2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# Contrastive Learning on 3D Point Clouds for Robotic Geometric Defect Detection

Alexander Tarvo[1] and Yusen Wan[2] and Xu Chen[3]

***Abstract*— Robotic quality inspection is emerging as a key enabler in intelligent manufacturing, allowing robots to transcend human limitations in endurance, consistency, and access to complex structures. By detecting subtle defects with speed and precision, robotic inspection enhances efficiency while elevating production quality. While most existing approaches emphasize 2D image-based surface defect detection, they often overlook geometric defects, which are more prevalent and challenging in industrial inspection. To overcome this gap, we formulate geometric defect detection as anomaly detection in 3D point clouds and propose a novel framework that integrates contrastive learning with spatially aware comparisons of local geometries. Specifically, we partition point cloud surfaces into patches and employ contrastive learning to train a neural network-based feature extractor capable of capturing rich geometric representations. An anomaly detection algorithm is then introduced to identify defects by comparing patch-level features in a spatially consistent manner. Evaluated on the recent Real3D-AD benchmark, our method achieves a mean area under the ROC curve of 0.901, establishing a new state of the art and demonstrating the potential of robotic inspection systems to move beyond human limitations in detecting subtle geometric anomalies.**

## I. Introduction

Robotic quality inspection is rapidly advancing in intelligent manufacturing, harnessing emergent artificial intelligence to detect defects in complex targets with precision and scalability that surpass manual inspection [1], [2], [3], [4]. The typical procedure of robotic quality inspection involves mounting a detector onto the robotic arm, which then maneuvers the detector to capture information (such as point clouds or images) of the target object. The collected data is subsequently analyzed to identify defect locations. Two fundamental, inseparable components exist in this process: data collection and defect detection. In the latter, automated defect detection is usually formulated as an unsupervised anomaly detection problem: given a "test" part to be examined, verify whether it matches an expected "good", anomaly-free part. The good part could be established from scans of high-quality reference parts or from CAD models.

One popular approach for anomaly detection is patch-based algorithms, which divide an image of "good" objects into patches, extract a vector of descriptive features for each patch, and store these features in a global memory bank. Similarly, they extract features from "test" objects and compare these with features in the memory bank. If features extracted from the "test" object's patch differ significantly from features extracted from "good" objects' patches, the "test" object is deemed defective.

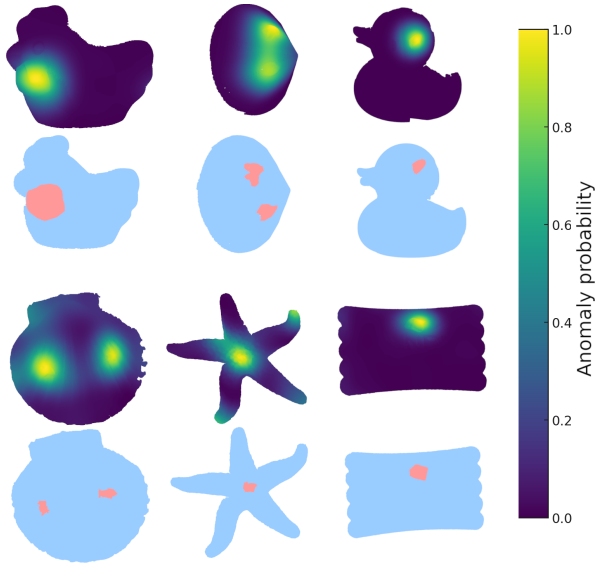


Fig. 1. Anomaly localization by COSARAD on Real3D-AD dataset. Color intensity denotes pointwise anomaly probabilities. Ground-truth point clouds are shown in light blue, with anomaly masks highlighted in red.

While existing approaches have made major advances in 2D image-based *surface* detection, inherent shortcomings limit their adoption for detecting critical volumetric and geometric defects such as structural deformations or missing sub-assemblies. We can still represent local geometry as patches of 3D point clouds. However, storing these patches in a single monolithic memory bank leads to the challenge of spatial indifference: a structural feature that is perfectly normal on the "base" of a part must be flagged as a defect if it appears on the "top". Yet, a global bank stores these spatially distant patches as indistinguishable feature vectors, making such detection impossible. This lack of coordinate-level accountability results in a system that "knows" what a geometric feature looks like but "forgets" where it belongs, leading to inconsistent detection and high false-positive rates.

To address this critical spatial ambiguity, we propose COSARAD – a **co**ntrastive **s**patially-**a**wa**r**e **a**nomaly **d**etection method. COSARAD enforces topological integrity by replacing the monolithic memory bank with a multitude of small, location-specific banks. By strictly comparing lo-

[1]A. Tarvo is with MACS lab at University of Washington, 3920 E. Stevens Way NE Seattle, WA 98195, USA alexta@uw.edu
[2]Y. Wan is with MACS lab at University of Washington, 3920 E. Stevens Way NE Seattle, WA 98195, USA yusenwan@uw.edu
[3]X. Chen is with the faculty of Mechanical Engineering at University of Washington, 3920 E. Stevens Way NE Seattle, WA 98195, USA chx@uw.edu

cal geometries only within their shared coordinate frame, COSARAD ensures that robotic inspection is not just geometrically precise, but spatially coherent. This is achieved via a two-step, training-and-inference process. During training, we generate triplets of surface patches from labeled point clouds and train a contrastive feature extractor that learns to distinguish anomalous patches from defect-free ones. During inference, we compare a "test" object against a set of anomaly-free "template" objects. We register all objects in a common coordinate frame and store template features in a multitude of location-specific memory banks, each strictly corresponding to a single location in that standard frame. A patch from the "test" object is then compared only against the corresponding local bank. Collectively, these technical designs lead to the following main contributions:

**Contrastive feature extractor.** To the best of our knowledge, we are the first to repurpose contrastive learning for 3D anomaly detection. Using a triplet loss, a type of contrastive loss, we develop a novel deep learning-based feature extractor that produces highly informative representations for anomaly detection.

**Spatially aware patch comparison.** We eliminate spatial ambiguity by registering template and test objects in a common coordinate frame. With this framework, we compare only patches that originate strictly from the same location within that standard frame.

Our COSARAD algorithm is simple and lightweight; it can be trained on a consumer GPU – important for the targeted manufacturing application. At the same time, COSARAD delivers superior accuracy on both Anomaly-ShapeNet and Real3D-AD datasets.

## II. Related Work

**Contrastive learning** extracts meaningful representations by contrasting positive and negative pairs of instances. The triplet loss, one of its foundational techniques, was introduced in the FaceNet architecture [5]. The InfoNCE loss [6] advanced the field by maximizing mutual information between different views of the data. Building on this, SimCLR [7] established a simple yet effective framework for contrastive learning, operating on unlabeled data through strong data augmentation and large batch sizes. MoCo [8] addressed the scalability issue by introducing a momentum encoder and a memory queue. CLIP [9], which takes advantage of multimodal features to learn high-quality visual embeddings, has been successfully applied to understand 3D point clouds (PCDs) [10].

