UNIVERSITÀ DEGLI STUDI DI MILANO

DOTTORATO IN INFORMATICA

DIPARTIMENTO DI INFORMATICA “GIOVANNI DEGLI ANTONI”

DOCTORAL DISSERTATION IN COMPUTER SCIENCE

THE CHALLENGE OF DOMAIN SHIFT IN REAL–WORLD ROBOTIC VISION: TOWARD SCALABLE, UNSUPERVISED, AND CLOUD–BASED ADAPTATION

Supervisor:
NICOLA BASILICO

Co–supervisor:
MATTEO LUPERTO

President of the PhD council:
ROBERTO SASSI

PhD candidate:
MICHELE ANTONAZZI
ID: 13700
ORCID: 0000-0001-6396-7567

AA 2025

Acknowledgments

There are many people that I would like to nominate for making this incredible journey possible. First of all, I want to express my gratitude to my advisors Nicola Basilico and Matteo Luperto: a sincere thank for guiding me in the first part of my scientific career and for being my source of inspiration. From them, I got the essential "survival kit" for a researcher: making intuitions through curiosity, inventing solutions with rigorous thinking, and consolidating them with dedication and passion. A special thank to them for recognizing the potential in my ideas, and for supporting me along the challenging path of their conceptualization, development, and refinement, until their full realization.

Many thanks also to all the members of the Applied Intelligent System Laboratory (AIS-Lab) with whom I had the pleasure to collaborate. Among them, a special thought goes to Alex Bassot, Matteo Alberti, and Mauro Tellaroli for the valuable work we carried out together and for the synergy we built around our respective competencies. Good research is the sum of outstanding work from many people, an important lesson I learned from them.

I want to mention the Robotics Lab at the Dalle Molle Institute for Artificial Intelligence (IDSIA), which I had the honor to join for a short visiting. Nicholas Carlotti, Luca Crupi, Elia Cereda, and Simone Arreghini: special greetings to you guys for including me in your group and for the amazing work done together. All of this was not possible without the guidance of Alessandro Giusti, Mirko Nava, and Daniele Palossi, thank you for your effort in guiding my research during the visiting and for promoting the collaboration with the members of the group.

To my wife, the person who supported me during this journey, with love, patience, and perseverance. You have always been on my side, helping me through the dark periods but, above all, sharing with me the successes that I often struggled to see. Thank you for making me feel at home, wherever we were.

A special thank goes to my parents, who were able to support me even if I was far from home. Thank you for all the sacrifices you made for me, you always believed in my potential, and if I am here today it is also thanks to you. But I promise, this will be the last degree!

Finally, to all my old friends, especially Riccardo and Matthias. I know we live far apart, but you always keep alive the relationship we built over the years. Thank you!

Abstract

Mobile robots are an emergent technology more and more present in contexts such as homes, offices, and hospitals to assist humans in daily life activities. Given the complexity of human–centric environments, robotic vision, namely computer vision embedded in mobile robots, has become an essential capability to acquire a semantically–rich understanding of the environment for improving core robotics tasks (such as navigation and localization), but also to enable high–level activities such as manipulation or human–robot interaction. Given the recent advances of deep learning, the naïve solution to implement robotic vision is to leverage publicly available deep neural networks that can be mounted in robotic platforms with limited effort. Despite being widely adopted, this approach suffers from important limitations caused by the so–called domain shift problem: being trained on simulated or generic datasets (source domain), deep neural networks dramatically fail to tackle the complexity of the real–world environments (target domain) in which the robots operate.

This dissertation investigates the challenges of domain shift in robotic vision and proposes novel adaptation strategies to enable robust and scalable real–world deployments of mobile robots. The contributions are structured into three complementary parts.

First, we analyze the limitations of the mainstream pipeline to implement robotic vision that combines pre–trained neural networks with manual fine–tuning. We propose photorealistic simulation–based pre–training using data compliant with the robot's perception modality, and we demonstrate that fine–tuning with limited high–quality manual annotations substantially increases the model's robustness in the specific operational environment of the robot.

Second, we remove the need for human supervision, proposing two alternative approaches for unsupervised model's adaptation. Our methods exploit the spatial constraints between neural network's predictions and the 3D world to enhance the quality of the pseudo–labels, thus enabling self–supervised adaptation.

Third, we address adaptation in cloud–based robotic perception, where intensive inference required by neural networks is offloaded to remote servers to deal with the limited hardware configuration of mobile robots. In this context, we design solutions to ensure the scalability of domain adaptation and enable privacy preservation, designing lightweight neural networks that can be run locally by the robot.

We validate our approaches using both real–world datasets and extensive experiments with real robotic platforms, considering multiple perception tasks such as object detection, 3D pose estimation, and semantic segmentation.

Contents

Chapter 1

Introduction

Indoor service robots represent a flourishing technology increasingly employed across a range of human–centric contexts (such as offices or domestic environments) for assisting humans in daily life activities [1, 2]. Some representative examples are healthcare, where robots assist patients and caregivers [3]; logistics, where they carry out repetitive tasks like delivery or monitoring [4, 5]; or at–home caregiving, where they aid in day–by–day activities like cleaning, personal assistance, or social entertainment [6, 7, 8].

While core robotic tasks (like obstacle avoidance or navigation) can be performed using proximity sensors (such as LiDAR or sonars) [9], mobile robots need an in–depth knowledge of the environment to carry out the aforementioned high–level activities. In this context, *robotic vision* — the ability to extract semantic knowledge from visual perceptions in real time — becomes a cornerstone for capturing semantic information from the robot's operational environment. Leveraging visual perceptions, a robot can detect structural features of the environment (like doors or passages) to improve mapping [10] and navigation [11]. Again, the detection of static objects, such as windows, tables, or doors, is also used for long–term localization [12] in semantic metric maps [13], while people detection is a fundamental building block for human–robot interaction [14, 15]. Another important visual task is semantic segmentation that, by assigning a semantic label to each pixel, enables robots to perform more fine–grained tasks such as manipulation [16, 17] or object finding [18].

Recently, data–driven approaches based on deep learning have advanced the capabilities of vision systems, outperforming in versatility and potential the old–style techniques based on hand–crafted features [19, 20]. As a result, the mainstream approach for implementing robotic vision is to leverage publicly available deep neural networks (DNNs) (e.g., for object detection [21, 22] or semantic segmentation [23]). Since they are pre–trained on standard datasets (such as COCO [24] or Scannet [25]), these models can be deployed in mobile robots with limited effort. However, despite their widespread use, adopting these off–the–shelf models in real–world robotic deployments still poses substantial methodological and practical challenges caused by the well–know *domain shift* problem. Domain shift refers to the discrepancy between the data distribution used to train a DNN (named *source* domain) and the data distribution encountered during testing (the *target* domain) [26, 27]. This is particularly relevant

in robotic scenarios, as the off–the–shelf DNNs used for robotic vision fail to fully capture the visual and operational constraints that robots encounter in real–world deployments, thus facing performance degradation [28, 29]. Being pre–trained in controlled settings (e.g., using simulation [30, 31] or generic datasets [24, 25]), they struggle when operating in complex and dynamic human–centric environments (such as houses, offices, hospitals, and schools) that are challenging both in their physical layouts (e.g., arrangement of rooms, walls, and furniture) and visual appearance.

This dissertation investigates the problem of domain shift in the scope of real–world deployments of mobile robots. Our contributions are structured around three main parts focusing on specific and complementary aspects of domain shift, but all targeted at developing domain adaptation solutions for robotic vision.

- At first, we investigate the main challenges of the mainstream pipeline for implementing robotic vision, addressing the DNN's pre–training using photorealistic data compliant with the robot perception model and studying the robustness of manual fine–tuning to domain shift.
- Then, we focus on removing the need, and so the costs, of human supervision for overcoming the domain shift in the operation environment of the robot.
- Lastly, we address adaptation considering the limited computational resources of mobile robots, that often require offloading the intensive inference required by DNNs to remote servers.

All these contributions are better contextualized in the following paragraphs.

Limitations and costs of classical adaptation (Chapter 3) The first part of this dissertation aims to investigate the challenges of the classical approach for embedding visual perception in mobile robots. The first step is to plug–and–play a publicly available pre–trained DNN inside the robot platform, according to the target task. Consequently, the second step is the real–world adaptation of the DNN using fine–tuning with manual annotations. This pipeline follows the typical lifecycle of a mobile robot, in which the first phase is the *development* of the robot platform (when the DNN is initialized), and its *deployment* in the real–world (when the DNN needs to be adapted for the target environment).

At development time, when the final working environment is unknown, the goal is to provide the robot with vision modules with good generalization abilities to tackle the challenging conditions encountered in unstructured indoor environments. This is a requirement that the off–the–shelf DNNs are not able to achieve because of the standard pre–training phase: realistic datasets from computer vision [24, 32, 25] are not compliant with the challenging perception constraints of service robots, while simulation [30, 33, 31] fails to match the visual aspect of real–world perceptions. Instead of using classical domain generalization approaches from computer vision [34], we propose an innovative method for training DNNs for service robots. By leveraging photorealistic simulation [35], our approach enables the collection of

large datasets combining photorealism and the robot's perspective. These datasets can then be used by robotics practitioners to obtain the so–called *general models*, namely DNNs specifically trained to reduce domain shift experienced by the mobile robots in the real world (this contribution is detailed in Section 3.2).

After deployment, service robots will operate in the same environment in a long–term fashion to carry out their assigned task. In this general setting, the robot can acquire new data and can further reduce the domain shift of its robotic vision modules by leveraging its experience directly acquired in its target environment. A straightforward solution is to fine–tune the DNN using new data with high–quality annotations provided by a human operator. Practically speaking, the previously introduced general model can be trained using additional data from the robot's operational environment, thus obtaining a *qualified model* with improved performance in that environment. Despite time–consuming, this naïve approach is often used in mission–critical use cases, such as healthcare or assistive robotics [36, 7], where mobile robots are installed with the support of a human technician to ensure the deployment's robustness. In this dissertation, we extensively validate the fine–tune paradigm for adaptation, studying the trade–off between the effort for manual labeling and the performance improvement of the adapted DNN. We demonstrate that a few high–quality annotations (obtained with limited effort) remarkably increase the performance of the adapted model while ensuring robustness to classical domain shift experienced by the robot in real–world missions, such as light variations and camera occlusions (refer to Section 3.3).

Unsupervised domain adaptation (Chapter 4) The second part of this dissertation is focused on solving a key limitation of the previous contributions: the need for human supervision for fine–tuning the general model into a qualified one after the robot's deployment. For many perception tasks, obtaining precise annotations is often impractical, as the labeling process is extremely laborious (e.g., in semantic segmentation [25, 29]) or it requires particular environmental setups such as motion capture systems (such as for 3D object detection [13, 37]). To solve this, a simple solution is to fine–tune the model in a self–supervised way, namely by using its own predictions (or pseudo–labels). Despite promising, this approach is impractical as pseudo–labels are very imprecise under domain shift, causing performance degradation if used for re–training. Classical approaches for unsupervised domain adaptation from computer vision benchmarks, that focus on improving the precision of the model's output [29], present limitations when used by robots in unstructured indoor environments, as the pseudo–labels are considered as uncorrelated predictions [38, 39]. Instead, the perceptions acquired by mobile robots during operation present spatial and temporal relations, a peculiarity that can be leveraged for improving unsupervised adaptation to tackle the visual challenges of indoor environments. In this dissertation, we provide two solutions for the unsupervised adaptation in robotic vision leveraging the consistency between the DNN's predictions and the 3D world in which the robot operates.

- In the first contribution, reported in Section 4.1, we improve the multi–view consistency proposed in [38] (where the model's predictions are aggregated in a dense 3D map of the

environment) by adding a refinement step to make the pseudo–labels to be coherent with the object instances in the scene. Our method, developed considering the semantic segmentation task, advances the in–depth understanding required by service robots operating in indoor environments.

- In the second contribution, detailed in Section 4.2, we remove the need of building a 3D spatial map for pseudo–label consistency. Instead, we propose an unsupervised approach for adaptation by forcing the spatial consistency of the pseudo–labels directly in the loss function during fine–tuning. Our approach substantially increases the precision of the DNN models for 3D object detection, a task often used by very low–powered platforms (such as nano–drones) for localization and navigation [37, 40].

Adaptation in cloud–based perception (Chapter 5) Running the intensive inference required by the modern DNNs directly on mobile robots is often unfeasible given their limited computational capabilities. To solve this, robotics practitioners often need to engineer inference pipelines decentralized over the cloud [41, 42, 43], where DNNs are deployed in remote servers. More specifically, these neural networks (that we refer as *TaskNets*) are general models running remotely, that robots can access through queries (by sending their perceptions) to obtain the result of the inference. The third part of this dissertation addresses two peculiar challenges generated by this architecture and specific to the robotic domain.

- **Scalability:** domain adaptation solutions based on standard fine–tuning cannot be applied in cloud robotics to obtain qualified models as they are not scalable [41, 43]. In this context, the remote general model can be shared by multiple robots deployed in different indoor environments. Since each environment represents a different domain, the naive approach of having a separate qualified model for each robot quickly becomes impractical. This is because training, deploying, and maintaining multiple DNNs requires extensive computational resources, generating prohibitive costs.

- **privacy preservation:** mobile robots, operating in daily life environments such as apartments or university facilities, acquire very privacy–sensitive images [44]. In the domain of cloud robotics, these perceptions are sent to remote and potentially untrusted cloud providers that can use our data for unauthorized purposes [45]. This is a major but largely unaddressed issue in cloud robotics. Conventional protection techniques, such as encryption, are ineffective in this scenario, as the images need to be decrypted by the cloud provider to be processed by the general model for inference.

In this dissertation, we provide innovative solutions to enable scalable domain adaptation and privacy preservation for object detection distributed over the cloud. The general idea is that, while the main and intensive task of detecting objects (e.g., doors or people) is offloaded to a remote server, privacy and adaptation can be solved locally by the robots using lightweight DNNs. More specifically, we design an encoder–decoder architecture to obfuscate the robot's perception before upload without compromising the TaskNet's inference (see Section 5.2). Then,

instead of fine–tuning multiple times the generic TaskNet, we design a small neural network to refine the predictions of the remotely deployed general model directly by the robot after inference, accordingly to its specific target domain (see Section 5.1). Our contributions enable scalable domain adaptation in cloud robotics, without compromising privacy.

Contributions overview The contributions provided by this dissertation (that come from the papers of [46, 47, 48, 49, 50, 51]) are organized in chapters according to the incremental challenges we encountered during the study of the domain shift problem in the context of robotic vision.

In Chapter 3 we investigate the practical problems of pre–training effective general models for mobile robotics and how they can be qualified for a target environment during deployment. In the conference paper of [46], we delineate the limitations encountered by mobile robots equipped with pre–trained off–the–shelf models. We prove that using our photorealistic dataset from the robot perspective increases the performance of a general model for robotic vision with respect to using real–world images from a standard dataset. Always in [46], we start our investigation about the impact of the fine–tuning for the model's qualification, demonstrating that a few manual annotations are sufficient for a substantial performance improvement. All these preliminary findings are strengthened in the journal publication of [47], where we propose a comprehensive pipeline to implement robotic vision. We further validate the proposed approach for training a general model including data coming from an additional simulator. Then, we increase the number of real–world environments for testing both the general and qualified models. Furthermore, we demonstrate the robustness of the qualification procedure in challenging situations typical of human–centric environments, such as light variations or camera occlusions. Lastly, we replicate our experiments using multiple end–to–end architectures to prove the robustness of our solutions.

In Chapter 4, we focus on an important limitation of the qualification procedure: the need for human supervision. Since the self–supervised paradigm is ineffective under domain shift, we explore solutions to increase the quality of pseudo–labels by leveraging the spatial consistency of the model's predictions. In this direction, we propose two alternative approaches based on this intuition. In the conference paper of [48], we aggregate the predictions in a 3D map to obtain spatially consistent pseudo–labels, that are then further refined by propagating the object categories according to the object instances. In the paper of [49], we propose an alternative approach to force the pseudo–labels consistency removing the computational overhead of building a 3D map. More specifically, we perform fine–tuning by embedding the concept of consistency in the loss function, forcing two predictions (obtained from different viewpoints) to be equal when reported in the same frame of reference using the estimated movement of the robot between them.

In Chapter 5 we face with the limited hardware configuration of mobile robots, which poses important practical limitations of the aforementioned solutions. To solve this, a common workaround is to deploy the general model in a remote server and make it accessible to multiple robots (operating in different domains) through queries. In such a scenario, standard

qualification procedures based on fine–tuning are not scalable because this means having multiple qualified models running on the cloud, one for each robot. To solve this, we propose in the conference paper of [50] a method to make the adaptation scalable in cloud robotics scenarios. We design a lightweight neural network for adaptation that can be run locally by the robot. More specifically, the general model (TaskNet) is maintained fixed and its predictions are refined by our approach according to the specific domain shift experienced by each robot, thus making adaptation scalable. Working on this, we faced with privacy preservation as robot's perceptions are sent remotely to a potentially untrusted cloud provider. The in–depth analysis we made to refine the prediction of a general model served as a basis to develop an innovative approach for privacy preservation, reported in the journal paper of [51]. We proposed to obfuscate robot's perceptions using a lightweight encoder–decoder co–trained with the TaskNet. To consolidate our method, we justify our intuitions using a theoretical framework in which we approximate our end–to–end architecture using matrices. In addition, we perform an extensive experimental campaign using both well–established datasets and a real robotic platform, further demonstrating the robustness of our solution.

Together, the contributions related to the three main parts of this dissertation explore different aspects of the domain shift in robotic vision and advance the adaptability of DNNs in real–world robot deployments. The rest of the dissertation is organized as follows:

- we discuss the state–of–the–art in Chapter 2;
- Chapter 3 reports our contributions regarding the limitations of the classical adaptation strategies applied in robotic vision, that we take from the works of [46, 47];
- our solutions for unsupervised adaptation (derived from the works of [48, 49]) are in Chapter 4;
- Chapter 5 details our methods for scalable adaptation and privacy in cloud robotics, published respectively in the works of [50] and [51];
- we conclude and delineate future directions in Chapter 6.

Chapter 2

Related Works

To robustly operate in previously unseen human–centric environments, a robot should be able to understand relevant properties through its vision. This is why *robotic vision*, namely computer vision embedded in mobile robots, has emerged as a fundamental component of autonomous agents [52]. In this chapter, we report the previous works relevant to this dissertation, discussing their limitations and highlighting the novelties we propose. At first, in Section 2.1 we introduce the visual perception tasks we consider, highlighting how they become useful for mobile robots. The remaining of the chapter is organized to match the challenges related to the previously defined three main parts in which this dissertation is divided. More specifically, Section 2.2 reports approaches of domain generalization for training the general model. Then, Section 2.3 treats unsupervised adaptation techniques to remove human supervision for qualification, reporting also approaches specifically designed for autonomous robots (Section 2.3.1). Lastly, Section 2.4 focuses on cloud robotics and discusses the challenges of scalable adaptation and privacy preservation (Section 2.4.1) within such a paradigm.

2.1 Perception Tasks for Mobile Robotics

2.1.1 Object Detection

An important task we consider is object detection, which consists of identifying the location and category of objects in the image plane. The recent advancements in deep learning applied to computer vision [53] have led to the development of high–performance pre–trained models (such as YOLO [54], DETR [55], and Faster R–CNN [21]), which can be exploited for *robotic vision*. Object detection is a fundamental building block for human–robot interaction [14, 15, 8], as it enables robots to identify people. This task is also often used in robotics to improve core robotics tasks such as mapping or navigation [13, 10] through the detection of environmental features.

Among others, detecting doors is crucial for mobile robots as their traversability status (`open` or `closed`) enables the availability of passageways between sub–areas of an environ-

ment. In turn, traversability directly determines the environment's topology, ultimately affecting the robot's navigational routes and accessibility of the areas therein. The location of doors is important for tasks such as room segmentation [56], which entails partitioning the map of an environment into semantically meaningful areas or rooms. This knowledge is also beneficial in predicting the layout of rooms not yet observed during exploration [57] or in identifying temporarily unreachable locations during mapping. Furthermore, it plays a key role in place categorization, a process where rooms on the occupancy map are assigned semantic labels (like "corridor" or "office") based on their visual appearance [58, 59]. Recent studies have shown that a robot's ability to recognize doors can significantly enhance its navigation capabilities in long–term scenarios. For instance, the work presented in [60] models the periodic changes in dynamic environments over extended periods. Similarly, the study in [61] introduces a navigation system designed for robots functioning in indoor environments for long periods, particularly where the traversability of the area varies over time.

The use of object detection methods is the mainstream approach to tackle door detection with mobile robots. Initial seminal methods in this domain relied on the extraction of handcrafted features [62, 63, 20], such as edges [19] and corners [64], to describe the characteristic rectangular shape of doors. However, the requirement to explicitly define and combine these features is a significant limitation of these approaches. This constraint hampers their robustness and adaptability, especially when dealing with the highly variable images encountered in real dynamic environments.

Deep learning end–to–end methods have brought significant improvements in the field. Their ability to automatically learn features that are robust to variations in scale, position, rotation, and lighting is a major advantage that has led to their widespread use in mobile robotics. A pioneering method for door recognition in mobile robot navigation was introduced in [65]. This method utilizes color and shape as key features to detect doors in office environments, employing two neural classifiers to identify these elements in images. These features are then integrated using a heuristic algorithm to determine if they form a typical door structure. The study in [66] presents a method for door detection aimed at enhancing the autonomous navigation of mobile robots. A convolutional neural network is trained to identify doors in indoor settings, demonstrating its utility in aiding a robot's efficiency in traversing passages. Additionally, recent research has explored the integration of RGB vision with other sensors commonly used in robot navigation [67] and the identification of doors and their handles to enable interactions like grasping [68, 69, 70]. For example, the research in [68] utilizes a YOLO–based deep learning framework [22] for the detection of door Regions Of Interest (ROI). This approach specifically targets the handles by focusing on the area encapsulated within the door's ROI, effectively locating the handles for interaction purposes.

2.1.2 3D Object Detection

Another notable task often used in robotics is 3D object detection. Instead of estimating the position of an object in the image plane, it consists of estimating the 3D location (relative to the

robot's position) of objects in the real world. This capability is particularly useful for localization in scenarios where occupancy grid maps are unreliable. For instance, when a robot operates in environments with a dynamic layout (such as offices or apartments), grid maps may become outdated. In such cases, 3D object detection enables long–term localization [12] by identifying static objects (e.g., doors, windows, tables, blackboards) in semantic maps [13]. 3D object detection is also fundamental to implement localization in very low–powered robots (such as nano–drones) where depth measurements acquired from active sensors are unavailable for battery saving [40, 71]. In this context, the limited processing power forces practitioners to use lightweight convolutional neural networks on low–resolution cameras [72, 73]. The straightforward approach for training these lightweight models for 3D object detection is to directly use data collected in controlled environmental setups, involving a motion capture systems or AprilTags [37, 13]. However, these systems are expensive and unsuitable for use across multiple environments. As such, alternative approaches learn from simulated data while mitigating the sim–to–real gap with strong image augmentations [40], or by randomizing the visual aspect of many simulated environments [74]. Another way to align the visual appearance of simulated and real images is to apply a pencil filter that highlights edges and disregards flat image regions [75].

2.1.3 Semantic Segmentation

The last visual task we consider in this dissertation is semantic segmentation. It enables pixel–level classification of images, where each pixel is assigned to a specific semantic category [76]. Unlike object detection, which identifies a limited number of targets with bounding boxes, semantic segmentation offers dense and detailed scene understanding, making it particularly useful in dynamic and unstructured environments. In mobile robotics, semantic segmentation is widely applied to enhance mapping and navigation. The work of [77] leverages semantic segmentation to detect walls for the 3D reconstruction of floor plans in multi–room environments. In [78], the authors build enhanced occupancy grid maps with semantic information coming from image segmentation. Their goal is to build a human–inspired localization system using semantically annotated LiDAR point clouds that focuses on the distribution of semantic elements, rather than geometric ones. Recent approaches embed semantic segmentation in scene graphs [79, 80] that reconstruct the spatial and hierarchical distribution of objects for planning [81]. Semantic segmentation is also often used for precise scene understanding in robotic manipulation. The work of [17] uses a 3D map annotated using semantic segmentation to infer semantically unsafe conditions for robotic manipulation. In [82], semantic segmentation is fused with instance segmentation obtained from foundation models to tackle the diversity of objects encountered in real–world manipulation tasks.

2.2 Domain Generalization

The fist challenge addressed in this dissertation regards the training of a general model that can be plug–and–played in mobile robots. Since the final working environment in which the robot operates is unknown at training time, the general model should be able to obtain acceptable performance in any possible target domain. For this purpose, domain generalization is an effective strategy to increase the perception abilities in unseen domains (that are operational environments in our case). The general idea is to have a "general" model that can perform reasonably well in any possible deployment condition and target environment. To reach this, domain generalization proposes to use training data from multiple source domains to cover a large portion of the input space, thus increasing the model's generalization. A prominent approach is to randomize the domain using simulators by altering the number and location of objects, modifying illumination conditions, and randomizing textures to match the complexity of real–world environments [83, 74]. Other approaches leverage data generation to synthesize randomized, realistic training data [84, 85] or alter the input space to render the two domains indistinguishable with autoencoders [86] or generative adversarial networks [87]. A simpler approach for making domains indistinguishable is to pre–process input images with a pencil filter to emphasize edges and remove distracting domain details [75].

The main limitation of the aforementioned approaches is that they achieve generalization considering only the visual appearance of training data. Instead, in Chapter 3, we propose a domain generalization approach specifically tailored to mobile robots. Our method allows robotic practitioners to acquire large datasets using photorealistic simulation with limited effort, where data are compliant with the motion model of the robot, thus increasing the generalization abilities of DNN for robotic vision in real–world deployments. We perform extensive experiments of our approach considering the door–detection task, which we evaluate in multiple real–world indoor environments.

2.3 Unsupervised Domain Adaptation

Domain generalization is not sufficient for adaptation as covering all the real–world visual appearances is unfeasible. However, the robotic domain offers the unique opportunity to go beyond a priori generalization as, during robot's deployment in the real–world, data can be collected in the target domain and used to overcome the domain shift problem. To this end, *domain adaptation* approaches aim at fine–tuning a model to capture the specific characteristics of the deployment domain. The simple solution is to use high–quality labeled data, often involving laborious hand–labeling [47, 12] or ad–hoc sensing apparatus [37]. In Chapter 3, we extensively validate this naïve approach for qualification in the scope of door–detection, considering also multiple challenging situations typical of human–centric environments (i.e., light variations and camera occlusions).

Despite promising, performing adaptation with human annotations is often impractical as manual labeling is laborious and time–consuming. To tackle this challenge, *unsupervised* adaptation represents an attractive alternative [26]. One of the mainstream approaches consists of performing domain confusion to align the distributions of source and target domains used at input, intermediate features, and/or output levels. Domain alignment is done with adversarial learning, by confusing a discriminator network [88, 89], or by minimizing a notion of domain divergence [90, 91], such as the Maximum Mean Discrepancy (MMD) [92].

Approaches to reduce domain shift at the input and feature levels perform style transfer between domains using generative models (e.g., Generative Adversarial Network [93]) to make the source and target input data indistinguishable. Then, they implement domain alignment through adversarial learning to align the embeddings' distributions. The pioneer work of [94] presents a general paradigm based on unpaired image–to–image translation to align the neural network's embeddings extracted from images from different domains. Each training image is translated from its source domain to the other, thus building a latent space in which the distribution of image features is domain invariant, and semantically similar images are projected into close vicinity. The work of [95] proposes a cycle–consistency approach by giving the additional objective to the model to reconstruct the original data from the translated version. This paradigm is extended by [96], which combines massive image–to–image translation using GANs with adversarial learning and pixel–wise consistency.

Another promising solution is to perform adversarial learning on the neural network's output space. The rationale behind this is that the spatial distribution of objects in the images from the training and evaluation datasets should be similar, even if their visual features differ. An example of this is the fact that tables and chairs are usually located close to each other. Consequently, when correctly identified, the spatial distribution of labels coming from different domains should be similar. Following this, the authors of [97] propose to train a CNN to fool a discriminator tasked to disambiguate the domain from which its output masks are generated. An extension of this approach is reported in [98], which employs an adversarial training scheme to align the entropy distribution of the target images (computed as the pixel–wise categorical probability distribution) with that of the source domain. This is done following the assumption that the entropy is usually low on data from the source domain and high on that from the target one.

The adversarial learning paradigm has some limitations: it is inherently unstable, and it focuses only on domain confusion. To overcome these issues, recent works leverage self–supervision with pseudo–labels inferred by the model itself from the target data. However, pseudo–labels from images in the target domain are noisy and inaccurate, and they cannot be directly used to fine–tune the source model. A promising solution to this is to balance the influence of pseudo–labels during training, promoting high–confident predictions while penalizing noisy annotations [99]. The confidence is estimated as the divergence between the outputs from two independent prediction layers. This uncertainty is then used as a regularization term during fine–tuning. An alternative approach is that of [100], which estimates the confidence of pseudo–labels by using the relative distance of the extracted features from their respective class

centroid. Other approaches update self–generated labels at each training iteration to mitigate noisy predictions [101]. This is the case of the method in [102], that weights the loss by the model uncertainty under different input augmentations. Conversely, the works of [103, 104] update only high–uncertainty labels.

Another general approach is to use a student network, tasked with adapting to the target domain, supervised by a teacher network that, being gradually updated using an exponential moving average of the student, provides more stable and consistent pseudo–labels. Following this, the work in [105] rectifies the pseudo–labels using feature clustering on pixel embeddings from the target domain. The label of each pixel is updated based on the assigned cluster, thus providing more coherent and stable annotations that are used to supervise the student training. A similar approach is to use Masked Image Consistency [106], in which the student has access to images in which random patches are masked, and it should be consistent with the teacher network, which uses unmasked ones. An extension of the teacher–student paradigm is presented in [107]. In such a work, a student network is supervised by two teachers: the first one is applied to the outputs while the second, bigger teacher is fixed during training and provides transfer knowledge in the feature space.

Compared to other unsupervised approaches coming from computer vision, this dissertation proposes solutions for unsupervised adaptation targeted to the robotic domain, exploiting the spatial relations between the model's predictions.

2.3.1 Unsupervised Adaptation using Robot Embodiment

One major limitation of the previously presented approaches is that they operate by processing each frame independently, thus neglecting the fact that they are acquired by an embodied agent capable of autonomous movements and equipped with different sensors. This poses the unique opportunity for developing unsupervised adaptation solutions specifically tailored to mobile robots. A promising direction (delineated in Section 2.3.2) is to filter the model's errors in the self–supervised fine–tuning by forcing the consistency of the pseudo–labels predicted from different viewpoints. A different approach (discussed in Section 2.3.3) is to self–supervise the models' adaptation using additional sources of supervision coming from the robot's sensors, such as depth or odometry, for solving downstream tasks.

2.3.2 Unsupervised Domain Adaptation enforcing View Consistency

Standard approaches for unsupervised adaptation neglect possible relations with previous and past perceptions. This ignores the fact that indoor mobile robots, while navigating, observe several similar views from the same scene; thus, the images in the sequence are highly correlated. This fact can be leveraged as a natural proxy for consistency, which can be exploited to improve pseudo–label quality, reduce uncertainty, and enhance the recognition of small, occluded, or

partially visible objects.

The work of [38], which introduces the concept of multi–view consistency, leverages the fact that model's predictions about the same locations of the environment must exhibit some level of similarity even if taken from multiple viewpoints. The method aggregates single–frame 2D semantic predictions of the robot into a 3D voxel map, which stores the probability for each voxel to belong to a certain semantic class. This dense representation of the environment is then used for rendering pseudo–labels that aggregate information from multiple viewpoints, thus removing the intrinsic noise of the frame–by–frame model's predictions. This approach is extended in [108], where the authors propose to leverage the robot embodiment for data acquisition in specific environmental locations. The idea is to aggregate in the dense environmental map not only the semantic labels, but also the uncertainty estimation of the model's predictions. In this way, the robot can collect novel views focusing on areas with high uncertainty to improve the quality of the 3D annotations. In the recent work of [39], the voxel map is substituted with a semantic NeRF, a more compact spatial representation of the environment that allows the rendering of novel views (RGB images) and their respective pseudo–labels.

While these methods present promising results, they still often struggle to resolve coherent misclassifications that persist across multiple frames. In Chapter 4, specifically in Section 4.1, we advance multi–view consistency leveraging a 3D map [38] by addressing this limitation. We integrate a zero–shot, instance–aware refinement phase after 3D map rendering that enforces single semantic classes per object instance while removing rendering artifacts.

2.3.3 Self–Supervised Robot Learning

Another way to implement unsupervised domain adaptation in the robotic domain is to self–supervise the fine–tuning using other sensors as a source of supervision. Typical examples include the use of proximity sensors and odometry to self–supervise the training of an obstacle detection model [109], or dense point clouds for traversability estimation [110]. Recent self–supervised approaches leverage a pretext task, namely a trivial learning objective (different from the task of interest) supervised by readings from other sensors. This pretext task is used to learn features similar to those needed to solve the desired task. Some examples are estimating, given an input image, the sound produced by a drone [111], or the state of the LEDs fitted on a robot [112].

Another relevant self–supervised approach is described in [113]. The proposed approach leverages the robot's odometry to enable multi–view consistency between DNN's predictions for 3D pose estimation. The proposed method proposes a loss function where two model's outputs (obtained from different positions) need to be equal when reported in the same reference frame using the robot's movement estimated using odometry.

Leveraging this idea, in Section 4.2 of Chapter 4 we propose a self–supervised approach for sim–to–real adaptation in 3D object detection. To do this, we leverage only images acquired by the robot and annotated with the estimated odometry to optimize the state consistency between the model's predictions.

2.4 Cloud Robotics

Given the limited computational resources of mobile robots, running intensive inference using the robot's hardware reaching real–time performance is often unfeasible. To solve this, cloud robotics [114] is an active area of research where a general model (called TaskNet) for robotic vision is deployed on a remote server and accessed by robots through queries [115, 116]. One of the mainstream approaches for cloud–based DNN workload distribution is Model Splitting [117] where, essentially, the model is divided into two or more portions that are run collaboratively across the network. Examples of this method for object detection are [118] where YOLOv3 undergoes a process of cloud–edge distribution and [119] where inference follows a hierarchical structure from the cloud to the end device.

A series of works, falling under the name of "Fog Robotics", investigated solutions based on distributing the computing resources in intermediate units between robots and cloud data centers. This paradigm is becoming increasingly widespread [42], with distributed object detection (typically in synergy with grasping) being among its real–world challenges. Examples of works on this line include [120], where models are initially trained in the cloud and subsequently adapted at the edge, [121] where authors focus on reducing latency by means of a Q–learning–based policy for load balancing, and [122] where object detection based on SSD [123] is served to the robot from a fog node cluster whose resources can be adapted to guarantee service quality. Other examples of similar offloading strategies for object detection in mobile robots have been proposed in [124] and [125].

Most of these works primarily focus on enhancing service quality but overlook the challenge of scalable domain adaptation in the constrained cloud setting we consider. Recently, [42] proposes a software architecture based on the Robot Operating System (ROS) to easily integrate cloud resources and robotic platforms, while the works of [41, 43] propose orchestration strategies to ensure quality of service for real–time cloud inference for robotic tasks.

In this dissertation, we address scalable domain adaptation. This problem is generated by the fact that a single general model working remotely is used by multiple robots, deployed in different domains. Given this, having multiple qualified models, one for each robot, quickly becomes unfeasible as the number of robots increases. In our architecture, related to the one proposed in [126], cooperation between a large cloud–based model and a smaller one operating locally on the robot is exploited. Our method differentiates in the role and design of the smaller model. In the previous work of [126], the smaller model is essentially a scaled–down, less accurate variant of the larger one, aimed at reducing costs. In contrast, our approach, delineated in Chapter 5 and Section 5.1, enhances the smaller model's role to not just serve as a cost–effective alternative but to specifically refine and adapt the cloud model's predictions for the robot's unique operational environment, offering a scalable solution to overcome domain shift in cloud robotics. For this investigation, we consider the task of door detection, as the visual appearance of doors strongly changes across environments. Because of this, our TaskNet is an object detector and our lightweight neural network has the role of refining the bounding boxes according to the target domain in which the robot operates.

2.4.1 Privacy Preservation in Cloud Robotics

Another issue of the cloud robotics paradigm addressed in this dissertation is privacy preservation. This problem is particularly relevant as images acquired by low–powered robotics platforms in indoor environments (like apartments, hospitals, or factories) are sent remotely to curious (and potentially untrusted) cloud providers. Despite the data stream can be protected during transfer using standard encryption, it needs to be decrypted on the server side for inference, as the TaskNet needs plain images [127]. The work by [44] examines these concerns in robot teleoperation, focusing on identifying key privacy features to be privatized, like object locations and types.

A computationally inexpensive yet effective way to privatize images is by applying filters, such as blurring, pixelating, or mosaicing. Blurring reduces image detail by averaging pixels, pixelating lowers resolution, and mosaicing breaks the image into smaller tiles and rearranges them. However, recent literature demonstrates that these techniques are not robust against deep–learning attack models [128]: the work of [129] proposes a super–resolution method to remove blur from faces in images, while [130] shows how generative models can reconstruct pixellated faces.

The recent advancements in machine learning have also facilitated the development of new privacy–preservation techniques [128]. A promising approach is Homomorphic Encryption [131], which aims to allow DNNs to work with both plain and encrypted data. However, these models struggle to work in real–time as needed by a mobile robot and still exhibit a large performance gap between the use of plain and encrypted data [132].

Another family of approaches aims at masking sensitive information from images (e.g., faces or people's silhouettes). A GAN–based architecture is exploited in [133] to alter people's appearance by generating a synthetic face, while the work of [134] aims at masking the whole person's figure. The method of [135] preserves the high–resolution background of images perceived by a mobile robot, while a person's identity is obfuscated using a low–resolution mask. In [136] authors propose to use a GAN to modify only the face of the user when seen by a social robot, to prevent its identification. Although these methods are promising, introducing privacy only on sensitive parts of an image is not enough, as sensitive data can be sent out due to faults in the system. Also, those methods have high computational requirements and are vulnerable to missing detections: a single misclassification in a data stream might expose a person's identity. Moreover, sensitive information can appear in the background (e.g., a credit card on a table). In this dissertation, we consider this more challenging scenario, seeking to obfuscate all the acquired data: not only people's identities but also their activities and the background scene.

An alternative approach is differential Privacy (DP), which is exploited in [137] to prevent face identification by adding noise to slightly alter the face's appearance without complete obfuscation. While DP is robust against single–frame attacks, it is less effective when applied to data streams, as the attacker can exploit the correlation between frames to filter the added noise.

The works of [138, 139] discuss how activity recognition can be performed on privacy–preserving low–resolution videos, as small as 16×12 pixels. Similar findings are reported

in [140] for posture classification and in [141] for low–resolution RGB–D data. Despite being promising for privacy preservation, these approaches cannot be used for object detection as low–resolution data compromises the spatial integrity of features.

The work of [142] presents some similarity with our architecture, in which a DNN is deployed on a remote server and accessed by remote edge devices. Such a work focuses on optimizing a TaskNet for image classification while minimizing the effectiveness of a DNN–based attacker that exploits adversarial reconstruction and classification techniques. Unlike ours, their framework relies on a *trusted* cloud environment, while we are considering a more challenging *untrusted* one. Another related approach is explored in [143], where a distributed perception pipeline is examined. In this setup, edge devices upload images to a cloud–based TaskNet. The study utilizes an autoencoder to erase sensitive information while retaining essential features for task performance. Unlike the work described in [144] and our setting, the proposed encoder–decoder architecture is embedded within the TaskNet backbone, which is divided into two segments: one deployed on the device and the other on the cloud. The system has been evaluated focusing on face and license plate recognition. While similar to our approach, this method results in obfuscation concentrated around edges and textured regions, thereby maintaining privacy–sensitive details in other parts of the image.

The work of [145] uses a Transformers–based approach to extract tokens used for the task of activity recognition in a video stream. Tokens are anonymized through adversarial learning. As the video stream is processed as a whole, the method cannot be used in an online setting like ours.

NinjaDesc [146] is an adversarial learning framework that extracts visual descriptors from images, for the task of matching similar images (e.g., for localization), while limiting an attacker from reconstructing obfuscated ones. To preserve accuracy in matching similar images, NinjaDesc descriptors retain the keypoint locations. As a consequence, patterns of keypoints can be used to identify persons as they represent the scene structure.

In this dissertation, we address the limitations of the aforementioned approaches when applied in scenarios where mobile robots, that operate in private environments, need to send their perceptions remotely. Instead of masking specific privacy–sensitive features (as performed in [135, 136, 133]), our method, presented in the Section 5.2 of Chapter 5, provides a novel training approach for global obfuscation of the image. It is specifically designed for removing as many details as possible while enabling a pre–trained TaskNet for object detection to work even with obfuscated images that do not contain privacy–sensitive details. To achieve this, we take inspiration from the work of [144] that aims to extract from images task–relevant feature representation for image classification. We provide theoretical evidence of its limitations in object detection tasks, and we formally define a novel co–training scheme based on weak loss to enable an encoder–decoder (running on the robot) to achieve privacy while preserving detection–oriented features. We assess the effectiveness of our approach using publicly available datasets as well as experiments with a real robot. In addition, we corroborate the robustness of our privatization technique exploiting the well–established Model Inversion Attack (MIA) [147] in its stronger version, where the malicious actor has full access to the encoder–decoder and the

TaskNet, as well as their training datasets. The attacker is implemented following the same approach of [143], which we extend by incorporating a more powerful edge–centric loss function.

Chapter 3

Addressing Domain Shift during Development and Deployment of Robotic Vision

In this chapter, we investigate and address two important challenges about domain shift in DNN for robotic vision that arise in the main phases of a mobile robot lifecycle: model pre–training during robot *development* and model qualification after *deployment* in an unseen environment.

During development (i.e., where the final working environment of the robot is unknown) the straightforward solution to enable the robot with vision capabilities is to plug–and–play a publicly–available off–the–shelf DNN [21, 22, 23]. In real–world scenarios, service robots operate under specific perception constraints that are intrinsic to the robotic domain. This condition introduces critical domain shift over the off–the–shelf DNNs that, being pre–trained on standard datasets such as COCO [24] or Pascal VOC [32], or Scannet [25], largely neglect the noisy, constrained, and challenging operational conditions that a robot faces in the field. To overcome this, robotic practitioners need a large quantity of annotated data, compliant with the robot's perspective, to initialize end–to–end models capable of working in robotic contexts. Obtaining real–world data at scale is difficult as the collection process is often expensive and time–consuming. Conversely, synthetic simulators [31, 30] allow for easy collection of large amounts of training data; however, differences with the real world in appearance and dynamics pose a challenge, the so–called sim–to–real gap [28, 148]. To address this, we propose training a *general model* during the robot's development using photorealistic simulated data captured from compliant with the perception model of service robots. We demonstrate that our approach effectively reduces the domain shift of vision modules mounted in service robots and used in novel real–world environments, unseen during training.

After the deployment, a service robot will operate in the same environment for a long time, probably for its entire life–cycle. In such settings, the robot will observe the same scene and the same objects multiple times, but from different viewpoints. This consistency can be used to improve the perceptual capabilities of the robot in its target environment, thereby compensating the domain shift of the general model. To achieve this, we propose to fine–tune the general

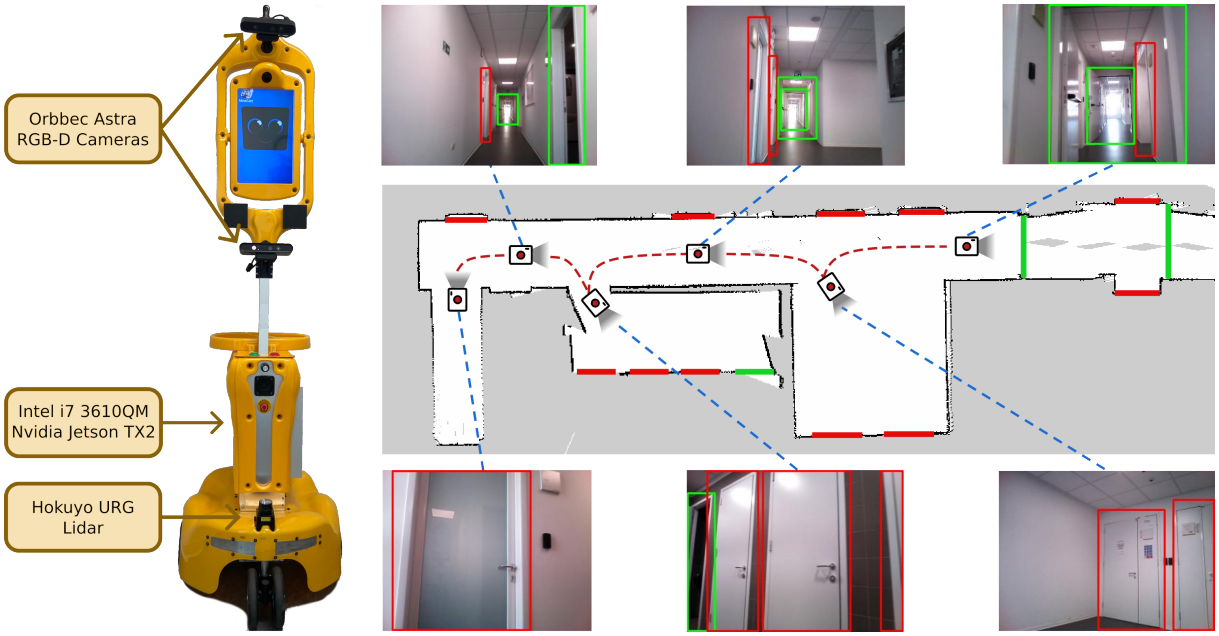


Figure 3.1: Our Giraff–X [7] robot performing door–status detection while navigating in an indoor environment. The green (red) bounding boxes represent open (closed) doors.

model with a small batch of high–quality annotated data acquired by the robot while exploring the environment. The resulting *qualified model* remarkably improves the performance of its general version in the operational environment of the robot, especially in challenging instances or under scene variations (e.g., moving people or lighting variations). Additionally, we provide an extensive evaluation in multiple real–world environments to study the trade–off between data–acquisition efforts and performance improvement.

We investigate the above challenges (model pre–training and qualification) considering the *door–status detection* task: the capability to recognize, in real time, the location and the traversability status (`open` or `closed`) of passages. In this chapter, we extend the conventional concept of "explicit" door [149] as a physical object composed of a leaf and a hinge. Instead, we consider "doors" also simple passages between two distinct rooms in the same environment, such as the connection between a corridor and a living room. The detection of such environmental features is particularly significant for mobile robots as it can be exploited to enhance the long–term navigation capabilities of service robots [60, 61]. We addressed this problem as an object detection task using RGB images, as illustrated in Figure 3.1: a mobile robot navigates in an environment to perform its tasks; at the same time, it acquires images through its onboard camera. For each image, it infers in real–time the bounding boxes of doors using an end–to–end object detector, distinguishing between open doors (depicted in green) and closed ones (depicted in red), also highlighted on the map.

Our contributions, published in [46, 47], are summarized as follows:

- We analyze the trade–offs involved in using simulations to train a general door–status detector capable of generalizing effectively across various environments (Section 3.2). We delineate the desiderata of this process and propose a framework based on simula-

tion performed in 3D real–world models, Gibson [35], that achieves a balance between data photorealism and acquisition costs (Section 3.2.1). Leveraging this procedure, we collect and release a visual dataset for door–status detection acquired in 10 photorealistic environments from the robot point of view.

- We explore fine–tuning to qualify door–status detectors to a robot's target environment, demonstrating how leveraging typical operational settings of service robots can enhance performance in challenging instances (Section 3.3).
- We argue that performance metrics used to evaluate computer–vision models are not well–suited to be used in a robotic context, and we propose new evaluation metrics specifically designed for object detection with mobile robots (Section 3.4.4).
- We conduct an extensive experimental campaign with three state–of–the–art and widely adopted deep–learning models, providing insights into how domain shift in object detection for robotic vision is influenced by the nature of the standard training datasets (Sections 3.5.1 and 3.5.2). We further validate our findings by assessing the robustness of our general and qualified door–status detectors to typical domain shifts occurring in long–term deployments inside the same environment (Section 3.5.3). To achieve this, we collected a dataset for door–status detection in four real–world environments using our robotic platform [7], and we have made it publicly available (Section 3.4.2).
- We evaluate the impact of different door–status detection methods on a downstream task, that is evaluating the current traversability status of the whole environment (Section 3.5.5).

3.1 Problem Formulation: General and Qualified Models for Door Detection

The purpose of this chapter is to address the domain shift problem in Robotic Vision at development and deployment time of a mobile robot. To do this, we consider the door–status detection task (defined in Chapter 3 and depicted in Figure 3.1). We focus on a service robot designed to autonomously operate in human–centric indoor environments. We assume a widely–used hardware setup, where the robot is equipped with one or more RGB cameras for vision. The primary objective is developing real–time door detection. To do this, we leverage an end–to–end object detector that processes an RGB image acquired by the robot to find passages and determine their traversability status. Each door is then described using a bounding box associated with a binary label (`open` or `closed`).

Performing door detection leveraging RGB data is primarily due to the limitations of alternative technologies. For example, laser range scanners, while robust and precise for distance measurements, are typically constrained by a 2D field of view, making it difficult to disambiguate a planar surface (such as a wall) from a closed door, especially in settings like a corridor

where the doors are perpendicular to the motion of the robot. In the same line, RGB–D cameras often exhibit a limited depth sensing range, making them less reliable over longer distances or in larger indoor spaces See, for example, the doors depicted in Figure 3.1: depth and LiDAR data alone are not adequate to detect the presence and the status of such challenging door instances when observed from constrained viewpoints (e.g., doors in images of the first row of Figure 3.1 are difficult to detect with LiDAR data). Furthermore, both types of sensors struggle with transparent or highly reflective surfaces, that are common in doors. An example is depicted in the first robot's perception (first image second row) of Figure 3.1, where depth sensors are not able to see the closed door with a glass panel on it. These limitations are also relevant in sensor fusion approaches (combining RGB with 3D data) as they inherit the challenges associated with the depth sensing. As previously mentioned, our target task often involves the detection of transparent or reflective surfaces, situations in which the depth data are missing or noisy. See for example the third image of the first row of Figure 3.1: the doors are perpendicular to the robot point–of–view and some of them have a glass panel. In these cases, 3D data can provide little knowledge to detect the status of doors, and the robot can rely only on vision. Integrating such unreliable information with knowledge acquired from RGB images is not straightforward, as it may represent an additional source of error. Consequently, our work focuses on RGB–only perception, which provides more consistent and robust visual cues for identifying the status of doors in the real–world.

The method we propose, graphically summarized in Figure 3.2, is structured around the two principal phases that define the lifecycle of a mobile service robot: the robot's *development* phase and the subsequent *deployment* phase. With this method, we aim to identify, experimentally evaluate, and solve some of the challenges that are intertwined with the usage of vision–based object detection methods on mobile robots.

The development phase for a service robot involves preparing and configuring the platform, including the installation and setup of hardware and software components. The objective here is to setup a robot that is ready to meet the challenges of real–world environments. This phase's focus lies in the domain of visual perception capabilities, aiming to equip the robot with vision skills that perform satisfactorily across various environments, thereby ensuring a high level of generalizability. The development phase is the starting ground for addressing our door–status detection task. Our approach involves creating a *General Detector* (GD), designed to recognize doors while adhering to the perception constraints of service robots and maintaining consistent performance in various environments. A significant part of our method involves utilizing simulation to develop a photorealistic visual dataset, representing typical visual perceptions of a robot. This dataset is then used to train a GD, ensuring it achieves baseline performance in the real world.

During the deployment phase, the service robot is introduced for autonomous operation in a target environment, usually for an extended period. This phase often involves a domain shift, presenting challenges to the performance of the previously developed GD. This is because the pre–built computer vision methods, focusing on the model's development, present difficulties introduced by our environmental setup that prevent their straightforward use on autonomous

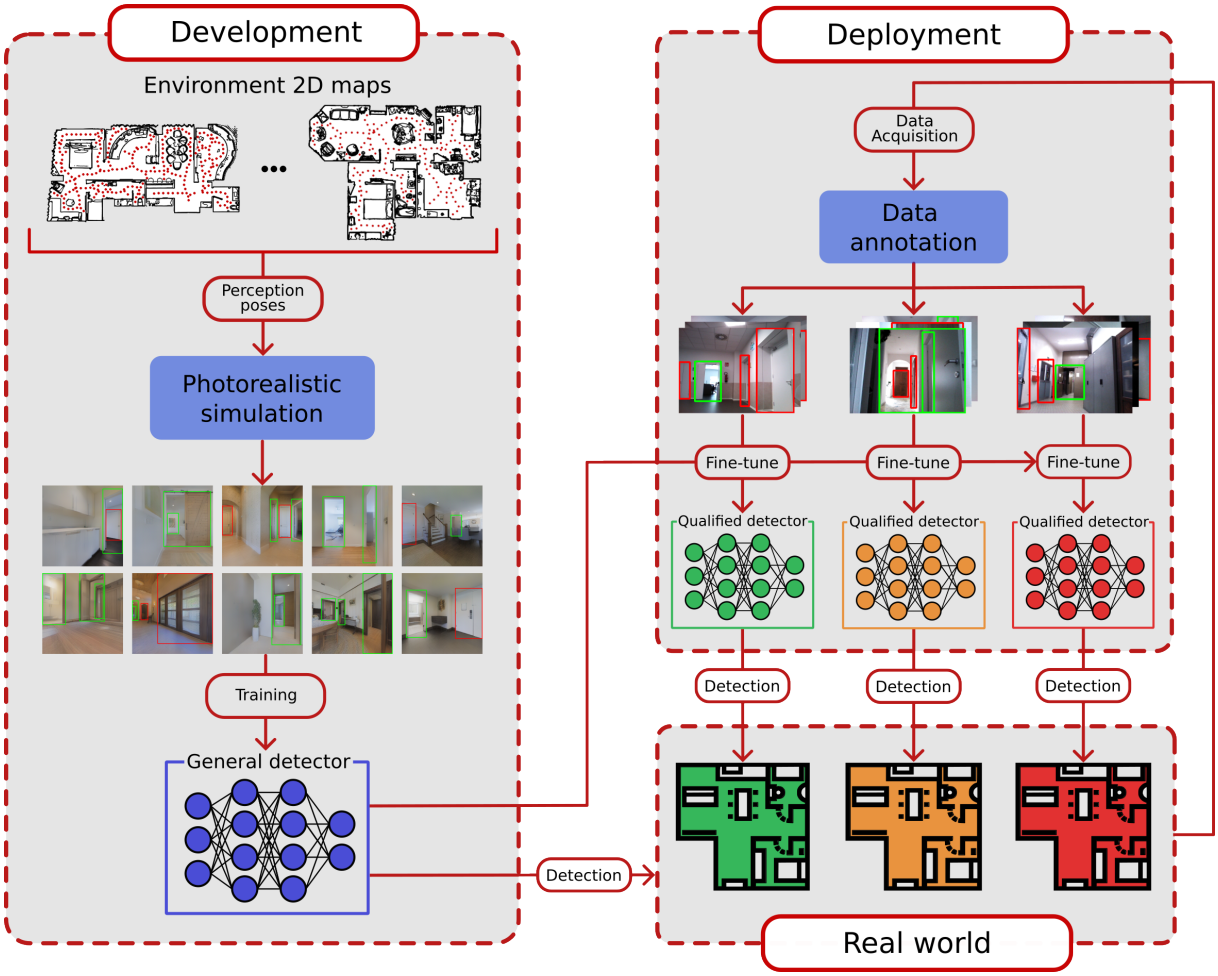


Figure 3.2: A general overview of our pipeline for the development and deployment of deep learning–based object detectors for robotic vision. At first, we build the General Detector (GD), a module that exhibits acceptable performance operating in novel environments, not included during the training phase. For doing this, we introduce a novel simulation framework to reduce the effort for acquiring training datasets from the robot's perspective. After the robot deployment, we study the domain shift experienced by the GD and we perform a fine–tuning, obtaining a Qualified Detector (QD), enhancing the detection performance in the operational environment of the robot.

mobile robots. Given the long–term nature of this phase, there is an opportunity to incrementally fine–tune the GD with data collected in the target environment, aligning it more closely with the specific visual features at hand, thus obtaining a *Qualified Detector* (QD). This detector can exploit the fact that usually multiple instances of the same object within the same environment present similar features, that are stable in time. Doors and windows are good examples of this fact: within a target environment, most of them are usually produced by the same manufacturer and are of the same type. The process of adapting the detection model to the target environment may require the collection and annotation of data for which we propose a method demonstrating a trade–off between the effort required and the resulting performance improvements.

