# Towards Robot Anomaly Detection

Ken Miyamoto
miyamoto.ken@bc.mitsubishielectric.co.jp
Masato Tsuchiya
tsuchiya.masato@aj.mitsubishielectric.co.jp
Takashi Ota
ota.takashi@dx.mitsubishielectric.co.jp
Lee Teng-Yok
lee.teng-yok@ap.mitsubishielectric.co.jp

Information Technology R&D Center
Mitsubishi Electric Corporation
Kanagawa, Japan

**Abstract**

Object detection is commonly implemented on autonomous driving robot for avoiding crashes. The purpose of this work enables robot to detect both volume-less object (*e.g*. plastic bag, vomit) and object with volume (*e.g*. human, suitcase) in a large environment. Since avoiding costly annotation task, we apply anomaly detection. Conventional works of anomaly detection sacrifice either memory size, storage size, or detection performance. Hence, we propose anomaly detection on dynamic cluster selection, which attempts better performance from the above three perspectives. Our proposition outperforms conventional works from F1 score and AUROC aspects, while keeping low storage and low memory. That makes easier to implement on an embedded machine of a robot.

## 1 Introduction

Autonomous driving robots become more popular in warehouse, factory, and house. Most of the robots have object detection for avoiding crashes. A typical sensor for object detection is supersonic, infrared light, or LiDAR, which measures distance or shape. These sensors can only detect objects with volume. Here, we aim to detect not only object with volume but also volume-less object. An example of volume-less object is a plastic bag. A robot wheel gets entangled when a robot moves on a plastic bag. Another example is vomit. A robot makes virus spread if a robot accidentally moves on vomit. We use vision to detect both volume-less object and object with volume, because a camera is commonly implemented on an autonomous driving robot.

There are two approaches to detect an anomaly object by vision. The first approach is supervised learning. Anomaly objects are trained on Transformer [7, 12] or Convolutional Neural Network (CNN) [10, 17]. There are two obstacles to apply supervised learning on a robot. It's hard to collect a large amount of annotated images, though supervised learning requires them at training. Additionally, supervised learning forces object definition that a robot should avoid. A user may not imagine diverse categories of anomaly objects.

The second approach is anomaly detection [1, 6, 8, 14, 18, 21]. Distribution of normal state is modeled at training. Anomaly objects are detected by deviation from normal state distribution. Anomaly detection does not require a large number of annotated images and category definition of anomaly objects. These two conditions suit to implement on a robot.

However, conventional works of anomaly detection are not affordable in a large environment because conventional works sacrifice either memory size, storage size, or detection performance. Anomaly detection with moving camera [18, 21] requires a large storage size for keeping either normal state models or images. One of the models or the images enables anomaly detection only in a narrow space. Therefore, storage size becomes tremendous when a robot covers entire area in a large environment. Though other works [1, 6, 8, 14] effectively compress features of normal states, evaluation of these works is limited under static camera condition. We hypothesized that the performance may be degraded, when applying these works under moving camera condition in a large environment. The reason why one anomaly detection model may not fit all image features in a large environment.

Hence, we propose anomaly detection on Dynamic Cluster Selection (DCS), which attempts better performance about memory size, storage size, and detection accuracy. A model of each class predicts anomaly behavior in a local area. The local modeling encourages better detection accuracy even if the above computational resources are limited. In addition, for avoiding spatial fragmentation of classes, our clustering deals with both image feature difference and spatial correlation. That invokes auxiliary effect to suppress frequent model changes while moving.

To summarize, our contribution is below.

- Anomaly detection for robot: We propose anomaly detection, which is affordable to implement on a robot. Classwise local modeling achieves higher F1 score and AUROC, while keeping low memory size.
- Clustering that cares both feature difference and spatial correlation: Our clustering finds scene changes under spatial restriction. It enables spatially local modeling, which suppresses frequent model changes while a robot is moving.
- Evaluation in a large environment : Evaluation is held on RISEdb [15], which is one of the large environment datasets.