Teacher-student architectures like DINO [11] are similar to contrastive learning in their architecture and reliance on data augmentation to learn high-quality representations. Here, weights of the student network are updated through backpropagation, while the teacher's weights are updated as an exponential moving average of the student's. M3DM [12] and CFM [13] models achieve strong results on the MVTec 3D-AD dataset [14] by combining a DINO-like architecture to learn RGB features and a traditional PointMAE backbone to learn 3D representations.

Wang et al. [15] employed contrastive learning on point clouds to estimate surface normals. However, to the best of our knowledge, we are the first to apply contrastive learning – specifically the triplet loss – to 3D anomaly detection. By learning the most informative features for anomaly detection, rather than relying on handcrafted features or pre-trained backbones, we significantly improve the anomaly detector's accuracy.

**3D defect detection** methods can be broadly categorized into reconstruction-based and patch-based (also known as feature- or embedding-based) methods. *Reconstruction-based methods*, such as R3D-AD [16] or IMRNet [17], train a model to reproduce "good" point clouds; when applied to a defective input, the model fails to reconstruct the anomalous regions, yielding high reconstruction loss and hence recognition of the defects. Knowledge distillation techniques [18], [19] similarly leverage teacher-student architectures.. The student network is trained to mimic a pre-trained teacher network on normal data, but fails to reconstruct anomalous shapes, leading to the capability of defect detection.

*Patch-based models* learn on embeddings (i.e. features) of point clouds. They assume that in the feature space, normal parts cluster together and anomalies appear as outliers. 3D-ST [20] uses a student-teacher model to learn representative patch features and uses the L2 loss between the outputs of a student and a teacher as an anomaly score. M3DM [12] combines 2D and 3D patches to learn rich representations, exceeding 0.90 AUC on MVTec-3D.

Building on the PatchCore [21] 2D algorithm, patch-based models extract features from regions of good and anomalous parts, store them into a memory bank, and match them using the min-max selection algorithm. Methods that use high-quality handcrafted features, such as BTF [22] or Gabor Filters [23], can be as accurate as deep learning models. Simple3D [24] used handcrafted multiscale features, reaching mean object-level area under the ROC curve (o-AUROC) 0.80 on Real3D-AD and 0.86 on Anomaly-ShapeNet.

Liu et al. [25] introduced the Real3D-AD dataset and found that the accuracy of PatchCore, BTF, and M3DM is inconsistent across object types. They proposed a first registration-based anomaly detection algorithm, Reg3D-AD. Reg3D-AD runs PatchCore separately on the spatially aligned raw point coordinates and unaligned patch representations, and returns a mean of their anomaly scores. PointCore [26] extends Reg3D-AD, assuming that patch features form a locally smooth field in the original 3D space. PointCore computes an anomaly score as a distance between features of a test patch and a linear interpolation of features for 3 spatially closest template patches. PointCore reaches o-AUROC of 0.83 on the Real3D-AD dataset, establishing a new state-of-the-art.

Recent works, such as Reg3D-AD and PointCore, register the test and template PCDs, but do not fully align patches before comparing their features. We implement a strict spatially-aware patch comparison using location-specific memory banks and demonstrate that strict spatial alignment substantially improves both object-level and point-

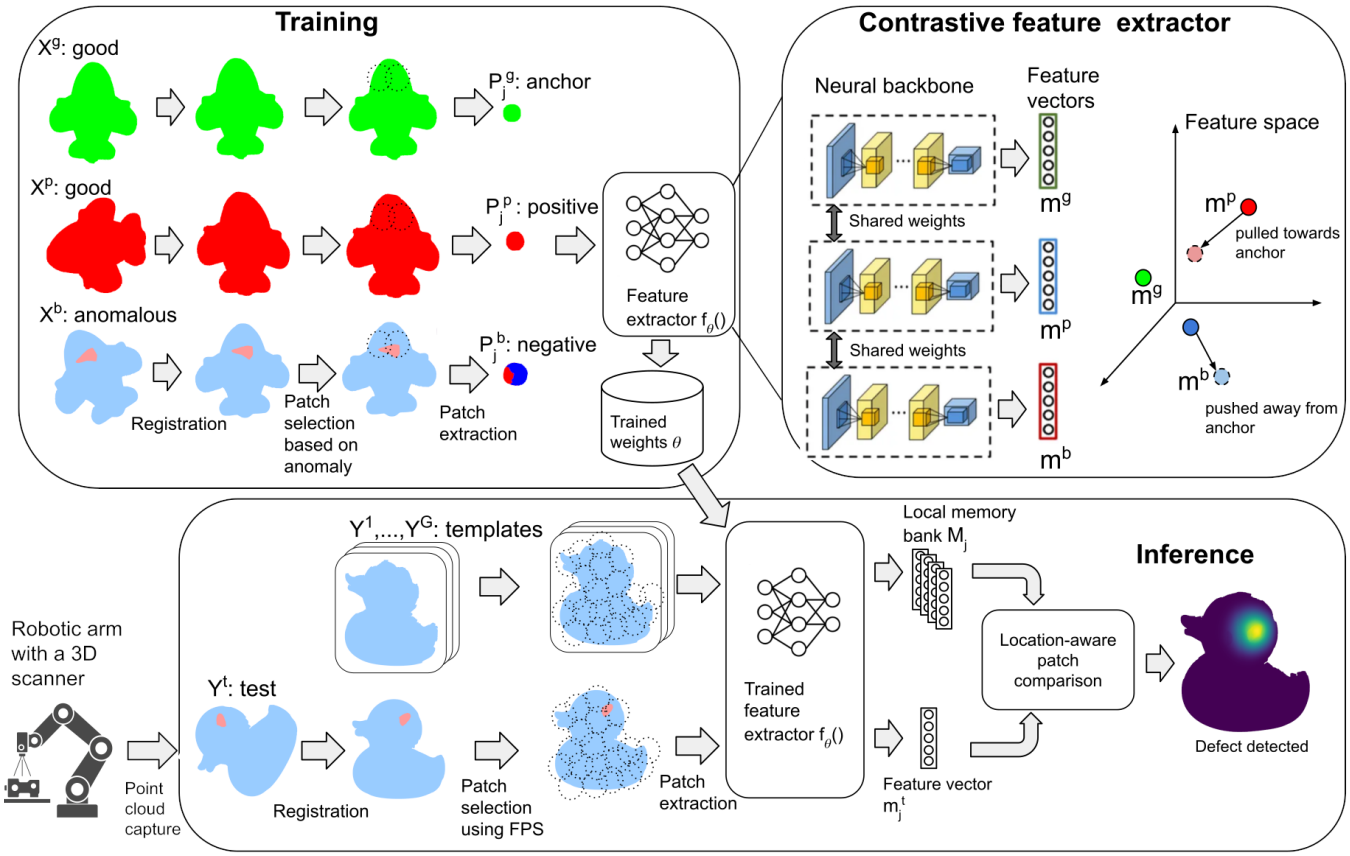


Fig. 2. The COSARAD pipeline for 3D geometric defect detection. Training: a contrastive extractor learns discriminative patch features from labeled point clouds. Inference: a test object is compared against templates via spatially aware, location-specific memory banks.

level detection accuracy. We compare a test object against multiple templates, thereby implicitly accounting for manufacturing tolerances and making our algorithm robust to measurement noise.