3.2 General Detector

The recent trends in object detection suggest that a straightforward way to address Robotic Vision is to plug and play a deep detector in a robotic platform [148]. Despite the availability of a large number of effective models, when faced with reality this simple approach presents several engineering challenges and an established method to provide a GD for service robots still needs a comprehensive investigation.

State–of–the–art object detectors are typically trained on prominent datasets (such as Pascal VOC [32], ImageNet [150], or, most commonly, MS COCO [24]) composed of thousands of images acquired from application–agnostic viewpoints in diverse contexts, including both indoor and outdoor settings. However, when these models are applied to robotics, two primary challenges arise. First, the dataset could not extensively represent the object of interest, compromising the detector's ability to recognize it. Our survey of existing datasets reveals that doors frequently suffer from this lack of representation. This is primarily due to our broad definition of a door as a variable–traversability passage, introduced in Section 3.1. Secondly, even when the object of interest is well–represented, distribution shifts between the training data and real–world scenarios can significantly affect performance. This widely recognized yet largely unresolved issue is particularly problematic in our context, as the shift can affect multiple aspects: the input data, the feature space, and the data collection process itself. (Here we rely on the discussion about domain shift types of [151]).

Perhaps the dataset for door detection that is most relevant to our work is *DeepDoors2* (DD2) [152], which contains around 3000 images of doors, each annotated with a bounding box and traversability status[1]. However, in our scenario, DD2 is susceptible to performance degradation due to distribution shifts, a fact that becomes expected already upon examining some of its examples. The images in DD2 are captured from human–like perspectives, often showing the door fully visible and centrally located, as depicted in the indoor and outdoor examples of Figure 3.3a and Figure 3.3b, respectively. This dataset overlooks instances such as partially visible or nested doors, which are common in robots' perceptions. Labels are provided only for doors that are completely within the frame and distinct enough for clear identification, as shown in the dashed bounding boxes of Figure 3.3c (partially visible door) and Figure 3.3d (nested doors). These shortcomings are, to varying degrees, present in nearly all conventional computer vision datasets [150, 32, 24, 152], reflecting their inherent limitations in capturing a robot's visual perception model [52]. As our experimental campaign will demonstrate concretely, these limitations significantly affect performance.

To address them, one common method is fine–tuning a large–scale pre–trained model (such as one trained on MS COCO [24]) with new examples that better represent the target object distribution. This approach is prevalent, especially in robotics [13, 153, 68], and the strategy we evaluate is based on it. Ideally, creating an effective door detector through fine–tuning

[1]Note how a dataset of this size is customary in several object detection tasks: as an example, in MS COCO, the average number of examples per class, is 3200; the only exception is the category `person`, which has more examples, over 10K.

(a) (b) (c) (d)

Figure 3.3: Examples from the DD2 dataset [152] of open and closed doors (in green and red, respectively). The dashed bounding boxes represent missing annotations.

requires a dataset that:

- demonstrates a high level of photorealism (to withstand distribution shifts at the input level);
- encompasses a variety of indoor environments with diverse features (to withstand distribution shifts at the feature level);
- accurately reflects the robot's perspective and perception model (to withstand distribution shifts in the data acquisition process).

Currently, no dataset fulfilling these criteria exists in the literature, as efficiently collecting it is still an open problem. An alternative to this issue is to use labeled sequences of images obtained by a robot or by a mobile platform, such as in ScanNet [25] or SUN3D [154]. However, these sequences are usually collected within single rooms and, as they are based on fixed trajectories, do not allow the sampling of new viewpoints from different perspectives that may be encountered by the robot while navigating. The most straightforward approach would involve an extensive data collection campaign using robots in real–world environments, gathering image samples and manually labeling them. However, the logistics and costs associated with this method are prohibitively high and well–recognized among robotics professionals. In the following, we tackle this problem by exploiting simulation [155], an approach frequently employed in robotics to mitigate the large costs of on–the–field experimentation. The empirical results we present later will demonstrate how, with appropriate design measures, simulations can provide a dataset from which an effective door detector can be trained.

3.2.1 The Proposed Simulation Framework

Common 3D physics simulators like Gazebo [156, 157] or CoppeliaSim [158, 159] are widely adopted for prototyping control software in robotics before real–world deployment [155]. However, their lack of a sophisticated rendering pipeline for realistic visual perceptions makes them

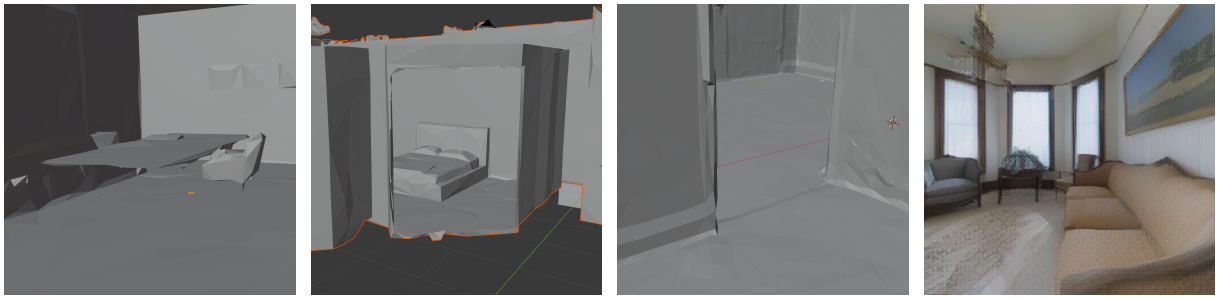
unsuitable for our setting. Early efforts to address this limitation have involved the use of 3D game engines, such as Unity3D [160] or Unreal [161], to recreate complex robotics scenarios, including unique environments or specialized physical laws, as seen in autonomous vehicles [162, 163], UAVs [164], or surgical robotics [165]. Despite their adaptability, customizing these game engines for a particular robotic application can be challenging [166]. Specifically, for the task considered in this dissertation, this would involve manually creating synthetic scenes that accurately reflect the structural features of real indoor environments, a task that is more aligned with environment design than robotics engineering.

Recently, the introduction of interactive realistic simulations for embodied AIs, as iGibson [30], has helped mitigate the limitations of traditional simulators for indoor robotic tasks. iGibson comes with 15 artificially constructed home–sized scenes, which are developed by populating layouts of actual environments with 3D controllable objects whose configuration, shape, material, and texture can be automatically changed. Unlike the simulators mentioned earlier, iGibson seamlessly integrates with the Robot Operating System (ROS), facilitating the collection of extensive, high–quality annotated image datasets from a robot's perspective. However, despite these significant advantages, simulators based on synthetic scenes still fall short in achieving the crucial aspect of photorealism. This limitation is something we empirically assess in our experimental campaign. Similar findings are identified in the work of [167], which shows how, for the task of visual navigation for an autonomous mobile robot, the higher the performance in simulation, the higher the gap in performance with a real robot, as the robot models often overfit on the synthetic features of the simulated environment, that are different to those of real ones.

In addressing these challenges, we adopted an approach that balances the photorealism of real–world data acquisition with the automation benefits of synthetic simulations. Our solution relies on Gibson [35], a simulator designed for embodied agents with an emphasis on enhancing the photorealism of visual perceptions. Gibson employs scene datasets scanned directly from real environments (such as Matterport3D [168] and Stanford–2D–3Ds [169]) that accurately capture and replicate the challenges typical of the real world. Additionally, it incorporates a neural rendering pipeline to further bridge the sim–to–real gap. These enhancements aid in the effective transfer of models trained within the simulator to real–world environments. Leveraging these features, we developed a simulation framework based on Gibson in conjunction with Matterport3D. It is a comprehensive RGB–D dataset comprising 90 digitized real scenes also including semantic tagging for both instance and category–level segmentation. This combination allows us to achieve a balance between photorealism and the controlled conditions necessary for effective simulation.

Gibson provides a middleware for controlling a ROS–based virtual robot. While camera perceptions can be easily simulated (setting resolution and FOV), navigation encounters several technical limitations. First, conducting a real–time acquisition campaign, even in a simulated environment, can be time–intensive. Moreover, this approach does not offer complete control over the data acquisition process, as much of it depends on the navigation stack of the simulated robot. Additionally, the 3D polygonal meshes of the Matterport3D environments, which

are digitized from real–world settings, are often cluttered and noisy. This results in various issues: furniture models appear malformed and incomplete (as shown in Figure 3.4a, where the legs of the table are not modeled), walls frequently have holes near windows or mirrors (see Figure 3.4b, where the bed should be behind a mirror and a wall, which are missing), and the surfaces of floors are irregular (see Figure 3.4c). These artifacts mostly concern the 3D meshes and not the images; as a result of this, there is a mismatch between the features of the images and the depth features perceived by the robot (e.g., with a LiDAR). As an example, in Figure 3.4a the robot sees (image) a table, which is not perceived in the corresponding depth sensor reading; in Figure 3.4c the robot sees a flat, smooth, pavement surface, where the 3D meshes are bumpy and are inaccurate. As a result, the robot's autonomous navigation is prone to failures and not robust. At the same time, image data are not much affected by these errors thanks to the Gibson's rendering pipeline that, using a neural network, corrects possible visual artifacts (as in Figure 3.4d).



(a) **(b)** **(c)** **(d)**

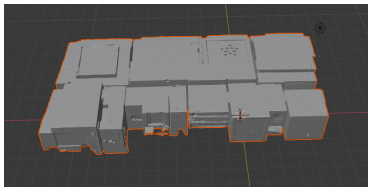
Figure 3.4: Matterport3D mesh malformations. (a) The table and chairs have no legs. (b) A wall in the bedroom, in front of the bed, is missing. (c) Floor surface irregularities. (d) An example of perception acquired from the Gibson–based simulation framework.

To address these shortcomings, we have developed an enhanced version of Gibson, introducing a highly controllable simulation mechanism. This upgraded simulation framework[2] allows to script robot teleporting actions to any location relaxing some constraints from the physics engine such as gravity or collisions. Such a capability can enable large–scale batch data acquisition without the risk of operational failures. With this system, the robot can effectively operate over uneven floor surfaces and across different floors without encountering issues related to architectural barriers, like stairs or elevators. This approach significantly streamlines the data–gathering process, ensuring efficient and uninterrupted data collection in simulated environments. Figure 3.4d shows an example of an acquisition obtained with this simulation framework.

[2]The pre-compiled python package is available at https://pypi.org/project/gibson/, the source code can be found at https://github.com/micheleantonazzi/GibsonEnv.

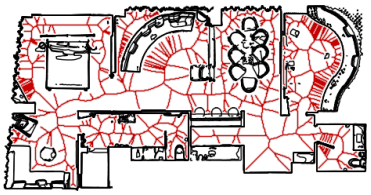
3.2.2 Pose Extraction

To effectively exploit the simulation framework described above it is crucial to ensure that the data acquired aligns with the perception model of a service robot. To model how a robot perceives human–centric environments, we rely on the experience obtained in a long–term deployment of service robots, described in [7]. To achieve this, we propose a method for principled selection of *perception poses*. First, data acquisition should occur from locations within the free space that also maintain a minimum clearance from the nearest obstacles. Additionally, these locations ought to be strategically positioned along the shortest paths connecting key areas of the environment. This positioning is key as these paths are the most likely to be covered by a robot during its service time. Third, it is important to distribute the locations uniformly throughout the environment to minimize redundancy and to ensure comprehensive coverage of the environment's visual features. Our method is composed of three distinct phases.

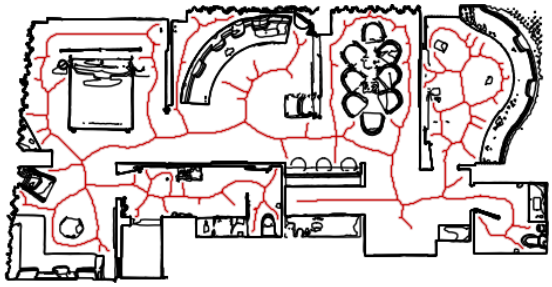


(a) A 3D mesh of an environment obtained from the simulator

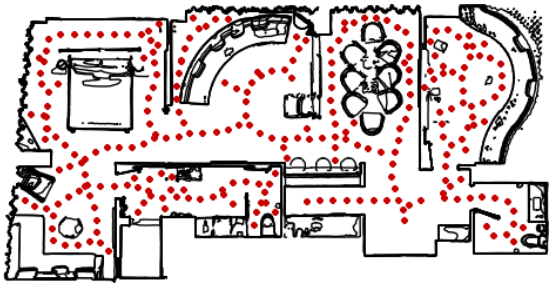
(b) Map of free (F, white) and obstacle (O, black) space



(c) Grid cells ($\mathcal{G}_0$, in red) lying on the Voronoi boundary



(d) Navigation graph ($\mathcal{G}$, in red)



(e) 2D locations (in red) of the perception poses (P)

Figure 3.5: Different phases of our pose extraction method. Starting from (a) the 3D mesh of the environment, our pipeline extracts (b) the 2D map of the traversable area. Then, it generates (c) the boundary of the Voronoi graph using obstacle contours pixels as centroids, that is pruned and cleaned obtaining (d) the navigation graph, which emulates a path compliant with those delineated by the navigation stack of a real robot. Finally, our procedure samples from the navigation graph (e) the 2D poses from which to acquire the robot's perceptions.

The initial phase focuses on generating a 2D map of the environment from the 3D mesh provided by the simulation framework. This process involves aggregating obstacles identified through multiple cross–sections of the 3D mesh, which are created using parallel planes starting from a few centimeters over the floor level. The resulting map undergoes erosion and dilation to eliminate small gaps between obstacles so as to exclude areas that are unreachable or too close to obstacles. Figure 3.5 presents some key outcomes of these steps. Specifically to this first phase, Figure 3.5a illustrates the 3D mesh of a simulated environment, while Figure 3.5b

displays the corresponding 2D map.

Algorithm 1 Compute navigation graph

Input: $\mathcal{M} = (F, O)$, the 2D map of the environment
Output: $\mathcal{G}$, the navigation graph

1: $O_v \leftarrow findContours(O)$
2: $(V_0, E_0) \leftarrow VoronoiBoundary(F, O_v)$
3: $\mathcal{G}_0 \leftarrow Grid_\epsilon(F, V_0, E_0)$
4: **do**
5: $\quad filter \leftarrow false$
6: $\quad$ **for** $c \in \mathcal{G}_0$ **do**
7: $\quad\quad$ **if** $degree(c) \leq 1$ **then**
8: $\quad\quad\quad \mathcal{G}_0 \leftarrow \mathcal{G}_0 \setminus c$ ▷ Filter spurious cell
9: $\quad\quad\quad filter \leftarrow true$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: **while** $filter$
13: $\mathcal{G} \leftarrow Skeletonize(\mathcal{G}_0)$ ▷ Apply skeletonization

In the second phase, the extracted map is used to compute a *navigation graph*, a data structure that represents the principal routes likely to be traversed by a robot. This process is detailed in Algorithm 1. The map, denoted as $\mathcal{M} = (F, O)$, comprises free and obstacle points sets denoted as $F, O \subseteq \mathbb{R}^2$, respectively. Initially, the algorithm identifies the contours of the obstacle shapes in O, resulting in a set of vertices $O_v \subseteq \mathbb{R}^2$ (line 1). This set contains the minimum number of vertices to represent the obstacle shapes without information loss. These vertices are used as basis points for calculating the Voronoi boundary within the free space F (line 2). This boundary, separating Voronoi cells that cover F, is structured as an undirected graph with vertices $V_0 \subseteq \mathbb{R}^2$ and edges $E_0 \subseteq V_0 \times V_0$. The algorithm then overlays the Voronoi boundary onto a grid map that discretizes the free space F at a resolution ϵ. Each grid cell c_i has an area of ϵ^2 and is centered at coordinates (c_i^x, c_i^y). A partial grid $\mathcal{G}_0$ is formed by selecting free space grid cells that contain at least one point from O_v (line 3, also illustrated in Figure 3.5c). Subsequently, $\mathcal{G}_0$ undergoes a heuristic filtration to eliminate spurious cells, specifically those with a degree (number of adjacent cells, assuming 8–connectivity) of 1 or less (lines 4–12), targeting isolated or excessively narrow grid branches. The final step involves a skeletonization process [170] to further simplify the grid structure (line 13). This involves converting $\mathcal{G}_0$ into a bitmap, applying the skeletonization algorithm, and then reconstructing a final grid $\mathcal{G}$, which effectively represents the navigation graph. An example of the obtained result is shown in Figure 3.5d.

In the third phase, the navigation graph $\mathcal{G}$ is utilized to determine the poses for data acquisition, a process detailed in Algorithm 2. A perception pose is defined by the tuple (x, y, h, θ), where x, y are the 2D coordinates on the map, corresponding to the center of a cell in $\mathcal{G}$. From this location, the robot acquires an image at height h and orientation θ. Essentially, the algorithm performs a depth–first search on $\mathcal{G}$, generating a cluster of poses each time a distance D is covered on the grid. This is achieved using a stack S and a set of explored cells, denoted

Algorithm 2 Pose extraction

Input:
- $\mathcal{G}$: the navigation graph
- D: a distance threshold

Output: P, the set of poses

1: $S, EXP, P \leftarrow \emptyset, d \leftarrow 0$ ▷ Initialization
2: $cur \leftarrow randomCell(\mathcal{G})$
3: $S.push(cur)$
4: **while** S not empty **do**
5: $c \leftarrow S.pop()$
6: $EXP \leftarrow EXP \cup \{c\}$
7: $d \leftarrow d + dist(cur, c)$
8: $cur \leftarrow c$
9: **if** $d \geq D$ **then**
10: **for** $h \in \{h_{high}, h_{low}\}$ **do**
11: **for** $i \in \{0, 1, \ldots, 7\}$ **do**
12: $P \leftarrow P \cup (c^x, c^y, h, \frac{\pi i}{4})$ ▷ Add perception pose
13: **end for**
14: **end for**
15: $d \leftarrow 0$
16: **end if**
17: **for** $c' \in \mathcal{N}(c) \setminus EXP$ **do**
18: $S.push(c')$
19: **end for**
20: **end while**

as EXP. The functions $d(\cdot, \cdot)$ and $\mathcal{N}(\cdot)$, applied over $\mathcal{G}$, compute the distance between cell pairs and identify the set of adjacent cells for a given cell, respectively (assuming again 8–connectivity). The exploration initiates from a randomly selected cell (line 2), and whenever a distance of at least D is covered (line 9), 16 poses centered on the current cell c are added to the set P. These poses are generated by iterating over two height values (h_{high} and h_{low}) and 8 different orientations ranging from 0 to 2π in $\frac{\pi}{4}$ increments (lines 10–14). An example of the set of 2D locations obtained over the navigation graph is depicted in Figure 3.5e.

3.3 Qualified Detector

During the deployment phase, the robot is set up for long–term operation in a specific target environment, denoted as e. A critical requirement for autonomous navigation is obtaining an on–site map. This process often involves a technician who either directly operates the robot or assists it in exploring the environment to acquire a map for later use. (We experienced this setup during an extensive experimental campaign conducted in the scope of an assistive robotics study where service robots have been installed in several private apartments [36, 7]. Beyond this, we deem that the setup is common and highly representative to a very large number of on–the–

field installations.) In this exploration phase, the robot has the opportunity to collect additional data, particularly images of the environment captured with its onboard RGB camera. A selected portion of these images can be labeled with doors and utilized to fine–tune the general detector developed in Section 3.2, tailoring it specifically to environment e. We refer to the adapted version of this detector as the *qualified detector* for environment e and we denote it as QD_e. A general overview of the proposed methodology is illustrated in Figure 3.6.

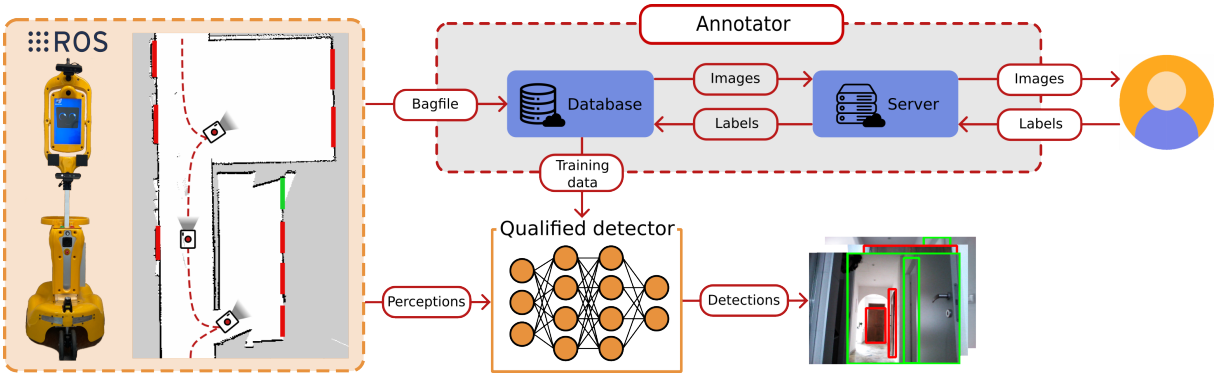


Figure 3.6: A general overview of the qualification procedure. During the initial phase of the robot's deployment, where it is tasked to acquire a map of the new environment, it collects new perceptions inside a ROS bag file. This file is then uploaded to our web–based annotation tool that extracts the RGB images and provides an interface for manual annotation. The new labels are finally exported and used to fine–tune the general detector in a qualified version with enhanced performance when used in the specific environment in which the robot will operate.

An intriguing approach would be to automatically label the additional acquired data in what essentially would be an instance of an unsupervised domain adaptation problem. One method is to use *pseudo–labels*, generated by applying our general detector to the new samples, a technique called self–supervised learning [29]. However, our preliminary experiments showed a significant performance drop of about 20% with this method compared to results with the general detector. This decline can be attributed to the inherent inaccuracy of pseudo–labels, as also observed in recent studies [38]. While pseudo–labels may improve performance in tasks where precise labels are less critical (such as semantic segmentation [171]), their lack of accuracy makes them unsuitable for object detection tasks, such as the one we consider. Indeed, re–training with missed or hallucinated bounding boxes produces a drift in the model in which errors keep getting reinforced. Exploring more advanced techniques for unsupervised domain adaptation (as discussed in [27]) is addressed in Chapter 4, where we propose two alternative approaches to improve the quality of the pseudo–labels for self–supervision. In contrast, the aim in this section is to empirically assess the trade–offs in enhancing a general detector. Consequently, we opt for manual labeling, which can be conveniently done during the robot's installation phase. This approach is widely accepted in robotics, e.g., see the work presented in [12], where manual annotations have been used to fine–tune an object detector for long–term localization tasks.

To facilitate this process, we have developed and released a ROS–integrated data annotation tool[3]. This tool allows transferring robot perceptions from a ROS bag into a database. It then

samples these perceptions at a given frequency and presents them to a technician, providing an interface for easy bounding box annotation. To enhance efficiency, bounding boxes from one image are retained in subsequent images, leveraging the robot's slow movement to reduce labeling workload and reuse prior annotations.

With our experimental campaign, we prove the benefits that the qualification procedure brings to the robot's performance, studying also the trade–off between the effort between labeling costs and the model performance gain. We empirically show that a relatively limited effort is sufficient to obtain remarkably better results in object detection. In addition, we show that this procedure is more effective when applied to a GD trained with data from the robot's point of view.

3.4 Experimental Setting

In the Sections below, we describe our experimental setting by detailing our model selection (Section 3.4.1), the datasets used for the trials and the details of their preparation (Section 3.4.2), the procedures and the hyperparameters used for training and testing the detectors (Section 3.4.3), and the evaluation metrics we propose to adopt (Section 3.4.4).

3.4.1 Model Selection

Research in deep learning for object detection primarily explored three types of deep learning architectures. Initially, the focus was on two–stage detectors, which were then followed by the development of one–stage models. More recently, considerable interest has been devoted to Transformers.

Two–stage detectors (such as [172, 21]) employ an architecture featuring two parts. The initial part generates *proposals*, namely regions likely containing objects of interest. The second part classifies and refines these proposals in a coarse–to–fine fashion. Following a more end–to–end approach, one–stage detectors (such as [173, 123]), perform object recognition in a single step. They simultaneously predict both the locations and the labels of objects using predefined bounding boxes, known as *anchors*, which are distributed uniformly across the image. Recently, Transformer–based models (such as [55, 174]) have gained importance as a novel paradigm in object detection. These models first create a spatial feature map from the input image, which is then processed by a Transformer [175]. This process allows for the parallel prediction of multiple objects' labels and locations, with the added advantage of considering inter–object relationships through the use of attention. (See [148] for a more comprehensive survey of these techniques.)

In our experimental campaign, we selected a representative model for each architecture type, based on availability and deployment feasibility on a robotic platform. These models are

[3]The datasets, models, scripts, and tools used for our experiments are available at https://aislab.di.unimi.it/research/doordetection

chosen as they are widely used and stable release, to mimic a choice of a robot practitioner that is selecting such methods for a long–term deployment. However, other (more recent) models of the same families of object detectors can be used instead.

For the two–stage architecture, we selected Faster R–CNN [21] as implemented in the PyTorch Hub framework. This model includes a Feature Pyramid Network (FPN) backbone based on ResNet–50 [176], coupled with a Region Proposal Network (RPN) [21] and a classifier for bounding box regression [172], totaling around 41 million parameters. For one–stage detectors, we opted for the medium–sized variant of YOLOv5 [54], which has approximately 20 million parameters. Both these two architectures apply a non–maximum suppression procedure to discard bounding boxes with a high overlap (for any pair of bounding boxes with an overlap of 50% or more, the one with the lower confidence is removed). As for the Transformer–based model, we selected DETR [55], which integrates a ResNet–50 backbone [176] with a Transformer module [175] and a four–layer MLP, summing up to 41 million weights in total. DETR requires setting a critical hyperparameter, N, which defines the fixed number of bounding boxes predicted per image. We set N to 10, a value slightly higher than the maximum number of doors observed in any single image in our datasets, to ensure comprehensive detection without excessive computational burden.

Recently, zero–shot architectures have been proposed as a promising solution also for the task of object detection, showing remarkable results. This family of methods can be particularly interesting as it does not require additional datasets to be adapted to new tasks. Thus, we performed some preliminary examples on our task of door detection, in order to add these families of models to the three investigated here. We tested two models: Language–SAM, which combines Grounding Dino [177] and Segment Anything [178], and the Transformer–based OWL–ViT [179], two state–of–the–art zero–shot object detectors in which object categories are specified as textual queries. We prompted both models with "open door" and "closed door". However, we observed that the performance of models, while being able to detect doors, was significantly lower than those of one–stage, two–stages, and Transformer–based ones. In particular, most of the doors are detected multiple times, both closed and open, with similar confidence, making it difficult to disambiguate such detection to a single category. Consequently, we deem that these models are not mature enough yet to be used on challenging tasks such as robotic vision and we have not used zero–shot architectures for further evaluation.

3.4.2 Datasets

In our experiments, we considered a total of four datasets composed of images and their relative door–status annotations.

The first dataset, which we refer to as $\mathcal{D}_{\texttt{DD2}}$, is derived from the DD2 dataset [152] discussed in Section 3.2. This dataset includes 3000 real–world images taken from a human perspective, as provided in DD2. In these images, doors are marked as open, semi–closed, or closed. For the purposes of our experiments, we re–labeled the dataset to include ground truth data for complex examples that were not initially annotated (similar to those shown in Figure 3.3). Additionally,

considering the operational constraints of a robot, which may not be able to navigate through partially opened doors, we categorized the doors marked as semi–closed as closed.

The second dataset, which we refer to as $\mathcal{D}_{\texttt{iG}}$, was generated using the iGibson simulator [30]. iGibson provides 15 artificial environments, designed to mirror the structural features of real indoor scenes. To capture data from the perspectives of robots, we implemented a pose extraction mechanism akin to the one outlined in Section 3.2.2. (The details of this method are not elaborated here, as our later results will show its limited performance.) By integrating this pose extraction process with the ability to control door configurations within the simulation, we successfully generated a large batch of around 35000 instances that were automatically annotated using the semantic data provided by the simulator. (Some examples are reported in Figure 3.7.)

Our third dataset, referred to as $\mathcal{D}_{\texttt{G}}$, was created using the Gibson–based simulation framework described in Section 3.2. This dataset comprises images generated from perception poses derived using Algorithms 1 and 2. For this dataset, we set the distance parameter D to $1\,\mathrm{m}$, and used two different robot embodiments with heights of $h_{low} = 0.1\,\mathrm{m}$ and $h_{high} = 0.7\,\mathrm{m}$ across 10 diverse Matterport3D environments, including small apartments and large villas with multiple floors and varied furniture styles. In processing these images, we utilized the semantic frames provided by Matterport3D, where each pixel is classified into an object category. We filtered out images without doors (i.e., where pixels labeled as "door" constituted less than 2.5% of the total image). Subsequently, we automatically generated bounding box proposals around door instances. This pre–processing step significantly simplified the final phase of manually verifying and completing the annotations, which was carried out by human operators. The resulting $\mathcal{D}_{\texttt{G}}$ dataset comprises 5457 images all captured from the perspective of a mobile robot. The dataset contains approximately 6000 door instances labeled as `open` and around 3000 labeled as `closed`. See some examples in Figure 3.7, where also the enhanced photorealism with respect to $\mathcal{D}_{\texttt{iG}}$ can be appreciated.

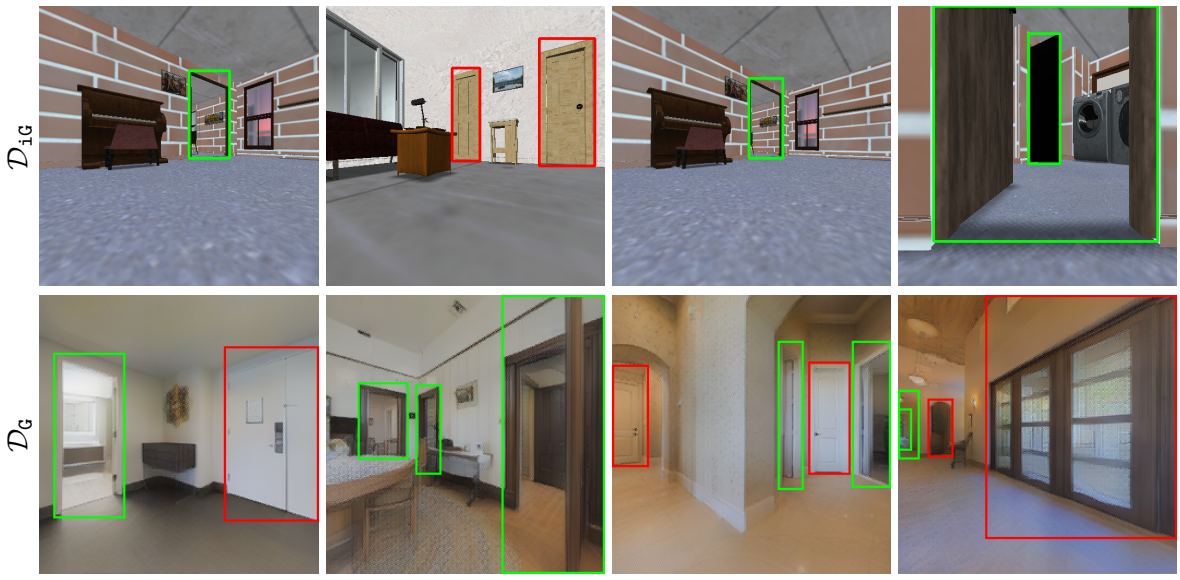


Figure 3.7: Example of annotated images obtained from simulations.

The final dataset in our study, named $\mathcal{D}_{\texttt{real}}$, is collected from a real deployment scenario of a service robot. This dataset consists of images acquired by a Giraff–X platform [36, 7], as depicted in Figure 3.1, during the exploration in 4 distinct indoor settings. These environments, as depicted in Figure 3.8, include a variety of settings. There is a university facility characterized by open spaces and classrooms (referred to as `Classrooms`), the floors of a department consisting of narrow corridors and regularly arranged offices (denoted as `Offices`), a research facility with laboratories (labeled as `Laboratories`), and a private apartment (identified as `House`). (In Figure 3.23–3.24 the floor plans of `Classrooms` and `Offices` are shown.) Data collection was performed using an Orbbec Astra RGB–D camera (the lower camera attached to the robot in Figure 3.1), capturing 320x240 RGB images at a rate of 1 fps. The dataset is composed of 3669 images in which open and closed doors are equally distributed (approximately 4000 instances per label). The images were then manually annotated with a particular attention on challenging door instances that are particularly relevant for our experimental campaign.

(e_1) `Classrooms` (e_2) `Offices` (e_3) `Laboratories` (e_4) `House`

Figure 3.8: Real environments we consider.

	Acquisition Effort	*Labeling Effort*	*Photorealism*	*Robot POV*	*Num. of Examples*
$\mathcal{D}_{\texttt{DD2}}$	**Medium** – Acquisitions taken by an operator	**High** – Manual labeling is required	**High** – Real–world images	**No**	$\approx$ 3000 images, from several environments
$\mathcal{D}_{\texttt{iG}}$	**Low** – Automatized batch acquisition	**Low** – Labels provided by the simulator	**Low** – The simulator uses synthetic graphics	**Yes**	$\approx$ 35000 images, from 15 different environments
$\mathcal{D}_{\texttt{G}}$	**Low** – Automatized batch acquisition	**Medium** – Manual labeling aided by simulator	**Medium** – Real–world scans with sim–2–real	**Yes**	$\approx$ 5500 images, from 10 different environments
$\mathcal{D}_{\texttt{real}}$	**High** – Real–robot deployment required	**High** – Manual labeling is required	**High** – Real–world images	**Yes**	$\approx$ 3700 images, from 4 different environments

Table 3.1: Overview of the main features of the datasets we built.

These datasets collectively offer a comprehensive overview of the trade–offs involved in training a door detector. $\mathcal{D}_{\texttt{DD2}}$ showcases what is typically available in literature but comes with significant drawbacks: the extensive effort needed for labeling and the lack in representing a robot's perception model. $\mathcal{D}_{\texttt{iG}}$ and $\mathcal{D}_{\texttt{G}}$, on the other hand, are products of efforts to address this limitation through the use of simulation frameworks. $\mathcal{D}_{\texttt{iG}}$ maximizes the advantages of simulated data collection: images are acquired and annotated in large batches, automatically, and from a robot–centric perspective. However, this comes with a critical downside given by the lack of photorealism. Our results will demonstrate that $\mathcal{D}_{\texttt{G}}$ achieves a more favorable compromise, allowing for batch data collection from the robot's viewpoint with reasonable effort, while

ensuring a decent degree of photorealism and easing the manual annotation process. $\mathcal{D}_{\texttt{real}}$, representing the ideal data set, offers the most authentic data but its high acquisition costs make it impractical for large–scale training. Table 3.1 summarizes these points, giving a broad comparison of the key characteristics of each dataset together with the number of samples.

3.4.3 Training and Testing

The general detectors are obtained by re–training the pre–trained versions of DETR, YOLOv5, and Faster R–CNN on COCO 2017 [24] using the following datasets: $\mathcal{D}_{\texttt{iG}}$, $\mathcal{D}_{\texttt{DD2}}$, $\mathcal{D}_{\texttt{G}}$, and $\mathcal{D}_{\texttt{DD2+G}}$. The last dataset, $\mathcal{D}_{\texttt{DD2+G}}$, is obtained by joining the examples of $\mathcal{D}_{\texttt{DD2}}$ and $\mathcal{D}_{\texttt{G}}$. We reduced the output layers of the three models to match the number of predicted object categories from 80 to 2. Then, we set our training parameters after a preliminary experimental campaign that explored various batch sizes ($\{1, 2, 4, 16, 32\}$) and epoch numbers ($\{20, 40, 60\}$). Training is performed keeping the first layers of the models' backbones frozen, as reported in Table 3.2. For Faster R–CNN and YOLOv5, we trained for 60 epochs with a batch size of 4, while DETR was trained for 60 epochs with a batch size of 1. We kept the other training hyperparameters (e.g., optimizer, learning rate, ...) as in [55, 54, 21] and we report the main ones in Table 3.2. We test the general detectors in each one of the 4 real environments e_1, e_2, e_3, e_4 of $\mathcal{D}_{\texttt{real}}$, depicted in Figure 3.8. For each environment e, we retain the randomly chosen 25% of the images as test set, called $\mathcal{D}^{\texttt{T}}_{\texttt{real,e}}$.

Then, we proceed with the qualification of the general detectors trained with $\mathcal{D}_{\texttt{DD2}}$, $\mathcal{D}_{\texttt{G}}$, and $\mathcal{D}_{\texttt{DD2+G}}$ on the environments of $\mathcal{D}_{\texttt{real}}$. The GD based on $\mathcal{D}_{\texttt{iG}}$ is not used, due to its unsatisfactory performance in the real world (see Section 3.5.1). To ease presentation, we say that a QD is *based on* a dataset $\mathcal{D}_{\texttt{x}}$ when it is obtained from a GD trained on such a dataset. Considering each real environment e, we performed a series of fine–tuning rounds of each general detector using increasing amounts of data from e (without considering the examples in $\mathcal{D}^{\texttt{T}}_{\texttt{real,e}}$). Doing this, we obtained a set of qualified detectors denoted as QD^{15}_e, QD^{25}_e, QD^{50}_e, and QD^{75}_e, where the superscripts denote the percentage of examples randomly chosen from $\mathcal{D}_{\texttt{real,e}}$ (the real data acquired in environment e) used for fine–tuning and can be interpreted as an indicator of the cost to acquire and label the additional samples. The fine–tuning is conducted using the same training parameters reported in Table 3.2, reducing the epochs to 40. Each qualified detector QD^x_e is tested in the corresponding environment e using the previously defined test set $\mathcal{D}^{\texttt{T}}_{\texttt{real,e}}$ (random 25% of images from $\mathcal{D}_{\texttt{real,e}}$ not used in any qualification round).

3.4.4 Performance Metrics

Our first performance metric is the mean Average Precision score (mAP), which averages the AP across all object categories (in our case, `open` and `closed` doors). The AP, as defined in [32], is the area under the precision/recall curve that is interpolated at 11 evenly spaced recall levels. In our evaluation, we refine this approach by introducing additional interpolation points at each recall level where the precision reaches a local maximum. This enhancement provides a more

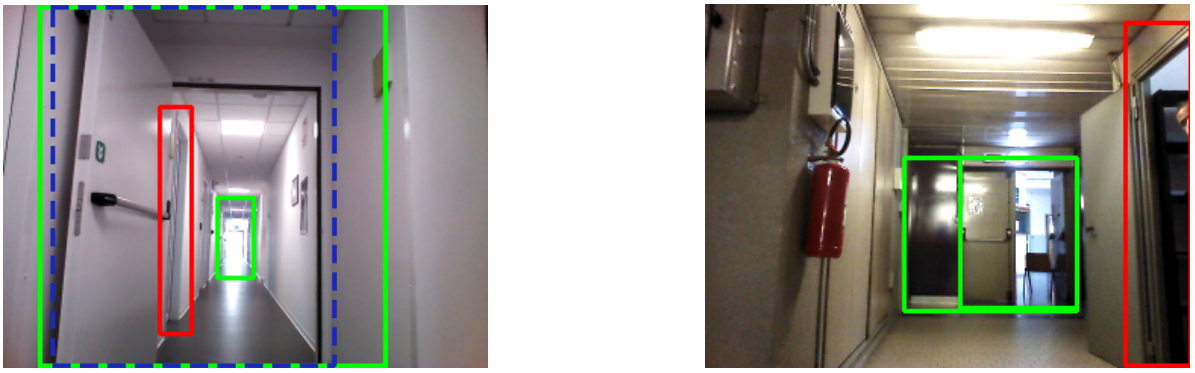
Hyperparameter	DETR	YOLOv5	Faster R–CNN
Epochs (GD/QD)	60/40	60/40	60/40
Fixed layers	11	27	11
Batch size	1	4	4
Optimizer	AdamW [180]	SGD	SGD
Learning rate	10^{-5}	10^{-2}	10^{-3}
Weight decay	10^{-4}	5×10^{-4}	5×10^{-4}
Momentum	–	9.37×10^{-1}	9×10^{-1}
Scheduler	–	LambdaLR	StepLR
Step size	–	1	3

Table 3.2: Hyperparameters used for training the general and the qualified detectors based on DETR [55], YOLOv5 [54], and Faster R–CNN [21]. "Frozen layers" refers to the CNN backbone layers keeping fixed during training (starting from the first). The learning rate of the CNN backbone of DETR is further decreased to 10^{-6} as in the original implementation [55]. The learning rate scheduler LambdaLR linearly reduces the learning rate by subtracting $\lambda = 1.65 \times 10^{-4}$ every epoch while StepLR multiplies the learning rate by a factor $\gamma = 10^{-1}$ every 3 epochs.

detailed approximation of the precision/recall curve, resulting in a more accurate assessment. To better align the AP to our robotics context, where object detection is used for the robot's decision–making, we set the threshold of the Intersection over Union (IoU) area for positive predictions $\rho_a = 50\%$. This tailors the AP to measure the correctness of door states instead of penalizing marginal localization errors that do not prevent the bounding boxes from being used to carry out robotics downstream tasks. Furthermore, we consider in the AP calculation only those bounding boxes with a confidence value $\geq 75\%$, thus reflecting the operational need of mobile robots in considering only high–confident predictions to avoid wrong decisions and prevent failures.

While the mAP is a widely accepted metric for object detection tasks, it has notable limitations in our robotic context. On one hand, certain errors disproportionately affect the AP relative to their actual impact on the robot's functionality. For instance, as illustrated in Figure 3.9a, minor inaccuracies in bounding box localization may have minimal effect on a service robot that is often primarily concerned with recognizing a door's traversability status rather than its precise localization. Furthermore, the AP treats multiple bounding boxes for the same door, as seen in Figure 3.9b, as false positives. However, a robot can resolve such ambiguities using additional information like its estimated pose and the map of the environment. On the other hand, the AP may not adequately reflect the severity of errors in identifying a door's traversability status if the bounding box is otherwise accurate. Once again, these errors are treated as false positives but, in our scenario, incorrectly classifying a `closed` door as `open` (or *vice versa*) can significantly impact the robot's efficiency, especially when these classifications inform the robot's decisions. An example of this type of error is depicted in Figure 3.9b.

Given these shortcomings, we suggest incorporating additional metrics better suited to the specific needs of the robotic application domain where door detection is crucial. These metrics are based on the premise that a service robot will invariably employ a method to sift through and select the most reliable predictions from a door detector. This process typically involves prioritizing high–confidence predictions and aggregating multiple bounding boxes that are localized in the same image region. The following definitions aim to encapsulate this approach,



(a) (b)

Figure 3.9: Errors made by a detector on Giraff–X (Figure 3.1). In (a) the foreground green bounding box is only slightly misaligned compared to its ground truth (in dashed blue). The error affects the AP but not the robot's typical task. Similarly, in (b) the two large green bounding boxes at the corridor's end correctly refer to the same open door; on the right, the closed door is a false positive. While the two errors affect the mAP similarly, the former is of little interest in the robotic domain, but the latter is critical for a navigating robot.

as well as enable the assessment of the asymmetrical nature of detection errors as previously discussed.

The overall procedure for the calculation of the additional metrics is detailed in Algorithm 3. Consider the i–th image $x^i \in X$ and call Y^i and $\hat{Y}^i$ the set of doors present in that image and the set of predictions computed by the detector, respectively (line 2). Given a predicted bounding box $\hat{y}$, we denote as $c(\hat{y})$ the confidence associated to it by the detector and we select those predictions whose confidence is above a threshold ρ_c, that is $\hat{Y}^i_c = \{\hat{y} \in \hat{Y}^i \mid c(\hat{y}) \geq \rho_c\}$ (line 4). Given two bounding boxes y_1 and y_2, we denote as $a_I(y_1, y_2)$ and $a_U(y_1, y_2)$ the area of their intersection and union, respectively. We compute the set of *Background False Detections* (BFD) as the confident predictions that cannot be assigned to any real door based on a threshold ρ_a on their maximum Intersection Over Union area (IOU) (line 5). Formally,

$$BFD^i = \left\{ \hat{y} \in \hat{Y}^i_c \middle| \max_{y \in Y^i} \frac{a_I(\hat{y}, y)}{a_U(\hat{y}, y)} < \rho_a \right\}.$$

BFDs occur when a robot mistakenly identifies a door in locations where none exists, such as on a wall or a closet. As previously discussed, this type of error relates to the mislocalization of doors. In principle, a robot might correct such errors using information from its navigation stack. For example, the robot could infer from its map that a door cannot exist in a place designated as a wall. Therefore, provided these errors are not excessively frequent, they are generally deemed acceptable within typical robotic scenarios.

Confident predictions that, instead, are well localized and have an above–threshold IOU for at least one door in the image are contained in a set called $\hat{Y}^i_{c,a} = \hat{Y}^i_c \setminus BFD^i$. This allows us to define, for each ground truth door y, the set of predictions that are confident and whose

area is maximally matched with it (line 8), formally

$$B(y) = \left\{ \hat{y} \in \hat{Y}^i_{c,a} \middle| \arg\max_{y \in Y^i} \frac{a_I(\hat{y}, y)}{a_U(\hat{y}, y)} = y \right\}.$$

(Notice that, provided that ties are broken, the same prediction can never be matched to more than one door.)

Algorithm 3 Calculation of the Operational Performance Indicators

Input: $Y = \{Y^i\}$, $\hat{Y} = \{\hat{Y}^i\}$: the sets of ground truth and predicted doors divided for each image i

Output: $TP_\%$, $FP_\%$, $BFD_\%$, the Operational Performance Indicators

1: $TP_\%, FP_\%, BFD_\%, \overline{Y} \leftarrow 0$ ▷ Initialization
2: **for** $Y^i, \hat{Y}^i \in Y, \hat{Y}$ **do**
3: $\overline{Y} \leftarrow \overline{Y} + |Y^i|$
4: $\hat{Y}^i_c \leftarrow \{\hat{y} \in \hat{Y}^i \mid c(\hat{y}) \geq \rho_c\}$ ▷ Select the most confident prediction
5: $BFD^i \leftarrow \left\{\hat{y} \in \hat{Y}^i_c \middle| \max_{y \in Y^i} \frac{a_I(\hat{y},y)}{a_U(\hat{y},y)} < \rho_a\right\}$
6: $TP^i, FP^i \leftarrow \emptyset$
7: **for** $y \in Y^i$ **do**
8: $B(y) \leftarrow \left\{\hat{y} \in \hat{Y}^i_c \setminus BFD^i \middle| \arg\max_{y \in Y^i} \frac{a_I(\hat{y},y)}{a_U(\hat{y},y)} = y\right\}$
9: $\hat{y}^* = \arg\max_{\hat{l} \in B(y)} c(\hat{y})$
10: **if** $l(\hat{y}^*) = l(y)$ **then**
11: $TP^i \leftarrow TP^i \cup \{\hat{y}^*\}$
12: **else** $FP^i \leftarrow FP^i \cup \{\hat{y}^*\}$
13: **end if**
14: **end for**
15: $TP_\%, \leftarrow TP_\% + |TP^i|$
16: $FP_\% \leftarrow FP_\% + |FP^i|$
17: $BFD_\% \leftarrow BFD_\% + |BFD^i|$
18: **end for**
19: $TP_\%, FP_\%, BFD_\% \leftarrow \frac{TP_\%}{\overline{Y}}, \frac{FP_\%}{\overline{Y}}, \frac{BFD_\%}{\overline{Y}}$

Finally, we define $\hat{y}^* = \arg\max_{\hat{l} \in B(y)} c(\hat{y})$ as the most confident prediction for door y (line 9), and it is this prediction we focus on, discarding any other predictions for the same door. We denote as $l(\hat{y})$ the label assigned to the prediction $\hat{y}$ by the object detector. If $\hat{y}^*$ correctly predicts the traversability of door y, it is included in the set of true positives (TP^i) (line 11). Conversely, if $\hat{y}^*$ incorrectly predicts the traversability status, it is assigned to the set of false positives (FP^i) (line 12). A false positive substantially differs from a BFD, as an FP is potentially more consequential. An FP can lead the robot to incorrectly assess a critical aspect of the environment's topology, such as mistaking a closed door for an open passage, which could significantly impact its decisions (notice how, in this example, the environment's map cannot be exploited to fix the error). In our evaluation, we apply the aforementioned method

across all images, defining

$$TP_{\%} = \frac{\sum_i |TP^i|}{\overline{Y}},\ FP_{\%} = \frac{\sum_i |FP^i|}{\overline{Y}},\ \text{and}\ BFD_{\%} = \frac{\sum_i |BFD^i|}{\overline{Y}},$$

where $\overline{Y} = \sum_i |Y^i|$. We call these *Operational Performance Indicators* (OPI), they represent the rates of true positives, false positives, and BFDs, respectively. In our experiments, the confidence threshold ρ_c is set to 75%, and the IOU threshold ρ_a is set to 50%.

3.5 Results

Here, we present and discuss the obtained results both with our general detectors (Section 3.5.1) and with the qualified ones (Section 3.5.2). We then assess the effectiveness of the qualification procedure in long–term robot deployments by testing the robustness of the qualified detectors on data with feature shifts and focusing on challenging door instances (results and discussion are in Section 3.5.3). After that, we show how the increase in performance due to our pipeline is general regardless of the object detection method used. To do so, we compare the results obtained with three popular object detection architectures and we select the configuration that better suits our target problem (Section 3.5.4). Finally, we study the impact of different performing door detectors on topology mapping, a downstream task useful to improve the long–term navigation capabilities of service robots that requires door detection [60, 61]. This last evaluation is reported in Section 3.5.5.

3.5.1 Evaluation of General Detectors

In this section, we evaluate our pipeline for synthesizing a GD using the training parameters of Section 3.4.3. The performance metrics are detailed in Table 3.3 and visually summarized in Figures 3.10 and 3.11. The mAP bars in Figure 3.10 are composed of a dashed and an undashed part, stating the AP contributions of the two labels (`open` door and `closed` door) to the final mAP value. Ideally, we want the dashed and undashed parts of the same size, i.e. a model that is equally able to detect both categories. The same representation is used for similar plots reporting mAP (Figures 3.13, 3.16, and 3.20). First, notice how the general detectors trained on $\mathcal{D}_{\texttt{iG}}$ exhibit very poor performance, as indicated by the blue bars in the figures. To elaborate, the YOLOv5–based GD correctly identified only one door instance (in e_4 – `House`). Meanwhile, its counterparts, DETR and Faster R–CNN, incur a high number of errors in terms of $FP_{\%}$ and $BFD_{\%}$ (as illustrated in Figure 3.11), which outweighs their very limited number of correct detections. These unsurprising outcomes confirm the intuition that training with simulations, even those designed to replicate real environmental features, is ineffective if they lack photorealism. This conclusion is further supported by observing the significant performance improvements achieved when transitioning from training with $\mathcal{D}_{\texttt{iG}}$ to $\mathcal{D}_{\texttt{DD2}}$ (among our training datasets, the one that maximizes photorealism).

Env.	Dataset	DETR [55] mAP↑	DETR [55] TP%↑	DETR [55] FP%↓	DETR [55] BFD%↓	YOLOv5 [54] mAP↑	YOLOv5 [54] TP%↑	YOLOv5 [54] FP%↓	YOLOv5 [54] BFD%↓	Faster R–CNN [21] mAP↑	Faster R–CNN [21] TP%↑	Faster R–CNN [21] FP%↓	Faster R–CNN [21] BFD%↓
e_1	$\mathcal{D}_{\texttt{iG}}$	0	1	**0**	26	0	0	**0**	**0**	2	2	**0**	**2**
	$\mathcal{D}_{\texttt{DD2}}$	13	18	9	**13**	2	3	2	1	18	25	14	9
	$\mathcal{D}_{\texttt{G}}$	26	30	7	22	30	31	1	8	20	25	6	11
	$\mathcal{D}_{\texttt{DD2+G}}$	**32**	**37**	6	17	**32**	**34**	2	3	**34**	**43**	10	14
e_2	$\mathcal{D}_{\texttt{iG}}$	0	1	**1**	22	0	0	**0**	**0**	0	1	**0**	**3**
	$\mathcal{D}_{\texttt{DD2}}$	14	19	8	**17**	3	5	1	3	22	27	4	18
	$\mathcal{D}_{\texttt{G}}$	**28**	**36**	6	21	14	21	9	9	14	17	4	10
	$\mathcal{D}_{\texttt{DD2+G}}$	24	31	10	19	**16**	**24**	10	9	**27**	**34**	5	20
e_3	$\mathcal{D}_{\texttt{iG}}$	0	2	**0**	35	0	0	**0**	**1**	0	1	**1**	11
	$\mathcal{D}_{\texttt{DD2}}$	9	15	3	**30**	3	3	0	1	10	20	8	40
	$\mathcal{D}_{\texttt{G}}$	13	19	6	33	4	6	3	10	2	4	2	**10**
	$\mathcal{D}_{\texttt{DD2+G}}$	**16**	**24**	4	44	**6**	**10**	2	12	**14**	**24**	8	34
e_4	$\mathcal{D}_{\texttt{iG}}$	1	5	**3**	25	0	0	**1**	4	1	3	7	7
	$\mathcal{D}_{\texttt{DD2}}$	22	20	14	**9**	14	12	3	**1**	31	35	9	14
	$\mathcal{D}_{\texttt{G}}$	31	**40**	9	11	16	22	2	4	12	18	**4**	**6**
	$\mathcal{D}_{\texttt{DD2+G}}$	**32**	35	10	13	**30**	**34**	7	7	**48**	**49**	7	16

Table 3.3: Real–World performance of general detectors. The best and second–best results among the training datasets are highlighted in bold and underlined, respectively.

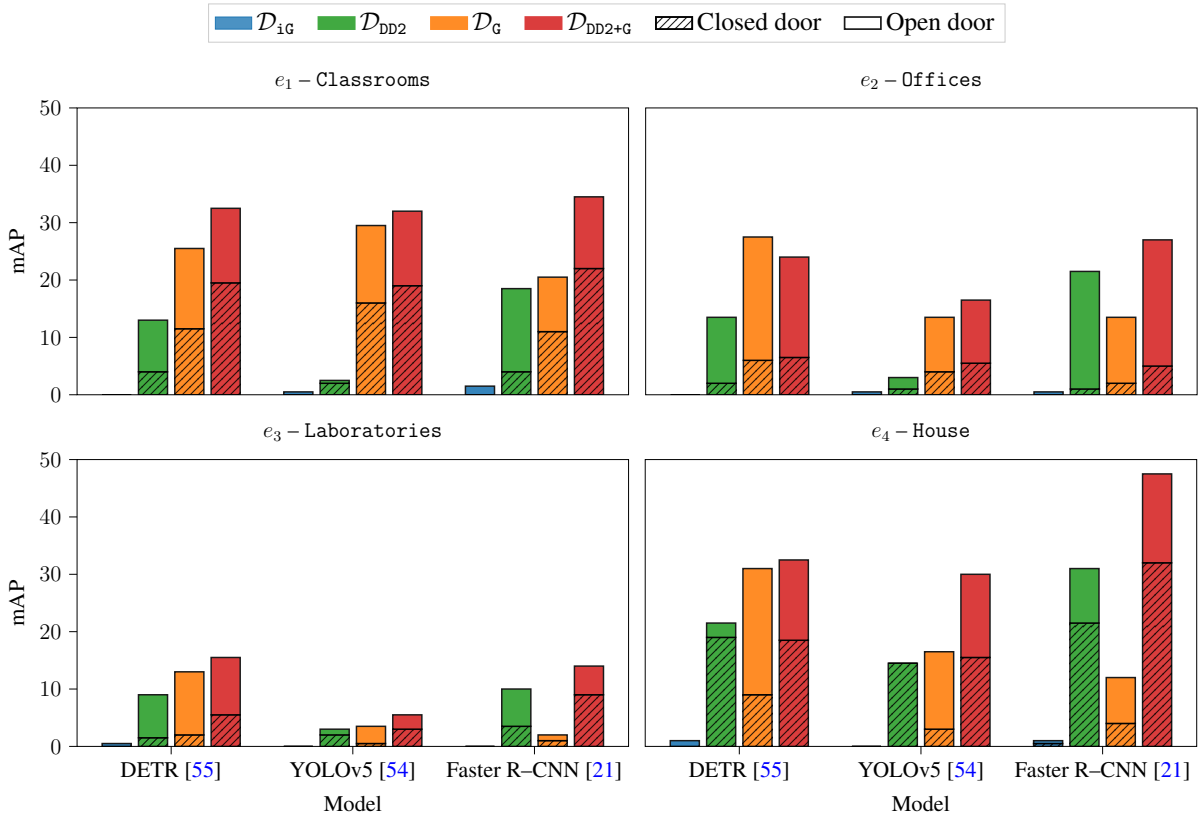


Figure 3.10: mAP of the general detectors trained with the 4 datasets in real environments.

