Juan Diego Toscano
Mechanical Engineering,
Universidad de las Fuerzas Armadas ESPE,
Sangolquí 171103, Ecuador
e-mail: jdtoscano@espe.edu.ec

Christian Zuniga-Navarrete
Industrial Engineering,
University of Louisville,
Louisville, KY 40292
e-mail: cszuni01@louisville.edu

Wilson David Jo Siu
Industrial and Systems Engineering,
University at Buffalo,
Buffalo, NY 14260
e-mail: wilsonda@buffalo.edu

Luis Javier Segura
Assistant Professor
Industrial Engineering, University of Louisville,
Louisville, KY 40292
e-mail: ljsegu01@louisville.edu

Hongyue Sun[1]
Assistant Professor
Industrial and Systems Engineering,
University at Buffalo,
Buffalo, NY 14260
e-mail: hongyues@buffalo.edu

Teeth Mold Point Cloud Completion Via Data Augmentation and Hybrid RL-GAN

Teeth scans are essential for many applications in orthodontics, where the teeth structures are virtualized to facilitate the design and fabrication of the prosthetic piece. Nevertheless, due to the limitations caused by factors such as viewing angles, occlusions, and sensor resolution, the 3D scanned point clouds (PCs) could be noisy or incomplete. Hence, there is a critical need to enhance the quality of the teeth PCs to ensure a suitable dental treatment. Toward this end, we propose a systematic framework including a two-step data augmentation (DA) technique to augment the limited teeth PCs and a hybrid deep learning (DL) method to complete the incomplete PCs. For the two-step DA, we first mirror and combine the PCs based on the bilateral symmetry of the human teeth and then augment the PCs based on an iterative generative adversarial network (GAN). Two filters are designed to avoid the outlier and duplicated PCs during the DA. For the hybrid DL, we first use a deep autoencoder (AE) to represent the PCs. Then, we propose a hybrid approach that selects the best completion to the teeth PCs from AE and a reinforcement learning (RL) agent-controlled GAN. Ablation study is performed to analyze each component's contribution. We compared our method with other benchmark methods including point cloud network (PCN), cascaded refinement network (CRN), and variational relational point completion network (VRC-Net), and demonstrated that the proposed framework is suitable for completing teeth PCs with good accuracy over different scenarios.
[DOI: 10.1115/1.4056566]

Keywords: data augmentation, hybrid RL-GAN, point cloud completion, teeth alignment

1 Introduction

A misaligned tooth can be treated with proper cosmetic dentistry products, also known as teeth aligners [1]. 3D printing of teeth aligners is promising since no patient has a similar set of teeth with the same dimensions and form of misalignment. Thus, 3D printed teeth aligners have been recently predominant in orthodontics as an alternative to traditionally manufactured teeth aligners [2]. The key advantages of 3D printed teeth aligners include fewer clinical emergencies and improved aesthetics, comfort, oral hygiene, periodontal health, and lack of soft tissue irritation. In addition, the 3D printed aligners have high-resolution digitally designed borders, smoother edges that do not need post-processing polishing, and customizable intra-aligner thickness, compared with traditional fabrications [2].

As shown in Fig. 1, teeth scans are required for many applications in restorative dentistry and orthodontics [3]. In particular, dentists use teeth scans to define a suitable treatment and design the aligner, which includes annotation, segmentation, alignment, and rotation [4]. Most 3D data are acquired using laser scanners, three-dimensional cameras, and computed tomography (CT)/magnetic resonance imaging scanners in the form of point clouds (PCs) [5,6]. PCs are highly memory efficient and preserve fine surface details [7,8]. Several deep learning (DL) approaches have addressed the shape completion problem for 3D PCs [6,7,9–11]. However, DL models generally require a significant amount of data for their training [12], which hampers their applications in some medical domains with limited data [13]. Therefore, there is a need for an efficient PC completion framework that works with limited data. To address these problems, we propose a two-step data augmentation (DA) technique, followed by a hybrid DL approach to complete the PCs.

To start, we use the human bilateral symmetry to split and recombine the teeth PCs to enlarge our dataset (see Fig. 2(*a*.1)). However, some combined PCs could be problematic. Specifically, if the combined PCs are too similar to other PCs (i.e., redundant PCs), the dataset could become redundant, which may result in model performance degradation [14]. Meanwhile, if the combined PCs are too different from the raw PCs (i.e., outlier PCs), the trained model may not be accurate [15]. Consequently, we develop two filters to discard the defective (i.e., redundant or outlier) PCs by comparing the raw and combined PCs using chamfer distance d_{CH}.

In addition, generative adversarial networks (GANs) have been used as a DA technique [13,16,17]. GANs can create fake data that resemble the real data from a random vector (seed **z**) [18] and are particularly useful in the medical domain [16]. Hence, in the second DA step, we train a latent space GAN (l-GAN) to generate fake PCs iteratively. In each iteration, we create fake PCs from a set of seed **z**. Then, we use our filters to isolate the useful PCs' seed **z** distribution and apply the new distribution to generate new PCs in the next iteration (see Fig. 2(*a*.2)). Consequently, an augmented dataset is obtained and then used to train a deep autoencoder (AE) and a reinforced-learning agent-controlled GAN (RL-GAN) [6].

RL is used to optimize system performance based on training so that the system can automatically learn to solve complex tasks from the input and the reward [19–21]. Then, we use the AE and RL-GAN to complete the incomplete PCs and select the best completion by comparing their similarity with the incomplete PCs (see Fig. 2(*b*)).

An ablation study is performed to analyze each component's contribution [22]. We compared our method with other benchmark methods including point cloud network (PCN), cascaded refinement network (CRN), and variational relational point completion network (VRC-Net).

[1]Corresponding author.

Contributed by the Computers and Information Division of ASME for publication in the Journal of Computing and Information Science in Engineering. Manuscript received July 12, 2022; final manuscript received December 18, 2022; published online January 10, 2023. Assoc. Editor: Wang Jun.

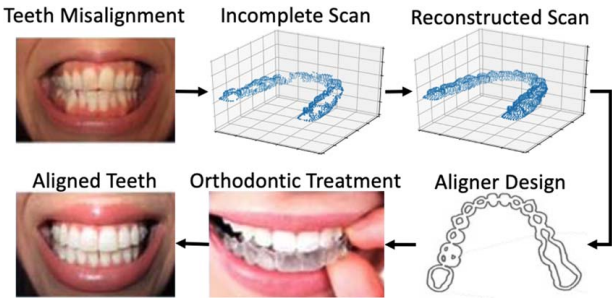

Fig. 1 Illustration of teeth alignment treatment procedures

The main contributions of this study are summarized as follows:

(1) We propose customized data augmentation and filtering methods that exploit the human teeth' bilateral symmetry and iterative l-GAN for fake PC generation.
(2) We use a hybrid AE and RL-GAN framework to identify the best teeth PC completion.

This paper is organized as follows. In Sec. 2, we review the related studies. Then, in Sec. 3, we discuss the proposed approach. In Sec. 4, we introduce our models' implementation and show the experimental results. Finally, in Sec. 5, we conclude the paper and discuss future work.

2 Literature Review

2.1 Teeth Molds and Intraoral Scans. Teeth molds/dental impressions are transcendental for patient dental diagnosis and treatment [23]. Traditionally, dental impressions have been performed on elastomers [24], alginates [25], wax [23], plaster [26], etc. For instance, Megremis et al. [24] evaluated eight elastomeric occlusal registration models for restorative dental procedures. Hellmann et al. [25] obtained dental impressions made from alginate for bite recording and prosthetic reconstruction planning. See also Refs. [27,28]. Although traditional dental impressions have benefited dental diagnosis and treatment, these methods are invasive, time-consuming, and produce high material waste.

Current digitization technology has enabled one to obtain digital impressions for subsequent diagnosis and procedure planning (e.g., orthodontia and surgery planning) [29,30]. Several scanning methods have been used to digitize the dental impressions, such as X-ray [31], optical scanning [32], and computer tomography (CT) [33]. For instance, Kamegawa et al. [34] measured dental casts with a micro-focus X-ray for a 3D morphological assessment of occlusion treatment. Kang et al. [35] used 3D optical scanning of dental casts for bite registration. See other examples in Refs. [36,37]. These methods have helped to ameliorate the limitations of conventional teeth molds/dental impressions, however, they still need conventional dental impressions as a starting point.

Intraoral scanning (IOS) can produce digital impressions with minimum patient invasion. Current IOS technologies include light projection, distance object determination, and reconstruction [38]. Ireland et al. [39] described the utilization of light projection (e.g., digital fringe) to obtain accurate digital dental impressions. Pradíes et al. [40] used stereophotogrammetric technology for obtaining intraoral digital impressions of implants. See similar studies in Refs. [41,42]. Generally, scanning technology has proven to be effective at representing 3D objects and facilitating the utilization of traditional manufacturing processes (e.g., milling) and additive manufacturing in the dentistry industry. However, irrespective of the scanning methods, the teeth molds/dental impressions suffer from outliers, occlusion, irregularity, and unstructuredness [43].

2.2 Point Cloud Shape Denoising and Completion. PCs have become popular to represent 3D objects in various fields, such as robotics, autonomous driving, and 3D modeling and fabrication [44]. The PCs need to undergo denoising and completion to represent an entire 3D object (e.g., teeth mold) [5,44].