## III. Spatial-Aware Defect Detection

Figure 2 illustrates the proposed COSARAD pipeline of 3D anomaly detection. The input to the training stage is object classes – e.g., a shelf, a screw, a microphone – and their PCDs. For a given object class, $X^1, \cdots, X^N$ denote anomaly-free, "good" PCDs, and $X^{N+1}, \cdots, X^M$ represent anomalous, "bad" PCDs. Each PCD $X$ is an unordered set of $X_n$ points $X = \{x_i\}_{i=1}^{X_n}, x_i \in \mathcal{R}^3$. For each anomalous PCD $X^{N+1}, \cdots, X^M$ we build anomaly masks – i.e. indices of all anomalous points. We split these PCDs into patches. A *patch* $P_j$ is a spherical region of PCDs with a fixed radius $R$ and a central location $c_j \in \mathcal{R}^3$, representing the geometry of an object in the vicinity of the center point $c_j$.

The proposed method is based on the following principle. A "bad" patch $P_j^b$ centered at the point $c_j$ within an anomalous region of a PCD $X^b$, $b \in [N+1, M]$, should differ from a "good" patch $P_j^g$ extracted from the *same* location $c_j$ of an anomaly-free PCD $X^g$, $g \in [1, N]$. A "positive" patch $P_j^p$ from the same location of another anomaly-free PCD $X^p$, $p \in [1, N]$, should be similar to $P_j^g$. The goal of the training stage is to produce a deep learning feature extractor $f_\theta()$ that maximizes distance $\|f(P_j^b) - f(P_j^g)\|$, while minimizing $\|f(P_j^p) - f(P_j^g)\|$.

In the inference stage, the input is PCDs from a previously unseen object class, e.g. a knife. The input includes anomaly-free PCDs (also called "templates") $Y^1, \cdots Y^G$ and an object under test $Y^t$. Our system registers these PCDs in the same reference frame, splits them into patches, extracts the patch features using a trained extractor $f_\theta()$, and assigns the anomaly score $\delta_j$ to each patch $P_j^t$. Finally, the anomaly scores for all patches are aggregated into a single anomaly score $\Delta$ for the test object $Y^t$.

Our patch comparison is *strictly spatially-aware* – we compare only patches that originate from the same location $c_j$. Thus, registration is a cornerstone operation of our method. For a given pair of PCDs $X$, $Y$ we use the Iterative Closest Point (ICP) algorithm [27] to learn a rigid transform $\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3$ that maps $Y$ into a reference frame of the template $X$.

### A. Training the Feature Extractor

Our feature extractor is based on a triplet deep learning network [5]. Triplet networks are trained to learn discriminative features by comparing triplets of an *anchor*, a *positive*, and a *negative* instance. The anchor and the positive belong to identical objects, while the negative is chosen from a different object. In our case, the anchor and the positive

correspond to two "good" patches $P_j^g$, $P_j^p$; and the negative is an anomalous "bad" patch $P_j^b$.

*1) Triplet Generation:* First, we produce tuples $(X^g, X^p, X^b)$ of registered PCDs. The anomaly-free PCDs $X^g$ and $X^p$, $p, g \in [1, \ldots, N]$, $p \neq g$, are registered in the frame of the anomalous PCD $X^b$, $b \in [N{+}1, \ldots, M]$, using the ICP algorithm. Next, we sample patch centers $c_j$ using the Farthest Point Sampling (FPS) algorithm [28] from the anomalous regions of $X^b$. Finally, we recover patches as spherical regions centered at $c_j$, i.e. $P_j = \{x_i \in X : \|x_i - c_j\|_2 \leq R\}$. The result is a set of patch triplets $(P_j^g, P_j^p, P_j^b)$, where the index $j$ denotes a $j$-th patch obtained from the PCD triplet $(X^g, X^p, X^b)$.

*2) Feature Extractor:* We use PointNet++ [29] as an underlying backbone network. PointNet++ adapts the idea of a convolutional network to the unstructured nature of PCDs. Specifically, we replace the standard classification head with the following contrastive head that accepts the output of the backbone $\mathbf{h} \in \mathcal{R}^{1024}$ and produces patch features $\mathbf{m} \in \mathcal{R}^d$:

$$\mathbf{z}_1 = \mathrm{ReLU}(\mathrm{BN}_{1024}(\mathrm{Dropout}_{p=0.4}(\mathbf{h}))), \quad \mathbf{z}_1 \in \mathbb{R}^{B\times 1024},$$
$$\mathbf{z}_2 = \mathrm{ReLU}(\mathrm{BN}_{256}(\mathrm{Linear}_{1024\to 256}(\mathbf{z}_1))), \quad \mathbf{z}_2 \in \mathbb{R}^{B\times 256},$$
$$\mathbf{m} = \frac{\mathrm{Linear}_{256\to d}(\mathbf{z}_2)}{\|\,\mathrm{Linear}_{256\to d}(\mathbf{z}_2)\|_2}, \quad \mathbf{m} \in \mathbb{R}^{B\times d}.$$

We then train the feature extractor with the triplet loss:

$$\mathcal{L} = \frac{1}{|S|}\sum \max\Big(0, \|m^g - m^p\|_2^2 - \|m^g - m^b\|_2^2 + \alpha\Big), \quad (1)$$

where $S$ is the training set; $\mathbf{m}^g = f(P_j^g), \mathbf{m^p} = f(P_j^p), \mathbf{m^b} = f(P_j^b)$ are the feature vectors of the anchor, positive, and negative patches, respectively. The margin $\alpha(>0)$ enforces a minimum separation between dissimilar patches, preventing trivial solutions in which all embeddings collapse to a single point.

In our problem, $P_j^g$, $P_j^p$, and $P_j^b$ come from a specific location $c_j$ within a global coordinate frame. *This introduces a subtle, but important, difference from a classic formulation of the triplet loss: Our triplets are fixed.* Most contrastive learning losses, like InfoNCE, expect that negative objects can (and should!) be freely exchanged between triplets. However, in our case, "good" patch $P_j^g$ and anomalous patch $P_j^b$ usually have similar geometry, with $P_j^b$ only slightly different from $P_j^g$. Exchanging $P_j^b$ with some other anomalous patch may create a clearly unrepresentative training set. As a result, standard contrastive learning approaches like SimCLR cannot be directly applied to our case "out-of-the-box". Instead, we rely on tailored triplet generation and data augmentation techniques and devise a custom dual-network architecture for efficient triplet mining.

