Automated Individualization of Object Detectors for the Semantic Environment Perception of Mobile Robots

Christian Hofmann[a], Christopher May[b], Patrick Ziegler[c],
Iliya Ghotbiravandi[d], Jörg Franke[e] and Sebastian Reitelshöfer[f]
Institute for Factory Automation and Production Systems, Friedrich-Alexander-Universität Erlangen-Nürnberg, Egerlandstraße 7, 91058 Erlangen, Germany

Keywords: Open Vocabulary Object Detection, Pseudo-Labeling, Large Language Model, Vision Language Model.

Abstract: Large Language Models (LLMs) and Vision Language Models (VLMs) enable robots to perform complex tasks. However, many of today's mobile robots cannot carry the computing hardware required to run these models on board. Furthermore, access via communication systems to external computers running these models is often impractical. Therefore, lightweight object detection models are often utilized to enable mobile robots to semantically perceive their environment. In addition, mobile robots are used in different environments, which also change regularly. Thus, an automated adaptation of object detectors would simplify the deployment of mobile robots. In this paper, we present a method for automated environment-specific individualization and adaptation of lightweight object detectors using LLMs and VLMs, which includes the automated identification of relevant object classes. We comprehensively evaluate our method and show its successful application in principle, while also pointing out shortcomings regarding semantic ambiguities and the application of VLMs for pseudo-labeling datasets with bounding box annotations.

1 INTRODUCTION

Autonomous mobile robots (AMRs) are used for various tasks in different domains such as industry, healthcare or construction. Recently, powerful machine learning approaches such as Large Language Models (LLMs) and Vision Language Models (VLMs) have been used to enable them to solve complex tasks autonomously.

However, these machine learning approaches require extensive computing resources. On the one hand, they can be executed externally, i.e., not on the robot itself, and are then accessible to the robot via communication systems. This requires the robot to have a reliable connection to the external systems in order to maintain its autonomous capabilities. In addition, there may be a noticeable delay in the robot's behavior and actions due to external processing. On the other hand, the required computing hardware could be integrated on-board the robot. This in turn leads to high power consumption and increased weight. Both alternatives are not particularly suitable for some types of mobile robots. AMRs like drones

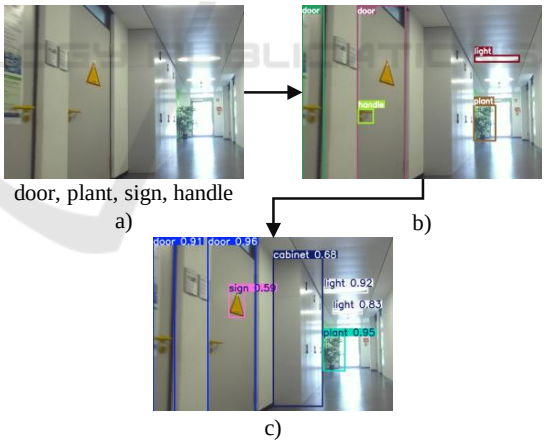

Figure 1: Simplified illustration of our proposed approach: a) First, object classes are identified in an image dataset and merged to consistent labels using an LLM, b) using a VLM, pseudo-labels for the dataset based on the identified object classes are generated and c) using the pseudo-labeled dataset, a lightweight object detector is trained.

[a] https://orcid.org/0000-0003-1720-6948
[b] https://orcid.org/0009-0003-2571-3461
[c] https://orcid.org/0009-0009-8234-8038
[d] https://orcid.org/0009-0007-8790-1782
[e] https://orcid.org/0000-0003-0700-2028
[f] https://orcid.org/0000-0002-4472-0208

Hofmann, C., May, C., Ziegler, P., Ghotbiravandi, I., Franke, J. and Reitelshöfer, S.
Automated Individualization of Object Detectors for the Semantic Environment Perception of Mobile Robots.
DOI: 10.5220/0013162500003912

In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 2: VISAPP*, pages 851-862
ISBN: 978-989-758-728-3; ISSN: 2184-4321

or transportation robots often don't have the possibility to carry powerful computing hardware that would significantly decrease payload capacity and operating time. A delayed reaction due to off-board computations may also lead to severe safety issues.

A crucial task and capability of today's AMRs is environment perception. In the state of the art, AMR mainly use semantic perception approaches to detect relevant objects in their environment based on camera or Lidar data as sensory input. However, AMRs often operate in changing environments. They should therefore also be able to continuously adapt their software and capabilities to changing environments.

Consequently, there is a need for enabling the automatic individualization of a robot's semantic perception of the environment, with the adapted perception software being executable on board of an AMR (Hofmann et al., 2023).

In this paper, we investigate how fully automated environment-specific object detector adaptation can be realized. We focus on a scenario where a robot is deployed in a corridor environment. A lightweight object detector should be automatically adapted to this specific environment by the proposed method, as illustrated in Figure 1. Our goal in this paper is to detect relevant objects in the specific environment as good as possible, i.e., to get an individualized object detector for the environment. It is not our goal to get a generalized object detection performing well in other scenarios and environments. If the robot moves to another environment or the environment changes, the detector should be adapted accordingly by the approach. From our point of view, adapting the detector is thus equivalent to individualizing the detector for the respective robot and environment.

The main contributions of this paper are:

- We propose an approach for fully automated environment-specific object detector adaptation using LLMs and VLMs.
- Our method also automates the identification of relevant object classes compared to related work.
- We present a comprehensive evaluation of each step of our proposed method with a focus on adaptation to a specific environment.

The paper is structured as follows: In section 2 related work is presented and discussed. The proposed method for automatic environment-specific individualization of object detectors is introduced in section 3. In section 4, an implementation of our approach is described, evaluated and discussed. We summarize the results and key findings in section 5.

2 RELATED WORK

In recent years, machine learning and especially neural networks have revolutionized object detection in images. Following, we provide a brief overview of the work in this field that is relevant to our approach.

