IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Zero-Shot Recognition of Test Tube Types by Automatically Collecting and Labeling RGB Data

Yu Tang[1], Weiwei Wan*[1], Hao Chen[2], Masaki Matsushita[3], Jun Takahashi[3], Takeyuki Kotaka[3], Kensuke Harada[1]

***Abstract*—This work presents a method for automatically detecting and recognizing test tube types in a rack. It leverages automatic segmentation, clustering, and labeling processes to eliminate the need for explicitly preparing training data. These processes are addressed by using combined global prediction and local cropping, where global prediction estimates the slot occupation states of a rack, and local cropping extracts tube pictures in the local regions of each slot for clustering and labeling. With the help of the proposed method, the robotic tube manipulation system no longer needs tailored data and explicit training in the presence of new tubes, thus achieving flexibility and efficiency. Experimental evaluations conducted with a RealSense D405 camera and the UFactory xArm Lite6 robot manipulator confirm the method's effectiveness in accurately identifying novel test tube types under real-world conditions.**

***Index Terms*—Test Tube Detection, Robotic Manipulation, Deep Learning**

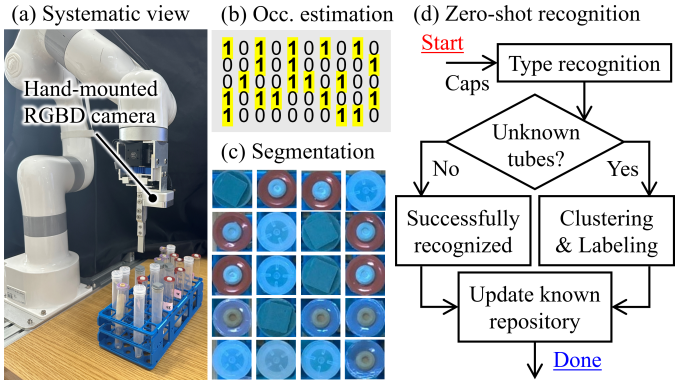

Fig. 1: Zero-shot test tube type recognition via slot occupancy estimation, segmentation, and automatic clustering-labeling.

I. Introduction

THIS paper presents a method for automatically detecting and recognizing test tube types in a rack. The method allows for the automatic segmentation, clustering, and labeling of unknown test tubes. It removes the need for preparing training data and adapts smoothly to new test tubes, thus significantly improving the flexibility of laboratory automation.

Test tubes play a crucial role in various scientific experiments. The ability to automatically detect and classify test tubes based on their physical characteristics—such as size, length, and geometry—is essential for advancing laboratory automation. However, this task is nontrivial because test tubes used in research settings do not conform to a single standard. When placed in racks, they may exhibit varying tilt angles and partial occlusions, complicating their detection and shape-based classification. In addition, test tubes are not industrially standardized products; manufacturers frequently update their designs, and new batches can differ significantly in appearance and dimensions. These variations require frequent reconfiguration of detection and classification systems. As a result, many existing laboratory automation systems avoid direct test tube type classification and instead rely on structured environments and precise robotic positioning for tube handling [1][2]. Our goal is to enable flexible and robust classification of test tubes by geometry, so that downstream robotic handling procedures can be uniformly defined for each tube type, reducing manual intervention and system reconfiguration.

Recently, with advancements in deep learning and large models, researchers have started exploring robust solutions to detecting and recognizing test tubes [3] in unstructured environments. For instance, Chen et al. [4] focused on test tube caps and trained a neural network to classify the caps for test tube recognition. The authors also presented a method that used point cloud data and the geometric constraints of the test tube rack to precisely infer the tilt of test tubes. Mac et al. [5] trained a neural network using custom data specifically designed for centrifuge tubes. The trained network can detect and segment test tubes that are placed on a flat, monochromatic surface, facilitating robotic manipulation. Additionally, the network provides information on whether a tube is properly capped.

Despite their improvements, the learning-based methods require preparing images and their corresponding labels for training neural networks. Such requirements significantly impede the flexibility of the detection and recognition system. Unlike existing methods, we propose a zero-shot approach that does not depend on pre-existing training data about particular tubes. Fig. 1 illustrates the basic workflow of our method. We focus on top-view tubes and use automatic segmentation, clustering, and labeling to eliminate the need for explicitly preparing training data. When a previously unseen tube type is encountered, the system automatically stores its appearance and augments the recognition database, enabling consistent identification in future encounters. Over time, it incrementally builds a tube-type library without human supervision, enhancing adaptability in dynamic laboratory environments.

In summary, our main contribution is that we develop a pipeline that enables zero-shot recognition of test tube types based on cap appearance, without any manual labeling. The system automatically segments, clusters, and labels new tube

1. Graduate School of Engineering Science, Osaka University, Japan. 2. College of Computer and Information Sciences, Fujian Agriculture and Forestry University, China. 3. H.U. Group Research Institute G.K., Japan
Contact: Weiwei Wan, wan@sys.es.osaka-u.ac.jp

types, and incrementally builds a recognition library for self-supervised tube classification. As a minor contribution, we also adopt the Depth Anything model for slot occupation prediction without camera calibration. We validate the proposed method using a UFactory xArm Lite6 robotic manipulator equipped with a RealSense D405 camera and various types of test tubes. Experimental results show that our method can recognize tube types without prior supervised training.

