the X-ray diffraction experiment.

However, the point cloud data generated by the 3D camera is relatively sparse and has certain deviations, which cannot meet the accuracy requirements needed to measure the residual stress of the target point accurately. The upsampling algorithm can process the original data to generate dense, complete, and uniform point cloud data [5]. With the rapid development of the deep learning method for point clouds [6–8], point cloud upsampling research based on deep learning has achieved state-of-the-art results, such as PU-Net [9], EC-Net [10], MPU [11], PU-GCN [12], and PU-GAN [13]. However, the point cloud obtained from the 3D camera contains a lot of noisy data. Inputting the raw data directly into those models will further amplify the noises. Some subtle features will be lost, and some new noise points and non-uniform points will be generated. If it occurs in the region of interest (ROI) of the blade's surface, the wrong point may be selected for the subsequent experiment. This problem can be mitigated by the statistical outlier removal (SOR) method [14], which can remove the scattered noise points and abnormal points in the point cloud data.

In this work, the point cloud upsampling algorithm is applied to improve the accuracy of the experiment. We improve the existing deep learning model using dense graph convolution blocks to make the generated point cloud closer to the real point cloud distribution, called the graph convolution point cloud upsampling generative adversarial network (GPU-GAN). In this paper, a new discriminator model is designed, which gives the model better anti-noise ability by learning the features between points and points in the coordinate space neighborhood. Before using the deep learning model, we use the SOR and pass-through filter [15] to preprocess the point cloud in order to reduce noise interference and improve the operation speed. We evaluate the effectiveness of GPU-GAN by analyzing upsampled point cloud data as well as X-ray diffraction experiments.

2. Methodology

The process of establishing the coordinates of the 3D space model is shown in Figure 2. Firstly, calibration experiments are carried out to reduce the deviation that exists in the process of the experiment. Then, the fast cluster SOR (FCSOR) is adopted to remove noise points, and a pass-through filter is selected to remove redundant points that are not relevant to the experiment. Finally, the point cloud processed by the filter is input to the upsampling model, and the final point cloud is obtained.

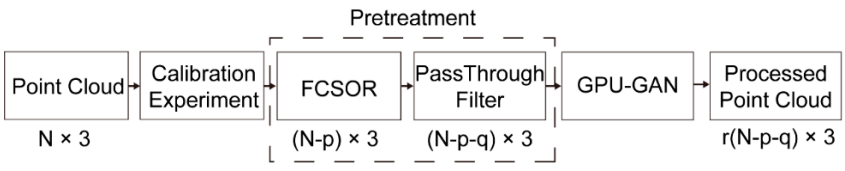

Figure 2. Overview of 3D space model coordinates. r is the upsampling rate.

Calibration Experiment: The platform on which the sample is placed can be moved along the six-axis to adjust the blade's posture. Before the experiment, it is necessary to calibrate the platform motion coordinate system and calculate the platform motion parameters, which can set up the mapping relationship between the real sample and the point cloud. There is a deviation when using the point cloud data to locate the sample and

move it to the target point. The deviation comes from two aspects: the platform motion deviation and the point cloud matching position deviation. The platform motion deviation mainly comes from the stepper motor, such as the wear of the stepper motor in use and environmental interference. Point cloud matching position deviation mainly comes from the 3D cameras. There is a certain deviation when the 3D camera obtains the coordinate information of the blade. The platform motion deviation can be eliminated by calculating the difference between the theoretical position and the actual position of the object for calibration. The platform motion deviation is calculated as follows:

$$e_{platform}(x, y, z, rx, ry, \theta) = P_{act}(x, y, z, rx, ry, \theta) - P_{theor}(x, y, z, rx, ry, \theta) \quad (1)$$

where P represents the center of the object for calibration, and x, y, z, rx, ry, θ represents the coordinates in the six degrees of freedom direction.

The object matching deviation can be eliminated by calculating the difference between the actual and the theoretical position of the sample. The object matching deviation is calculated as follows:

$$e_{camera}(x, y, z) = P'_{act}(x, y, z) - P'_{theor}(x, y, z) \quad (2)$$

where P' represents the center of the sample, and x, y, z represent the 3D coordinates of the point.

Through the above calibration experiments, the deviation can be reduced and the positioning accuracy of the sample can be improved. Because the calibration experiment is the preprocess step of our research, the detail is not described here.