2.1 Object Detection

Object detection describes the task to discover objects in images, to determine their position in the image and to classify the discovered and localized objects. The results of an object detection task are often bounding boxes enclosing the area in which a specific object is visible as well as a label classifying the object in the bounding box.

In the state of the art, convolutional neural networks (CNNs) are commonly used for object detection. CNNs for object detection can be generally divided into two categories: one-stage detectors and two-stage detectors. One-stage detectors solve the whole object detection task (localization and classification) within a single CNN, while two-stage detectors use one CNN for localizing possible objects in the image and another CNN at second stage verifies and classifies the object in the prior identified areas (Zhao et al., 2019). Prominent examples are YOLO as a single-stage object detector and Faster-RCNN as a two-stage detection approach (Zhao et al., 2019).

A further and more recent powerful approach for object detection are Vision Transformers. In contrast to CNNs, Vision Transformers work on an attention-based encoder-decoder architecture (Han et al., 2023).

CNNs and Vision Transformers have in common that they are trained with a large amount of annotated training data for a specific set of classes to be detected (Gao et al., 2022), (Hofmann et al., 2023). The classes are normally selected manually according to a task to be solved. Changing or adding classes requires retraining the whole network with an adapted dataset (Gao et al., 2022). However, the research area of continuous learning aims to overcome this problem (Lesort et al., 2020). Nevertheless, mobile robots often encounter a large variety of objects, depending on the environments they are deployed in. Consequently, it is not possible to train all possible object classes a robot could encounter in advance of deployment (Hofmann et al., 2023). This aspect is addressed in the research areas of Open World Object Detection, Zero-Shot Object Detection and Open Vocabulary Object Detection, among others (Joseph et al., 2021), (Firoozi et al., 2024), (Zhu and Chen, 2024).

Next to CNNs and Vision Transformers, Vision Language Models (VLMs) are a recent powerful ap-

proach for object detection, including Zero-Shot Object Detection and Open Vocabulary Object Detection (Firoozi et al., 2024). VLMs take both an image and a text prompt as input to solve varying tasks. In addition to object detection, examples for such tasks are semantic segmentation or image editing. However, in the context of practical use on robots, VLMs have problems such as high inference times, granularity (when should parts of an object be considered as one object or as individual objects) and lack of uncertainty quantification, e.g., for detecting hallucinations of the network (Firoozi et al., 2024).

2.2 Pseudo-Labeling for Open World Learning Scenarios

In the investigated setting, more or less unlimited access to images of the objects to be detected can be assumed. The images are provided by the robot operating in the environment. However, it is often not known in advance which classes are present and relevant in the specific environment the robot is deployed to. This represents an Open World Learning Scenario (Wu et al., 2024). Further, we do not require the detector adapdation process to be performed in realtime. Thus, we assume access to powerful hardware on external computers for adapting the detector. However, an individualized object detector with low inference times should be finally available for the robot.

This setting is well suited for the use of pseudo-labeling (Zhu and Chen, 2024), (Wu et al., 2024), (Kim et al., 2024). Pseudo-labeling and pseudo-labels are often used in Open World Learning Scenarios, for Domain Adaption and Novel Class Discovery. Example works are (Kim et al., 2024), (Cheng et al., 2024), (Wang et al., 2023), (Vaze et al., 2022), (Gao et al., 2022), (Sun et al., 2022) and (Han et al., 2022).

The benefit of pseudo-labels was introduced in 2013 (Lee et al., 2013). Lee et al. define pseudo-labels as ”target classes for unlabeled data as if they were true labels” (Lee et al., 2013, p. 3). This means, they assume a predicted label for unlabeled data as true and reuse the newly generated label for fine-tuning their model in a semi-supervised training.

The recent approaches presented in (Gao et al., 2022) and (Kim et al., 2024) successfully automatically generate pseudo-labels for datasets using VLMs, which fits our needs for the described scenario well. (Gao et al., 2022) use the localization ability of pre-trained VLMs to generate bounding-box annotations as pseudo-labels from large scale image-caption pairs. (Kim et al., 2024) use a VLM to verify the detections of an object detector and thus generate valid bounding-box annotations. A quite simple approach to extract pseudo-labels is the Roboflow autodistill framework (Gallagher, 2023). It uses foundation models like GroundingDINO (Liu et al., 2023), (IDEA Research, 2023) to detect or segment objects in an image dataset based on an input text prompt comprising the object classes of interest. Based on the detections or segmentation masks created in this way as pseudo-labels, smaller networks like YOLOv8 (Jocher et al., 2023) can be automatically trained.

The mentioned approaches require a human in the loop, either for selecting the object classes to be detected or to prepare and provide suited datasets for training. In the context of AMR, the goal should be to remove the human from the loop at as many points as possible in order to increase the robot’s autonomy (Hofmann et al., 2023).

To overcome this issue and enable fully automated adaptation of an object detector to a specific environment based on pseudo-labeling, we propose and investigate an approach that uses VLMs and LLMs to automatically extract relevant object classes from images of the specific environment. Compared to the work in (Kim et al., 2024) this paper presents and investigates the automated retrieval of object classes relevant for a specific environemt. Building upon the concpets introducted in (Hofmann et al., 2023), we propose an improved and more flexible pipeline for automated object detector approach. Further, inspired by the autodistill framework (Gallagher, 2023), we investigate and critically evaluate the training of a lightweight object detector based on a pseudo-labeled training dataset with bounding-box annotations.

3 APPROACH FOR AUTOMATED OBJECT DETECTOR INDIVIDUALIZATION

Our method for fully automated environment-specific object detector adaptation is shown as flowchart in Figure 2. Following the architecture for mobile robot software adaptation presented in (Hofmann et al., 2023), the adaptation pipeline is intended to be executed ”offline”. Offline in this context means that the robot is either not in operation at the time of adaptation or the adaptation process is performed on an external computer.