## 2 Related Work

Most of the robots commonly implement object detection for avoiding crashes. Mainly, there are two object detection approaches. The first approach uses point cloud. VoxelNet [22] detects objects by shape feature extracted in each voxel. PointPillar [11] projects point cloud to image plane. Objects are detected from a projected image by CNN. Other works are shown in [9], comprehensively. Point cloud object detection can detect only object with volume. Here, we aim to detect also a volume-less object (*e.g.* plastic bag and vomit).

The second approach uses vision. CNN [10, 17] has been known as a typical backbone architecture. Multiple convolutional layers compress image feature for arbitrary tasks. Visual transformer [7, 12] is another backbone. Transformer trains multi-layered exploitation mechanism of key, value, and query. We avoid to retrain these backbones because it takes much cost to collect annotated data. Furthermore, definitions of anomaly objects are occasionally hard under practical situation.

Anomaly detection is another way to detect both object with volume and volume-less object by vision. Anomaly state is detected by deviation from normal state model. GANomaly [1]

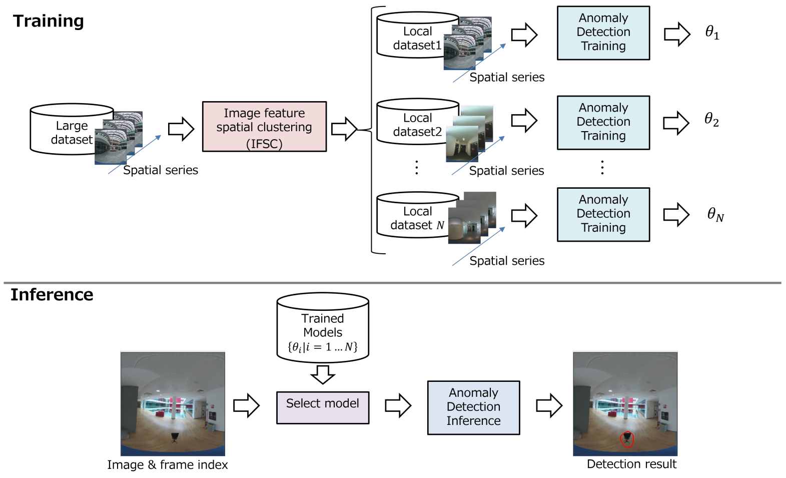

Figure 1: The pipeline of our robot anomaly detection with the dynamic clustering scheme. (bottom) During the inference, we select a model from the cluster based on the frame index. In this example, the detected anomaly is highlighted by the red circles.

generates normal state model by generative adversarial network. PaDiM [6] uses normal distribution to model embedding vectors on normal state. Each embedding vector is generated by concatenating multiple outputs of front layers. CFLOW-AD [8] applies normalizing flow, which is trained by supervised learning to estimate distribution of normal states. PatchCore [14] uses embedding vector as well as PaDiM [6]. Instead of normal distribution modeling in PaDiM [6], PatchCore [14] uses K-Nearest Neighbor (KNN) and memory bank. Anomaly state is detected by the minimum distance from embedding vectors in memory bank. Though these works [1, 6, 8, 14] effectively compress features of normal states, evaluation of these works is limited under static camera condition.

Some works describe anomaly detection under moving camera condition. However, storage size is bottleneck to apply these works in a large environment. One of the works uses principal component analysis [18]. Firstly, the method aligns an inferring frame to a database frame by viewpoint transformation. Anomaly state is detected by discrepancy from principal eigen vectors on normal state. The method allows only subtle viewpoint difference, because viewpoint transformation is processed on local image features commonly extracted from an inferring frame and a database frame. Anomaly detection system [21] stores images and their viewpoints on a database. Siamese network detects an anomaly object by comparing stored images and an inferring image. Siamese network is not ideal from computational aspect due to processing multiple stored images in a certain area. Computational cost is mitigated, when stored images are turned into embedding vectors previously. However, the approach with Siamese Network remains excessive computational load. All stored embedding vectors are processed for comparing an inferring image. As the summary of the two works [18, 21], both works require a large memory and a large storage for keeping images or vectors, because either an image or a vector enables anomaly detection only in a narrow space.