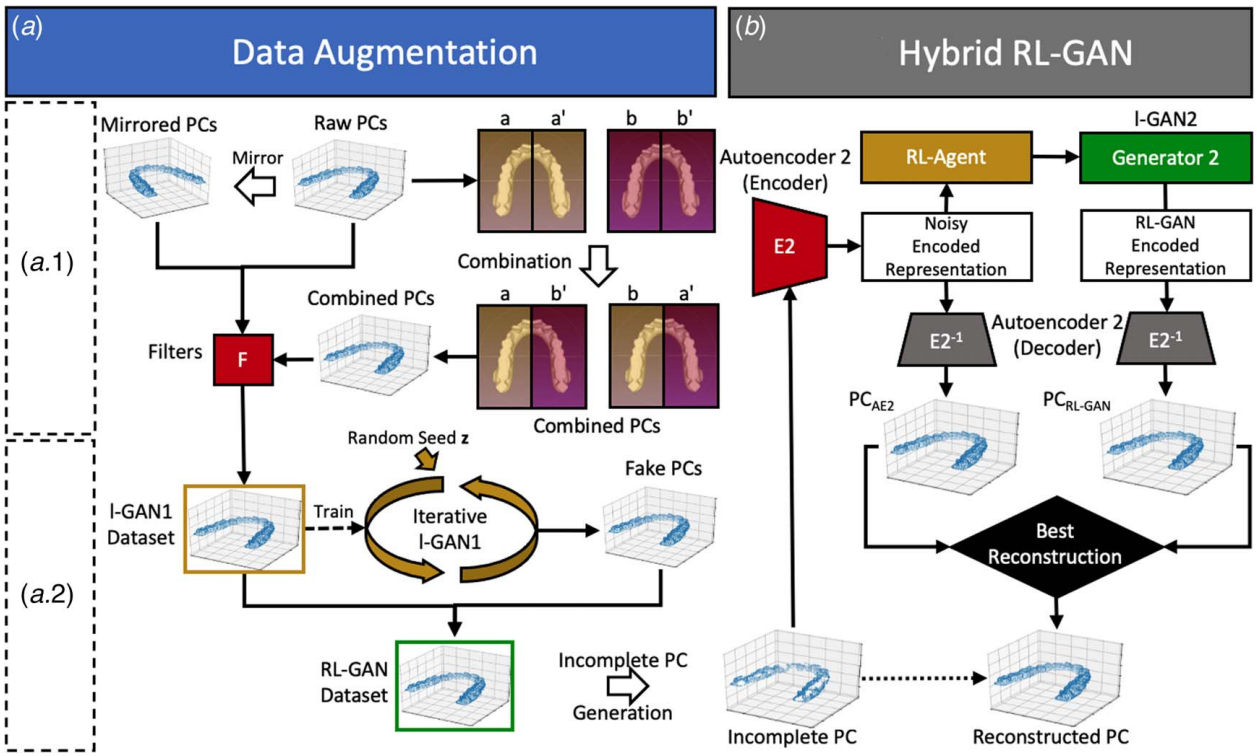

Fig. 2 Proposed framework: (*a*) data augmentation, (*a*.1) point cloud combination and, (*a*.2) iterative l-GAN1, and (*b*) hybrid RL-GAN

Conventional methods, such as density-based and geometry-based methods, have been deployed for PC denoising and completion, respectively [45]. Ester et al. [46] developed a density-based algorithm for discovering clusters in large spatial databases with noise. Zhao et al. [47] presented a robust hole-filling algorithm for triangular mesh. Here, new vertices are re-positioned by solving the Poisson equation. See other similar studies in Refs. [48,49]. These methods heavily rely on assumptions, such as symmetry and shape similarity, which are not suitable for unstructured data, as is the case of PCs.

Machine learning (ML) approaches have also demonstrated important progress for PC denoising and completion via dimensional reduction and regression techniques [50–52]. Duan et al. [50] applied a principal component analysis-based approach for low-complexity PC denoising for LiDAR data. Sarkar et al. [51] developed a structured low-rank matrix factorization for PC denoising. Gandler et al. [52] presented an object shape estimation approach based on sparse Gaussian process implicit surfaces combining visual data and tactile exploration. See also Refs. [53,54]. Although ML methods are robust, their performances are limited for complex shapes or considerably large missing areas in the PCs.

DL methods have demonstrated good performance for PC denoising and completion [6,44]. For instance, Yuan et al. [55] proposed a point cloud completion network (PCN). This pioneering work consists of an encoder–decoder network to reconstruct dense and complete point sets from an incomplete point cloud. Pan et al. [56] exploited multi-scale local point features to reconstruct point clouds with fine-grained geometric details to predict local and thin shape structures in their VRC-Net. In addition, AE and GAN-based approaches have outperformed traditional methods [57,58]. Zong et al. [59] proposed a denoising AE for learning robust local region features from partial inputs. Wang et al. [60] developed a CRN for point cloud completion. See also Refs. [7,61]. The performance of these methods is affected by small sample sizes and robustness during training [62].

In addition, training a GAN is an unstable process and may suffer from model collapse [6]. To address these issues, Sarmad et al. [6] presented an RL-GAN network for real-time PC completion. However, the model performance is still insufficient for small sample sizes and can be improved by deploying DA techniques and l-GAN-based fake PC generation, which will be addressed in this paper.

3 Proposed Framework

Figure 2 shows our proposed framework to complete the 3D PCs with a limited number of PCs. First, we propose a two-step DA technique to enlarge the quantity and diversity of the PCs. In the first step, we generate new PCs by splitting and recombining the raw PCs based on the bilateral symmetry of human teeth, as shown in Fig. 2(*a*.1). Then, we apply two filters to remove the outliers and the redundant PCs from the combined PCs. In the second step, we use the raw and filtered combined PCs to train AE1. AE1 creates a latent representation of the PCs, which are used to train the l-GAN1. The l-GAN1 can generate new PC's encoded representations from a random vector (i.e., seed **z**). To make sure the generated PCs are consistent with the teeth molds, we propose to use filters to isolate the useful PCs and modify the seed **z** distribution iteratively (see Fig. 2(*a*.2)).

Second, we deploy a hybrid approach that completes the PCs using two methods, namely, AE2 and RL-GAN, as shown in Fig. 2(*b*). In the first method, AE2 takes the encoded representation and decodes it back into a completed teeth PC (PC_{AE2}). Consequently, in the second method, the RL agent uses the encoded representation from AE2 to control the l-GAN2 generator to get RL-GAN encoded representation which is turned into a complete PC (PC_{GAN2}) using the AE2 decoder. Finally, we select the best completion by computing the similarity between the output PCs (PC_{AE2} and PC_{GAN2}) and the input PC (i.e., incomplete PC). We then perform the ablation study to investigate the contribution of each component. We introduce details of the proposed framework in the following sections.

3.1 Point Cloud Combination.

A small training dataset may cause overfitting and can significantly affect the generalization capability of a neural network [63]. Data augmentation is a general technique to alleviate the problems caused by data sparsity [64]. Hence, after mirroring our dataset, we combine our raw PCs following the procedure described in Fig. 2(*a*.1). To start, we take two PCs (aa' and bb', where $aa' \neq bb'$) from the raw dataset and divide them into left (i.e., a and b) and right sides (i.e., a' and b') by the median plane. A median plane is a sagittal plane placed in the center of the human body that divides it into two symmetrical parts [65].

The median plane is determined as follows: (1) The PCs are translated to be centered and scaled to unit length. (2) We compute the principal component axes of the first PC using principal component analysis and then aligned the x-, y-, and z-axis with the principal component axes. This step allows us to align the PC with a reference [66]. Since the PCs of teeth molds are symmetric, after the alignment, the y–z plane coincides with the median plane. (3) Finally, we register the remaining PCs to the first PC using an iterative closes point algorithm [54]. The relative positions of the teeth point cloud and the median plane are determined based on the Euclidean distance of the scaled PCs to the origin of the x-, y-, and z-axis. Then, we combine the right halves with the left halves (i.e., a with b' and b with a') to obtain two new PCs per combination. Hence, we generate $n_g = 2\binom{T}{2} = T(T-1)$ combined samples, where T is the number of PCs in the raw dataset.

PC combination helps to enlarge our dataset. However, some generated PCs can be outliers or redundant PCs. To address this issue, we design two filters to discard outliers and redundant PCs. We use chamfer distance (d_{CH}), a broadly adopted metric to measure the similarity between two PCs [67], to quantify the differences between PCs. The d_{CH} between two PCs (P_1 and P_2) is defined as

$$d_{CH}(P_1, P_2) = \frac{1}{|P_1|}\sum_{x\in P_1}\min_{y\in P_2}\|x-y\|_2^2 + \frac{1}{|P_2|}\sum_{y\in P_2}\min_{x\in P_1}\|x-y\|_2^2 \quad (1)$$

where each point $x \in P_1$ finds its nearest neighbor $y \in P_2$ and vice versa. All the point-level pairwise distances are averaged to produce the shape-level distance [67].