II. Related Work

A. General Object Detection

Supervised object detection received significant attention with the advance of machine learning-based approaches. Examples include but are not limited to the CNN series [6][7][8], the YOLO series [9][10][11][12], transformer-based methods [13][14]. While effective in their specific applications, the above methods rely on pre-labeled training data, which limits their flexibility and adaptability. Unlike them, the method developed in this work uses automatic segmentation, clustering, and labeling to achieve a zero-shot learning paradigm. It can adapt to a wide range of test tube types without manual data preparation. Especially for segmentation, we utilize Segment Anything Model 2 (SAM2) [15] to segment test tube caps from locally cropped slot images. Early studies on image segmentation focused on classical computer vision methods including setting threshold [16], edge detector [17], and normalized cuts [18]. A major advancement was the use of Fully Convolutional Networks (FCNs) [19] for semantic segmentation. Afterwards, vision-transformer and prompt-based learning have been applied to the segmentation process. Among them, the Segment Anything Model (SAM) [20] emerged as a significant breakthrough and offered a task-agnostic segmentation approach. SAM achieved high precision by generating multiple masks without requiring specific category information and inspired numerous variants that extended across diverse segmentation challenges. For instance, the Track Anything Model (TAM) [21] integrated SAM for efficient interactive tracking with minimal human intervention. PerSAM [22] focused on private-object segmentation and was optimized for applications requiring object-specific identification. Segment Any RGBD (SAD) [23] emphasized geometric features over texture by incorporating depth information and exhibited satisfying performance for transparent and 3D panoptic scenarios. Stable-SAM [24] addressed stability problems by modulating image feature sampling. Compared to prior methods, Segment Anything Model 2 (SAM2) [15] achieved faster image segmentation by using a novel data engine and transformer architecture with streaming memory. It has been used in various applications like medical segmentation on video segmentation dataset [25].

B. Robotic Vision in Lab Automation

Early laboratory automation primarily relied on automatically controlled robots [26][27] and emphasized simplified programming approaches [28]. Related research predominantly falls within the domain of automation. They preferred utilizing sensor-based systems rather than high-information-density visual devices for specialized detection tasks. The discussions on vision-based recognition were limited.

More recently, emerging studies have explored the integration of visual detection and motion generation methods to improve autonomy. For instance, Shiri et al. [29] explored using computer vision for liquid screening. Ochs et al. [30] used deep learning methods to estimate the confluence of cell cultures from images. Azizi et al. [31] used a robot manipulator to move a plate under a hyperspectral imaging system for measuring polymer degradation. However, most robotic vision systems in laboratory automation have primarily aimed at detecting specimens rather than the surrounding experimental apparatus.

Focusing specifically on apparatus detection, Luo et al. [32] used active vision to recognize glassware in the lab. De Jesus et al. [33] trained a YOLOv8 model to detect test tubes for pre-analytical manipulation. Ge et al. [34] studied the end-to-end grasp prediction for test tubes on a tray. Tiong et al. [35] trained a varied single-shot detector to detect various transparent chemical vessels like flasks, vials, etc. There are also several companies that provide vision solutions for lab automation [Note 1]. Most of the work on robotic vision in lab automation falls into the category of methods that rely on pre-trained datasets, as discussed in the first part of this section. Such approaches are challenging to apply to unknown states or previously unseen objects. Different from them, the focus of this study is on the continuous changes in test tubes and the challenge of adapting to new types of tubes without the need for dataset re-preparation. From this perspective, our proposed method addresses a research gap that has not yet been explored and offers a promising solution for automatic adaptation to newly introduced test tubes.

III. Rack Slot Occupation Estimation

The objective of rack slot occupation estimation is to predict which slots contain test tubes based on a given top-view image of a rack. The success rate of the prediction heavily depends on the viewpoint and the perspective distortion introduced by the camera. Images captured from different viewpoints and positions exhibit significant variation, which can easily lead to misclassification. To address this issue, we use an approach that integrates model-based detection with the prediction process. Compared to test tubes, the structures of racks are standardized and consist of 5×10 slots with a fixed geometric model. They have rich geometric features and are commonly made of opaque plastic materials. These characteristics make rack detection relatively straightforward. Leveraging this characteristic, we employ the algorithm presented in [4] to match the rack with its point cloud, thereby determining the 3D position and orientation of the rack. After that, the robotic arm is controlled to move its in-hand camera to a predefined fixed position above the rack to capture RGBD images of the rack. In this way, the system reduces perspective distortion and improves the accuracy and reliability of the slot occupation estimation process. After capturing images, we employ a neural network to predict whether slots in the rack contain test tubes. The use of a neural network is motivated by the fact that the pose of test tubes within the rack is not fixed. Rack slots

[Note 1] https://shorturl.at/7qF2u

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

are typically designed to be slightly larger than the diameter of the test tubes, in order to provide clearance to facilitate manual handling. The clearance, however, allows test tubes to tilt, making it challenging to accurately determine their corresponding slot using model-based methods. To address this issue, we adopt a neural network approach. The details of data collection and network architecture selection for the approach are shown below.

A. Data Collection

To achieve high accuracy in the neural network, a large amount of training data is required. We adopted a hybrid approach combining simulated data generation and semi-automatic methods to efficiently accomplish this task.

1) Simulated Data Generation

First, we constructed a simulation environment using Blender to generate simulated data. In the simulation, the rack was positioned on a plane, while the camera was fixed at a specific height directly above the center of the rack. Note that in the real world although the rack's pose is detected and the camera is repositioned to obtain views from consistent locations and angles, it cannot be guaranteed that the captured rack is always in the exact same relative position. Errors arise from multiple sources, including noise in the point cloud data obtained by the visual sensor, rack pose matching errors, camera calibration inaccuracies, and the robotic arm's absolute position control errors. They result in positional and orientational deviations in the captured rack. To simulate the errors in the real world, we added noise to the camera's position and orientation in the simulation. In addition to the errors, we also adjusted the data generation process to progressively increase the number of empty slots and gain variation. Through them, we can generate diverse datasets with varying test tube configurations and their corresponding ground truth. We could also simulate positional and orientation errors. Fig. 2(a) exemplifies some training data generated by the simulation. Note that although the figure illustrates two types of test tubes, the neural network designed to predict slot occupancy is independent of specific tube types. The independence is achieved by incorporating a pre-processing step that uses the Depth Anything V2 model (DA-v2) [36] to convert the RGB image into a depth image, preserving shape information while eliminating texture details.