Pretreatment: We select SOR to denoise the point cloud. For each point, the average distance d is calculated from all points in its k neighborhood. The average distance corresponding to each point is obtained in the input point cloud. For all points in the input point cloud, it is assumed that each element in the distance array constitutes a Gaussian distribution, which is determined by the mean and standard deviation of the sample. Corresponding points whose d value is outside the standard range (defined by the mean and variance of the sample) can be defined as outliers and removed from the data set. Because there are millions of points in the data sets, the method proposed by FCSOR to speed up the calculation time is introduced [16]. For each point, the average squared Euclidean distance of its k-nearest neighbors are calculated and the points are divided into different clusters. Then, the average number of points in each cluster is counted and the SOR calculations are only performed on clusters with fewer points than the average.

After performing the filtering, we use the pass-through filter to remove irrelevant data and retain the point cloud in the area required for the experiment. Algorithm 1 shows the detailed calculation process of the pretreatment.

μ is the denoising coefficient. When d_i is not within the range of $\overline{d} \pm 3\sigma$, the point p_i can be removed as a noise point. In the 3D point cloud denoising processing, μ can be adjusted as needed. If μ is larger, fewer points will be deleted; if μ is smaller, more points will be deleted [14]. The filtering effect of SOR is affected by the noise removal coefficient μ and the number of points k in the field. In the actual application process, by comparing the filtering effects under different parameters, the parameters with the better effect are selected, which can remove noise points as much as possible and retain the real value. In this paper, we define the number of clusters as 10, k as 50, and μ as 1. These data are obtained by actual experimental testing. For different point cloud data, different parameters need to be tested to obtain a better filter effect.

Algorithm 1 Pretreatment

Input: Point cloud $P = \{p_i\},\ p_i = (x_i, y_i, z_i)$
Output: Filtered point cloud $O = \left\{o_j\right\}, o_j = \left(x_j, y_j, z_j\right)$

function Fast cluster statistic outlier removal (P)
  Defining cluster size
  $N \leftarrow Clusternumber$
  $ClusterLenth = max\{x_i\} - min\{x_i\}/N$
  $ClusterWidth = max\{y_i\} - min\{y_i\}/N$
  $ClusterHeight = max\{z_i\} - min\{z_i\}/N$
  Subdividing point cloud space into N clusters $C = c_k$
  for $p_i \in P$ **do**
    $d_i = \left(\sum_i^k k\_nearestNeighbourDistance(p_i)\right)/k$
    add p_i appropriate cluster c_k
  end for
  $\mu_\mu = i/N$
  Retain clusters C_u with number of points less than μ_μ
  for $p_i \in C_u$ **do**
    $\overline{d}_i = \frac{1}{n}\sum_i^n d_i$
    $\sigma = \sqrt{\frac{1}{n}\sum_i^n \left(d_i - \overline{d}_i\right)^2}$
    $P' = \left\{p_i \in P \middle| \left(\overline{d}_i - \mu\sigma\right) \le d_i \le \left(\overline{d}_i + \mu\sigma\right)\right\}$
  end for
  return P'
  end **function**

  function Pass-through filter (P')
    Defining experimental area (EA)
    $x_{min} \leftarrow EAx_{min}$
    $x_{max} \leftarrow EAx_{max}$
    $y_{min} \leftarrow EAy_{min}$
    $y_{max} \leftarrow EAy_{max}$
    return $O = \left\{p_i' \in P' \middle| x_{min} < x_i < x_{max}, y_{min} < y_i < y_{max}\right\}$
  end **function**

Upsampling: After filtering the point cloud, we use GPU-GAN to upsample the point cloud, which is an improvement based on PU-GAN [13]. The upsampling model is divided into two parts: the generator network and the discriminator network, as shown in Figure 3.

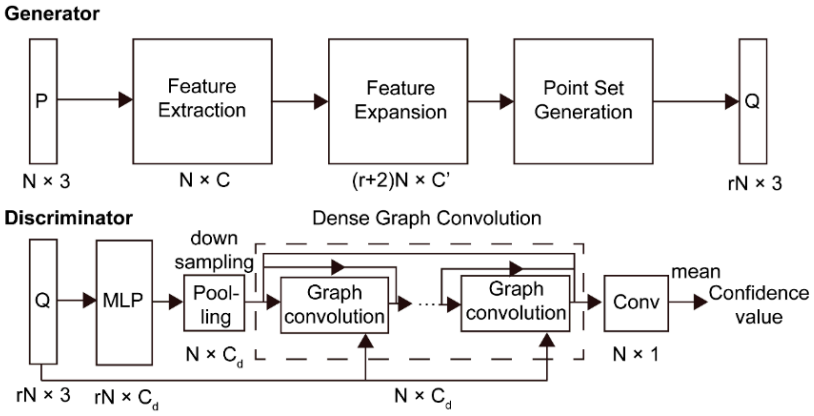

Figure 3. Overview of GPU-GAN's architecture. MLP represents multi-layer perceptron; Conv represents convolutional layer; N is the number of points in P; r is the upsampling rate; and C, C', C_d are the number of feature channels.