Following we describe the steps of the adaptation pipeline in detail.

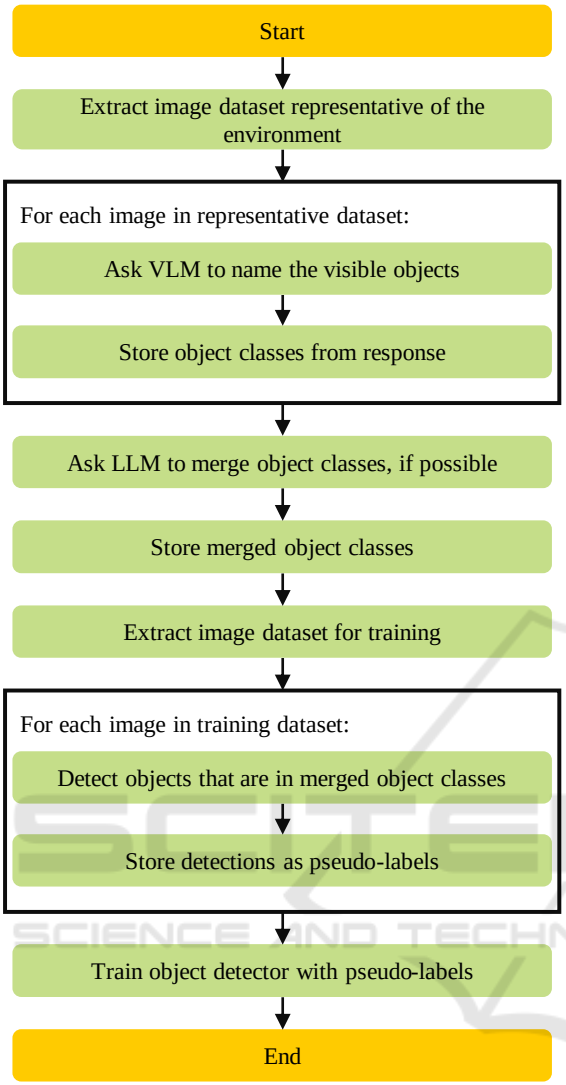

Figure 2: Flowchart of the investigated pipeline for fully automated environment-specific object detector adaptation.

3.1 Representative Image Dataset Generation

The first step is to extract sample images that show the specific operating environment. Therefore, the robot should capture images with its camera when operating in the environment. There is no additional effort in saving images or videos during operation. Depending on the size of the environment and the robot's operating time, it is not necessary to use all images. For the following pipeline, several images that represent and depict the environment and objects therein well are sufficient. Thus, a time- or location-based extraction of sample images can be used to automatically obtain a dataset representative of the environment.

3.2 Automatic Object Classes Identification

To obtain a fully automated process, we use an VLM to identify and name objects in the environment. Each image of the representative dataset is input into a VLM with the task of naming all objects visible in the image. The response, i.e., the visible object classes output by the VLM, is saved for each image.

3.3 Object Classes Merging

Since the example images depict the same environment, it is probable that objects of the same class are visible multiple times and consequently the object class is output multiple times. However, a possible outcome could be a slightly varying name or a different class specification (e.g., fire-extinguisher and extinguisher). We introduce a step to merge such class variations and synonyms that name the same object class into one label. Since this is a text-based reasoning task, we propose LLMs to be used for this task. The outcome is a list of object classes that are present in the specific environment.

Note that also manually provided labels could be included in this process to make the LLM use certain labels for object classes.

3.4 Dataset Pseudo-Labeling

Based on images of the specific environments and the list with the merged object class labels, VLMs can be used to generate pseudo-labels (Gao et al., 2022), (Gallagher, 2023), (Kim et al., 2024).

At this step, we introduce a new "training dataset" that may differ from the prior used representative dataset. The representative dataset should just provide a good overview over the environment. However, the training dataset should be later used for training an object detector, thus contain more images and samples of the objects to be detected from different perspectives. To create the image dataset, again a time- or location-based extraction policy can be used. In this process, each image in the dataset is evaluated by a VLM with the merged list of identified object classes as text input. The outcome are pseudo-labels in form of bounding boxes with object-class annotations for the whole dataset.

This process could also be easily adapted for semantic segmentation, by using VLMs that output segmentation masks.

3.5 Object Detector Training

The final step of the proposed method is the training of a suited object detection network based on the pseudo-labeled training dataset. This step could be seen as a kind of "knowledge distillation" via the pseudo-labels. The pseudo-labeled dataset thus represents an abstraction of the distilled knowledge of the VLM. Thus, the flexible knowledge distillation to almost any other object detection network is enabled.

Additionally, the pseudo-labeled dataset allows easy integration of the data from multiple robots operating in the same environment. All image data provided can be processed by the method and accumulated in an overall pseudo-labeled dataset for the environment, cf. (Hofmann et al., 2023).

The abstraction of the pseudo-labeled dataset also allows including further data, like from datasets available online or artificially created datasets, e.g., using simulations or generative neural networks. This enables a more robust object detector training while also focusing the perspective of the robot on the specific environment. In addition, object classes not or not yet present could be thus included.

Moreover, the approach enables flexible adaptation to new data collected in the environment. This allows the automated incremental and continuous integration of changes in the environment, e.g., new or disappearing object classes. Beyond this, the dataset can be continuously improved and expanded with newly collected images using the method presented.

4 IMPLEMENTATION, EVALUATION AND DISCUSSION

This section first presents an exemplary implementation of the proposed approach. Further, the evaluation of the proposed approach based on the implementation is presented and discussed.

4.1 Exemplary Implementation

We implemented the processing pipeline with GPT-4o (OpenAI, 2024) as LLM and VLM to automatically identify the object classes visible in the representative image dataset and to merge the object classes into consistent labels. We chose GPT-4o since it has the capability to process the image and text prompt for object class identification (VLM) as well as the completely text-based prompt for merging the identified object classes (LLM). However, other LLMs and VLMs could be used for these tasks.