2) Semi-Automatic Method

In addition to simulation data, we also employed a semi-automated method to collect a portion of real-world data. Due to the difference between the images obtained from the simulation environment and those from real-world usage scenarios—specifically in terms of lighting conditions, rack textures, and test tube shapes. We employ the semi-automatically collected real-world data to fine-tune (FT) the model initially trained on simulated data.

Fig. 2(b) illustrates the workflow of semi-automatic data collection. This process begins with a manually initialized state, where the ground-truth positions of the test tubes are known. Starting from this initial state, the robot alternates between capturing data, randomly rearranging a test tube, and capturing data again. During the operation, the robot is aware

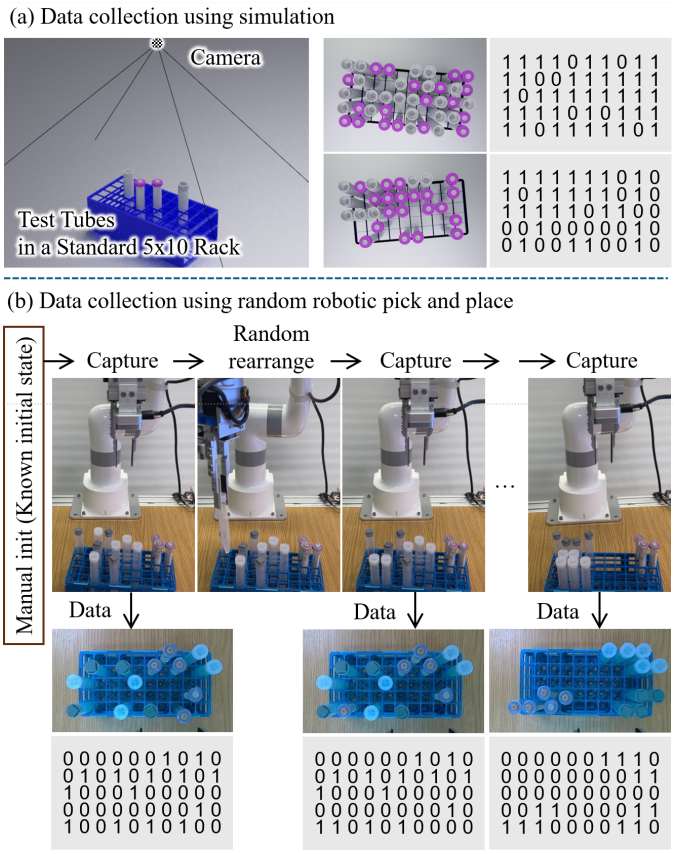

Fig. 2: (a) Generating training data using simulation. (b) Semi-automatic data collection in the real world using random robotic rearrangement.

of which test tube it has rearranged, allowing it to determine the ground-truth state at each step. The captured data can be correctly paired with the corresponding ground-truth values for training. Like in the simulation, each captured image is processed by the DA-v2 model to eliminate texture details during training. Although the process is automatic, collecting real-world data is considerably more time-consuming compared to collecting simulation data. Therefore, we did not rely entirely on the semi-automatically collected data for model training. Instead, we used it to fine-tune the results obtained from training on simulation data.

B. Network Design

Fig. 3 shows the architecture of our neural network. The network begins with an RGB input of dimensions $224 \times 224 \times 3$. The DA-v2 module is right after the input image and converts it into a depth image. The depth image is then passed through a patch embedding layer, where the image is divided into fixed-size patches with a position vector indicating the position information of the patch. These patch embeddings are then sent to a backbone network for feature extraction. The patch embedding layer in our ViT-based model divides each 224×224 input image into 16×16 non-overlapping patches, yielding 196 patches per image. Each patch is projected into a 192-dimensional embedding using a 2D convolution layer with both kernel size and stride set to 16. The backbone network

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

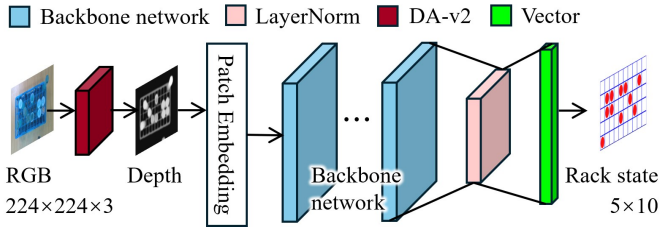

Fig. 3: Neural network architecture for globally predicting the rack slot occupations. It comprises a DA-v2 module for eliminating textures and a Resnet or ViT backbone for feature extraction and spatial reasoning. The output is a 1×50 vector transformed to a 5×10 matrix with each element indicating the availability of a tube in the corresponding rack slot.