An interesting and perhaps counter–intuitive observation emerges when comparing the training results of $\mathcal{D}_{\texttt{DD2}}$ (real–world images) with $\mathcal{D}_{\texttt{G}}$ (our simulation framework outlined in Section 3.2). Common intuition suggests that a detector trained on real–world data should out-

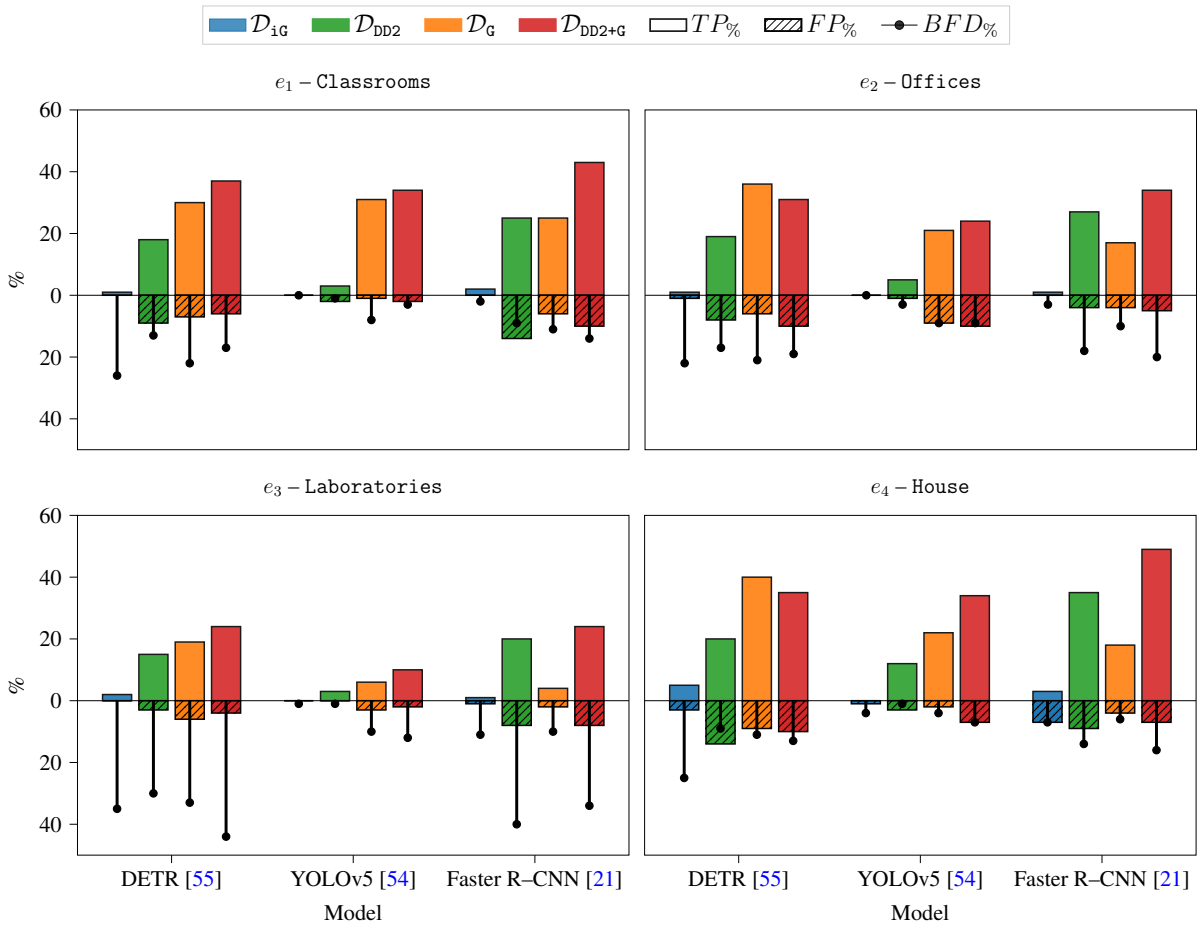


Figure 3.11: Operational performance indicators of the GDs trained with the 4 datasets.

perform one trained on a simulation, even if photorealistic. However, as shown by the mAP scores in Figure 3.10 and the $TP_{\%}$ in Figure 3.11, we see that two out of the three detectors, specifically those based on DETR and YOLOv5, actually have better performance when trained on $\mathcal{D}_{\text{G}}$ rather than $\mathcal{D}_{\text{DD2}}$. This result indicates that while photorealism, a characteristic highly present in $\mathcal{D}_{\text{DD2}}$, is important, it is not the unique key feature for creating effective general detectors for robots. It appears that the slightly compromised visual quality in $\mathcal{D}_{\text{G}}$ might be effectively balanced by a closer alignment with a robot's perception model, thereby reducing, to some extent, the sim–to–real gap. This also suggests that in real robot deployments, the shift in data distribution might be more significantly influenced by the data acquisition process rather than by the characteristics of the input space.

This trend does not hold for the detector based on Faster R–CNN, which shows better results with $\mathcal{D}_{\text{DD2}}$. Upon closer examination, this can be attributed to the Region Proposal Network, which, by localizing and classifying bounding boxes based on features extracted from the Pyramid Backbone, is more sensitive to the photorealistic quality of images. To support this observation, we consider the performance of Faster R–CNN trained on $\mathcal{D}_{\text{DD2+G}}$, a dataset that combines $\mathcal{D}_{\text{DD2}}$'s high photorealism with $\mathcal{D}_{\text{G}}$'s representation of the robot's viewpoint. As indicated by the red bars in Figure 3.10, Faster R–CNN's performance improves, while DETR and YOLOv5 are only slightly impacted by the absence of real–world data. The $TP_{\%}$ in Figure 3.11

Figure 3.12: Real–world door instances correctly recognized by GDs trained on $\mathcal{D}_{\texttt{DD2+G}}$.

shows that the correct door status detections with $\mathcal{D}_{\texttt{DD2+G}}$ slightly surpass those with $\mathcal{D}_{\texttt{G}}$. In some cases, our simulated data even yield better results, as Ad by DETR's performance in environments e_2 and e_3. However, it's noteworthy that mixing training data often leads to an increase in erroneous detections, as evidenced by the $FP_{\%}$ and $BFD_{\%}$ indicators in Figure 3.11.

These results prove the effectiveness of our simulation framework, which strikes a balance between photorealism and alignment with the robot's perception model. This approach is hence both viable and efficient for building general door detectors, reducing training costs while still achieving an acceptable performance level. The general detectors we developed are capable of accurately recognizing doors across diverse real–world environments, demonstrating a fair level of generalization. However, this strength is mainly evident in straightforward door instances, and less so in more complex ones involving occluded views, multiple nested doors, or combinations of these. Figure 3.12 showcases some representative examples where our GDs excel. To bridge the gap in identifying such difficult cases, it is essential to qualify the general detectors for the target environment where they are set to operate.

3.5.2 Evaluation of Qualified Detectors

In this section, we assess how the process of qualifying a model to the robot's target environments enhances performances when compared with those of a general detector. The detailed results can be found in Table 3.4. Data are collected by using all three methods (DETR, YOLOv5, Faster R–CNN) and averaged. The same setting is also used in Section 3.5.3 and for the remainder of this work.

Env.	Exp.	$\mathcal{D}_{\texttt{DD2}}$ mAP↑	$\mathcal{D}_{\texttt{DD2}}$ TP%↑	$\mathcal{D}_{\texttt{DD2}}$ FP%↓	$\mathcal{D}_{\texttt{DD2}}$ BFD%↓	$\mathcal{D}_{\texttt{G}}$ mAP↑	$\mathcal{D}_{\texttt{G}}$ TP%↑	$\mathcal{D}_{\texttt{G}}$ FP%↓	$\mathcal{D}_{\texttt{G}}$ BFD%↓	$\mathcal{D}_{\texttt{DD2+G}}$ mAP↑	$\mathcal{D}_{\texttt{DD2+G}}$ TP%↑	$\mathcal{D}_{\texttt{DD2+G}}$ FP%↓	$\mathcal{D}_{\texttt{DD2+G}}$ BFD%↓
e_1	GD	11 ± 8	15 ± 11	8 ± 6	$\mathbf{8} \pm 6$	25 ± 5	29 ± 3	$\mathbf{5} \pm 3$	14 ± 7	$\mathbf{33} \pm 1$	$\mathbf{38} \pm 5$	6 ± 4	11 ± 7
	QD_e^{15}	49 ± 21	53 ± 19	$\mathbf{3} \pm 2$	$\mathbf{10} \pm 9$	63 ± 12	67 ± 9	3 ± 1	15 ± 15	$\mathbf{63} \pm 7$	$\mathbf{67} \pm 7$	3 ± 2	14 ± 10
	QD_e^{25}	59 ± 20	63 ± 18	$\mathbf{2} \pm 1$	16 ± 15	72 ± 14	75 ± 12	3 ± 2	$\mathbf{14} \pm 13$	$\mathbf{73} \pm 12$	$\mathbf{76} \pm 9$	2 ± 2	15 ± 12
	QD_e^{50}	72 ± 19	76 ± 16	$\mathbf{1} \pm 1$	13 ± 13	$\mathbf{81} \pm 9$	83 ± 7	1 ± 1	$\mathbf{11} \pm 9$	80 ± 10	$\mathbf{83} \pm 9$	1 ± 1	11 ± 8
	QD_e^{75}	78 ± 15	81 ± 13	$\mathbf{1} \pm 1$	11 ± 9	85 ± 9	87 ± 7	1 ± 1	$\mathbf{9} \pm 7$	$\mathbf{85} \pm 8$	$\mathbf{87} \pm 7$	1 ± 1	9 ± 6
e_2	GD	13 ± 9	17 ± 11	$\mathbf{4} \pm 4$	$\mathbf{13} \pm 8$	18 ± 8	25 ± 10	6 ± 3	13 ± 7	$\mathbf{22} \pm 5$	$\mathbf{30} \pm 5$	8 ± 3	16 ± 6
	QD_e^{15}	48 ± 21	53 ± 19	3 ± 1	$\mathbf{16} \pm 16$	$\mathbf{65} \pm 12$	$\mathbf{69} \pm 9$	$\mathbf{2} \pm 1$	24 ± 25	62 ± 11	66 ± 10	3 ± 1	24 ± 19
	QD_e^{25}	60 ± 18	66 ± 17	$\mathbf{3} \pm 2$	23 ± 24	71 ± 8	74 ± 7	3 ± 1	$\mathbf{17} \pm 16$	$\mathbf{72} \pm 10$	$\mathbf{76} \pm 8$	3 ± 0	20 ± 19
	QD_e^{50}	70 ± 17	74 ± 14	$\mathbf{2} \pm 0$	18 ± 19	80 ± 8	83 ± 6	2 ± 1	$\mathbf{14} \pm 11$	$\mathbf{80} \pm 7$	$\mathbf{84} \pm 7$	2 ± 1	16 ± 17
	QD_e^{75}	75 ± 13	80 ± 9	$\mathbf{2} \pm 0$	18 ± 18	$\mathbf{84} \pm 5$	$\mathbf{86} \pm 4$	2 ± 1	$\mathbf{12} \pm 11$	82 ± 7	85 ± 6	2 ± 1	18 ± 15
e_3	GD	7 ± 4	13 ± 9	$\mathbf{4} \pm 4$	24 ± 20	6 ± 6	10 ± 8	4 ± 2	$\mathbf{18} \pm 13$	$\mathbf{12} \pm 5$	$\mathbf{19} \pm 8$	5 ± 3	30 ± 16
	QD_e^{15}	41 ± 23	48 ± 20	$\mathbf{2} \pm 1$	$\mathbf{26} \pm 20$	55 ± 16	63 ± 11	5 ± 3	27 ± 20	$\mathbf{56} \pm 14$	$\mathbf{63} \pm 10$	4 ± 2	33 ± 24
	QD_e^{25}	54 ± 25	59 ± 21	$\mathbf{3} \pm 2$	$\mathbf{21} \pm 18$	64 ± 19	70 ± 13	3 ± 3	26 ± 28	$\mathbf{68} \pm 13$	$\mathbf{75} \pm 9$	3 ± 2	21 ± 15
	QD_e^{50}	68 ± 21	74 ± 16	$\mathbf{2} \pm 1$	19 ± 18	76 ± 13	81 ± 10	2 ± 2	$\mathbf{17} \pm 15$	$\mathbf{76} \pm 15$	$\mathbf{81} \pm 12$	3 ± 2	19 ± 17
	QD_e^{75}	76 ± 19	81 ± 15	$\mathbf{1} \pm 1$	14 ± 14	82 ± 12	86 ± 9	1 ± 1	$\mathbf{12} \pm 10$	$\mathbf{82} \pm 11$	$\mathbf{86} \pm 8$	1 ± 1	15 ± 16
e_4	GD	22 ± 8	22 ± 12	9 ± 6	8 ± 7	20 ± 10	27 ± 12	$\mathbf{5} \pm 4$	$\mathbf{7} \pm 4$	$\mathbf{37} \pm 9$	$\mathbf{39} \pm 8$	8 ± 2	12 ± 5
	QD_e^{15}	60 ± 18	64 ± 19	2 ± 1	18 ± 12	76 ± 4	77 ± 4	$\mathbf{1} \pm 1$	$\mathbf{14} \pm 7$	$\mathbf{76} \pm 6$	$\mathbf{78} \pm 8$	3 ± 2	18 ± 16
	QD_e^{25}	70 ± 18	73 ± 17	$\mathbf{2} \pm 1$	14 ± 11	79 ± 7	81 ± 5	3 ± 1	14 ± 9	$\mathbf{82} \pm 8$	$\mathbf{83} \pm 8$	4 ± 2	$\mathbf{12} \pm 9$
	QD_e^{50}	81 ± 10	84 ± 10	2 ± 1	13 ± 11	$\mathbf{91} \pm 4$	92 ± 3	$\mathbf{1} \pm 1$	$\mathbf{8} \pm 6$	90 ± 5	$\mathbf{92} \pm 3$	1 ± 1	8 ± 7
	QD_e^{75}	90 ± 8	91 ± 6	1 ± 1	5 ± 4	96 ± 2	96 ± 2	$\mathbf{0} \pm 1$	4 ± 3	$\mathbf{96} \pm 2$	$\mathbf{96} \pm 2$	0 ± 0	$\mathbf{3} \pm 2$

Table 3.4: Results of the qualification procedure (averaged over detectors, together with the standard deviations) when the GD is trained with the $\mathcal{D}_{\texttt{DD2}}$, $\mathcal{D}_{\texttt{G}}$, and $\mathcal{D}_{\texttt{DD2+G}}$. Bold entries indicate the best performance on each metric across the three datasets.

A first evident observation is that the qualification procedure boosts the performance of the general detectors for the target environment and, unsurprisingly, the performance (together with the data preparation costs) increases as more samples are included, from QD_e^{15} to QD_e^{75}. This can be appreciated in the mAP and $TP_{\%}$ improvements visually depicted in Figures 3.13 and 3.14 and by the decreasing trend (after the first qualification round) of $FP_{\%}$ and $BFD_{\%}$ in Figure 3.14.

However, the increments follow a diminishing–returns trend, with large gains in the first qualification rounds and marginal ones as more data are used. Focusing on the average mAP and $TP_{\%}$ it can be seen how the qualified detector that scores the highest performance improvements is QD_e^{15}, despite requiring a relatively affordable effort for data preparation. From a practical perspective, this observation suggests how just a coarse visual inspection of the target environment might be enough to obtain an environment–specific detector whose performance is significantly better than the corresponding general one. In such a case the robot's deployment time is only marginally affected. To give a concrete idea, annotating the 15% of the data collected by the robot's first exploration of a new environment (approximately 80 images) required a human operator using our tool (Section 3.3) about half an hour. Another key finding is how the improvements through qualification are distributed across various types of instances encountered by the detector. Upon direct inspection, we observed how these were particularly

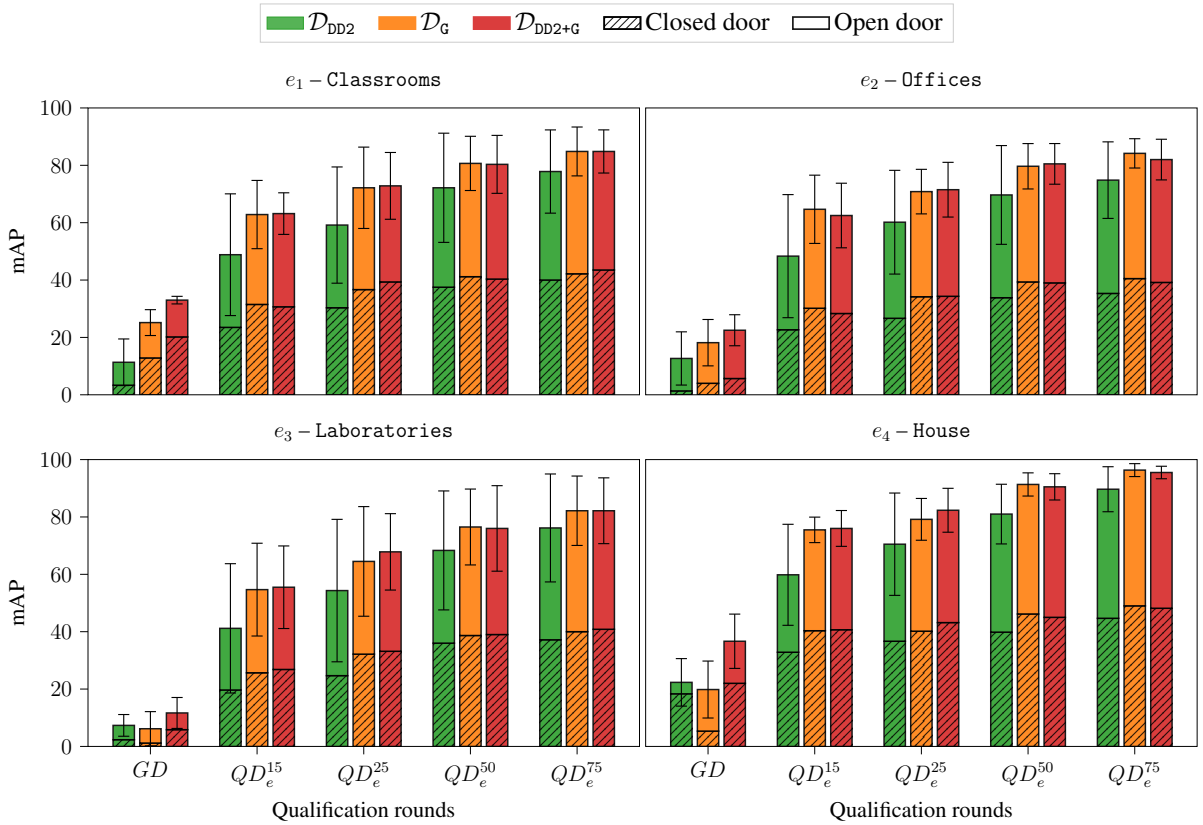


Figure 3.13: Real–world evaluation of the qualified detectors where GDs are based on different datasets. The mAP is averaged over the three models, for which we report also the standard deviation.

notable in challenging instances. Figure 3.15 showcases significant examples of this, illustrating how the QD_e^{15} model, based on our dataset $\mathcal{D}_{\texttt{G}}$, successfully detects doors in highly challenging instances. These include scenarios with nested or partially occluded doors and even situations where the door is hidden in the background.

It is important to notice that the dataset chosen to train the general detector does affect the benefits of the qualification. The trends observed in Figure 3.13 indicate that QDs based on $\mathcal{D}_{\texttt{DD2}}$ generally demonstrate lower performance compared to those based on $\mathcal{D}_{\texttt{G}}$ and $\mathcal{D}_{\texttt{DD2+G}}$. This observation is further supported by the data presented in Figure 3.14. Although the error rates ($FP_{\%}$ and $BFD_{\%}$), are substantially similar, there is a noticeable difference in the $TP_{\%}$. Specifically, detectors based on $\mathcal{D}_{\texttt{G}}$ or $\mathcal{D}_{\texttt{DD2+G}}$ show better $TP_{\%}$ performance than those based on $\mathcal{D}_{\texttt{DD2}}$. Confirming the findings from the previous section, this again suggests that training on images not representing the robot's point of view, although taken from the real world, hits a performance limit. A simulated dataset from the robot perspective with an adequate level of photorealism (as $\mathcal{D}_{\texttt{G}}$), when included in the training phase, enables the detectors to reach better performance when qualified for a target environment.

To further support the effectiveness of the method of Section 3.2, we can notice that $\mathcal{D}_{\texttt{DD2+G}}$, which integrates the realism in $\mathcal{D}_{\texttt{DD2}}$ and the robot perception model of $\mathcal{D}_{\texttt{G}}$, does not introduce significant variations in the performance of the qualified detectors when compared with those solely based on $\mathcal{D}_{\texttt{G}}$. This can be easily seen by comparing the (substantially similar) orange and

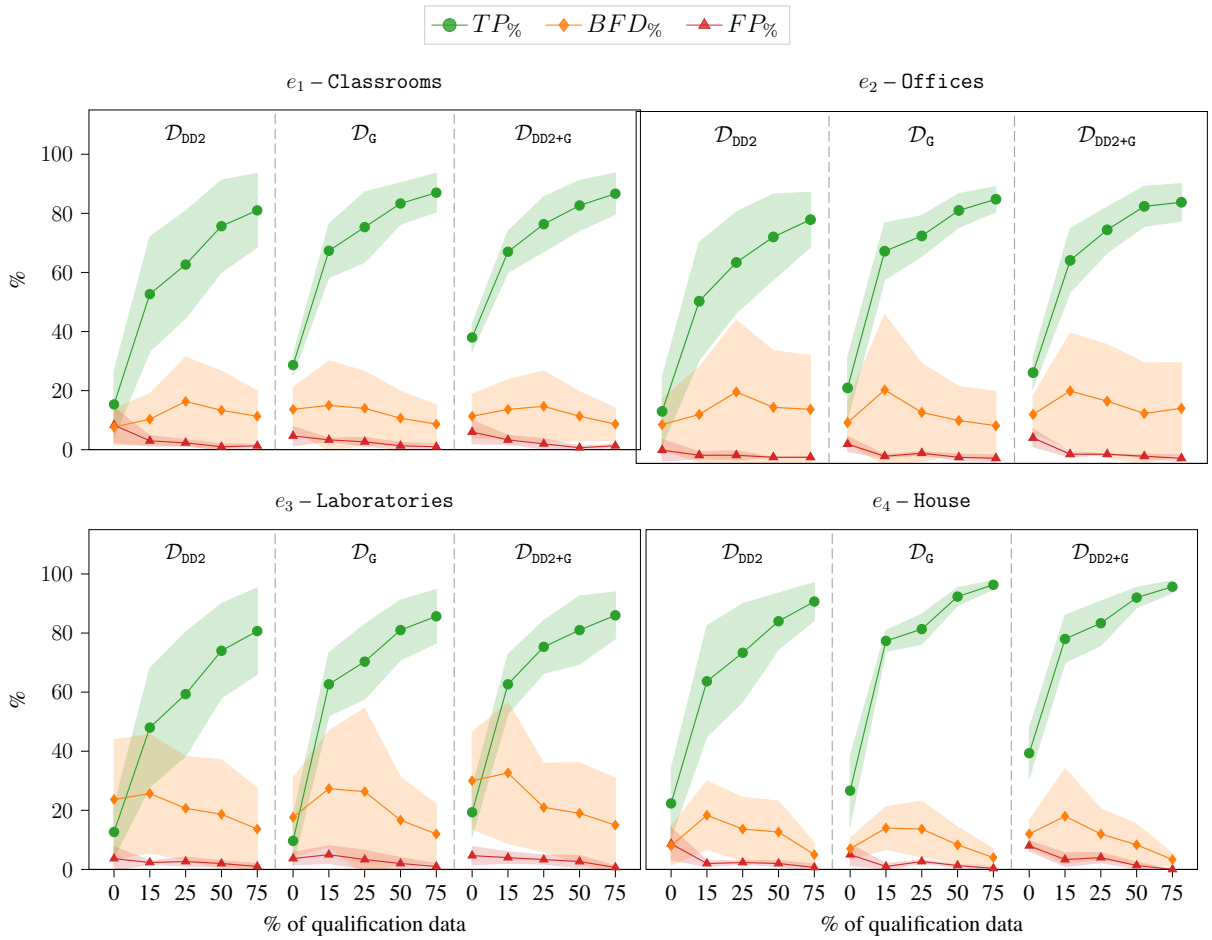


Figure 3.14: Operational performance indicators (averages over the three models with the standard deviation) with GDs trained on different datasets.

red bars of Figure 3.13 that refer to $\mathcal{D}_{\texttt{G}}$ and $\mathcal{D}_{\texttt{DD2+G}}$, respectively. In addition, while $TP_{\%}$ reaches comparable performance, Table 3.4 shows that $\mathcal{D}_{\texttt{G}}$ enables the qualified detectors to reduce the rate of BFD with respect to $\mathcal{D}_{\texttt{DD2+G}}$.

It is important to remark that the qualification procedure is effective if the detector is qualified and then used inside the same environment, a condition that perfectly matches the practical deployments of service robots. To assess this claim, we conduct additional experiments to assess the performance of qualified detectors on data from a different distribution (i.e., acquired from a different environment). To do this, we fine–tune the general detectors using data from one or more environments and we test using images from a new one (e.g., fine–tuning QD on e_1, e_2, e_3, testing on e_4). We observed that when a few examples are used for fine–tuning, this procedure results in minor performance improvements when compared with a GD; still, performances are below those of QD_e^{15} (trained with the data of the target environment). Moreover, using more data for qualification results in a performance drop as the qualified detectors overfit the training data that come from different environments to the one used for testing. We omit these results for the sake of brevity.

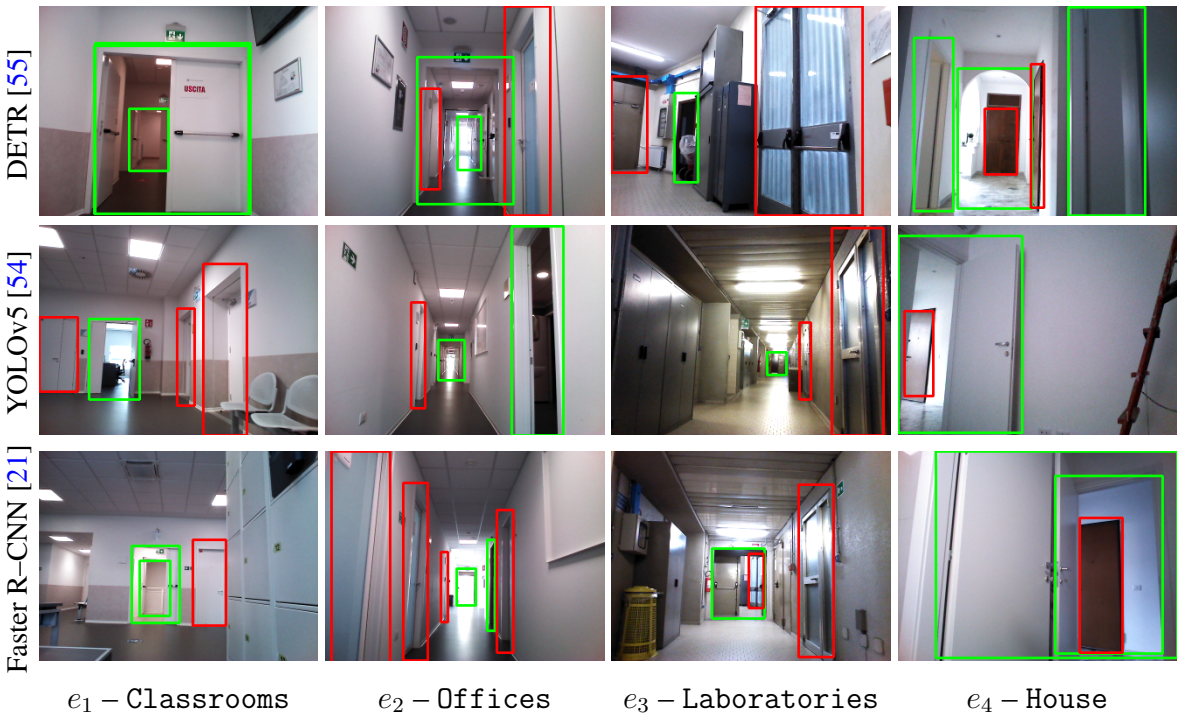


Figure 3.15: Challenging doors correctly detected by QD_e^{15} (GDs divided by model and trained on $\mathcal{D}_{\mathtt{G}}$).

3.5.3 Evaluation in Challenging Settings

As discussed in Section 3.3, the advantage represented by a qualified detector is enabled by the long–term deployment of the robot in the same target environment, where the same object instances get repeatedly observed. However, while the observed doors are the same, transient changes in the environment's appearance might still take place resulting in unpredictable domain shifts.

We deem that one of the most significant shifts might occur at the *feature level* of the robot's perceptions [151]. For a long–term deployment in a human–centric environment, we considered two possible factors of such a feature shift: the variations in illumination between day and night and dynamic camera occlusions caused by people walking around. In the first case, changes in lighting can significantly alter the appearance of doors and these variations in illumination can be widespread throughout the entire environment (e.g., day/night), or being localized (e.g., light reflections). In the second case, dynamic actors walking freely within the environment can obstruct the robot's view, especially in confined areas such as narrow corridors or passageways.

To test the robustness of our approach with light variations, we included in our real–world dataset (following the same procedure of Section 3.4.2) additional data from e_1 and e_2 during nighttime, when only artificial light is present and some rooms are entirely dark. For camera occlusions, we acquired new data in e_2 while having people intentionally walking by the robot or loitering in its vicinity. We used these data to test our qualified detectors (Section 3.5.2), which were trained during daytime hours when the environment was sparsely populated (as is typi-

	Env.	Exp.	$\mathcal{D}_{\texttt{DD2}}$ mAP↑	$\mathcal{D}_{\texttt{DD2}}$ TP%↑	$\mathcal{D}_{\texttt{DD2}}$ FP%↓	$\mathcal{D}_{\texttt{DD2}}$ BFD%↓	$\mathcal{D}_{\texttt{G}}$ mAP↑	$\mathcal{D}_{\texttt{G}}$ TP%↑	$\mathcal{D}_{\texttt{G}}$ FP%↓	$\mathcal{D}_{\texttt{G}}$ BFD%↓	$\mathcal{D}_{\texttt{DD2+G}}$ mAP↑	$\mathcal{D}_{\texttt{DD2+G}}$ TP%↑	$\mathcal{D}_{\texttt{DD2+G}}$ FP%↓	$\mathcal{D}_{\texttt{DD2+G}}$ BFD%↓
Nighttime - light variations	e_1	GD	10 ± 6	15 ± 8	5 ± 2	$\mathbf{9} \pm 6$	26 ± 2	31 ± 2	$\mathbf{4} \pm 1$	10 ± 4	$\mathbf{31} \pm 5$	$\mathbf{37} \pm 6$	6 ± 2	13 ± 6
		QD_e^{15}	27 ± 10	32 ± 10	$\mathbf{4} \pm 2$	$\mathbf{14} \pm 11$	41 ± 5	47 ± 2	4 ± 2	17 ± 13	$\mathbf{41} \pm 3$	$\mathbf{47} \pm 1$	4 ± 2	17 ± 12
		QD_e^{25}	39 ± 10	44 ± 9	$\mathbf{4} \pm 2$	$\mathbf{16} \pm 12$	$\mathbf{48} \pm 5$	53 ± 2	4 ± 3	18 ± 14	47 ± 5	$\mathbf{53} \pm 3$	5 ± 2	18 ± 13
		QD_e^{50}	44 ± 12	49 ± 11	$\mathbf{4} \pm 2$	15 ± 13	52 ± 4	57 ± 4	4 ± 1	$\mathbf{14} \pm 11$	$\mathbf{54} \pm 7$	$\mathbf{58} \pm 6$	4 ± 3	17 ± 13
		QD_e^{75}	50 ± 10	54 ± 9	5 ± 2	13 ± 11	$\mathbf{56} \pm 5$	$\mathbf{60} \pm 3$	5 ± 2	13 ± 11	52 ± 6	57 ± 5	$\mathbf{4} \pm 3$	$\mathbf{12} \pm 9$
	e_2	GD	12 ± 9	18 ± 14	$\mathbf{2} \pm 2$	$\mathbf{9} \pm 8$	16 ± 3	27 ± 4	6 ± 2	17 ± 10	$\mathbf{24} \pm 5$	$\mathbf{34} \pm 7$	5 ± 0	17 ± 8
		QD_e^{15}	30 ± 10	39 ± 17	$\mathbf{3} \pm 2$	$\mathbf{19} \pm 16$	39 ± 7	49 ± 11	3 ± 2	27 ± 21	$\mathbf{42} \pm 9$	$\mathbf{51} \pm 13$	3 ± 1	25 ± 19
		QD_e^{25}	39 ± 13	47 ± 15	4 ± 1	24 ± 19	48 ± 4	56 ± 5	4 ± 2	$\mathbf{22} \pm 16$	$\mathbf{50} \pm 4$	$\mathbf{59} \pm 6$	$\mathbf{3} \pm 1$	22 ± 16
		QD_e^{50}	48 ± 16	54 ± 13	$\mathbf{3} \pm 1$	18 ± 15	57 ± 8	$\mathbf{64} \pm 5$	3 ± 1	$\mathbf{15} \pm 12$	$\mathbf{58} \pm 6$	63 ± 6	3 ± 0	17 ± 10
		QD_e^{75}	54 ± 13	60 ± 9	$\mathbf{3} \pm 1$	15 ± 12	61 ± 7	67 ± 5	3 ± 1	$\mathbf{15} \pm 10$	$\mathbf{62} \pm 7$	$\mathbf{68} \pm 6$	4 ± 2	16 ± 12
Occlusions	e_2	GD	10 ± 7	14 ± 9	$\mathbf{3} \pm 2$	14 ± 10	13 ± 7	18 ± 8	7 ± 1	$\mathbf{14} \pm 9$	$\mathbf{17} \pm 2$	$\mathbf{23} \pm 3$	7 ± 1	16 ± 9
		QD_e^{15}	26 ± 11	34 ± 15	$\mathbf{4} \pm 2$	$\mathbf{22} \pm 19$	38 ± 4	46 ± 6	5 ± 2	27 ± 25	$\mathbf{38} \pm 7$	$\mathbf{47} \pm 9$	5 ± 1	28 ± 21
		QD_e^{25}	35 ± 14	43 ± 13	6 ± 3	27 ± 22	43 ± 6	51 ± 6	5 ± 1	$\mathbf{23} \pm 17$	$\mathbf{44} \pm 6$	$\mathbf{52} \pm 6$	$\mathbf{5} \pm 2$	24 ± 17
		QD_e^{50}	43 ± 17	50 ± 14	5 ± 1	21 ± 17	52 ± 7	58 ± 6	$\mathbf{4} \pm 2$	$\mathbf{20} \pm 14$	$\mathbf{53} \pm 5$	$\mathbf{58} \pm 4$	4 ± 1	20 ± 14
		QD_e^{75}	47 ± 11	54 ± 8	$\mathbf{5} \pm 2$	22 ± 19	54 ± 7	60 ± 8	6 ± 3	$\mathbf{20} \pm 14$	$\mathbf{55} \pm 8$	$\mathbf{62} \pm 7$	5 ± 3	24 ± 18

Table 3.5: General and qualified detector performance (averaged over detectors, together with the standard deviations) tested in nighttime and with camera occlusions. Bold entries indicate the best performance on each metric across the three datasets.

cal during a deployment phase). The metrics' average performance obtained by DETR [55], YOLOv5 [54], and Faster R–CNN [21] are detailed in Table 3.5 and visually shown in Figure 3.16 (mAP) and Figure 3.17 ($TP_{\%}$, $FP_{\%}$, and $BFD_{\%}$).

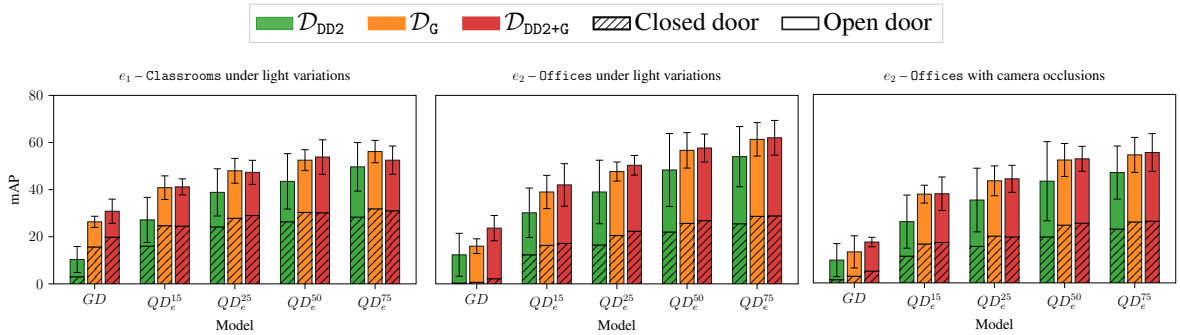


Figure 3.16: Real–world evaluation of the detectors under light variations conditions (left, middle) and with occlusions (right). GDs are based on different datasets and the mAP is averaged over the three models with the standard deviation.

Figure 3.16 shows how the mAP performance of the GDs are similar to those of Figure 3.13, indicating that the GDs are robust to illumination changes and camera occlusions. As reported in Section 3.5.1, the general detectors based on $\mathcal{D}_{\texttt{DD2}}$ exhibit the worst performance while those trained with our simulated dataset $\mathcal{D}_{\texttt{G}}$ perform remarkably better, especially in e_1 during nighttime (see also the $TP_{\%}$ in Figure 3.17). The $\mathcal{D}_{\texttt{DD2+G}}$–based GDs, despite improving the average mAP as shown in Figure 3.10, increase also the performance gap between the models (see the standard deviation of the orange and red bars in Figure 3.16). More interestingly, it

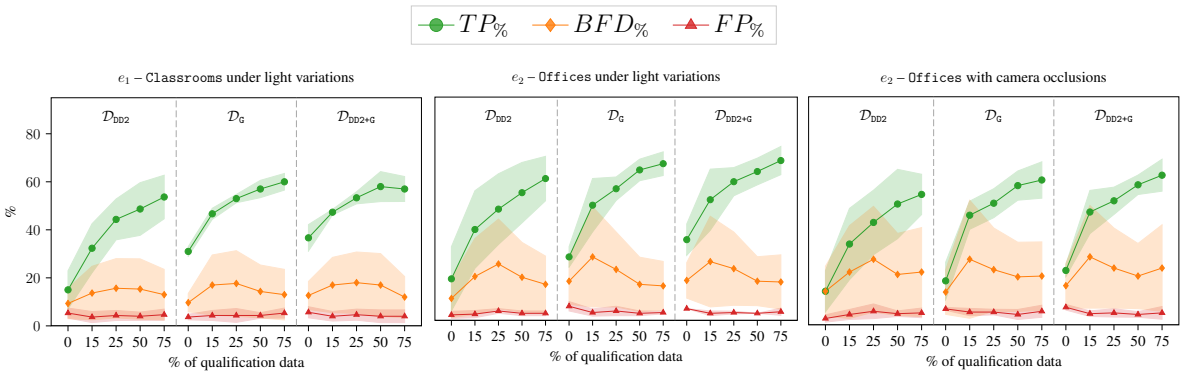


Figure 3.17: Operational performance indicators under light variations (left, middle) and with camera occlusions (right) averaged over the three models (with standard deviations) with GDs trained on different datasets.

can be seen in Table 3.5 how the improvement provided by the qualified detectors is maintained also in the (challenging) long–term deployment conditions of light variations and camera occlusions (Figure 3.18 reports some representative detections of QD_e^{15}). Despite this, the qualification procedure we propose enables QD_e^{15} to perform door detection also in (very) challenging situations where doors are almost entirely occluded (see the first and third examples in the last column of Figure 3.18). The performance decrease observable comparing Tables 3.5 and 3.4 is a direct consequence of the fine–tune, which produces QDs that slightly overfit the conditions (different light and no limited occlusions) seen during the robot's deployment. Despite this, our method ensures a performance improvement to the GDs when used in long–term scenarios with illumination changes and dynamic obstacles hiding doors' portions, enabling the QDs to still solve challenging examples, as shown in Figure 3.18. Once again, QD_e^{15}, albeit using a few examples for fine–tuning, ensures the best performance improvement also in challenging long–term deployment conditions.

As mentioned before, the qualified detector QD can detect doors from challenging points of view, thus improving its performance in a target environment; the same does not hold for GD. To further prove the qualification's benefits, we test the performance of the detectors of Section 3.5.2, qualified with the data from the robot's initial deployment, on a new run of the robot obtained a year later and containing challenging images of doors. More precisely, we performed an additional acquisition campaign targeted at capturing only door instances from difficult viewpoints, which the robot will encounter in long–term deployments: data are acquired when the robot is navigating through the main corridor of e_2 – `Offices` (see Figure 3.24 for the floorplan). In such a corridor, detecting doors is particularly challenging: there are multiple doors, often far away from the robot, and perpendicular to the robot's motion. Note that, in this data, the doors are perceived by the robot in the same environmental settings (daylight and no dynamic obstacles) as those encountered during the initial deployment (whose data are used to train the QD); still, the status of some doors is different (doors that were open/closed may be closed/open). In this way, we can observe if the qualification procedure overfits to the initial training data (e.g., if the model is biased to detect a door as open/closed because in the dataset used to train the QD such a door is open/closed). Examples of these changes can be seen in the

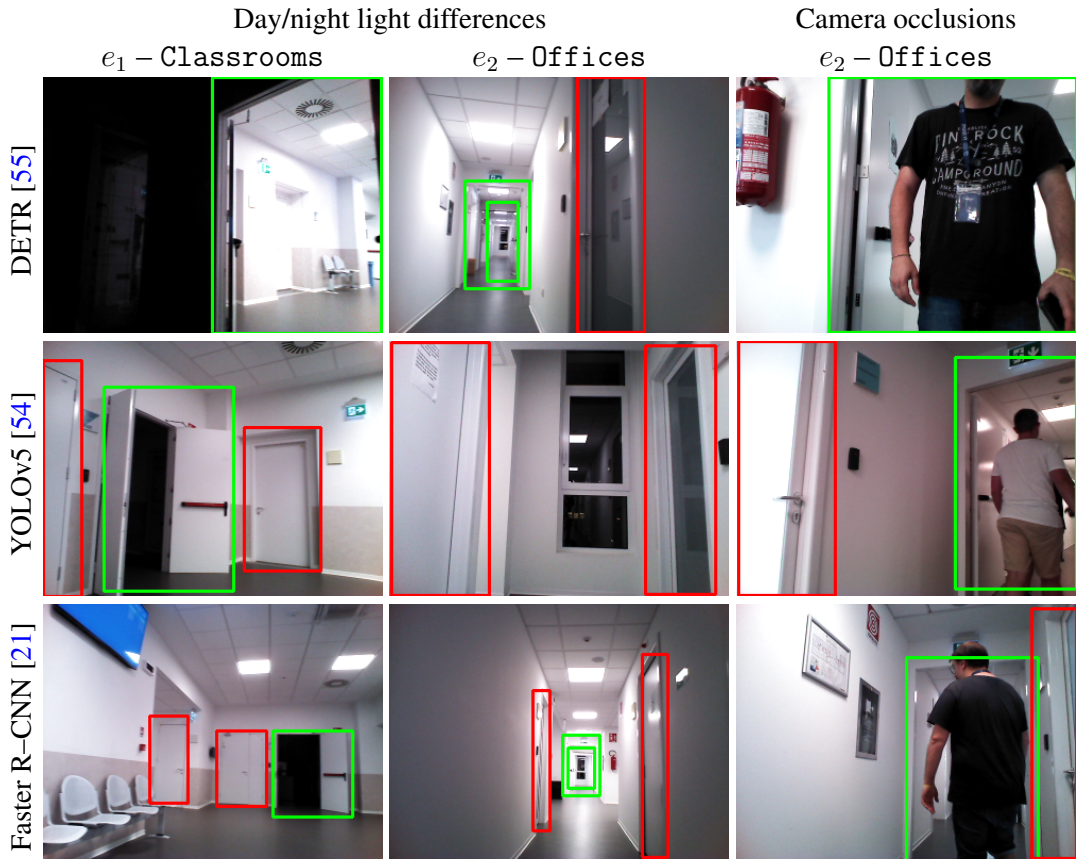


Figure 3.18: Challenging doors detected by QD_e^{15} (based on our dataset $\mathcal{D}_{\texttt{G}}$) under light changes (first and second column) and camera occlusions (third column).

first two columns of Figure 3.19.

The results of this experiment are in Table 3.6. It can be seen how the GDs work fairly well also on this challenging run, with performance close to that of the GD on the less challenging dataset of Table 3.4. Also for this experiment, the use of our dataset $\mathcal{D}_{\texttt{G}}$ to train the GD improves its performance (mAP and $TP_{\%}$), when compared with the GD trained on $\mathcal{D}_{\texttt{DD2}}$. Again, $\mathcal{D}_{\texttt{DD2+G}}$ increases the detection accuracy in terms of mAP and $TP_{\%}$ of the GDs also reducing the discrepancy between the tested models (low σ for all metrics). More interestingly, the qualified detectors, fine–tuned with the data acquired during the robot's initial deployment (those tested in Section 3.5.2), have remarkably higher performance when tested on new, challenging, examples (see the first part of Table 3.6). (Note that the performance decrease, when compared against Table 3.4, is because door images are taken from challenging points of view, while in the initial dataset many images of doors have a clear, frontal view of the target.) The fact that the performance of Table 3.5 is close to that of Table 3.6 shows how the qualification procedure is robust against overfitting on data used for the qualification procedure; the fact that doors are observed in a given status (open/closed) during the initial deployment does not cause a drop in performance when the same door is observed, later on, with a different status (closed/open). Once again, the QD_e^{15} ensures the best performance improvements when compared to the subsequent qualification rounds. This is corroborated by the challenging detections reported in

Figure 3.19 (third column), where YOLOv5, based on $\mathcal{D}_{\texttt{G}}$ and qualified with the 15% of the data collected during the robot deployment, successfully identifies challenging door instances when viewed from narrow side angles, at large distance from the camera, and with different statuses.

A robot deployed in the long term is constantly acquiring new data from its environment; some of them can be potentially used, after a labeling step, to perform further qualification runs on the QD. We have thus performed preliminary tests to evaluate the impact of this procedure. To do so, we compared a QD as trained in Section 3.4.3, using *deployment* data, with a QD that has been trained with data acquired during the initial deployment and with additional data acquired in different environmental condition (i.e., during nighttime). The former is indicated as *Deployment*, the latter as *Deployment + nighttime* in Table 3.6. Note that the Deployment and the Deployment + nighttime datasets have different sizes (the latter includes the former). The results reported in Table 3.6 show that using more data for qualification enables the QDs to better identify doors from challenging perspectives and, importantly, this happens even when mixing images with a feature shift (i.e., different light conditions). Even in this case, the QD trained on our dataset $\mathcal{D}_{\texttt{G}}$ have better performances than those $\mathcal{D}_{\texttt{DD2}}$ ($\mathcal{D}_{\texttt{DD2+G}}$) in terms of mAP, $TP_{\%}$, and $BFD_{\%}$. In particular, QD_e^{15} benefits more from using more data for the qualification, reaching performance close to the one reported in Table 3.4. Some improved detections can be seen in the last column of Figure 3.19, where YOLOv5 based on $\mathcal{D}_{\texttt{G}}$ and qualified with more data manages in detecting two very challenging closed doors (the second one with changed status) on the left (first row) and the right (second row) of the corridor.

		$\mathcal{D}_{\texttt{DD2}}$				$\mathcal{D}_{\texttt{G}}$				$\mathcal{D}_{\texttt{DD2+G}}$			
Exp.	**Qual.**	**mAP↑**	**TP$_{\%}$↑**	**FP$_{\%}$↓**	**BFD$_{\%}$↓**	**mAP↑**	**TP$_{\%}$↑**	**FP$_{\%}$↓**	**BFD$_{\%}$↓**	**mAP↑**	**TP$_{\%}$↑**	**FP$_{\%}$↓**	**BFD$_{\%}$↓**
GD	–	14 ± 11	15 ± 11	$\mathbf{1} \pm 1$	$\mathbf{14} \pm 10$	16 ± 7	19 ± 8	6 ± 1	14 ± 11	$\mathbf{20} \pm 5$	$\mathbf{24} \pm 5$	5 ± 1	20 ± 8
QD_e^{15}	Deployment	37 ± 16	44 ± 18	$\mathbf{3} \pm 2$	$\mathbf{26} \pm 24$	$\mathbf{48} \pm 10$	$\mathbf{56} \pm 10$	5 ± 1	31 ± 28	46 ± 10	55 ± 13	5 ± 1	33 ± 24
QD_e^{25}		45 ± 14	53 ± 12	$\mathbf{5} \pm 3$	33 ± 30	53 ± 8	60 ± 8	5 ± 2	$\mathbf{25} \pm 20$	$\mathbf{54} \pm 9$	$\mathbf{63} \pm 8$	6 ± 1	30 ± 22
QD_e^{50}		55 ± 17	62 ± 14	$\mathbf{4} \pm 2$	26 ± 25	$\mathbf{63} \pm 10$	$\mathbf{69} \pm 9$	4 ± 2	$\mathbf{21} \pm 18$	62 ± 9	69 ± 8	5 ± 1	26 ± 19
QD_e^{75}		59 ± 15	67 ± 13	$\mathbf{4} \pm 1$	25 ± 24	65 ± 11	71 ± 11	5 ± 1	$\mathbf{23} \pm 18$	$\mathbf{65} \pm 11$	$\mathbf{72} \pm 9$	6 ± 1	26 ± 19
QD_e^{15}	Deployment + nighttime	51 ± 16	59 ± 14	$\mathbf{3} \pm 1$	35 ± 35	$\mathbf{60} \pm 11$	$\mathbf{68} \pm 9$	4 ± 1	$\mathbf{29} \pm 21$	60 ± 11	67 ± 9	4 ± 1	30 ± 23
QD_e^{25}		54 ± 19	63 ± 17	$\mathbf{3} \pm 2$	37 ± 31	$\mathbf{62} \pm 10$	$\mathbf{70} \pm 7$	5 ± 1	$\mathbf{30} \pm 24$	61 ± 13	69 ± 10	5 ± 1	31 ± 22
QD_e^{50}		64 ± 13	71 ± 12	$\mathbf{4} \pm 3$	30 ± 24	$\mathbf{69} \pm 9$	$\mathbf{74} \pm 8$	6 ± 1	$\mathbf{24} \pm 16$	66 ± 13	73 ± 10	5 ± 1	25 ± 20
QD_e^{75}		65 ± 13	72 ± 10	$\mathbf{4} \pm 2$	32 ± 26	$\mathbf{70} \pm 9$	$\mathbf{76} \pm 8$	5 ± 1	$\mathbf{24} \pm 19$	68 ± 12	74 ± 9	5 ± 1	25 ± 20

Table 3.6: Performance of the QD's in e_2 – `Offices` focusing on challenging examples when the qualification is performed (top) using the data acquired during the first robot's deployment and (bottom) adding the images collected during nighttime. Results are averages and standard deviations computed over the three models. Bold values indicate the best performance on each metric across the three datasets.

3.5.4 Model Comparison

In this section, we analyze the three selected models (DETR [55], YOLOv5 [54], and Faster R–CNN [21]) highlighting their strengths and weaknesses to provide insights for helping tech-

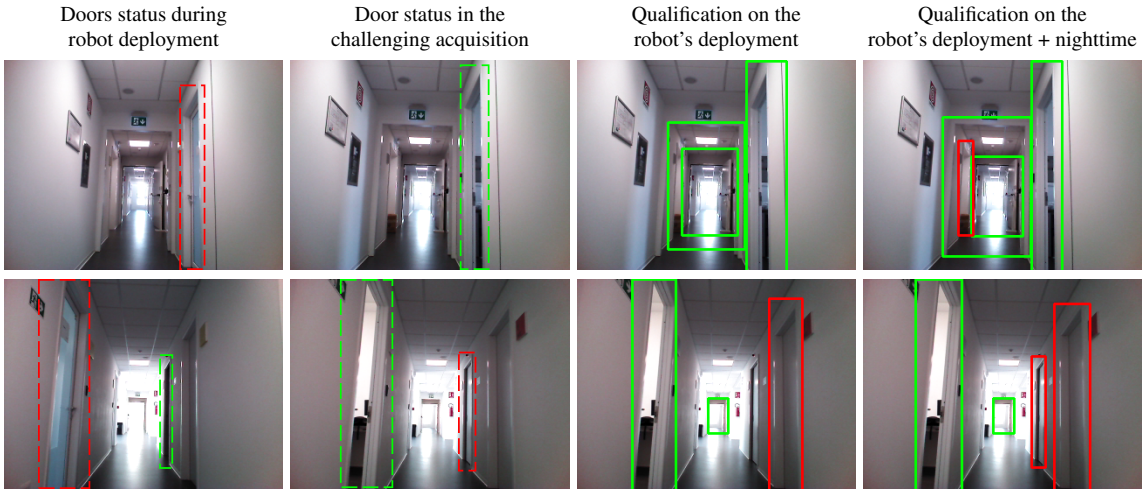


Figure 3.19: Door–status detections in the challenging run in the corridor of e_2 – `Offices` performed by the QD_e^{15} based on YOLOv5 and our dataset $\mathcal{D}_{\texttt{G}}$. The first and second columns highlight doors with a different status (in dashed green and red) between the robot deployment and the challenging run while the third and fourth columns report the detections when the qualification is performed using only the data from the robot deployment and adding the nighttime images.

nicians working in Robotic Vision scenarios to choose the best one according to their requirements.

From our experience in setting up the three models for the specific task of door detection, DETR turned out to be the easiest to adapt. Instead of learning how to activate a set of predefined anchor boxes according to the image features, DETR directly regresses the coordinates of the bounding boxes by construction. Moreover, it does not require a non–maximum suppression step to discard multiple detections of the same object. This is achieved by its loss function that matches the (limited) inferred bounding boxes to a single target. On the contrary, the detection performance of our detectors based on YOLOv5 and Faster R–CNN are strongly influenced by the hyperparameters setting: the anchor dimension and scale should be compliant with the object shape while the non–maximum suppression procedure can delete correct bounding boxes (such as those of nested doors). In other words, while the competitors need to encode task–specific prior knowledge in the model, DETR offers the possibility to share the same configuration among different applications (such as [12]).

After these considerations, we compare the performance of the detectors (based on our dataset $\mathcal{D}_{\texttt{G}}$) to study how they work, on average, in the four real environments of $\mathcal{D}_{\texttt{real}}$. Table 3.7 reports in detail the metrics results, depicted also in Figure 3.20 (mAP) and Figure 3.21 ($TP_{\%}$, $FP_{\%}$, and $BFD_{\%}$).

By observing the mAP performance shown in Figure 3.20 we can see that the best GD is based on DETR that, not requiring task–oriented knowledge, better addresses the sim–to–real gap (between our dataset $\mathcal{D}_{\texttt{G}}$ and the real acquisitions of $\mathcal{D}_{\texttt{real}}$). While YOLOv5 lies in the middle, Faster R–CNN reaches the worst performance when trained in simulation (with our dataset $\mathcal{D}_{\texttt{G}}$) and tested in the real world. As discussed in Section 3.5.1, Faster R–CNN, being a two–stage detector, tends to overfit the distribution of the training data acquired in simulation.