*3) Data Augmentation:* PointNet++ is not inherently rotationally invariant because its local feature extraction relies on Euclidean coordinates of points in the patch [30]. To make our model robust to arbitrary changes in object orientation and to reduce the risk of overfitting, we employed several data augmentation techniques:

- Global rotation – a large random rotation along Euler angles $(\alpha, \beta, \gamma)$ uniformly sampled from a range of $\pm 90°$. The same rotation is applied to the anchor, positive, and negative patches in a triplet.
- Individual rotation – a small ($\pm 3°$) random rotation applied to each patch in the triplet independently.
- Translation – a small (e.g., ±0.03 units) random translation is added to each patch independently.
- Gaussian noise with low standard deviation (0.005 units) added to each patch independently.
- Edge crop – random removal of a few small spherical areas with $r \in (0.05R, 0.15R)$ near the edge of a patch.

**Algorithm 1** EMA-based triplet miner with yield control

**EMA update:** $\theta_m \leftarrow \lambda \cdot \theta_m + (1-\lambda)\cdot\theta$
**Embeddings:** $m^g, m^p, m^b \leftarrow f_{\theta_m}([P_j^g; P_j^p; P_j^b])$
**Distances:** $d^+_{(i)} = \|m^g_{(i)} - m^p_{(i)}\|_2, d^-_{(i)} = \|m^g_{(i)} - m^b_{(i)}\|_2$
Define triplet sets:
- hard: $\mathcal{H} = \{\mathcal{B}_{(i)} : d^-_{(i)} - d^+_{(i)} \leq 0\}$
- semi-hard: $\mathcal{S} = \{\mathcal{B}_{(i)} : 0 < d^-_{(i)} - d^+_{(i)} \leq \alpha\}$
- easy: $\mathcal{E} = \{\mathcal{B}_{(i)} : d^-_{(i)} - d^+_{(i)} > \alpha\}$

$\mathcal{B}' \leftarrow \mathcal{S}$; compute yield $y = |\mathcal{B}'|/|\mathcal{B}|$
**if** $y < \gamma$ **then**
    $k = \lfloor(\gamma - y)\,|\mathcal{B}|\rfloor$; add $k$ items from $\mathcal{H}$ to $\mathcal{B}'$
    update $y = |\mathcal{B}'|/|\mathcal{B}|$
**if** $y < \gamma$ **then**
    $k = \lfloor(\gamma - y)\,|\mathcal{B}|\rfloor$; add $k$ items from $\mathcal{E}$ to $\mathcal{B}'$
**return** $\mathcal{B}'$, $y$

*4) Triplet Mining:* We perform triplet mining to improve the efficiency of training and to reduce training time. As training progresses, most triplets become "easy": the distances between anchor and positive $d^+ = \|m^g - m^p\|$ and between anchor and negative $d^- = \|m^g - m^b\|$ become greater than the margin, i.e. $d^- - d^+ > \alpha$. "Easy" triplets yield zero loss, provide little gradient information, and slow the convergence [5]. By selecting hard $d^- < d^+$ or semi-hard $d^+ < d^- < d^+ + \alpha$ triplets, the network is forced to learn more discriminative features.

Since our triplets are fixed, we cannot use standard triplet mining algorithms such as MoCo [8]. We devise a memory-efficient EMA-based algorithm that maintains a pair of networks: a *trainer* network $f_\theta()$ whose weights $\theta$ are updated by a backpropagation, and a *miner* network $f_{\theta_m}()$ whose weights $\theta_m$ are updated as an exponential moving average (EMA) of $\theta$ with momentum $\lambda \in (0, 1)$.

From each training batch $\mathcal{B}$, a miner network creates a new batch $\mathcal{B}'$ of mined triplets (see Algorithm 1). The miner first adds all semi-hard triplets to $\mathcal{B}'$ and calculates the mining yield $y = |\mathcal{B}'|/|\mathcal{B}|$. If the yield is too low, the miner adds hard and then easy triplets to maintain the minimum output batch size $|\mathcal{B}'| \geq \gamma|\mathcal{B}|$, where $\gamma \in [0, 1]$ is a hyperparameter. The trainer network then performs a single gradient descent step on a batch of mined triplets $\mathcal{B}'$.

At the end of each epoch, we evaluate the average yield $\overline{y}$ and dynamically update the threshold $\alpha$ to maintain the yield in a given range $[y_{min}, y_{max}]$: if $\overline{y} < y_{min}$, the $\alpha$ is

**Algorithm 2** Location-aware patch comparison

1: Register $Y^1, \cdots, Y^G, Y^t$ into same coordinate frame
2: Compute patch centers $(c_1, \cdots, c_L)$ = FPS($Y^1$)
3: **for** each patch center $c_j \in (1, L)$ **do**
4: Initialize memory bank $\mathcal{M}_j \leftarrow \emptyset$
5: **for** each good object $Y^i \in (Y^1 \cdots Y^G)$ **do**
6: Extract patch $P_j^i$ from $Y^i$ with center $c_j$
7: Compute feature vector $\mathbf{m}_j^i = f(P_j^i)$
8: Append $\mathbf{m}_j^i$ to memory bank $\mathcal{M}_j$
9: Initialize $\mathcal{D}_j \leftarrow \emptyset$
10: **for** each pair $(\mathbf{m}_j^i, \mathbf{m}_j^k)$ in $\mathcal{M}_j$ **do**
11: Append $\left\|\mathbf{m}_j^i - \mathbf{m}_j^k\right\|_2$ to $\mathcal{D}_j$
12: **for** each patch $P_j^t$ in the "test" object $Y^t$ **do**
13: **if** $|P| < |P|_{min}$ **then continue**
14: Compute feature vector $\mathbf{m}_j^t = f(P_j^t)$
15: Initialize $\mathcal{D}_j^t \leftarrow \emptyset$
16: **for** each $\mathbf{m}_j^i$ in $\mathcal{M}_j$ **do**
17: Append $\left\|\mathbf{m}_j^i - \mathbf{m}_j^t\right\|_2$ to $\mathcal{D}_j^t$
18: Compute patch anomaly score $\delta_j$ using (2)
19: **return** Object anomaly score $\Delta = \mathcal{A}_{op}^{(k)}(\delta^{(1)}, \cdots, \delta^{(k)})$

increased; if $\overline{y} > y_{max}$, the $\alpha$ is decreased.

### *B. Location-aware Patch Comparison*

During inference, we compare the "test" object $Y^t$ against the set $Y^1, \cdots Y^G$ of "good" template objects. Algorithm 2 summarizes the workflow, starting with registering $Y^t$ and $Y^1, \cdots Y^G$ into the same coordinate frame, defined by $Y^1$.

*1) Patch Selection:* For anomaly detection, we need full and uniform coverage of PCDs $Y^1, \cdots Y^G, Y^t$ by patches. We sample patch centers $c_1, \cdots, c_L$ using the FPS algorithm (line 2). Next, for each location $c_j$, we extract patches $P_j^1, \cdots, P_j^G$ from good objects $Y^1, \cdots Y^G$, and a patch $P_j^t$ from a test object $Y^t$. The number of patches $L$ and their centers $c_j, j \in [1, \cdots, L]$ are the same across all objects.