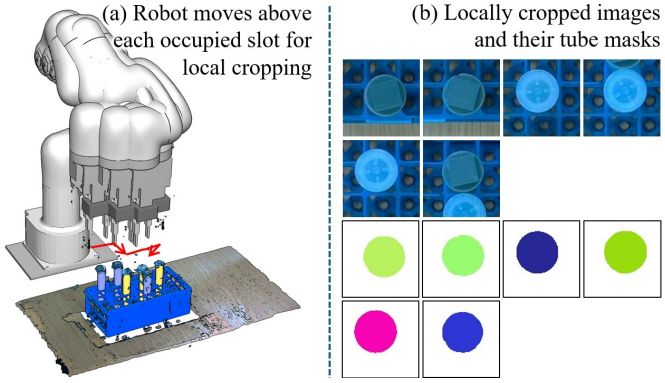

Fig. 4: Robot moving its in-hand camera above each occupied slot to capture locally cropped images.

can be either ResNet-based or ViT-based. Specific choices can be made based on factors such as accuracy requirements, available training data, and computational constraints of the deployment environment. We did not apply normalization after patch projection, as we found no benefit in our task setup and followed common practice in ViT-based architectures where normalization is omitted and deferred to later layers where it can be added as needed. Following the backbone network, the final feature is output through a fully connected layer. This layer downsamples the feature into a 50-dimensional vector, which is reshaped into a 5×10 matrix representing the predicted rack occupation state. The value of each element in the matrix is either 0 or 1. They indicate the absence or presence of a tube. During training, a binary cross-entropy loss function is employed to learn the occupations. We will assess this design in the experimental section by analyzing the influence of different datasets and backbones.

IV. Test Tube Clustering and Labeling

Our method primarily relies on the cap region rather than the entire tube body for classification, as cap color and shape are typically standardized by manufacturers to reflect physical properties such as tube size, geometry, and rack compatibility. These features provide consistent and distinguishable visual cues, making them well-suited for classification. In contrast, the tube body is often transparent or semi-transparent, and its appearance may vary depending on lighting or the contained liquid, which can lead to ambiguity. Cap-based recognition also facilitates integration with top-down camera systems, which are commonly used in lab automation. Once the slot occupation is determined, the tube positions are inferred from the detected rack location and the geometric layout of the slots. However, because tubes may be slightly inclined, directly cropping the slot region is not sufficient for accurate cap extraction. Therefore, we employ a local cropping method to ensure that the cap region is precisely captured for subsequent processing.

A. Cap Extraction

Based on the real-world coordinates of the occupied slots, the robot moves its in-hand camera to a position directly above each slot and captures locally aligned images. These close-up views ensure that the cap region is clearly visible and properly framed for segmentation. We then apply the officially pre-trained SAM2 model to extract one test tube from each image. SAM2 is a prompt-free segmentation framework that generates high-quality masks for arbitrary objects in an image without requiring prior class knowledge. Its transformer-based architecture and streaming memory enable fast and task-agnostic segmentation, making it well-suited for our test tube cap detection. To prevent mis-segmentation, we set the size of the local capture region such that it can accommodate tubes at any inclination within the target slot while avoiding entirely including tubes tilted from adjacent slots. With this size constraint, the segmentation results fall into three possible categories: the tube in the target slot, the part of a tube tilted from an adjacent slot, and the background. These three cases can be distinguished based on the Intersection over Union (IoU) metric of the segmented masks. The segmented mask that is more like a previously seen region in the training dataset is identified as the segmentation result for the tube in the target slot [Note 2]. Fig. 4 illustrates an example of our robot moving above the predicted occupied slots to capture local images.

B. Clustering and Labeling

After segmenting the tube, we use a ViT layer to extract its features and then use HDBSCAN [37] to cluster them. Fig. 5(a) illustrates the process. Here, the Vision Transformer (ViT) is responsible for extracting features, while HDBSCAN is employed to judge if these features are known. When features are known and can be successfully classified, the segmented test tube can be correctly recognized, and its feature is used to update the corresponding class token (the average of all previously seen features belonging to that class). If clustering fails, a new class ID is assigned, and the newly generated class ID, along with its features (token), is added to the known class token repository for future recognition.

Fig. 5(b.1) and (b.2) present two examples. In the first example (b.1), the robot has not encountered any test tubes before and there is no known token. Consequently, all segmentation and clustering results are assigned new IDs and stored in the token repository along with their averaged features. The test tubes are then identified based on these newly assigned IDs.

[Note 2] We use size and aspect ratio filters to eliminate small or narrow masks before comparing IoU.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

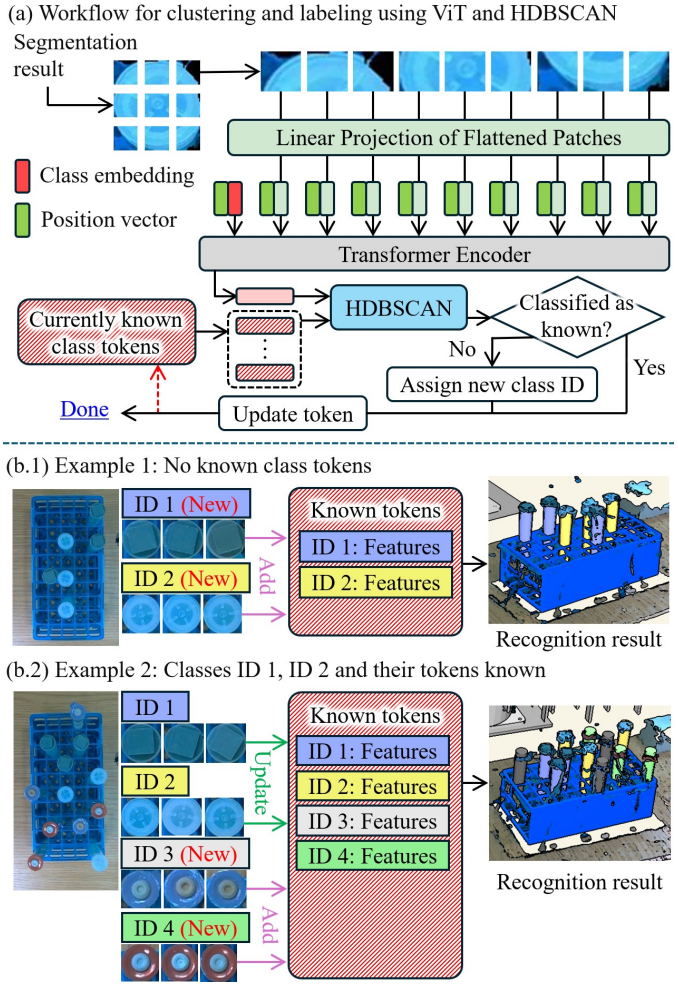

Fig. 5: Classification and labeling workflow.