We compute the d_{CH} between every pair of PCs in the raw dataset and define the min and max thresholds as the minimum and maximum d_{CH}, respectively. These thresholds are used in our designed filters to remove redundant and outlier PCs. In particular, the first filter (F1) is designed to remove outliers. We first calculate the d_{CH} between the generated PCs and the first PC in the raw dataset. Then F1 removes the outlier PCs that have d_{CH} larger than the max threshold. Here, only the first sample is picked to avoid the computational burden otherwise incurred in comparing with all raw PCs. Then, the second filter (F2) iteratively removes the redundant PCs by maintaining a pairwise matrix of d_{CH} of generated PCs. In each iteration, F2 removes the PC with the maximum number of redundant samples (i.e., d_{CH} that are smaller than the min threshold) with other PCs. Then, it updates the pairwise distance matrix. The filtering process is repeated until all the redundant PCs have been removed (i.e., there is no d_{CH} in the pairwise distance matrix that is less than the min threshold).

Consequently, the final number (n_f) of generated PCs is $n_f = n_g - n_{F1} - n_{F2}$, where n_g is the original number of generated PCs and n_{F1} and n_{F2} are the number of PCs removed by F1 and F2, respectively. Finally, we group the gathered data (i.e., raw, mirrored, and n_f combined PCs) as the l-GAN1 dataset, which is used to train our iterative l-GAN1 network for the second step of DA.

3.2 Iterative L-GAN1. As shown in Fig. 3, we propose an iterative l-GAN framework to iteratively generate PCs, remove outlier and redundant PCs, isolate the useful PCs' seed $\mathbf{z}$ distribution, and use the new distribution to generate fake PCs for DA. The above steps are repeated until a certain number of iterations is achieved.

3.2.1 Autoencoder 1. An AE is composed of an encoder (E) and a decoder (E^{-1}). The E is a network unit through which the input (i.e., PC) is transformed into a multidimensional array referred to as a global feature vector (GFV) (i.e., latent representation). On the other hand, the decoder E^{-1} is a fully connected network that reverts the process by transforming the GFV back into the raw PC space. To train our AE, we implement a weighted loss function:

$$L_{\text{AE}} = \omega_{\text{CH}} L_{\text{CH}} + \omega_{\text{GFV}} L_{\text{GFV}} \tag{2}$$

where L_{CH} is the d_{CH} between the input (PC_{in}) and output (PC_{out}) PCs and L_{GFV} is the L_2 distance between the input and output PC's GFV (i.e., E (PC_{in}) and E (PC_{out})). ω_{CH} and ω_{GFV} are the corresponding weights.

To train our AE, we use the Adam stochastic gradient descent optimizer [68]. The detailed architecture, momentum, learning rate, and other parameters will be introduced in Sec. 4.3.1. To train the AE1, we use the l-GAN1 dataset (i.e., raw, mirrored, and combined PCs). Then, we use the AE1's encoder ($E1$) to generate a latent representation of our l-GAN1 dataset.

3.2.2 Latent Space Generative Adversarial Network 1. Several studies have shown that training a GAN on a latent representation leads to more stable results compared to training on raw PCs [6,9]. Here, we apply our pre-trained AE1 to obtain our l-GAN1 dataset's GFVs. Then, we use the GFVs to train the l-GAN1 (see Fig. 2(*a*.2)).

An l-GAN is composed of two separate deep neural networks trained against each other. One network, the generator, is responsible for generating the actual output. The other network, the discriminator, is responsible for distinguishing between the generator's output and the ground truth data. To train our l-GAN, we apply a self-attention framework [69] with a Wasserstein GAN gradient penalty (WGAN-GP) adversarial loss [70] as proposed by Sarmad et al. The discriminator (D) and generator (G) loss functions are described in Eqs. (3) and (4), respectively.

$$L_D = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \tag{3}$$

$$L_G = -\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] \tag{4}$$

where $\mathbb{E}$ is the expectation, $\mathbb{P}_r$ is the encoded raw input data distribution, $\mathbb{P}_g$ is the distribution for $\tilde{x} = G(\boldsymbol{z})$, $\boldsymbol{z}$ is a seed $\mathbf{z}$, λ is a regularization parameter, $\hat{\boldsymbol{x}} = \varepsilon \boldsymbol{x} + (1-\varepsilon)\tilde{\boldsymbol{x}}$ is an intermediate variable computed at each training step using a random number ε, and $\mathbb{P}_{\hat{x}}$ is the distribution of $\hat{\boldsymbol{x}}$ [70]. L_D is a modified *Earth-Mover* distance constructed using the Kantorovich–Rubinstein duality and a gradient penalty to circumvent tractability issues [70]. Since D estimates the probability that a sample comes from the real data, L_G is large when G produces GFVs that do not resemble the real data. The detailed l-GAN architecture, learning rate, and other parameters will be introduced in Sec. 4.3.2.

3.2.3 Iterative L-GAN1. Once trained, the l-GAN1 generator (G1) transforms a noise vector (i.e., seed $\mathbf{z}$) into the desired target distribution (i.e., fake GFV). The fake GFV can be decoded into a fake PC with $E1^{-1}$. Hence, we propose to use our pre-trained l-GAN1 and filters to iteratively modify the seed $\mathbf{z}$'s distribution and generate the fake PCs.

An overview of the iterative l-GAN1 algorithm is shown in Algorithm 1. We first use the pre-trained G1 to generate n_g fake GFVs from n_g seeds $\mathbf{z}_s$. Then, we apply our $E1^{-1}$ to the fake GFVs to decode them into fake PCs. We then use F1 and F2 to remove the outlier and redundant PCs, and isolate useful seed $\mathbf{z}_s$ (Z_1) from the filtered PCs and store the remaining PCs in an accumulator PC_T. Consequently, we estimate the Z_1's distribution by fitting its mean and covariance matrix and use them to generate a new set of n_g GFVs. The above processes are repeated for $\text{it}_{\max}$ iterations. Finally, we remove the potential redundant PCs by applying F2 to the accumulated data in PC_T (see Algortithm 1). By using the pre-trained l-GAN1 iteratively, we produce $n_{l-\text{GAN1}}$ fake PCs. Then, we group the raw and the generated data (i.e., mirrored, combined, and $n_{l-\text{GAN1}}$ l-GAN1 PCs) into an RL-GAN dataset that is used to train the hybrid RL-GAN.

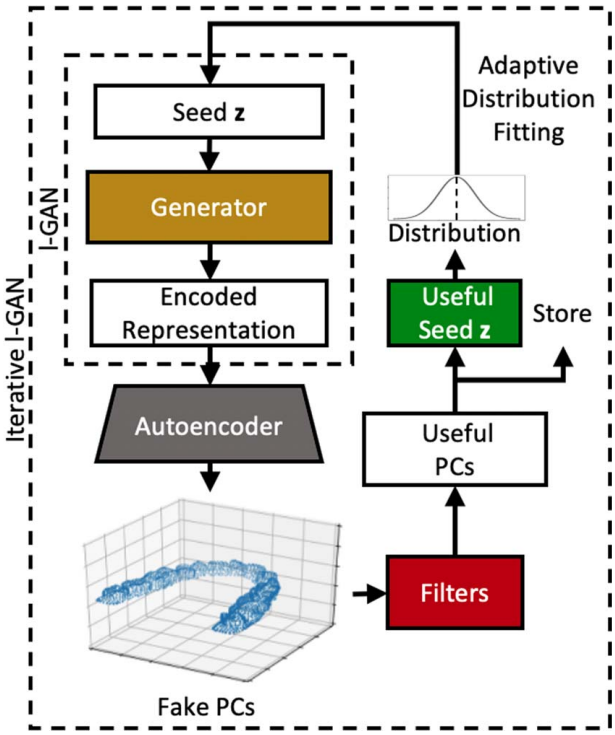

Fig. 3 Iterative l-GAN1 for data augmentation

Algorithm 1 Training iterative l-GAN1

Input models:
Pre-trained AE1 decoder: E_1^{-1}
Pre-trained l-GAN1 generator: G_1
Input functions:
Filter-1 (removes the outliers): $F1$
Filter-2 (removes the redundant PCs): $F2$
Input data:
Number of iterations: $\text{it}_{\max}$
Number of PCs generated per iteration: n_g
Final output:
Set of l-GAN1 PCs: PC_f

Initialize the mean: $\boldsymbol{\mu} = \mathbf{0}$
Initialize the covariance matrix: $\text{Cov} = I$
Initialize an empty array to store the generated PCs: PC_T
for it $<$ $\text{it}_{\max}$ **do**
  Use $\boldsymbol{\mu}$ and Cov to randomly generate a matrix containing n_g seed vectors $\mathbf{z}$: Z_0
  Obtain n_g GFVs: $\text{GFV}_0 = G_1(Z_0)$
  Obtain n_g PCs: $\text{PC}_0 = E_1^{-1}(\text{GFV}_0)$
  Remove the outliers with F1: $\text{PC}_1 = F1(\text{PC}_0)$
  Remove the redundant PCs with F2: $\text{PC}_2 = F2(\text{PC}_1)$
  Store the PC_1's corresponding seed $\mathbf{z}$: Z_1
  Store the PC_2 on PC_T
  Update the mean: $\boldsymbol{\mu} = \text{mean}(Z_1)$
  Update the covariance matrix: $\text{Cov} = \text{cov}(Z_1)$
end for
Remove the redundant PCs with F2: $\text{PC}_f = F2(\text{PC}_T)$

3.3 Hybrid RL-GAN. Our hybrid RL-GAN has three components, AE2, l-GAN2, and RL agent. We use the RL-GAN dataset to train the AE2 and l-GAN2 following the procedures described in Secs. 3.2.1 and 3.2.2, respectively. We will introduce the RL and hybrid RL-GAN in the next.