The generator network has three components to process the input point cloud sequentially. The purpose of the feature extraction component is to extract features from the input point cloud. The feature expansion component aims to expand the point feature. It upsamples the point feature to obtain the expanded feature, and then downsamples that to compute the difference between the features before and after upsampling. The difference is added with the first-step expanded feature to self-correct the expanded feature. The point set generation component returns a set of 3D coordinates from the point expansion feature through a set of multi-layer perceptrons. Finally, rN points are generated through a farthest point sampling step. r is the upsampling magnification. N is the number of points.

The purpose of the discriminator network is to distinguish whether the generator is input or not to help the generator training. Inspired by AR-GCN [17], we introduce a graph convolution module and redesign the discriminator part. Firstly, we extract global features of the input point cloud with shape $rN \times c$ using a set of MLPs. Then, a pooling block is used to downsample the global features to obtain features with the shape $N \times c$. Our introduction of a graph convolution block to further process the features and its structure is shown in Figure 4.

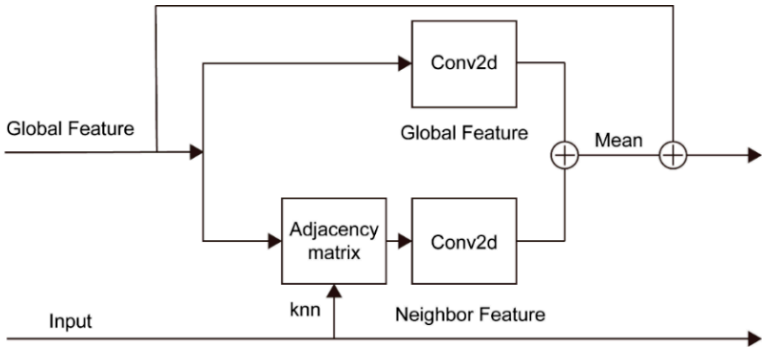

Figure 4. The structure of a graph convolution block.

Since the point cloud is disordered and does not have a predefined adjacency matrix, we define the adjacency matrix $N(p)$ by using input point cloud to query the k nearest neighbors of p. Firstly, we calculate the Euclidean distance between each point in the point cloud and other points, and arrive at the k-nearest neighbors of point p_i. Then, the index information of the k neighbors is recorded to obtain the adjacency matrix $N(p_i)$ corresponding to the point p_i. Finally, according to the neighborhood information defined by the $N(p_i)$, the local features corresponding to each point are learned through the convolution operation. The feature calculation method is as follows:

$$F_i^p = F_i^p + \sum_{n=1}^{k} F_i^n,\ k \in N(p) \tag{3}$$

where F_i^n represents the feature of point p at layer i. To avoid the degradation problem in deep learning network training and speed up the network's training, we introduce a dense connection between each graph convolutional block. Compared with residual links [18], dense connections [19] can enable each layer to receive the features of all previous layers, which can achieve feature reuse, improve training efficiency, and improve the performance of the discriminator. After the graph convolution block, we process the features through a convolutional layer, output the one-dimensional feature values of 256 points, and arrive at the confidence value $D(Q)$ via averaging them.

The network is trained with a patch-based approach, which finds 200 seed positions on each model, and then uses Poisson disk sampling at each point to generate rN points as a patch, denoted as $\hat{Q}$. N points are selected randomly from $\hat{Q}$ as the input P of the

network. The least squares loss is defined as the adversarial loss for the generator network and the discriminator network:

$$min\ L_{G\_adv}(Q) = \frac{1}{2}[D(Q) - 1]^2 \tag{4}$$

$$min\ L_{D\_adv}(Q) = \frac{1}{2}\left[D(Q)^2 + \left(D(\hat{Q}) - 1\right)^2\right] \tag{5}$$

The generator fools the discriminator by minimizing L_{G_adv}, and the discriminator distinguishes between the real point cloud $\hat{Q}$ and the generated point cloud Q by minimizing L_{D_adv}.