In the second example (b.2), the robot already has classes ID 1 and ID 2 in its known token repository. When processing newly segmented tubes, the system attempts to identify them using the existing tokens. Only unrecognized tubes are assigned new IDs and added to the token repository. For known tubes, we update their correspondent feature values in the repository by additionally averaging the new observations.

Note that the current work represents a front-end extension and refinement of our previous studies [4, 38]. It provides essential input to the downstream 6D pose estimation and manipulation processes presented therein. Therefore, we do not repeat the details of 6D pose estimation and manipulation planning in this paper. Interested readers are referred to those prior works for a comprehensive implementation.

V. Experiments and Analysis

This section presents the experiments conducted to assess the efficiency and robustness of our proposed methods. First, we evaluate the performance of the slot occupation estimation network by analyzing the influence of different datasets and network backbones. Second, we assess the performance and overall execution costs of extracting test tubes from locally cropped images. Finally, we verify the clustering and labeling process and its effectiveness in dealing with new types of test tubes. Our experimental platform includes a PC, a UFactory XArm Lite 6 robot manipulator, and a Realsense D405 camera mounted on the robot hand. The PC was equipped with an Intel Core i7-13700KF CPU (3.0 GHz), 64 GB of DDR5 memory (4800 MHz), and an NVIDIA GeForce RTX 3060Ti GPU. The WRS robotic system [Note 3] was employed for motion planning and robot control. Our software is run on Python 3.11.9, PyTorch 2.4.1, and CUDA 12.4. Detailed experimental procedures can be found in the supplementary video.

A. Slot Occupation Prediction

We conducted an ablation study on our slot occupation prediction method. Following the approach described above, we collected both simulation and real-world datasets. The simulation dataset consists of 10,000 images, and the real-world dataset contains 400 images. They were designed to incorporate randomness in lighting conditions, camera poses, rendering parameters, and test tube types. The details of the ablation study using the data sets are as follows.

1) Influence of the DA-v2 module

First, we investigated the influence of the DA-v2 module in the designed network using the simulation data. Specifically, we consider two variants of the simulation data: raw and dav2. The raw data consists of pure RGB images, while the dav2 data incorporates the transformation applied by the DA-v2 module. We compared two backbone architectures: ResNet-50 and ViT-tiny. The 10,000 simulation images were split into training, validation, and test sets in a 7:2:1 ratio. Training was performed using the Adam optimizer with a learning rate of 10^{-3}, a 10 step decay, and hyperparameters $\beta_1 = 0.9$. The loss function used was Binary Cross-Entropy Loss. To prevent overfitting, early stopping was applied with a patience of 30 epochs based on the validation loss. As the evaluation metric, we computed the prediction accuracy as the percentage of correctly predicted slots relative to the total number of slots in each rack.

Fig. 6 illustrates the experimental results. In this figure, ViT-tiny (raw) refers to a ViT-tiny backbone trained on the raw RGB dataset, while ViT-tiny (dav2) and ResNet-50 (dav2) denote models trained using data preprocessed by the DA-v2 module. As shown, both ViT-tiny (dav2) and ResNet-50 (dav2) outperform ViT-tiny (raw), indicating that the depth-aware preprocessing provided by DA-v2 enhances feature extraction and overall model performance. To further investigate the impact of dataset size, we conducted an additional comparison under limited-data conditions. Specifically, we selected the first 1,000 images from the 10,000-sample simulation dataset to form a reduced dataset, maintaining the same training, validation, and testing split ratio. The corresponding results are presented as ViT-tiny (dav2_small) and ResNet-50 (dav2_small) in Fig. 6. Notably, even these models trained on the reduced DA-v2 dataset outperform the model trained on the full raw dataset.

These results confirm the effectiveness of the DA-v2 module in enhancing training data quality. The depth-aware transformation improves feature representation and supports better generalization, particularly in data-limited scenarios.

[Note 3] https://github.com/wanweiwei07/wrs

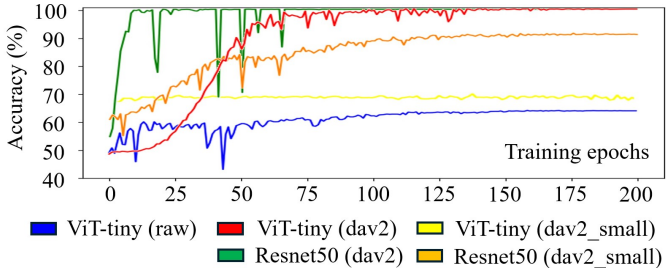

Fig. 6: Training accuracy for ResNet-50 and ViT-tiny on different scales of simulated data.

2) Influence of real-world data

Second, we investigated the influence of real-world data. We collected real-world data following the procedure described in Section III.A.2). The average duration of a single rearrangement and capture step was approximately 18.56 seconds, with the shortest instance taking 13 seconds and the longest 28 seconds. Based on this dataset, we conducted comparative experiments to evaluate model performance under real-world scenarios. Table I shows the results. ResNet-50 (dav2) and ViT-tiny (dav2) correspond to models trained solely on simulation data preprocessed with the DA-v2 module using the 7:2 train-validation split. The 10% test data was ignored. ResNet-50 (dav2+real) and ViT-tiny (dav2+real) indicate models that were first trained on the simulation dataset using the 7:2 split and subsequently fine-tuned using real-world data. All models were evaluated in the real world with random tube arrangement and varying illumination (see details in the supplementary file).