We used the following prompt to automatically identify the relevant object classes with GPT-4o:

```
I need help labeling objects in an image for
training an object detector.

Instructions:
1. Identify and label all visible objects,
including those in the background.
2. Provide one label per object class,
even if multiple instances are present.
3. Use concise, single-word labels that align
with standard object categories from
datasets like COCO (e.g., chair, table,
bicycle, person, bottle).

Output Format:
Respond in a list, all lowercase:
- label_1
- label_2
- label_3
- ...
- label_n
```

The prompt is kept quite general in order to identify as many object classes as possible in the images of the representative dataset. A practical application where it is advantageous to detect as many object classes as possible is localization using semantic maps, e.g., presented in (Zimmerman et al., 2023). However, the prompt could be adapted so that specifically objects required for a certain task should be identified.

The automatic class merging was performed with the following prompt and GPT-4o:

```
You are given a list of object labels.
Your task is to identify if any of these
labels can be grouped under a single label.

Instructions:
- Identify labels that are synonyms,
variations, or refer to the same object
category.
- Only suggest merges when the labels
unambiguously belong to the same object
category.
- If it makes sense, prefer to group labels
under existing labels from the provided list.
- Group all labels referring to
structural elements of buildings
(e.g., "wall," "ceiling") or
rooms (e.g., "kitchen," "bathroom")
under the label "building".
- Keep your response short and concise,
listing only the labels to be merged and
their target category.

Output Format:
For each group, use the following format:
[label_1, label_2, ...] -> category_name

List of object labels:
{', '.join(labels)}
```

Again, the prompt is kept quite general. Nevertheless, we have introduced the instruction to group all labels referring to structural elements of buildings (e.g., "wall," "ceiling") or rooms (e.g., "kitchen," "bathroom") under the label "building". This provides the option of simply deleting such object classes and labels (cf. subsection 4.3).

To obtain the pseudo-labels for the images in the training dataset on the basis of the previously identified and merged object classes, the VLM GroundingDINO is used (Liu et al., 2023) (IDEA Research, 2023). Pseudo-labeling using GroundingDINO (and other foundation models) is also presented in the autodistill framework (Gallagher, 2023). Again, other approaches with similar capabilities as GroundingDino may be used for this task.

As exemplary lightweight object detection approach to be deployed on a robot, YOLOv8s (Jocher et al., 2023) is trained using the training dataset with annotations from the pseudo-labeling step.

4.2 Evaluation Setup

Our evaluation setup consists of three consecutive runs in which a robot moved through a corridor (25 m long) in our institute. The corridor is the environment to which the robot, or more precisely its object detector (YOLOv8s), is to be automatically adapted. During each run, a video was captured with a camera on the robot (approx. 1 m above the floor). The videos therefore show the robot's perspective of its environment. The videos were recorded at a frame rate of 15 Hz and a resolution of 1920 x 1080 pixels. During each run, the robot moves from one end of the corridor to the other, turns around and moves back to its starting position. The runs took place on different days, so the robot's trajectory and the lighting conditions in the videos differ slightly. However, the environment itself and the objects in it did not change significantly between the runs.

Our expectation for this evaluation setup is an increasing adaptation, i.e., improved object detection results, after each video has been processed and the new training data has been used to train the YOLOv8s model.

In the next subsection, the evaluation results of the proposed method are presented and discussed in detail based on the following evaluation steps:

1. **Identification of Object Classes Based on the Representative Dataset.** We extracted an image from each of the three videos approximately every four seconds. This results in 30 images in the representative dataset for run 1, 33 images for run 2 and 29 images for run 3. We sent each image to GPT-4o via the API together with the prompt described above. We saved the response, i.e., the identified object classes, for each image for evaluation.
2. **Merging of Identified Object Classes.** We passed the list of identified object classes from each representative dataset with the previously described merge prompt to GPT-4o, again via API. This merging step was performed five times for each list of indentified object classes and each response was stored.
3. **Pseudo-Labeling Using GroundingDINO.** For evaluating GroundingDINO for pseudo-labeling, we extracted 100 images from all three videos (run 1: 34 images, run 2: 33 images, run 3: 33 images). We manually annotated the bounding boxes for each object class in the merged object classes (previous step) in these 100 images. This hand-labeled dataset represents our "Ground-Truth Dataset". We apply GroundingDINO to detect the merged object classes in these 100 images, i.e., to generate the pseudo-labels for these images. We compare the pseudo-labels generated by GroundingDINO with varied parameters (text threshold and box threshold) to the manual annotations. This evaluation step provides a parameter selection guideline for the following pseudo-labeling of the training datasets.
4. **Performance of YOLOv8s Trained on Pseudo-Labeled Datasets.** In the final evaluation step, we investigate the result of training YOLOv8s with pseudo-labeled training datasets. Therefore, we trained YOLOv8s using the training datasets that were generated with different parameters settings of GroundingDINO. Additionally, we study whether an adaptation or improvement effect can be observed, when the amount of training data is increased with the newly captured images after each run. Further, we investigate the effects of a differing image amount included in the training dataset. Finally, we also investigate combining a pseudo-labeled training dataset with a dataset that is online availabe.

4.3 Evaluation Results and Discussion

Following, the results of the four previously described evaluation steps are presented and discussed.