In order to improve the generated point cloud quality, we introduce the uniform loss proposed by PU-GAN [13]:

$$L_{uni}(S_j) = \sum_{j=1}^{M}\left[\frac{\left(|S_j| - \hat{n}\right)^2}{\hat{n}} \times \sum_{j=1}^{|S_j|}\frac{\left(d_{j,k} - \hat{d}\right)^2}{\hat{d}}\right] \tag{6}$$

where M is obtained by farthest sampling the generated point cloud Q. S_j is the point set obtained by using a ball query for each point in M with radius r_d, and M is set to 50. $\hat{n} = \hat{Q} \times r_d^2$ is the expected number of points in S_j. $d_{j,k}$ is the distance from each point in S_j to its k nearest neighbors. $\hat{d} = \sqrt{\frac{2\pi r_d^2}{|S_j|\sqrt{3}}}$ is the expected distance of the point in the uniform point cloud to its k nearest neighbors. The deviation of S_j from $\hat{n}$, $d_{j,k}$ from $\hat{d}$ is evaluated using a chi-squared model.

Chamfer distance (CD) [9] and Earth mover's distance (EMD) [20] are selected to construct the reconstruction loss to encourage the generated points to lie on the target surface:

$$L_{rec} = \sum_{q_i \epsilon Q} min_{p_j \in \hat{Q}}\left|\left|q_i - P_j\right|\right|_2^2 + \min_{\phi : Q \to \hat{Q}} \sum_{q_i \epsilon Q}\left|\left|q_i - \phi(q_i)\right|\right|_2 \tag{7}$$

where $\phi : Q \to \hat{Q}$. is the bijection mapping.

Finally, the above losses are weighted and summed to obtain the compound loss:

$$L_G = w_a L_{G\_adv} + w_u L_{uni} + w_r L_{rec} \tag{8}$$

$$L_D = L_{D\_adv} \tag{9}$$

where w_a, w_u, w_r are weights, w_a is set as 1, w_u is set as 20, and w_r is set as 100. The generator and discriminator are optimized alternatively.

3. Experiments

3.1. Data and Implementation Details

One hundred and twenty point clouds provided by PU-GAN [13] plus three blade surface point clouds are used for training. Then, 200 patches are cropped from each 3D model as the input of the network for a total of 24,600 patches, and the number of points N per patch is 256. Each patch is composed of low-resolution point clouds and ground truth point clouds sampled by Poisson disks. The batch size is set to 32. The upsampling rate r is set to 4. The Adam algorithm [21] with a two time-scale update rule (TTUR) [22] is adopted to train the network. The learning rate of the generator is 0.001, and the learning rate of the discriminator is 0.0001. After 30,000 iterations, the learning rate is gradually reduced by a decay rate of 0.8 per 30,000 iterations until 10^{-6}. The network is implemented on NVIDIA GeForce RTX 2080Ti GPU and Intel Xeon Gold 5218 CPU using TensorFlow [23] on the Ubuntu 16.04 system.

In order to reduce the interference of noise and reduce the amount of data, we use the pass-through filter to prune parts of the point cloud that are not relevant to the experiment

and use FCSOR to remove noise values from point clouds. For FCSOR, we set the noise denoising coefficient μ to 1 and the number of points k in the neighborhood to 50. The software framework is developed using the C++ programming language based on the Point Cloud Library (PCL) [24].

The 3D camera, Sizector HD40, which is designed with phase-shifting structured light technology, produced by Shanghai ShengXiang Industrial Detection Technology, is selected to obtain point cloud data for evaluating the algorithm's performance. The point cloud data used in the X-ray diffraction experiment are acquired by Sizector 3D R600 and also produced by Shanghai ShengXiang Industrial Detection Technology. The experimental hardware setup is shown in Figure 5. The robot holds the 3D camera through an adaptor. The blade sample is positioned on the goniometer platform of the diffractometer.

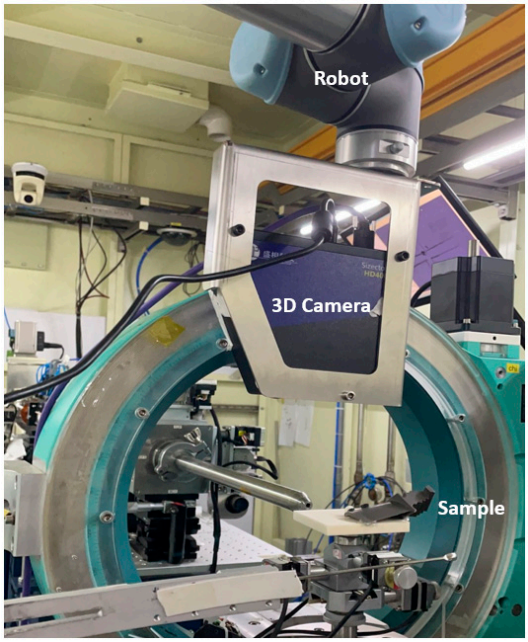

Figure 5. The hardware of the experimental setup.

Meshlab [25] is used for point cloud visualization, which is an open-source, portable, and extensible 3D geometry processing system, mainly used for interactive processing and unstructured editing of 3D triangular meshes.