The real-world test was divided into two groups: single tube type and multiple tube types. In the single-tube-type setting, all test tubes in the rack were of the same kind. In the multiple-tube-type setting, two to four different tube types were randomly mixed and inserted into the rack. To vary the level of task complexity, we tested each setting with 10, 20, and 30 inserted test tubes. The recorded accuracy values for the single-tube-type setting represent averages over 25 real-world trials. For the multiple-tube-type setting, each model was tested in 35 trials using racks containing 2, 3, or 4 different tube types. On average, rack detection required 8.17 seconds, while slot occupation prediction took 6.11 seconds.

TABLE I: Merged Results of Real-World Tests

	Single Tube Type			Multiple Tube Types		
Backbone & Dataset	10	20	30	10	20	30
ResNet-50 (dav2)	73.7	65.4	80.1	77.5	71.5	74.5
ViT-tiny (dav2)	88.2	88.4	89.6	92.2	85.7	89.3
ResNet-101 (dav2)	87.5	75.4	89.2	88.0	79.6	84.4
ResNet-50 (dav2+real)	98.7	97.3	97.2	98.4	98.3	97.5
ViT-tiny (dav2+real)	99.0	97.9	97.3	99.0	98.1	97.4
ResNet-101 (dav2+real)	98.8	97.8	96.6	98.8	99.1	97.6

The results show that the inclusion of real data significantly improves the recognition performance. ResNet-50 (dav2+real) is less effective than ViT-tiny (dav2+real) in scenarios with a smaller quantity (e.g., the single tube type with 10 or 20 tubes) but is more robust as the task complexity increases, either by

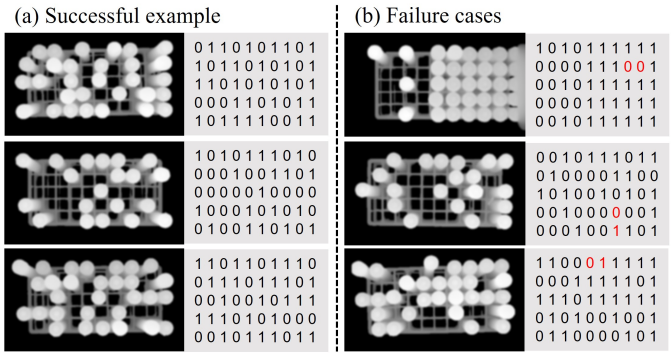

Fig. 7: Three sets of successful and failed predictions for direct observation. (a) Successful predictions. (b) Failures. The red digits highlight the failed slots. The occupation was unrecognized in the first failure and was misrecognized as other slots in the second and third failures.

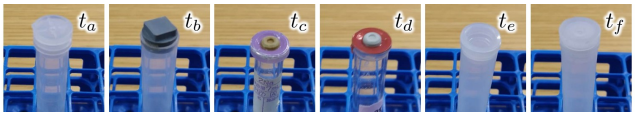

Fig. 8: Six types of tubes are uised in the experiments.

increasing the number of tubes or introducing multiple tube types. ResNet-50 (dav2+real) achieves the best accuracy of 97.5% in the most complex scenarios (multiple tube types with 30 tubes). The accuracy of ViT-tiny (dav2+real) declines from 99.0% at 10 tubes to 97.3% at 30 tubes for a single tube type. A similar trend is observed for multiple tube types (from 99.0% at 10 to 97.4% at 30).

In addition to the ResNet-50 and ViT-tiny, we have included the results of ResNet-101 in Table I to provide some insights to readers interested in the effect of increasing the number of layers in ResNet. The results show that increasing the number of layers has a limited effect on performance. Such limitation may be attributed to the small amount of data used for training.

Due to the higher performance of ResNet-50 (dav2+real), we selected it for later studies. Fig. 7 exemplifies some successful and failed predictions using the ResNet-50 (dav2+real).

B. Test Tube Extraction

This subsection presents experimental results on test tube extraction. Since this process involves sequentially moving an in-hand camera above each tube to crop local images, our experiments focus not only on the precision of cap segmentation but also on the execution time required for image acquisition. We conducted experiments with a diverse number of test tubes (10, 20, 30, 40) and tube types (6 types in total, respectively t_a, t_b, t_c, t_d, t_e, t_f, see Fig. 8) to evaluate the influence of tube quantity on extraction accuracy and execution time. Among the six common types t_a and t_d were prioritized due to higher availability. The others were gradually added in the order: $t_b \rightarrow t_c \rightarrow t_e \rightarrow t_f$.

Table II shows the experimental results. From the second column, we can see that the proposed method can extract tubes with consistently more than 99.8% success rate across all tested numbers of tubes. Notably, the success rate exhibits a

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

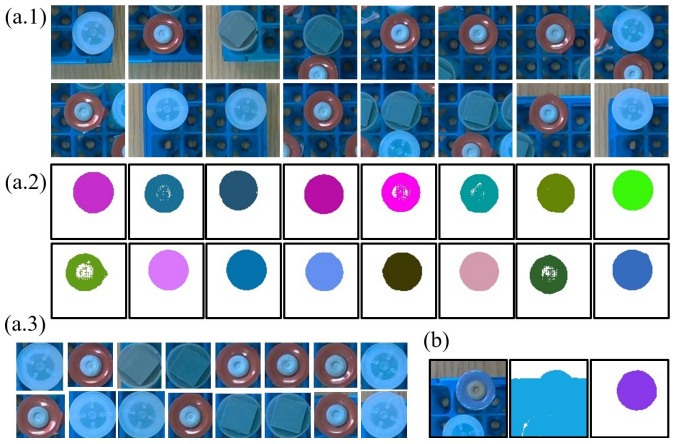

Fig. 9: (a) Successful tube extraction example. (a.1) Locally cropped images. (a.2) Masks predicted by SAM2. (a.3) Segmented caps. (b) Failure example. From left to right: Input, SAM2 (small), SAM2 (large).