3.3.1 Reinforcement Learning. The objective of the RL agent is to learn behavior through trial-and-error interactions with the dynamic environment [71]. In this study, the environment is composed of the pre-trained AE2 and l-GAN2, while the action is the input (seed **z**) for the l-GAN2 generator (G2). We train our RL agent using the RL-GAN dataset, following the procedures shown in Algorithm 2 [6]. To start, the agent obtains an input state by encoding the input PC and picks a suitable seed **z**. Then, G2 uses the seed **z** to create an RL-GAN GFV, and the decoder ($E2^{-1}$) transforms the GFV into a complete PC. Depending on the quality of the action, the environment gives a reward (r) back to the agent. As shown in Fig. 4, the reward for the completion is the combination of negated loss functions that evaluate the intermediate results computed along the process. Specifically, we include the $r_{\text{CH}} = -L_{\text{CH}}$ to ensure that the complete PCs resemble the input PCs, the $r_{\text{GFV}} = -L_{\text{GFV}}$ to quantify the similarity between the input and output latent representations, and $r_D = D(GFV)$ to guarantee that the fake GFV follows the encoded real data distribution. The final combined reward function is shown in Eq. (5), where ω_{CH}, ω_{GFV}, and ω_D are the corresponding weights to each reward.

$$r = \omega_{\text{CH}} r_{\text{CH}} + \omega_{\text{GFV}} r_{\text{GFV}} + \omega_D r_D \tag{5}$$

Algorithm 2 Training RL-GAN [6]

Input models:
Pre-trained AE2 encoder: E_2
Pre-trained AE2 decoder: E_2^{-1}
Pre-trained l-GAN2 generator: G_2
Pre-trained l-GAN2 discriminator: D_2
Input data:
Number of iterations: $t_{\max}$
Starting time: t_0
Final output:
Completed PC: $\text{PC}_{\text{RL-GAN}}$

Initialize the environment ***Env***: E_2, E_2^{-1}, G_2 D_2
Initialize the policy π with ***DDPG***, actor ***A***, critic ***C***, and replay buffer ***R***
for $t < t_{\max}$ **do**
  Get PC_{in}
  if $t > 0$ **then**
    Train actor ***A*** and critic ***C*** with ***R***
  end if
  Get state: $s_t = E_2(\text{PC}_{\text{in}})$
  if $t < t_0$ **then**
    Obtain a random action: a_t
  else
    Obtain action: $a_t = \boldsymbol{A}(s_t)$
  end if
  Implement action: $\text{GFV}_{\text{RL-GAN}} = G_2(a_t)$
  Obtain final PC: $\text{PC}_{\text{RL-GAN}} = E_2^{-1}(\text{GFV}_{\text{RL-GAN}})$
  Compute the reward with Eq. (5): r_t
  Obtain new state: $s_{t+1} = E_2(\text{PC}_{\text{RL-GAN}})$
  Store transition (s_t, a_t, r_t, s_{t+1}) in ***R***
end for

To train the RL agent, we use a deep deterministic policy gradient (DDPG) [72]. The DDPG algorithm relies on a parameterized actor and a critic network. The actor–network specifies the current policy (π) by deterministically mapping states to a specific action [72]. On the other hand, the critic network provides a measure of the quality of action concerning the input state [6].

3.3.2 Hybrid RL-GAN. RL-GANs can complete the incomplete PCs but the completed PCs may not always preserve the local details well [6]. In contrast, a pre-trained AE can accomplish shape completion but its performance degrades drastically as the percentage of missing data increases [9]. To address these problems, we use a hybrid RL-GAN to select the best completion between RL-GAN and AE. In particular, we complete a PC using AE2 to get PC_{AE2} and RL-GAN to get $\text{PC}_{\text{RL-GAN}}$. Then, we compute the d_{CH} of PC_{AE2} and $\text{PC}_{\text{RL-GAN}}$ to the incomplete PC, respectively, and select the smaller d_{CH} between the two as the completion output.

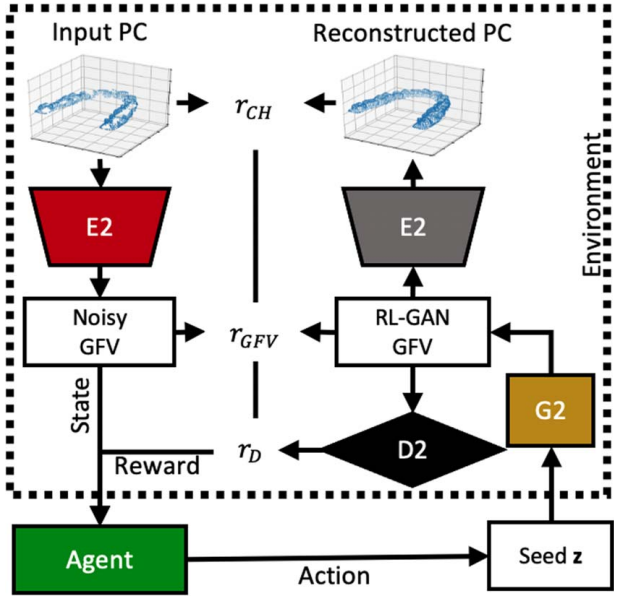

Fig. 4 RL-GAN for PCs completion

3.4 Ablation Study

In deep learning, an ablation study involves measuring the performance of a system after removing one or more of its components to help understand the relative contribution of the ablated components to overall performance [73–75].

To run our ablation study, we divide the framework into five modules (i.e., PC combination, filters, iterative l-GAN1, RL-GAN, and hybrid RL-GAN), as shown in Fig. 5. Then, we remove one module at a time and compare the ability of the ablated system to perform the reconstruction task against the original framework. To evaluate the ablated system performance, we compare it to the original system performance and calculate the relative increase in the chamfer distance (%IncCH), as shown in Eq. (6).

$$\%\text{IncCH}(\%A) = \frac{\bar{d}_{\text{CHas}}(\%A) - \bar{d}_{\text{CHos}}(\%A)}{\bar{d}_{\text{CHos}}(\%A)} \tag{6}$$

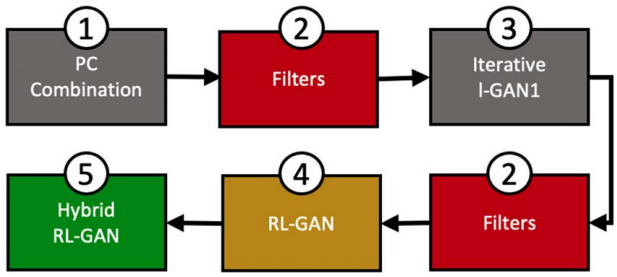

Fig. 5 Ablation Study Modules

where $\%A$ is the percentage of missing area, $\bar{d}_{\text{CHas}}$ is the average chamfer distance on the ablated system, and $\bar{d}_{\text{CHos}}$ is the average chamfer distance on the original system. The larger the %IncCH, the more important the module is to the framework.

4 Case Study

To start, we use the open-source library Open3D [76] to downsample, translate, and align our raw PCs. Then, we employ Pytorch [77] and Pytorch 3D [78] to implement our DL models. We train and evaluate our models on Google Colab Pro using an NVIDIA Tesla P100 server graphics card.

4.1 Data Acquisition and Preparation. Figure 6 shows the data acquisition and preparation procedures, including 3D printing, 3D scanning, and processing. We introduce the details below.

4.1.1 3D Printing and 3D Scanning. The lower jaw teeth molds were printed with an Ender Creality Pro printer. The printer specifications are: nozzle size 0.4 mm, infill density 20%, printing speed 50 mm/s, wall thickness 0.8 mm, and extruder temperature 200 °C. Forty-five teeth molds are printed in total. After the printing, a 3D light-based scanner, Solutionx C500 3D, with an incorporated base plate was used to scan the printed parts. The parameters used for the scanner are set to scanning area FOV350 (diagonal distance up to 350 mm), scanning volume $264 \times 218 \times 120$ mm, and point spacing 0.110. A customized scan path is used to optimize the scanning process time and improve the PC quality, following the steps proposed in Ref. [79]. A total of forty-five positions were used to finish a teeth mold scan. To obtain a single 3D mesh from a scan path, a multi-step registration was applied to the scanning sequence.

4.1.2 Data Preparation. We apply a voxel downsampling [76] to reduce the PCs' dimension to 2048 points. The downsampled PCs are translated to be centered and scaled to unit length. Additionally, to allow the combination process, we register our downsampled PCs by applying an iterative closest point algorithm [76], where all the PCs are registered to the first PC in the training dataset.

Once we process and prepare our raw data, we obtain 45 PCs, referred to as raw PCs. The raw PCs are then randomly and equally split into training, validation, and testing datasets. We apply the DA techniques described in Secs. 3.1 and 3.2 to our training data, and use this enlarged dataset to train the hybrid RL-GAN. Since the framework works in a latent representation, we must ensure that our AEs perform well for the encoding of new PCs.