3.2. Evaluation Metrics

Following previous point cloud sampling work, we utilize the standard chamfer distance (CD) [9] and Earth mover's distance (EMD) [20] to measure the difference between Q and $\hat{Q}$, for which smaller is better. CD can calculate the average of the distance between the generated point cloud point and the nearest point in the ground truth. EMD measures the minimum cost of turning generated point cloud into the ground truth. The noise contained in the input point cloud can interfere with the accuracy of CD and EMD. Because the point cloud data of the blade are too dense and contain many delicate wavy surfaces, it was difficult to reconstruct high-quality mesh data, so we did not choose the commonly used point-to-surface distance.

The noise contained in the input point cloud interferes with the accuracy of CD and EMD, so we report an F-score [17] to further evaluate the model performance, for which larger is better. F-score defines point cloud upsampling as a classification problem, evaluating precision and recall by examining the percentage of points in Q or $\hat{Q}$ that are able to find another neighbor within a certain threshold τ.

Deviation [17] is also used to measure the difference between the generated point cloud and the ground truth, and the normalized uniformity coefficient (NUC) [9] is used to measure the uniformity, for which smaller is better.

3.3. Analysis and Comparison of Experimental Results

GPU-GAN, MPU and PU-GAN's upsampling results are qualitatively and quantitatively compared. For MPU and PU-GAN, we use their public code and retrain their networks using our training data. Since the blade samples are very expensive and the quantity is limited, we use four blades, and each blade takes three sets of point cloud data from different angles. We sample 25% points using the uniform downsampling method from the ground truth as the input. We choose representative point clouds to show qualitative results.

Quantitative results: As shown in Table 1, GPU-GAN achieves significant improvements over MPU and PU-GAN under most metrics for data with noisy points. Even for dense blade point clouds, the NUC stays the lowest for all different p, indicating that GPU-GAN can generate more uniform points. It should be noted that for the EMD, the result of GPU-GAN is higher than that of PU-GAN, but lower than that of MPU because the points generated by MPU are too concentrated near the real point, which makes the EMD calculation result of MPU low. For the standard (std) of deviation, the result of MPU is several times higher than that of PU-GAN and GPU-GAN, because there are so many outliers in the point cloud that they interfere with the results. For the point cloud processed with filter, GPU-GAN shows superior performance too. For model training and testing time, GPU-GAN takes more time than MPU and PU-GAN, which is due to the complex model structure. Compared to GPU-GAN's improvement in upsampling, these increased time costs are acceptable. The test point cloud has about 900,000 points, and after filtering, there are about 700,000 points.

Table 1. Quantitative comparisons with the state-of-the-arts.

Method	CD	EMD	F-Score		NUC with Different p			Deviation		Time	
			τ = 0.01	τ = 0.02	0.2%	0.4%	0.6%	mean	std	Trian (h)	Test (min)
MPU	0.026	0.888	0.091	0.495	19.087	10.807	8.225	0.021	0.068	15.7	42.1
PU-GAN	0.032	1.357	0.258	0.564	17.967	9.641	6.753	0.018	0.017	19.2	55.6
GPU-GAN	0.024	1.126	0.328	0.649	17.855	9.505	6.522	0.016	0.023	21.1	56.3
Filter + MPU	0.016	0.362	0.106	0.583	13.201	9.869	5.772	0.019	0.009	-	29.3
Filter + PU-GAN	0.016	0.447	0.293	0.654	12.317	8.909	4.845	0.016	0.009	-	38.5
Filter + GPU-GAN	0.014	0.405	0.375	0.753	12.295	8.866	4.792	0.014	0.008	-	38.9

Qualitative results: Figure 6 shows the qualitative upsampling results of MPU, GPU-GAN, and PU-GAN. The raw data contain a lot of noise points, which cannot be avoided in experiments. After upsampling the input point cloud, MPU generates fewer outliers, and the surrounding point cloud shows a striped distribution, as shown in the blue rectangle in Figure 6f, which means that the points generated by the MPU are too concentrated near the real points, making upsampling less effective. Although PU-GAN generates more uniform points, it learns the characteristics of noisy points and generates more outliers, as shown in Figure 6g. GPU-GAN improves this phenomenon and constrains the generated points near the real surface, as shown in Figure 6h. It can be observed that the noise-prone holes in Figure 6 significantly reduce outliers. In Figure 6, we remove some elements, because the X-ray diffraction experiment only needs the point cloud data of the blade surface, and only a smaller part may be retained during the actual experiment. Figure 7 shows the effect of adding filters on the upsampling results. The filter can effectively remove noise interference, reduce the amount of data, and improve the running rate of the algorithm.