slight upward trend from 99.80% to 99.98% when the number of tubes increases from 10 to 30, followed by a minor decrease to 99.92% at 40 tubes. The results indicate that increasing the number of test tubes does not negatively influence the accuracy of our segmentation method. On the other hand, the third and fourth rows of Table II show that both the robot movement time and the segmentation time increase as the number of test tubes grows. Specifically, the robot movement time rises from 12.79 s to 43.78 s, while the segmentation time increases from 3.02 s to 10.28 s. Handling larger numbers of tubes substantially prolongs overall operation duration.

TABLE II: Results of Test Tube Extraction Experiments

	10 Tubes	20 Tubes	30 Tubes	40 Tubes
Succ. Rate (%)	99.80	99.95	99.98	99.92
Rbt Motion Time (s)	12.79	22.94	30.32	43.78
SAM2 Time (s)	3.02	5.27	7.29	10.28

Fig. 9(a) exemplifies several successful examples. It has three rows respectively showing (a.1) locally cropped images, (a.2) segmentation masks, and (a.3) extracted test tubes. When extracting the test tubes, we cropped a square region with a side length equal to the diameter of the mask. As a result, the images in (a.3) exhibit slight variations in size.

Fig. 9(b) shows a failure example. The cap of the test tube in the figure possesses a color similar to the blue rack, which makes it difficult to distinguish the mask of the cap by a segmentation model. The network wrongly extracted the mask shown in the middle as the tube. We can use a larger SAM model to solve this problem, which, however, takes a longer time for inference. The last figure of Fig. 9(b) shows the results of a larger model.

The performance and error analysis validate the effectiveness and robustness of the active local cropping method in extracting tubes, although it has an inevitable increase in execution time when the number of test tubes increases.

TABLE III: Accuracy of Test Tubes Clustering

# Tb	# Types	M Acc. (%)	N Acc. (%)	
			+1	+2
10	2 ($6t_a$+$4t_b$)	100	100 (t_d)	100 (t_c+t_d)
	3 ($4t_a$+$3t_b$+$3t_c$)	95.78	78.95 (t_d)	90.32 (t_d+t_f)
	4 ($2t_a$+$3t_b$+$3t_c$+$2t_d$)	100	92.31 (t_f)	87.10 (t_e+t_f)
20	2 ($6t_a$+$14t_b$)	99.0	100 (t_d)	94.59 (t_d+t_e)
	3 ($4t_a$+$4t_b$+$12t_d$)	100	100 (t_b)	96.55 (t_b+t_f)
	4 ($6t_a$+$4t_b$+$4t_c$+$6t_d$)	91.0	95.0 (t_e)	88.24 (t_e+t_f)
30	2 ($6t_a$+$24t_d$)	100	94.12 (t_b)	97.14 (t_b+t_f)
	3 ($6t_a$+$4t_c$+$20t_d$)	97.84	75.0 (t_f)	82.76 (t_b+t_f)
	4 ($6t_a$+$4t_b$+$4t_c$+$16t_d$)	86.69	100 (t_f)	58.82 (t_e+t_f)

Meanings of abbreviations: # Tb – Number of tubes; M Acc. – Matching accuracy; N Acc. – Accuracy when new tube types are included.

C. Clustering and Recognition

In this section, we evaluate the performance of our clustering and recognition approach applied to the segmented caps. We conducted two types of experiments: the first investigates the accuracy of matching cap features with their corresponding assigned class IDs, while the second assesses performance when new test tubes are introduced, specifically whether the approach can correctly identify new test tube types and assign them new class IDs. We carried out experiments by using a variety of tube numbers and types. The average time required for the clustering and labeling process was approximately 0.50 seconds for 10 test tubes, 0.54 seconds for 20 test tubes, and 0.57 seconds for 30 test tubes. The corresponding matching accuracy and the accuracy of assigning newly introduced tubes to new class labels are summarized in Table III.

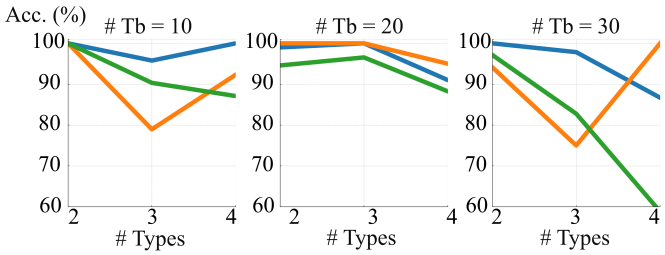

Fig. 10: Blue: M Acc.; Orange: N Acc (N=+1); Green: N Acc (N=+2). See Table III for meanings of abbreivations.

Fig. 10 is a graphical illustration of the results in Table III for readers' convenience. The results indicate that matching performs well when the number of tubes is small or when there are fewer tube types. The lowest accuracy occurs when there are 30 tubes in 4 types, with the matching accuracy declining to only 86.69%. For newly introduced tubes, the system can robustly assign new class labels when the known tube types are limited. However, the performance declined significantly as the number of tube types increased. In the worst case, the probability of successfully assigning a new class label is only 58.82%. We investigated the reasons for these low accuracy values and found that ViT+HDBSCAN easily failed to effectively distinguish between t_c and t_d, and t_a and t_f. Especially in cases where there are many tube types and the variance in inter-type distances is large, the failure rate is high.