1. Identification of Object Classes Based on the Representative Dataset

We passed all images of the three representative datasets with the previously described prompt (cf. subsection 4.1) to GPT-4o. The following classes

with the number of occurrences in brackets behind them were identified by GPT-4o for the three representative data sets:

- Run 1: door (30), light (23), wall (16), ceiling (15), floor (15), plant (11), sign (4), handle (4), cabinet (3), mat (2), table (2), hallway (2), bicycle (1), frame (1), poster (1), sofa (1), extinguisher (1), fire_extinguisher (1), ceiling_light (1), hallway (2).
- Run 2: door (33), light (21), ceiling (18), floor (16), wall (13), plant (13), sign (7), cabinet (7), handle (3), table (2), poster (2), hallway (2), mat (1), extinguisher (1), ceilinglight (1), ceilinglamp (1), camera (1), chair (1), window (1).
- Run 3: door (26), light (17), ceiling (13), plant (11), wall (10), floor (10), hallway (4), poster (3), handle (3), cabinet (3), sign (2), ceiling_light (2), table (1), extinguisher (1), fire_extinguisher (1), corridor (1).

The identified classes cover the objects present in the representative dataset and the environment quite well (cf. Figure 1). The bicycle (run 1) and the camera (run 2) are the only clearly false class identifications. There are also a few missing objects like a first aid kit, a switch and a fire alarm button. Some classes like the class window, were only identified seldom, although they are quite often present in the real environment and also in the representative datasets. Additionally, some variations of the same class are obvious (light and ceiling_light, extinguisher and fire_extinguisher). The following class merging step was introduced to get rid of such variations and synonyms.

Further, some classes like ceiling, wall and hallway were correctly identified but from our point of view they are most times not relevant for a mobile robot's object detection. Consequently, we added an instruction in the following class merging prompt to group such structural elements and room names into a group "building" to easily remove them from the next process steps (cf. subsection 4.1).

Another point we observed is that GPT-4o's responses to the exact same prompt with the exact same image are not always the same (also when sent in immediate succession). Having multiple images of the same environment in representative dataset, this effect can be equalized in parts. However, when there are only few objects of a class in an environment or only few images in the representative dataset, some object classes will be possibly not identified. With one or more robots that repeatedly capture data of the environment, the proposed method is able to integrate also "rare" objects.

Moreover, our prompt is quite general, which may lead to semantic ambiguities. Considering the class "door" in our evaluation scenario, it is clear that doors to rooms are present in a corridor. However, the cabinets in the investigated corridor have doors as well. Such semantic ambiguities and uncertainties regarding object granularity can cause problems in the following adaptation process and thus for the robot's object detection. By a more detailed specification of the robot's task or even relevant objects in the prompt, the object classes in the representative dataset could be identified more specifically.

Summed up, the object class identification using GPT-4o as VLM based on the representative datasets was successful but some aspects like semantic ambiguities and uncertainties regarding object granularity have to be investigated and improved in future work. In addition, the prompt may be improved to identify all classes in the representative dataset and the environment at a high rate and reliability.

2. Merging of Identified Object Classes

To overcome the problem of variations and synonyms for the same class and to easily remove unwanted classes, we included a merging step for the identified object classes. The merging step is fully text based as described in subsection 4.1.

In this step, we face the same problem as in the previous step, namely that the responses of GPT-4o are differing, even though the input does not change. This may lead to differing class merges after each run. To evaluate this effect, we asked GPT-4o to merge the classes five times in immediate succession for each run.

As an example, the results for run 1 are shown in Table 1. It is noticeable, that no response and thus class merging scheme is identical. We also observed this for the two other runs. We therefore do not believe that class merges should be derived from a single response of GPT-4o.

However, by selecting merges that are named multiple times or most times, a quite good merging scheme can be automatically constructed. Using this procedure for the identified and merged labels of the three runs, the resulting merge schemes are displayed in Table 2.

For run 2 we multiple times observe a merge that was not intended: "[door, handle] $\rightarrow$ door". However, this merge is not obviously false. Here we face the granularity problem, as described in (Firoozi et al., 2024). Again, more details about the robot's task and the object classes required for the task could help to overcome this problem.

Overall, this kind of statistical evaluation to create the merge schemes is quite successful. Variations of the same class, e.g., light and ceiling_light are suc-

Table 1: Responses from GPT-4o to the requests for merging the identified object classes in the representative dataset from run 1. Class merges that were mentioned more than once are highlighted in bold.

Merge Request	Response
1	[ceiling, floor, wall, hallway, cabinet] → building **[fire_extinguisher, extinguisher] → extinguisher**
2	**[ceiling, floor, wall, hallway] → building** **[extinguisher, fire_extinguisher] → extinguisher**
3	**[ceiling, floor, wall, hallway] → building** **[extinguisher, fire_extinguisher] → fire_extinguisher** **[light, ceiling_light] → light**
4	**[ceiling, floor, hallway, wall] → building** [cabinet, handle] → door **[fire_extinguisher, extinguisher] → fire_extinguisher**
5	[door, handle] → door [ceiling, wall, floor, hallway, kitchen, bathroom] → building **[light, ceiling_light] → light** **[extinguisher, fire_extinguisher] → extinguisher**

Table 2: Multiple times named class merging schemes for the three runs.

Run	Merge Scheme
1	[ceiling, floor, wall, hallway] → building
	[extinguisher, fire_extinguisher] → extinguisher
	[light, ceiling_light] → light
2	[wall, floor, ceiling, hallway, corridor] → building
	[light, ceilinglight, ceilinglamp] → light
	[handle, door] → door
3	[wall, floor, ceiling, hallway, corridor] → building
	[extinguisher, fire_extinguisher] → extinguisher
	[light, ceiling_light] → light

cessfully merged. The instruction to sort out room names and structural building elements works well.

After merging the classes for each run, next to the building group, object classes that were named only once are sorted out assuming that they are false object identifications like the class "bicycle". However, in this process, the correct classes sofa (run 1), chair (run 2), and window (run 2) are removed. Nevertheless, these classes would be added again, if they are identified in further runs. Also, an automatic removal of classes that are not detected anymore after some runs is possible.

Note that it would depend on whether the merge schemes are stored after each run for the following runs or not, whether the class handle is included in the classes after run three. We decided to include it for further evaluation. Since the environment did not change significantly between the runs, as expected no new classes were discovered in the consecutive runs.