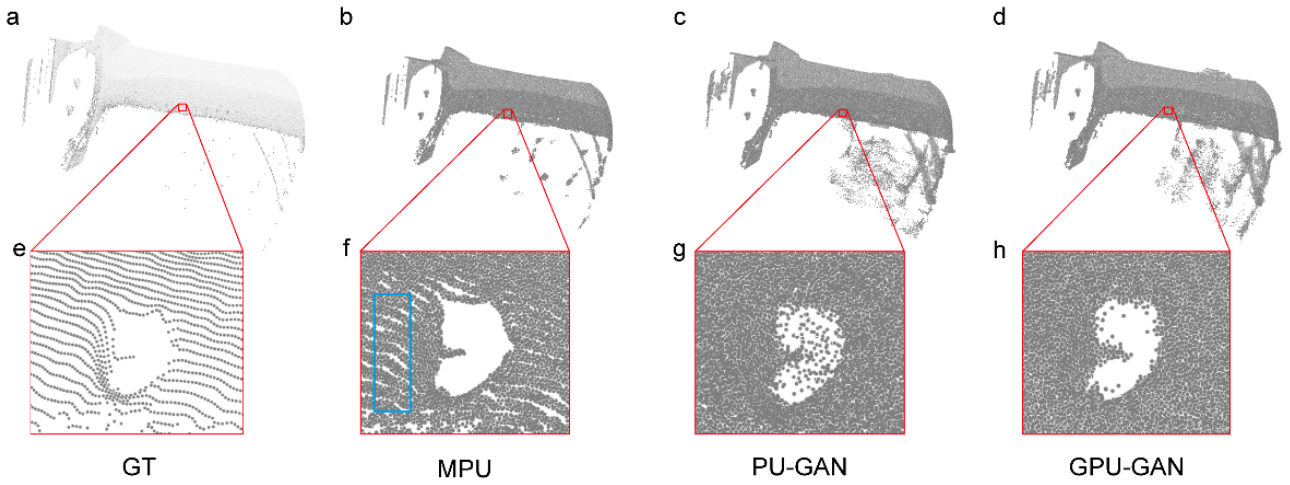

Figure 6. Qualitative comparisons with PU-GAN. (**a**–**d**) are the ground truth and the upsampling point cloud of MPU, PU-GAN, GPU-GAN respectively. (**e**–**h**) are the point clouds at the outlet of the air-cooling channel enlarged corresponding to (**a**–**d**) point cloud respectively.

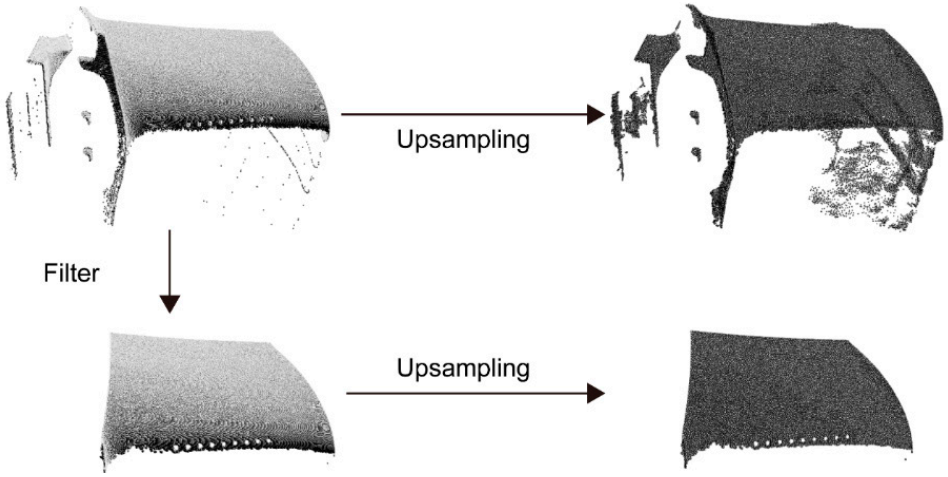

Figure 7. Comparison of results after adding filters.

3.4. X-ray Diffraction Experiment

The X-ray diffraction (XRD) experiment on the thermal coatings of the aeroengine blade reflects the necessity of the upsampling algorithm. The experiment is carried out on the beamline BL14B1 at Shanghai Synchrotron. The 3D camera used is the Sizector 3D R600. Compared with the HD40, this camera has a larger standard field of view (XY) and z-axis measurement range and the ability to resist ambient light interference. However, the point cloud it generates is not dense enough, and experiments can only be performed at a scale of 1 mm (X–Y axis). We use the upsampling algorithm to upsample the collected point cloud and construct a 3D space motion coordinate system by generating the point cloud. The measurement accuracy at the X–Y axis can reach 0.5 mm.