A more flexible clustering method needs to be explored.

VI. Conclusions and Future Work

This paper presents a zero-shot pipeline for detecting and recognizing diverse and frequently updated test tubes. Under diverse test tube scenarios, we evaluate the pipeline's various components. The experiment results validate both its effectiveness and adaptability. Currently, the proposed pipeline only applies to top-view perspectives of the test tube rack. Extending to arbitrary overhead views will be our future work.

References

[1] N. Yachie and T. Natsume, "Robotic crowd biology with maholo labdroids," *Nature biotechnology*, vol. 35, no. 4, pp. 310–312, 2017.

[2] I. Holland and J. A. Davies, "Automation in the life science research laboratory," *Front. Bioeng. Biotechnol.*, vol. 8, p. 571777, 2020.

[3] A. Wolf, S. Beck *et al.*, "Towards robotic laboratory automation plug & play: Lapp pilot implementation with the mobert mobile manipulator," in *IEEE SISY*, 2024, pp. 000 059–000 066.

[4] H. Chen, W. Wan *et al.*, "In-rack test tube pose estimation using RGB-D data," in *ROBIO*, 2023, pp. 1–6.

[5] T. T. Mac, D. N. Trinh *et al.*, "The development of robotic manipulator for automated test tube," *Acta Polytech. Hung.*, vol. 21, no. 9, 2024.

[6] R. Girshick, J. Donahue *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014, pp. 580–587.

[7] R. Girshick, "Fast R-CNN," *arXiv:1504.08083*, 2015.

[8] S. Ren, K. He *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *NeurIPS*, vol. 28, 2015.

[9] J. Redmon, S. Divvala *et al.*, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv:1804.02767*, 2018.

[11] ——, "Yolo9000: better, faster, stronger," in *CVPR*, 2017, pp. 7263–7271.

[12] A. Wang, H. Chen *et al.*, "Yolov10: Real-time end-to-end object detection," *arXiv:2405.14458*, 2024.

[13] N. Carion, F. Massa *et al.*, "End-to-end object detection with transformers," in *ECCV*, 2020, pp. 213–229.

[14] Z. Liu, Y. Lin *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021, pp. 10 012–10 022.

[15] N. Ravi, V. Gabeur *et al.*, "Sam 2: Segment anything in images and videos," *arXiv:2408.00714*, 2024.

[16] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[17] J. Canny, "A computational approach to edge detection," *TPAMI*, no. 6, pp. 679–698, 1986.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," *TPAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[20] A. Kirillov, E. Mintun *et al.*, "Segment anything," in *ICCV*, 2023, pp. 4015–4026.

[21] J. Yang, M. Gao *et al.*, "Track anything: Segment anything meets videos," *arXiv:2304.11968*, 2023.

[22] R. Zhang, Z. Jiang *et al.*, "Personalize segment anything model with one shot," *arXiv:2305.03048*, 2023.

[23] J. Cen, Y. Wu *et al.*, "Sad: segment any rgbd," *arXiv:2305.14207*, 2023.

[24] Q. Fan, X. Tao *et al.*, "Stable segment anything model," *arXiv:2311.15776*, 2023.

[25] J. Zhu, Y. Qi, and J. Wu, "Medical sam 2: Segment medical images as video via segment anything model 2," *arXiv:2408.00874*, 2024.

[26] J. Boyd, "Robotic laboratory automation," *Science*, vol. 295, no. 5554, pp. 517–518, 2002.

[27] T. Chapman, "Lab automation and robotics: Automation on the move," *Nature*, vol. 421, no. 6923, pp. 661–663, 2003.

[28] Á. Wolf, S. Romeder-Finger *et al.*, "Towards robotic laboratory automation plug & play: Survey and concept proposal on teaching-free robot integration with the lapp digital twin," *SLAS Technol.*, vol. 28, no. 2, pp. 82–88, 2023.

[29] P. Shiri, V. Lai *et al.*, "Automated solubility screening platform using computer vision," *iScience*, vol. 24, no. 3, 2021.

[30] J. Ochs, F. Biermann *et al.*, "Fully automated cultivation of adipose-derived stem cells in the stemcelldiscovery—a robotic laboratory for small-scale, high-throughput cell production including deep learning-based confluence estimation," *Processes*, vol. 9, no. 4, p. 575, 2021.

[31] S. Azizi, E. Asadi *et al.*, "Autonomous hyperspectral characterisation station: Robot aided measuring of polymer degradation," *TASE*, 2024.

[32] R. C. Luo, P.-J. Lai, and V. W. S. Ee, "Transparent object recognition and retrieval for robotic bio-laboratory automation applications," in *IROS*, 2015, pp. 5046–5051.

[33] M. De Jesus, J. Norberto *et al.*, "Detection and manipulation of test tubes in the pre-analytical phase of the laboratory sector," in *CASE*, 2024, pp. 1672–1677.

[34] S. Ge, B. Hou *et al.*, "Pixel-level collision-free grasp prediction network for medical test tube sorting on cluttered trays," *RA-L*, 2023.

[35] L. C. O. Tiong, H. J. Yoo *et al.*, "Machine vision-based detections of transparent chemical vessels toward the safe automation of material synthesis," *npj Comput. Mater.*, vol. 10, no. 1, p. 42, 2024.

[36] L. Yang, B. Kang *et al.*, "Depth anything v2," *arXiv:2406.09414*, 2024.

[37] L. McInnes, J. Healy, S. Astels *et al.*, "HDBSCAN: Hierarchical density-based clustering." *JOSS*, vol. 2, no. 11, p. 205, 2017.

[38] H. Chen, W. Wan *et al.*, "Robotic test tube rearrangement using combined reinforcement learning and motion planning." *arXiv:2401.09772*, 2024.