If only a fraction of the test object surface was scanned, we must exclude the non-scanned regions of the PCD from inference. We use a simple heuristic and compare only patches with $|P| > |P|_{min}$, where $|P|_{min}$ is an object-specific hyperparameter that denotes the minimum number of points in a patch. $|P|_{min} = 0$ for a fully scanned PCD (line 13 of Algorithm 2).

*2) Location-aware Comparison Algorithm:* Like PatchCore [21], we employ the concept of a *memory bank*. Unlike PatchCore, which uses a global memory bank $\mathcal{M}$, we use $L$ local memory banks for patch locations $c_1, \cdots, c_L$ – i.e. a separate local memory bank $\mathcal{M}_j$ for each location $c_j$.

Each local memory bank $\mathcal{M}_j$ contains features $\mathbf{m}_j^1, \cdots, \mathbf{m}_j^G \in \mathcal{R}^d$ extracted from the "good" patches $P_j^1, \cdots, P_j^G$ (lines 4–8). The number of patch centers $L$ is much less than the number of points in $Y^1, \cdots, Y^G, Y^t$. Thus, the total amount of memory $GLd$ used by all local banks is not a performance concern.

Small $G$ poses a problem for patch comparison. We must compare the feature vector $\mathbf{m}_j^t$ with the sample in $\mathcal{M}_j$. However, statistical techniques such as the Mahalanobis distance expect the size of the bank to be greater than the dimensionality of a feature, that is, $|\mathcal{M}_j| = G > d$. But real-world scenarios provide only a few "good" samples; e.g. $G = 4$ for the Real3D-AD dataset. This is much less than the typical dimensionality $d$. For example, an FPFH descriptor [31], a widely used handcrafted feature to encode local surface geometry, has $d = 33$.

To address this, we instead compare the pairwise distances between the features $\mathbf{m}_j^t$ and the contents of the memory bank $\mathcal{M}_j$. Let the distance $\text{dist}_j(t, i) = \|\mathbf{m_j^t} - \mathbf{m_j^i}\| \in \mathcal{R}$ implicitly represent the difference in geometry between the "test" patch $P_j^t$ and the corresponding patch $P_j^i$ from the "good" object $Y^i$. Together, $\mathcal{D}_j^t = \{\text{dist}_j(t, i)\}, i \in [1, \cdots, G]$ is a set of distances between the "test" patch $\mathbf{m}_j^t$ and the sample of good patches $\mathcal{M}_j$, extracted from $Y^1, \cdots Y^G$ at the location $c_j$ (lines 14–17).

High values of $\mathcal{D}_j^t$ can indicate the presence of a defect, as there is a significant difference in geometry between the test object and the "good" objects in the neighborhood of $c_j$. However, $\mathcal{D}_j^t$ can be interpreted only in the context of how variable the geometries around $c_j$ are in general.

Let $\text{dist}_j(k, i) = ||\mathbf{m_j^k} - \mathbf{m_j^i}|| \in \mathcal{R}$ be a pairwise distance between the features $\mathbf{m_j^k}, \mathbf{m_j^i} \in \mathcal{M}_j$ of two good patches $P_j^i$ and $P_j^k$, $i, k \in [1, \cdots, G], k \neq i$. Then $\mathcal{D}_j = \{\text{dist}_j(k, i)\}$ is a set of mutual distances between features of all "good" patches; $|\mathcal{D}_j| = G(G-1)$. The variance $\sigma(\mathcal{D}_j)$ captures the expected geometric variability due to measurement noise and manufacturing tolerances in the vicinity of $c_j$ (lines 9–11).

We employ Cohen's effect size $\delta_j$ between the samples of pairwise distances $\mathcal{D}_j^t$ and $\mathcal{D}_j$ as the final anomaly score for a "test" patch $P_j^t$ (line 22).

Let $\overline{\mathcal{D}_j}$ and $\overline{\mathcal{D}_j^t}$ be the mean values of $\mathcal{D}_j$ and $\mathcal{D}_j^t$ respectively. Then Cohen's effect size [32] is computed as:

$$\delta_j = \left| \frac{\overline{\mathcal{D}_j} - \overline{\mathcal{D}_j^t}}{s_j} \right|; \quad s_j = \sqrt{\frac{(|\mathcal{D}_j| - 1)\sigma^2(\mathcal{D}_j) + (|\mathcal{D}_j^t| - 1)\sigma^2(\mathcal{D}_j^t)}{|\mathcal{D}_j| + |\mathcal{D}_j^t| - 2}}, \tag{2}$$

where $s_j$ is the pooled standard deviation.

We compute an object anomaly score (line 19) as an aggregation $\Delta = \mathcal{A}_{op}^{(k)}(\delta^{(1)}, \cdots, \delta^{(k)})$ of $k$ patches with the highest anomaly scores $\delta^{(1)}, \cdots, \delta^{(k)}$, where $\mathcal{A}_{op}$ is an aggregation operator. PatchCore and other popular algorithms use $\Delta = \mathcal{A}_{max}^{(1)}$, i.e. the object anomaly score is the score of the most anomalous patch. We also experiment with the sum $\mathcal{A}_{sum}^{(k)}$ and the mean $\mathcal{A}_{mean}^{(k)}$ of the $k$ highest scores.

To precisely localize anomalies, for each point $x^t \in Y^t$, we compute the per-point anomaly score as a weighted average of patch anomaly scores, with weights based on the distance to each patch center $c_j$:

$$\delta(x^t) = \frac{\sum_{j=1}^{K} \delta_j \mathcal{N}(\left\|x^t - c_j\right\|, R)}{\sum_{j=1}^{K} \mathcal{N}(\left\|x^t - c_j\right\|, R)}. \tag{3}$$

We employ a Gaussian kernel as the distance weight: $\mathcal{N}(\|x^t - c_j\|, R) = \exp\left(-\frac{\|x^t - c_j\|^2}{2R^2}\right)$.

## IV. Results

*1) Datasets:* We use the Anomaly-ShapeNet v2 [17] and Real3D-AD [25] datasets to validate our approach. Anomaly-ShapeNet contains 1720 synthetic point clouds of 52 object classes. Real3D-AD contains 648 real scans of 12 object classes, out of which 300 are defective.

*2) Implementation Details:* We extracted 5000 and 15000 patch triplets from the Real3D-AD dataset and Anomaly-ShapeNet, respectively. After rescaling PCDs from Anomaly-ShapeNet by a constant factor of 20 to match the scale of the Real3D-AD dataset, we used a patch radius $R = 2$ and resampled all patches to a constant size of 512 points. 75% of all patches were used for training, and the remainder for validation.

We used the PointNet++ implementation [33] with multiple-scale grouping (MSG), a patch feature dimensionality (embedding size) of $d = 64$, and a batch size of 84. We trained our network using the Adam optimizer and a cosine rate scheduler, with an initial learning rate of $5 \times 10^{-4}$.