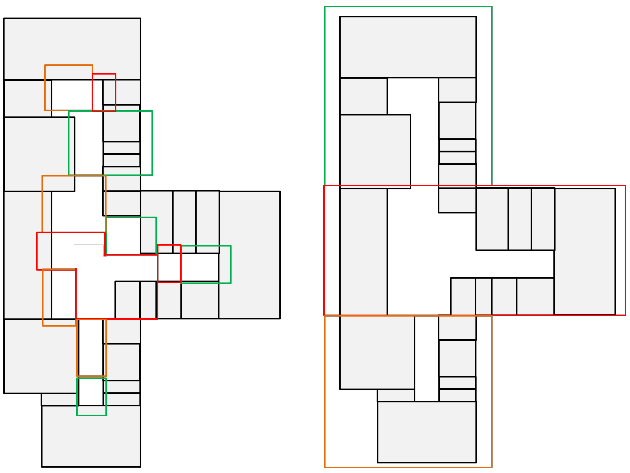

Figure 2: Conceptual effort of our image feature spatial clustering (IFSC). Combination of gray boxes and black lines describes a floor map. Red, green, orange boxes mean conceptual clustering result.: (a) fragmentation occurs spatially, if clustering deals with frames independently; (b) spatial fragmentation is suppressed by IFSC due to spatial restriction.

## 3 Robot Anomaly Detection

Our work is inspired by the anomaly detection system, which utilizes robots to navigate the environment [21]. We focus on embedding vector modeling to realize anomaly detection with low memory and low storage. PaDiM [6] can reduce memory and storage by fitting embedding vectors on normal state to normal distribution. PatchCore [14] reduces the number of embedding vectors by core-set sampling. Only important embedding vectors are stored for inference. We hypothesized that PaDiM and PatchCore make efforts under moving camera condition.

In addition to embedding vector modeling, we tailor clustering. We hypothesize that one anomaly detection model may not fit an entire area in a large environment (*e.g.* a building or shopping mall). Therefore, we divide a large dataset into several local datasets, which follows Dynamic Cluster Selection (DCS) [5, 19]. DCS separates entire feature space to multiple subspaces by a machine learning(*e.g.* support vector machine). Subsequently, a model is trained in each subspace.

Our anomaly detection model $\{\theta_i | i = 1...N\}$ is trained on each local dataset. The pipeline is shown in Figure1. $N$ represents the total number of classes separated by Image Feature Spatial Clustering (IFSC). $\theta_i$ refers to parameters of each model, and the contents in $\theta_i$ depend on a selected anomaly detection algorithm. For example, in the case of PatchCore, $\theta_i$ includes embedding vectors in a memory bank and a pair of an image threshold and a pixel threshold.

## 3.1 Optimal Anomaly Detection

For clarifying optimal anomaly detection under moving camera condition, we evaluate PaDiM [6] and PatchCore [14]. There are two reasons to choose the two algorithms. Firstly, PaDiM and PatchCore achieved the state of the art on MVTec dataset [3]. Secondly, network retraining is not required. Storage size increases linearly by the number of local models as shown in Figure1. We would suppress size of each model by avoiding to restore parameters of a network.

Evaluation detail is shown in Section4.3. Consequently, PatchCore outperforms PaDiM under moving camera condition.

## 3.2 Image Feature Spatial Clustering

We describe a clustering under DCS framework. Our subspace clustering is processed by image features extracted from respective frames. If subspace clustering deals with image features independently without any restrictions, subspaces are fragmented spatially. Then, a local model should often be changed while a robot moves in a large environment. That makes model management difficult. Hence, our subspace clustering employs spatial restriction. We implement spatial restriction in dendrogram, which repeats to merge two clusters $i$ and $j$. We call the clustering as Image Feature Spatial Clustering(IFSC).

When dendrogram addresses image features independently, distances among all the features are calculated. Mathematically, feature $i$ and $j$ are merged by Eq. (1).

$$\arg\min_{i,j\in G, i\neq j} d_{i,j} \tag{1}$$

$G$ is a group, which contains features. Initially, $G$ is consisted of features extracted from all respective frames. An initial feature in $G$ is created by concatenating embedding vectors on some patches. Here, five patches are concatenated, as shown in Figure3. Initial features in $G$ are merged by repetitive dendrogram process. $d_{i,j}$ is distance between features in cluster $i$ and $j$.