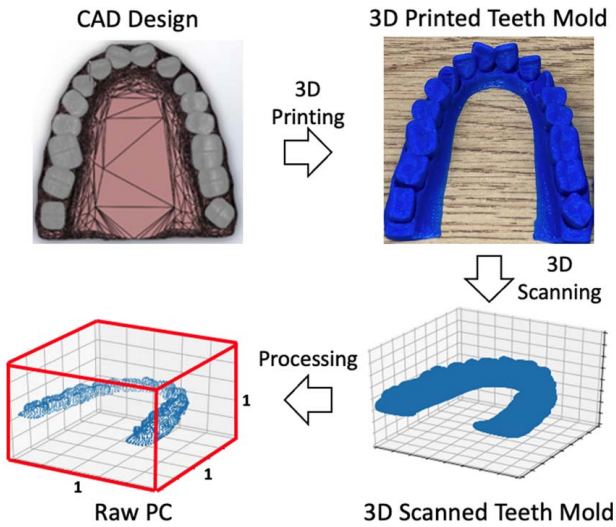

Fig. 6 Data acquisition and preparation

Consequently, we use the validation dataset to select the best AE1 and AE2 for the l-GAN1 and l-GAN2, respectively.

4.2 Point Cloud Combination. We then perform the first step of the DA. In particular, we duplicated our training dataset by mirroring the 15 raw PCs. After that, we combined our 15 raw PCs and generated $n_g = 210$ PCs. Then, we computed the d_{CH} between every pair of PCs in our raw dataset to get the minimum ($\min = 5.141 \times 10^{-5}$) and the maximum ($\max = 4 \times 10^{-4}$) d_{CH} allowed distance. Consequently, we applied our F1 and F2 to remove outliers and redundant PCs. Since we combined the raw PCs by their plane of symmetry (i.e., median plane), the generated data resembled the real PCs; hence, F1 identified no outliers (i.e., $n_{F1} = 0$). However, the combination process inevitably created redundant data, and F2 discarded $n_{F2} = 35$ PCs. Through the first DA step, we generated $n_f = 175$ final combined PCs.

Finally, we grouped the raw (i.e., 15 training PCs) and generated data (i.e., 15 mirrored and 175 final combined PCs) into an l-GAN1 dataset to train our iterative l-GAN model.

4.3 Iterative L-GAN1

4.3.1 Autoencoder 1. The encoder architecture follows the design principle described by Ref. [80]. Specifically, 1D convolutional layers with kernel size one and an increasing number of features. Max-pooling layers are used in the encoder network for spatial downsampling of input data [81]. In our implementation, the encoder is composed of five 1D convolutional layers with 128, 128, 256, 128, and 128 channels. The decoder is a fully connected neural network (FCN) with four layers of 128, 128, 256, and 6144 neurons. Both networks use batch normalization and ReLU activation functions. We trained our AE models by minimizing the combined loss function described in Eq. (2) with $\omega_{\text{CH}} = 100$ and $\omega_{\text{GFV}} = 30$. We used a batch size $b = 49$, Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.99$, and a learning rate $lr = 5 \times 10^{-4}$ for 10,000 iterations. To select the final AE, we evaluate our model performance (i.e., L_{AE}) with the validation dataset.

Then, we apply the selected AE1 encoder $E1$ to obtain the latent representation of the l-GAN1 data.

4.3.2 Latent Space Generative Adversarial Network 1. To train the l-GAN (i.e, generator and discriminator) models, we implemented the self-attention framework described in Refs. [6,69]. Both models were trained using a WGAN-GP adversarial loss with $\lambda = 10$, Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.99$, and $lr = 5 \times 10^{-4}$ for 10,000 iterations [70]. During this process, we use $b = 41$ (i.e., batch size) 32-dimensional seed $\mathbf{z}_s$, randomly sampled from a multivariate normal distribution with the initial $\boldsymbol{\mu} = \mathbf{0}$ and $\text{Cov} = I$.

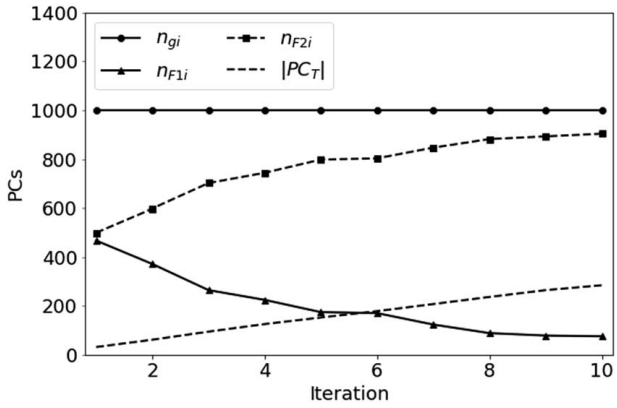

Fig. 7 Iterarive l-GAN results. n_{gi}, n_{F1i}, n_{F2i}, and $|PC_T|$ are the number of PCs generated, removed by F1, removed by F2, and accumulated per iteration

4.3.3 Iterative L-GAN1. We apply our iterative l-GAN1 $it_{max} = 10$ iterations and generate $n_{gi} = 1000$ PCs per iteration. After each iteration, we remove the outlier and redundant PCs by applying F1 and F2 filters and store the filtered PCs (n_{fi}) in an accumulator PC_T. Figure 7 shows the number of PCs removed by F1 and F2 (i.e., n_{F1i} and n_{F2i}) and the number of PCs ($|PC_T|$) accumulated over iterations. Since we update the seed **z** distribution after applying F1 in each iteration, we can observe that the number of outliers (i.e., n_{F1i}) decreases in each iteration. However, most of the remaining PCs were redundant, so the number of PCs removed by F2 increased in each step. In the end, we generated $|PC_T| = 285$ PCs; however, after removing the redundant PCs with F2, our dataset is reduced to $n_{fT} = 49$ PCs.

4.4 Hybrid RL-GAN

4.4.1 Reinforcement Learning Agent. To train our RL agent, we adopted the actor-critic method proposed by Sarmad et al. [6]. This network has four layers of 400, 400, 300, and 300 neurons, with ReLU activation for the first three layers and Tanh activation for the last layer. Similarly, the critic is an FCN with four layers of 400, 432, 300, and 300 neurons, with the ReLU activation function in the first three layers and no activation function in the last layer.

The training process of the agent is composed of two steps [6]. First, we collect experience using one sample at a time. Towards this end, the agent picks a seed **z** and we evaluate its performance using Eq. (5) with $\omega_{CH} = 100$, $\omega_{GFV} = 10$, and $\omega_D = 0.001$ [6]. In the second step, we train our actor-critic network using DDPG with $b = 100$ [6]. In this study, the total number of iterations was $t_{max} = 20,000$ with a starting time of $t_0 = 1000$. The state dimension is the GFV size (i.e., 128), while the action dimension is the number of elements in the seed **z** (i.e., 32).

4.4.2 Hybrid RL-GAN. We use the RL-GAN dataset (i.e., 15 raw, 15 mirrored, 175 combined, and 49 l-GAN PCs) to train our hybrid RL-GAN. We trained our AE2, l-GAN2, and RL Agent following the procedures described in Secs. 4.3.1, 4.3.2, and 4.4.1, respectively. To evaluate the completion performance, we use our testing samples (i.e., 15 PCs) to generate six incomplete datasets ($N = 5, 10, 15, \ldots, 30\%$ missing area), where N random locations are selected, and an area of 1% is removed at each location. We use the incomplete datasets to evaluate the hybrid RL-GAN performance against benchmark models PCN [55], CRN [60], VRC-Net [56], AE1 trained for the iterative l-GAN1, AE2, and RL-GAN.

Figures 8(*a*)–8(*f*) display the shape completion results for 5–30% missing area for an example sample in the testing dataset. The first column shows the raw PCs as the ground truth, while the second column shows the incomplete PCs. The other columns show the performance of the hybrid RL-GAN and benchmark methods.

Figure 8 corroborates that VRC-Net and CRN reconstructions miss the detailed PC structure, and only the general teeth shape is obtained. PCN-completed PCs are more uniformly distributed compared to the VRC-Net and CRN approaches. Compared with the benchmark models, the hybrid RL-GAN framework improves the accuracy of missing PC data completion. On the other hand, one can see that both AE2 and RL-GAN can complete the shape well when the missing area is small. However, the AE's performance degrades when the missing area is big (see Fig. 9), while RL-GAN is more stable when the missing area is increased. Hybrid RL-GAN selects the best completion between the AE2 and the RL-GAN outputs by evaluating their d_{CH} to the incomplete PC. The hybrid RL-GAN selection is marked in Fig. 8 with a black star.

Furthermore, we quantify the PC completion performance for the proposed and benchmark methods using d_{CH}. The smaller the distance, the better the completion. Figure 9 shows the average d_{CH} between the completed PCs and the ground truth PCs in the testing dataset. One can see that VRC-Net has the highest d_{CH}, followed by the CRN. This could happen because the amount of training PCs is not enough for these methods since they need massive datasets for training [56,60]. Although PCN generates better results than VRC-Net and CRN, our approach outperforms the state-of-the-art methods, as displayed in Fig. 9. By including the l-GAN1 PCs, we reduced the AE's completion error; hence AE2 outperforms AE1.