We set the initial triplet margin to $\alpha = 0.6$, the miner momentum to $\lambda = 0.99$, the critical yield $\gamma = 0.5$, and the yield boundaries $y_{min} = 0.5$, $y_{max} = 0.75$. We trained the contrastive model for 150 epochs on an NVIDIA RTX5090 GPU with 32 GB VRAM.

The accuracy of the spatially-aware comparison relies on the precise registration of the test PCD $Y^t$ against the template $Y^1$. Registration may be affected by scan artifacts such as random noise, minor omissions, and, especially, reflections along one of the axes. ICP, a randomized algorithm, is prone to occasional failures on highly symmetric objects such as starfish. To ensure registration quality, we performed 5 retries of the ICP algorithm and measured the one-sided Chamfer distance from the test PCD to the template:

$$\text{Chamfer}(Y^t \to Y^1) = \frac{1}{|Y^1|} \sum_{x \in Y^1} \min_{y \in (\mathbf{R}Y^t + \mathbf{t})} \|x - y\|_2^2. \quad (4)$$

We chose $\mathbf{R}, \mathbf{t}$ that produce the minimum Chamfer distance $\text{Chamfer}(Y^t \to Y^1)$.

Real3D-AD contains only a top-down or bottom-up view of test objects. To exclude missing areas from inference, we set a minimum point threshold $|P|_{min} = 0.5\mu(|P|)$, where $\mu(|P|)$ is the mean number of points across all the patches. $|P|_{min} = 0$ for Anomaly-ShapeNet, as it contains full PCDs.

To make our baseline comparable with related work [21], [25], [26], we set the object anomaly score to $\Delta = \mathcal{A}^{(1)}_{max}$.

The mean inference time on an Intel Core i7 CPU ranges from 2.7 to 13.1 seconds.

*3) Accuracy:* We use the Area Under the Receiver Operating Characteristic (AUROC) metric, computing both object-wise AUROC (O-ROC) and point-wise AUROC (P-ROC) for each object class.

We use 10-fold cross-validation to evaluate our method, with each fold containing a subset of object classes. Thus, every time we perform an inference on an object class, we use a separate extractor *not trained* on that object class. In this way, we completely exclude the leakage of training data in the test set and evaluate how well COSARAD generalizes to unseen objects.

TABLE I
Average accuracy on the Anomaly-ShapeNet dataset.

| Metric | PO3AD | Simple3D | MC3D-AD | **COSARAD** |
|---|---|---|---|---|
| O-ROC (%) | 83.9 | 86.4 | 84.1 | **91.3** |
| P-ROC (%) | 89.9 | 92.9 | – | **96.1** |

We compare the accuracy of COSARAD on the Real3D-AD dataset with the state of the art (SOTA) in the Table II. As shown, our approach outperforms the current SOTA by a wide margin. Our average object-level AUC is 90.1% compared to 82.9% by the nearest competitor, the PointCore algorithm. Furthermore, our approach localizes defects with a high degree of accuracy. Figure 1 visualizes the localization of the anomalies using our model.

We also significantly advance SOTA on the Anomaly-ShapeNet dataset; see Figure 3 and Table I (detailed per-object results are omitted due to space constraints).

COSARAD is accurate for highly prevalent anomaly types such as "bulge" and "sink", and less accurate for rare types (see Table III). This suggests that accuracy is partly driven by anomaly prevalence in training data, motivating future work on class-balanced contrastive learning.

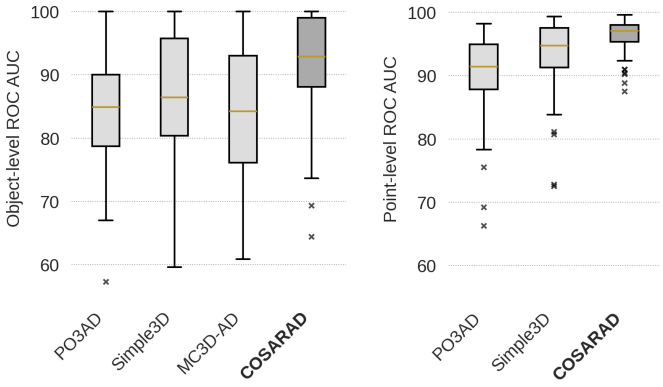


Fig. 3. Accuracy on the Anomaly-ShapeNet dataset.

*4) Ablation Studies:* To evaluate how our key innovations – pertaining specifically to contrastive representation learning and spatially aware patch comparison – contribute to high accuracy, we selectively disable each one while leaving the rest of the COSARAD pipeline intact. We present the results of our ablation studies in Table IV.

Replacing contrastive features with FPFH descriptors reduces O-AUROC from 90.1 to 71.5%, bringing COSARAD's accuracy closer to that of a Reg3D-AD baseline. These results clearly demonstrate that a *contrastive feature extractor is critical for high anomaly detection accuracy*.

To measure the contribution of spatially aware patch comparison, we replace local memory banks with the PatchCore reference implementation [21], which stores contrastive

TABLE II

ACCURACY ON THE REAL3D-AD DATASET, AS O-ROC/P-ROC VALUES. THE BEST RESULT IS IN **BOLD**, SECOND BEST IN UNDERLINE.

| Method → Object | Reg3D-AD [25] | PO3AD [34] | Simple3D [24] | MC3D-AD [17] | PointCore [26] | COSARAD |
|---|---|---|---|---|---|---|
| Airplane | 71.6/63.1 | 80.4/- | 76.5/88.1 | 85.0/62.8 | 66.0/60.8 | **95.6/98.1** |
| Candybar | 82.7/72.2 | 78.5/- | 85.1/96.2 | 77.8/73.6 | 97.6/76.0 | **100.0/98.6** |
| Car | 69.7/71.8 | 65.4/- | **98.1/99.2** | 74.9/81.9 | 86.6/70.6 | 92.2/98.0 |
| Chicken | 85.2/67.6 | 68.6/- | 82.6/86.1 | 71.5/64.0 | 84.1/78.0 | **92.1/90.2** |
| Diamond | 90.0/83.5 | 80.1/- | **100/99.0** | 95.5/94.2 | 96.3/81.0 | **100.0**/97.9 |
| Duck | 58.4/50.3 | 82.0/- | 78.7/96.6 | 83.1/82.2 | 68.4/71.2 | **98.0/98.1** |
| Fish | 91.9/85.2 | 85.9/- | 91.2/**99.6** | 91.2/90.6 | **99.2**/78.2 | 97.6/96.6 |
| Gemstone | 41.7/54.5 | 69.3/- | 70.4/**97.3** | 56.0/45.8 | 53.4/51.5 | **97.6**/97.0 |
| Seahorse | 76.2/81.7 | 75.6/- | 93.0/94.2 | 90.1/**95.0** | **97.3**/84.1 | 63.4/91.7 |
| Shell | 35.8/81.1 | 80.0/- | 85.1/**97.6** | 51.5/47.1 | **86.1**/78.1 | 77.8/89.8 |
| Starfish | 50.6/71.7 | 75.8/- | 69.5/85.8 | 76.6/69.0 | 65.2/73.6 | **82.3/94.1** |
| Toffees | 68.5/75.9 | 77.1/- | 88.9/96.8 | 78.3/93.4 | **92.9**/74.5 | 84.7/**91.4** |
| Mean | 70.4/70.5 | 76.7/- | 80.4/92.3 | 78.2/76.8 | 82.9/73.1 | **90.1/95.1** |