As mentioned in Seq.2 Calibration Experiment, the mapping relationship between the real sample and the point cloud is set up first. Then, we select five points in the original point cloud, and another four points from the upsampled point cloud for experiments. These four points are located between the five points of the original point cloud. The diffraction data are collected by the Rayonix MX225 CCD detector, and processed by the Fit2D software [18]. The location diagram of those diffraction points and the integration result of the XRD data are shown in Figure 8.

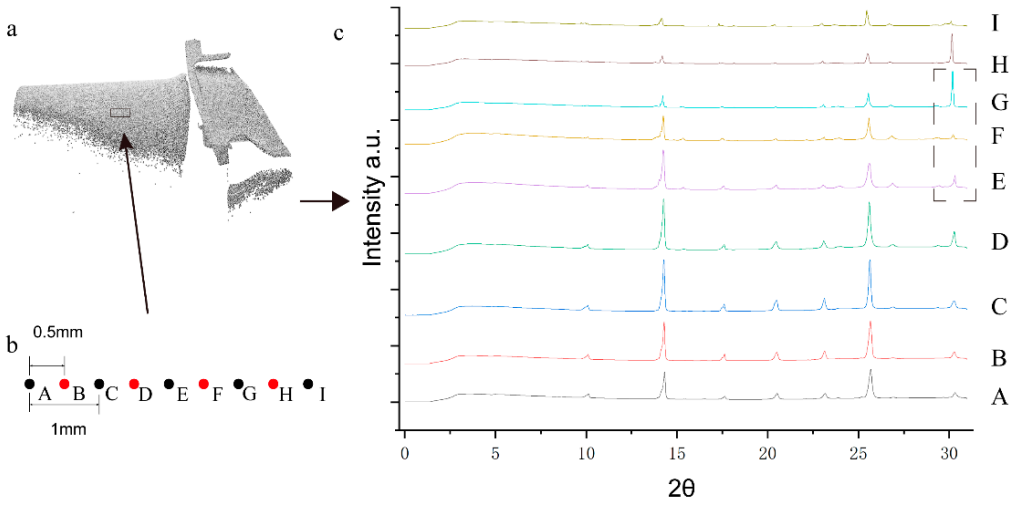

Figure 8. Relative position and intensity information of different points. (**a**) is the blade point cloud, (**b**) is the detection point, and (**c**) is the diffraction data corresponding to each detection point.

The black points (A/C/E/G/I) are selected in the original point cloud, and the red points (B/D/F/H) are selected in the generated point cloud. The interval between them is 500 µm. Figure 8b is the diagram showing locations of those points only on the x axis. By observing data curve E, F, and G in Figure 8c, the diffraction peak appears at ~31° (2θ) and shows a trend of the diffraction signal weakening and then enhancing. This conclusion cannot be drawn without the information measured at the F point. The supplemented points from upsampling the data cloud can effectively complement the integrity of the original point cloud data and can obtain more continuous and complete experiment results.

Further processing of the X-ray diffraction data can obtain more information on the thermal coatings of the blade, for example, the evolution of the stress on the coatings. The diffraction experiment here shows the effectivity of our upsampling method. The further data processing and results are not discussed here.

4. Conclusions

A 3D camera is useful to accurately define the X-ray incidence angle in X-ray experiments, but it is met with the problems of insufficient resolution and too many noisy points. This paper proposes a new upsampling method, GPU-GAN, which improves the upsampling algorithm based on the PU-GAN and dense graph convolution method. The experimental results show that our method can conduct experiments more efficiently and accurately. In addition, our method can save hardware expenses, so that low-resolution 3D cameras can also meet experimental needs.

However, current deep learning models are not ideal for upsampling data with a large number of noise points. In addition, because the blade point cloud data are too large, the time for the algorithm to run once is also unacceptable. In future work, we will collect more blade data and add it to the data set for training. We are also considering extracting regional point cloud data required for experiments and streamlining the network structure to reduce program running time.

Author Contributions: Methodology, W.Z.; writing—original draft, W.Z.; software, W.Z. and Y.Z. (Ying Zhang); supervision, Y.Z. (Yan Zhang) and B.S.; writing—review and editing, Y.Z. (Yan Zhang), X.G. and B.S.; data curation, K.L.; validation, K.L., W.W. and G.Y.; funding acquisition, W.W.; investigation, Q.W.; project administration, X.G. and B.S.; conceptualization, X.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by funds the National Key Research and Development Program of China (2017YFA0403400) and the National Science Foundation of China (U1932201).