Similar to the previous study [6], the AEs' performance degrades drastically and becomes unstable as the percentage of missing data increases. This behavior can be caused by the high GFV variability

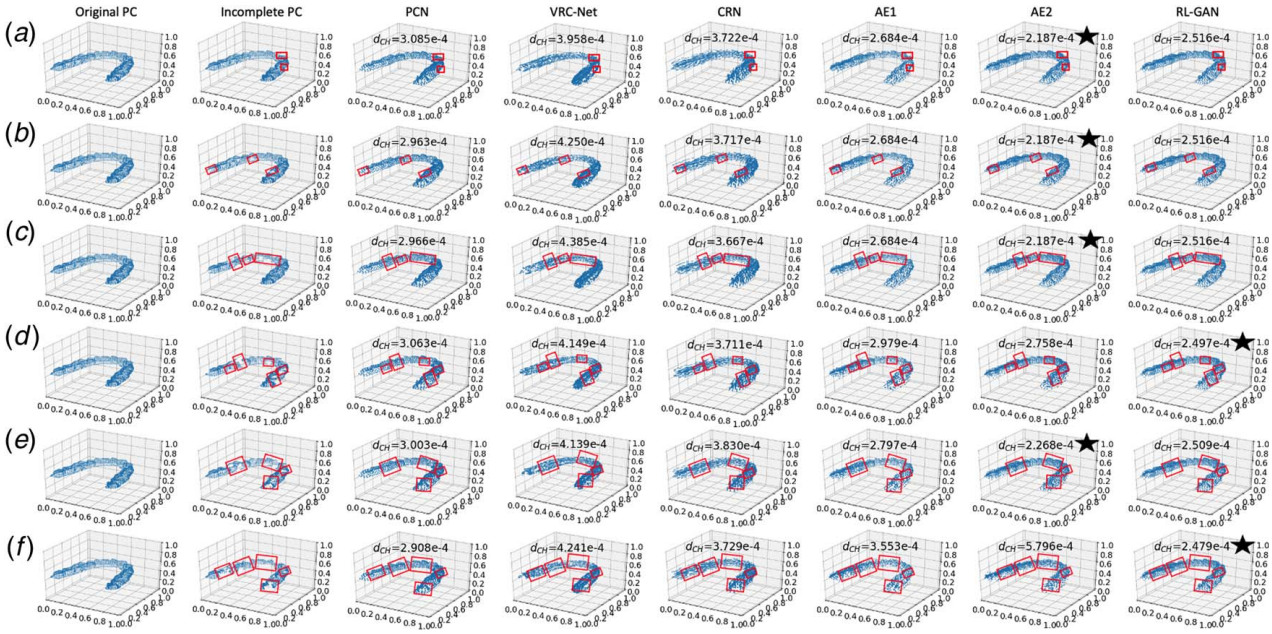

Fig. 8 Shape completion results for a testing PC under various methods, where the hybrid RL-GAN results are marked with a star: (*a*) 5% missing area, (*b*) 10% missing area, (*c*) 15% missing area, (*d*) 20% missing area, (*e*) 25% missing area, and (*f*) 30% missing area

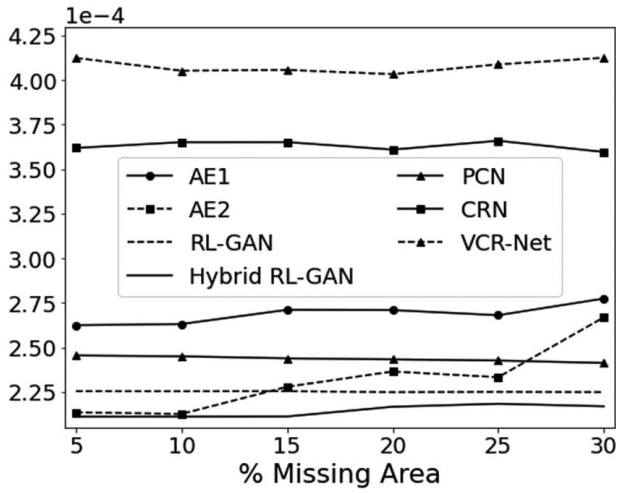

Fig. 9 Comparison of average chamfer distance to ground truth

when the missing area is getting bigger. On the other hand, the RL-GAN's completion error is much more stable, yet the generated RL-GAN PCs may fail to preserve local details. Consequently, by selecting the best completion, the hybrid approach addresses the AE and RL-GAN problems and reduces the d_{CH} to the ground truth.

4.5 Ablation Study. In the ablation study, we remove one module at a time and obtained five ablated systems. Then, we compute the %IncCH using Eq. (6). In all cases, we evaluate the ablated systems and the original system's performance using the validation dataset.

For the first ablated system (AS1), we remove the PC-Combination module. After mirroring our dataset, we directly run the iterative l-GAN1 (Sec. 4.3.3). However, since 30 PCs is not enough to train a suitable AE, the iterative l-GAN1 could not produce any useful data. Therefore, we use 30 PCs to train RL-GAN (i.e., l-GAN2 and RL Agent) and reconstruct the incomplete data following the procedure described in Secs. 4.4.1 and 4.4.2, respectively. As shown in Table 1, when we remove the PC Combination, the average chamfer distance for all the missing areas increases by 233.34%.

In the second ablated system (AS2), we do not use the filters that control the data augmentation. Thus, we perform PC-Combination (Sec. 4.2) and obtain 210 PCs. Without the filters, we cannot control the l-GAN1's seed $\mathbf{z}$ distribution, so we generate 1000 PCs in a single step. Then, we follow Secs. 4.4.1 and 4.4.2 to train the remaining modules and reconstruct the incomplete PCs accordingly. Table 1 shows that when we remove the filters, the average chamfer distance for all the missing areas increases by 38.34%.

For the third ablated system (AS3), we remove the iterative l-GAN1. Therefore, we augment our data using PC-Combination (Sec. 4.2). Then, we use our data to train RL-GAN (i.e., l-GAN2 and RL Agent) following Sec. 4.4.1. Finally, we process the incomplete PCs and select the best reconstruction between the RL-GAN2 and the AE2 as described in Sec. 4.4.2. Table 1 shows that without the iterative l-GAN1, the average chamfer distance for all the missing areas increases by 43.20%.

In the fourth ablated system (AS4), we remove the RL-GAN module. Therefore, after augmenting our data following Secs. 4.2 and 4.3.3, finally, we process the incomplete data as described in Sec. 4.4.2. Without the RL-GAN model, the hybrid RL-GAN always chooses the AE2 reconstruction. As shown in Table 1, when we remove the RL-GAN module, the average chamfer distance increases by 13.42%.

For the final ablated system (AS5), we remove the hybrid RL-GAN module. We start by augmenting our dataset following Secs. 4.2 and 4.3.3. After that, we train RL-GAN (i.e., l-GAN2 and RL Agent) as described in Sec. 4.4.1 and use RL-GAN to reconstruct the incomplete data. Table 1 shows that without the iterative l-GAN1, the average chamfer distance for all the missing areas increases by 5.34%.

As shown in Table 1, the ablation study demonstrated that all the modules in the proposed framework are helpful in PC completion.

Table 1 Ablation study results

Missing area (%)	AS1 (%)	AS2 (%)	AS3 (%)	AS4 (%)	AS5 (%)
5	237.33	39.44	28.20	6.22	6.62
10	235.96	38.96	69.93	16.85	6.15
15	230.64	37.43	32.68	17.85	4.43
20	235.14	38.76	30.59	12.22	6.03
25	223.54	34.23	35.64	8.65	2.17
30	237.42	41.19	62.14	18.75	6.61
Average	233.34	38.34	43.20	13.42	5.34

Note: AS1, remove PC-Combination; AS2, remove filters; AS3, remove iterative l-GAN1; AS4, remove RL-GAN; AS5, remove hybrid RL-GAN.

5 Conclusion and Future Work

3D scanned PCs are more and more widely used in orthodontics applications. One critical issue of the PCs is the missing areas due to factors such as limited viewing angles and occlusions. In this paper, we proposed a systematic framework for 3D PC completion with limited data. The framework consists of (1) a two-step data augmentation technique based on the bilateral symmetry of human teeth and an iterative GAN, and (2) a hybrid RL-GAN method that selects the best completion from AE and RL-GAN. Through the demonstration in the 3D teeth mold PCs, the proposed framework can achieve accurate PC completion. The proposed PC completion framework can be applied to other PC completion scenarios with limited data and complex shapes.

In the future, based on the completed 3D PCs, we will explore the forecast of the teeth alignment of a patient over time. This forecast will help the dentists in the monitoring and planning of the alignment procedures. Another direction is to consider the biographic information, habit, etc. for the personalized PC completion and forecast.

Acknowledgment

This work is partially supported by the NSF Grant No. FM-2134409 and Sustainable Manufacturing and Advanced Robotics Technologies, Community of Excellence (SMART CoE) at the State University of New York at Buffalo.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Weir, T., 2017, "Clear Aligners in Orthodontic Treatment," Aust. Dent. J., **62**(S1), pp. 58–62.

[2] Tartaglia, G. M., Mapelli, A., Maspero, C., Santaniello, T., Serafin, M., Farronato, M., and Caprioglio, A., 2021, "Direct 3D Printing of Clear Orthodontic Aligners: Current State and Future Possibilities," Materials, **14**(7), p. 1799.