Finally, after run three, the following ten object classes would have been identified for our corridor environment: door, light, plant, sign, handle, cabinet, mat, table, poster, extinguisher.

3. Pseudo-Labeling Using GroundingDINO

As described in subsection 4.2, we manually annotated all instances of the ten identified object classes with bounding boxes in 100 images extracted from the three runs.

We investigate the use of GroundingDINO for generating pseudo-labels. GroundingDINO has the parameters text_threshold and box_threshold to only output boxes and words, i.e., labels, that have a similarity above the threshold (IDEA Research, 2023).

For gaining insight into the influence of these parameters on the pseudo-labeling, we run a parameter sweep on the two parameters. As the values suggested on GroundingDINO's GitHub page are box_threshold = 0.35 and text_threshold = 0.25 (IDEA Research, 2023), we start the parameter sweep at 0.15

for both parameters and increase the values independently by 0.05 during the sweep until both reach the value 0.4. During the parameter sweep, for each parameter constellation the 100 images are evaluated by GroundingDINO with the ten classes as text input. The output are the detection results for these classes, thus our pseudo-labels.

The pseudo-labeling results of GroundingDINO are compared to our manually-annotated labels in Table 3. The table shows the results of parameter settings where the box threshold value is equal to the text threshold value, as they provide a good overview of the behavior of GroundingDINO when the parameters are varied. Other parameter settings where the box threshold value does not equal the text threshold value fit well with the pattern shown.

In the evaluation, a pseudo-label is evaluated as correct (true positive) if the class matches the manually annotated class and if the pseudo-label bounding box has an intersection over union (IoU) of more than 0.5 with the manually annotated bounding box.

Table 3: Results of comparing the predicted pseudo-labels of GroundingDino to our manual labels for different box and text thresholds.
Threshold: box threshold / text threshold, TP: True Positive, FP: False Positive, FN: False Negative, P: Precision, R: Recall, F1: F1-Score.

Threshold	TP	FP	FN	P	R	F1
0.10/0.10	1017	3235	1191	0.24	0.46	0.31
0.15/0.15	1042	1831	1166	0.36	0.47	0.41
0.20/0.20	999	890	1209	0.53	0.45	0.49
0.25/0.25	840	363	1368	0.70	0.38	0.49
0.30/0.30	570	117	1638	0.83	0.26	0.39
0.35/0.35	361	41	1847	0.90	0.16	0.28
0.40/0.40	218	17	1990	0.93	0.10	0.18

It is obvious, that the pseudo-labeling is not really good. At low thresholds, 0.1 and 0.15, the number of true positives is quite high while also providing a huge number of false positives. At medium thresholds, 0.2, 0.25 and 0.3, the number of false positives decreases significantly while the number of true positives also decreases moderately. At high threshold values, 0.35 and 0.4, the number of true positive and false positive results decreases further. At high threshold values, however, the recall, i.e., the number of true positives in relation to all ground truth labels, is quite low.

We observed almost the same results when applying the autodistill framework (Gallagher, 2023) using GroundingDINO.

For the evaluation of training YOLOv8s based on the pseudo-labeled datasets, we chose two parameter options for pseudo-labeling with GroundingDINO:

- **0.2 / 0.2**: These parameter settings show the highest F1 score with a high number of true positive but also a nearly equal number of false positives.
- **0.3 / 0.3**: These settings provide a lower but still acceptable number of true positives but also a significantly decreased number of false positives.

Our evaluation includes ten different object classes, so we assume that this parameter selection is transferable to similar settings. However, describing the objects to be detect more detailed in the input prompt, e.g., by using attributes of the objects, may improve the results of GroundingDINO an thus other parameter setting may provide better results. Further, since our evaluation environment comprises quite common object classes, we expect even worse results when using GroundingDINO to pseudo-label less common objects, e.g., from an industrial environment.

4. Performance of YOLOv8s Trained on Pseudo-Labeled Datasets

For training YOLOv8s, we first created training datasets (cf. section 3). Similar to the creation of the representative datasets, we did not use all images from the videos recorded in the three runs, but extracted images in specific time steps. The following evaluation is mainly based on a training dataset that contains every fifth image extracted from the videos, which corresponds to a frame rate of 3 Hz. Thus, we want to avoid including too many similar perspectives of the environment and objects therein.

We also created a training dataset based on extracting every fifteenth frame, i.e., even less images. Further, we also investigate adding other data than the self-captured image to the training dataset. Therefor we extracted the images and bounding box annotations for the classes ”door” and ”handle” from a dataset available online (Arduengo et al., 2021).

The training data pseudo-labeling for our datasets was performed according to our automated process using GroudingDINO, with the threshold settings 0.2 / 0.2 and 0.3 / 0.3. We did not manually adapt or improve the pseudo-labeling-based annotation results, but used the detections of GroundingDINO as is as pseudo-labels. The text input prompt for GroundingDINO were again the previously identified classes: door, light, plant, sign, handle, cabinet, mat, table, poster and extinguisher.

After training YOLOv8s using the newly created training datasets, we evaluated the detection performance using the 100 manually annotated images. Those images were included in none of the train-

ing datasets. Nevertheless, they look quite similar to training data, since they display exactly the same environment. However, we do not consider this fact as negative because our goal is the adaption or individualization to this specific environment. Thus, we only require good results for this specific environment.

The results of training YOLOv8s with the training datasets created from the videos of all three runs (1+2+3) with every fifth image extracted (3Hz) using different settings of GroundingDINO for pseudo-labeling (thresholds 0.2/0.2 or 0.3/0.3) are shown as first results in Table 4. The data in Table 4 shows, that YOLOv8s trained with pseudo-labels based on GroundingDINO thresholds at 0.3 (YOLOv8s@0.3) provides better results from a robot's or roboticist's perspective. The mAP is higher in this case and the number of false positives (FP) is significantly lower compared to the YOLOv8s trained with pseudo-labels generated with GroundingDINO at thresholds at 0.2 (YOLOv8s@0.2). Nevertheless, even the better performance of YOLOv8s@0.3 is not well suited for a real application.