Instead of Eq. (1), IFSC applies Eq. (2) for implementing spatial restriction. Eq. (2) restricts distance calculation only with spatial neighbors.

$$\arg\min_{i\in G, j\in\eta(i)} d_{i,j} \tag{2}$$

$\eta(i)$ is neighbors of cluster $i$. Here, neighbors are limited to the next and the previous clusters spatially. Computational cost in Eq. (2) is $O(N)$ reduced from $O(N^2)$ in Eq. (1).

Conceptual effort of IFSC is shown in Figure2. Classes are fragmented spatially if image features are addressed independently. IFSC avoids the fragmentation by spatial restriction.

## 4 Evaluation

### 4.1 Dataset Generation

Since there is no publicly available dataset that evaluates anomaly detection under moving camera condition, to our knowledge, we synthesized a dataset by combining two public datasets. One is RISEdb [15], which is one of the large environment datasets. Frames in RISEdb are video sequences taken from various viewpoints. We use the original RISEdb

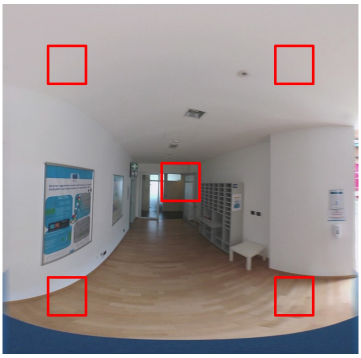

Figure 3: Five patches to create an image feature for clustering. Embedding vectors on five red patches are concatenated.

frames as the normal ones, and synthesized objects to their frames as abnormal ones. The synthesized objects were from the other dataset Pix3D [16], which includes diverse furniture images.

The detail synthesis procedure is described as follows. First, since the images of RISEdb [15] are captured by fisheye cameras, as shown in Figure 3, we first crop 50% of the frames horizontally to form squared patches. Then we sample 20% frames of RISEdb, as the frames are from video sequences and the adjacent ones could be redundant. Among the sampled frames, two of the third were used for training and the other for testing. To generate abnormal frames, we randomly selected objects from Pix3D [16], and scaled the patches of the sampled objects to be around 7 - 21% of the entire images.

## 4.2 Implementation

We implement our algorithm based on anomalib [2], and we use the pretrained Wide Residual Network [20] to extract the feature vectors. Frames were resized to $224 \times 224$ before forwarding through the neural networks.

We evaluate PaDiM [6] and PatchCore [14] under moving camera condition. When testing with PatchCore, we sample 0.2 of the feature vectors to form the core-set, and use the feature of the second and the third layers to form the embedding vector. We use the deviation from 9 the neighbors to detect anomaly objects. When testing with PaDiM, we concatenate the features of the first three layers. For IFSC, we set 600 as clustering threshold of dendrogram. The threshold was empirically specified to form 6 classes.

## 4.3 Optimal Anomaly Detection

Table 1 shows our result with PatchCore and PaDiM. It can be seen that PatchCore outperforms PaDiM with moving cameras. The difference could come from two reasons. First, PaDiM models the distribution of features per patches as a multivariate normal distribution, which could be insufficient to model the embedding vectors of frames captured by different viewpoints. In constrast, as PatchCore is based on the k-nearest-neighbors of the sampled embedding vectors, which is therefore more suitable on moving cameras. Second, PaDiM