Exp.	DETR [55] mAP↑	DETR [55] TP%↑	DETR [55] FP%↓	DETR [55] BFD%↓	YOLOv5 [54] mAP↑	YOLOv5 [54] TP%↑	YOLOv5 [54] FP%↓	YOLOv5 [54] BFD%↓	Faster R–CNN [21] mAP↑	Faster R–CNN [21] TP%↑	Faster R–CNN [21] FP%↓	Faster R–CNN [21] BFD%↓
GD	**24** $\pm$ 8	**31** $\pm$ 9	7 $\pm$ 1	22 $\pm$ 9	16 $\pm$ 11	20 $\pm$ 10	**4** $\pm$ 4	**8** $\pm$ 3	12 $\pm$ 8	16 $\pm$ 9	4 $\pm$ 2	9 $\pm$ 2
QD_e^{15}	53 $\pm$ 15	63 $\pm$ 11	4 $\pm$ 3	35 $\pm$ 15	66 $\pm$ 5	67 $\pm$ 5	**2** $\pm$ 1	**3** $\pm$ 3	**74** $\pm$ 7	**78** $\pm$ 4	3 $\pm$ 2	22 $\pm$ 9
QD_e^{25}	59 $\pm$ 12	66 $\pm$ 9	4 $\pm$ 2	35 $\pm$ 16	75 $\pm$ 5	76 $\pm$ 5	**2** $\pm$ 1	**2** $\pm$ 1	**81** $\pm$ 4	**84** $\pm$ 3	3 $\pm$ 0	16 $\pm$ 2
QD_e^{50}	72 $\pm$ 11	78 $\pm$ 8	2 $\pm$ 1	21 $\pm$ 9	85 $\pm$ 5	86 $\pm$ 5	**0** $\pm$ 1	**2** $\pm$ 0	**89** $\pm$ 4	**91** $\pm$ 3	2 $\pm$ 1	15 $\pm$ 2
QD_e^{75}	80 $\pm$ 11	84 $\pm$ 8	2 $\pm$ 1	17 $\pm$ 7	88 $\pm$ 6	88 $\pm$ 6	**0** $\pm$ 0	**2** $\pm$ 1	**93** $\pm$ 4	**94** $\pm$ 3	1 $\pm$ 0	10 $\pm$ 5

Table 3.7: Real–world performance obtained by the three selected models (GDs are based on $\mathcal{D}_{\texttt{G}}$). We report the averages and standard deviations over the four real environments in $\mathcal{D}_{\texttt{real}}$. Bold entries indicate the best performance on each metric across the three models.

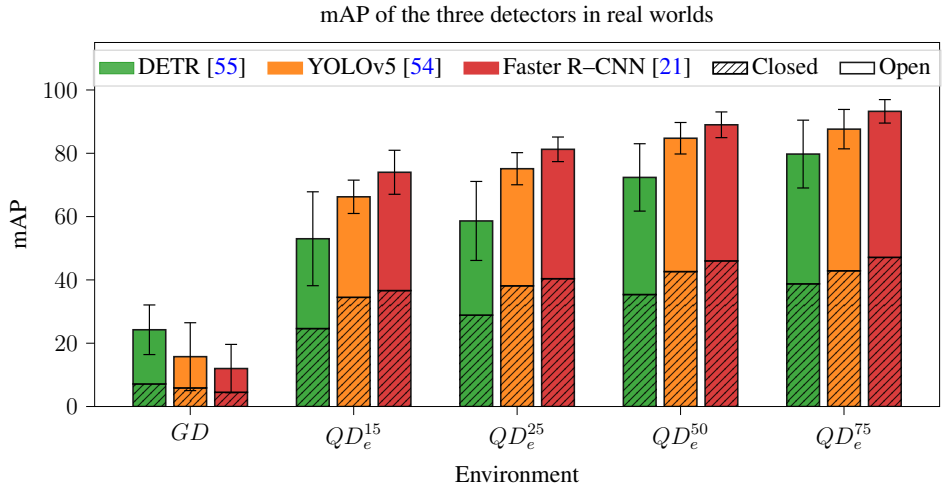


Figure 3.20: Real–world mAP (averaged over the four real environments, with standard deviations) with our three selected models (GDs are based on $\mathcal{D}_{\texttt{G}}$).

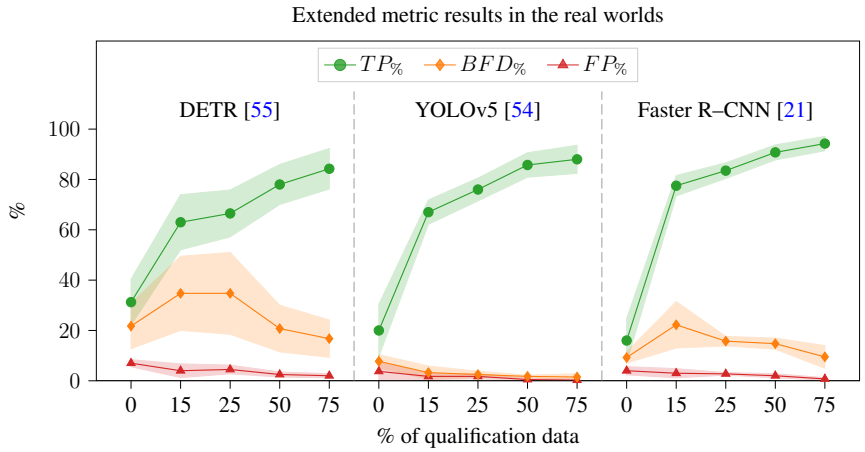


Figure 3.21: Real–world performance of the operational performance indicators with our three selected models (GD is based on $\mathcal{D}_{\texttt{G}}$). Results are averages and standard deviations across the four real environments of $\mathcal{D}_{\texttt{real}}$.

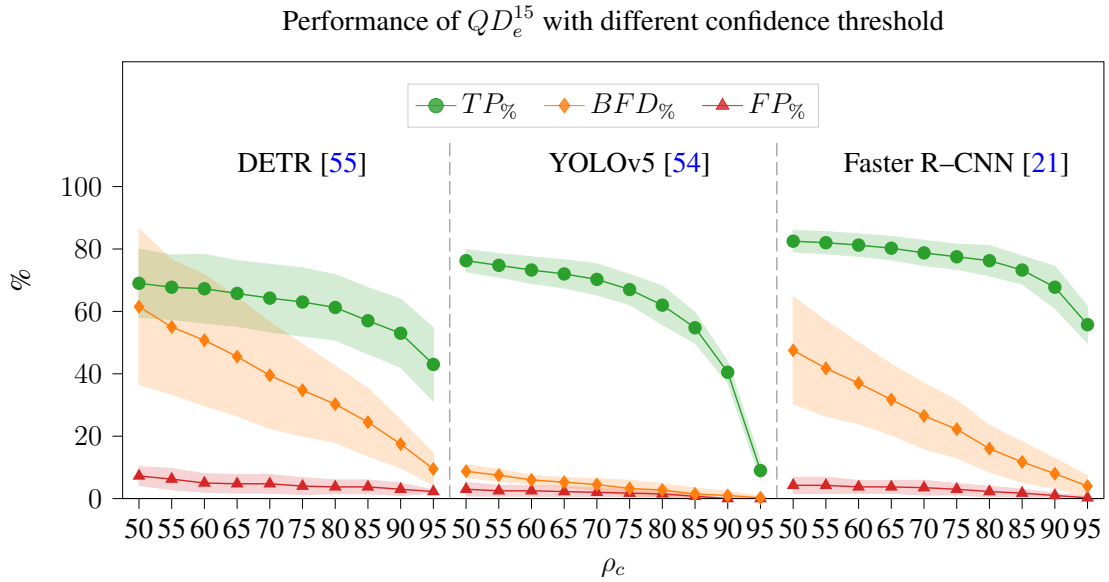


Figure 3.22: Operational performance indicators w.r.t. confidence for QD_e^{15} (averages and standard deviations over the real environments, GDs based on $\mathcal{D}_{\mathtt{G}}$).

This outcome is reverted by the qualification procedure. When fine–tuned for a target environment, Faster R–CNN reaches the best mAP results while DETR the worst (see the green and red bars in Figure 3.20). This is caused by the Transformer that, although popular, requires huge amounts of data (hundreds of millions) to effectively learn the architecturally inherent biases of the CNN–based models (such as the translation equivariance and the locality principle [181]). Moreover, by carefully examining the extended metric's results, we can see that the $BFD_{\%}$ of YOLOv5 is considerably lower than the other detectors (both the GD and its qualified versions). In a robotic domain where detections are translated into actions, this fact is extremely important because drastically reduces robot failures. Figure 3.22 shows how the additional indicators of QD_e^{15} vary according to the confidence threshold. The results demonstrate that our choice of $\rho_c = 75\%$ is a good compromise between the correct ($TP_{\%}$) and the wrong ($FP_{\%}$, $BFD_{\%}$) predictions.

Despite it is well–known from the literature that the two–stage detectors (like Faster R–CNN) are generally better than single–stage ones [148], YOLOv5 is more suitable for edge devices typically mounted in service robots. First, it is compatible with the NVIDIA Jetson TX2 mounted on our Giraff–X robotic platform [36, 7] (depicted in Figure 3.1) where it can run at 20 fps with the TensorRT framework. Since the other models are not compatible with the NVIDIA SDK for our specific hardware, we deploy all the architectures relying on ONNXRuntime, a less efficient inference framework able to run YOLOv5, DETR, and Faster R–CNN at 14, 6, and 0.7 fps, respectively. In our experimental setting, YOLOv5 represents the best compromise between performance and inference time, thus appearing as the most convenient model for door detection with service robots.

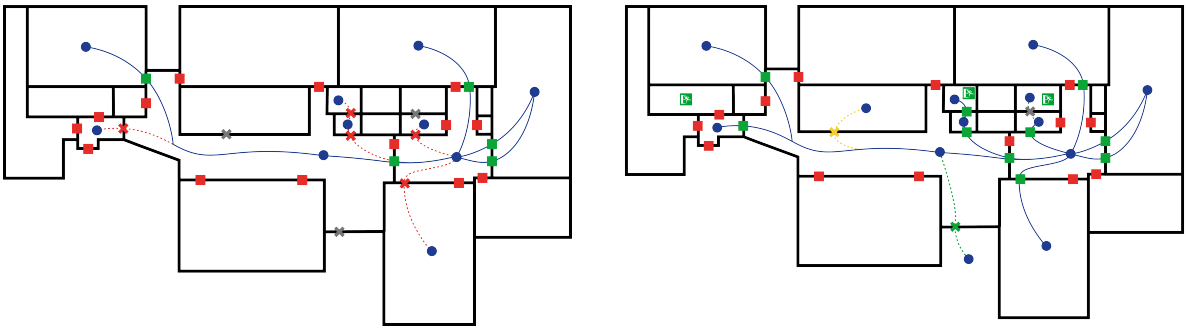
3.5.5 Evaluation on a Downstream Task: Topology Mapping

The goal of equipping an autonomous mobile robot with an object detection method is to allow the robot to have an updated representation of its working environment that can be used to plan and execute the tasks assigned to the robot. In the scenario we consider, the ability to detect doors can be used by the robot for the downstream task of reconstructing the current topology of the environment, that is, to infer which are the sub–areas that are currently accessible by the robot and those that are not. In the experiments presented in this section, we consider the environment's topology to be a graph, wherein the nodes represent rooms and the edges correspond to open paths connecting them. This knowledge can be used by the robot to plan its activities [61] considering the constraint that only a subset of the sub–areas are accessible at the current time.

We evaluate how the door detector can be used to obtain such a knowledge. As discussed in Section 3.3, we assume that the robot can rely on a 2D map acquired during its setup, but we consider a situation where the current topology of the environment has changed with respect to the one encoded in such a map, since some doors might be closed at the moment (for obvious reasons, when the 2D map is acquired all the doors are left opened). The task that the robot must face is to infer the current topology of the free space it can cover, assuming that door statuses do not change during the execution of this task. To carry it out, we consider the setting exemplified in Figure 3.1: the robot follows a trajectory spanning multiple rooms (the trajectory can be either functional to this topology–inference task or to another higher–level task the robot is performing). While doing so, it can observe, on purpose or incidentally, the status (`open` or `closed`) of multiple doors. While some doors are perceived from a frontal view, others will likely be observed only from a side angle as the robot moves in a different direction. We assume that the robot has full knowledge of all the locations of doors on the map $D = \{d_1, d_2, \dots d_n\}$. Furthermore, we assume to have a method that, given the image $x \in X$ where a door $\hat{y} \in \hat{Y}$ has been identified by a door detector, along with the pose from which the image was acquired, can determine the specific door instance $d \in D$ being observed at the moment. Note that multiple doors can be observed within the same image. The result of this step is that each $\hat{y} \in \hat{Y}$ that is not a BFD is associated to a door instance d. The robot thus counts, along the whole trajectory, how many times each door $d \in D$ has been identified either as `open` or `closed`, and infers its status as the one of the majority label. This information is used to infer the current topology, which, for evaluation, we compare with the one obtained by repeating the process using true detections instead of predictions.

We had the robot following two trajectories in a real–world experiment with the same setup described in Section 3.4.2. The first trajectory is performed in e_1 – `Classrooms` during nighttime, using the same run of Section 3.5.3. The second trajectory is performed in e_2 – `Offices` with daylight. We compare the performance in inferring the topology with GD against QD_e^{15} (both based on YOLOv5 and $\mathcal{D}_{\mathtt{G}}$). In both cases, the QD_e^{15} is trained with data collected at mapping time, with daylight. Note that the evaluation in e_1 is performed in challenging conditions because the qualified detector is tested under light variations (i.e., with nighttime data). The floor plans and the topologies of e_1 – `Classrooms` and e_2 – `Offices` inferred with

Exp.	■■↑	✖✖↓	✖↓	✖↓	$RA\uparrow$
GD	20	4	1	2	71%
QD_e^{15}	25	1	1	1	89%



(a) GD **(b)** QD_e^{15}

Figure 3.23: Topology of the environment for e_1 – `Classrooms` during night–time as identified using GD and QD_e^{15} to detect the status of each door.

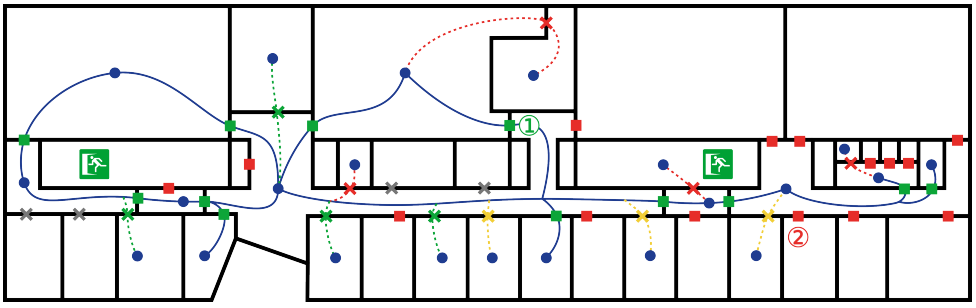
this framework are shown in Figure 3.23 and 3.24, respectively. We indicate with ■ (■) a door correctly recognized as `open` (`closed`) and with ✖ (✖) a door that has been wrongfully recognized as `closed` (`open`) when its current status is `open` (`closed`). We indicate with ✖ the event where the number of detections $\hat{y}$ where a door is labeled as `open` is equal to those where it is perceived as `closed`, and thus the robot is undecided. If a door has been observed in multiple images, but the door detector was always unable to detect any door due to false negatives, we label such door with a ✖. We indicate with ● the location of a room that the robot can access, and we highlight the location of the main entrance/exits of the environment. We indicated with a solid blue line a path across two different rooms that is open for the robot, and with a dashed line a path between two rooms that has been wrongfully estimated, following the same color schema as above: a red (green) path when a passage is estimated to be `closed` (`open`) when actually it is `open` (`closed`).

To understand the impact of having a qualified detector in estimating the topology of an environment, we report the topology obtained with GD and QD_e^{15} in Figure 3.23–3.24, as well as the number of doors whose status is correctly/wrongly detected, and the total *recognition accuracy* RA, that is the percentage of doors d whose status has been successfully detected during the robot run. These metrics are specific to the detection domain and are downstream OPI, following the definition of Section 3.4.4.

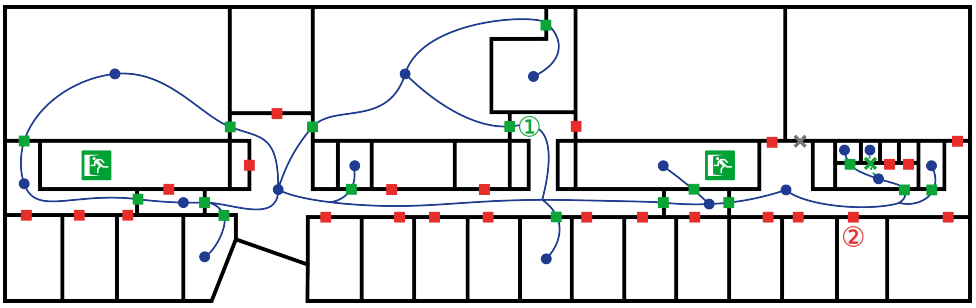
From Figures 3.23–3.24, we can see how the qualified detector can correctly identify the topological status of the environment, albeit making minor errors. In both environments, the QD identifies correctly the status of most doors, with an RA of around 90% ($89,29\%$ in e_1, $95,23\%$ in e_2).

We noticed how, while both GD and QD_e^{15} could detect successfully the status of a door when it is observed from a frontal position by the robot, the GD often fails when the robot is at one side of a door, when the door's view is partially occluded, or when there are challenging light conditions. In all those cases, QD_e^{15} does not suffer from the same limitations. An example

Exp.	■■↑	✖✖↓	✖↓	✖↓	$RA\uparrow$
GD	26	9	4	3	62%
QD_e^{15}	40	1	1	0	95%



(a) GD



(b) QD_e^{15}

Figure 3.24: Topology of the environment for e_2 – `Offices` during daytime as identified using GD and QD_e^{15} to detect the status of each door.

(a) **(b)** **(c)** **(d)**

Figure 3.25: Two examples where the QD identifies the door ① in tho challenging images (a–b), and the door ② in other two challenging images (c–d). In all four cases, the GD does not identify the two doors in these images.

of this can be seen in the doors connected to the main corridor of e_2, shown in Figure 3.24. While QD_e^{15} can identify how most doors are closed (✖), GD often fails to understand the status of those doors, thus identifying them as open (✖ or ✖) or failing to identify them (✖).

To better highlight this event, we provide detailed results about how many times two doors, that are highlighted with ① and ② in Figure 3.24, have been viewed by the robot in the two runs. Door ① was observed in 60 images by the robot. While QD_e^{15} was able to correctly detect the status of the door 52 times and was unable to detect the door on 8, the GD was able to detect the door only on 2 occasions and was unable to detect it 58 times. Figure 3.25(a–b) shows two of the 60 images, with the bounding box identified by QD_e^{15}. At the beginning of the run, the door

was closed, and was briefly perceived in that condition when the robot was outside the room; nevertheless, the robot was able to correctly label it, as in Figure 3.25a. Later, the robot enters the room and the door status is open, as in Figure 3.25b. In both cases, GD fails to identify any bounding box from those images. The door ② was observed in 40 images as closed; QD_e^{15} was able to correctly identify the status of the door 32 times, and was unable to detect the door 8 times. The GD is far less accurate in detecting doors in the same set of images: it correctly identified the door as `closed` 5 times, wrongly identified the door as `open` 4 times, and did not identify any door in 31 perceptions. Two examples of these images are shown in Figure 3.25(c–d); in Figure 3.25c the door ② is the second one on the right, while in Figure 3.25d it is the one on the left side of the corridor. In both images, QD_e^{15} was able to identify successfully the door status and location, while GD failed to identify the presence of a door. Similar examples can be made for all of the rooms that are connected to the central corridor of Figure 3.24, that are seen by the robot from a similar perspective.

These results show how the general detector manages to partially reconstruct the topology of the environment due to a high number of false positives and wrong detections. On the other hand, the qualified detector obtains more stable and robust performance compared to its general version when used in its deployment environment. This demonstrates how the qualification step described in Section 3.3 substantially improves (with a little cost) the performance of the detection method in a downstream task, providing more accurate domain–specific knowledge to the robot.

3.6 Lessons Learned and Future Works

With our extensive experimental campaign in the real world, we assess the effectiveness of our pipeline involving simulation and qualification for the development and deployment of deep learning–based door–status detectors for mobile robots. Here, we synthesize some key lessons that, while specific to our scenario, might apply to object detection with service robots and then we delineate the limitations and future directions.

Analogously to what happens in other robotic domains, simulation can be properly engineered to synthesize domain–relevant training data for object detection with service robots. In this specific scenario, domain relevance is not only influenced by photorealism. The alignment with the robot's perception model plays a major role that cannot be neglected in the development phase. Simulations offering an acceptable level of photorealism and, at the same time, allowing to generate robot–centric perceptions produce valuable training data. This simulated data is remarkably more cost–effective when compared with real–world acquisition with mobile robots.

Training on varied data in the attempt of generalizing across different environments will inevitably hit a performance ceiling. During its operational time, a mobile robot will encounter detection instances that remarkably shift away from the training distribution and that constitute hard cases peculiar of the specific environment in which the robot is deployed. For mobile robots, the priority is to be capable of dealing with such instances and not to generalize on each

possible environment. Qualification leverages this condition and allows to break this performance limit. Its impact is remarkable since difficult detection instances are typically connected with critical steps in a robot's task. Additionally, qualification shows a diminishing–return trend in performance where the first improvement steps outperform the subsequent ones and already lead to effective and robust detectors. As a consequence, training a qualified detector incurs in affordable data preparation costs.

In our robotic scenario, detections are meant to directly translate to decisions and actions. This is an aspect often, and rightfully, neglected in the broad field of object detection. Selecting the proper model to deploy and identify the most relevant performance metrics plays a crucial role in tailoring the robotic setup to the use case at hand.

A key limitation of our proposed pipeline is the need for human supervision to perform the qualification. Despite we demonstrate that the (low) effort in providing manual annotations is justified by the (high) performance improvement, in some context the robot should be able to adapt to a new environment avoiding human–in–the–loop. Despite fine–tuning the model using pseudo–labels is a promising solution, the noise experienced by the model in a new domain makes self–supervision impractical. To overcome this, Chapter 4 proposes two methods to improve the quality and the precision of the pseudo–labels leveraging the spatial consistency they must have in 3D. Another limitation is that performing inference with DNNs on–board of mobile robots is often unfeasible due to their limited hardware configuration in terms of computational power. To overcome this, a solution is to deploy the general model in the cloud, but this poses additional challenges in terms of scalable adaptation and privacy preservation. In Chapter 5, we propose efficient approaches based on lightweight neural networks to solve these issues using the robot's hardware.

Chapter 4

Removing Human Supervision in Domain Adaptation

As described in the previous Chapter, collecting and manually annotating additional data from the robot's operational environment is a common approach to address domain shift at deployment time [47]. Despite promising, this process is costly, time–consuming, and often impractical in real robotic deployments. The goal of this Chapter is to propose innovative solutions to enable the robot to automatically adapt its vision modules, without any kind of human supervision.

To address this challenge, Unsupervised Domain Adaptation enables the adaptation of a general model to a qualified version with improved performance in a new environment, without relying on ground–truth labels or humans in the loop [148, 29]. A naive approach for unsupervised adaptation is the so–called self–supervision: fine–tuning a DNN using its predictions (or pseudo–labels) [26]. Despite promising, this approach cannot be applied in novel environments as pseudo–labels are inherently noisy under domain shift, causing performance degradation if used for fine–tuning. Unlike in static computer vision benchmarks, robotics offers the unique opportunity of leveraging the relations between pseudo–labels and the 3D world in which the robot operates. More specifically, a robot is continuously immersed in its target environment, can acquire multiple observations of the same scene, and can exploit mapping and spatial consistency as natural proxies for model adaptation. In other words, the predictions of the model are not independent but related by spatio–temporal constraints that can be exploited to reduce the model's error and improve the fine–tune quality.

The literature suggests two distinct paradigms to enforce prediction consistency in the robotic domain:

- The **multi–view consistency** proposed in [38, 39] leverages a 3D environmental representation (like a voxel map) to aggregate pseudo–labels for semantic segmentation. The semantically annotated map is then used to obtain, through rendering, new and spatially consistent semantic masks for fine–tuning.

- The **state–consistency**, originally proposed in [113], offers an alternative and more ef-

ficient way to enforce spatial consistency for spatial perception tasks, such as obstacle mapping. Instead of using a dense 3D map, the state–consistency is an additional loss term optimized during the model's adaptation. Under the assumption that the environment is almost static in short time windows, the state–consistency loss forces two consecutive predictions close in time but produced from different viewpoints to be equal (or consistent) when reported in the same frame of reference using the robot odometry.

In this chapter, we propose two methods for unsupervised domain adaptation specific to robotic scenarios, leveraging the aforementioned paradigms.

- We extend the multi–view consistency [38] by adding an additional instance–aware refinement step. Our method enables semantic masks not only to be consistent at a fine–grained level (e.g., voxel–wise), but also coherent at the level of object instances of the scene. This contribution, reported in Section 4.1, enables the adaptation of a DNN for semantic segmentation to a novel environment, avoiding human supervision.

- We then propose a self–supervised approach leveraging the state–consistency. Our approach enables the sim–to–real transfer of a DNN for 3D object detection without relying on ground truth labels. Instead, the fine–tuning is supervised using only the state–consistency loss that only needs the estimated odometry of the robot to force prediction consistency between consecutive frames. This contribution is delineated in Section 4.2.

4.1 Instance–Guided Unsupervised Domain Adaptation

Despite being successfully used for adaptation in unsupervised settings, the multi–view consistency for semantic segmentation as defined in [38, 39] remains vulnerable to instance–level incoherence that might persist across frames. As shown in Figure 4.1, the multi–view consistent pseudo–labels (middle column) obtained from the 3D map are not coherent with object instances, meaning that a single object can be assigned to multiple classes. Examples are the `sofa` and the `fridge` in the second and third rows of Figure 4.1 that, in some areas, are also labeled as `chair` and `cabinet`, respectively. Furthermore, the rendered pseudo–labels contain bad visual artifacts (e.g., very imprecise contours) generated by the rendering process.

In this section, we tackle and mitigate these issues by proposing an instance–guided approach to extend the multi–view consistency, thus improving unsupervised adaptation for robotic semantic segmentation. Starting from multi–view consistent pseudo–labels, we integrate a refinement stage that leverages the zero–shot instance segmentation capabilities of a foundation model, queried via two different and complementary automated prompting strategies. Our method enforces instance–level coherence and mitigates rendering artifacts, yielding more accurate and stable annotations for self–supervised fine–tuning (see Figure 4.1, third column). Experiments on real–world data show that our method significantly outperforms existing base-

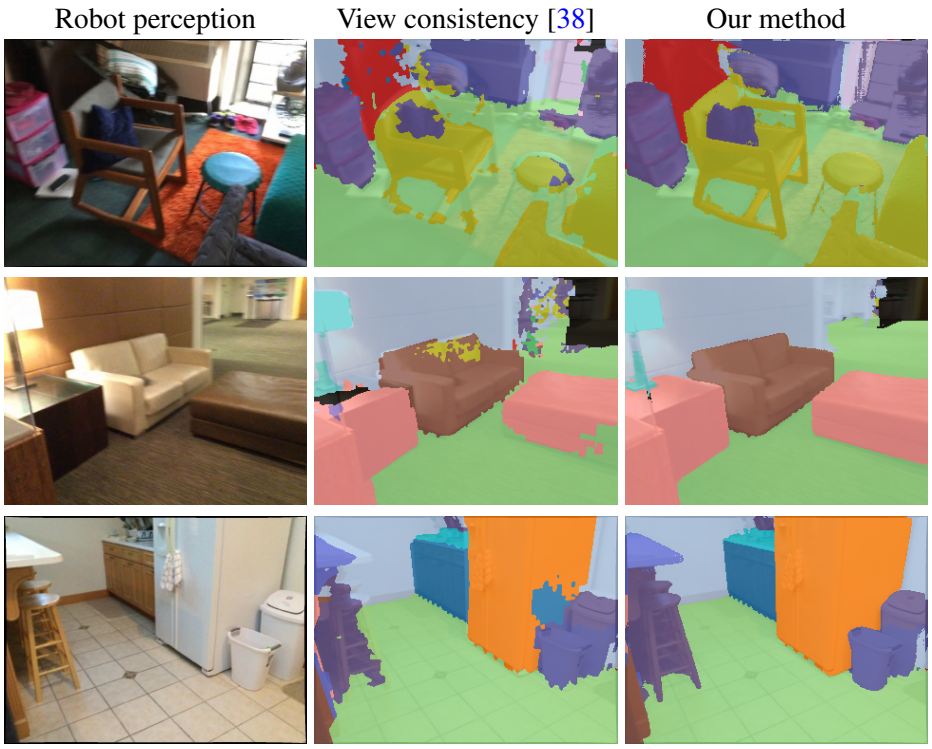


Figure 4.1: Examples of the improvement provided by our method (third column) to the view–consistent pseudo–labels [38] (middle column).

lines, while requiring no ground–truth labels in the target domain. In summary, the contributions we provide (reported in the work of [48]) are:

- we propose a novel framework for unsupervised domain adaptation in robotic semantic segmentation that combines multi–view consistency with instance–aware pseudo–label refinement (Sections 4.1.2 and 4.1.3);
- we integrate Segment Anything (SAM) [182], a foundation model for instance segmentation, into the refinement stage, devising two automatic and complementary prompting strategies (Section 4.1.3);
- we carry out an extensive experimental evaluation on real–world data assessing how our method improves adaptation performance over existing baselines both in terms of pseudo–label quality (Section 4.1.5) and segmentation performance (Section 4.1.6).

4.1.1 Problem Formulation: Adaptation for Semantic Segmentation

We consider a reference scenario in which a mobile robot relies on a neural network to perform semantic segmentation from images acquired with an on–board RGB camera, while depth information is obtained either from stereo or an RGB–D sensor. Formally, we describe the task with

a function as $Y^* = f_{\theta_s}^{seg}(I)$ that predicts from an RGB image I a semantic mask Y^*, where each pixel is assigned to an object category $o \in \mathcal{O}$. The network $f_{\theta_s}^{seg}$ is a general model as its parameters θ_s are pre–trained on a *source* dataset $\mathcal{S} = \{\mathbf{I}^s, \mathbf{Y}^s\}$. $\mathcal{S}$ is composed of a set of images ($\mathbf{I}^s$) and their corresponding ground–truth semantic annotations ($\mathbf{Y}^s$) coming from multiple indoor scenes. We aim to mitigate the domain shift, that is the performance degradation that occurs when a robot's perception system encounters visual conditions, layouts, or object appearances that differ from its training data, experienced by the robot when deployed in a novel and previously unseen *target* environment t, where no ground–truth semantic labels are available. From that environment, we suppose to have a sequence of N perceptions $\mathcal{T} = \{\mathbf{I}^t, \mathbf{D}^t, \mathbf{H}^t\}$ acquired by the robot from multiple viewpoints, where each colored image $I_n \in \mathbf{I}^t$ and its corresponding depth $D_n \in \mathbf{D}^t$ are localized using the camera extrinsics $H_n \in \mathbf{H}^t$ (with $n \in \{1, \dots, N\}$). In our scenario, we suppose that the perceptions in $\mathcal{T}$ are acquired by the robot in a first exploration round that can be useful, for example, for mapping the new environment. Then, the data $\mathcal{T}$ are used to adapt the general model $f_{\theta_s}^{seg}$ for the target environment t, under the assumption that no ground–truth are available. Our goal is to obtain a new qualified semantic network $f_{\theta_t}^{seg}$ with improved performance in t.

Figure 4.2 presents a general overview of our proposed approach, which performs a self–supervised fine–tuning of $f_{\theta_s}^{seg}$ on data from the target scene t using directly its predictions as pseudo–labels. Our method aims at improving the quality of these pseudo–labels to prevent the performance degradation typically caused by re–training with the raw model's outputs, which are particularly noisy due to domain shifts. Following [38], we start by aggregating the per–frame predictions of $f_{\theta_s}^{seg}$ in a 3D environmental representation, which is used for rendering multi–view consistent pseudo–labels (Section 4.1.2, Figure 4.2 (a)). Then, we add a novel instance refinement step to ensure instance–coherent semantic annotations and fix the artifacts produced by the rendering procedure (Section 4.1.3, Figure 4.2 (b)). Finally, the refined pseudo–labels are used to fine–tune $f_{\theta_s}^{seg}$, thus obtaining $f_{\theta_t}^{seg}$ with improved performance in the target environment t (Figure 4.2 (c)).

4.1.2 Multi–view Consistent Pseudo–Labels

To retrieve multi–view consistent pseudo–labels we follow the method described in [38]. From the sequence $\mathcal{T}$ we extract the set $\mathbf{Y}^* = \bigcup_{n=1}^{N} \{Y_n^* | Y_n^* = f_{\theta_s}^{seg}(I_n)\}$ containing the single–frame model's prediction for each image in $\mathbf{I}^t$. Then, we use Kimera Semantics [183] to aggregate $\mathcal{T}$ together with $\mathbf{Y}^*$ in a 3D semantically–annotated representation of the environment. Kimera Semantics aggregates depth images in a 3D volumetric map represented using a voxel–based truncated signed distance function (TSDF), calculated using voxblox [184]. Each voxel close to the TSDF surface stores a probability distribution among semantic classes. This distribution is updated every frame by a semantically–annotated point cloud obtained by combining a prediction Y_n^* with its depth information D_n. The 3D points are associated with a one–hot encoding of the semantic labels and used to update the stored probability of the touched voxels.

After integrating all the N measurements, we export a high–resolution mesh that we use to

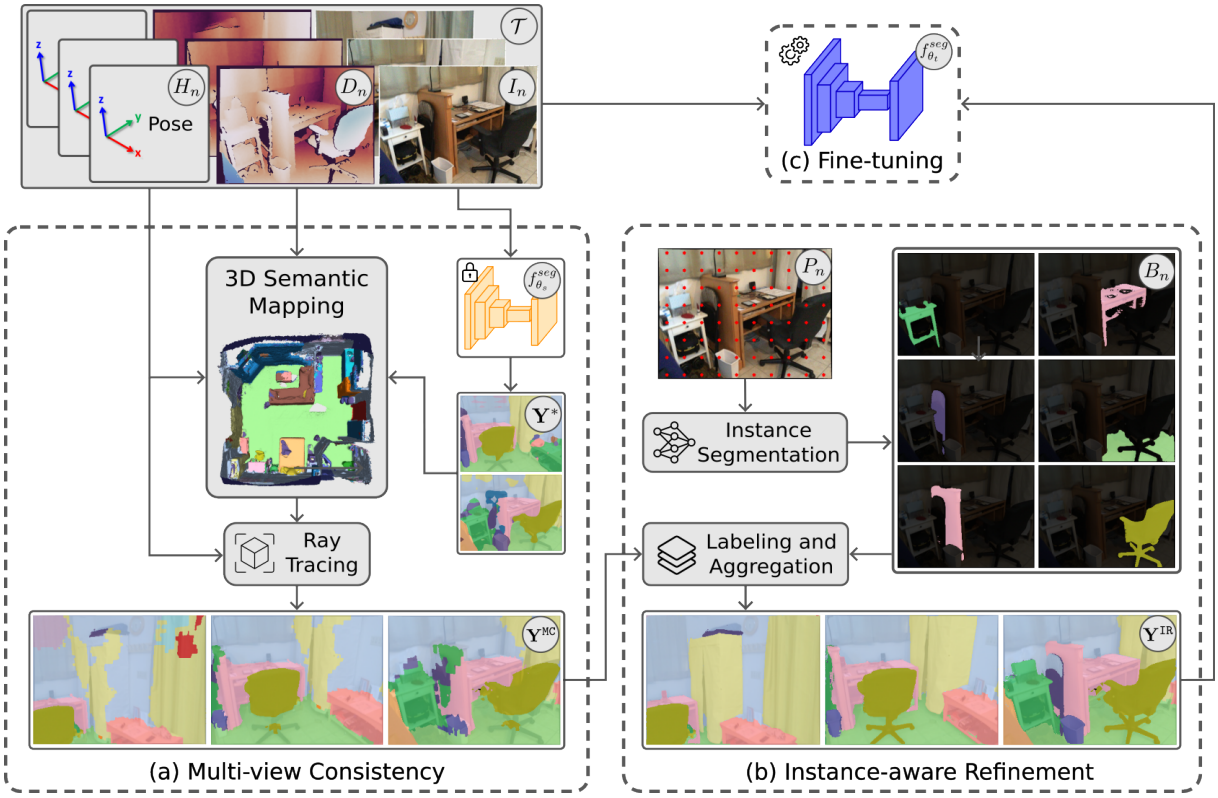


Figure 4.2: The general overview of our method for unsupervised domain adaptation. Its goal is to adapt a neural network for semantic segmentation $f_{\theta_s}^{seg}$ pre–trained on a source dataset $\mathcal{S}$ to a new and previously unseen target environment t. We achieve this in the challenging settings where no ground–truth labels are available but only a sequence $\mathcal{T}$ of N measurements acquired in t, that combines RGB images $\mathbf{I}^t$ and depth information $\mathbf{D}^t$ localized in the environment by camera extrinsic $\mathbf{H}^t$. At first, our proposed pipeline aggregates the model's predictions $\mathbf{Y}^*$ in a volumetric map of the environment, which is used to render multi–view consistent pseudo–labels $\mathbf{Y}^{\texttt{MC}}$ from each pose $\mathbf{H}^t$. Then, the resulting annotations are further refined leveraging a foundation model for instance segmentation that extracts object instances B_n from an image I_n, that are coherent with a sequence of prompts P_n. The object instances are filled with the object classes of $Y_n^{\texttt{MC}}$ and aggregated in a new annotation $Y_n^{\texttt{IR}}$. The final instance–coherent pseudo–labels $\mathbf{Y}^{\texttt{IR}}$ are used to fine–tune the source model, obtaining $f_{\theta_t}^{seg}$.

obtain new multi–view consistent pseudo–labels $\mathbf{Y}^{\texttt{MC}}$. To do this, we perform ray–casting from each camera pose H_n to associate each pixel with the corresponding voxel in the 3D world. Then, we fill the pixel class with the most probable object category stored in the corresponding voxel. The resulting annotations (for which some examples are reported in Figure 4.1 and Figure 4.2 (a)) aggregate information from different frames, thus filtering out the model's misclassifications. We stress the fact that Kimera Semantics [183] represents only a building block of our framework, which we leverage to implement the multi–view consistency. Our main contribution, which is delineated in the following section, abstracts from the mapping framework (we can use, for example, nvblox [185]) as it is an extension of the multi–view consistency proposed in [38].

4.1.3 Instance–aware Refinement

Despite the rendered pseudo–labels $\mathbf{Y}^{\texttt{MC}}$ show improved quality with respect to single–frame predictions (as they are persistent in consecutive viewpoints), they have some limitations that motivate our additional refinement step. This is a result of how robots perceive the environment while performing navigation. Robots often perceive objects only partially during navigation, leading to systematic errors in label aggregation. Take as a example a robot observing a cabinet from multiple views. When first observing it, the robot may only capture small corners or partially occluded views, which are difficult to recognize and thus are frequently misclassified as other categories, such as `wall`, as happens in the examples of Figure 4.2 (a). Because these partial views are encountered repeatedly, their incorrect labels end up dominating in the voxel–wise majority of the resulting 3D map. Later, once the cabinet is fully visible, the robot correctly classifies it, but those few correct predictions remain outnumbered. As a result, the final aggregated label is wrong, despite the robot having seen the cabinet clearly. Our instance–aware refinement step resolves this issue by leveraging SAM to group pixels into coherent object instances. Within each instance, the correct labels from full observations can outweigh the scattered errors from partial ones, allowing our method to propagate the right semantic category to the entire object, as shown in Figure 4.2 (b). Additionally, the multi–view consistent pseudo–labels are affected by artifacts introduced by the 3D reconstruction (such as missing depth in reflective surfaces) and by the rendering process (e.g., inaccurate boundaries caused by the voxel discretization).

To address these limitations, we propose to further refine the pseudo–labels of $\mathbf{Y}^{\texttt{MC}}$ according to the object instances contained in their corresponding RGB images in $\mathbf{I}^t$. (See Figure 4.3 for a detailed overview of this process.) To do this, our method leverages the zero–shot capabilities of Segment Anything (SAM) [178, 182], a prompt–based foundation model for instance segmentation. From an RGB image I_n and a list of J prompts P_n (in which each prompt $p_{n,j} \in P_n$ is expressed with a point and/or a bounding box), we use SAM to obtain a set of binary masks

$$B_n = \bigcup_{j=1}^{J} \left\{ b_{n,j} | b_{n,j} = f_\theta^{sam}(I_n, p_{n,j}) \right\},$$

where f_θ^{sam} denotes the SAM model with pre–trained parameters θ. Inside each $b_{n,j}$, a pixel u assumes the value $b_{n,j}(u) = 1$ if it lies inside the instance identified by SAM from the relative prompt $p_{n,j}$, otherwise $b_{n,j}(u) = 0$.

For prompting SAM, we propose two alternative strategies, which we later compare. The first one, named `grid`, uses a fixed prompt, which is created once and used for every frame. In this strategy, P_n is defined as a list of points uniformly arranged in a grid structure on the image plane, where d specifies the distance between adjacent points in the horizontal and vertical axes. In the second one, called `informed`, the prompt is procedurally extracted for each frame using its rendered pseudo–label. At first, a pseudo–label $Y_n^{\texttt{MC}}$ is partitioned into clusters, where each cluster is a connected region of pixels sharing the same semantic class. Then, for each cluster larger than a percentage a of the image area, it calculates the bounding box and its centroid. P_n

is finally composed by aggregating the bounding box/centroid pairs derived from all clusters.

Independently from the prompting strategy, our method proceeds by combining the object instances of B_n with the classes contained in the corresponding pseudo–label $Y_n^{\texttt{MC}}$. To do this, for each instance mask $b_{n,j} \in B_n$, we extract the most frequent object category according $Y_n^{\texttt{MC}}$, formally defined as

$$o_{n,j}^* = \arg\max_{o \in \mathcal{O}} \sum_{u \in \Omega} b_{n,j}(u) \mathbb{1}_{\{Y_n^{\texttt{MC}}(u)=o\}},$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function and Ω are the pixels indices. All the binary masks $b_{n,j} \in B_n$ are then merged, together with their relative object category $o_{n,j}^*$, in a single instance–aware pseudo–label $Y_n^{\texttt{IR}}$. At the beginning, we set $Y_n^{\texttt{IR}} = Y_n^{\texttt{MC}}$. Then, for each mask $b_{n,j} \in B_n$, we update $Y_n^{\texttt{IR}}$ by overriding the pixels's category inside the region defined by $b_{n,j}$ with $o_{n,j}^*$, formally

$$Y_n^{\texttt{IR}}(u) = \begin{cases} o_{n,j}^* & \text{if } b_{n,j}(u) = 1 \\ Y_n^{\texttt{IR}}(u) & \text{if } b_{n,j}(u) = 0 \end{cases} \quad \forall u \in \Omega.$$

Note that this step is sequentially applied for all $j \in \{1, \ldots, J\}$, meaning that previously added semantic labels can, in principle, be replaced in later stages. When B_n is derived using the `informed` prompting strategy, this replacement is not possible: the object instances are disjoint by construction since they are extracted from non–overlapping clusters of labels. In contrast, when the `grid` strategy is used, the order in which the instances $b_{n,j}$ are aggregated might impact on the quality of the pseudo–labels. This is because B_n contains a high number of overlapping instances, meaning that small objects may be completely overwritten by larger ones aggregated later. To avoid this, we sort the B_n in descending order of instance dimension, thus preserving the fine–grained details identified by SAM when prompted with the `grid` strategy. Practical examples of this process can be observed in Figure 4.3. Let's consider the chair on the left side of the desk, highlighted by the first red box in $Y_n^{\texttt{IR}}$. By inverting the aggregation order, the chair (correctly identified in the instance number `3`) would be overwritten by instance `1`. Although aggregating in descending order is generally the best choice for maintaining fine–grained instances, it becomes problematic when SAM oversegments a single object into multiple instances. Consider, for example, the wheeled legs of the office chair in Figure 4.3, highlighted by the red box in the right. While the chair is correctly segmented in instance number `2`, its legs will be labeled as `floor` in the final pseudo–label because SAM incorrectly identifies them as a separate object instance (the number `4` in the sequence).

The entire instance–oriented refinement procedure is performed for all the images/pseudo–labels pairs in $\mathbf{I}^t$ and $\mathbf{Y}^{\texttt{MC}}$. The resulting instance–aware annotations (for all the measurements in $\mathcal{T}$) are contained in the set $\mathbf{Y}^{\texttt{IR}}$. Finally, we use refined pseudo–labels $\mathbf{Y}^{\texttt{IR}}$ to finetune the model, thus doing unsupervised adaptation, as shown in Figure 4.2 (c).

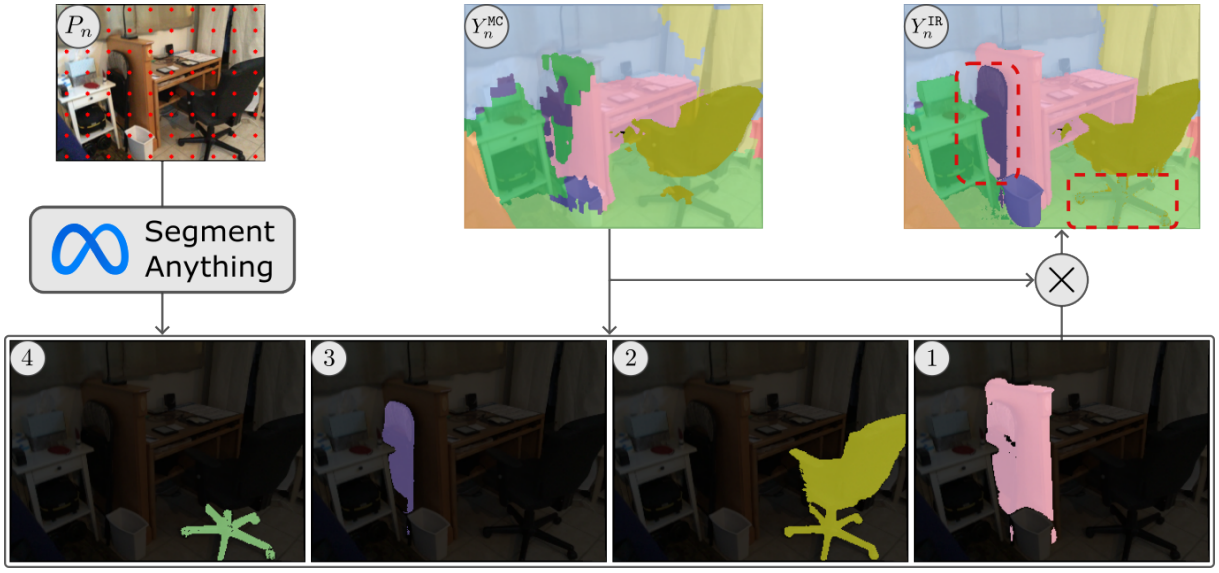


Figure 4.3: A more detailed overview of our instance refinement step. Given a robot perception I_n and a sequence of prompts P_n (a grid of points in this case), Segment Anything generates a sequence of object instances (for visualization purposes, we report only a subset of them). The instances are then filled with the most frequent object category in the multi–view consistent pseudo–label $Y_n^{\texttt{MC}}$. Finally, the instances are aggregated in descending order of their dimension, measured by the number of pixels, as indicated by the progressive labels from 1 to 4. The red bounding boxes show how the aggregation order influences the final instance–aware pseudo–label $Y_n^{\texttt{IR}}$.

4.1.4 Experimental Setting

Dataset For the evaluation of our proposed method, we rely on ScanNet [25], a real–world dataset containing 1513 sequences acquired in 707 different indoor environments. Each sequence includes RGB–D image pairs with their relative camera poses, estimated using BundleFusion [186]. The dataset also provides the per–frame ground–truths, which are manually annotated with the NYU40 object categories [187]. In all the experiments, we rescale the images to a resolution of 320×240 pixels, to be compliant with the hardware setup of a low–powered mobile robot.

Network pre–training Similarly to [38, 39], the scenes from 11 to 707 represent our source domain and are used to initialize the weights of the neural network for semantic segmentation $f_{\theta_s}^{seg}$. For simplicity, we rely on the DeepLabV3 [23] (with a ResNet–101 [176] as backbone) provided by [39], which was pre–trained on a dataset $\mathcal{S}$ counting $\approx$ 25k images/ground–truths pairs from environments 11–707, sampling one frame every 100. $\mathcal{S}$ was randomly split in $\approx$ 20k and $\approx$ 5k samples for training and validation, respectively. The network is optimized using Adam for 150 epochs, with a batch size of 4. To measure the quality of the predictions, we use the Intersection over Union averaged over the objects categories (mIoU).

Network adaptation Scenes 1–10, that represent the target environments, are used to obtain $f^{seg}_{\theta_t}$ by fine–tuning $f^{seg}_{\theta_s}$ with the pseudo–labels provided by our method. We consider the first sequence for each environment, from which the first 80% of the frames are used to produce the pseudo–labels and fine–tuning $f^{seg}_{\theta_s}$, while the last 20% is used for testing. This is the same method adopted in [38] and [39], which we replicate but also extend to achieve a more representative evaluation. Although training and testing frames are temporally disjoint, they are sampled from the same sequence within a given environment. In contrast, real–world deployments typically involve two distinct runs: one to collect data for unsupervised adaptation and another to evaluate performance in a different trajectory. To better reflect this setting, we perform an additional evaluation in which two separate sequences are used for each environment: one for pseudo–label generation and fine–tuning, and the other for testing. (We cannot run this protocol on environments 5, 8, and 9, as only a single sequence is available in the dataset.) Overall, this experimental design aims at better reflecting the practical conditions of robotic operation: a model $f^{seg}_{\theta_s}$ is first pre–trained on a large source dataset, the robot then collects an initial set of observations (e.g., while mapping the environment) to generate pseudo–labels and adapt $f^{seg}_{\theta_s}$ in an unsupervised fashion. Finally, the adapted model $f^{seg}_{\theta_t}$ is tested on new trajectories.

Instance–aware refinement To generate the multi–view consistent pseudo–labels $\mathbf{Y}^{\texttt{MC}}$, we leveraged the open–source implementation of [38]. We set the voxel size of Kimera Semantics [183] to 3cm and 5cm, to evaluate the performance under different resolutions, following the works of [38, 39]. More precisely, setting the voxel size to 3cm (5cm) creates a more (less) detailed 3D representation, requiring higher (lower) processing power for real–time robot deployment. The object instances B_n are obtained using the Ultralytics' implementation of Segment Anything v2 [182]. For prompting SAM, the points of the `grid` strategy are generated with a distance $d = 32$, while the cluster of labels smaller than $a = 0.1\%$ of the image size are not considered in the `informed` approach. To evaluate the performance improvements related to the two prompting strategies, we compare three versions of $f^{seg}_{\theta_t}$ for each environment: (i) using only `grid`, (ii) using only `informed`, and (iii) combining the two. For the single–method settings, $f^{seg}_{\theta_s}$ is optimized using $\mathbf{Y}^{\texttt{IR}}$ and their corresponding RGB images for 10 epochs using Adam with a learning rate of 10^{-5}, and batch size of 4. When the methods are combined, the model processes the same RGB image twice inside the same epoch, each time with different pseudo–labels. To maintain a comparable training budget, the total number of epochs is halved (from 10 to 5). Random flipping of orientation and color jitter are used as data augmentation.

Baseline. The baseline we use to evaluate our approach is the method of [38], which performs adaptation only via multi–view consistency (as detailed in Section 4.1.2 this is also an important building block of our pipeline). For a fair comparison, we replicate the results by re–generating the pseudo–labels from scratch using the aforementioned pre–trained DeepLabV3. With the multi–view consistent annotations $\mathbf{Y}^{\texttt{MC}}$ and their relative RGB images, we fine–tune a model $f^{seg}_{\theta_t}$ for each target environment, using the Adam optimizer with a learning rate of 10^{-5} and batch size of 4 for 10 epochs.

Env	$\mathbf{Y}^*$	Baseline (5cm) $\mathbf{Y}^{\mathtt{MC}}$	$\Delta_{\mathbf{Y}^*}$	Our method $\mathbf{Y}^{\mathtt{IR}}_{\mathtt{G}}$	$\Delta_{\mathbf{Y}^{\mathtt{MC}}}$	$\mathbf{Y}^{\mathtt{IR}}_{\mathtt{I}}$	$\Delta_{\mathbf{Y}^{\mathtt{MC}}}$
1	41,2	48,1	16,7%	**52,3**	8,7%	<u>49,1</u>	2,1%
2	**34,8**	28,8	-17,2%	<u>32,5</u>	12,8%	31,5	9,4%
3	23,7	<u>26</u>	9,7%	25,4	-2,3%	**26,5**	1,9%
4	62,8	63,9	1,8%	**65,2**	2,0%	<u>64,7</u>	1,3%
5	**49,8**	42,6	-14,5%	<u>45,5</u>	6,8%	44,9	5,4%
6	48,7	49,3	1,2%	<u>52,6</u>	6,7%	**53,3**	8,1%
7	40,3	48,4	20,1%	<u>48,8</u>	0,8%	**50,8**	5,0%
8	31,4	34,8	10,8%	<u>35,4</u>	1,7%	**36,4**	4,6%
9	31,8	**32,8**	3,1%	30,8	-6,1%	<u>32,1</u>	-2,1%
10	52,1	55,8	7,1%	**59,6**	6,8%	<u>58,8</u>	5,4%
Avg	41,7	43,1	3,9%	<u>44,8</u>	3,8%	**44,8**	4,1%

Table 4.1: mIoU of the instance–aware pseudo–labels with our method prompted with `grid` ($\mathbf{Y}^{\mathtt{IR}}_{\mathtt{G}}$) and `informed` ($\mathbf{Y}^{\mathtt{IR}}_{\mathtt{I}}$) against the baseline ($\mathbf{Y}^{\mathtt{MC}}$) and the predictions of the pre–trained model ($\mathbf{Y}^*$). Best and second best performances are respectively in **bold** and <u>underlined</u>.

4.1.5 Evaluation of the Pseudo–Labels Improvement

This section evaluates the quality of the instance–aware pseudo–labels produced by our method compared to those of the baseline. The quantitative results are reported in Table 4.1, which shows the precision of the pseudo–labels (measured in mIoU) obtained from the first 80% of the first sequence for each environment, using a voxel size of 5cm. We refer to the refined pseudo–labels as $\mathbf{Y}^{\mathtt{IR}}_{\mathtt{G}}$ or $\mathbf{Y}^{\mathtt{IR}}_{\mathtt{I}}$ if obtained using the `grid` or the `informed` prompting strategy.

The metrics show that our instance–refinement step improves the multi–view consistent annotations $\mathbf{Y}^{\mathtt{MC}}$ of the baseline, on average by 3.8% and 4.1% for `grid` and `informed`, respectively. Despite the baseline improves the raw model's predictions $\mathbf{Y}^*$ by $3,9\%$ on average, its impact is not stable and changes according to the environment. In some cases, the multi–view consistency yields substantial improvements (such as in environments 1 and 7 with an increment of $16,7\%$ and 20.1%), while in others it strongly degrades the quality of the per–frame predictions (e.g., in environments 2 and 5 with a decrease of $-17,2\%$ and $-14,5\%$). In contrast, our instance–aware refinement step increases the quality of $\mathbf{Y}^{\mathtt{MC}}$ in all environments, with only marginal decreases observed in scenes 3 and 9. As shown in Figure 4.4, our approach substantially improves the precision of the pseudo–labels by propagating the object classes according to instances identified in the RGB perception, and fixes the rendering artifacts produced by the ray tracing.

Another interesting outcome from the results of Table 4.1 is that, although the two prompting strategies achieve the same average precision, their performances vary across environments. This can be explained by their markedly different behaviors, which are, to some extent, complementary.