It is notable, that the object detection results of YOLOv8s@0.3 are nearly equal, even slightly better, compared to the ones of GroundingDINO with thresholds at 0.3 (cf. Table 3). This shows well working knowledge distillation from GroundingDINO to YOLO. For YOLOv8s@0.2 the results are slightly worse compared to the ones of GroundingDINO with thresholds at 0.2 but still prove working knowledge distillation via the pseudo-labels.

The following evaluation only considers the datasets with pseudo-labels from GroundingDINO with a box and text threshold at 0.3.

In Table 5 the results of YOLOv8s trained on a differing image amount in the training datasets are displayed. First, the results of YOLOv8s@0.3 trained on every fifth image extracted from the three videos, in total 1089 images, are shown. Second, the results of YOLOv8s@0.3 trained on every fifteenth image (1 Hz) extracted from the three videos, in total 364 images, are shown. The performance of YOLOv8s@0.3 decreased slightly with the decreasing number of images in the 1 Hz training dataset. The higher number of images is thus beneficial. Due to the quite similar performance of YOLOv8s@0.3 trained on the 3 Hz dataset compared to GroundingDINO with thresholds at 0.3, we expect that increasing the number of extracted images for training further does not lead to a significantly increased performance.

Table 4: Evaluation results of YOLOv8s trained on datasets with every fifth image extracted from videos from all three runs (1+2+3) and pseudo-labeled with GroundingDINO either with box and text threshold set to 0.2 / 0.2 or 0.3 / 0.3. The mean average precision (mAP) is calculated over all ten classes with an IoU threshold for correct detections at 0.5 (mAP@0.5). TP: True Positive, FP: False Positive, FN: False Negative.

Dataset	mAP	TP	FP	FN
1+2+3@0.2	0.56	866	733	1342
1+2+3@0.3	0.77	575	96	1633

Table 5: Evaluation Results for YOLOv8s trained on datasets with every fifth image (3 Hz) and every fifteenth image (1 Hz) extracted from videos from all three runs (1+2+3) and pseudo-labeled with GroundingDINO at box and text threshold set to 0.3 / 0.3. The mean average precision (mAP) is calculated over all ten classes with an IoU threshold for correct detections at 0.5 (mAP@0.5).

Dataset	mAP	TP	FP	FN
1+2+3@3Hz	0.77	575	96	1633
1+2+3@1Hz	0.73	557	107	1651

Table 6 shows the results of the consecutive adaption with the addition of data and information from the successive runs. All datasets consist of every fifth image (3 Hz) extracted from the corresponding video. The last row (1+2+3+O) shows the results for a dataset, to which additionally to the self-captured images and pseudo-labels, training data from a dataset available online (Arduengo et al., 2021) was included. Only manually-labeled training data for the classes "door" and "handle" was included from the online dataset, in total 928 images with annotations.

Table 6: Evaluation Results for YOLOv8s trained on datasets with every fifth image successively extracted from the three videos (1, 2, 3) and pseudo-labeled with GroundingDINO with box and text threshold set 0.3 / 0.3. The last row (1+2+3+O) shows the results of additional adding data from a manual-labeled online dataset (O) for the classes "door" and "handle". The mean average precision (mAP) is calculated over all ten classes with an IoU threshold for correct detections at 0.5 (mAP@0.5). TP: True Positive, FP: False Positive, FN: False Negative

Dataset	mAP	TP	FP	FN
1	0.74	559	99	1649
1+2	0.75	592	109	1616
1+2+3	0.77	575	96	1633
1+2+3+O	0.74	598	116	1610

The performance of YOLOv8s increases slightly with a growing dataset. Thus, continuously adding new data from the same environment seems bene-

ficial. However, it should be investigated in future work, to which extent this effect is observable. To benefit from continuously generated new data during detector training, methods from the field of continuous learning seem promising. Especially replay-based methods (Lesort et al., 2020) and dataset distillation methods (Yu et al., 2024) appear suitable for the setting investigated in this paper.

However, adding data from the online dataset did not improve the detector's performance. When looking at the detection results of the two classes "door" and "handle" in detail, only a minimal improvement compared to the training without online data is visible. This may be due to the fact that the evaluation dataset is based on the same environment as the training dataset and a not optimal class balance in the resulting training dataset.

5 CONCLUSION AND FUTURE WORK

To achieve the goal of fully automated environment-specific object detector adaptation for mobile robots, we proposed and investigated an automated method for relevant object class identification, pseudo-labeling the dataset with bounding box annotations for the identified classes using a VLM and finally training a lightweight object detector with the pseudo-labeled dataset. We critically evaluated and discussed each step of the processing pipeline. We have gained the following key insights from the work in this paper:

- VLMs, more specifically GPT-4o, can be used to identify relevant object classes in an image dataset. However, some objects are not identified, even when visible in several images of the dataset. Preliminary experiments of providing patches of an image containing one object to be identified as salient object have shown promising results to identify all objects at a high rate. Nevertheless, the missing context in the patches leads to some misclassifications.
- LLMs can be used to merge variations and synonyms of a class to one consistent class label. Further, a rule-based merging of different object classes (e.g., room names) to one class label was successfully demonstrated. However, the response of GPT-4o to our merging prompt regularly varied. Thus, only a "statistical" selection of class merging schemes was successful for us. On the one hand, a more detailed prompt including which classes should be merged based on given rules, compared to our quite general prompt could partially resolve this issue, cf. (Vemprala et al., 2024). On the other hand, we see the difficulty of consistently resolving semantic ambiguities and granularity using LLMs due to the varying responses to the same input prompt. Further aspects like prompt brittleness (Kaddour et al., 2023) and missing uncertainty quantification for responses (Firoozi et al., 2024) should be also considered in future work. Also, the use of further prior information, like ontologies of relevant object classes, in combination with the LLM may help to overcome semantic ambiguities.
- GroundingDINO showed no good results in detecting the identified relevant classes in our individual setting and thus also provided quite poor bounding-box annotations as pseudo-labels. We assume that this is caused in parts by the difficulty of resolving semantic ambiguities and granularity ("cabinet door"). More detailed input text prompts may improve the results. Further, integrating an automated annotation verification using another VLM, like presented in (Kim et al., 2024), seems promising to improve the quality of the pseudo-labels. Thus, GroundingDINO with lower threshold values could be used to obtain more true positives while sorting out the high number of false positives with a verification VLM.
- Training a lightweight object detector, in this paper YOLOv8s, using the pseudo-labeled dataset worked well. We observed a quite good "knowledge distillation" from GroundingDINO to YOLOv8s via the pseudo-labeled dataset for our specific environment.
- In future work, we plan to improve and extend the proposed method with a VLM to verify the pseudo-labels, a way to automatically generate more pseudo-labels through simulations or generative neural networks, and to investigate the approach in different environments with less common object classes.

ACKNOWLEDGEMENTS

This work was supported by the research projects BauKIRo and K3I-Cycling. The research project BauKIRo (22223 BG) is funded by the Federal Ministry for Economic Affairs and Climate Action based on a resolution of the German Bundestag. The research project K3I-Cycling (033KI216) is funded by the Federal Ministry of Education and Research based on a resolution of the German Bundestag.

REFERENCES

Arduengo, M., Torras, C., and Sentis, L. (2021). Robust and adaptive door operation with a mobile robot. *Intelligent Service Robotics*, 14(3):409–425.

Cheng, T., Song, L., Ge, Y., Liu, W., Wang, X., and Shan, Y. (2024). Yolo-world: Real-time open-vocabulary object detection. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16901–16911. IEEE.

Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., Zhu, Y., Song, S., Kapoor, A., Hausman, K., Ichter, B., Driess, D., Wu, J., Lu, C., and Schwager, M. (2024). Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*.

Gallagher, J. (2023). Distill large vision models into smaller, efficient models with autodistill. *Retrieved 21.10.2024 from Roboflow Blog: https://blog.roboflow.com/autodistill/*.

Gao, M., Xing, C., Niebles, J. C., Li, J., Xu, R., Liu, W., and Xiong, C. (2022). Open vocabulary object detection with pseudo bounding-box labels. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T., editors, *Computer Vision – ECCV 2022*, volume 13670 of *Lecture Notes in Computer Science*, pages 266–282. Springer Nature Switzerland, Cham.

Han, K., Rebuffi, S.-A., Ehrhardt, S., Vedaldi, A., and Zisserman, A. (2022). Autonovel: Automatically discovering and learning novel visual categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6767–6781.

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, Xu, C., Xu, Y., Yang, Z., Zhang, Y., and Tao, D. (2023). A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):87–110.

Hofmann, C., Taliercio, F., Walter, J., Franke, J., and Reitelshöfer, S. (2023). Towards adaptive environment perception and understanding for autonomous mobile robots. In *2023 IEEE Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI)*, pages 1–8. IEEE.

IDEA Research (2023). Groundingdino. *Retrieved 21.10.2024 from https://github.com/IDEA-Research/GroundingDINO*.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8. *Retrieved 21.10.2024 from https://github.com/ultralytics/ultralytics*.

Joseph, K. J., Khan, S., Khan, F. S., and Balasubramanian, V. N. (2021). Towards open world object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5826–5836. IEEE.

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., and McHardy, R. (2023). Challenges and applications of large language models. *Retrieved 13.10.2024 from http://arxiv.org/pdf/2307.10169v1*.

Kim, J., Ku, Y., Kim, J., Cha, J., and Baek, S. (2024). Vlm-pl: Advanced pseudo labeling approach for class incremental object detection via vision-language model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4170–4181. IEEE.

Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., Zhu, J., and Zhang, L. (2023). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *Retrieved 13.10.2024 from http://arxiv.org/pdf/2303.05499v5*.

OpenAI (2024). Gpt 4o. *Retrieved 21.10.2024 from https://openai.com/index/hello-gpt-4o/*.

Sun, T., Lu, C., and Ling, H. (2022). Prior knowledge guided unsupervised domain adaptation. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T., editors, *Computer Vision – ECCV 2022*, volume 13693 of *Lecture Notes in Computer Science*, pages 639–655. Springer Nature Switzerland, Cham.

Vaze, S., Hant, K., Vedaldi, A., and Zisserman, A. (2022). Generalized category discovery. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491. IEEE.

Vemprala, S. H., Bonatti, R., Bucker, A., and Kapoor, A. (2024). Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, 12:55682–55696.

Wang, Z., Li, Y., Chen, X., Lim, S.-N., Torralba, A., Zhao, H., and Wang, S. (2023). Detecting everything in the open world: Towards universal object detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11433–11443. IEEE.

Wu, J., Li, X., Xu, S., Yuan, H., Ding, H., Yang, Y., Li, X., Zhang, J., Tong, Y., Jiang, X., Ghanem, B., and Tao, D. (2024). Towards open vocabulary learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(7):5092–5113.

Yu, R., Liu, S., and Wang, X. (2024). Dataset distillation: A comprehensive review. *IEEE transactions on pattern analysis and machine intelligence*, 46(1):150–170.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.

Zhu, C. and Chen, L. (2024). A survey on open-vocabulary detection and segmentation: Past, present, and future. *IEEE transactions on pattern analysis and machine intelligence*, PP.

Zimmerman, N., Guadagnino, T., Chen, X., Behley, J., and Stachniss, C. (2023). Long-term localization using semantic cues in floor plan maps. *IEEE Robotics and Automation Letters*, 8(1):176–183.