TABLE III

O-ROC BY ANOMALY TYPE. PERCENTAGES IN PARENTHESES INDICATE PREVALENCE OF A SPECIFIC ANOMALY TYPE IN THE DATASET.

| | Bending | Broken | Bulge | Crack | Hole | Hybrid | Scratch | Sink |
|---|---|---|---|---|---|---|---|---|
| Real3D-AD | – | – | 92.2 (71%) | – | – | 100.0 (2%) | – | 83.6 (27%) |
| Anomaly-ShapeNet | 97.5 (1%) | 60.0 (6%) | 95.0 (38%) | 72.3 (3%) | 72.7 (6%) | – | 87.9 (7%) | 94.3 (39%) |

features in a single global memory bank. This results in a loss of accuracy across most objects.

Notably, even with less expressive FPFH features, COSARAD achieves higher per-point accuracy than prior methods [25], [17], [26], and replacing local banks with a global one degrades O-AUROC despite using the same contrastive features. These results demonstrate that *spatially aware patch comparison is important for accurate anomaly detection, and even more important for anomaly localization*.

Finally, we hypothesize that computing an object anomaly score $\Delta$ from multiple patches may improve accuracy. We calculate $\Delta$ by aggregating the 3 to 5 most anomalous patches, using $\mathcal{A}^{(k)}_{sum}$ and $\mathcal{A}^{(k)}_{mean}$ aggregators. Compared to our baseline, which computes $\Delta$ as the score of the most anomalous patch $\Delta = \mathcal{A}^{(1)}_{max}$, these simplistic aggregators produced modest, although inconsistent, improvements. These results motivate us to explore more sophisticated techniques for computing $\Delta$ from multiple patch scores.

## V. DISCUSSION

We present COSARAD — a **co**ntrastive **s**patially-**a**wa**r**e **a**nomaly **d**etection algorithm for robotic quality inspection. Through two key innovations — a *contrastive feature extractor* and a *spatially-aware patch comparison* — we establish new state-of-the-art on Real3D-AD and Anomaly ShapeNet benchmarks. Ablation studies confirm that the contrastive feature extractor is critical for achieving previously unattainable detection accuracy, while spatially-aware comparison enables precise anomaly localization, essential for downstream corrective action. Our source code is available at `https://github.com/alextarvo/cosarad`.

Beyond the demonstrated results, COSARAD shows strong potential for improvement. For example, it fails consistently on the "Seahorse" object. We quantitatively analyzed the per-point distribution of anomaly scores and discovered two distinct failure modes. First, the edges of partially scanned PCDs have a fringe-like shape that the algorithm misidentifies as anomalies. Second, while COSARAD accurately handles "smooth" curve-like geometries common across many objects, accuracy is lower for out-of-distribution (OOD) geometries in objects such as "Seahorse" or "Shell".

We alleviated the first issue with a more aggressive threshold $|P|_{min}$ for a small subset of Real3D-AD objects; the second one — with extensive data augmentation that improves feature extractor accuracy on OOD geometries. The "edge crop" augmentation alone improved O-ROC on "Starfish" from 56.2 to 77.8 before further refinements raised it to 82.3. The most recent version of COSARAD, which employs a rotation-invariant backbone and semi-supervised contrastive learning, has made these heuristics unnecessary.

The location-aware patch comparison algorithm is the most promising direction for improvement. Currently, it compares only patch features $\mathbf{m}^{\mathbf{i}}_{\mathbf{j}}$ and $\mathbf{m}^{\mathbf{t}}_{\mathbf{j}}$, without accounting for residual displacement between nominally co-located patch centers after registration, which may itself signal an anomaly. Furthermore, the aggregation of patch-level scores into the object anomaly score $\Delta$ does not consider the spatial arrangement of anomalous patches. We aim to replace the deterministic comparison with a learned model that jointly reasons over patch features, spatial relationships, and registration uncertainty. Such models require large amounts of labeled data. Since manual annotation of 3D defects is costly and error-prone, developing generative approaches to inject realistic anomalies into existing PCDs is a complementary research direction.

TABLE IV

ABLATION STUDIES. $\downarrow$ INDICATES A SIGNIFICANT ($p < 0.05$) DECREASE IN ACCURACY COMPARED TO BASELINE. $\uparrow$ INDICATES AN IMPROVEMENT.

| Study → Object | Baseline | FPFH | PatchCore | Multi-point aggregation for computing global anomaly score | | | |
|---|---|---|---|---|---|---|---|
| | | | | $\mathcal{A}^{(3)}_{sum}$ | $\mathcal{A}^{(3)}_{mean}$ | $\mathcal{A}^{(5)}_{sum}$ | $\mathcal{A}^{(5)}_{mean}$ |
| Airplane | 95.6/98.1 | 76.6$^{\downarrow}$/91.1$^{\downarrow}$ | 72.8$^{\downarrow}$/96.2$^{\downarrow}$ | 96.5 /98.3 | 95.9 /98.0 | 96.3 /97.8 | 95.3 /98.0 |
| Candybar | 100./98.6 | 87.0$^{\downarrow}$/97.7$^{\downarrow}$ | 90.2$^{\downarrow}$/98.2$^{\downarrow}$ | 100. /98.6 | 100. /98.7 | 100. /98.6 | 99.9 /98.6 |
| Car | 92.2/98.0 | 71.8$^{\downarrow}$/92.4$^{\downarrow}$ | 80.5$^{\downarrow}$/95.3$^{\downarrow}$ | 93.9 /98.1 | 93.4 /98.1 | 93.1 /98.0 | 95.8$^{\uparrow}$/98.0 |
| Chicken | 92.1/90.2 | 83.7 /91.7 | 72.2$^{\downarrow}$/85.9$^{\downarrow}$ | 91.6 /88.9 | 91.9 /90.9 | 89.4 /90.0 | 91.7 /88.9 |
| Diamond | 100./97.9 | 90.2$^{\downarrow}$/97.7 | 95.7$^{\downarrow}$/97.7 | 100. /98.0 | 100. /97.6$^{\downarrow}$ | 100. /97.8 | 100. /97.8 |
| Duck | 98.0/98.1 | 63.7$^{\downarrow}$/90.4$^{\downarrow}$ | 92.1$^{\downarrow}$/98.1 | 98.6 /98.3$^{\uparrow}$ | 98.3 /98.1 | 98.8 /98.3 | 99.2 /98.1 |
| Fish | 97.6/96.6 | 77.3$^{\downarrow}$/93.0$^{\downarrow}$ | 89.8$^{\downarrow}$/97.9$^{\uparrow}$ | 98.8 /96.7 | 97.9 /96.7 | 98.6 /96.6 | 98.9 /96.6 |
| Gemstone | 97.6/97.0 | 63.7$^{\downarrow}$/82.9$^{\downarrow}$ | 77.0$^{\downarrow}$/96.1$^{\downarrow}$ | 98.8 /96.8 | 97.4 /96.7 | 97.5 /96.9 | 99.2 /96.9 |
| Seahorse | 63.4/91.7 | 63.8 /92.2 | 75.4$^{\uparrow}$/89.3$^{\downarrow}$ | 67.1 /92.2 | 56.2 /90.6$^{\downarrow}$ | 68.4 /92.8$^{\uparrow}$ | 69.3 /91.3 |
| Shell | 77.8/89.8 | 52.3$^{\downarrow}$/74.0$^{\downarrow}$ | 66.3$^{\downarrow}$/85.7$^{\downarrow}$ | 84.4$^{\uparrow}$/92.3$^{\uparrow}$ | 82.1 /91.7$^{\uparrow}$ | 80.7 /91.4 | 83.2$^{\uparrow}$/92.7$^{\uparrow}$ |
| Starfish | 82.3/94.1 | 53.2$^{\downarrow}$/70.8$^{\downarrow}$ | 77.5 /92.8 | 84.0 /94.1 | 85.5 /94.8 | 78.4 /93.5 | 85.8 /93.9 |
| Toffees | 84.7/91.4 | 64.3 /79.3$^{\downarrow}$ | 86.1 /96.8$^{\uparrow}$ | 86.6 /90.1 | 88.4 /91.0 | 89.5$^{\uparrow}$/91.4 | 86.5 /91.4 |
| Mean | 90.1/95.1 | 70.6 /87.8 | 81.3 /94.2 | 91.7 /95.2 | 90.6 /95.2 | 90.9 /95.3 | 92.1 /95.2 |