Data Availability Statement: Data not yet publicly available.

Acknowledgments: We thank the staff from beamline BL14B1, the Experimental Auxiliary System, and the Data Center of Shanghai Synchrotron Radiation Facility (SSRF) for on-site assistance.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Padture, N.P.; Gell, M.; Jordan, E.H.J.S. Thermal barrier coatings for gas-turbine engine applications. *Science* **2002**, *296*, 280–284. [CrossRef]
2. Schulz, U.; Leyens, C.; Fritscher, K.; Peters, M.; Saruhan-Brings, B.; Lavigne, O.; Dorvaux, J.-M.; Poulain, M.; Mévrel, R.; Caliez, M.J.A.S.; et al. Some recent trends in research and technology of advanced thermal barrier coatings. *Aerosp. Sci. Technol.* **2003**, *7*, 73–80. [CrossRef]
3. Drakopoulos, M.; Connolley, T.; Reinhard, C.; Atwood, R.; Magdysyuk, O.; Vo, N.; Hart, M.; Connor, L.; Humphreys, B.; Howell, G. I12: The joint engineering, environment and processing (JEEP) beamline at diamond light source. *J. Synchrotron Radiat.* **2015**, *22*, 828–838. [CrossRef] [PubMed]
4. Siddiqui, S.F.; Knipe, K.; Manero, A.; Meid, C.; Wischek, J.; Okasinski, J.; Almer, J.; Karlsson, A.M.; Bartsch, M.; Raghavan, S.J.R.o.S.I. Synchrotron X-ray measurement techniques for thermal barrier coated cylindrical samples under thermal gradients. *Rev. Sci. Instrum.* **2013**, *84*, 083904. [CrossRef] [PubMed]
5. Tie-Ying, Y.; Wen, W.; Guang-Zhi, Y.; Xiao-Long, L.; Mei, G.; Yue-Liang, G.; Li, L.; Yi, L.; He, L.; Xing-Min, Z. Introduction of the X-ray diffraction beamline of SSRF. *Nucl. Sci. Tech.* **2015**, *26*, 20101-020101. [CrossRef]
6. Charles, R.Q.; Su, H.; Kaichun, M.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 77–85. [CrossRef]
7. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5105–5114.
8. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *Acm Trans. Graph.* **2019**, *38*, 1–12. [CrossRef]
9. Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.-A. Pu-net: Point cloud upsampling network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2790–2799. [CrossRef]
10. Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.-A. Ec-net: An edge-aware point set consolidation network. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 386–402. [CrossRef]
11. Yifan, W.; Wu, S.; Huang, H.; Cohen-Or, D.; Sorkine-Hornung, O. Patch-based progressive 3d point set upsampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5958–5967. [CrossRef]
12. Qian, G.; Abualshour, A.; Li, G.; Thabet, A.; Ghanem, B. PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
13. Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.-A. Pu-gan: A point cloud upsampling adversarial network. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 7203–7212. [CrossRef]
14. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Dolha, M.; Beetz, M. Towards 3D point cloud based object maps for household environments. *Robot. Auton. Syst.* **2008**, *56*, 927–941. [CrossRef]
15. Miknis, M.; Davies, R.; Plassmann, P.; Ware, A. Near real-time point cloud processing using the PCL. In Proceedings of the 2015 International Conference on Systems, Signals and Image Processing (IWSSIP), Singapore, 10–12 September 2015; pp. 153–156. [CrossRef]
16. Balta, H.; Velagic, J.; Bosschaerts, W.; De Cubber, G.; Siciliano, B. Fast statistical outlier removal based method for large 3D point clouds of outdoor environments. *IFAC-PapersOnLine* **2018**, *51*, 348–353. [CrossRef]
17. Wu, H.; Zhang, J.; Huang, K. Point cloud super resolution with adversarial residual graph networks. *arXiv Prepr.* **2019**, arXiv:02111. [CrossRef]
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
19. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
20. Fan, H.; Su, H.; Guibas, L.J. A point set generation network for 3d object reconstruction from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, London, UK, 21–26 July 2017; pp. 605–613. [CrossRef]
21. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv Prepr.* **2014**, arXiv:1412.6980. [CrossRef]
22. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6629–6640.

23. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. {TensorFlow}: A System for {Large-Scale} Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
24. Rusu, R.B.; Cousins, S. 3d is here: Point cloud library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4. [CrossRef]
25. Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G. Meshlab: An open-source mesh processing tool. In Proceedings of the Eurographics Italian Chapter Conference, Salerno, Italy, 2–4 July 2008; pp. 129–136. [CrossRef]