| Datasets (Classes) | Training Frames | Test Frames | Models | AUROC (image) | F1 Score (image) | AUROC (pixel) | F1 Score (pixel) |
|---|---|---|---|---|---|---|---|
| (0) | 261 | 180 | PaDiM | 0.824 | 0.730 | 0.991 | 0.369 |
| | | | PatchCore | **0.879** | **0.778** | **0.993** | **0.489** |
| (1) | 141 | 90 | PaDiM | 0.819 | 0.667 | 0.985 | 0.362 |
| | | | PatchCore | **0.864** | **0.682** | **0.988** | **0.398** |
| (2) | 1102 | 758 | PaDiM | 0.791 | 0.667 | 0.983 | 0.283 |
| | | | PatchCore | 0.791 | **0.691** | 0.983 | **0.309** |
| (3) | 48 | 22 | PaDiM | **0.905** | **0.800** | **0.998** | **0.655** |
| | | | PatchCore | 0.818 | 0.645 | 0.997 | 0.573 |
| (4) | 279 | 188 | PaDiM | 0.851 | 0.667 | **0.992** | 0.433 |
| | | | PatchCore | **0.864** | **0.674** | 0.991 | **0.437** |
| (5) | 107 | 78 | PaDiM | 0.807 | 0.696 | 0.993 | 0.469 |
| | | | PatchCore | **0.881** | **0.825** | **0.994** | **0.500** |
| Total | 496 | 1316 | PaDiM | 0.809 | 0.680 | 0.986 | 0.339 |
| | | | PatchCore | **0.859** | **0.739** | **0.992** | **0.462** |

Table 1: Comparison between PaDiM and PatchCore on sequence1 in RISEdb. "Total" describes weighted average of metrics. Test frames are used as the weight. Datasets(Classes) are created by IFSC shown in Section 3.2

.

| Datasets (Classes) | Models | AUROC (image) | F1 Score (image) | AUROC (pixel) | F1 Score (pixel) |
|---|---|---|---|---|---|
| (0) | PatchCore-S | 0.756 | 0.667 | 0.979 | 0.425 |
| | PatchCore-L | 0.873 | 0.667 | **0.993** | 0.462 |
| | IFSC+PatchCore-S | **0.875** | **0.772** | 0.992 | **0.494** |
| (1) | PatchCore-S | 0.719 | 0.667 | 0.971 | 0.285 |
| | PatchCore-L | 0.833 | 0.667 | 0.987 | 0.326 |
| | IFSC+PatchCore-S | **0.875** | 0.667 | **0.987** | **0.400** |
| (2) | PatchCore-S | 0.701 | 0.665 | 0.981 | 0.283 |
| | PatchCore-L | 0.739 | 0.665 | **0.986** | 0.359 |
| | IFSC+PatchCore-S | **0.806** | **0.745** | 0.986 | **0.366** |
| (3) | PatchCore-S | **0.864** | 0.667 | 0.994 | 0.383 |
| | PatchCore-L | 0.856 | 0.667 | 0.997 | 0.379 |
| | IFSC+PatchCore-S | 0.860 | 0.667 | **0.997** | **0.553** |
| (4) | PatchCore-S | 0.658 | **0.667** | 0.981 | 0.378 |
| | PatchCore-L | 0.849 | **0.667** | **0.991** | 0.424 |
| | IFSC+PatchCore-S | **0.855** | 0.657 | 0.991 | **0.434** |
| (5) | PatchCore-S | 0.648 | 0.667 | 0.988 | 0.414 |
| | PatchCore-L | 0.803 | 0.667 | 0.994 | 0.499 |
| | IFSC+PatchCore-S | **0.882** | **0.805** | 0.994 | **0.501** |
| Total | PatchCore-S | 0.703 | 0.667 | 0.981 | 0.326 |
| | PatchCore-L | 0.785 | 0.667 | 0.988 | 0.389 |
| | IFSC+PatchCore-S | **0.850** | **0.721** | **0.990** | **0.438** |

Table 2: Comparison between conventional anomaly detection by global modeling (*i.e.* PatchCore-S, PatchCore-L) and local modeling with DCS (*i.e.* IFSC+PatchCore-S). PatchCore-S limits embedding vectors in memory bank under 31360. PatchCore-L stores 111641 embedding vectors in memory bank. IFSC+PatchCore-L creates local models under DCS framework. A memory bank and thresholds are modeled locally. Embedding vectors in each memory bank are limited under 31360. These algorithms are evaluated on generated dataset, which combines RISEdb and Pix3D.

models the distribution in the level of patches, which make it less ideal when the normal patterns are moving. In the latest section, we describe our algorithm with PatchCore.