The `grid` approach exploits all the foundational model's strengths in instance segmenta-

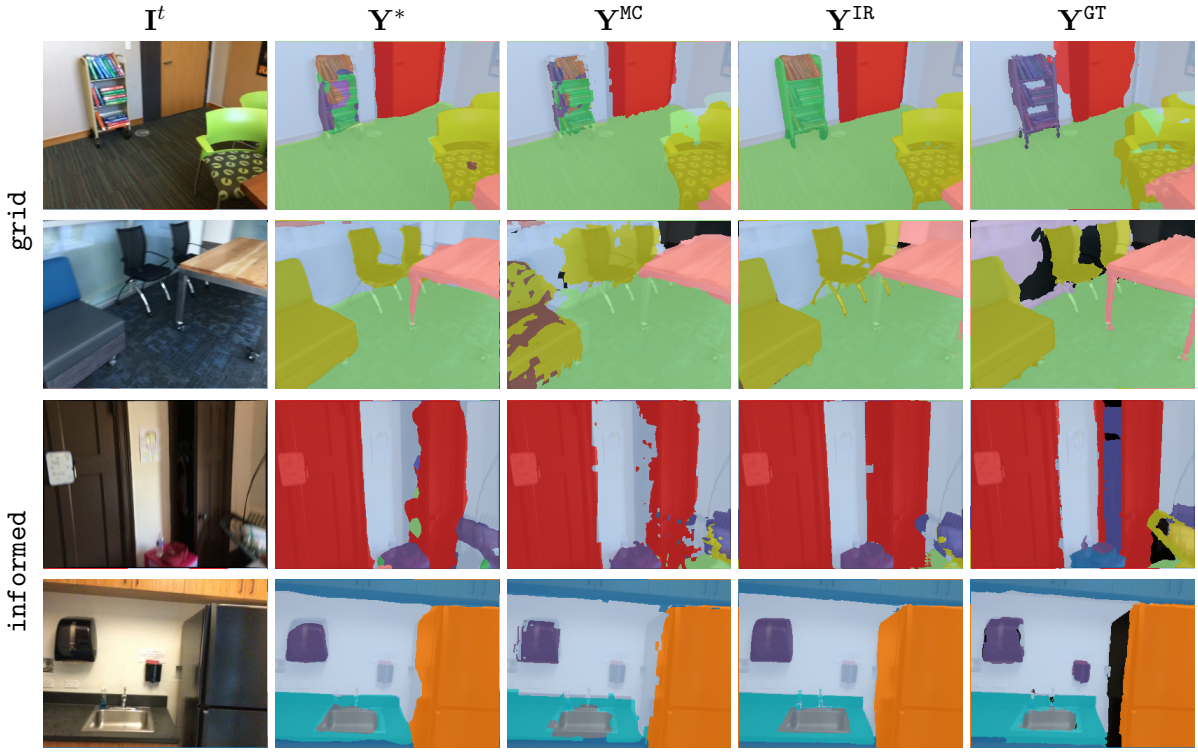


Figure 4.4: Instance–aware pseudo–labels ($\mathbf{Y}^{\mathtt{IR}}$) generated with `grid` (top) and `informed` (bottom), compared to multi–view consistent ($\mathbf{Y}^{\mathtt{MC}}$) and raw ($\mathbf{Y}^*$) ones. $\mathbf{I}^t$ and $\mathbf{Y}^{\mathtt{GT}}$ denote the RGB images and ground truths.

tion by using a fixed grid of points for each frame. This setting makes this method particularly effective in challenging situations where the objects have a complex shape or the multi–view consistent pseudo–labels are particularly noisy. Examples are reported in Figure 4.5, respectively in the first and second columns, where we can see that the `grid` approach precisely identifies the instances of a guitar and a chair, which is partially occluded by a pillow. Compared to the automatic approach, the `informed` prompts derived from $\mathbf{Y}^{\mathtt{MC}}$ produce less precise object instances. In the first case, the bounding box cuts the guitar's neck, while in the second one, the couch and the chair are grouped in the same prompt (because they form a single cluster), but only the couch is segmented.

In contrast, `informed` is a more conservative approach because the prompts are extracted from the multi–view consistent annotations. This is particularly useful to prevent the over–segmentation, a well–known limitation of SAM that often splits a single object into multiple instances, especially when the prompt is particularly dense (as in the case of `grid`). See, for example, the third column of Figure 4.5, where `informed` identifies the fridge as a unique object, while `grid` considers its left side as a separate instance, which is then wrongly labeled. Another important effect of `informed` is to guide the predictions of SAM when objects have similar textures, a situation in which an automatic segmentation is particularly challenging. This is illustrated in the last column of Figure 4.5, where `informed` identifies improved and more precise instances than `grid`, which partially mixes the table and the couches with the floor.

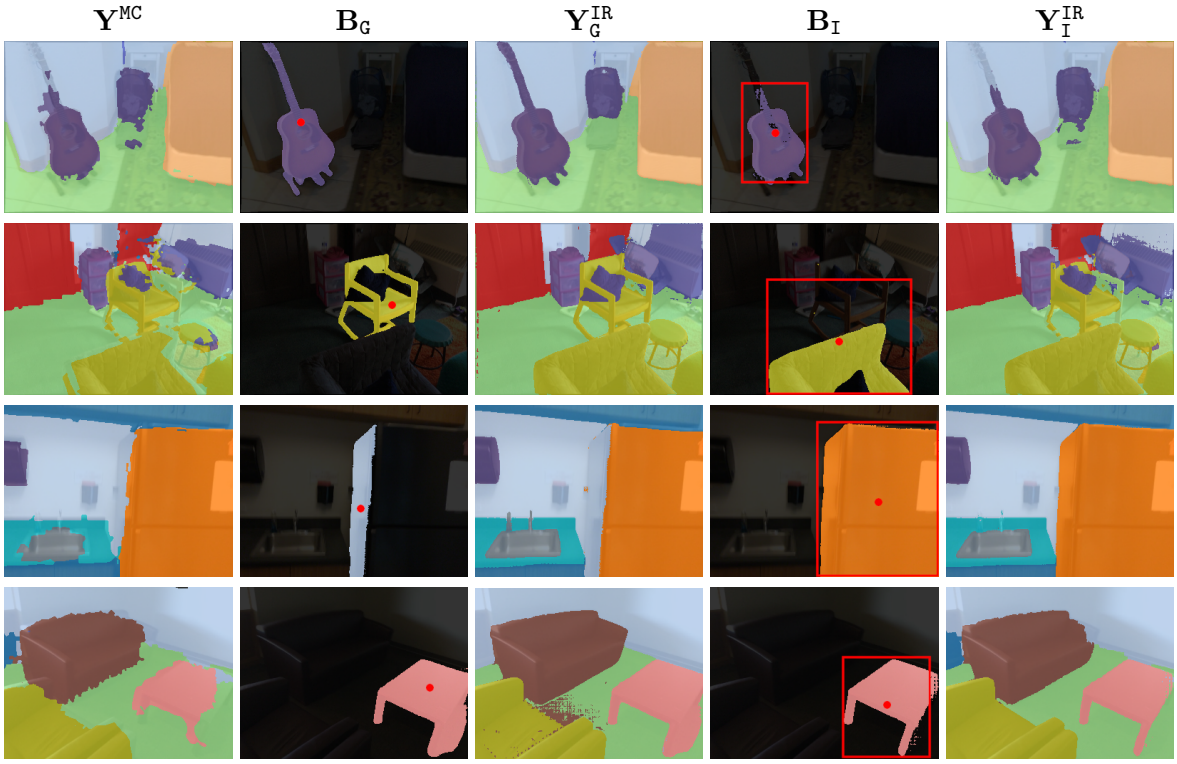


Figure 4.5: Comparison of pseudo–labels from our method using `grid` ($\mathbf{Y}^{\mathtt{IR}}_{\mathtt{G}}$) and `informed` ($\mathbf{Y}^{\mathtt{IR}}_{\mathtt{I}}$) prompts, with representative SAM instances and their prompts ($\mathbf{B}_{\mathtt{G}}$, $\mathbf{B}_{\mathtt{I}}$). $\mathbf{I}^t$ and $\mathbf{Y}^{\mathtt{MC}}$ denote the RGB images and multi–view annotations. The first two columns highlight the strengths of `grid`, while the last two show the improved precision of `informed`.

4.1.6 Evaluation of the Adapted Model

In this section, we present the experimental results evaluating the performance of our method on the semantic segmentation task. Table 4.2 shows the results obtained when fine–tuning and testing inside the same sequence, using the first 80% and the last 20% of the frames, respectively (as done in [38] and [39]). We run all the experiments setting the voxel size of Kimera Semantics to 3cm and 5cm.

The results demonstrate that our instance–aware refinement remarkably increases the model performance with respect to the multi–view consistency baseline of [38] (MC), and the raw predictions of the pre–trained $f^{seg}_{\theta_s}$ (PT). Considering a voxel size of 5cm, our method improves the baseline MC by $4,3\%$ and $5,7\%$ using the `grid` ($\text{IR}_{\mathtt{G}}$) and `informed` ($\text{IR}_{\mathtt{I}}$) prompt strategies. As observed in the previous section, the impact of the multi–view consistency on the pre–trained model PT varies a lot across environments, registering remarkable improvements (e.g., in environments 1, 7, and 9), but also important degradations (such as in environments 2, 6, and 8). Starting from this, our method improves the performance of MC in both cases: the instance–aware refinement step not only increases the segmentation accuracy when MC is effective, but also mitigates and fixes errors of the multi–view consistency in case of failure. See, for example, scenes 2 and 6: while MC degrades PT by $\approx$ -14% and -9%, our method $\text{IR}_{\mathtt{I}}$ substantially

Env	PT	Baseline (5cm) MC	Δ_{PT}	Our method IR_G	Δ_{MC}	IR_I	Δ_{MC}	IR_{GI}	Δ_{MC}	PT	Baseline (3cm) MC	Δ_{PT}	Our method IR_G	Δ_{MC}	IR_I	Δ_{MC}	IR_{GI}	Δ_{MC}
1	44,2	48,2	9,0%	**51**	5,8%	49,3	2,3%	50,9	5,6%	44,2	47,6	7,7%	50,1	5,3%	50,4	5,9%	**51**	7,1%
2	**36,2**	31,3	-13,5%	33,6	7,3%	34,4	9,9%	34,8	11,2%	**36,2**	34,4	-5,0%	34,5	0,3%	34,9	1,5%	33,8	-1,7%
3	22,9	21	-8,3%	21,4	1,9%	**23,2**	10,5%	22,9	9,0%	22,9	21,7	-5,2%	22,6	4,1%	23,1	6,5%	**23,2**	6,9%
4	50,1	51,1	2,0%	52,5	2,7%	52,5	2,7%	**53,4**	4,5%	50,1	52,4	4,6%	52,9	1,0%	49,5	-5,5%	**53,7**	2,5%
5	39,7	40,1	1,0%	44,5	11,0%	44,4	10,7%	**44,7**	11,5%	39,7	45	13,4%	47,2	4,9%	46,1	2,4%	**47,3**	5,1%
6	**35**	31,9	-8,9%	33,9	6,3%	34,1	6,9%	34,9	9,4%	35	33,5	-4,3%	39,5	17,9%	38,8	15,8%	**40,1**	19,7%
7	56,7	63,5	12,0%	60,7	-4,4%	**63,5**	0,0%	60,9	-4,1%	56,7	**63**	11,1%	60,9	-3,3%	56,7	-10,0%	59,5	-5,6%
8	**29,5**	25,1	-14,9%	25,2	0,4%	26,3	4,8%	27,6	10,0%	**29,5**	25,1	-14,9%	24,1	-4,0%	27	7,6%	25,7	2,4%
9	55,9	65,3	16,8%	**70,2**	7,5%	69	5,7%	69,2	6,0%	55,9	66,9	19,7%	**70,9**	6,0%	69,8	4,3%	69,9	4,5%
10	73,4	71,9	-2,0%	**75,3**	4,7%	74,7	3,9%	75	4,3%	73,4	73,2	-0,3%	**75,7**	3,4%	75,6	3,3%	75,3	2,9%
Avg	44,4	44,9	-0,7%	46,8	4,3%	46,8	5,7%	**47,4**	6,7%	44,4	46,3	2,7%	47,8	3,6%	47,2	3,2%	**48,0**	4,4%

Table 4.2: Improvements of $f^{seg}_{\theta_t}$ trained using the instance–aware pseudo–labels using `grid` (IR_G), `informed` (IR_I), and the combination (IR_{GI}) of the strategies compared with the fine–tuning using the multi–view consistent annotations (MC) and the outputs of the pre–trained model $f^{seg}_{\theta_s}$ (PT). The voxel size is set to 5cm and 3cm. Training and testing is performed using the first 80% and the last 20% of the first sequence of each environment. Best and second best performances are in **bold** and underlined.

improves the quality of the baseline $\approx 10\%$ and 7%, obtaining segmentation accuracies close to PT. Table 4.2 also shows the benefits of fine–tuning the source model $f^{seg}_{\theta_s}$ combining the pseudo–labels obtained with the `grid` and `informed` prompting strategies (IR_{GI}). Mixing the two different types of instance–aware annotations during training mitigates the negative intrinsic effects of `grid` and `informed` (previously described in Section 4.1.5), helping the model in providing more accurate outputs. More specifically, IR_{GI} reaches an average increment of $6,7\%$ over the baseline MC, achieving the best or the second best performance across all environments (except for scene 7 where IR_I and MC perform better). All these outcomes are confirmed using a voxel size of 3cm. As expected, the quality of the baseline MC is improved from $44,9$ to $46,3$ mIoU. This is because a smaller voxel size limits the discretization of the world encoded in the 3D map, thus reducing the artifacts produced by the ray–tracing procedure. Despite this, our method increases the performance of the baseline MC by $3,6\%$ and $3,2\%$ with IR_G and IR_I, respectively. Again, the best performances are obtained by mixing the prompting strategies (IR_{GI}) during training, with an average increment of $4,4\%$ over MC. Interestingly, while the baseline MC is strongly influenced by the voxel size of Kimera Semantics, our method achieves comparable performance regardless of the voxels' resolution (0.6 mIoU difference obtained IR_{GI} with 3cm and 5cm). This is particularly useful in practical deployment scenarios, where the voxel dimension can be set to a higher value, thus reducing the computational overhead of the robot in integrating the frames in Kimera Semantics.

While the aforementioned results of Table 4.2 are collected in a subportion of the environment, Table 4.3 reports the performance of $f^{seg}_{\theta_t}$ considering the entire environment, as training and testing are performed in different sequences. The results of this additional experiment further strengthen the findings of the previous evaluation. Using a voxel size of 5cm (3cm), our method outperforms the MC baseline on average by $\approx 7\%$ (4%), obtaining a remarkable performance increment in all scenes. The only case where the pre–trained model (PT) performs worse after adaptation is in scene 2, which is a problematic environment since almost the 30% of the

Env	PT	Baseline (5cm) MC	Δ_{PT}	Our method IR_G	Δ_{MC}	IR_I	Δ_{MC}	IR_{GI}	Δ_{MC}	PT	Baseline (3cm) MC	Δ_{PT}	Our method IR_G	Δ_{MC}	IR_I	Δ_{MC}	IR_{GI}	Δ_{MC}
1	47,1	49,6	5,3%	**55**	10,9%	51,2	3,2%	54,3	9,5%	47,1	48,9	3,8%	50,6	3,5%	49,5	1,2%	**50,8**	3,9%
2	**44,1**	35,2	-20,2%	39,6	12,5%	39,6	12,5%	39,3	11,6%	**44,1**	42,1	-4,5%	41,8	-0,7%	41,6	-1,2%	41,6	-1,2%
3	30,6	30,5	-0,3%	32,8	7,5%	34,4	12,8%	**34,7**	13,8%	30,6	31,5	2,9%	**36,5**	15,9%	35,1	11,4%	36,3	15,2%
4	55,3	55,8	0,9%	57,2	2,5%	56,2	0,7%	**57,7**	3,4%	55,3	57,1	3,3%	**57,4**	0,5%	56,8	-0,5%	57,2	0,2%
6	49,2	50,5	2,6%	52,8	4,6%	**54,7**	8,3%	53	5,0%	49,2	51,2	4,1%	54,8	7,0%	**55,8**	9,0%	54,8	7,0%
7	52	54,4	4,6%	55,5	2%	**57,2**	5,1%	56,2	3,3%	52	54,1	4%	54,8	1,3%	55,1	1,8%	**55,3**	2,2%
10	57,5	63,2	9,9%	68,7	8,7%	67,6	7,0%	**68,7**	8,7%	57,5	65,2	13,4%	69,1	6,0%	68,3	4,8%	**70,1**	7,5%
Avg	48	48,5	0,4%	51,7	7%	51,6	7,1%	**52,0**	7,9%	48	50	3,9 %	52,1	4,8 %	51,7	3,8%	**52,3**	5,0%

Table 4.3: Improvements of $f_{\theta_t}^{seg}$ trained using the instance–aware pseudo–labels using `grid` (IR_G), `informed` (IR_I), and the combination (IR_{GI}) of the strategies compared with the fine–tuning using the multi–view consistent annotations (MC) and the per–frame outputs of the pre–trained model $f_{\theta_s}^{seg}$ (PT). The voxel size is set to 5cm and 3cm. Training and testing are performed using different sequences of the same environment. Best and second best performances are in **bold** and underlined.

frames have missing positions and a large portion of the scene is not annotated. Despite this, our instance refinement strongly increase the baseline's performance of $\approx 12\%$ with respect to MC. Again, the best averaging performances are obtained by IR_{GI}, which reaches almost the same performance using 5cm and 3cm as voxel sizes, respectively 52 and $52,3$ mIoU. The improvements provided by our method can be observed in the qualitative examples of Figure 4.6. Despite the baseline (MC) effectively removes the noise of the raw model's predictions (PT), the $f_{\theta_t}^{seg}$ learns the errors that are persistent errors between multiple frames, like the doors in the first and last rows that are confused with the wall. In contrast, our method remarkably increases the quality of the segmentation masks produced by $f_{\theta_t}^{seg}$, solving also some artifacts of the ground–truth. This can be seen by observing the precision of the masks related to the sink and the furniture in the second and third rows.

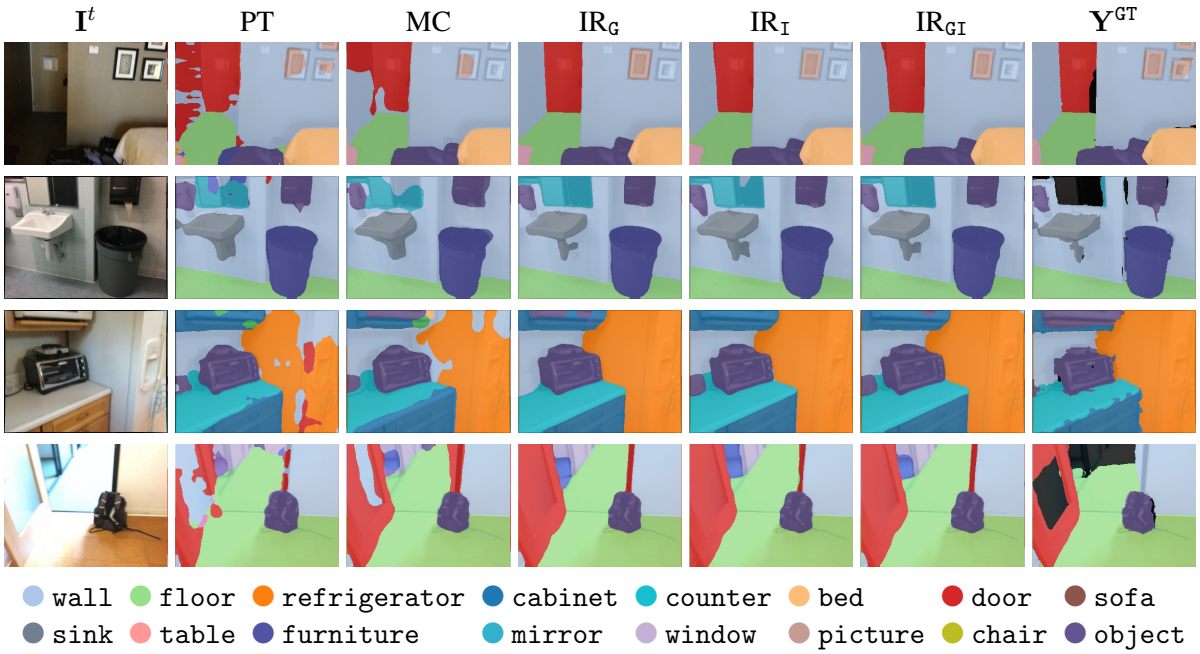


Figure 4.6: Segmentation improvements with different prompting strategies ($\mathrm{IR}_{\mathtt{G}}$, $\mathrm{IR}_{\mathtt{I}}$, $\mathrm{IR}_{\mathtt{GI}}$) compared to the baselines (MC and PT). The first two rows use a 5cm voxel size, the last two a 3cm. Black areas in $\mathbf{Y}^{\mathtt{GT}}$ are due to rendering errors or missing labels; these segmentation errors are improved by our IR.

4.2 Self–Supervised Domain Adaptation with State–Consistency

The state–consistency [113] enforces two distinct predictions, obtained from perceptions acquired from different viewpoints, to be consistent according to the robot's movement between the two locations in which the frames have been acquired. Consider, for example, a robot that uses a neural network to predict the relative 3D position of a static object using visual perceptions. During operation, the robot observes the same object multiple times and, since the object is static, the predictions should be consistent with the robot's movements. This means that, if we consider the predictions of two consecutive frames, the estimated location of the object from the second frame should be equal to the estimation obtained from the first frame, but translated to the position of the second frame. This can be easily obtained by projecting the two predictions in the same frame of reference using the estimated movement between the two poses (i.e., the robot's odometry). Instead of using a dense 3D environmental representation of the environment (as performed by the multi–view consistency), the state–consistency is directly embedded during the fine–tuning using a loss function. Considering two examples (or frames) inside the same batch during training, this loss term at first (i) projects in the same frame of reference their corresponding pose estimations using the robot's odometry and then (ii) forces them to be equal (see Figure 4.7 for a visual example).

Leveraging the state–consistency, we propose a self–supervised approach for 3D object detection. This task enables the robot to infer the displacement of objects in 3D with respect to its position leveraging visual perceptions. This task is particularly relevant for ultra–low powered robots, such as nano–drones, where standard localization algorithms cannot be run in real–time for computational constraints. In this context, we specifically consider the 3D pose estimation of racing gates from monocular perceptions [40], which is essential in drone racing competitions where drones have to pass through gates in a precise order to win a race (see Figure 4.7). We adopt the Bitcraze Crazyflie 2.1 Brushless[1] nano–drone platform (depicted in Figure 4.7), which mounts a grayscale, low–resolution camera with a limited dynamic range. The robot is initially equipped with a general 3D pose estimator pre–trained using a large quantity of simulated data. Our goal is to self–supervise the adaptation of the aforementioned model to the real–world where perceptions are noisy, affected by motion blur and vignetting, and, crucially, significantly differ from those acquired in simulation (see the examples in Figure 4.7). Differently from the naive adaptation solution of fine–tuning the model on target domain data acquired labeled with a motion capture system [37], our approach performs sim–to–real domain transfer using target domain data (images and odometry) autonomously–collected by the drone flying random trajectories in front of a gate. We derive a supervision signal from the odometry data by enforcing a geometric state–consistency loss [113] between pairs of poses predicted by the model. Given two images of the gate taken from different locations, our loss imposes that the two predicted relative gate poses must be compatible with the drone's measured movement

[1] https://www.bitcraze.io/products/crazyflie-2-1-brushless

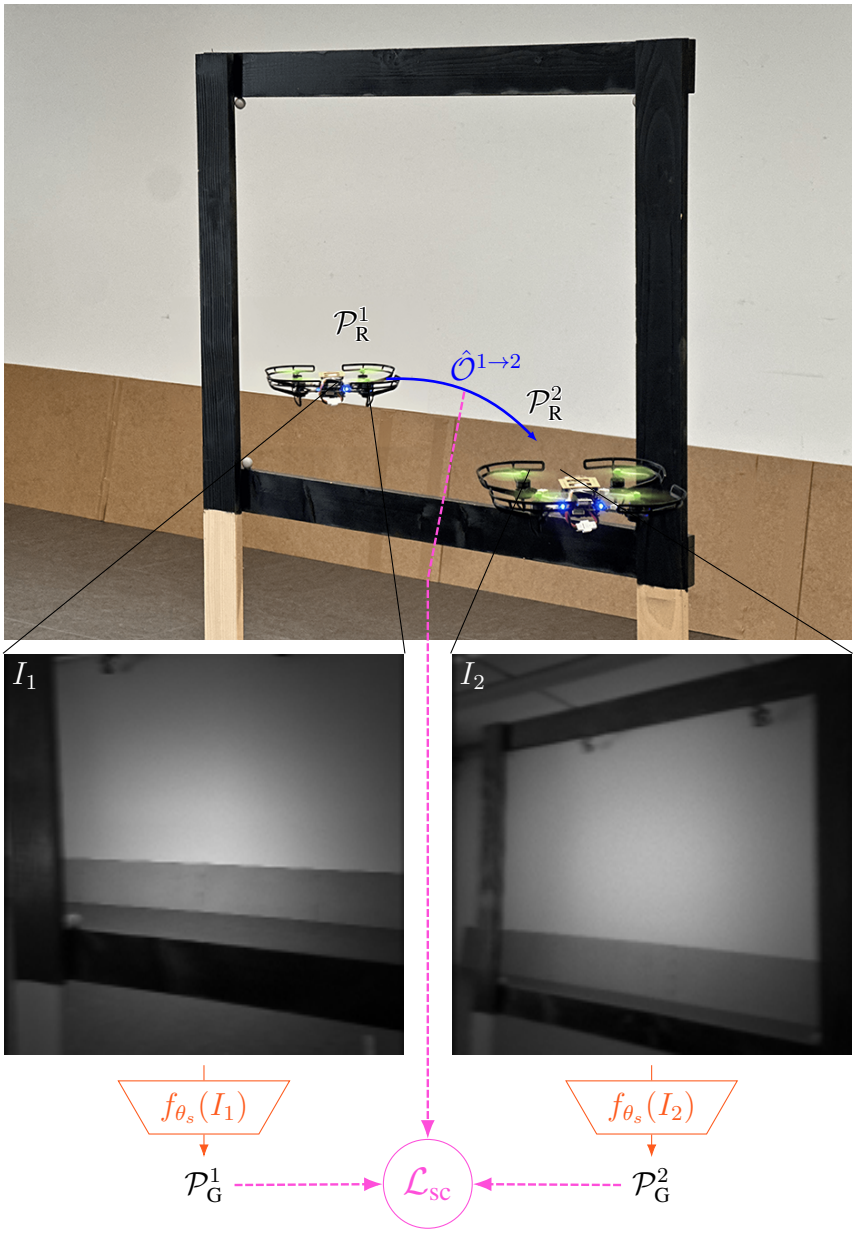


Figure 4.7: We estimate the pose (x, y, z, ψ) of drone racing gates by transferring a model trained in simulation to a real–world environment and deploying it aboard the Bitcraze Crazyflie 2.1 Brushless nano-UAV. Lets consider the drone moving from position $\mathcal{P}_{\mathrm{R}}^{1}$ to $\mathcal{P}_{\mathrm{R}}^{2}$. The model f_{θ_s}, where weights θ_s are initialized using a source simulated dataset, predicts the relative poses of the gate $\mathcal{P}_{\mathrm{G}}^{1}$ and $\mathcal{P}_{\mathrm{G}}^{2}$ using the frames I_1 and I_2 acquired from those positions. To implement the state–consistency, the gate estimations are reported in the same frame of reference using the robot's estimated odometry $\hat{\mathcal{O}}^{1\to2}$, and then they are forced to be equal by the state–consistency loss $\mathcal{L}_{\mathrm{sc}}$.

between the two locations (see Figure 4.7). Compared to standard computer vision approaches for domain adaptation, our work is designed for robotic platforms and uses the robot's sensors as the basis of the approach. Our main contribution (which comes from the work of [49]) is a novel application of the state–consistency loss for the domain transfer of a 3D gate pose estimation task; the loss is combined with a self–supervised data collection pipeline, making the

approach practical and easily applicable in any target domain.

4.2.1 Problem Formulation: Adaptation for 3D Pose Estimation

The problem we consider (visually explained in Figure 4.7) is the estimation of the pose of a drone racing gate $\mathcal{P}_{\text{G}} \in SE(3)$ relative to an observing drone, given a single grayscale image I with a resolution of 160×160 pixels. The model $f_{\theta_s}(I)$ parametrized by weights θ_s takes a single image I as input and produces the estimated gate pose $\hat{\mathcal{P}}_{\text{G}}$. Initially, the model's parameters θ_s are trained using a source dataset $\mathcal{S}$ coming from a simulator, allowing us to easily collect images labeled with the ground–truth gate pose $\mathcal{P}_{\text{G}}$. Our goal is to transfer the general model trained in simulation to the real world by fine–tuning it with a new dataset $\mathcal{T}$ collected by a real drone flying in front of a gate. $\mathcal{T}$ is annotated with the drone's odometry while having no access to information about the gate and its pose. The drone odometry collected at time $t = 1$ w.r.t. the world reference frame is denoted by $\hat{\mathcal{O}}^{1 \to w}$.

4.2.2 State–Consistency as a Loss Function

At first, we pre–train the general model in simulation using $\mathcal{S}$. This is reached by minimizing $\mathcal{L}_{\text{pose}} = \text{MSE}(\mathcal{P}_{\text{G}}, \hat{\mathcal{P}}_{\text{G}})$, that is the mean squared error (MSE) loss between the ground truth gate pose $\mathcal{P}_{\text{G}}$ and its estimation $\hat{\mathcal{P}}_{\text{G}}$ produced by the model. While the position is directly passed to the MSE loss, the orientation components are first converted to the 6D representation proposed in Zhou et al. [188], preserving the continuity of 3D rotations to aid the training process.

For the real–world qualification using $\mathcal{T}$, we enforce a State Consistency (SC) loss [113] between two model predictions exploiting the known drone odometry. Given a pair of images taken by the drone in a static scene from two different locations, the difference between the estimated relative gate's poses as perceived by the drone in the two locations must be equal to the drone's movement between the first and second locations (see Figure 4.8 for a general overview). Formally, given two input images I_1, I_2, the gate poses $\mathcal{P}^1_{\text{G}}, \mathcal{P}^2_{\text{G}}$, and the drone odometries $\hat{\mathcal{O}}^{1 \to w}, \hat{\mathcal{O}}^{2 \to w}$, we first make the model predict $\hat{\mathcal{P}}^1_{\text{G}}, \hat{\mathcal{P}}^2_{\text{G}}$. Then, we warp $\hat{\mathcal{P}}^1_{\text{G}}$ to the frame of reference of the drone at the position in which I_2 was taken $\hat{\mathcal{P}}^{1 \to 2}_{\text{G}} = \hat{\mathcal{P}}^1_{\text{G}} \cdot \hat{\mathcal{O}}^{2 \to w} \cdot (\hat{\mathcal{O}}^{1 \to w})^{-1}$, where $(\cdot)$ denotes matrix multiplication and $\mathcal{M}^{-1}$ denotes the inverse of matrix $\mathcal{M}$. The SC loss is then written as:

$$\mathcal{L}_{\text{sc}} = \text{MSE}(\hat{\mathcal{P}}^{1 \to 2}_{\text{G}}, \hat{\mathcal{P}}^2_{\text{G}}) \tag{4.1}$$

where orientation components of the poses are converted to the 6D representation proposed in Zhou et al. [188].

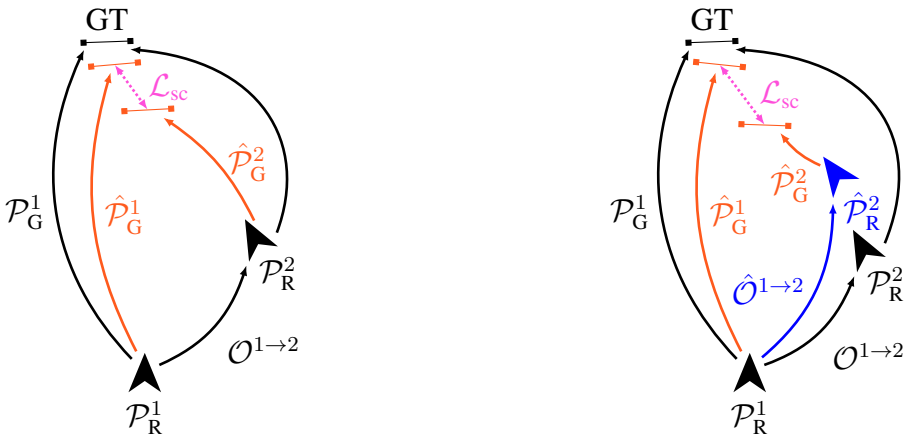


Figure 4.8: Assuming perfect odometry (left), the state consistency loss forces the relative gate poses $\hat{\mathcal{P}}^1_G$ and $\hat{\mathcal{P}}^2_G$, predicted from the drone's poses at $\mathcal{P}^1_R$ and $\mathcal{P}^2_R$, to be coherent with the drone's relative odometry $\mathcal{O}^{1\to2}$ between the two poses. In our experiments, the measured odometry $\hat{\mathcal{O}}^{1\to2}$ is collected by the drone itself and is affected by drift and noise (right).

Figure 4.9: The Crazyflie 2.1 Brushless nano–drone used in our experiments.

4.2.3 Experimental Setting

Robot Platform We target the Crazyflie 2.1 Brushless by Bitcraze, a commercial off–the–shelf nano–drone extended with the pluggable GAP9Shield [189] and the Flow–deck expansion modules, depicted in Figure 4.9. The GAP9Shield features a GreenWaves Technologies GAP9 ten–core RISC–V System–on–Chip (SoC). It features a dedicated NE16 neural accelerator capable of up to $150\,\mathrm{MAC/cycle}$ in 8–bit integer arithmetic, a peak throughput of $55\,\mathrm{GOp/s}$ when running at $370\,\mathrm{MHz}$, while consuming less than $70\,\mathrm{MHz}$. The SoC integrates a $512\,\mathrm{kB}$ L2 memory and a low–latency $64\,\mathrm{kB}$ L1 scratchpad, complemented by off–chip $32\,\mathrm{MB}$ OctaSPI RAM and $64\,\mathrm{MB}$ Flash memories, while a VGA–resolution OmniVision OV5647 camera provides visual input, and the Espressif ESP32 module offers Wi–Fi connectivity. The Crazyflie 2.1 base platform employs an STM32 microcontroller to handle low–level flight–control tasks and sensor interfacing: a 6–DoF IMU (three–axis accelerometer and gyroscope), an STMicroelectronics VL53L1X laser time–of–flight altitude sensor, and a PMW3901 optical–flow sensor to track horizontal displacement. The sensor data is fused by an Extended Kalman Filter [190] to provide the onboard odometry.

Figure 4.10: Examples of (left) simulated images from $\mathcal{S}^{\text{train}}$, (middle) real–world robot images in $\mathcal{T}^{\text{train}}$, and (right) after applying the pencil filter [75].

Data Collection We collect training data in simulated and real–world environments. In both domains, our $100\text{cm} \times 80\text{cm}$ gates are composed of four black–painted wooden beams placed at $75\,\text{cm}$ height, based on the setup from the first Nanocopter AI challenge at IMAV 2022 [40]. For the simulated data, we use the Webots simulator [33] to generate a dataset of 75K images depicting a gate in random poses. To align the visual aspect of the simulated data in $\mathcal{S}$ with the images captured in the real world by the drone, we convert the images into grayscale and we perform data augmentations during training using Gaussian blur, vignetting, multiplicative Gaussian noise, and random exposure (see Figure 4.10).

To collect real–world images, we use the Crazyflie 2.1 Brushless piloted by a human to follow random trajectories around the gate while keeping it in the camera's FOV. During flight, we collect images at 25 FPS with a resolution of 160×160 pixels. For testing purposes only, we collect ground truth pose data for both the drone and the gate with a motion capture system installed in our laboratory. Real data is split into the training set $\mathcal{T}^{\text{train}}$ (51K samples), the validation set $\mathcal{T}^{\text{val}}$ (8K samples), and the testing set $\mathcal{T}^{\text{test}}$ (21K samples). Examples of real-world images acquired by our drone are depicted in Figure 4.10 (middle).

Model Training and Onboard Deployment Our model architecture is a convolutional neural network with four convolutional blocks totaling 6M parameters. For simulation training, we optimize the $\mathcal{L}_{\text{pose}}$ loss using AdamW [180] as the optimizer with a learning rate of $1e^{-3}$ for 100 epochs. Then, we pick the model parameters with the lowest loss on $\mathcal{S}^{\text{val}}$ and fine–tune it on real–world data by optimizing the $\mathcal{L}_{\text{sc}}$ loss using AdamW with a learning rate of $1e^{-6}$ for 100 epochs. During fine–tuning, we use a small validation set $\mathcal{T}^{\text{val}}$, which does not contain pose labels, to pick the best model weights over the training process.

We deploy our model on the GWT GAP9 SoC to run entirely aboard our target nano–drone. For maximum inference throughput, we aim to take advantage of the chip's NE16 neural accelerator, which only supports mixed–precision integer arithmetic. We employ the GWT NNTool + AutoTiler deployment pipeline to quantize the model to 8–bit integers, with no loss in regression performance, and generate the C inference code. This includes NE16 convolutional and fully–connected kernels, optimized parallel CPU kernels for the remaining layers, and automatically scheduled memory transfers across the chip's memory hierarchy.

Evaluation Metrics We evaluate the model's ability to estimate the gate's relative pose on the real–world testing set $\mathcal{T}^{\text{test}}$ by measuring the mean absolute error (MAE) and the Pearson

correlation coefficient (ρ) between predicted and ground truth components of the 3D position and yaw of the gate; specifically for the yaw angle, we consider the circular Pearson correlation coefficient [191]. Due to the drone's flight dynamics, roll and pitch angles of the gate relative to the drone have a low variability in our data. As such, metrics on these variables would be uninformative; we choose not to present them in our evaluation.

Our model predictions correctly capture the pose of the gate but are inherently affected by a constant bias stemming from the nature of the state consistency loss: The loss is minimized as long as the predicted pose pairs are spatially consistent with the drone's movements, yet they are not forced to be anchored at the gate's center. This bias can be corrected with a calibration procedure by picking a small batch of images where the position of the gate relative to the camera is known. The model is run on these images, and the mean offset between the gate's position and its estimate is measured. To correct the bias, a constant offset is added to predicted positions. In presenting our results, we apply this calibration procedure by considering the whole testing dataset. This correction only affects the MAE metric for $x, y,$ and z, since the linear correlation coefficient is not affected by a constant bias. For fairness, we apply the same bias correction procedure to all the considered baselines.

Baselines We validate our approach considering different baselines. The first one (*Zero-Shot*) is our neural network trained only with the simulated dataset $\mathcal{S}^{\text{train}}$ and tested directly on $\mathcal{T}^{\text{test}}$. The second baseline we consider is *PencilNet* [75], a state–of–the–art domain generalization approach for gate pose estimation. It consists of pre–processing the model's input images using a pencil filter to align the visual aspect of simulated and real–world data, as shown on the right of Figure 4.10. In this way, the model is trained in simulation and deployed to real scenarios in a zero–shot fashion. Our third baseline (*DA*) is the well–established method for unsupervised domain adaptation described in [91]. This approach uses a two–stream architecture to align the distributions of the source and target domains with Maximum Mean Discrepancy (MMD). Furthermore, an additional loss function forces the parameters of the two streams to be linearly dependent. Similarly to our approach, no ground truth annotations in the real world are required. To ensure a fair evaluation, we re–implement all the baselines using our model architecture and our training dataset acquired in simulation, reporting the performance obtained by averaging results over three training runs with different parameter initialization. We also report the performance of the *Mean Predictor*, which always predicts the average gate pose in the test set $\mathcal{T}^{\text{test}}$.

4.2.4 Evaluation of the Adapted Model

We report the results for our approach and all baselines in Table 4.4. We observe that our method consistently outperforms all the baselines: The *Zero–Shot* baseline, namely the model trained in simulation and tested in the real world, brings only marginal improvements with respect to the lower bound represented by the *Mean Predictor*. This is observed also with *PencilNet* that, while reaching performance comparable to the *Mean Predictor*, remarkably degrades the

Model	Supervision	MSE ↓	MAE ↓				ρ ↑				Plot for MSE$_{xyz}$ [cm] ←
		xyz [cm]	x [cm]	y [cm]	z [cm]	ψ [deg]	x [%]	y [%]	z [%]	ψ [%]	Error bars mark 95% CI
Mean Predictor	$\mathcal{T}^{\text{test}}$	90.7	57.0	48.7	23.1	20.9	–	–	–	–	
Zero–Shot	$\mathcal{S}^{\text{train}}$	79.7	51.6	41.0	24.5	33.5	38	53	40	5	
PencilNet [75]	$\mathcal{S}^{\text{train}}$	88.3	56.7	46.8	22.5	32.7	4	40	44	2	
DA [91]	$\mathcal{S}^{\text{train}}$, $\mathcal{T}^{\text{train}}$	103.5	66.0	34.7	33.0	28.8	30	69	52	5	
Ours	$\mathcal{T}^{\text{train}}$ (Odometry only)	**44.7**	**25.6**	**28.2**	**10.5**	**13.1**	**88**	**79**	**91**	**82**	0 30 60 90

Table 4.4: Mean absolute error MAE and Pearson correlation coefficient ρ on the real–world testing set $\mathcal{T}^{\text{test}}$, 3 runs per row.

estimation of the gate's distance (x) in terms of MAE and ρ. This suggests that the domain generalization approach fails to overcome a large sim–to–real gap, such as the one found in our setup. The results also show that our domain adaptation (*DA*) baseline from [91] is not able to improve the performance of the *Mean Predictor*. Despite being beneficial for y, it degrades the estimation of both x and z, demonstrating that performing distribution alignment is not sufficient to fill the domain gap of our scenario. Compared with *Mean Predictor*, our approach improves the MAE by 54% in x and z (distance and height of the gate) and by 42% in y (horizontal displacement). Moreover, our approach is the only one that manages to reduce the error in the estimation of ψ, with a gain of 37%. Notably, the geometric relation between predictions imposed by the state consistency loss strongly improves the correlation ρ of the model outputs with their respective ground truths (see Figure 4.11).

Figure 4.12 reports qualitative examples of our model on gate pose estimation. In the first three images, the model precisely localizes the gate, even when it is far away from the drone. In the last two examples, challenging situations where the gate partially exits the image boundaries or is perceived from a narrow point of view cause the model to fail.

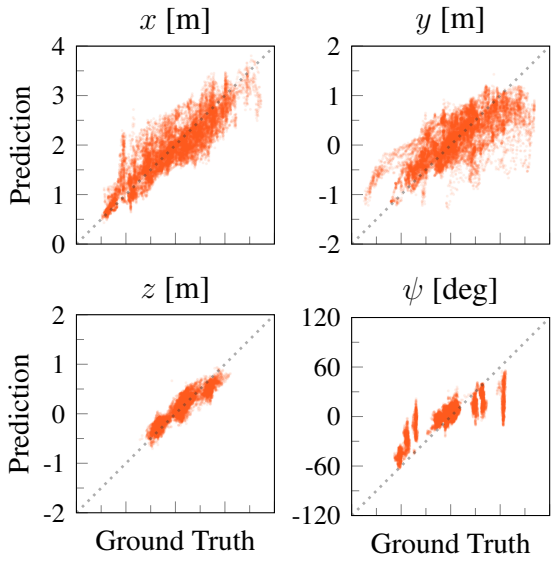


Figure 4.11: Our self–supervised domain adaptation approach vs ground truth on $\mathcal{T}^{\text{test}}$ for the components of the gate pose (x, y, z, ψ).

Following the pipeline described in Section 4.2.3, we deploy our network in the Crazyflie 2.1 quadrotor. Counting $6\,\mathrm{M}$ parameters, the model requires $318\,\mathrm{MMAC}$ per inference and $30.4\,\mathrm{ms}$ for processing a single frame on the NE16. With an inference throughput of $32.8\,\mathrm{frames/s}$, our architecture is fully suitable for closed–loop autonomous control aboard the drone during real–world flights.

Figure 4.12: Input frames overlayed with gate skeleton (in orange) using the pose predicted by our approach. The model correctly locates the gate (1-3), while failures occur when a large portion of the gate is occluded and for shallow angles (4-5).

As a final experiment, we analyze the performance of our approach with varying amounts of target domain data: 1, 10, 20, 40, and 60 recording sequences, each one lasting 30 seconds and amounting to 850 samples. Results reported in Figure 4.13, show that training our approach with 10 sequences is enough to surpass all of the baselines' performance. By training with 40 sequences, we closely match the performance of the full dataset, showing diminishing returns, especially in estimating the gate's position. These results prove that the approach is effective, and even small amounts of target domain data suffice to outperform state–of–the–art baselines.

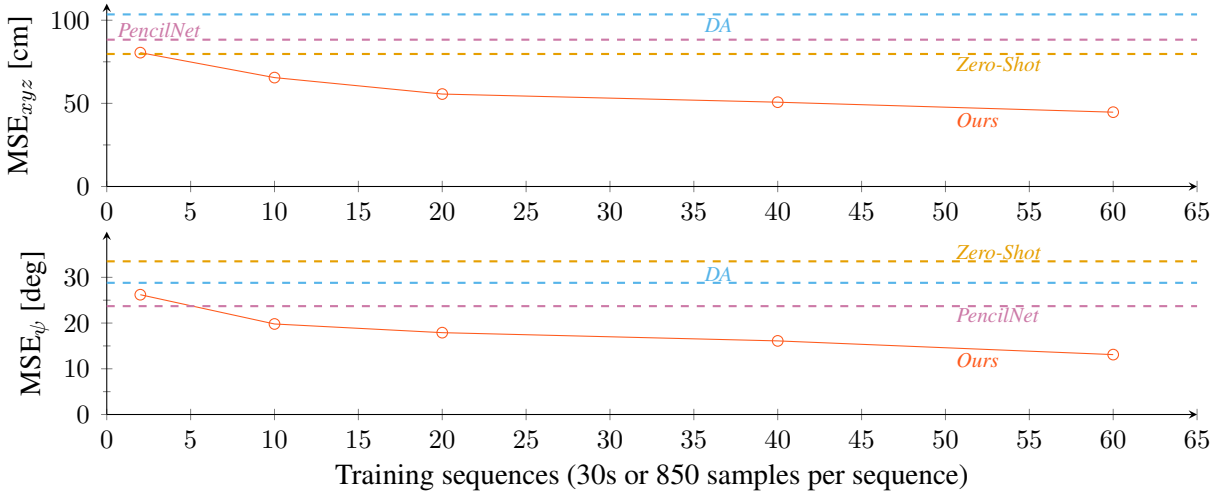


Figure 4.13: Position (top) and orientation (bottom) performance measure with MSE of our approach with increasing amounts of fine–tune data. DA is trained on the full real–world dataset (60 sequences), while PencilNet and Zero–Shot are trained only on $\mathcal{S}^{\text{train}}$.

4.3 Lessons Learned and Future Works

This chapter reports two alternative methods to enable the unsupervised adaptation of DNNs for computer vision used by mobile robots. More specifically, we present an approach to improve the multi–view consistency using 3D maps forcing the object categories to be consistent with object instances, thus improving the quality of the pseudo–labels in dense visual tasks (such as semantic segmentation). In addition, we show that the perceptions' consistency between different viewpoints can be directly embedded in the loss function working with pose estimation tasks, avoiding the computationally intensive step of aggregating perceptions in a 3D dense representation of the environment.

With our extensive experiments with real–world data, we demonstrate that a promising way to perform unsupervised adaptation in robotics is to perform a fine–tuning with pseudo–labels leveraging their consistency with the 3D world and the robot's movements to filter out the noise of the model, thus enabling an effective self–supervision. Since our contributions are applied in different perception tasks (semantic segmentation and 3D pose estimation), future works will explore alternative solutions to merge the two solutions we provide in a single framework, suitable for multiple kinds of visual tasks used in robotics. A solution could be extending the state–consistency using depth data while training DNN's for semantic segmentation.

Chapter 5

Domain Adaptation and Privacy Preservation in Cloud Robotics

Robotic systems increasingly require advanced computational capabilities to handle the intensive inference demands of modern deep learning architectures. These architectures allow robots to autonomously perform tasks such as perception, navigation, and planning, but they also pose challenges in terms of efficiency and real–time performance. The computational resources required by the modern end–to-end architectures based on deep learning are at odds with the typical profiles of mobile robots: not only are they devices with limited resources, but they need to be. Keeping affordable hardware costs and preserving energy consumption at operational time are mandatory requisites in many real–world scenarios.

Cloud robotics is an increasingly relevant paradigm since it enables robotic practitioners to engineer architectures in which the robot collects data, while inference is carried out remotely, after transmission [114, 125, 50]. This paradigm allows dealing with the typical conflict between low–powered robots and high–demand DNNs inference [124]. Consider a mobile robot deployed in the real world that needs to perform a visual task (e.g., object detection) from camera readings. Instead of processing frames locally using its hardware, the robot sends the perceptions to a remote server that performs inference using a DNN for robotic vision, and then returns the predictions to the robot. In this chapter, we refer to that model as *TaskNet*. We implement the TaskNet as a general model, as it can be shared by multiple robots operating in different environments. Robots and TaskNets can communicate through the network using low–latency frameworks that can be easily integrated with the Robot Operating System (such as FogROS [42, 41]). Furthermore, recent solutions on orchestration platforms for mixed–critical robotic applications [43, 41] allow low–powered robotic platforms to access extensive computational resources in the cloud with real–time performance. Despite offloading computationally demanding inference with DNNs is an emerging trend in the field, two primary issues arise.

- **Scalable domain adaptation:** since the TaskNet is deployed in the cloud, it can be queried by multiple robots operating in different environments. Given this, the TaskNet is a general model as it needs to perform reasonably well with data coming from different domains. In such a scenario, the standard solutions for domain adaptation based on fine–

tuning [27, 151] exhibit important scalability issues. Given that fine–tuning on a domain will degrade performance on a different one, the cloud provider must have an independent qualified TaskNet for every robot. Clearly, developing, re–training, and maintaining a DNN for each domain is prohibitively expensive and quickly becomes unfeasible as the number of robots increases.

- **Privacy preservation:** service robots often operate in human–centric environments, performing tasks such as domestic assistance, healthcare, and logistics [2] using readings from high–resolution cameras for semantic perception [7]. When the inference pipeline is decentralized in the cloud, privacy becomes a major issue as robots are sending very privacy–sensitive data (i.e., RGB images) to potentially untrusted cloud providers [44].

In this chapter, we provide solutions to these challenges when the TaskNet is a neural network for object detection. To do this, we leverage a key architectural aspect of the modern DNNs for object detection [148]. To precisely locate multiple objects in the image plane, they produce a dense set of overlapping *object proposals* (typically in the order of thousands) to densely classify image features. To avoid redundant detections, this dense set is then filtered with a heuristic algorithm called Non Maximum Suppression (NMS) [192], where groups of high–overlapping proposals are suppressed, maintaining only the one with the highest confidence. Instead of modifying the TaskNet, we reach scalable adaptation and privacy preservation by using two lightweight neural networks that can be run with the robot's hardware. To train them, we manipulate this dense set of object proposals in two different ways before the NMS.

- For scalable adaptation, we add a *proposal refinement* step after the TaskNet's inference to adjust the parameters of the object proposals and fix the TaskNet's error according to the domain of the input images. For this purpose, we specifically design R2SNet, a small DNN for bounding–box adjustment that, being run on the robot, ensures scalability in domain adaptation. This contribution is detailed in Section 5.1.

- For privacy preservation, we add a lightweight encoder–decoder on the robot, tasked to obfuscate the robot's perceptions before they are sent remotely for inference. We propose an innovative co–train scheme where the encoder–decoder is supervised by the TaskNet's loss with an additional *proposal selection* step. Doing this, the encoder–decoder distills the task–relevant features from images while discarding privacy–sensitive details, thus obtaining obfuscation without compromising the TaskNet's detection. This contribution is reported in Section 5.2.

5.1 Scalable Domain Adaptation in Cloud Robotics with Proposal Refinement

Consider a robotic ecosystem where multiple independent units are deployed across different environments and rely on a cloud–based general TaskNet. Assume that the robots' working

environments are initially unknown and the number of robots in the system is expected to increase by deploying new units in novel environments. The fields of Cloud Robotics [114], and more recently Fog Robotics [42], come in handy by studying inference–serving solutions that distribute, in an adaptive way, the computational and storage loads across robots and cloud services. However, the application of domain adaptation techniques over these architectures is subject to scalability issues. First, it requires full access to the cloud DNN; this is not always feasible if it is provided by an external vendor. Then, performing domain adaptation on the full model would pose significant costs; each time a new robot is deployed, a new network is to be trained, deployed, and maintained, as it cannot be shared by multiple robots after performing domain adaptation.

In this section, we propose a method for scalable domain adaptation in the aforementioned scenario, a problem at the intersection of cloud robotics and deep learning that, despite being relevant to many real–world settings, is still largely underexplored. The general scenario we consider is depicted in Figure 5.1: multiple robots need to carry out object detection from RGB images acquired in their respective environments. To such end, they rely on a general–purpose pre–trained DNN (called TaskNet), which is provided as a third–party cloud service, making it accessible exclusively through queries. More specifically, each robot sends its visual perceptions (RGB images) to the remote server to receive back a list containing the object proposals produced by the TaskNet. Each proposal is a bounding box, described by its position in the image plane (expressed with center coordinates, width, and height), the assigned object category, and the predicted confidence.

The core contribution of our approach (reported in the work of [50]) is to perform domain adaptation as an efficient downstream proposal–refinement stage, running locally on the robots. As previously introduced, this strategy is inspired by the observation that state–of–the–art object detectors typically work by generating dense sets (up to thousands) of bounding–box proposals which then undergo heuristic post–processing via confidence thresholding and non–maximum suppression [192]. Our findings indicate that a substantial portion of the performance degradation due to domain shifts can be mitigated by introducing, before such a post–processing heuristics, a proposal–refinement step adapted to the target environment. To such end, we introduce R2SNet (Section 5.1.2), a novel lightweight neural network architecture for proposal refinement that focuses on three different types of corrective actions: relabeling, rescoring, and suppression of bounding boxes. To carry out such a task, R2SNet leverages the acquired images and the geometrical features of the corresponding bounding–box proposals, and it can be run downstream and locally on the robot.

In Section 5.1.5, we evaluate this method in a real–world testbed where mobile service robots must perform real–time door–status detection as defined in Chapter 3. For service robots, this object detection task is key for navigation, but also one recognized as very much affected by domain shifts [46]. The obtained results show how our method enables scalable adaptation, effectively mitigating the performance losses due to domain shifts encountered with the general pre–trained model, all while avoiding the need for substantial computational costs in training and inference.

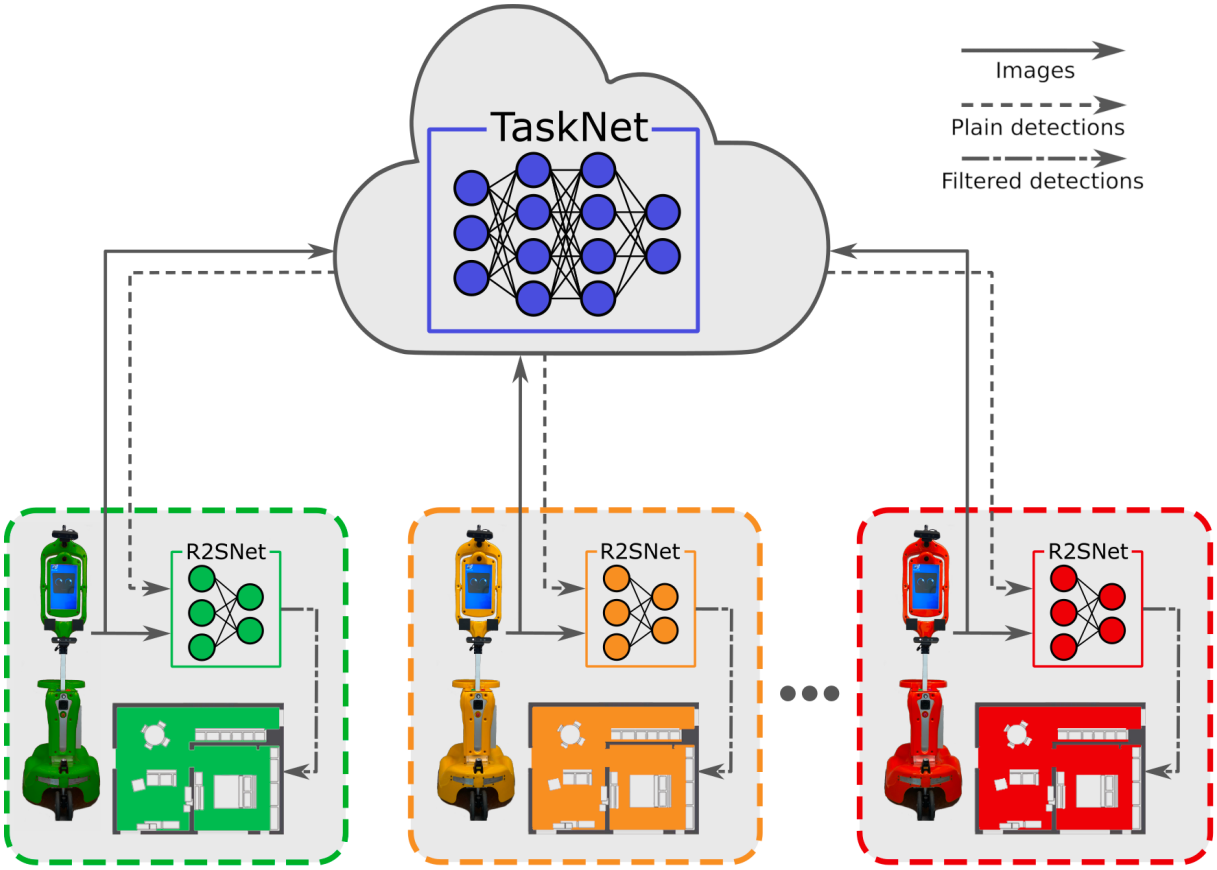


Figure 5.1: A general overview of the cloud–based scenario we consider. A fleet of robots, deployed in different indoor environments, sends their perceptions to a remote TaskNet for object detection. The TaskNet returns to the robots the dense sets of object proposals that are locally refined using our lightweight neural network (R2SNet) for domain adaptation.