[3] Logozzo, S., Zanetti, E. M., Franceschini, G., Kilpelä, A., and Mäkynen, A., 2014, "Recent Advances in Dental Optics—Part I: 3D Intraoral Scanners for Restorative Dentistry," Opt. Lasers Eng., **54**, pp. 203–221.
[4] Huang, J., Sun, H., Kwok, T.-H., Zhou, C., and Xu, W., 2020, "Geometric Deep Learning for Shape Correspondence in Mass Customization by Three-Dimensional Printing," ASME J. Manuf. Sci. Eng., **142**(6), p. 061003.
[5] Fischer, A., 2011, "Engineering-Oriented Geometry Methods for Modeling and Analyzing Scanned Data," J. Comput. Inf. Sci. Eng., **11**(2), p. 031006.
[6] Sarmad, M., Lee, H. J., and Kim, Y. M., 2019, "Rl-Gan-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, June 16–20, pp. 5898–5907.
[7] Gurumurthy, S., and Agrawal, S., 2019, "High Fidelity Semantic Shape Completion for Point Clouds Using Latent Optimization," 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, Jan. 7–11, IEEE, pp. 1099–1108.
[8] Vučina, D., Milas, Z., and Pehnec, I., 2012, "Reverse Shape Synthesis of the Hydropump Volute Using Stereo-photogrammetry, Parameterization, and Geometric Modeling," ASME J. Comput. Inf. Sci. Eng., **12**(2), p. 021001.
[9] Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L., 2018, "Learning Representations and Generative Models for 3D Point Clouds," International Conference on Machine Learning, Vancouver, BC, Canada, Apr. 30–May 3, PMLR, pp. 40–49.
[10] Halimi, O., Imanuel, I., Litany, O., Trappolini, G., Rodolà, E., Guibas, L., and Kimmel, R., 2020, "The Whole is Greater Than the Sum of its Nonrigid Parts," preprint arXiv:2001.09650.
[11] Litany, O., Bronstein, A., Bronstein, M., and Makadia, A., 2018, "Deformable Shape Completion With Graph Convolutional Autoencoders," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, June 18–22, pp. 1886–1895.
[12] Maniadis, I., Solachidis, V., Vretos, N., and Daras, P., 2020, "Data Augmentation Using GANs for 3D Applications," *Recent Advances in 3D Imaging, Modeling, and Reconstruction*, A. Voulodimos and A. Doulamis, eds., IGI Global, pp. 229–269.
[13] Wu, E., Wu, K., Cox, D., and Lotter, W., 2018, "Conditional Infilling GANs for Data Augmentation in Mammogram Classification," *Image Analysis for Moving Organ, Breast, and Thoracic Images*, D. Stoyanov, Z. Taylor, B. Kainz, G. Maicas, R. R. Beichel, A. Martel, L. Maier-Hein, et al., eds., Springer, pp. 98–106.
[14] Chen, X.-G., Zhang, W., Yang, X., Li, C., and Chen, H., 2021, "ACP-DA: Improving the Prediction of Anticancer Peptides Using Data Augmentation," Front. Genet., **12**, p. 1131.
[15] Li, W., Mo, W., Zhang, X., Squiers, J. J., Lu, Y., Sellke, E. W., Fan, W., DiMaio, J. M., and Thatcher, J. E., 2015, "Outlier Detection and Removal Improves Accuracy of Machine Learning Approach to Multispectral Burn Diagnostic Imaging," J. Biomed. Opt., **20**(12), p. 121305.
[16] Tanaka, F. H. K. d. S., and Aranha, C., 2019, "Data Augmentation Using GANs," preprint arXiv:1904.09135.
[17] Li, M., and McComb, C., 2022, "Using Physics-Informed Generative Adversarial Networks to Perform Super-Resolution for Multiphase Fluid Simulations," ASME J. Comput. Inf. Sci. Eng., **22**(4), p. 044501.
[18] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Nets," Communications of the ACM, **63**(11), pp. 139–144.
[19] Ji, H., and Jin, Y., 2022, "Knowledge Acquisition of Self-Organizing Systems With Deep Multiagent Reinforcement Learning," ASME J. Comput. Inf. Sci. Eng., **22**(2), p. 021010.
[20] López, C. E., Cunningham, J., Ashour, O., and Tucker, C. S., 2020, "Deep Reinforcement Learning for Procedural Content Generation of 3d Virtual Environments," ASME J. Comput. Inf. Sci. Eng., **20**(5), p. 051005.
[21] Ororbia, M. E., and Warn, G. P., 2022, "Design Synthesis Through a Markov Decision Process and Reinforcement Learning Framework," ASME J. Comput. Inf. Sci. Eng., **22**(2), p. 021002.
[22] Du, L., 2020, "How Much Deep Learning Does Neural Style Transfer Really Need? An Ablation Study," Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass, CO, Mar. 1–5, pp. 3150–3159.
[23] Punj, A., Bompolaki, D., and Garaicoa, J., 2017, "Dental Impression Materials and Techniques," Dental Clinics, **61**(4), pp. 779–796.
[24] Megremis, S., Tiba, A., and Vogt, K., 2012, "An Evaluation of Eight Elastomeric Occlusal Registration Materials," J. Am. Dent. Assoc., **143**(12), pp. 1358–1360.
[25] Hellmann, D., Etz, E., Giannakopoulos, N., Rammelsberg, P., Schmitter, M., and Schindler, H., 2013, "Accuracy of Transfer of Bite Recording to Simulated Prosthetic Reconstructions," Clin. Oral Investig., **17**(1), pp. 259–267.
[26] Amuk, N. G., Karsli, E., and Kurt, G., 2019, "Comparison of Dental Measurements Between Conventional Plaster Models, Digital Models Obtained by Impression Scanning and Plaster Model Scanning," Int. Orthodont., **17**(1), pp. 151–158.
[27] Dua, P., Gupta, S., Ramachandran, S., and Sandhu, H., 2007, "Evaluation of Four Elastomeric Interocclusal Recording Materials," Med. J. Armed Forces India, **63**(3), pp. 237–240.
[28] Wieckiewicz, M., Grychowska, N., Zietek, M., and Wieckiewicz, W., 2016, "Evaluation of the Elastic Properties of Thirteen Silicone Interocclusal Recording Materials," BioMed. Res. Int., **2016**, p. 7456046.
[29] Runkel, C., Güth, J.-F., Erdelt, K., and Keul, C., 2020, "Digital Impressions in Dentistry–Accuracy of Impression Digitalisation by Desktop Scanners," Clin. Oral Investig., **24**(3), pp. 1249–1257.
[30] Marques, S., Ribeiro, P., Falcão, C., Lemos, B. F., Ríos-Carrasco, B., Ríos-Santos, J. V., and Herrero-Climent, M., 2021, "Digital Impressions in Implant Dentistry: A Literature Review," Int. J. Environ. Res. Public Health, **18**(3), p. 1020.
[31] Latos, I., and Janóczki, M., 2011, "Stability Investigations of Automatic X-Ray Inspection Systems," Soldering Surface Mount Technol., **23**(3), pp. 91–103.
[32] Rokicki, P., Budzik, G., Kubiak, K., Dziubek, T., Zaborniak, M., Kozik, B., Bernaczek, J., Przeszlowski, L., and Nowotnik, A., 2016, "The Assessment of Geometric Accuracy of Aircraft Engine Blades With the Use of an Optical Coordinate Scanner," Aircr. Eng. Aerosp. Technol.: Int. J., **88**(3), pp. 374–381.
[33] Keeling-Roberts, C. S., 2002, "Use of a Proforma for Reporting Staging CT Scans of the Thorax," Br. J. Clin. Governance, **7**(4), pp. 273–278.
[34] Kamegawa, M., Nakamura, M., Kitahara, K., Ohtomo, H., Hasegawa, T., Nakakura, T., and Tsutsumi, S., 2008, "3D Morphological Assessment of Occlusal Treatment by Measuring Dental Casts With a Micro-focus X-Ray CT," J. Oral Rehabil., **35**(5), pp. 382–389.
[35] Kang, S.-H., Lee, J.-W., Lim, S.-H., Kim, Y.-H., and Kim, M.-K., 2014, "Dental Image Replacement on Cone Beam Computed Tomography With Three-Dimensional Optical Scanning of a Dental Cast, Occlusal Bite, or Bite Tray Impression," Int. J. Oral Maxillofacial Surg., **43**(10), pp. 1293–1301.
[36] Nilsson, J., Richards, R. G., Thor, A., and Kamer, L., 2016, "Virtual Bite Registration Using Intraoral Digital Scanning, CT and CBCT: In Vitro Evaluation of a New Method and Its Implication for Orthognathic Surgery," J. Cranio-Maxillofacial Surg., **44**(9), pp. 1194–1200.
[37] Swennen, G. R., Mommaerts, M. Y., Abeloos, J., De Clercq, C., Lamoral, P., Neyt, N., Casselman, J., and Schutyser, F., 2007, "The Use of a Wax Bite Wafer and a Double Computed Tomography Scan Procedure to Obtain a Three-Dimensional Augmented Virtual Skull Model," J. Craniofacial Surg., **18**(3), pp. 533–539.
[38] Richert, R., Goujat, A., Venet, L., Viguie, G., Viennot, S., Robinson, P., Farges, J.-C., Fages, M., and Ducret, M., 2017, "Intraoral Scanner Technologies: A Review to Make a Successful Impression," J. Healthcare Eng., **2017**, p. 8427595.
[39] Ireland, A. J., McNamara, C., Clover, M., House, K., Wenger, N., Barbour, M. E., Alemzadeh, K., Zhang, L., and Sandy, J. R., 2008, "3D Surface Imaging in Dentistry—What We Are Looking at," Br. Dental J., **205**(7), pp. 387–392.
[40] Pradíes, G., Ferreiroa, A., Özcan, M., Giménez, B., and Martínez-Rus, F., 2014, "Using Stereophotogrammetric Technology for Obtaining Intraoral Digital Impressions of Implants," J. Am. Dental Assoc., **145**(4), pp. 338–344.
[41] Giménez, B., Özcan, M., Martínez-Rus, F., and Pradíes, G., 2015, "Accuracy of a Digital Impression System Based on Active Wavefront Sampling Technology for Implants Considering Operator Experience, Implant Angulation, and Depth," Clin. Implant Dent. Rel. Res., **17**(S1), pp. e54–e64.
[42] Taneva, E., Kusnoto, B., and Evans, C. A., 2015, "3D Scanning, Imaging, and Printing in Orthodontics," Issues Contemp. Orthodont., **148**(5), pp. 862–867.
[43] Bello, S. A., Yu, S., Wang, C., Adam, J. M., and Li, J., 2020, "Deep Learning on 3D Point Clouds," Remote Sens., **12**(11), p. 1729.
[44] Fei, B., Yang, W., Chen, W., Li, Z., Li, Y., Ma, T., Hu, X., and Ma, L., 2022, "Comprehensive Review of Deep Learning-Based 3D Point Clouds Completion Processing and Analysis," preprint arXiv:2203.03311.
[45] Hou, W., Chan, T., and Ding, M., 2012, "Denoising Point Cloud," Inverse Probl. Sci. Eng., **20**(3), pp. 287–298.
[46] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., 1996, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, Aug. 2–4, Vol. 96, pp. 226–231.
[47] Zhao, W., Gao, S., and Lin, H., 2007, "A Robust Hole-Filling Algorithm for Triangular Mesh," Visual Comput., **23**(12), pp. 987–997.
[48] Kriegel, H.-P., and Pfeifle, M., 2005, "Density-Based Clustering of Uncertain Data," Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, Aug. 21–24, pp. 672–677.
[49] Davis, J., Marschner, S. R., Garr, M., and Levoy, M., 2002, "Filling Holes in Complex Surfaces Using Volumetric Diffusion," Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission, Padua, Italy, June 19–21, IEEE, pp. 428–441.
[50] Duan, Y., Yang, C., Chen, H., Yan, W., and Li, H., 2021, "Low-Complexity Point Cloud Denoising for Lidar by PCA-Based Dimension Reduction," Opt. Commun., **482**, p. 126567.
[51] Sarkar, K., Bernard, F., Varanasi, K., Theobalt, C., and Stricker, D., 2018, "Structured Low-Rank Matrix Factorization for Point-Cloud Denoising," International Conference on 3D Vision (3DV), Verona, Italy, Sept. 5–8, IEEE, pp. 444–453.
[52] Gandler, G. Z., Ek, C. H., Björkman, M., Stolkin, R., and Bekiroglu, Y., 2020, "Object Shape Estimation and Modeling, Based on Sparse Gaussian Process Implicit Surfaces, Combining Visual Data and Tactile Exploration," Rob. Auton. Syst., **126**, p. 103433.
[53] Zhang, F., Zhang, C., Yang, H., and Zhao, L., 2019, "Point Cloud Denoising With Principal Component Analysis and a Novel Bilateral Filter.," Traitement du Signal, **36**(5), pp. 393–398.
[54] Zhou, L., Sun, G., Li, Y., Li, W., and Su, Z., 2022, "Point Cloud Denoising Review: From Classical to Deep Learning-Based Approaches," Graph. Models, **121**, p. 101140.
[55] Yuan, W., Khot, T., Held, D., Mertz, C., and Hebert, M., 2018, "PCN: Point Completion Network," 2018 International Conference on 3D Vision (3DV), Verona, Italy, Sept. 5–8, IEEE, pp. 728–737.
[56] Pan, L., Chen, X., Cai, Z., Zhang, J., Zhao, H., Yi, S., and Liu, Z., 2021, "Variational Relational Point Completion Network," Proceedings of the IEEE/

CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, June 20–25, pp. 8524–8533.
[57] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J., 2016, "Volumetric and Multi-View CNNS For Object Classification on 3D Data," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, June 27–30, pp. 5648–5656.
[58] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E., 2015, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, Dec. 7–13, pp. 945–953.
[59] Zong, D., Sun, S., and Zhao, J., 2021, "ASHF-NET: Adaptive Sampling and Hierarchical Folding Network for Robust Point Cloud Completion," Proceedings of the AAAI Conference on Artificial Intelligence, Held Virtually, Feb. 2–9, Vol. 35, pp. 3625–3632.
[60] Wang, X., Ang, M. H., and Lee, G. H., 2021, "Cascaded Refinement Network for Point Cloud Completion With Self-Supervision," IEEE Trans. Pattern Anal. Mach. Intell., **44**(11), pp. 8139–8150.
[61] Huang, Z., Yu, Y., Xu, J., Ni, F., and Le, X., 2020, "PF-NET: Point Fractal Network for 3D Point Cloud Completion," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, June 13–19, pp. 7662–7670.
[62] Warey, A., Raul, V., Kaushik, S., Han, T., and Chakravarty, R., 2023, "Generative Inverse Design of Aerodynamic Shapes Using Conditional Invertible Neural Networks," ASME J. Comput. Inf. Sci. Eng., **23**(3), p. 031006.
[63] Li, R., Li, X., Heng, P.-A., and Fu, C.-W., 2020, "Pointaugment: An Auto-Augmentation Framework for Point Cloud Classification," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, June 13–19, pp. 6378–6387.
[64] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., and Gupta, S. K., 2021, "Image-Based Surface Defect Detection Using Deep Learning: A Review," ASME J. Comput. Inf. Sci. Eng., **21**(4), p. 040801.
[65] Dorland, W. A. N., 1925, *Dorland's Illustrated Medical Dictionary*, WB Saunders, Zanesville, OH.
[66] He, L., Wang, X., and Zhang, H., 2016, "M2DP: A Novel 3D Point Cloud Descriptor and Its Application in Loop Closure Detection," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea, Oct. 9–14, IEEE,pp. 231–237.
[67] Wu, T., Pan, L., Zhang, J., Wang, T., Liu, Z., and Lin, D., 2021, "Balanced Chamfer Distance as a Comprehensive Metric for Point Cloud Completion," Adv. Neural Inf. Process. Syst., **34**, pp. 29088–29100.
[68] Kingma, D. P., and Ba, J., 2014, "ADAM: A Method for Stochastic Optimization," preprint arXiv:1412.6980.
[69] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A., 2019, "Self-Attention Generative Adversarial Networks," International Conference on Machine Learning, PMLR, Beach, CA, June 9–15, PMLR, pp. 7354–7363.
[70] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C., 2017, "Improved Training of Wasserstein GANs," Annual Conference on Neural Information Processing Systems, Long Beach, CA, Dec. 4–9.
[71] Kaelbling, L. P., Littman, M. L., and Moore, A. W., 1996, "Reinforcement Learning: A Survey," J. Artif. Intell. Res., **4**, pp. 237–285.
[72] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., 2015, "Continuous Control With Deep Reinforcement Learning," preprint arXiv:1509.02971.
[73] Sheikholeslami, S., Meister, M., Wang, T., Payberah, A. H., Vlassov, V., and Dowling, J., 2021, "Autoablation: Automated Parallel Ablation Studies for Deep Learning," Proceedings of the 1st Workshop on Machine Learning and Systems, Virtual, Apr. 26, pp. 55–61.
[74] Hamilton, N., Dunlap, K., Johnson, T. T., and Hobbs, K. L., 2022, "Ablation Study of How Run Time Assurance Impacts the Training and Performance of Reinforcement Learning Agents," preprint arXiv:2207.04117.
[75] Meyes, R., Lu, M., de Puiseau, C. W., and Meisen, T., 2019, "Ablation Studies in Artificial Neural Networks," preprint arXiv:1901.08644.
[76] Zhou, Q.-Y., Park, J., and Koltun, V., 2018, "Open3D: A Modern Library for 3D Data Processing," preprint arXiv:1801.09847.
[77] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., and Killeen, T., 2019, "Pytorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver Canada, Dec. 8–14.
[78] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., and Gkioxari, G., 2020, "Accelerating 3D Deep Learning With Pytorch3D," arXiv:2007.08501.
[79] Ding, L.-j., Dai, S.-g., and Mu, P.-a., 2016, "CAD-Based Path Planning for 3D Laser Scanning of Complex Surface," Procedia Comput. Sci., **92**, pp. 526–535.
[80] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., 2017, "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, July 21–26, pp. 652–660.
[81] Sofi, A. R., and Ravani, B., 2023, "Sub-Second Prediction of the Heatmap of Powder-Beds in Additive Manufacturing Using Deep Encoder–Decoder Convolutional Neural Networks," ASME J. Comput. Inf. Sci. Eng., **23**(2), p. 021008.