### 4.4 Local Modeling by Image Feature Spatial Clustering

We compared three settings. One is PatchCore-S, which trains frames in the entire area of a large environment. The embedding vectors in the memory bank were limited to 31360. A pair of an image threshold and a pixel threshold are applied to entire area (*i.e.* all test frames). The second one is PatchCore-L, which includes all embedding vectors extracted by IFSC+PatchCore-S. For our synthesized dataset, IFSC+PatchCore-S generates 111641 embedding vectors. The last one is IFSC+PatchCore-S where each local model can have at most 31360 embedding vectors in its own memory bank and have its own image-wise and pixel-wise thresholds to detect the anomaly, trained by its own data.

Table 2 shows the quantitative result. It clarifies that the local modeling (*i.e.* both a local memory bank and local thresholds) encourages higher F1 score and AUROC from comparison between PatchCore-L and IFSC+PatchCore-S. Memory bank of PatchCore-L includes all embedding vectors in all memory banks of IFSC+PatchCore-S. However, F1-score and AUROC of IFSC+PatchCore-S are higher than PatchCore-L. It means that local image and pixel thresholds are meaningful to divide normal and anomaly states in a local area.

Figure 4 shows the qualitative results. Among these results, IFSC+PathCore-S can detect anomaly that cannot be detected by PatchCore-S and PathCore-L, such as scene 1, 3, and 4. It should be noted that the anomaly maps of all methods are similar, although the detection results are different. Because of this observation, we suppose that the higher accuracacy of IFSC+PathCore-S is due to the local-model-wise thresholds.

## 5 Conclusion

For enabling a robot to detect both object with volume and volume-less object in a large environment, we propose robot anomaly detection, which is a combination of anomaly detection and dynamic cluster selection. Our evaluation proves that robot anomaly detection achieves higher F1 score and AUROC, while keeping low memory size and low storage size. Furthermore, our clustering of dynamic cluster selection avoids spatial class fragmentation. The effort makes model management easier while a robot moves. There is no public dataset for evaluating anomaly detection in a large environment. Hence, we create a dataset by combining two public datasets.

## Acknowledgement

Authors thank the staff of Mitsubishi Electric Information Technology Research Development Center and Business Innovation. Special gratitude to Lee Teng-Yok for his support in review and technical discussion, also to Masato Tsuchiya.