5.1.1 Problem Formulation: Scalable Adaptation with Proposal Selection

Object Detection (OD) amounts to identifying the location and dimension of objects in an image. Deep learning is today the leading approach to building detectors, which are typically based on architectures that analyze the input image through different stages to ensure its comprehensive coverage [148]. Two–stage detectors (such as Faster R–CNN [21]) use a Region Proposal Network (RPN) to predict, in a first stage, proposals of bounding boxes from multi–scale image embeddings. In the second stage, such proposals are classified into object categories. Differently, one–stage models (such as YOLO [173]) directly predict object classes for a set of predefined bounding boxes called *anchors*, which uniformly cover the image with multiple scales and sizes.

Both architectures share a characteristic: they produce many overlapping bounding–box proposals, typically numbering in the thousands, which are independently scored using the image's features. To distill meaningful detections from this dense set, a heuristic two–step post–processing is commonly executed. The first step, called Non–Maximum Suppression

(NMS) [192], iteratively selects pairs of proposals whose Intersection over Union area (IoU) exceeds a threshold ρ_{IoU} and suppresses the one with the lowest confidence. In the second step, any proposal with a confidence lower than a threshold ρ_c is also discarded.

For achieving scalable adaptation, we suggest relocating the post–processing step to operate locally on each robot. This entails integrating it as a downstream module of a global cloud–based object detector we call TaskNet (see Figure 5.1) which has been configured to return raw proposals by modifying hyperparameters. Additionally, we augment this post–processing by incorporating R2SNet, a lightweight deep architecture tailored to the robot's target environment.

From an image x, we obtain from the TaskNet a set of raw proposal $\hat{Y} = \{\hat{y}\}$, with

$$\hat{y} = \left[\hat{c}_x, \hat{c}_y, \hat{w}, \hat{h}, \hat{c}, \text{hot}(\hat{o})\right]_{1\times f} \tag{5.1}$$

where $\hat{c}_x, \hat{c}_y \in [0, 1]$ are the center coordinates, $\hat{w}, \hat{h} \in [0, 1]$ represent width and height, $\hat{c}$ is the confidence, $\hat{o} \in \mathcal{O}$ is an integer indicating the object category, and $\text{hot}(\cdot)$ is its one–hot encoding (so $f = 5 + |\mathcal{O}|$). Once received by the robot, the k most confident proposals, where $k \gg O$, with O indicating the maximum number of identifiable objects in an image, are given as input to R2SNet. Before presenting its architecture, we examine the three primary types of interventions along which the network is trained and used: Relabeling, Rescoring, and Suppression (hence the acronym R2SNet). We focus on a specific object detection task, *door detection*, as it is particularly significant for this task, as discussed in the examples below. However, our considerations are general to other detection tasks for autonomous robots.

Relabeling Frequently, a TaskNet generates several overlapping proposals over the same target object; some of these proposals often have contrasting labels. Figure 5.2a shows an example of different overlapping proposals that label the same door both as closed and open. These errors are frequent when involve objects that might resemble each other (e.g., open and closed doors or chairs and armchairs). The standard post–processing based on NMS would select the proposal with the highest confidence disregarding the correctness of its object category. In our method, we improve on this by relabeling all overlapping proposals to a single category, forcing a consensus. Also, we identify isolated proposals not overlapping with any others as spurious. Based on our empirical observations, these isolated proposals often correspond to errors in object localization. Consequently, we relabel them as `background`, an additional category introduced in this stage.

Rescoring It is well–known that confidence scores may not consistently reflect the actual uncertainty, and thus the likelihood of correctness, of the proposals computed with TaskNet [193]. When a poorly localized proposal receives high confidence, NMS might erroneously reject nearby proposals that better match with the object. Conversely, if a properly localized proposal receives low confidence, the thresholding step could erroneously discard it. We observed this phenomenon in challenging instances, such as the one depicted in Figure 5.2b, where an open door is partially hidden behind a corner, often leading to errors. To address this, we correlate the confidence of each proposal to the IoU area they have with the best overlapping ground

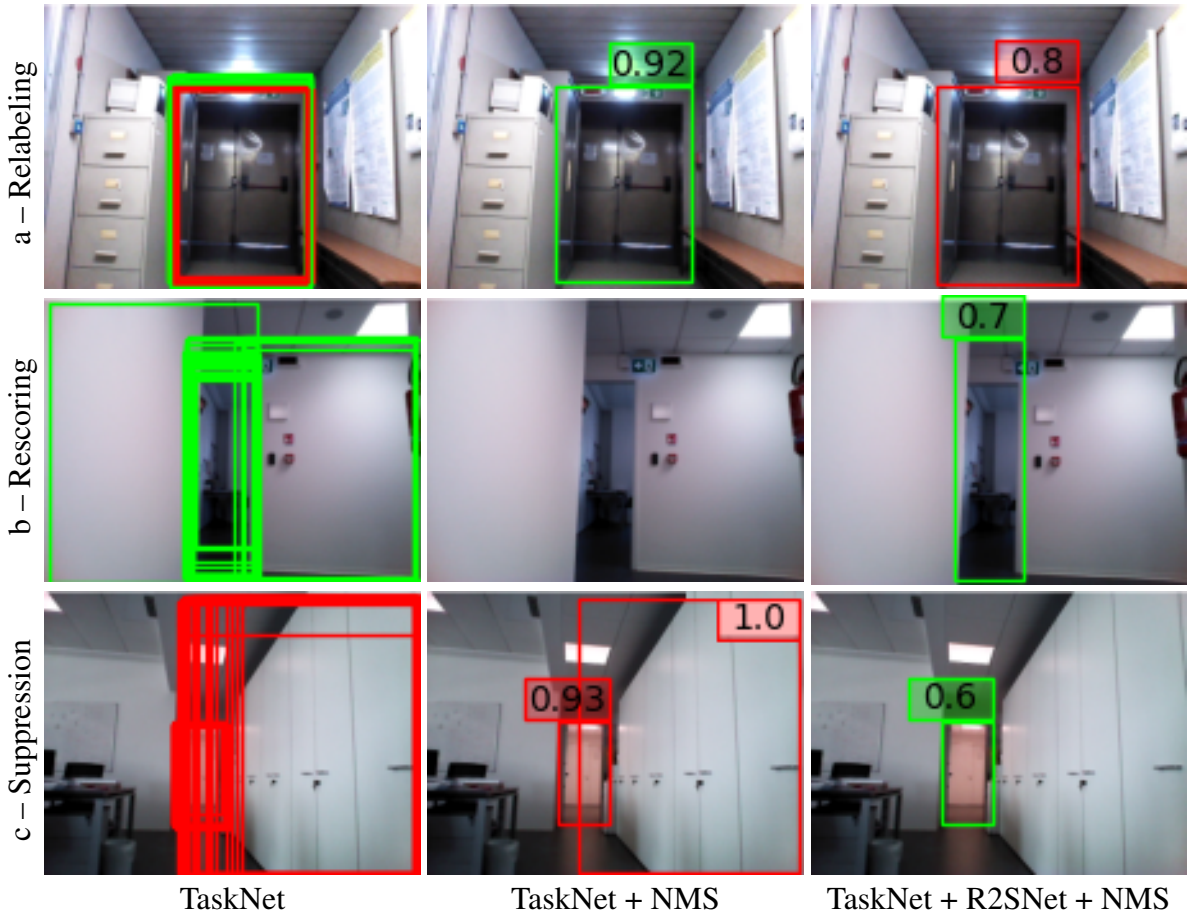


Figure 5.2: R2SNet refinements in filtering dense proposals, compared to standard post–processing. Green/red bounding boxes are open/closed doors.

truth box. In this way, the IoU threshold ρ_{IoU} becomes the only hyperparameter for the post–processing techniques.

Suppression Other frequent errors occur when dense sets of proposals are situated in parts of the image where no objects are present. This might happen because the features in these regions mimic an object category that the detector is trained to identify. For instance, Figure 5.2c highlights instances where cabinets (or windows) are misclassified as doors. While relabeling partially mitigates this issue, we introduce a suppression phase to directly address it by learning a feature embedding from the image portion of each proposal to differentiate between background areas and those containing an object.

5.1.2 R2SNet

First, given the k most confident proposals computed by the TaskNet, R2SNet extracts a matrix of Bounding–box Descriptors $BD = \left[\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k\right]_{k \times f}$ by stacking the vectors defined

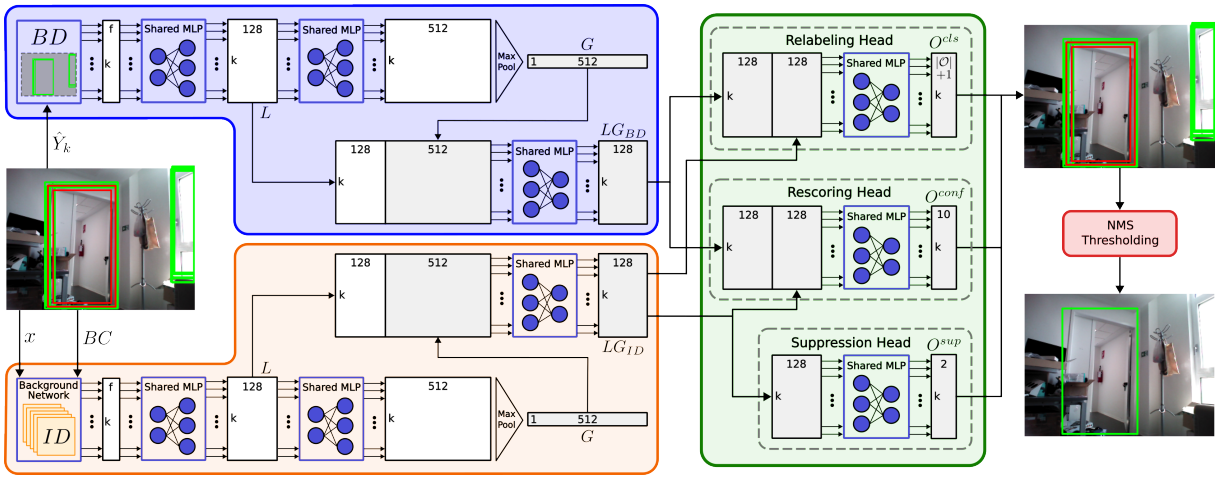


Figure 5.3: The R2SNet architecture. Batch normalization and ReLU activation functions are applied to all layers of the shared MLPs.

in Eq. 5.1. Additionally, it extracts a feature vector from the portion of the image x corresponding to each proposal, using a convolutional architecture we call BFNet (Bounding–box Feature Network, detailed in Section 5.1.3), to compute a matrix of image descriptors $ID_{[k\times 8]}$. BD and ID can be seen as projections of the k most confident bounding boxes in two distinct spaces $\mathbb{R}^f$ and $\mathbb{R}^8$, which are meant to capture their geometrical and visual features.

Given these preliminaries, R2SNet (depicted in Figure 5.3) is inspired by PointNet [194], which is designed to classify and segment dense point clouds and to be invariant to input permutation (proposals, in our setting). R2SNet processes BD and ID with two symmetric sub–networks. At first, each of them maps the input to a high–dimensional space using a Multi–Layer Perceptron (MLP) shared across the k proposal descriptors, to obtain local features $L_{[k\times l]}$. In the MLPs, the same weights are applied to each descriptor, making the size of the network fixed regardless of the number k of proposals. After this step, the local features are expanded again with another MLP and then aggregated using $\max$, to obtain a global feature vector $G_{[1\times g]}$. This last one is concatenated with each row of $L_{[k\times l]}$ and then mixed with a shared MLP, obtaining an embedding $LG_{[k\times 128]}$ that represents both local and global features of the k proposals. The outputs LG_{BD} and LG_{ID} of the two sub–networks are fed into three heads to handle the relabeling, rescoring, and suppression of the proposals.

We denote as Y the set of ground truth bounding boxes for image x where each $y \in Y$ is encoded as per Eq. 5.1 by setting $c = 1$. We define a matching rule to assign a proposal $\hat{y}$ to a ground–truth bounding box $\hat{y}^{GT} = \arg\max_{y\in Y} a_{IoU}(\hat{y}, y)$, where $a_{IoU}(\hat{y}, y)$ is the IoU area between $\hat{y}$ and y.

The relabeling head, starting from the concatenation of LG_{BD} and LG_{ID}, assigns to each proposal $\hat{y}$ the probabilities for each object class in the set $\mathcal{O} \cup \{\texttt{background}\}$, producing an output $O^{cls}_{[k\times|\mathcal{O}|+1]}$. This head is trained with the following log–loss:

$$\mathcal{L}_{cls}(O^{cls}) = -\frac{1}{k}\sum_{p=1}^{k} \log\left(O^{cls}_p\right) \cdot \text{hot}(\hat{o}_p), \tag{5.2}$$

where $(\cdot)$ is the dot product and $\hat{o}_p$ is the true class for the p-th proposal determined by our matching rule:

$$\hat{o}_p = \begin{cases} Class(\hat{y}_p^{GT}) & \text{if } a_{IoU}(\hat{y}_p^{GT}, \hat{y}_p) \geq \rho_{IoU} \\ \texttt{background} & \text{otherwise.} \end{cases}$$

The rescoring head aligns the confidence of a proposal $\hat{y}$ to its IoU area with its associated ground truth $\hat{y}^{GT}$. To achieve this, the confidence score $c \in [0, 1]$ is discretized into 10 intervals. The rescoring head is then tasked with predicting the likelihood that the confidence score falls within each of these intervals, yielding an output matrix $O^{conf}_{[k\times 10]}$. For training, we construct a target vector $v(\hat{y})$ for a proposal $\hat{y}$, whose values peak at the interval corresponding to the IoU score between $\hat{y}$ and its corresponding ground truth $\hat{y}^{GT}$, and decrease in a Gaussian–like manner on either side of the peak. In such a way, we obtain a measure of the error that increases with the distance between the predicted and true peaks. This error is adopted for the rescoring loss:

$$\mathcal{L}_{res}(O^{conf}) = \frac{1}{k}\sum_{p=1}^{k} \|O_p^{conf} - v(\hat{y}_p)\|_1. \tag{5.3}$$

The confidence assigned to each $\hat{y}_p$ is $\arg\max_j O^{conf}_{p,j}$.

Finally, the suppression head is trained with a loss obtained by adapting Eq 5.2 for binary classification between proposals that correspond to an object (those for which $\hat{o}_p \neq \texttt{background}$) and those falling on the background.

5.1.3 BFNet

BFNet, shown in Figure 5.4, guides R2SNet to identify those proposals that are wrongly placed on the background and that can be suppressed. This task is challenging when only the descriptors BD (location, confidence, and class) are used, thus image descriptors ID are needed. BFNet partitions the input image with a low–resolution grid mask $M_{[W\times H]}$. Then it extracts a feature encoding $IF_{[8\times W\times H]}$ (8 channels for each cell), which is mapped to each proposal's region of interest.

More precisely, BFNet extracts a multi–scale feature hierarchy of the input image using a CNN–based backbone with residual connections [176]. The last three embeddings are re–scaled with dimensions $[W \times H]$, $\left[\frac{W}{2} \times \frac{H}{2}\right]$, and $\left[\frac{W}{4} \times \frac{H}{4}\right]$ using adaptive average pooling layers. To aggregate features at different scales, Feature Pyramid Networks [195] (FPN) are commonly used. Differently from what is done in FPNs, to have more descriptive features, each embedding is processed by three parallel convolutional backbones and step–by–step top–down aggregated through upsampling and summation. The resulting embeddings are concatenated and mixed through convolution to generate a feature map IF.

Then, we need to obtain the portion of IF covered by each proposal. Rather than iteratively slicing IF according to each bounding box coordinates, we perform a faster parallel end–to–end mask generation process. More precisely, we use a series of MLPs that produces a binary

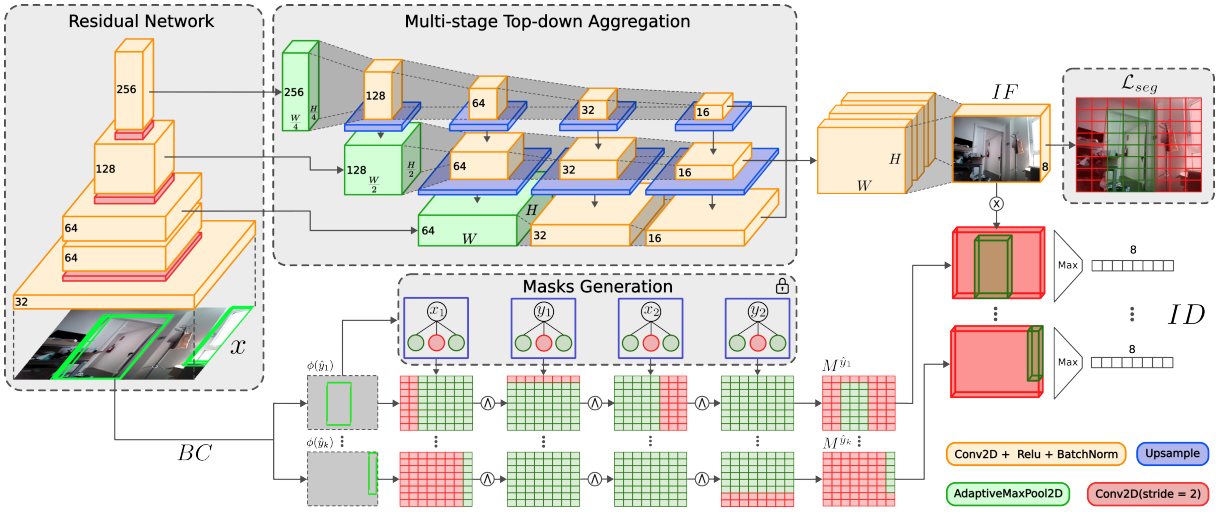


Figure 5.4: The BFNet architecture. The image features IF are obtained with an extension of the standard Feature Pyramid Network where the last three multi–scale embeddings (extracted by a hierarchy of convolutional blocks with residual connections) are stage–per–stage top–down aggregated while being processed by parallel convolutional backbones and finally stacked. IF is then multiplied with k segmentation masks inferred by four MLPs with fixed parameters that are max pooled along the mask's width and height obtaining the image descriptors $ID_{[k\times 8]}$ for the R2SNet.

mask $M^{\hat{y}}_{[W\times H]}$ for each proposal $\hat{y}$ where an element is set to 1 (0) if inside (outside) the area of $\hat{y}$. This mask is used to suppress the features of IF exceeding the bounding box's boundaries. First, BFNet receives in input a matrix $BC = [\phi(\hat{y}_1)\dots\phi(\hat{y}_k)]_{k\times 4}$, where $\phi : \mathbb{R}^f \rightarrow \mathbb{N}^4$ encodes an input proposal $\hat{y}$ to a vector $[x_0, y_0, x_1, y_1]$ containing the coordinates of the bottom–left and top–right corners in the grid mask $M_{[W\times H]}$. It then computes, for each proposal $\hat{y}$, four binary grids defined as

$$\begin{aligned} M^{x_j} &= \mathbb{1}_{\leq}\Big((-1)^j(x_j - A)\Big) \\ M^{y_j} &= \mathbb{1}_{\leq}\Big((-1)^j(y_j - B^T)\Big) \end{aligned} \tag{5.4}$$

for $j \in \{0, 1\}$, where $A = \mathrm{diag}(I_H) \times [0\dots W-1]$ and $B = \mathrm{diag}(I_W) \times [H-1\dots 0]$. We obtain the matrices using four MLPs, each comprising one input and $W \times H$ output neurons. The weights are initialized according to Eq. 5.4 and remain fixed during the training process. The mask of each proposal $\hat{y}$, obtained as

$$M^{\hat{y}}_{[W\times H]} = \bigwedge_{j=0}^{1} M^{x_j} \wedge M^{y_j},$$

is combined with the embedding IF to suppress the features outside the bounding box boundaries. The results are then compressed along the last two dimensions with a $\max$ operation, obtaining $ID_{[k\times 8]}$ that encodes the image descriptors of each proposal for R2SNet.

Before training the whole R2SNet, BFNet is pre–trained for addressing a low–resolution binary segmentation task. The image features $IF_{[8\times W\times H]}$ are convoluted into a binary grid mask

$M^{seg}_{[2\times W\times H]}$ obtained by training on this loss:

$$\mathcal{L}_{seg}(M^{seg}) = -\frac{1}{WH}\sum_{w=1}^{W}\sum_{h=1}^{H}\log\left(M^{seg}_{w,h}\right)\cdot \text{hot}\big(l_{w,h}\big), \tag{5.5}$$

where the ground truth label for each cell $l_{w,h}$ is

$$l_{w,h} = \begin{cases} 1 & \text{if } \exists\ y \in Y \mid \phi(y) \text{ contains the cell } w,h \\ 0 & \text{otherwise.} \end{cases}$$

5.1.4 Experimental Setting

Training is performed using 2 publicly available datasets (see [47, 46] for more details) for door detection in RGB images. The first one, *DeepDoors2* ($\mathcal{D}_{\texttt{DD2}}$) [152], has 3K real–world images containing doors, taken from a human perspective. The second one, $\mathcal{D}_{\texttt{G}}$, is taken in 10 environments of Gibson [35] and contains around 5K photorealistic images acquired from the viewpoint of a mobile robot. The experimental evaluation is performed on a third, real–world, dataset, $\mathcal{D}_{\texttt{real}}$, collected with our Giraff–X robot (Figure 5.1) [7]; it contains four runs collected when different indoor environments e_* (three university facilities and an apartment, see Figure 5.2) have been fully mapped. All datasets have $\mathcal{O} = \{\texttt{closed}, \texttt{open}\}$. We split each run in each environment of $\mathcal{D}_{\texttt{real}}$ in $75/25\%$ for training/testing R2SNet.

We use a Faster R–CNN [21] as TaskNet due to its widespread use; including its ResNet–50 [176] backbone, it has 41M parameters. We train the TaskNet on the full $\mathcal{D}_{\texttt{DD2}}$ and $\mathcal{D}_{\texttt{G}}$ for 60 epochs with a batch size of 4. We then deploy it for inference disabling the NMS and thresholding.

R2SNet needs to be trained from scratch on a large sample of dense proposals obtained by a TaskNet on unseen data; to do so, we augment $\mathcal{D}_{\texttt{DD2}}$ and $\mathcal{D}_{\texttt{G}}$ as in the following. For each image, alongside the ground–truth bounding boxes of doors, we must include the corresponding TaskNet proposals. To do this, generating these proposals using images unseen by TaskNet during its training is key, ruling out the use of the reference TaskNet trained on the full $\mathcal{D}_{\texttt{DD2}}$ and $\mathcal{D}_{\texttt{G}}$. To overcome this, we train 11 versions of Faster R–CNN, dividing the datasets into 11 segments (the first includes $\mathcal{D}_{\texttt{DD2}}$, the others contain images from one of the 10 environments of $\mathcal{D}_{\texttt{G}}$), using a leave–one–out approach. Each TaskNet, trained on 10 segments, is used to extract proposals from the remaining, unseen, one. R2SNet is thus pre–trained with the dataset obtained by combining the 11 segments. We run a first pass training only BFNet using $\mathcal{L}_{seg}$, then in a second pass we train the whole architecture using $\mathcal{L}_{R2S} = \mathcal{L}_{cls} + \mathcal{L}_{conf} + \mathcal{L}_{sup}$ (both passes with 60 epochs and batch size of 16, $k = 30$, and $W = H = 32$).

The pre–trained R2SNet is adapted with data specific to its deployment environment using examples from $\mathcal{D}_{\texttt{real}}$. We label this customized version as $\text{R2S}^{\#proposals}_{\#data}$, where the superscript denotes the number of proposals k per training example, and the subscript indicates the percentage of data utilized relative to the total available data in $\mathcal{D}_{\texttt{real}}$ from the deployment environment.

Exp.	e_1				e_2				e_3				e_4				$\overline{e}$			
	mAP↑	**TP↑**	**FP↓**	**BFD↓**	**mAP↑**	**TP↑**	**FP↓**	**BFD↓**	**mAP↑**	**TP↑**	**FP↓**	**BFD↓**	**mAP↑**	**TP↑**	**FP↓**	**BFD↓**	**mAP↑**	**TP↑**	**FP↓**	**BFD↓**
TaskNet	33	41%	10%	13%	27	33%	5%	18%	13	24%	7%	34%	47	47%	6%	14%	30	36%	7%	20%
$R2S^{30}_{25}$	39	50%	7%	11%	30	36%	4%	10%	20	26%	7%	12%	58	64%	4%	11%	37	44%	6%	11%
$R2S^{30}_{50}$	40	49%	8%	9%	35	40%	6%	6%	22	29%	7%	10%	59	63%	5%	9%	39	45%	6%	9%
$R2S^{30}_{75}$	49	54%	5%	7%	35	39%	6%	6%	26	31%	8%	10%	62	62%	1%	5%	43	46%	5%	7%

Table 5.1: R2SNet performance evaluation when trained with an increasing amount of data. $\overline{e}$ is the average.

In particular, we assessed the impact of using 10, 50, and 100 proposals, alongside 25%, 50%, and 75% of the data (which correspond to $\approx$ 80, 160, and 240 training images, respectively). The remaining 25% of the data are used for testing the R2SNet, using the TaskNet as a baseline. We apply a NMS step to the TaskNet and TaskNet+R2SNet proposals, choosing a set of conservative thresholds: $\rho_{IoU} = 50\%$ and $\rho_c = 75\%$ for the former, and $\rho_{IoU} = \rho_c = 50\%$ for the latter. Note that the domain adaptation of R2SNet is performed once, using the data acquired during the initial deployment of the robot, and is used later on for the whole operative life of the robot.

The performance metrics are the mean Average Precision [32] overall object categories (mAP) with 3 additional indicators (formally defined in Section 3.4.4) measuring the percentage of doors detected with the correct (wrong) label denoted as TP (FP) and the rate between the background detections (i.e., false positives detections placed on the background) and the total number of ground truth objects, named *BFD*. We release the implementation of R2SNet and the code to run the experiments in a publicly available repository[1].

5.1.5 Evaluation of R2SNet

Figure 5.5 shows the performance of $R2S^{k}_{75}$ varying the number of proposals k. A value $k \in \{30, 50\}$ improves the mAP of $\approx 45\%$, while also reducing both FP and BFD, showing the effectiveness of our R2SNet. Despite the filtering becomes challenging due to the high number of noisy low–confidence proposals, a higher value of $k = 100$ results in a further increase of the mAP, at the expense of higher BFD (that are still close to those obtained by the TaskNet). Interestingly, also a low value of $k = 10$ has some positive aspects; while it does not improve the mAP, it still reduces the FP and BFD. Figure 5.5 shows the benefits brought by the $R2S^{100}_{75}$ refinement to the TaskNet raw proposals.

Another interesting remark, confirming the solidity of our approach, comes from evaluating the results when a different and increasing number of samples obtained in the robot target environment are used to train R2SNet, as shown in Table 5.1. For this test, we used $k = 30$. Even with a few examples, $R2S^{30}_{25}$ increases both the mAP and TP of $\approx 20\%$ while halving the percentage of BFD. Of course, exposing the R2SNet to a higher number of examples, as in $R2S^{30}_{50-75}$, increases performances, but our findings suggest that a few examples are enough for R2SNet to improve the TaskNet performance on the target environment. We emphasize that

[1] https://aislab.di.unimi.it/research/r2snet

R2SNet improves performance only when used in the same environment in which it was trained. This is because it not only has access to images from that environment but, more importantly, it learns how to fix the specific errors computed by the TaskNet in a particular scene. This means that, when the environment changes, a domain shift occurs not only in the input images but also in the TaskNet's predictions, limiting R2SNet's effectiveness.

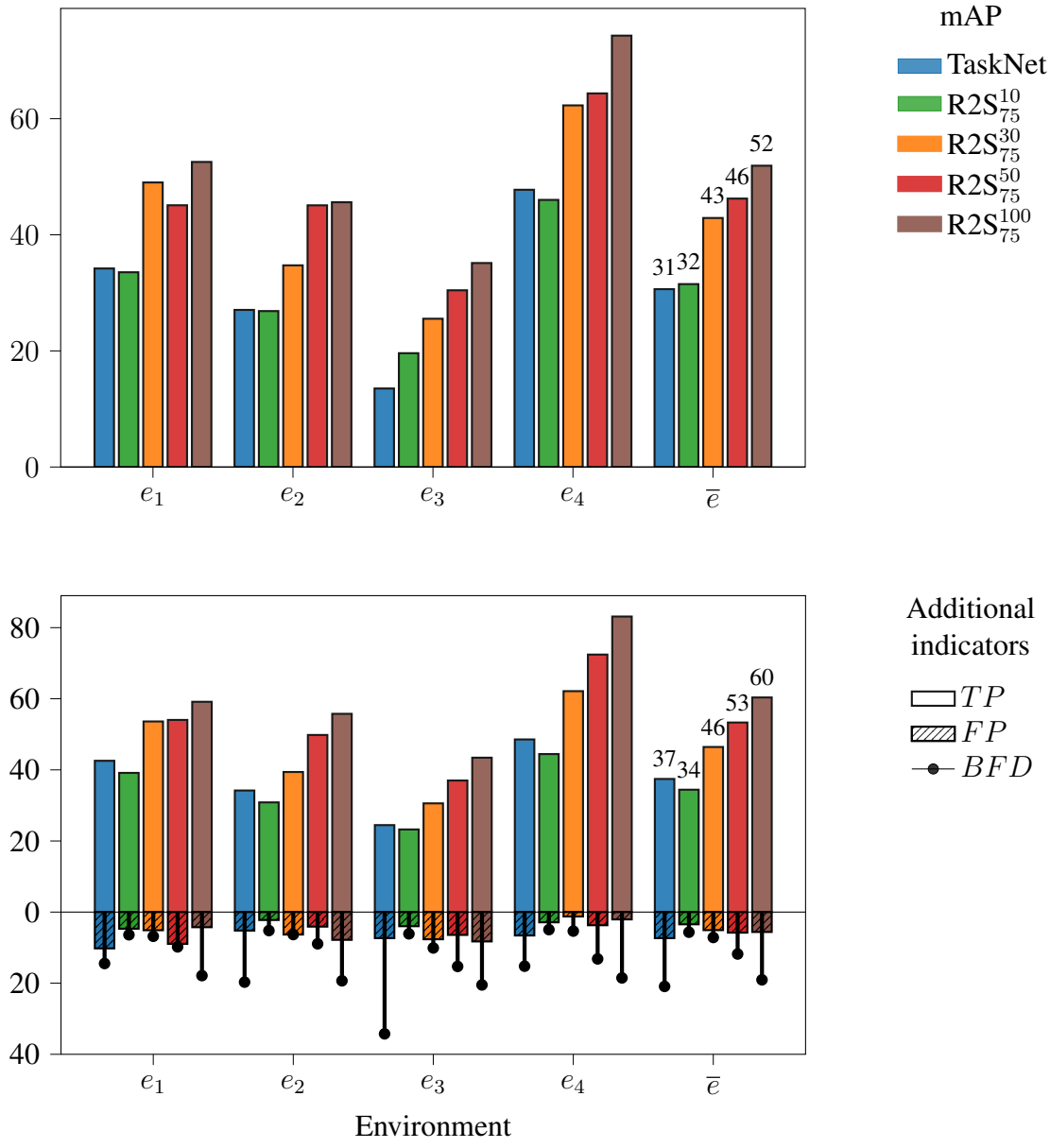


Figure 5.5: R2S$^k_{75}$ performance when varying k expressed with the mAP (top row) and the additional indicators (bottom row).

To further evaluate the contribution of each network's component, we conduct an ablation study where the relabeling, rescoring, and refinement heads are incrementally activated. To do this, we use as a reference R2S$^{100}_{75}$, which has the best performances. Table 5.2 reports the results averaged over the 4 environments. From this analysis, it can be seen how the relabeling head alone ensures a significant mAP and TP improvement while reducing the FP and BFD.

The use of the rescoring head similarly improves the mAP and TP, without reducing FP and BFDs. The best overall performance is shown when all three heads are used. However, the suppression head has less impact on the performance than the other two heads; still, it is needed for training, as disabling it increases the BFD of the 3%.

				$\overline{e}$		
Rel.	**Res.**	**Sup.**	**mAP↑**	**TP↑**	**FP↓**	**BFD↓**
			34	44%	10%	35%
✓			44	48%	4%	6%
	✓		41	54%	15%	34%
		✓	37	43%	9%	14%
✓	✓		52	61%	6%	20%
✓		✓	44	47%	4%	5%
	✓	✓	41	53%	15%	31%
✓	✓	✓	52	60%	6%	19%

Table 5.2: Ablation study results.

Our R2SNet has 8M parameters. To assess its computational demands for inference, we installed it on an edge device, an NVIDIA Jetson TX2, mounted on the Giraff–X robot. On this hardware, commonly available in service robots, R2SNet demonstrates remarkable efficiency, processing images at a rate of 16.7 Hz on the GPU and 2.6 Hz on the CPU. As a reference, TaskNet processes at significantly lower frequencies of 1.1 Hz and 0.06 Hz, respectively, on the same platforms. The R2SNet training with a RTX 3090 GPU took a few minutes. These results show how our method can be used in real–time on a mobile robot, illustrating R2SNet's capability for efficient deployment in robots that utilize edge devices, even those without a GPU.

5.2 Privacy Preservation in Cloud Robotics with Proposal Selection

Considering a setup where a service robot deployed in human–centric environment leverages a remote TaskNet for object detection, adding data protection to the data sent to the cloud becomes fundamental to preserve the privacy of the stakeholders. The work of [196] defines privacy preservation for data–streams acquired by service robots with two key requirements: minimizing the risk of exposing human–interpretable images of the robot's operational environment and limiting the information that enables their reconstruction by a third party. End–to–end encryption, a widely used technique to prevent man–in–the–middle attacks, is ineffective in our scenario because the remote inference engine must access the decrypted plain data (both for training and inference), leaving no technological safeguards to potential leaks or misuses by a *curious* untrustworthy end–point. A renowned example of this critical concern is the case of [45], where privacy–sensitive data, depicting owners of vacuum cleaner robots in their homes, were transmitted to the company's servers using secure channels but later leaked by employees.

Against this threat, we need to fulfill two competing objectives. On the one hand, the robot's data stream transmitted from a trusted environment should be obfuscated to resemble noise to a curious observer and be robust against adversarial post–processing for reconstruction. On the other hand, obfuscated images should preserve enough meaningful features to minimize the task–performance gap between a trusted service (that can access the plain images) and a curious one working with obfuscated data. Furthermore, privacy solutions should operate online on the robot, with computational and latency constraints.

The limited computational capabilities and energy–preserving requirements of mobile robots make it difficult to run in real–time large object–detection models directly on mobile robots. Because of this, we propose to achieve privacy by adding a lightweight encoder–decoder that can be run by the robot for adding privacy to its perceptions. More specifically, we consider the distributed architecture of Figure 5.6. A robot collects images using its onboard camera that are processed by a lightweight encoder–decoder *obfuscator*, and then sent to a remotely–deployed TaskNet for object–detection. This network is an off–the–shelf proposal–based model trained on plain images that performs the task. The obfuscator hence safeguards the data against curious attackers with access to the cloud.

Recent works have pointed out how neural networks can be trained to distill features that contain only the necessary information to effectively solve a given task, without revealing any sensible information once the image is reconstructed [146]. A relevant approach is described in [144], wherein it is shown that an encoder–decoder, when co–trained with a TaskNet, extracts a bottleneck representation of the perceptual data that, although not interpretable by humans, can be effectively used for image classification. In this dissertation, we demonstrate that the same approach cannot be applied to object detection, since this task requires a more feature–rich representation. This is due to the task's complexity: multiple targets, of unknown size and location, have to be identified in the same image, forcing the encoder–decoder to preserve redundant

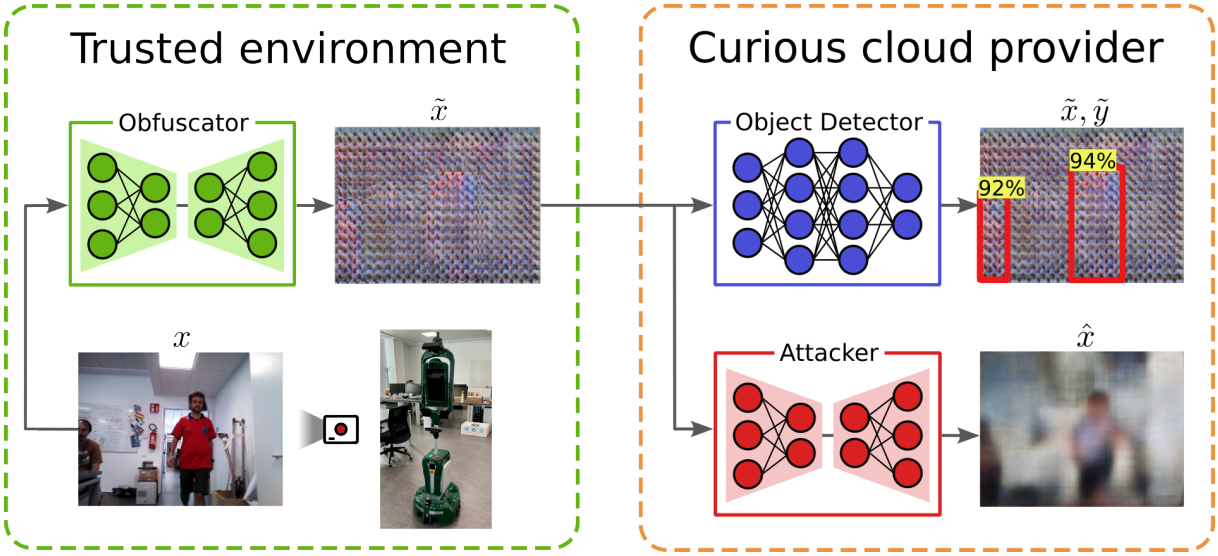


Figure 5.6: A general overview of the cloud robotic scenario we consider. A robot collects visual perceptions (x) from its working environment (bottom left) that are obfuscated by a local encoder–decoder, obtaining privatized images $\tilde{x}$ (top left). Its goal is to remove sensitive information from images while maintaining the key features for executing a perception task. Then, the obfuscated robot's perceptions are sent remotely to a curious cloud provider running a TaskNet (pre–trained on plain images) that performs object detection (top right). In our case, the TaskNet extracts a set of bounding boxes $\tilde{Y}$ containing people. In the meanwhile, a malicious actor intercepts the obfuscated images $\tilde{x}$ to reconstruct the original perception x with an adversarial encoder–decoder (bottom right).

details, thus compromising privacy. To overcome this, we propose a novel co–training scheme that systematically selects subsets of proposals over which the loss is computed at each back-propagation pass. We refer to this approach as *weak loss via proposal selection*, and we show that it enables the obfuscator to distill task–relevant features, effectively balancing the competing desiderata of object detection and privacy. Without assuming any specific motion model, our approach obfuscates the data stream, thus providing a general–purpose detection–oriented privatization method that is specifically designed for real–world robotic scenarios, deployable in heterogeneous contexts and with different embodiments.

The obfuscated data stream provided by our method appears indistinguishable to noise from a human observer and, more importantly, is robust to adversarial post–processing carried out by an attacker aiming to reconstruct the original data, matching the definition of privacy outlined in [196]. To assess this last property, we test the robustness of our method against a highly capable malicious actor that, operating within the cloud provider (e.g., an employee), performs a Model Inversion Attack (MIA) against collaborative inference [147] aiming at reconstructing the original perceptions from their obfuscated version (see Figure 5.6). Our contributions (described in the work of [51]) are summarized as follows:

- we theoretically analyze the properties needed by an obfuscator encoder–decoder to obtain both detection and privacy, overcoming the limitations of [144] (Section 5.2.2);
- we propose a co–training scheme based on a weak loss with proposal selection to extract

only the necessary features for object detection while discarding sensitive information (Section 5.2.4);

- we investigate the trade–off between object detection performance and privacy implementing an attack model which aims at reconstructing the original data (see Figure 5.6);

- we provide an extensive experimental evaluation of our method relying on publicly available datasets (Section 5.2.7) for object detection and performing real–word experiments using a robotic platform (Giraff–X [7]) deployed on the field (Section 5.2.9).

5.2.1 Problem Formulation: Proposals Selection for Privacy Preservation

In the scenario we consider (Figure 5.6), the robot collects images $x \in \mathbb{R}^D$, with $D = W \times H \times C$ and TaskNet is assumed to be a differentiable object detector following an architecture that generates a dense set of object proposals (labeled bounding boxes), which are then filtered to produce a small set of final predictions [148]. This type of architecture is widely adopted by many prominent and popular detectors [21, 173]. We assume that the TaskNet's parameters θ_t are given and fixed (for example, because the service provider has trained the DNN from any of the publicly available pre–trained object detectors [148]). We denote by $Y = f(x; \theta_t)$ its dense set of unfiltered proposals for image x, where each proposal is a labeled bounding box with some level of confidence.

Before being transmitted to the cloud, x is processed by a differentiable encoder–decoder DNN which produces an obfuscated version $\tilde{x} \in \mathbb{R}^D$, formally defined as $\tilde{x} = q(x; \theta_e, \theta_d)$, where θ_e and θ_d are the DNN's learnable parameters. We indicate with $z = e(x; \theta_e)$, the low–dimensional latent representation in $\mathbb{R}^Z$, with $Z \leq D$ being the *bottleneck* dimension, computed by the encoder. The final obfuscated representation is denoted as $\tilde{x} = d(z; \theta_d)$. TaskNet computes objects' locations and classes for the image x by inference on its obfuscated representation $\tilde{x}$. Against this background, we introduce the problem of training our encoder–decoder module so that TaskNet achieves comparable performance when working on $\tilde{x}$ as it does when receiving as input the original image x. At the same time, we seek representations $\tilde{x}$ from which sensible information cannot be extracted. Note that θ_t has been obtained by training TaskNet on plain images x.

The first objective can be achieved by minimizing a *task loss* $\mathcal{L}_{task}(Y, \tilde{Y})$. This loss is proportional to the difference between the dense sets of proposals returned by TaskNet with plain and obfuscated data, respectively. It rewards obfuscated images that preserve task–related features. For the second objective, we introduce a *forward privacy loss*, $\mathcal{L}^{\gg}_{priv}$. Specifically, $\mathcal{L}^{\gg}_{priv}(x, \tilde{x})$ measures the weakness of the representation $\tilde{x}$ against the threat posed by our scenario. This loss is meant to promote representations that suppress features related to sensible information. We use the term "forward" to remark its relationship with the process of privatizing an image x into its obfuscation $\tilde{x}$ that is forwarded to an untrusted environment. Expressing

a trade–off between the two objectives with a parameter λ, the learning problem can be formalized as follows.

Problem 1 (Co–training for Perception and Privacy)**.** *Given $\lambda \in [0, 1]$ and a differentiable pre–trained TaskNet $f(\cdot; \theta_t)$, find encoder–decoder bottleneck dimension Z and parameters (θ_e, θ_d) such that, for data distribution $\mathcal{D}$, the following is minimized:*

$$\mathop{\mathbb{E}}_{x \sim \mathcal{D}} \lambda \mathcal{L}_{task}(Y, \tilde{Y}) + (1 - \lambda)\mathcal{L}^{\gg}_{priv}(x, \tilde{x}).$$

Then, we introduce an *attacker*, explicitly formalizing the process of executing our scenario's threat, which is inspired by MIA [147]. The attacker takes as input the image representation $\tilde{x}$ and aims to compute a new representation $\hat{x}$ where sensible information is restored to some degree. In line with the MIA paradigm, we adopt an encoder–decoder architecture for the attacker too, denoted by $(\hat{\theta}_e, \hat{\theta}_d)$. To train such a model, we introduce a *backward privacy loss* $\mathcal{L}^{\ll}_{priv}$, designed to promote the attacker's ability to restore sensible information of x, working backward from its reconstruction $\tilde{x}$, as follows[2].

Problem 2 (Privacy Violation)**.** *Given a DNN $q(\cdot; \theta_e, \theta_d)$, find the encoder–decoder bottleneck dimension $\hat{Z}$ and parameters $(\hat{\theta}_e, \hat{\theta}_d)$ such that, for the data distribution $\mathcal{D}$, the following is minimized:* $\mathop{\mathbb{E}}_{x \sim \mathcal{D}} \mathcal{L}^{\ll}_{priv}(x, \hat{x})$.

Formally defining $\mathcal{L}^{\gg}_{priv}$ and $\mathcal{L}^{\ll}_{priv}$ is no easy task [128]. Given its high variability, subjectivity, and domain dependence, obfuscating or restoring sensible information in an image is difficult to model and encode in a loss function. It could be argued that $\mathcal{L}^{\gg}_{priv}$ and $\mathcal{L}^{\ll}_{priv}$ are related to reconstruction. Thus, $\mathcal{L}^{\gg}_{priv}$ can be defined as inversely proportional to a reconstruction loss, while $\mathcal{L}^{\ll}_{priv}$ can be defined as directly proportional. Following the last implication, we set $\mathcal{L}^{\ll}_{priv} = \mathcal{L}_{rec}$ (where $\mathcal{L}_{rec}$ is a reconstruction loss) as we assume that the attacker of Problem 2 seeks reconstruction because it implies the restoration of any sensible information (the same rationale behind MIA). However, implementing $\mathcal{L}^{\gg}_{priv}$ of Problem 1 as an inverse reconstruction loss is likely to be ineffective: not reconstructing clearly does not always imply adding privacy.

An alternative approach is to set $\lambda = 1$, thereby neglecting the privacy objective represented by $\mathcal{L}^{\gg}_{priv}$, while decreasing the bottleneck dimension Z of the encoder–decoder. The rationale is that it is possible to learn representations which, due to high inner compression, are not human–interpretable once reconstructed, yet can still recall a similar task performance. This approach has been shown to be effective in [144] when TaskNet is a classifier. In the following, we prove that this method cannot be used for object detection tasks providing a theoretical analysis (Section 5.2.2) and confirming it with empirical evidence (Section 5.2.3).

Our solution We introduce a novel approach to address Problem 1 without the need to explicitly define $\mathcal{L}^{\gg}_{priv}$. Our method is based on the concepts of *weak task loss* and *proposal selection*

[2]We use the symbol $\gg$ to indicate the action of adding privacy (forward, i.e., from the data source), while $\ll$ to compromise privacy (backward, i.e., from the attacker).

(Section 5.2.4). Specifically, we train the encoder–decoder using a weakened version of the TaskNet's loss, computed over a limited subset of proposals at each iteration. This encourages the encoder–decoder to retain only essential task–relevant features, while discarding irrelevant details (thereby enhancing privacy) without requiring a reduction in the bottleneck size. The proposal selection strategy (detailed in Section 5.2.5) selects a small number of proposals for each target, with the goal of promoting correct predictions while reducing errors.

5.2.2 Detection vs. Privacy: Theoretical Analysis

In [144], the authors propose a task–oriented framework for data compression in robotic applications. The method co–trains a variational autoencoder (VAE) [197] with a TaskNet for classification. Then, the encoder of the VAE is deployed on the robot to produce compact and task–specific representations of visual perceptions that are transmitted to a remote server over a low–bandwidth connection. On the server side, the VAE's decoder recovers the original dimensionality of the compressed perceptions (while maintaining task–related features) that are finally fed into TaskNet for inference. Such a work demonstrates that decreasing the encoder–decoder bottleneck to a dimension Z that is substantially lower than the input's dimensionality yields representations that are not human–interpretable (and so private) since reconstruction features are not needed (so not learned). The authors provide theoretical support for their findings by studying a linearized model of their framework. This kind of analysis is common in the literature (see [198] for a recent example) as studying such simplified models can offer formal (hence explainable) insights into the behavior of more complex ones. In this section, we adopt this approach to examine the trade–offs that characterize our scenario and justify our proposed method, which we devise and evaluate in the subsequent sections.

Following the same methodology of [144], we model TaskNet as a proposal–based detector, aligning with one of the mainstream approaches in deep neural network architectures for object detection [148]. Proposal–based architectures extract a meaningful representation of the input using a convolutional backbone (such as a Residual Network, ResNet [176]) which is then processed by a dense set of overlapping object–proposal heads with different dimensions and scales that, being uniformly distributed across the image, are responsible of detecting objects. This approach is implemented by the off–the–shelf one–stage and two–stage detectors with a key difference in how proposals are handled. In one–stage detectors like YOLO [173], the object proposals are specified by means of hyperparameters, while, in two–stage detectors such as Faster R–CNN [21], the location and size of object proposals are dynamically computed during inference. In defining a generic proposal–based TaskNet for the linear setting, we explicitly model this structure encompassing a backbone and a set of heads.

Definition 1 (Linear TaskNet for Object Detection)**.** *In the linear setting, we consider a TaskNet for Object Detection composed of (i) a backbone defined as a full–rank matrix* $K \in \mathbb{R}^{S \times D}$ *with* $S \leq D$ *and (ii) a set of object proposals* $\mathcal{H} = \{H_1, \ldots, H_n\}$*, in which each* $H_i \in \mathbb{R}^{|\mathcal{C}| \times S}$ *is a classification head that maps elements of* Kx *to a tuple of scores for object categories in* $\mathcal{C} = \{c_1, c_2, \ldots, \texttt{background}\}$*, with* $|\mathcal{C}| \leq S$*.*

In the above definition, element (i) models the fact that the TaskNet compresses inputs $x \in \mathbb{R}^D$ using the backbone K to produce a meaningful representation of the image, where the features are informative and not redundant. Element (ii) describes that the compressed input $Kx \in \mathbb{R}^S$ is processed by multiple classification heads to detect objects. The rationale behind these two properties is that to perform well, object detectors must retain relevant image features and have a large and dense set of classification heads covering multiple regions of the image. In real object detectors, $|\mathcal{H}|$ is typically in the order of thousands to ensure an accurate and comprehensive object localization [148]. In our derivations, similar to [144], we represent the image x as a one–dimensional vector. We assume each head H_i focuses on a consecutive segment of Kx, ranging from the u-th to the v-th component, where $u, v \in [1, S]$ and $u \leq v$. These assumptions are without loss of generality with respect to the 3D case and facilitate the formal derivations we provide in the following. Since each H_i is a generic matrix, this can be represented by assuming all columns of H_i are zero except for those ranging from column u to column v. In this setup, the task loss for co–training can be defined as:

$$\mathcal{L}_{task}(x) = \sum_{i=1}^{n} \left\| H_i K x - H_i K B A x \right\|_2^2 \tag{5.6}$$

where $A \in \mathbb{R}^{Z \times D}$ and $B \in \mathbb{R}^{D \times Z}$ are the encoder–decoder matrices with bottleneck dimension Z. Intuitively, this loss function quantifies the difference between TaskNet's performance on plain inputs x and their corresponding obfuscated versions $\tilde{x}$. Within this framework, we can derive the following result.

Theorem 1 (Linear Detection–Aware Compression)**.** *Consider a TaskNet as per Definition 1, with a bottleneck $K \in \mathbb{R}^{S \times D}$, and encoder–decoder matrices $A \in \mathbb{R}^{Z \times D}, B \in \mathbb{R}^{D \times Z}$. Then, when setting the bottleneck dimension $Z = rank(K) = S$ it is possible to achieve $\mathcal{L}_{task}(x) = 0$ for any x and for any $\mathcal{H}$. Moreover, when setting $Z < rank(K)$ there always exist $\mathcal{H}$ and x such that $\mathcal{L}_{task}(x) > 0$.*

Proof. With a bottleneck dimension $Z = rank(K)$ we can apply the same technique of [144]. Consider the compact singular value decomposition (SVD) of $K = U\Sigma V^\top$, where $U \in \mathbb{R}^{S \times S}$ unitary, $\Sigma \in \mathbb{R}^{S \times S}$, and $V^\top \in \mathbb{R}^{S \times D}$ semi–unitary. Solving Eq. 5.6 for zero is feasible for any input x and any set $\mathcal{H}$ by assigning $A^\top = B = V$. Given that $V^\top V = I_S$, it follows that

$$\begin{aligned}
\mathcal{L}_{task}(x) &= \sum_{i=1}^{n} \left\| H_i \Big(\underbrace{U\Sigma V^\top}_{K} (x - \underbrace{VV^\top}_{BA} x) \Big) \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i \Big(U\Sigma (V^\top x - \underbrace{V^\top V}_{I_S} V^\top x) \Big) \right\|_2^2 = 0.
\end{aligned}$$

Now, we show that for any $Z < rank(K)$ there exists $\mathcal{H}, x$ such that $\mathcal{L}_{task}(x) > 0$. Suppose to use an encoder–decoder given by full–rank matrices $\overline{A} \in \mathbb{R}^{Z \times D}$ and $\overline{B} \in \mathbb{R}^{D \times Z}$ with $Z < rank(K)$. Now consider an $\mathcal{H}$ where each head has the form of $H_i = \begin{bmatrix} 0 & I_{|\mathcal{C}|} & 0 \end{bmatrix} \in \mathbb{R}^{|\mathcal{C}| \times S}$. Assume that the identity matrix $I_{|\mathcal{C}|}$ is shifted differently in different heads, such that

$\left[H_1^\top, \ldots, H_n^\top\right]^\top$ has no all–zeros columns. At an intuitive level, such an $\mathcal{H}$ represents a straightforward set of heads that ensures no feature computed by the backbone is entirely disregarded. If the set of heads has this property, then the only way to achieve zero task loss is to have $Kx = K\overline{BA}x$ for any input $x \in \mathbb{R}^D$. This means we must have $K = K\overline{BA}$, but this is impossible since $rank(K\overline{BA}) \leq Z < rank(K)$. □

The above result provides an optimal solution to Problem 1 when $\lambda = 1$ through a factorization of the backbone K. In this solution, regardless of the classification heads and the input x, the dimensionality of the encoder–decoder bottleneck must be at least as large as the rank of K.