## REFERENCES


[1] J. E. See, "Visual inspection reliability for precision manufactured parts," *Human Factors*, vol. 57, no. 8, pp. 1427–1442, 2015.

[2] A. Nandagopal, J. Beachy, C. Acton *et al.*, "A robotic surface inspection framework and machine-learning based optimal segmentation for aerospace and precision manufacturing," *Journal of Manufacturing Processes*, vol. 134, pp. 146–157, 2025.

[3] Q. Zhou, X. Chen, and J. Tang, "GANs fostering data-augmentation for automated surface inspection with adaptive learning bias," *The International Journal of Advanced Manufacturing Technology*, 2024.

[4] M. Gerges and X. Chen, "Adaptive lighting for curved and nonuniform objects in optomechanical inspection systems," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5792–5802, December 2022.

[5] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[6] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," in *Advances in Neural Information Processing Systems*, 2018.

[7] T. Chen, S. Kornblith, M. Norouzi *et al.*, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, 2020.

[8] K. He, H. Fan, Y. Wu *et al.*, "Momentum contrast for unsupervised visual representation learning," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[9] A. Radford, J. W. Kim, C. Hallacy *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021.

[10] R. Zhang, Z. Guo, W. Zhang *et al.*, "Pointclip: Point cloud understanding by clip," in *Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8542–8552.

[11] M. Caron, H. Touvron, I. Misra *et al.*, "Emerging properties in self-supervised vision transformers," in *International Conference on Computer Vision (ICCV)*, 2021.

[12] Y. Wang, J. Peng, J. Zhang *et al.*, "Multimodal industrial anomaly detection via hybrid fusion," *Conference on Computer Vision and Pattern Recognition*, pp. 8032–8041, 2023.

[13] A. Costanzino, P. Z. Ramirez, G. Lisanti *et al.*, "Multimodal industrial anomaly detection by crossmodal feature mapping," in *Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 234–17 243.

[14] P. Bergmann, X. Jin, D. Sattlegger *et al.*, "The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization," in *Proc. Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2022.

[15] W. Wang, X. Lu, D. de Silva Edirimuni *et al.*, "Deep point cloud normal estimation via triplet learning," *arXiv:2110.10494*, 2024.

[16] Z. Zhou, L. Wang, N. Fang *et al.*, "R3d-ad: Reconstruction via diffusion for 3d anomaly detection," *arXiv:2407.10862*, 2024.

[17] W. Li, X. Xu, Y. Gu *et al.*, "Towards scalable 3d anomaly detection and localization: A benchmark via 3d anomaly synthesis and a self-supervised learning network," in *Conf. on Computer Vision and Pattern Recognition*, 2024.

[18] P. Bergmann, M. Fauser, D. Sattlegger *et al.*, "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings," in *Conf. on Computer Vision and Pattern Recognition*, 2020.

[19] H. Deng and X. Li, "Anomaly detection via reverse distillation from one-class embedding," in *Conf. on Computer Vision and Pattern Recognition*, 2022.

[20] P. Bergmann and D. Sattlegger, "Anomaly detection in 3d point clouds using deep geometric descriptors," in *Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 2612–2622.

[21] K. Roth, L. Pemula, J. Zepeda *et al.*, "Towards total recall in industrial anomaly detection," in *Conference on Computer Vision and Pattern Recognition*, 2022.

[22] E. Horwitz and Y. Hoshen, "Back to the feature: Classical 3d features are (almost) all you need for 3d anomaly detection," in *Conf. on Computer Vision and Pattern Recognition Workshops*, 2023.

[23] E. T. Lee, Z. Fan, and B. Sencer, "A new approach to detect surface defects from 3d point cloud data with surface normal gabor filter," *Journal of Manufacturing Processes*, vol. 92, pp. 196–205, 2023.

[24] Y. Cheng, Y. Sun, H. Zhang *et al.*, "Towards high-resolution 3d anomaly detection: A scalable dataset and real-time framework for subtle industrial defects," *arXiv:2507.07435*, 2025.

[25] J. Liu, G. Xie, R. Chen *et al.*, "Real3d-ad: A dataset of point cloud anomaly detection," *arXiv:2309.13226*, 2023.

[26] B. Zhao, Q. Xiong, X. Zhang *et al.*, "Pointcore: Efficient unsupervised point cloud anomaly detector using local-global features," *arXiv:2403.01804*, 2024.

[27] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[28] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[29] C. R. Qi, L. Yi, H. Su *et al.*, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Conference on Neural Information Processing Systems 2017*, 2017, pp. 5099–5108.

[30] S. Kim, J. Park, and B. Han, "Rotation-invariant local-to-global representation learning for 3d point cloud," in *Intl. Conf. on Neural Information Processing Systems*, ser. NIPS '20, 2020.

[31] R. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms for 3d registration," in *Intl. Conf. on Robotics and Automation*, 2009.

[32] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[33] X. Yan, https://github.com/yanx27/Pointnet\Pointnet2\pytorch, 2019.

[34] J. Ye, W. Zhao, X. Yang *et al.*, "Po3ad: Predicting point offsets toward better 3d point cloud anomaly detection," in *Conference on Computer Vision and Pattern Recognition*, June 2025, pp. 1353–1362.