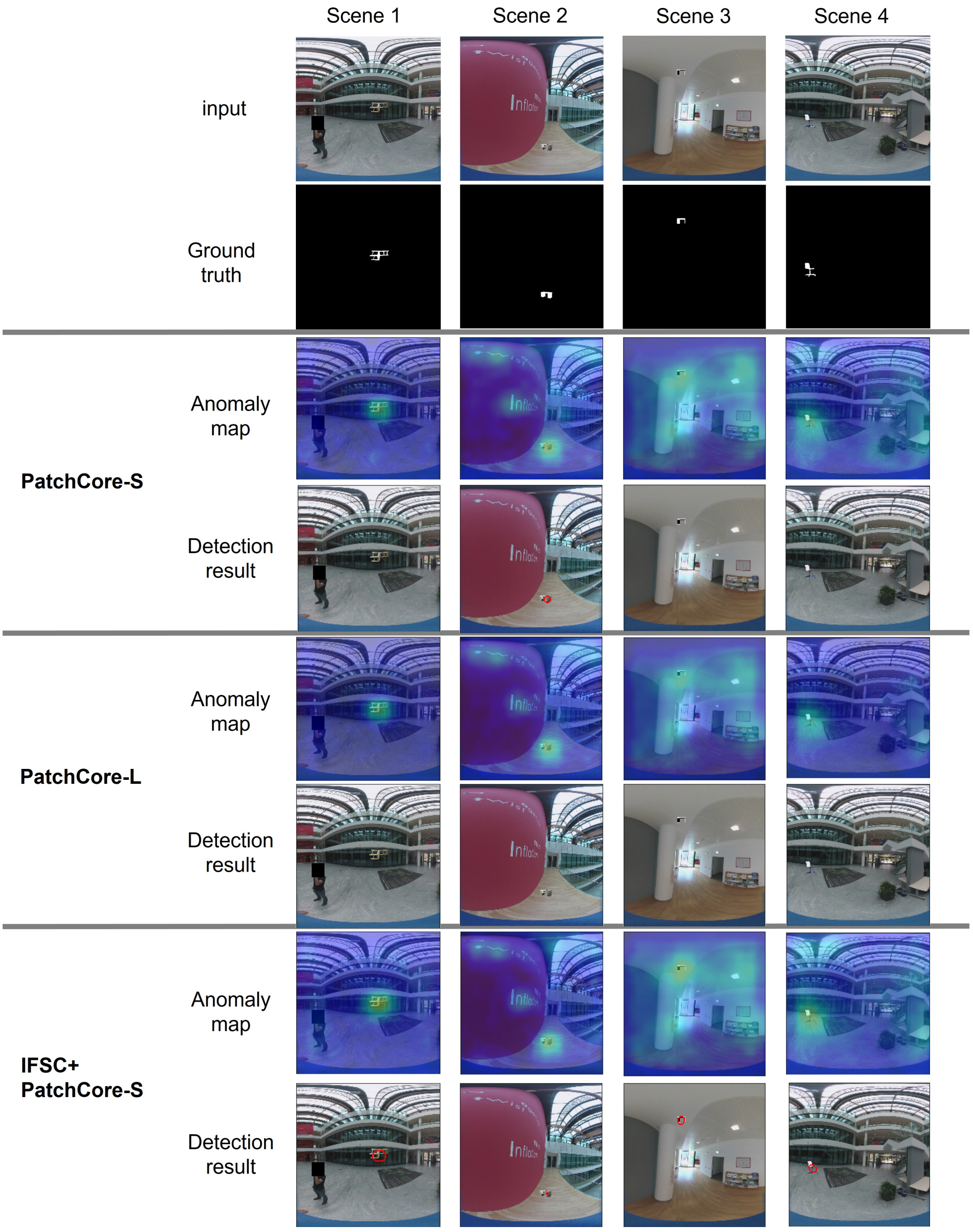

Figure 4: Qualitative comparison between PatchCore-S, PatchCore-L, and IFSC+PatchCore-S. In anomaly map, yellow color means high anomaly level. Conversely, blue is low. In detection result, red convex depicts a found anomaly object.

## References

[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision(ACCV)*, pages 622–637, 2018.

[2] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection, 2022.

[3] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *CVPR*, 2019.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, ISBN: 0387310738, 2006.

[5] Rafael M.O. Cruz, Robert Sabourin, and George D.C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41: 195–216, 2018. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.209.010. URL https://www.sciencedirect.com/science/article/pii/S1566253517304074.

[6] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *Pattern Recognition. ICPR International Workshops and Challenges*, pages 475–489, 2021.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[8] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *WACV*, 2022.

[9] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[13] Kevin P. Murphy. *Machine Learning A Probabilistic Perspective*. The MIT Press, LLC, ISBN: 9780262018029, 2013.

[14] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Scholkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *CVPR*, 2022.

[15] Carlos Sanchez-Belenguer, Erik Wolfart, Alvaro Casado-Coscolla, and Vitor Sequeira. Risedb: a novel indoor localization datasets. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 2020.

[16] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018.

[17] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020.

[18] Lucas A. Thomaz, Eric Jardim, and Allan F. da Silva. Anomaly detection in moving-camera video sequences using principal subspace analysis. In *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1003–1015, 2017.

[19] Biao Wang and Zhizhong Mao. A dynamic ensemble outlier detection model based on an adaptive k-nearest neighbor rule. *Information Fusion*, 63:30–40, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2020.05.001. URL `https://www.sciencedirect.com/science/article/pii/S1566253520302645`.

[20] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL `https://dx.doi.org/10.5244/C.30.87`.

[21] Muhammad Zaigham Zaheer, Arif Mahmood, M. Haris Khan, Marcella Astrid, and Seung-Ik Lee. An anomaly detection system via moving surveillance robots with human collaboration. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021.

[22] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.