In the following, we highlight an important relation between the rank of the encoder–decoder matrices and the reconstruction quality that, as discussed in Section 5.2, can be mapped to the objective of the attacker defined in Problem 2. To do this, we define a dataset as a full–rank matrix $X \in \mathbb{R}^{D \times N}$ containing $N \geq D$ examples. With a slight notation overload, we reformulate the task loss from Eq. 5.6 so that it applies to a dataset X rather than a single instance x and exploit the Cauchy–Schwarz inequality to derive an upper bound on it. Let X_c denote the c–th column of X, representing the c–th example in the dataset, and use $\|\cdot\|_F$ to indicate the Frobenius norm, then

$$\begin{aligned}
\mathcal{L}_{task}(X) &= \sum_{i=1}^{n}\sum_{c=1}^{N} \left\|H_i(KX_c - KBAX_c)\right\|_2^2 \\
&= \sum_{i=1}^{n} \left\|H_i(KX - KBAX)\right\|_F^2 \\
&= \sum_{i=1}^{n} \left\|H_i(K - KBA)X\right\|_F^2 \\
&\leq \sum_{i=1}^{n} \left\|H_i\right\|_F^2 \left\|K - KBA\right\|_F^2 \left\|X\right\|_F^2.
\end{aligned}$$

This formulation provides a way to understand changes in task loss through the use of encoder–decoder matrices with ranks $Z < S$. The upper bound indicates that a method to limit $\mathcal{L}_{task}$, without taking into account both the classification heads $\mathcal{H}$ and the dataset X (which, in our setting, we don't control), involves approximating the backbone K with low–rank encoder–decoder matrices. This can be accomplished by solving the following optimization problem:

$$\begin{aligned}
&\arg\min_{A,B} \left\|K - KBA\right\|_F^2 \qquad \text{subject to} \\
&rank(A) = rank(B) = Z < S.
\end{aligned} \tag{5.7}$$

For the Eckart–Young theorem [199], the solution of (5.7) is to have $KBA = U_Z \Sigma_Z V_Z^\top$, which is the Z–truncated SVD of K. In this context, setting $B = V_Z$ and $A = V_Z^\top$ is an optimal

solution since

$$\begin{aligned} KBA &= U\Sigma V_S^\top V_Z V_Z^\top \\ &= \begin{bmatrix} U_Z & U_{S-Z} \end{bmatrix} \begin{bmatrix} \Sigma_Z & 0 \\ 0 & \Sigma_{S-Z} \end{bmatrix} \begin{bmatrix} V_Z^\top \\ V_{S-Z}^\top \end{bmatrix} V_Z V_Z^\top \\ &= \begin{bmatrix} U_Z\Sigma_Z & U_{S-Z}\Sigma_{S-Z} \end{bmatrix} \begin{bmatrix} I_Z \\ 0 \end{bmatrix} V_Z^\top \\ &= U_Z\Sigma_Z V_Z^\top. \end{aligned}$$

In other words, for a TaskNet's backbone K with rank S, optimizing the task loss in a heads– and dataset–independent manner using encoder–decoder matrices with rank $Z < S$ means setting $A = V_Z^\top$ and $B = V_Z$, where V_Z is obtained by removing the last $S - Z$ columns of V_S extracted according to Theorem 1. We exploit these derivations to provide a result that shows how, in our setting, diminishing the encoder–decoder's rank through bottleneck compression offers privacy to the robot's perceptions by hindering the attacker's reconstruction capability.

Theorem 2 (Compression vs. Reconstruction)**.** *Let $X \in \mathbb{R}^{D\times N}$ be a full–rank dataset matrix with $N \geq D$ examples and consider a TaskNet with backbone K according to Definition 1. Consider a reconstruction loss $\mathcal{L}_{rec}^{Z} = \|X - V_Z V_Z^\top X\|_F^2$ where encoder–decoder matrices V_Z and $V_Z^\top$ are obtained by removing the last $S - Z$ columns from V_S obtained from K according to Theorem 1. Then, for any $Z_1 < Z_2 \leq S$, we have $\mathcal{L}_{rec}^{S}(X) \leq \mathcal{L}_{rec}^{Z_2}(X) < \mathcal{L}_{rec}^{Z_1}(X)$ for all X.*

Proof. The matrix $V_S \in \mathbb{R}^{D\times S}$ derived from Theorem 1 consists of Z orthogonal columns. For $Z < S$, the matrix V_Z is constructed by removing columns from V_S according to the resolution of (5.7). Define $V_D = \begin{bmatrix} V_S & V_{D-S} \end{bmatrix}$, where V_{D-S} provides $D - S$ additional columns to ensure V_D is unitary. This is achieved by composing V_{D-S} with columns that, together with those of V_S, create an orthogonal basis for the vector space $\mathbb{R}^D$. For any $Z < D$, we can write

$$V_Z V_Z^\top = V_D \begin{bmatrix} I_Z & 0 \\ 0 & 0 \end{bmatrix} V_D^\top.$$

Thus, we can expand the reconstruction loss as

$$\begin{aligned}
\mathcal{L}^{Z}_{rec}(X) &= \left\|X - V_Z V_Z^\top X\right\|_F^2 \\
&= \left\|I_D X - V_D \begin{bmatrix} I_Z & 0 \\ 0 & 0 \end{bmatrix} V_D^\top X\right\|_F^2 \\
&= \left\|V_D V_D^\top X - V_D \begin{bmatrix} I_Z & 0 \\ 0 & 0 \end{bmatrix} V_D^\top X\right\|_F^2 \\
&= \left\|V_D (V_D^\top X - \begin{bmatrix} I_Z & 0 \\ 0 & 0 \end{bmatrix} V_D^\top X)\right\|_F^2 \\
&= \left\|V_D^\top X - \begin{bmatrix} I_Z & 0 \\ 0 & 0 \end{bmatrix} V_D^\top X\right\|_F^2 \\
&= \left\|[V_D^\top X]_{Z:D}\right\|_F^2,
\end{aligned}$$

where $[V_D^\top X]_{Z:D} \in \mathbb{R}^{(D-Z)\times N}$ is the sub–matrix of $V_D^\top X$ containing the last $D - Z$ rows. With the reconstruction loss expressed in this form, it is easy to see that

$$Z_1 < Z_2 \implies \underbrace{\left\|[V_D^\top X]_{Z_2:D}\right\|_F^2}_{\mathcal{L}^{Z_2}_{rec}(X)} < \underbrace{\left\|[V^\top X]_{Z_1:D}\right\|_F^2}_{\mathcal{L}^{Z_1}_{rec}(X)},$$

which can be generalized as:

$$\mathcal{L}^{S}_{rec}(X) < \mathcal{L}^{S-1}_{rec}(X) < \ldots < \mathcal{L}^{1}_{rec}(X) < \left\|X\right\|_F^2. \qquad \square$$

The above results shed some light on a significant trade–off of our robotic scenario in Figure 5.6. In this context, we have the ability to design the robot's perception module, yet we do not have control over the TaskNet responsible for object detection, as we depend on an external provider. TaskNet runs in the cloud, and the robot can interface with it by uploading sensory input. Our aim is to diminish an attacker's ability to reconstruct the robot's sensory data while maintaining the performance level TaskNet achieves with direct images. We introduce an obfuscator with an encoder–decoder to process the sensory inputs, and we co–train it using the TaskNet supervision to obtain the same detection performance as using plain images, i.e., with the objective of minimizing Eq. 5.6. However, preserving TaskNet's performance requires the obfuscator to avoid excessive compression of the robot's perceptions. If its bottleneck matches the TaskNet's backbone, full performance recovery is achievable (Theorem 1, first part). However, choosing a smaller bottleneck might lead to performance degradation. This degradation may occur if the TaskNet's components or the inputs themselves (which in this scenario are beyond our control) are such that reducing the bottleneck increases the task loss (Theorem 1, second part). In contrast, obstructing an attacker's ability to reconstruct the robot's sensory data from the obfuscated images could be facilitated by opting for a smaller bottleneck in the obfuscator (Theorem 2). Thus, safeguarding the robot's perceptual data using obfuscation while simultaneously achieving task performance with obfuscated images drives the design of the

robot's perception processing in conflicting directions. In the following section, we empirically validate this intuition within a realistic setting.

5.2.3 Detection vs. Privacy: Empirical Evaluation

We compare the setting of [144], which uses a variational autoencoder (VAE) [197], against a setup in which we replace the VAE with a convolutional encoder–decoder. The key difference between the two is the bottleneck compression: while the VAE encodes input data in a flattened inner representation with dimension Z, the encoder–decoder maps images with dimension $D = (W \times H \times 3)$ to a 2D inner space with $Z = \left(\frac{W}{16} \times \frac{H}{16} \times 1024\right)$, resulting in an adaptive and substantially larger bottleneck dimension. We compare the encoder–decoder with multiple variational autoencoder configurations (named VAE_Z) changing the bottleneck dimension Z on the task of people detection. We use the encoder–decoder architecture as the main setting, named Encoder–Decoder Obfuscator (EDO). For this evaluation, we consider the task of people detection and implement the TaskNet with a Faster R–CNN [21] re–trained on the `person` object category of COCO dataset [24]. We perform testing using COCO and Pascal VOC 2012 [200] validation sets. Object detection performances are quantified using the AP and AP_{50} of COCO [24]. AP is the area under the precision/recall curve averaged over multiple IoU thresholds for the correct detections (0.5 to 0.95 in steps of 0.05), while AP_{50} considers only the case where the IoU threshold is 0.5. (The specifics of this preliminary evaluation's experimental setup mirror those of our primary campaign and are thoroughly outlined in Section 5.2.6.)

Setting	COCO		Pascal VOC	
	AP ↑	AP_{50} ↑	AP ↑	AP_{50} ↑
TaskNet (TN)	59	87	55	86
EDO	47	76	39	68
VAE_{256}	9	24	5	16
VAE_{128}	8	22	5	15
VAE_{64}	7	21	4	14

Table 5.3: Comparison of people detection performance as a function of the encoder–decoder bottleneck dimension. VAE_Z refers to the setup from [144], which uses a variational autoencoder. EDO represents our approach, where the VAE is replaced by an encoder–decoder obfuscator. TN (TaskNet on plain images) denotes the upper–bound reference performance. For both EDO and VAE, results are shown in decreasing order of bottleneck size. Performance on in–distribution (COCO) and out–of–distribution (Pascal VOC) data highlights how reducing the bottleneck dimension progressively degrades Tasknet's reference performance.

The results reported in Table 5.3 show that the VAE substantially degrades the TaskNet's detection performance. TaskNet's performance on plain images, labeled TN, serves as the upper bound reference value. When the images from the COCO benchmark are used, VAE–based architectures have significantly lower performance with respect to TN. More precisely, AP and AP_{50} are only $\approx 15\%$ and 27% of the values reached by TaskNet with plain images (TN).

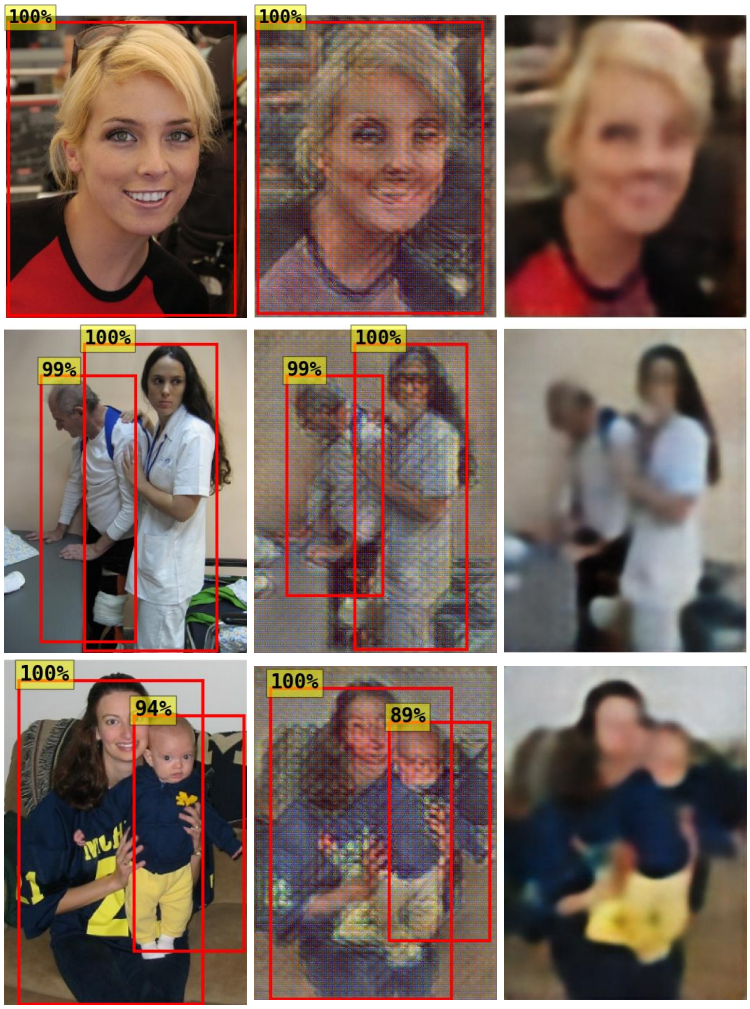


Figure 5.7: (Left) Original images from the Pascal VOC dataset, (middle) obfuscated outputs from our EDO, and (right) reconstructions generated by the attacker. Red bounding boxes indicate TaskNet detections y and $\tilde{y}$ with confidence scores $\sigma(y), \sigma(\tilde{y}) \geq 0.75$, filtered using Non–Maximum Suppression (NMS) with an IoU threshold of 0.5. The examples show that when the bottleneck size is close to the input dimensionality, the obfuscation retains substantial visual information.

This is caused by the limited bottleneck dimension Z of the VAE, which is unable to encode a dense and uniformly distributed feature representation for precise and dense object detection. Conversely, as suggested by Theorem 1, a larger bottleneck allows us to fill the gap between the performance of the TaskNet with plain (TN) and obfuscated images (EDO). When the encoder–decoder is used in place of the VAE, higher performances are retained. TaskNet, when using images $\tilde{x}$ obfuscated by the EDO, achieves approximately 80% of the AP and 87% of the AP_{50} obtained by TN on the COCO dataset. These results hold true even in out–of–distribution settings. Specifically, on the Pascal VOC dataset, the EDO reaches about 70% of TN's reference value of AP and 79% of its AP_{50}, while VAE has a significant drop in performance, retaining only about 9% and 18% of AP and AP_{50}, respectively.

These findings demonstrate that, in accordance with Theorem 1, achieving optimal task performance requires a bottleneck size that is (large and) nearly equal to the size of the TaskNet's backbone. Conversely, to obfuscate sensitive information, in line with Theorem 2, additional compression of the encoder–decoder bottleneck is required. This is also confirmed by the examples in Figure 5.7: the size of the EDO's bottleneck, being near the input dimensionality, preserves privacy–related features compromising identity preservation in both $\tilde{x}$ and $\hat{x}$.

5.2.4 Weak Loss via Proposal Selection

The above analysis indicates that within the conventional learning framework for object detectors utilizing dense proposal sets, modifying the size of the encoder–decoder bottleneck alone does not suffice to jointly achieve effective perception and privacy.

Theorem 1 demonstrates that setting the encoder–decoder considering only the backbone K is a convenient solution when the bottleneck dimension $Z = rank(K)$. However, setting $Z < rank(K)$ may lead to degradation of task performance. At the same time, Theorem 2 suggests that lowering Z results in a bad reconstruction by the attacker (that is, good privacy). Considering Theorem 1, we propose a solution designed to jointly minimize task loss and achieve privacy. Our method extends the learning framework considered in our analysis by adding a proposal selection mechanism, limiting the proposals employed by the learning algorithm during the computation of the loss function. The concept of proposal selection is well known, but with popular off–the–shelf object detectors it generally functions as a downstream filtering process applied during inference time to improve bounding box accuracy. Common techniques include confidence thresholding and non–maximum suppression [148].

We formulate our approach in linear settings considering the framework introduced in Section 5.2. Our goal is to find sets of classification heads $\mathcal{H}$ that satisfy the following property.

Property 1. *Given a TaskNet according to Definition 1,*

$$\exists A \in \mathbb{R}^{Z \times D}, B \in \mathbb{R}^{D \times Z} \text{ with } Z < S \mid \forall x \; \mathcal{L}_{task}(x) = 0.$$

The following result characterizes any TaskNet with Property 1.

Theorem 3 (Compression for Detection and Privacy)**.** *Given a TaskNet with backbone $K \in \mathbb{R}^{S \times D}$ and $\mathcal{H} = \{H_1, \ldots, H_n\}$ as per Definition 1, and considering the matrix $M \in \mathbb{R}^{n|\mathcal{C}| \times D}$ defined as*

$$M \coloneqq \begin{bmatrix} H_1 \\ \vdots \\ H_n \end{bmatrix} \cdot K = \begin{bmatrix} H_1 K \\ \vdots \\ H_n K \end{bmatrix},$$

we have that the TaskNet has Property 1 $\iff$ $rank(M) < S$.

Proof. We separately prove the two directions of ($\iff$).

($\Longleftarrow$) : By construction, we have that

$$H_i K = I_i M \in \mathbb{R}^{|\mathcal{C}| \times D},$$

where $I_i = \begin{bmatrix} 0 & I_{|\mathcal{C}|} & 0 \end{bmatrix} \in \mathbb{R}^{|\mathcal{C}| \times n|\mathcal{C}|}$ in which the identity matrix $I_{|\mathcal{C}|}$ is positioned from column $(i-1)|\mathcal{C}|+1$ to column $i|\mathcal{C}|$ to extract the i^{th} head from M. Now, considering the compact SVD of the matrix $M = U\Sigma V^T$ where $U \in \mathbb{R}^{n|\mathcal{C}| \times Z}$, $\Sigma \in \mathbb{R}^{Z \times Z}$ and $V^T \in \mathbb{R}^{Z \times D}$, with $rank(M) = Z < S$ and setting $\overline{B} = V$ and $\overline{A} = V^T$ produces zero task loss for any input x:

$$\begin{aligned}
\mathcal{L}_{task}(x) &= \sum_{i=1}^{n} \left\| H_i K x - H_i K \overline{BA} x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - I_i M \overline{BA} x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - I_i M V V^\top x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - I_i U \Sigma \underbrace{V^\top V}_{I_Z} V^\top x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - I_i U \Sigma V^\top x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - I_i M x \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| H_i K x - H_i K x \right\|_2^2 = 0.
\end{aligned}$$

($\Longrightarrow$) : Notice that, in general, $rank(M) \leq S$. Thus, suppose by contradiction that $rank(M) = S$ and there exists a couple of matrices $\overline{A} \in \mathbb{R}^{Z \times D}, \overline{B} \in \mathbb{R}^{D \times Z}$ with rank $Z < S$ such that $\mathcal{L}_{task}(x) = 0, \forall x \in \mathbb{R}^D$. This implies that $H_i K = H_i K \overline{BA}$ for all i, that is true iff $M = M\overline{BA}$. This is in contradiction since $rank(M\overline{BA}) \leq Z < S = rank(M)$. □

According to Theorem 3, we can guarantee Property 1 by reducing the rank of the matrix M obtained by stacking the product between each head and the backbone K. With this framework, the following corollaries report two easy methods to select sets of classification heads to ensure $rank(M) < S = rank(K)$, thus allowing the reduction of the bottleneck dimension to enhance privacy maintaining $\mathcal{L}_{task} = 0$.

Corollary 1 (Weak Set)**.** *$\forall \mathcal{H}$ such that $|\mathcal{H}| < \frac{S}{|\mathcal{C}|}$ the TaskNet satisfies Property 1.*

Proof. Given that $M \in \mathbb{R}^{n|\mathcal{C}| \times S}$ and $|\mathcal{H}| = n < \frac{S}{|\mathcal{C}|}$, then $rank(M) \leq n|\mathcal{C}| < S$. For Theorem 3, the TaskNet satisfies Property 1. □

Corollary 2 (Partial Set)**.** *$\forall \mathcal{H} = \{H_1, \ldots, H_n\}$ for which $\exists\, j \in \{1, \ldots, S\} \mid \forall H_i \in \mathcal{H}$ the j^{th} column is 0 the TaskNet satisfies Property 1.*

Proof. If the j^{th} column of all heads is set to 0 the rank of the matrix obtained by stacking all the $H \in \mathcal{H}$ cannot exceed $S-1$. Given this, we can argue that

$$rank(M) = rank\left(\begin{bmatrix} H_1 \\ \vdots \\ H_n \end{bmatrix} \cdot K\right) \leq S-1,$$

which implies that the TaskNet satisfies Property 1 for Theorem 3. □

Corollaries 1 and 2 devise two classes of sets of classification heads to ensure Property 1: *Weak* and *Partial*. The former suggests to lower the cardinality of $\mathcal{H}$ at a value less than the upper bound $\frac{S}{|\mathcal{C}|}$ determined by the backbone compression (S) and the number of object categories, while the latter focuses on ignoring a sub–portion of the backbone and, consequently, one or more features. In real scenarios involving deep learning–based detectors, limiting the number of heads as suggested by Corollary 1 to balance both detection and privacy is a promising solution as proposal–based detectors use a large number of heads densely distributed across the input [148]. On the contrary, the approach outlined in Corollary 2 is impractical as it prevents the detector from finding objects in those input areas that lack proposal coverage.

5.2.5 Co–training with Weak Loss

Following these intuitions, we propose a co–training scheme to obtain both detection and privacy. It consists of training an encoder–decoder optimized by a "weakened" task loss calculated using a (very) limited but meaningful subset of proposals. As suggested by Corollary 1, this approach forces the encoder–decoder to extract from the input x only the essential features to activate a small but targeted subset of TaskNet proposals, while discarding those features exploitable by potential attackers for reconstructing the original x from its obfuscated version $\tilde{x}$.

Our method is detailed in Algorithm 4. First, the encoder–decoder parameters are randomly initialized (line 1). Then for a given number of iterations, it randomly samples an example (x, Y) from the training set $\mathcal{D}$ (line 3), where x is the image and Y the set of ground truth bounding boxes. The next step aims at computing the set of unfiltered bounding boxes $\tilde{Y}$, which are computed by the TaskNet using the obfuscated version of the image x (line 4). The encoder–decoder parameters are updated by backpropagating the same loss used for training TaskNet (line 6), but computed over a reduced set of selected bounding–box proposals (line 5). The TaskNet's loss is thus "weakened" with respect to its original counterpart, which integrated all the large and dense set of proposals. The TaskNet leverages thousands of proposals with high overlap and low confidence from which the most promising are selected using heuristic algorithms like Non–Maximum Suppression and thresholding [148]. Proposal selection (defined from line 8) operates on a single example, extracting two types of proposals: *negative* and *positive*. The procedure, takes as arguments the ground truth bounding boxes (Y_{GT}), the dense proposal set computed by TaskNet (Y_t), and the number of required positive (p) and neg-

Algorithm 4 Co–Training with Proposal Selection

Input:

- $f(\,\cdot\,;\theta_t)$: a fixed and pre–trained object detector
- $q(\,\cdot\,;\theta_e^0,\theta_d^0)$: an encoder–decoder with random weights
- $\mathcal{D}=\{(x_i,Y_i)\}_{i=1}^{|D|}$: training dataset
- N_{iter}: the number of training rounds
- p,n: the number of positive and negative proposals
- $\bar{\rho}$: the IoU threshold

Output: the trained parameters θ_e^N,θ_d^N

1: $\theta_e^0,\theta_d^0 \leftarrow RandInit()$
2: **for** $\tau \leftarrow 0$ **to** N_{iter} **do**
3: $\quad(x,Y)\sim\mathcal{D}$
4: $\quad\tilde{Y}=f(q(x;\theta_e^\tau,\theta_d^\tau);\theta_t)$
5: $\quad\tilde{Y}^{\text{SEL}}=\textsc{SelectProposals}(Y,\tilde{Y},p,n)$
6: $\quad\theta_e^{\tau+1},\theta_d^{\tau+1}\leftarrow\textsc{BackProp}\big(\mathcal{L}_{\textit{TaskNet}}(Y,\tilde{Y}^{\text{SEL}}),\theta_e^\tau,\theta_d^\tau,\theta_t\big)$
7: **end for**
8: **procedure** $\textsc{SelectProposals}(Y_{GT},Y_t,p,n)$
9: $\quad S\leftarrow\emptyset$
10: $\quad$**for all** $g\in Y_{GT}$ **do**
11: $\quad\quad P\leftarrow\{t\in Y_t\mid\arg\max_{y\in Y_{GT}}\rho(t,y)=g\}$
12: $\quad\quad P^{\text{SEL}}\leftarrow\{t_1\dots t_p\mid t_i\in P,\rho(t_i,g)\geq\rho(t_j,g),\forall j\geq i\}$
13: $\quad\quad N\leftarrow\{t\in Y_t\setminus P\mid\arg\max_{y\in Y_{GT}}\rho(t,y)=g,\rho(t,g)<\bar{\rho}\}$
14: $\quad\quad N^{\text{SEL}}\leftarrow\{t_1,\dots,t_n\mid t_i\in N,\sigma(t_i)\geq\sigma(t_j),\forall j\geq i\}$
15: $\quad\quad S\leftarrow S\cup P^{\text{SEL}}\cup N^{\text{SEL}}$
16: $\quad$**end for**
17: $\quad$**return** S
18: **end procedure**

ative (n) proposals. Call $\rho(t,g)$ the intersection–over–union area (IoU) between a TaskNet's proposal t and a ground–truth bounding box $g\in Y_{GT}$. Also, denote with $\sigma(t)$ the confidence that TaskNet assigns to proposal t. A proposal t matches $g\in Y_{GT}$ if $\rho(t,g)$ is larger than that with any other $g'\neq g$. For $g\in Y_{GT}$, we define the positive proposals as those matching with g (line 11) and we select the first p in terms of IoU with g (line 12). Negative proposals, instead, are obtained with the same matching rule, but after ruling out the positive ones and imposing an upper bound $\bar{\rho}$ on the IoU (line 13). The algorithm selects the first n in terms of confidence (line 14). The rationale is that positive and negative proposals should configure as a "sparse", yet task–relevant, set on which evaluating the loss of a GT g. Positive proposals represent likely correct predictions, these are essential to learn the task. Negative proposals represent likely wrong ones, falling outside of the object region with high confidence. Note that our method can be run offline in combination with any off–the–shelf object detector working with dense proposals, such as YOLO [173], since it modifies only the number of bounding boxes used by the loss function.

Figure 5.8 reports a visual example of the proposal selection procedure in Algorithm 4 with $p=n=1$ and (IoU threshold) $\bar{\rho}=0.5$. At first, an image x is processed by the EDO and

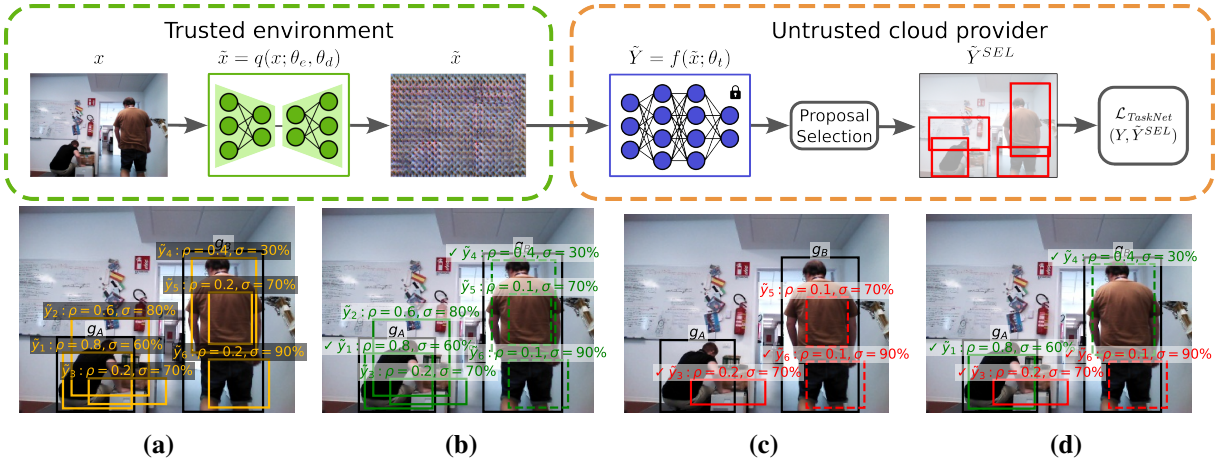


Figure 5.8: Our co–training scheme for detection and privacy detailed in Algorithm 4. (First row) Given an image x and the ground truth bounding boxes Y, the encoder–decoder obfuscator is trained using the loss function of the TaskNet (whose weights are frozen) calculated between Y and a subset ($\tilde{Y}^{SEL}$) of the proposals produced by the TaskNet on obfuscated images $\tilde{x}$. (Second row) An example of the proposal selection procedure of Algorithm 4 applied on a perception acquired by our Giraff–X [7]. (a) The GT bounding boxes g_A and g_B (in black) and some proposals from the TaskNet's dense set (in orange). Note that, for visualization purposes, we depict only a few proposals among the thousands produced by the TaskNet, many of which overlap and have low confidence and IoU. (b) Positive proposals matched with g_A (in green) and g_B (in dashed green); the selected proposals are marked with a ✓. (c) Negative proposals associated with g_A (in red) and g_B (in dashed red) with $\bar{\rho} = 0.5$. (d) The final $\tilde{Y}^{\text{SEL}}$ when $p = n = 1$.

then fed to TaskNet which produces a dense set of proposals $\tilde{Y}$. Figure 5.8a depicts the ground truths (g_A and g_B) of x and some relevant proposals from $\tilde{Y}$. Then, our algorithm matches each proposal $\tilde{y} \in \tilde{Y}$ with the ground truth g that obtains the largest IoU area $\rho(\tilde{y}, g)$ with respect to all the others $g' \neq g$ (line 11). This can be seen in Figure 5.8b, where the proposals matched with g_A (g_B) are depicted with a continuous (dashed) line. After matching, our algorithm selects, for each ground truth g, the best p proposal in terms of IoU area with their corresponding g (line 12), that are then removed from further computation (line 13). In our example, y_1 and y_4 are selected as positive proposals for g_A and g_B, respectively (see Figure 5.8b). From the remaining bounding boxes, our algorithm defines the negative proposals for each g (shown in Figure 5.8c) using the same matching rule but considering only those with a poor overlap with g, specifically $\rho(\tilde{y}, g) < \bar{\rho}$ (line 13). For example, $\tilde{y}_2$ is discarded as its IoU area with g_A exceeds $\bar{\rho}$. Note that the proposals marked as positive in the previous step are not considered in this phase, even if their intersection with a ground truth is below the threshold (like y_4 for which $\rho(\tilde{y}_4, g_B) < \bar{\rho}$). The computation proceeds by choosing the most confident n negative predictions for each ground truth (line 14). Figure 5.8d shows the final set of selected proposals $\tilde{Y}^{\text{SEL}} = \{\tilde{y}_1, \tilde{y}_3, \tilde{y}_4, \tilde{y}_6\}$.

5.2.6 Experimental Setting

The primary evaluation of our approach focuses on a highly privacy–sensitive object–detection task concerning service robots, specifically *people detection*. We implemented the related TaskNet with a standard approach that a third–party cloud provider can use: fine–tuning a stable off–the–shelf object detector using a publicly–available dataset for people detection. Following this, we use Faster R–CNN [21] (composed by ResNet–50 [176] as backbone) which is then fine–tuned using the (plain) $\approx$ 64k images from COCO 2017 dataset [24] that contain the object category `person`. Note that any proposal–based OD method could be used instead.

The encoder–decoder obfuscator (EDO) is implemented with 4 convolutional and de–convolutional blocks, that are composed of a pair of convolution layers (with 3 as kernel size and ReLU as activation function) followed by max pooling (for the former) and upscaling (for the latter). To simulate the considered scenario, where service robots need to obfuscate their visual perceptions for remote inference, we train our EDO using only a subset of the COCO images containing `person`, specifically those collected indoors. To extract this subset, we select images with at least a `person` along with another object category commonly found indoors. Examples are `table` or `refrigerator`; the full list is reported in Table 5.4. Note that this sample selection procedure maintains the proportions between the train and validation splits of COCO, reducing them from 64k to 11k and from 2.5k to 500 images, respectively. Training is carried out using our co–training scheme with weak loss of Algorithm 4 for different amounts of positive p and negative n proposals; specifically, we consider $p \in \{1, 2, 3, 4\}$ and $n \in \{0, 1, 2, 3, 4\}$.

`bench`	`bird`	`cat`
`dog`	`backpack`	`umbrella`
`handbag`	`tie`	`suitcase`
`bottle`	`wine glass`	`cup`
`fork`	`knife`	`spoon`
`bowl`	`banana`	`apple`
`sandwich`	`orange`	`broccoli`
`carrot`	`hot dog`	`pizza`
`donut`	`cake`	`chair`
`couch`	`potted plant`	`bed`
`dining table`	`toilet`	`tv`
`laptop`	`mouse`	`remote keyboard`
`cell phone`	`microwave oven`	`toaster`
`sink`	`refrigerator`	`book`
`clock`	`vase`	`scissors`
`teddy bear`	`hair drier`	`toothbrush`

Table 5.4: Categories from COCO 2017 [24] used to identify images likely acquired in indoor environments. The images labeled with these categories have been used to train our EDO architecture.

Attack model The attack model of Problem 2 is implemented using the same architecture as EDO. The attacker is tasked to reconstruct the original images from those that have been

obfuscated with our method. We follow the Model Inversion Attack (MIA) approach described in [143]; thus, training is performed with Mean Absolute Error (MAE), that is also the natural implementation of the general–purpose reconstruction loss defined in our theoretical framework (see Theorem 2). We used the same hyperparameters of the EDO, but we increased the number of epochs (from 50 to 80) to ensure the best attacker–reconstruction performance. Note that we train, for each EDO, a different attacker using its obfuscated images obtained from the same training dataset, thus relying on the most powerful MIA according to [147, 143].

To further assess the privacy–preserving capabilities of our method, we introduced an enhanced attacker model trained with the edge–centric (EC) loss function from [143], which improves reconstruction power by considering not only individual pixel differences (as in MAE), but also local pixel surroundings. Following the insights of [143], this enhanced loss function is strictly related to privacy as it promotes the reconstruction of sensitive fine–grained details, such as facial features or small textures. The final loss function of this improved MIA is defined as:

$$\|x - \hat{x}\|_1 + \beta\|S_h * x - S_h * \hat{x}\|_1 + \beta\|S_v * x - S_v * \hat{x}\|_1, \tag{5.8}$$

where S_h and S_v are the horizontal and vertical Sobel kernels while $(*)$ is the convolution operator. As in [143], β is set equal to 5. We test this configuration with $p = n \in \{1, 2, 3, 4\}$ and we report the results in Section 5.2.8.

We compare our approach for privacy preservation with a state–of–the–art baseline derived from [144], where we replaced, following the considerations detailed in the previous section, the VAE with the larger EDO trained with Algorithm 4 without performing proposal selection, so using all the TaskNet's proposals (label ALL). As a reference, we report the upper bound of the performance achieved by the TaskNet on plain images (label TN). Note how, differently from the assumptions made in our reference scenario, in the TN setup the cloud provider must be trusted.

Testing is performed on the COCO validation split, filtered for indoor examples ($\approx$ 500 images) as described above. Furthermore, we validate our approach in out–of–distribution settings using the validation split of Pascal VOC 2012 [200] dataset ($\approx$ 2k images acquired indoors and outdoors). Differently from testing on COCO, we here do not perform the filtering procedure in order to further challenge and assess the robustness of our method.

To assess generalizability across various privacy–demanding tasks, we conduct an additional evaluation campaign focused on the multi–class task of *vehicle detection*. Specifically, we run the experiments described above with `bicycle`, `airplane`, `bus`, and `train` as object categories. We train our framework with the COCO dataset following the same procedure used for people detection but considering only two proposal configurations, $p = n \in \{2, 3\}$. Also in this case, the performances are evaluated on the validation splits of COCO and the out–of–distribution instances from Pascal VOC.

The full details on the hyperparameters adopted for training the aforementioned modules (TaskNet, EDO, and the attackers) in the two considered tasks are reported in Table 5.5. Additionally, we enrich training with data augmentation by implementing random horizontal flip (with a probability of 0.5) and random resize in which the images are rescaled (main-

taining the aspect ratio) setting the length of the smallest dimension to each of the values in $\{256, 288, 320, 352, 384, 416\}$. Our implementation is based on PyTorch and, since the code makes use of random calls in different steps (for instance in line 1 of Algorithm 4), we ensure full reproducibility by fixing the random seeds in every random call, notably in those performed by `torch`, `os`, `numpy`, and `random` libraries. Our code, along with all other low–level implementation details, is made accessible in a publicly available repository[3].

	People detection			Vehicle detection		
	TaskNet	EDO	Attacker	TaskNet	EDO	Attacker
Epochs	10	50	80	10	50	80
Batch Size	8	4	4	2	2	2
Optimizer	SGD	SGD	SGD	SGD	SGD	SGD
Learning Rate	1e-3	5e-4	5e-4	4e-4	3e-4	3e-4
Weight Decay	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4
Nesterov Momentum	0.9	0.9	0.9	0.9	0.9	0.9
Scheduler	StepLR	RLROnP	RLROnP	StepLR	RLROnP	RLROnP
Step Size / Gamma	3 / 0.1	–	–	3 / 0.1	–	–
Patience / Factor	–	2 / 0.5	4 / 0.5	–	2 / 0.5	4 / 0.5

Table 5.5: Hyperparameters used to train TaskNet, EDO, and the attacker's encoder–decoder in the experiments reported.

We complement the above empirical evaluation by testing our method on real robotic platforms (see Section 5.2.9). To achieve this, we deploy our approach using Giraff–X, a service autonomous robot that features a camera with a resolution of 256×256 pixels. This platform, depicted in Figure 5.6 and detailed in [7], has been widely used in human–centric assisted living environments where addressing privacy concerns is crucial but largely neglected. We gathered a stream of images as the robot autonomously navigated through our university building, where people moved and walked by freely. We sampled the robot's perceptions at 1 Hz, resulting in approximately 250 examples, and we annotated all instances of people appearing in the images. This dataset was exclusively used for testing purposes. Additionally, Section 5.2.10 evaluates the computational demands of our implementation by measuring the inference time required on low–powered hardware typically utilized in mobile robot configurations. We carried out assessments on Giraff–X, which is fitted with an NVIDIA Jetson TX2 (GPU), and also conducted tests using a TurtleBot3 equipped with a Raspberry PI 4 (CPU).

Performance Metrics The evaluation of performance in both tasks is conducted using the standard Average Precision (AP) and AP_{50} metrics from COCO [24]. The AP and AP_{50} are customary metrics that offer a reliable assessment of object detection performance across varying parameters.

Apart from these AP–based metrics (commonly used in object detection), we incorporate two additional performance metrics that are more relevant to the deployment of the object detection module in our robotic setting (defined in Section 3.4.4): True Positive (TP) and Background False Detection (BFD) rates. True positive (TP) is defined as the rate at which ground truth bounding boxes are accurately paired with at least one prediction. A pair is considered a

[3] https://aislab.di.unimi.it/research/privacyweakloss/

match when both the predicted and ground truth (GT) labels are the same and their Intersection over Union (IoU) area is greater than a given threshold ρ_{IoU}. On the other hand, Background False Detection is calculated as the proportion of predictions, normalized by the total GTs, that end up in the background (i.e., having an IoU area with all GTs below the threshold ρ_{IoU}). In this analysis, we focus only on the predictions with the highest confidence, selecting those predictions for which the probability of the predicted object category exceeds a threshold ρ_c. To maintain a conservative assessment, we have set $\rho_{IoU} = \rho_c = 75\%$.

The potential privacy breach by the attacker is assessed utilizing the Multi–Scale Structural Similarity Index Measure (MS–SSIM) as defined in [201], with reconstructed images $\hat{x}$ serving as inputs. This metric evaluates the perceptual quality of an image after undergoing a degradation process. It is widely recognized as a method to quantify the utility of an altered image, using information degradation as a proxy. The score ranges from 100, representing a perfect match to the original image, to values approaching 0, indicating a significant loss of structural and perceptual information with respect to the original. In the following tables and charts we report averaged values obtained across the images test sets. Similarly, L–PIPS [202], another well–recognized metric, was evaluated alongside MS–SSIM. As both metrics yield largely consistent findings, the results with L–PIPS are omitted for brevity.

5.2.7 Evaluation of the Weak Loss for Privacy

	COCO [24]												Pascal VOC 2012 [200]											
	AP ↑				AP_{50} ↑				MS ↓				AP ↑				AP_{50} ↑				MS ↓			
$p =$	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
$n = 0$	18	30	33	35	40	57	60	62	36	45	46	47	14	25	27	28	34	51	54	56	35	44	45	46
$n = 1$	26	33	36	32	52	60	64	61	46	51	46	53	20	27	30	27	44	54	58	53	44	50	45	52
$n = 2$	32	36	40	41	59	64	68	69	43	46	54	57	25	28	32	32	50	56	61	60	42	45	53	55
$n = 3$	30	32	38	39	58	60	67	68	44	46	53	51	24	26	31	31	51	53	59	59	43	44	52	49
$n = 4$	29	34	34	40	56	61	62	69	45	46	46	53	23	27	28	33	48	54	55	60	44	45	45	52
ALL	47				76				69				39				68				68			
TN	59				87				100				55				86				100			

Table 5.6: AP, AP_{50}, and MS–SSIM (MS) results for different values of p and n, representing the number of positive and negative proposals selected during co–training. Results are compared against two baselines: training with all proposals (ALL) and using the TaskNet on plain (non–obfuscated) images (TN). For TN, MS–SSIM is set to 100, indicating no privacy protection. The results highlight the trade–off between task performance and privacy: increasing p and n generally enhances detection accuracy (higher AP and AP_{50}), but reduces privacy (higher MS). Intermediate settings (e.g., $p = 2$, $n = 2$) offer a favorable balance between the two objectives.

The results reported in Table 5.6 show that our co–training scheme based on weak loss induces a tunable trade–off between perception performance and privacy, enabling the encoder–decoder to discard sensitive information, reducing the reconstruction power of the attacker while

preserving the necessary features to allow object detection. Specifically, the detection capability (AP and AP_{50}) of the TaskNet on obfuscated images increases with higher values of p and n while privacy improves by reducing the amount of proposals used in training. It is interesting to observe how using (very) few proposals for training the EDO produces a marginal decrease in performance while drastically increasing privacy with respect to the ALL baseline (see Figure 5.10), where thousands of proposals are used. As an example, on the COCO dataset, when $p = n = 4$, AP_{50} has a $\approx 9\%$ drop against ALL while MS–SSIM improves by $\approx 23\%$. This outcome is further corroborated by the images from the Pascal VOC dataset, validating the robustness of our method in out–of–distribution settings. In particular, with Pascal VOC's data, AP_{50} decreases by $\approx 12\%$ while MS–SSIM improves by $\approx 24\%$ when $p = n = 4$ with respect to the ALL baseline.

From the additional indicators, whose values are reported in Figure 5.9, we can highlight and examine a performance trade–off not captured by the standard AP. On the one hand, increasing the positive proposals p allows the TaskNet to reach a better true positive rate (TP), obtaining values close to the ALL baseline both with COCO and Pascal VOC datasets. In particular, when setting $p = n = 4$, there is a reduction of $\approx 19\%$ and $\approx 21\%$ in true positives (TPs) compared to the ALL baseline for COCO and Pascal VOC images, respectively. On the other hand, increasing the negative proposals n reduces the number of false positive bounding boxes on the background (BFD). As expected, training the EDO without considering negative proposals ($n = 0$) results in obfuscated images that fail to suppress the false positive predictions produced by the TaskNet. This can be seen in the second row of Figure 5.9, which shows a significantly higher BFD rate compared to the ALL baseline. Interestingly, the BFD values obtained by the TaskNet on obfuscated images using 2 or more negative proposals are remarkably lower than those obtained both with the ALL baseline and the TaskNet on plain images (TN). This demonstrates that our approach reduces the number of errors, making the detection process more conservative. In particular, training the EDO using only 4 positive and negative proposals drops the BFD of $\approx 36\%$ and $\approx 29\%$ compared to using all proposals on COCO and Pascal VOC, respectively.

Overall, our extensive experimental campaign gives some guidelines for choosing the values of p and n to balance detection and privacy. Higher positives (p) increase the true positive predictions (TP) while the errors (BFD) are reduced by increasing the negatives (n). These combined findings are due to the fact that, while the p proposals promote the activation of bounding boxes close to the targets, the n ones force the suppression of spurious detections (BFD). Given this, the plots in Figure 5.9 suggest that choosing values of p and n close to each other ensures a good compromise between TP and BFD, in particular when $p = n \in \{2, 4\}$. This can be seen by comparing the raw values reported in Table 5.6: while the configurations with $p = n \in \{1, 2\}$ and $p = n \in \{3, 4\}$ reach comparable obfuscation results, setting $p = n = 4$ ($p = n = 2$) increases the detection performance compared to $p = n = 3$ ($p = n = 1$).

The benefit of reducing the number of proposals can be qualitatively appreciated in the out–of–distribution examples of Pascal VOC reported in Figure 5.10. At first, we can notice how the images obfuscated with the baseline (ALL) are similar to the original inputs and preserve

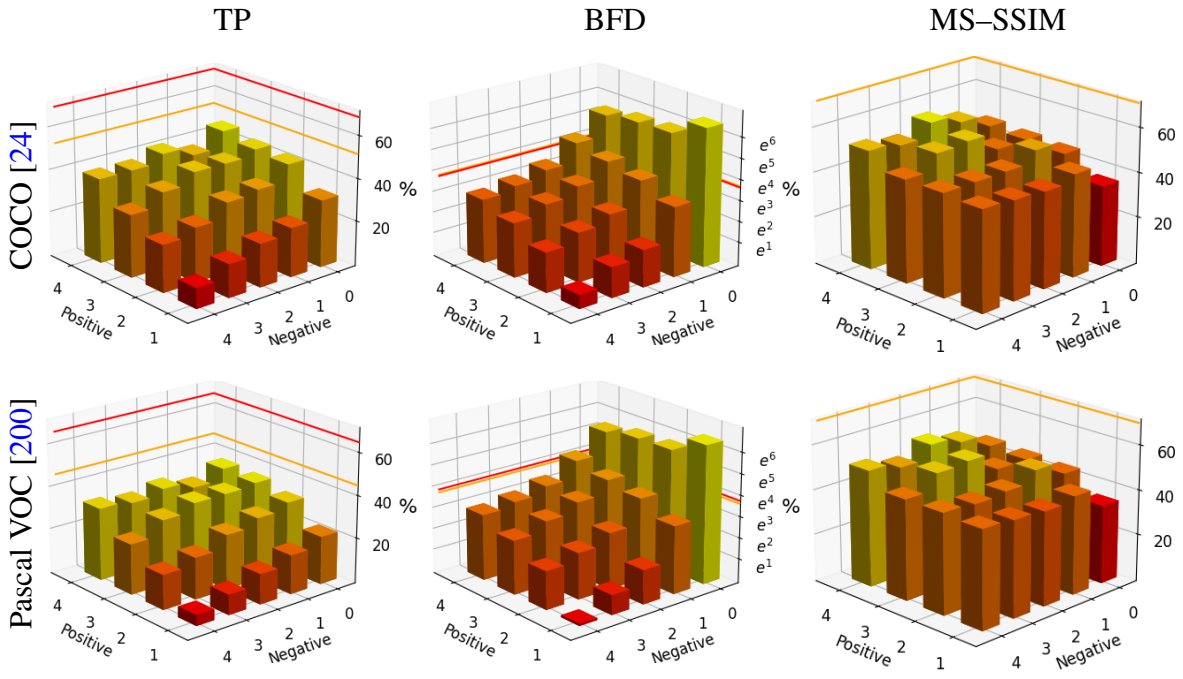


Figure 5.9: True Positive rate (TP), Background False Detection rate (BFD), and MS–SSIM (MS) for our co–training scheme under different values of p and n (positive and negative proposals). The orange line represents the performance of the EDO trained with all proposals (ALL), while the red line denotes the TaskNet operating on plain images (TN). For reference, TN's MS–SSIM is fixed at 100, indicating no privacy. The results further illustrate the trade–off between perception performance and privacy.

privacy–sensitive details that enable the attacker to obtain a good reconstruction of the original input. In particular, the baseline retains characteristics like facial expressions, hairstyles, body silhouettes, and background elements, which not only help identify individuals but also offer context regarding the locations where the images were taken. Conversely, the privacy ensured by our method, with both p and n set to 4, is remarkably better than those obtained with the ALL baseline. Furthermore, reducing the number of proposals further enhances the level of obfuscation, removing critical visual clues such as clothing colors and altering the shapes of faces, bodies, and objects in the scene, preserving user identity. Despite the obfuscation provided by the EDO, the TaskNet maintains robust detection performance even on privatized images.

These findings are also confirmed in the vehicle detection task. The results reported in Table 5.7 confirm that our method strongly improves privacy (lower MS–SSIM compared to ALL) while preserving good detection performance. In particular, setting $p = n = 3$ reduces the AP of $\approx 11\%$ while MS–SSIM improves by $\approx 19\%$ on COCO. This trade–off is also more evident in the Pascal VOC benchmark, where the AP decreases by $\approx 5\%$ while MS–SSIM shows a remarkable improvement of $\approx 20\%$. Interestingly, the performances obtained with the out–of–distribution dataset of Pascal VOC are better than those obtained using the in–distribution COCO. This is due to the fact that COCO images contain some challenging small targets that are difficult to detect in obfuscated images. In the task of people detection, using 2 or 3 positive and negative proposals enhances MS–SSIM, maintaining similar detection performance. Con-

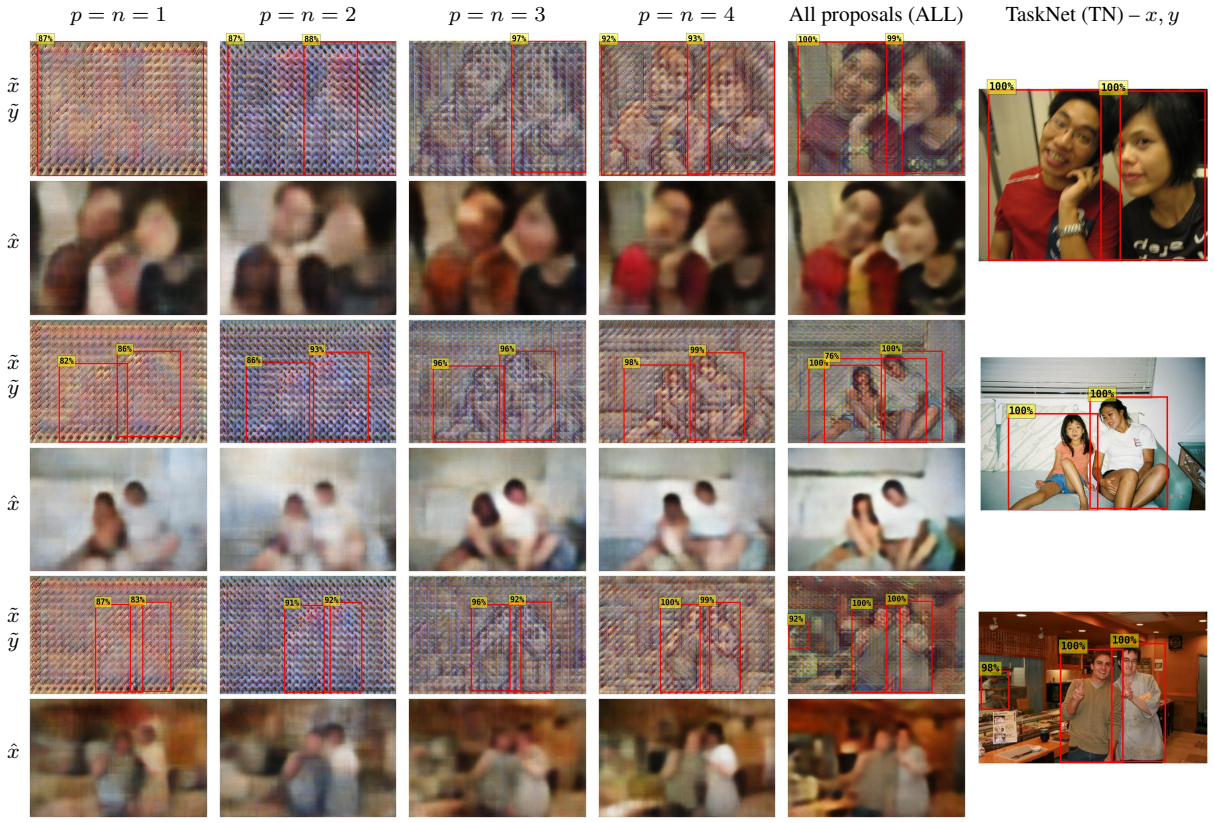


Figure 5.10: Examples of our method on out–of–distribution images from Pascal VOC 2012 [200]. (Left) Obfuscated images ($\tilde{x}$) and their reconstructions ($\hat{x}$) generated by the attacker. (Right) Corresponding plain images (x) with TaskNet detections (y) shown for reference. Red bounding boxes indicate TaskNet predictions (y and $\tilde{y}$) with confidence scores $\sigma(y), \sigma(\tilde{y}) \geq 0.75$, filtered using Non–Maximum Suppression (NMS) with an IoU threshold of 0.5. Results show that, despite strong obfuscation, TaskNet retains detection capabilities on $\tilde{x}$, while reconstruction quality remains low.

	COCO [24]					Pascal VOC 2012 [200]				
p, n	AP↑	AP_{50}↑	TP↑	BFD↓	MS↓	AP↑	AP_{50}↑	TP↑	BFD↓	MS↓
2, 2	23	38	26	42	46	33	54	38	42	49
3, 3	30	48	35	45	47	42	64	49	56	49
ALL	34	56	38	40	58	44	71	51	50	61
TN	52	77	59	33	100	63	88	74	30	100

Table 5.7: Performance of our method in a 4–class vehicle detection scenario. Results are reported for two settings of the weak loss parameters ($p = n = 2$ and $p = n = 3$), and compared with the TaskNet operating on plain images (TN). The results show that our approach generalizes to multi-class settings.

versely, for vehicle detection, setting $p = n = 2$ reduces detection performance without offering any privacy advantage over $p = n = 3$. This indicates that, as seems reasonable, the number of proposals in detection tasks involving multiple object categories should be marginally higher than in single–class object detection.

Visual examples of out–of–distribution data from Pascal VOC can be seen in Figure 5.11.

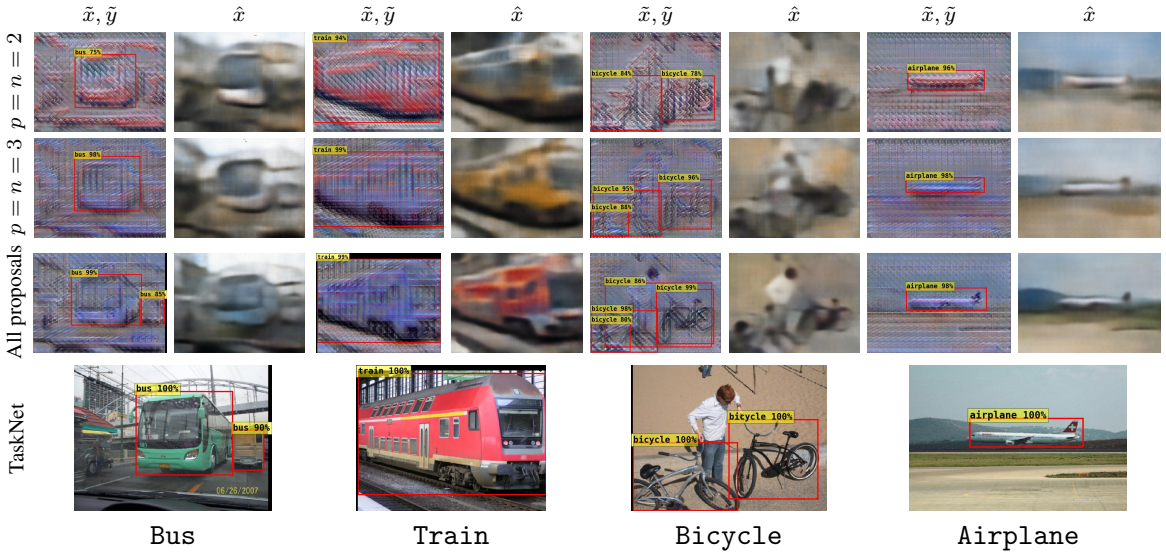


Figure 5.11: Qualitative examples for the vehicle detection task on out–of–distribution images from Pascal VOC 2012 [200]. (Top) Obfuscated images ($\tilde{x}$) with corresponding TaskNet detections ($\tilde{y}$) and reconstructions by the attacker ($\hat{x}$). (Bottom) Plain images (x) and TaskNet detections (y) used as reference. Red bounding boxes indicate TaskNet predictions (y and $\tilde{y}$) with confidence scores $\sigma(y), \sigma(\tilde{y}) \geq 0.75$, filtered using Non–Maximum Suppression (NMS) with an IoU threshold of 0.5. These examples illustrate that our method preserves detection capabilities on obfuscated images also in a multi-class detection task.

Also in this context, the ALL baseline preserves critical features from the original images, such as the shape and color of vehicles, as well as other contextual details in the background, allowing the attacker to restore privacy–sensitive information. In contrast, our method significantly enhances privacy, making it extremely difficult to distinguish the types of vehicles. For instance, the outline of the bus strongly degrades, the color and shape of the train are completely lost, the person near the bicycles is mixed with the background, and the features to identify the airplane (such as the tail and wings) completely disappear. Despite this, our method is still able to perform vehicle detection also with the challenging obfuscated images. Interestingly, our method solves some errors produced by the ALL baseline, such as the small van near the bus or the multiple overlapped bicycles.

5.2.8 Evaluation with an Enhanced Model Inversion Attack

We further evaluate the protection provided by our method against MIA threats. To do this, we enhance the reconstruction power of the attacker using the loss function of Eq. 5.8 that aims to recover the privacy–sensitive features from the obfuscated image. Table 5.8 report the MS–SSIM performance achieved by the two different attacker models on both in– and out–of–distribution datasets (COCO and Pascal VOC) with the EDO trained with different proposal configurations.

The results demonstrate that our method is robust to enhanced attackers using more sophis-

p, n	COCO MAE	COCO EC	Pascal VOC MAE	Pascal VOC EC
$1, 1$	46	45	44	43
$2, 2$	46	46	45	45
$3, 3$	53	54	52	53
$4, 4$	53	53	52	51
ALL	69	69	68	68

Table 5.8: Comparison of the reconstruction power (measured with MS–SSIM) of the MIAs trained with different loss functions: the Mean Absolute Error (MAE) and the enhanced edge–centric loss of Eq. 5.8 (EC). Results show how the attacker's reconstruction power does not substantially change for a more sophisticated attacker.

ticated loss functions. The MIA trained with the EC loss reaches MS–SSIM values (very) close to the MIA using the simpler MAE. This further corroborates the effectiveness of our method in compromising the reconstruction power of a malicious actor by removing privacy–sensitive features that are redundant for the task execution.

Figure 5.12 shows qualitative examples comparing the reconstruction obtained by the MAE and EC attackers. Our method is robust against both the simpler MAE loss and the more complex EC one: the reconstructed images obtained by the two attack models are similar at first glance, and only a closer inspection reveals the few positive effects brought by the more powerful EC loss. As an example, the images reconstructed by the attacker using the EC loss have less noise in uniform areas (e.g., walls), and have slightly better details in edges and contours. Still, data obfuscated by our method and reconstructed with the more powerful EC loss contain significant less details when compared against data reconstructed from images obfuscated with our baseline method (EDO trained with all proposals). While the enhanced loss indeed produces minor improvements from the point of view of the attacker when compared with the less powerful MAE, it fails to restore privacy–sensitive fine details, thus demonstrating again the effectiveness of our approach.

5.2.9 Evaluation on a Real Robot

In this section, we assess the efficacy of our approach when deployed on a real mobile robot, which has to autonomously carry out the task of people detection while navigating in an indoor environment. Motivated by the results reported in Section 5.2.7, we test our framework setting $p = n \in \{1, 2, 3, 4\}$.

The results presented in Table 5.9, obtained with Giraff–X, show that our approach achieves satisfying detection and privacy preservation capabilities. Our method remarkably improves obfuscation while maintaining detection capabilities very close to the ALL baseline. Specifically, setting $p = n = 4$ degrades AP_{50} from 87 to 85 ($\approx 2\%$) while improving MS–SSIM from 81 to 61 ($\approx 25\%$). Again, reducing p and n slightly degrades detection but strongly increases

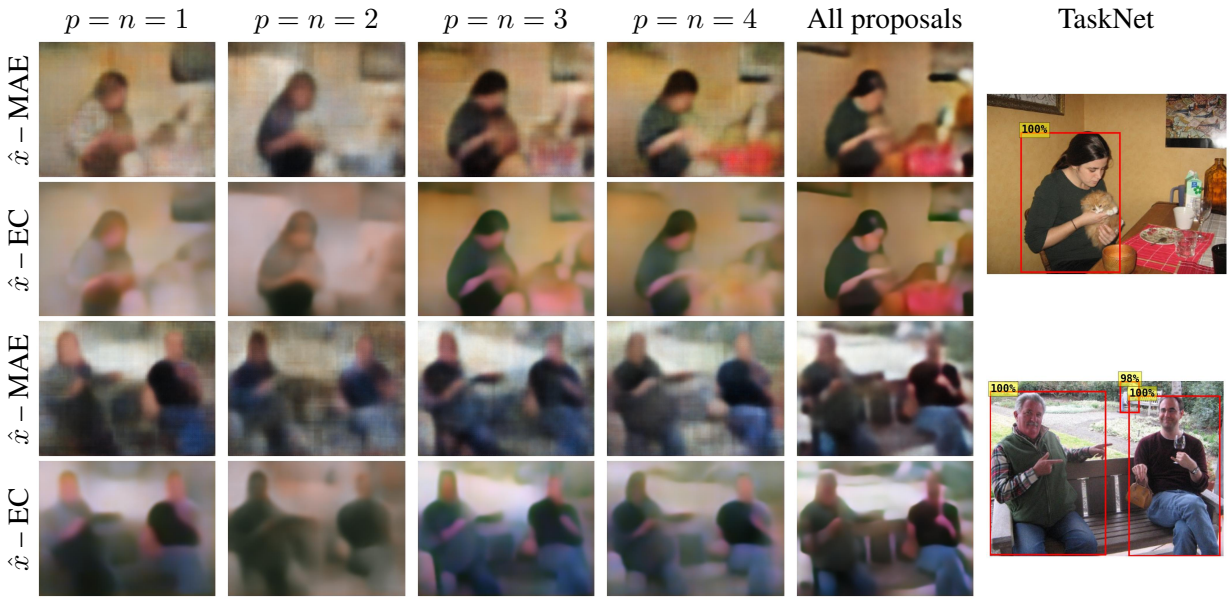


Figure 5.12: Qualitative comparison of two MIA strategies trained with different loss functions: Mean Absolute Error (MAE) and the enhanced edge–centric loss (EC) defined in Eq. 5.8. Reconstructions ($\hat{x}$) are shown for various EDO configurations on out–of–distribution examples from Pascal VOC 2012 [200]. (Top) Reconstructed images produced by the attacker. (Bottom) Corresponding plain images (x) and TaskNet detections (y) shown for reference. Red bounding boxes indicate TaskNet predictions with confidence $\sigma(y) \geq 0.75$, filtered using Non–Maximum Suppression (NMS) at IoU threshold 0.5. Despite the more sophisticated EC loss, visual inspection reveals no substantial differences.

privacy. As an example, setting $p = n = 2$ further improves privacy by $\approx 33\%$ (MS–SSIM) at the cost of $\approx 4\%$ AP_{50} drop. Another interesting fact can be observed by comparing the AP, TP, and BFD performance of the configurations where $p = n \in \{2, 3, 4\}$. While the AP values remain almost the same, the true positive (TP) and spurious detection in the background (BFD) report an evident decreasing trend. Specifically, when the number of proposals is reduced from 4 to 2, TaskNet misses $\approx 16\%$ of the ground truths (with TP decreasing from 80 to 67) but, at the same time, the rate of error is halved (BFD drops from 11 to 5). This further demonstrates that our method makes the detection process more conservative as it reduces the number of false positive detections while maintaining the AP stable. Similar to the people detection evaluation on COCO, setting $p = n \in \{2, 4\}$ better balances detection and privacy compared to $p = n \in \{1, 3\}$: using 4 (or 2) positive and negative proposals ensures a comparable level of privacy with higher detection performance than using 3 (or 1) proposals per type.

Figure 5.13 visualizes some representative examples of this evaluation. The images have been acquired from the point of view of our robot freely moving in the environment (using its navigation stack) while performing people detection on obfuscated images (TaskNet detections are reported for reference). It is easy to see how the perceptions obfuscated by the ALL baseline have visual characteristics that closely resemble those of plain images. Consequently, the attacker can generate restored images that are similar to the original perceptions, albeit with a slight blurring effect. From Figure 5.13, we can see how the ALL baseline preserves environmental features (such as the scene's structure or the furniture's outline) as well as visual clues

p, n	AP ↑	AP_{50} ↑	TP ↑	BFD ↓	MS ↓
$1, 1$	37	68	58	8	53
$2, 2$	56	83	67	5	54
$3, 3$	57	84	77	10	61
$4, 4$	59	85	80	11	61
ALL	66	87	86	17	81
TN	78	96	95	19	100

Table 5.9: Detection and privacy performance of our method deployed on the Giraff–X robotic platform for the task of people detection. Results are reported in terms of AP, AP_{50}, and MS–SSIM (MS). These results demonstrate that our method remains effective when deployed on a real robot.

related to the people allowing their identification. On the contrary, our method substantially increases data protection by removing important details from the privatized images. This is evident from the first two examples of Figure 5.13 that contain people in the foreground (close to the robot's camera). Thanks to co–training with weak loss, the EDO strongly degrades the quality of the obfuscated images, preventing the attacker from reconstructing privacy–sensitive details such as the body shapes, the clothing colors, and the background structure, protecting people from identification. Moreover, the obfuscated images preserve the necessary features for detecting people with high precision, even in challenging instances such as those affected by poor illumination or when the targets are at a high distance from the robot (see, respectively, the last two examples of Figure 5.13). A video of this experiment is available in the graphical abstract.

5.2.10 Evaluation of the Computational Performance

To assess the computational requirements of our framework, we deploy the EDO on two real robotics platforms with different hardware configurations: a TurtleBot 3 equipped with a Raspberry PI 4 (CPU) and Giraff–X [7] mounting an NVIDIA Jetson TX2 (a GPU accelerator specifically designed for low–powered devices). On the two robots, EDO reaches 0.5 and 5 FPS in CPU and GPU, respectively, while processing a 256×256 camera stream. FPS can be further improved by reducing the dimensionality of the encoder–decoder obfuscator. We test this solution by halving the channels of the EDO's convolutional layers: this increases the FPS to 1 (16) in CPU (GPU) with a marginal loss in detection performance (≈ 2 AP points) while preserving MS–SSIM. To better contextualize the efficiency of our method, we test the framerate of the Faster R–CNN with a ResNet–50. It reaches 1 FPS on the Jetson TX2, while the Raspberry PI 4 didn't manage to process a single frame in a reasonable amount of time. Despite the GPU acceleration, the TaskNet is 5x and 16x slower than our EDO in the two configurations with full and halved channels. The TaskNet offers various configurations with different accuracy and size. On the robot's low–powered hardware, we used a TaskNet with a reduced number of parameters. This contrasts with the cloud provider, which can employ larger backbones (e.g., ResNet–152) to maximize detection accuracy [21]. Running these more extensive models di-

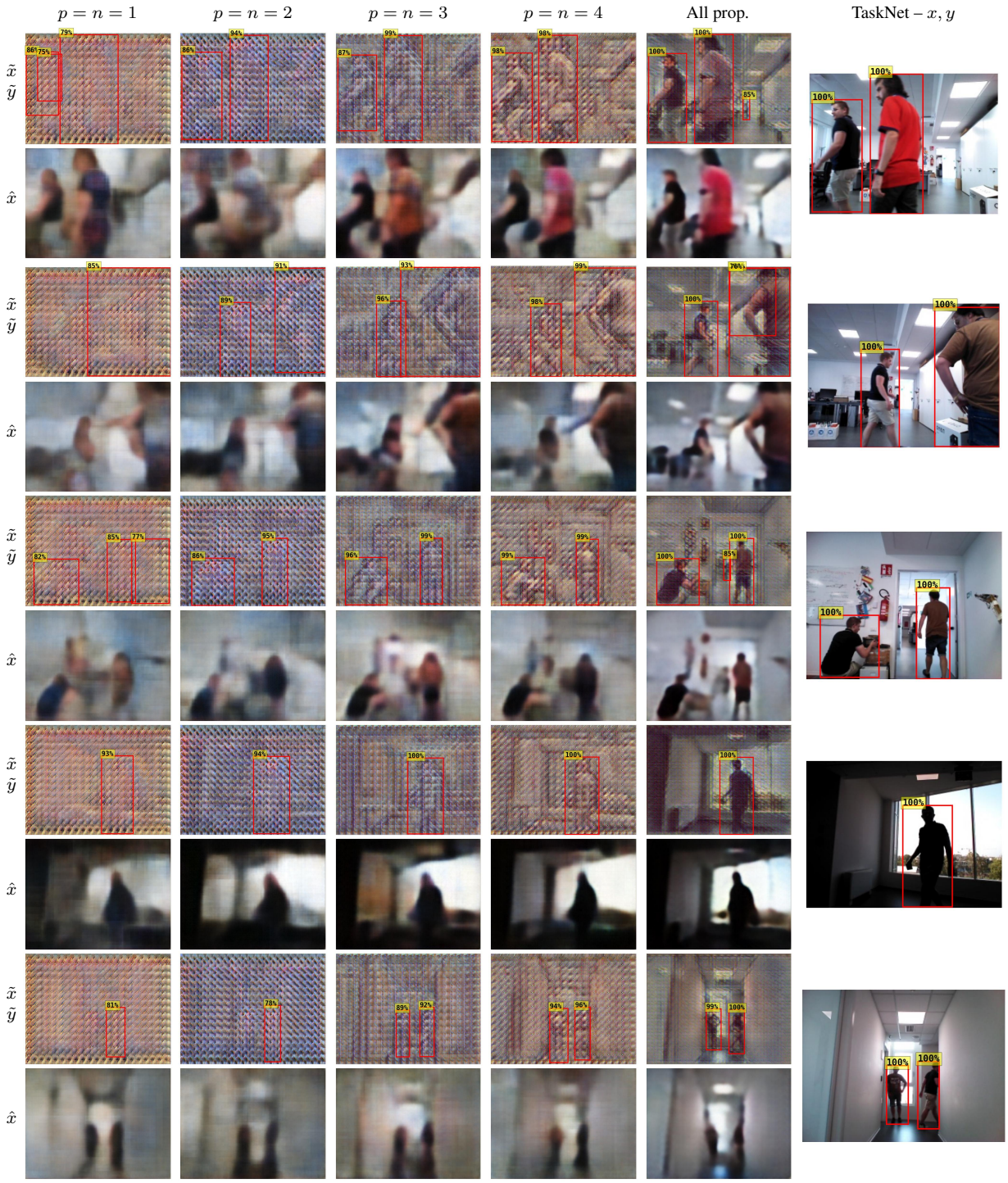


Figure 5.13: Qualitative examples of our method applied to images acquired from the onboard camera of the Giraff–X robot during real-world operation. (Left) Obfuscated images ($\tilde{x}$) and corresponding reconstructions ($\hat{x}$) generated by the attacker. (Right) Plain input images (x) with TaskNet detections (y) shown for reference. Red bounding boxes indicate TaskNet predictions (y and $\tilde{y}$) with confidence scores $\sigma(y), \sigma(\tilde{y}) \geq 0.75$, filtered using Non–Maximum Suppression (NMS) at an IoU threshold of 0.5. The examples highlight the ability of our method to preserve detection performance while ensuring privacy in real robotic deployments.

rectly on the robot is not feasible due to hardware limitations or further increases the inference time. In contrast, our EDO demonstrates superior efficiency and its dimension can be easily tuned according to the hardware setups. In addition, while the TaskNet saturates the computational capabilities of the device even at 1 FPS, our EDO performs inference in just 0.2 seconds (≈ 0.06 seconds in the halved configuration). This allows practitioners to limit the EDO's inference speed to run multiple inference tasks or to preserve energy, thus enhancing operational autonomy.

The frame rate achieved by our method is consistent with the latency required for online computation by the modern architectures for fog and cloud robotics, as the one reported in [42] and, in our specific task of object detection for robotic vision, the frame rate of our implementation is in line with that of [203]. To properly contextualize these results, consider that the output of object detection is often used for mission–critical decision tasks that do not require hard real–time performance.

5.3 Lessons Learned and Future Works

This chapter describes two solutions to tackle the scalability of domain adaptation and privacy preservation in cloud robotics, namely when a generic model (called TaskNet) is deployed in a remote server and accessed by multiple robots through queries. This paradigm is becoming popular to avoid intensive inference using the limited hardware of mobile robots. Our contributions demonstrate the synergy between robotics and cloud computing, in which the main perception task (in our case, object detection) is offloaded to the cloud, while robot–specific sub–tasks are efficiently executed on–board using lightweight neural networks. More specifically, privacy is reached using an encoder–decoder trained with a weak loss mechanism tasked to obfuscate robot's perceptions before uploading, while scalable adaptation is ensured by R2SNet that refines the TaskNet's predictions according to the target domain (or environment) in which the robot operates.

A limitation of R2SNet is that it requires human supervision to be qualified for a target environment. Because of this, in future works, we will investigate unsupervised approaches for unsupervised adaptation in object detection. We plan to do this by leveraging 3D environmental representations for filtering out spurious bounding boxes, thus extending the solutions of Chapter 4. Regarding privacy preservation, we want to investigate the possibility of moving the decoder to the server side, thus further reducing the computational load of the robot. Another valuable extension of our contributions consists of fusing privacy and adaptation in a single framework by developing a unified mechanism to train the small DNNs for privacy and adaptation. This is a promising direction as the aforementioned solutions for privacy and adaptation are well connected, as they are specifically designed for object detection, sharing also the same TaskNet.

Chapter 6

Conclusions

This dissertation extensively studies and addresses the problem of domain shift in robotic vision, proposing innovative solutions for the adaptation of DNNs in real–world robots deployment.

In the first part, we devise and evaluate the straightforward pipeline for implementing robotic vision. To do this, we consider the task of object detection with service robots, focusing on the real–time detection of doors, intended as variable–traversability passages. We leverage state–of–the–art deep–learning techniques combined with simulation and fine–tuning to cost–effectively synthesize detectors that operate with satisfying performance, even when faced with challenging instances and conditions. We conducted an extensive experimental campaign exploiting and adapting public datasets and simulation frameworks, while also carrying out on–the–field data acquisition and experimentation in four distinct real–world settings.

We envisage future directions building upon the limitations of our method. Enhancing the photorealism in our simulation framework would allow to further close the sim–to–real gap. One interesting objective in this direction is to improve the visual quality of synthetic simulators, such as iGibson, to fully exploit its high level of automation. Our method could gain a significant boost by integrating automatic scene design/generation, overcoming the limit to rely on hand–crafted scenes. This is a flourishing area of research whose recent progress could find in our setting an intriguing use case. While LiDAR and depth data have well–known limits for the task of robotic vision, integrating them in the RGB pipeline with a sensor fusion approach could introduce significant advantages. An interesting solution is to use these technologies to confirm the status of previously identified doors when the robot is close enough to them. Another undoubtedly interesting direction of research would be to conduct a large–scale experimentation in a pilot campaign with a fleet of service robots deployed in real setups.

The second part of this dissertation investigates how removing the necessity of human supervision for adaptation. The first contribution we present is a novel approach for unsupervised domain adaptation in robotics applications based on self–supervision with instance–aware pseudo–labels. At first, our method leverages 3D environmental mapping to force the spatial consistency between the model's prediction and the environment. Then, we use a foundation model for instance segmentation, to further refine the pseudo–labels according to the object instances and remove rendering artifacts. We extensively validate our approach with a real–world

dataset in multiple indoor environments, demonstrating that our approach effectively tackles the challenges of robots' deployments. Our second contribution for removing human supervision in model adaptation is a self–supervised approach for 3D object detection. Using solely the estimated odometry as the supervision signal for adapting a source–domain model, our method forces the model's predictions to be consistent with the robot's movements with the state consistency loss. The extensive experimental evaluation in the real world proves that our method outperforms both zero–shot and unsupervised domain adaptation baselines. In addition, we demonstrate that our approach is effective, even when a limited amount of target domain data is available.

In future works, we plan to extend our approaches with few–shot adaptation techniques to reduce the amount of target domain data necessary for fine–tune the model. Then, we want to introduce uncertainty estimation into the training setup, aiming to minimize the entropy and reduce the impact of wrong detections. Another interesting line of work is to fuse instance segmentation with mapping to better segment object instances directly in the 3D world. Once this step is reached, another promising extension consists of fixing the object class assigned to entire objects using visual language models.

In the third part of this dissertation, we address the dual problems of scalable adaptation and privacy preservation in cloud robotics, an emerging paradigm to offload intensive inference tasks from mobile robots to remote servers. We focus on the task of object detection and our solutions involve lightweight neural networks that can be run directly by the robot. To enable scalable adaptation and privacy preservation, we manipulate in two different ways the object proposals generated by the modern end–to–end object detector. Specifically, for scalable adaptation, we propose a refinement step to adjust the bounding box parameters after remote inference, thus solving the errors of the detector according to the target domain in which the robot operates. To do this, we design R2SNet, a lightweight neural network that performs the relabeling, rescoring, and suppression of object proposals. We validate our approach with the door status detection task performed in 4 real–world environments. Instead, for privacy preservation, we propose a novel co–training scheme where the loss of the object detector is weakened with a preliminary step of proposal selection. With this approach, we train an encoder–decoder that removes sensitive information while maintaining task–relevant features from the robot's perceptions before sending. For evaluation, we use the people detection task and we extensively assess the obfuscation properties of our method implementing attackers with various loss functions aiming at image reconstruction.

In future works, we will test these approaches with other object detection methods. Furthermore, we plan to fuse them together in order to design a complete architecture for adaptation and privacy in cloud robotics. Another promising future direction is to extend these methods to other tasks, such as semantic segmentation.

References

[1] In Lee. “Service robots: A systematic literature review”. In: *Electronics* 10.21 (2021), p. 2658.

[2] Mary B. Alatise and Gerhard P. Hancke. “A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods”. In: *IEEE Access* 8 (2020), pp. 39830–39846.

[3] Jane Holland, Liz Kingston, Conor McCarthy, Eddie Armstrong, Peter O’Dwyer, Fionn Merz, and Mark McConnell. “Service robots in the healthcare sector”. In: *Robotics* 10.1 (2021), p. 47.

[4] Feng Xiao, Jie Fang, Xing Guo, Youhai Zhang, and Rubing Huang. “Enhanced dynamic visual SLAM system for hospital logistics robots: Nonlinear optimal filtering, deep learning, and real-time positioning”. In: *Robotics and Autonomous Systems* 193 (2025), p. 105081.

[5] Giuseppe Fragapane, Rene De Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda”. In: *European Journal of Operational Research* 294.2 (2021), pp. 405–426.

[6] Ha Manh Do, Minh Pham, Weihua Sheng, Dan Yang, and Meiqin Liu. “RiSH: A robot-integrated smart home for elderly care”. In: *Robotics and Autonomous Systems* 101 (2018), pp. 74–92.

[7] Matteo Luperto, Marta Romeo, Javier Monroy, Jennifer Renoux, Alessandro Vuono, Francisco-Angel Moreno, Javier Gonzalez-Jimenez, Nicola Basilico, and N. Alberto Borghese. “User feedback and remote supervision for assisted living with mobile robots: A field study in long-term autonomy”. In: *Robotics and Autonomous Systems* 155 (2022), p. 104170.

[8] Luca Raggioli, Raffaella Esposito, Alessandra Rossi, and Silvia Rossi. “Exploring the Role of Robot’s Movements for a Transparent Affective Communication”. In: *IEEE Robotics and Automation Letters* 10.5 (2025), pp. 4364–4371.

[9] Pui Yee Leong and Nur Syazreen Ahmad. “LiDAR-Based Obstacle Avoidance With Autonomous Vehicles: A Comprehensive Review”. In: *IEEE Access* 12 (2024), pp. 164248–164261.

[10] Renato Martins, Dhiego Bersan, Mario FM Campos, and Erickson R Nascimento. “Extending maps with semantic and contextual object information for robot navigation: a learning-based framework using visual and depth cues”. In: *Journal of Intelligent & Robotic Systems* 99.3 (2020), pp. 555–569.

[11] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. “Visual language maps for robot navigation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 10608–10615.

[12] Nicky Zimmerman, Tiziano Guadagnino, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. “Long-Term Localization Using Semantic Cues in Floor Plan Maps”. In: *IEEE Robotics and Automation Letters* 8.1 (2023), pp. 176–183.

[13] Nicky Zimmerman, Matteo Sodano, Elias Marks, Jens Behley, and Cyrill Stachniss. “Constructing Metric-Semantic Maps Using Floor Plan Priors for Long-Term Indoor Localization”. In: *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 1366–1372.

[14] Elisa Maiettini, Vadim Tikhanoff, and Lorenzo Natale. “Weakly-Supervised Object Detection Learning through Human-Robot Interaction”. In: *Proceedings of the 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. 2021, pp. 392–399.

[15] Simone Arreghini, Gabriele Abbate, Alessandro Giusti, and Antonio Paolillo. “A Service Robot in the Wild: Analysis of Users Intentions, Robot Behaviors, and Their Impact on the Interaction”. In: *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 2536–2541.

[16] Yiqian Yang, Yuanduo Hong, Yeqing Yuan, Huihui Pan, and Weichao Sun. “Difference-Aware Fusion Network for Efficient RGB-D Semantic Segmentation in Indoor Robots”. In: *IEEE Transactions on Industrial Informatics* 21.10 (2025), pp. 7424–7434.

[17] Lukas Brunke, Yanni Zhang, Ralf Römer, Jack Naimer, Nikola Staykov, Siqi Zhou, and Angela P. Schoellig. “Semantically Safe Robot Manipulation: From Semantic Scene Understanding to Motion Safeguards”. In: *IEEE Robotics and Automation Letters* 10.5 (2025), pp. 4810–4817.

[18] Finn Lukas Busch, Timon Homberger, Jesús Ortega-Peimbert, Quantao Yang, and Olov Andersson. “One Map to Find Them All: Real-time Open-Vocabulary Mapping for Zero-shot Multi-Object Navigation”. In: *Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025, pp. 14835–14842.

[19] John Canny. “A Computational Approach to Edge Detection”. In: *Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698.

[20] Xiaodong Yang and Yingli Tian. “Robust door detection in unfamiliar environments by combining edge and corner features”. In: *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition – Workshops*. 2010, pp. 57–64.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: vol. 28. 6. 2015.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[23] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *Proceedings of European Conference on Computer Vision (ECCV) 2014*. 2014, pp. 740–755.

[25] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5828–5839.

[26] Jingjing Li, Zhiqi Yu, Zhekai Du, Lei Zhu, and Heng Tao Shen. “A Comprehensive Survey on Source-Free Domain Adaptation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.8 (2024), pp. 5743–5762.

[27] Poojan Oza, Vishwanath A. Sindagi, Vibashan Vishnukumar Sharmini, and Vishal M. Patel. “Unsupervised Domain Adaptation of Object Detectors: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[28] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. “Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning”. In: *IEEE Access* 9 (2021), pp. 153171–153187.

[29] Manuel Schwonberg, Joshua Niemeijer, et al. “Survey on Unsupervised Domain Adaptation for Semantic Segmentation for Visual Perception in Automated Driving”. In: *IEEE Access* 11 (2023), pp. 54296–54336.

[30] Bokui Shen et al. “iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes”. In: *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7520–7527.

[31] Manolis Savva et al. “Habitat: A Platform for Embodied AI Research”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[32] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88 (2009), pp. 303–338.

[33] Webots. *http://www.cyberbotics.com*. Ed. by Cyberbotics Ltd. Open-source Mobile Robot Simulation Software.

[34] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. “Domain Generalization: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2023), pp. 4396–4415.

[35] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. “Gibson env: Real-world perception for embodied agents”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9068–9079.

[36] Francesca Lunardini, Matteo Luperto, Marta Romeo, Jennifer Renoux, Nicola Basilico, Andrej Krpič, Nunzio Alberto Borghese, and Simona Ferrante. “The MOVECARE Project: Home-based Monitoring of Frailty”. In: *Proceedings of the 2019 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*. 2019.

[37] Lorenzo Scarciglia, Antonio Paolillo, and Daniele Palossi. *A Map-free Deep Learning-based Framework for Gate-to-Gate Monocular Visual Navigation aboard Miniaturized Aerial Vehicles*. 2025.

[38] Jonas Frey, Hermann Blum, Francesco Milano, Roland Siegwart, and Cesar Cadena. “Continual Adaptation of Semantic Segmentation Using Complementary 2D-3D Data Representations”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11665–11672.

[39] Zhizheng Liu, Francesco Milano, Jonas Frey, Roland Siegwart, Hermann Blum, and Cesar Cadena. “Unsupervised Continual Semantic Adaptation Through Neural Rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 3031–3040.

[40] Lorenzo Lamberti, Elia Cereda, Gabriele Abbate, Lorenzo Bellone, Victor Javier Kartsch Morinigo, Michał Barciś, Agata Barciś, Alessandro Giusti, Francesco Conti, and Daniele Palossi. “A Sim-to-Real Deep Learning-Based Framework for Autonomous Nano-Drone Racing”. In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1899–1906.

[41] Long Wen, Yu Zhang, Markus Rickert, Jianjie Lin, Fengjunjie Pan, and Alois Knoll. “Cloud-Native Fog Robotics: Model-Based Deployment and Evaluation of Real-Time Applications”. In: *IEEE Robotics and Automation Letters* 10.1 (2025), pp. 398–405.

[42] Jeffrey Ichnowski, Kaiyuan Chen, Karthik Dharmarajan, Simeon Adebola, Michael Danielczuk, Víctor Mayoral-Vilches, Nikhil Jha, Hugo Zhan, Edith LLontop, Derek Xu, et al. “FogROS2: An adaptive platform for cloud and fog robotics using ROS 2”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5493–5500.

[43] Francesco Lumpp, Franco Fummi, Hiren D. Patel, and Nicola Bombieri. “Enabling Kubernetes Orchestration of Mixed-Criticality Software for Autonomous Mobile Robots”. In: *IEEE Transactions on Robotics* 40 (2024), pp. 540–553.

[44] Daniel J Butler, Justin Huang, Franziska Roesner, and Maya Cakmak. “The privacy-utility tradeoff for remotely teleoperated robots”. In: *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*. 2015, pp. 27–34.

[45] Eileen Guo. “A Roomba recorded a woman on the toilet. How did screenshots end up on Facebook”. In: *https://www.technologyreview.com/2022/12/19/1065306/roomba-irobot-robot-vacuums-artificial-intelligence-training-data-privacy* (2022).

[46] Michele Antonazzi, Matteo Luperto, Nicola Basilico, and N. Alberto Borghese. “Enhancing Door-Status Detection for Autonomous Mobile Robots During Environment-Specific Operational Use”. In: *Proceedings of the 2023 European Conference on Mobile Robots (ECMR)*. 2023, pp. 1–8.

[47] Michele Antonazzi, Matteo Luperto, N. Alberto Borghese, and Nicola Basilico. “Development and Adaptation of Robotic Vision in the Real-World: the Challenge of Door Detection”. In: *Journal of Field Robotics (Wiley)* (2025).

[48] Michele Antonazzi, Lorenzo Signorelli, Matteo Luperto, and Nicola Basilico. *Instance–Guided Unsupervised Domain Adaptation for Robotic Semantic Segmentation*. Submitted to 2026 IEEE International Conference on Robotics and Automation (ICRA). 2026.

[49] Nicholas Carlotti, Michele Antonazzi, Elia Cereda, Mirko Nava, Nicola Basilico, Daniele Palossi, and Alessandro Giusti. *Self-supervised Domain Adaptation for Visual 3D Pose Estimation of Nano-drone Racing Gates by Enforcing Geometric Consistency*. Submitted to 2026 IEEE International Conference on Robotics and Automation (ICRA). 2026.

[50] Michele Antonazzi, Matteo Luperto, N. Alberto Borghese, and Nicola Basilico. “R2SNet: Scalable Domain Adaptation for Object Detection in Cloud–Based Robotic Ecosystems via Proposal Refinement”. In: *Proceedings of 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 2676–2682.

[51] Michele Antonazzi, Matteo Alberti, Alex Bassot, Matteo Luperto, and Nicola Basilico. “Privacy–Preserving Robotic Perception for Object Detection in Curious Cloud Robotics”. In: *IEEE Transactions on Robotics* (2025), pp. 1–19.

[52] Niko Sünderhauf et al. “The limits and potentials of deep learning for robotics”. In: *The International Journal of Robotics Research* 37.4–5 (2018), pp. 405–420.

[53] Junyi Chai, Hao Zeng, Anming Li, and Eric WT Ngai. “Deep learning in computer vision: A critical review of emerging techniques and application scenarios”. In: *Machine Learning with Applications* 6 (2021), pp. 100–134.

[54] Glenn Jocher. *YOLOv5 by Ultralytics*. https://github.com/ultralytics/yolov5. 2020.

[55] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. “End-to-End Object Detection with Transformers”. In: *Proceedings of European Conference on Computer Vision (ECCV) 2020*. 2020, pp. 213–229.

[56] Richard Bormann, Florian Jordan, Wenzhe Li, Joshua Hampp, and Martin Hägele. “Room segmentation: Survey, implementation, and analysis”. In: *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1019–1026.

[57] Matteo Luperto, Federico Amadelli, Moreno Di Berardino, and Francesco Amigoni. “Mapping beyond what you can see: Predicting the layout of rooms behind closed doors”. In: *Robotics and Autonomous Systems* 159 (2023), p. 104282.

[58] P. Espinace, T. Kollar, A. Soto, and N. Roy. “Indoor scene recognition through object detection”. In: *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 1406–1413.

[59] Niko Sünderhauf, Feras Dayoub, Sean McMahon, Ben Talbot, Ruth Schulz, Peter Corke, Gordon Wyeth, Ben Upcroft, and Michael Milford. “Place categorization and semantic mapping on a mobile robot”. In: *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 5729–5736.

[60] Tomáš Krajník, Jaime P. Fentanes, João M. Santos, and Tom Duckett. “FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments”. In: *IEEE Transactions on Robotics* 33.4 (2017), pp. 964–977.

[61] Lorenzo Nardi and Cyrill Stachniss. “Long-Term Robot Navigation in Indoor Environments Estimating Patterns in Traversability Changes”. In: *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 300–306.

[62] Nosan Kwak, Hitoshi Arisumi, and Kazuhito Yokoi. “Visual recognition of a door and its knob for a humanoid robot”. In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2079–2084.

[63] Iñaki Monasterio, Elena Lazkano, Iñaki Rañó, and Basilo Sierra. “Learning to traverse doors using visual information”. In: *Mathematics and Computers in Simulation* 60.3 (2002), pp. 347–356.

[64] Xiaochen He and Nelson Hon Ching Yung. “Corner detector based on global and local curvature properties”. In: *Optical Engineering* 47.5 (2008).

[65] G. Cicirelli, T. D’orazio, and A. Distante. “Target recognition by components for mobile robot navigation”. In: *Journal of Experimental & Theoretical Artificial Intelligence* 15.3 (2003), pp. 281–297.

[66] Wei Chen, Ting Qu, Yimin Zhou, Kaijian Weng, Gang Wang, and Guoqiang Fu. “Door recognition and deep learning algorithm for visual based robot navigation”. In: *Proceedings of the 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. 2014, pp. 1793–1798.

[67] Taehyeon Kim, Minwoo Kang, Sumin Kang, and Donghan Kim. “Improvement of Door Recognition Algorithm Using Lidar and RGB-D Camera for Mobile Manipulator”. In: *Proceedings of the 2022 IEEE Sensors Applications Symposium (SAS)*. 2022.

[68] Adrian Llopart, Ole Ravn, and Nils. A. Andersen. “Door and cabinet recognition using Convolutional Neural Nets and real-time method fusion for handle detection and grasping”. In: *Proceedings of the 2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. 2017, pp. 144–149.

[69] R. Cupec, I. Vidović, V. Šimundić, P. Pejić, S. Foix, and G. Alenyà. “Teaching a Robot Where Doors and Drawers Are and How To Handle Them”. In: *Proceedings of the 2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2023, pp. 2288–2294.

[70] Keunwoo Jang, Sanghyun Kim, and Jaeheung Park. “Motion Planning of Mobile Manipulator for Navigation Including Door Traversal”. In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 4147–4154.

[71] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. “Alphapilot: Autonomous drone racing”. In: *Springer Autonomous Robots* 46.1 (2022), pp. 307–320.

[72] Elia Cereda, Alessandro Giusti, and Daniele Palossi. “Training on the Fly: On-Device Self-Supervised Learning Aboard Nano-Drones Within 20 mW”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.11 (2024), pp. 3685–3695.

[73] Nicky Zimmerman, Hanna Müller, Michele Magno, and Luca Benini. “Fully Onboard Low-Power Localization with Semantic Sensor Fusion on a Nano-UAV using Floor Plans”. In: *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 11913–11919.

[74] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. “Deep Drone Racing: From Simulation to Reality With Domain Randomization”. In: *IEEE Transactions on Robotics* 36.1 (2020), pp. 1–14.

[75] Huy Xuan Pham, Andriy Sarabakha, Mykola Odnoshyvkin, and Erdal Kayacan. “PencilNet: Zero-Shot Sim-to-Real Transfer Learning for Robust Gate Perception in Autonomous Drone Racing”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11847–11854.

[76] Sourav Garg, Niko Sünderhauf, Feras Dayoub, Douglas Morrison, et al. “Semantics for Robotic Mapping, Perception and Interaction: A Survey”. In: *Foundations and Trends® in Robotics* 8.1–2 (2020), pp. 1–224. ISSN: 1935-8253.

[77] Jose-Luis Matez-Bandera, Javier Monroy, and Javier Gonzalez-Jimenez. “Sigma-FP: Robot Mapping of 3D Floor Plans With an RGB-D Camera Under Uncertainty”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 12539–12546.

[78] Oscar Mendez, Simon Hadfield, Nicolas Pugeault, and Richard Bowden. “Sedar: Reading floorplans like a human—using deep learning to enable human-inspired localisation”. In: *International Journal of Computer Vision* 128.5 (2020), pp. 1286–1310.

[79] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. “3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[80] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. “Clio: Real-Time Task-Driven Open-Set 3D Scene Graphs”. In: *IEEE Robotics and Automation Letters* 9.10 (2024), pp. 8921–8928.

[81] Zhe Ni, Xiaoxin Deng, Cong Tai, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, Xiang Wu, and Long Zeng. “GRID: Scene-Graph-based Instruction-driven Robotic Task Planning”. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 13765–13772.

[82] Peng Zhou, Jihong Zhu, Shengzeng Huo, and David Navarro-Alarcon. “LaSeSOM: A Latent and Semantic Representation Framework for Soft Object Manipulation”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5381–5388.

[83] Josh Tobin et al. “Domain Randomization and Generative Models for Robotic Grasping”. In: *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3482–3489.

[84] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. “Multi-Component Image Translation for Deep Domain Generalization”. In: *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 579–588.

[85] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. “Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[86] Tadanobu Inoue, Subhajit Choudhury, Giovanni De Magistris, and Sakyasingha Dasgupta. “Transfer learning from synthetic to real images using variational autoencoders for precise position detection”. In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 2725–2729.

[87] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. “Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[88] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. “Adversarial Discriminative Domain Adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[89] Yixin Zhang and Zilei Wang. “Joint Adversarial Learning for Domain Adaptation in Semantic Segmentation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (2020), pp. 6877–6884.

[90] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S. Yu. “Visual Domain Adaptation with Manifold Embedded Distribution Alignment”. In: *Proceedings of the ACM International Conference on Multimedia*. Seoul, Republic of Korea: Association for Computing Machinery, 2018, pp. 402–410.

[91] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. “Beyond Sharing Weights for Deep Domain Adaptation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.4 (2019), pp. 801–814.

[92] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. “A kernel method for the two-sample-problem”. In: vol. 19. 2006.

[93] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[94] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. “Image to Image Translation for Domain Adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[95] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. “CyCADA: Cycle-Consistent Adversarial Domain Adaptation”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 1989–1998.

[96] Yun-Chun Chen, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. “CrDoCo: Pixel-Level Domain Transfer With Cross-Domain Consistency”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[97] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. “Learning to Adapt Structured Output Space for Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[98] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. “ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[99] Zhedong Zheng and Yi Yang. “Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation”. In: *International Journal of Computer Vision* 129.4 (2021), pp. 1106–1120.

[100] Pan Zhang, Bo Zhang, Ting Zhang, et al. “Prototypical Pseudo Label Denoising and Target Structure Learning for Domain Adaptive Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[101] Divya Kothandaraman, Rohan Chandra, and Dinesh Manocha. “SS-SFDA: Self-supervised source-free domain adaptation for road segmentation in hazardous environments”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2021, pp. 3049–3059.

[102] Hao Yan, Yuhong Guo, and Chunsheng Yang. “Augmented self-labeling for source-free unsupervised domain adaptation”. In: *Proceedings of the NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*. 2021.

[103] Weijie Chen, Luojun Lin, Shicai Yang, Di Xie, Shiliang Pu, and Yueting Zhuang. “Self-Supervised Noisy Label Learning for Source-Free Unsupervised Domain Adaptation”. In: *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10185–10192.

[104] Tong Chu, Yahao Liu, Jinhong Deng, Wen Li, and Lixin Duan. “Denoised maximum classifier discrepancy for source-free unsupervised domain adaptation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 472–480.

[105] Ruihuang Li, Shuai Li, Chenhang He, Yabin Zhang, Xu Jia, and Lei Zhang. “Class-Balanced Pixel-Level Self-Labeling for Domain Adaptive Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, p. 1159.

[106] Lukas Hoyer, Dengxin Dai, Haoran Wang, and Luc Van Gool. “MIC: Masked Image Consistency for Context-Enhanced Domain Adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 11721–11732.

[107] Seongwon Jeong, Jiyeong Kim, Sungheui Kim, and Dongbo Min. “Revisiting Domain-Adaptive Semantic Segmentation via Knowledge Distillation”. In: *IEEE Transactions on Image Processing* 33 (2024), pp. 6761–6773.

[108] René Zurbrügg, Hermann Blum, Cesar Cadena, Roland Siegwart, and Lukas Schmid. “Embodied Active Domain Adaptation for Semantic Segmentation via Informative Path Planning”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8691–8698.

[109] Mirko Nava, Jérôme Guzzi, R. Omar Chavez-Garcia, Luca M. Gambardella, and Alessandro Giusti. “Learning Long-Range Perception Using Self-Supervision From Short-Range Sensors and Odometry”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1279–1286.

[110] Junwon Seo, Taekyung Kim, Kiho Kwak, Jihong Min, and Inwook Shim. “Scate: A scalable framework for self-supervised traversability estimation in unstructured environments”. In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 888–895.

[111] M. Nava, A. Paolillo, J. Guzzi, L. M. Gambardella, and A. Giusti. “Learning Visual Localization of a Quadrotor Using its Noise as Self-Supervision”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2218–2225.

[112] Nicholas Carlotti, Mirko Nava, and Alessandro Giusti. “Learning to Estimate the Pose of a Peer Robot in a Camera Image by Predicting the States of its LEDs”. In: *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 2763–2769.

[113] Mirko Nava, Luca Maria Gambardella, and Alessandro Giusti. “State-consistency loss for learning spatial perception tasks from partial labels”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1112–1119.

[114] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. “Cloud robotics: architecture, challenges and applications”. In: *IEEE Network* 26.3 (2012), pp. 21–28.

[115] Xiaofei Wang, Yiwen Han, Victor CM Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. “Convergence of edge computing and deep learning: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 869–904.

[116] Yundi Guo, Beiji Zou, Ju Ren, Qingqing Liu, Deyu Zhang, and Yaoxue Zhang. “Distributed and efficient object detection via interactions among devices, edge, and cloud”. In: *IEEE Transactions on Multimedia* 21.11 (2019), pp. 2903–2915.

[117] Sharif Abuadbba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A Camtepe, Yansong Gao, Hyoungshick Kim, and Surya Nepal. “Can we use split learning on 1d cnn models for privacy preserving training?” In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 2020, pp. 305–318.

[118] Imen Chakroun, Tom Vander Aa, Roel Wuyts, and Wilfried Verachtert. “Distributing intelligence for object detection using edge computing”. In: *Proceedings of 2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. IEEE. 2021, pp. 681–687.

[119] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. “Distributed deep neural networks over the cloud, the edge and end devices”. In: *Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 328–339.

[120] Ajay Kumar Tanwani, Nitesh Mor, John Kubiatowicz, Joseph E Gonzalez, and Ken Goldberg. “A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering”. In: *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 4559–4566.

[121] Ajay Kumar Tanwani, Raghav Anand, Joseph E Gonzalez, and Ken Goldberg. “RILaaS: Robot inference and learning as a service”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4423–4430.

[122] Dinsha Vinod and PS SaiKrishna. “Development of an autonomous fog computing platform using control-theoretic approach for robot-vision applications”. In: *Robotics and Autonomous Systems* 155 (2022), p. 104158.

[123] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *Proceedings of the European Conference on Computer Vision (ECCV) 2016*. 2016, pp. 21–37.

[124] William J Beksi, John Spruth, and Nikolaos Papanikolopoulos. “Core: A cloud-based object recognition engine for robotics”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 4512–4517.

[125] Manoj Penmetcha, Shyam Sundar Kannan, and Byung-Cheol Min. “Smart cloud: Scalable cloud robotic architecture for web-powered multi-robot applications”. In: *Proceedings of The International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 2397–2402.

[126] Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. “Network offloading policies for cloud robotics: a learning-based approach”. In: *Autonomous Robots* 45.7 (2021), pp. 997–1012.

[127] Adam K Taras, Niko Suenderhauf, Peter Corke, and Donald G Dansereau. “Inherently privacy-preserving vision for trustworthy autonomous systems: Needs and solutions”. In: *Journal of Responsible Technology* (2024), p. 100079.

[128] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. “When machine learning meets privacy: A survey and outlook”. In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–36.

[129] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. “Learning to Super-Resolve Blurry Face and Text Images”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[130] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. “Pulse: Self-supervised photo upsampling via latent space exploration of generative models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2437–2445.

[131] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2016, pp. 201–210.

[132] Hervé Chabanne, Amaury De Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. “Privacy-preserving classification on deep neural network”. In: *Cryptology ePrint Archive* (2017).

[133] Håkon Hukkelås, Rudolf Mester, and Frank Lindseth. “Deepprivacy: A generative adversarial network for face anonymization”. In: *Proceedings of the International symposium on visual computing*. Springer. 2019, pp. 565–578.

[134] Xiaoyi Yu, Kenta Chinomi, Takashi Koshimizu, Naoko Nitta, Yoshimichi Ito, and Noboru Babaguchi. “Privacy protecting visual processing for secure video surveillance”. In: *Proceedings of the 2008 15th IEEE International Conference on Image Processing*. IEEE. 2008, pp. 1672–1675.

[135] Myeung Un Kim, Harim Lee, Hyun Jong Yang, and Michael S Ryoo. “Privacy-preserving robot vision with anonymized faces by extreme low resolution”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 462–467.

[136] Jiacheng Lin, Yang Li, and Guanci Yang. “FPGAN: Face de-identification method with generative adversarial networks for social robots”. In: *Neural Networks* 133 (2021), pp. 132–147.

[137] Yunqian Wen, Bo Liu, Jingyi Cao, Rong Xie, Li Song, and Zhu Li. “IdentityMask: deep motion flow guided reversible face video de-identification”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.12 (2022), pp. 8353–8367.

[138] Michael Ryoo, Kiyoon Kim, and Hyun Yang. “Extreme low resolution activity recognition with multi-siamese embedding learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[139] Michael Ryoo, Brandon Rothrock, Charles Fleming, and Hyun Jong Yang. “Privacy-preserving human activity recognition from extreme low resolution”. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 31. 1. 2017.

[140] Munkhjargal Gochoo, Tan-Hsu Tan, Fady Alnajjar, Jun-Wei Hsieh, and Ping-Yang Chen. “Lownet: Privacy preserved ultra-low resolution posture image classification”. In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 663–667.

[141] Amitesh Singh Rajput, Balasubramanian Raman, and Javed Imran. “Privacy-preserving human action recognition as a remote cloud service using RGB-D sensors and deep CNN”. In: *Expert Systems with Applications* 152 (2020), p. 113349.

[142] Ang Li, Jiayi Guo, Huanrui Yang, Flora D. Salim, and Yiran Chen. “DeepObfuscator: Obfuscating Intermediate Representations with Privacy-Preserving Adversarial Learning on Smartphones”. In: *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 2021, pp. 28–39.

[143] Bardia Azizian and Ivan V. Bajić. “Privacy-Preserving Autoencoder for Collaborative Object Detection”. In: *IEEE Transactions on Image Processing* 33 (2024), pp. 4937–4951.

[144] Manabu Nakanoya, Sai Shankar Narasimhan, Sharachchandra Bhat, Alexandros Anemogiannis, Akul Datta, Sachin Katti, Sandeep Chinchali, and Marco Pavone. “Co-design of communication and machine inference for cloud robotics”. In: *Autonomous Robots* 47.5 (2023), pp. 579–594.

[145] Ming Li, Xiangyu Xu, Hehe Fan, Pan Zhou, Jun Liu, Jia-Wei Liu, Jiahe Li, Jussi Keppo, Mike Zheng Shou, and Shuicheng Yan. “STPrivacy: Spatio-Temporal Privacy-Preserving Action Recognition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 5106–5115.

[146] Tony Ng, Hyo Jin Kim, Vincent T Lee, Daniel DeTone, Tsun-Yi Yang, Tianwei Shen, Eddy Ilg, Vassileios Balntas, Krystian Mikolajczyk, and Chris Sweeney. “NinjaDesc: content-concealing visual descriptors via adversarial learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12797–12807.

[147] Zecheng He, Tianwei Zhang, and Ruby B Lee. “Model inversion attacks against collaborative inference”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. 2019, pp. 148–162.

[148] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. “Object Detection in 20 Years: A Survey”. In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276.

[149] Emily Whiting, Jonathan Battat, and Seth Teller. “Topology of urban environments”. In: *Computer-Aided Architectural Design Futures (CAADFutures) 2007: Proceedings of the 12th International CAADFutures Conference*. Springer. 2007, pp. 114–128.

[150] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115 (2015), pp. 211–252.

[151] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. “Surgical fine-tuning improves adaptation to distribution shifts”. In: *International Conference on Learning Representations* (2023).

[152] João Ramôa, Vasco Lopes, Luís Alexandre, and Sandra Mogo. “Real-time 2D–3D door detection and state classification on a low-power device”. In: *SN Applied Sciences* 3 (2021).

[153] Agnese Chiatti, Riccardo Bertoglio, Nico Catalano, Matteo Gatti, and Matteo Matteucci. “Surgical Fine-Tuning for Grape Bunch Segmentation Under Visual Domain Shifts”. In: *Proceedings of the 2023 European Conference on Mobile Robots (ECMR)*. 2023.

[154] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “Sun3d: A database of big spaces reconstructed using sfm and object labels”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1625–1632.

[155] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. “A review of physics simulators for robotic applications”. In: *IEEE Access* 9 (2021), pp. 51416–51431.

[156] Nathan Koenig and Andrew Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2004, pp. 2149–2154.

[157] Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. “Simulation environment for mobile robots testing using ROS and Gazebo”. In: *Proceedings of the 2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*. 2016, pp. 96–101.

[158] Lei Tai, Giuseppe Paolo, and Ming Liu. “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation”. In: *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 31–36.

[159] Jian Chen, Bingxi Jia, and Kaixiang Zhang. “Trifocal Tensor-Based Adaptive Visual Trajectory Tracking Control of Mobile Robots”. In: *IEEE Transactions on Cybernetics* 47.11 (2017), pp. 3784–3798.

[160] Yoshiaki Mizuchi and Tetsunari Inamura. “Cloud-based multimodal human-robot interaction simulator utilizing ROS and unity frameworks”. In: *Proceedings of the 2017 IEEE/SICE International Symposium on System Integration (SII)*. 2017, pp. 948–955.

[161] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. “USARSim: a robot simulator for research and education”. In: *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 1400–1405.

[162] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*. 2017.

[163] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. “Airsim: High-fidelity visual and physical simulation for autonomous vehicles”. In: *Proceedings of the Field and Service Robotics: Results of the 11th International Conference*. 2018, pp. 621–635.

[164] Pushkal Katara, Mukul Khanna, Harshit Nagar, and Annapurani Panaiyappan. “Open Source Simulator for Unmanned Underwater Vehicles using ROS and Unity3D”. In: *Proceedings of the 2019 IEEE Underwater Technology (UT)*. 2019.

[165] Eleonora Tagliabue, Ameya Pore, Diego Dall’Alba, Enrico Magnabosco, Marco Piccinelli, and Paolo Fiorini. “Soft Tissue Simulation Environment to Learn Manipulation Tasks in Autonomous Robotic Surgery”. In: *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 3261–3266.

[166] Mirella Santos Pessoa de Melo, José Gomes da Silva Neto, Pedro Jorge Lima da Silva, João Marcelo Xavier Natario Teixeira, and Veronica Teichrieb. “Analysis and Comparison of Robotics 3D Simulators”. In: *Proceedings of the 2019 21st Symposium on Virtual and Augmented Reality (SVR)*. 2019, pp. 242–251.

[167] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. “Navigating to objects in the real world”. In: *Science Robotics* 8.79 (2023), eadf6991.

[168] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. *Matterport3D: Learning from RGB-D Data in Indoor Environments*. 2017.

[169] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. “3D Semantic Parsing of Large-Scale Indoor Spaces”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1534–1543.

[170] Tongjie Y Zhang and Ching Y. Suen. “A fast parallel algorithm for thinning digital patterns”. In: *Communications of the ACM* 27.3 (1984), pp. 236–239.

[171] René Zurbrügg, Hermann Blum, Cesar Cadena, Roland Siegwart, and Lukas Schmid. “Embodied Active Domain Adaptation for Semantic Segmentation via Informative Path Planning”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8691–8698.

[172] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[173] Ali Farhadi and Joseph Redmon. *YoloV3: An incremental improvement*. 2018.

[174] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. “Dynamic DETR: End-to-End Object Detection With Dynamic Attention”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 2988–2997.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Proceedings of the Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[176] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[177] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. *Grounding dino: Marrying dino with grounded pre-training for open-set object detection*. 2023.

[178] Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 4015–4026.

[179] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. “Scaling Open-Vocabulary Object Detection”. In: *Proceedings of the Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 72983–73007.

[180] Ilya Loshchilov and Frank Hutter. *Fixing Weight Decay Regularization in Adam*. 2017.

[181] Kai Han et al. “A Survey on Vision Transformer”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2023), pp. 87–110.

[182] Nikhila Ravi et al. *SAM 2: Segment Anything in Images and Videos*. 2024.

[183] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. “Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping”. In: *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1689–1696.

[184] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning". In: *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1366–1373.

[185] Alexander Millane, Helen Oleynikova, Emilie Wirbel, Remo Steiner, Vikram Ramasamy, David Tingdahl, and Roland Siegwart. "nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 2698–2705.

[186] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration". In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), p. 1.

[187] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor segmentation and support inference from rgbd images". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2012, pp. 746–760.

[188] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. "On the Continuity of Rotation Representations in Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[189] Hanna Müller, Victor Kartsch, and Luca Benini. "GAP9Shield: A 150GOPS AI-Capable Ultra-low Power Module for Vision and Ranging Applications on Nano-drones". In: *Proceedings of the European Robotics Forum 2024*. Ed. by Cristian Secchi and Lorenzo Marconi. Cham, 2024, pp. 292–297.

[190] Mark W Mueller, Michael Hamer, and Raffaello D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation". In: *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1730–1736.

[191] Kanti V Mardia and Peter E Jupp. *Directional statistics*. John Wiley & Sons, 2009.

[192] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. "Learning Non-Maximum Suppression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[193] Teodora Popordanoska, Aleksei Tiulpin, and Matthew B. Blaschko. "Beyond Classification: Definition and Density-Based Estimation of Calibration in Object Detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2024, pp. 585–594.

[194] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[195] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[196] Adam K Taras, Niko Sünderhauf, Peter Corke, and Donald G Dansereau. “Inherently privacy-preserving vision for trustworthy autonomous systems: Needs and solutions”. In: *Journal of Responsible Technology* 17 (2024), p. 100079.

[197] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022.

[198] Randall Balestriero and Yann Lecun. “How Learning by Reconstruction Produces Uninformative Features For Perception”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 235. 2024, pp. 2566–2585.

[199] Carl Eckart and Gale Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[200] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results”. In: 2012.

[201] Z. Wang, E.P. Simoncelli, and A.C. Bovik. “Multiscale structural similarity for image quality assessment”. In: *Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. 2003, pp. 1398–1402.

[202] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 586–595.

[203] Dinsha Vinod and PS SaiKrishna. “Development of an autonomous fog computing platform using control-theoretic approach for robot-vision applications”. In: *Robotics and Autonomous Systems* 155 (2022), p. 104158.