ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in
COMPUTER SCIENCE AND ENGINEERING
Ciclo 31°

Settore Concorsuale di afferenza:	09/H1
Settore Scientifico-Disciplinare:	ING-INF/05

COMPUTER VISION AND DEEP LEARNING FOR RETAIL STORE MANAGEMENT

Presentata da: Alessio Tonioni

Coordinatore dottorato	Relatore
Chiar.mo Prof. Ing.	Chiar.mo Prof. Ing.
Paolo Ciaccia	Luigi Di Stefano

Esame finale anno 2019

Alessio Tonioni: *Computer Vision and Deep Learning for Retail Store Management,* Dottorato di ricerca in Computer Science and Engineering, © 2018

SUPERVISOR:
Luigi Di Stefano

LOCATION:
Bologna, Italy

ABSTRACT

The management of a supermarket or retail store is a quite complex process that requires the coordinated execution of many different tasks (*e.g.*, shelves management, inventory, surveillance, customer support...). Currently, the vast majority of those tasks still rely on human personnel that for this reason spend most of their working hours on repetitive and boring jobs. More often than not this will cause below expectancy efficiency that is obviously undesirable from the management perspective. Thank to recent advancements of technology, however, many of those repetitive tasks can be completely or partially automated easing and speeding up the job for sales clerks, improving the overall efficiency of the store, providing real-time analytics and resulting in potential additional services for the customers (*e.g.*, customized shopping experience). One of the key technology requirement shared between most of the aforementioned tasks is the ability to understand something specific about a scene based only on information acquired by a camera, for this reason, we will focus on how to deploy state of the art computer vision techniques to solve some management problems inside a grocery retail store. In particular, we will address two main problems: (a) how to detect and recognize automatically products exposed on store shelves and (b) how to obtain a reliable 3D reconstruction of an environment using only information coming from a camera. A good solution to (a) will be crucial to automate management tasks like visual aisle inspection or automatic inventory. At the same time it could drastically improve the customer experience with visual aid during the shopping, improved interaction via augmented reality or automatic assistance for visually impaired. We will tackle (a) both in a constrained version where the objective is to verify the compliance of observed items to a planned disposition, as well as an unconstrained one where no assumption on the observed scenes are considered. For both cases, we will show how to overcome some shortcoming resulting from naively applying state of the art computer vision techniques in this domain. As for (b), a good solution represents one of the first crucial steps for the development and deployment of low-cost autonomous agents able to safely navigate inside the store either to carry out management jobs or to help customers (*e.g.*, autonomous cart or shopping assistant). We believe that algorithms for depth prediction from stereo or mono camera are good candidates for the solution of this problem thanks to the low price of the required sensors, the great deployment flexibility and the achievable precision. However, the current state of the art algorithms for depth estimation from mono or stereo rely heavily on machine learning by deploying huge convolutional neural networks that take RGB images as input and output an estimation of the distance of each pixel from the camera. Those techniques might be hardly applied in the retail environment due to problems arising from the domain shift between data used to train them (usually synthetic images) and the deployment scenario (real indoor images). To overcome those limitations we will introduce techniques to adapt those algorithms to unseen environments without the need of acquiring costly ground truth data and, potentially, in real time as soon as the algorithm starts to operate on unseen scenes.

ACKNOWLEDGEMENTS

I would like to thank Centro Studi srl[1] for financing my Ph.D. and providing support for the development of some of the projects discussed in this work. I would like to thank my supervisor Prof. Luigi Di Stefano for all the time and effort spent mentoring me during the three years of my Ph.D.My thanks extend to Prof. Philip Torr for offering me the possibility to spend six months in the Torr Vision Group at the University of Oxford, a period that I consider fundamental for my growth as a researcher. Moreover, I would like to thank all the people with whom I have worked together during these years: Matteo Poggi, Fabio Tosi, Stefano Mattoccia, Pierluigi Zama Ramirez, Daniele De Gregorio, Riccardo Spezialetti, Samuele Salti, Federico Tombari, Eugenio Serra, Alioscia Petrelli, Tommaso Cavallari, Gianluca Berardi, Paolo Galeone, Oscar Rahnama, Tom Joy and Ajanthan Thalaiyasingam. Nevertheless, I would also like to thank all the other people that have worked at CVLab in Bologna and Torr Vision Group in Oxford for the insightful talks and discussions inside and outside the office. Finally, I would like to thank my family and friends for always supporting me during all my university and Ph.D. years. Last but not least, my greatest thanks goes to my girlfriend Arianna for helping me during my Ph.D. and always being there when I needed support during particularly stressful periods.

1 https://www.orizzontiholding.it/centro-studi/

CONTENTS

AUTHOR'S PUBLICATIONS

During the PhD period, the author contributed to the following publications. Research conducted during the development of some of them is integral part of this thesis.

[1] Alessio Tonioni and Luigi Di Stefano. 'Product recognition in store shelves as a sub-graph isomorphism problem'. In: *International Conference on Image Analysis and Processing*. Springer. 2017, pp. 682–693.

[2] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia and Luigi Di Stefano. 'Unsupervised adaptation for deep stereo'. In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 7. 2017, p. 8.

[3] Pierluigi Zama Ramirez, Alessio Tonioni and Luigi Di Stefano. 'Exploiting Semantics in Adversarial Training for Image-Level Domain Adaptation'. In: *International Conference on Image Processing, Applications and Systems*. 2018.

[4] Alessio Tonioni, Samuele Salti, Federico Tombari, Riccardo Spezialetti and Luigi Di Stefano. 'Learning to Detect Good 3D Keypoints'. In: *International Journal of Computer Vision* 126.1 (2018), pp. 1–20.

[5] Alessio Tonioni, Eugenio Serra and Luigi Di Stefano. 'A deep learning pipeline for fine-grained products recognition on store shelves'. In: *International Conference on Image Processing, Applications and Systems*. 2018.

[6] Oscar Rahnama, Tommaso Cavallari, Stuart Golodetz, Alessio Tonioni, Tom Joy, Luigi Di Stefano, Simon Walker and Philip H.S. Torr. 'Real-Time Highly Accurate Dense Depth on a Power Budget using an FPGA-CPU Hybrid SoC'. In: *Procedings of the IEEE International Symposium on Circuits and Systems*. 2019.

[7] Daniele De Gregorio, Alessio Tonioni, Gianluca Palli and Luigi Di Stefano. 'Semi-Automatic Labeling for Deep Learning in Robotics'. In: *Machine Vision and Applications*. under review.

[8] Alessio Tonioni and Luigi Di Stefano. 'Domain invariant hierarchical embedding for grocery products recognition'. In: *Computer Vision and Image Understanding* (under review).

[9] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia and Luigi Di Stefano. 'Unsupervised Domain Adaptation for Depth Prediction from Images'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. under review.

[10] Alessio Tonioni, Oscar Rahnama, Tom Joy, Luigi Di Stefano, Ajanthan Thalaiyasingam and Philip Torr. 'Learning to adapt for stereo'. In: *Conference on Computer Vision and Pattern Recognition*. under review.

[11] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattocia and Luigi Di Stefano. 'Real-time self-adaptive deep stereo'. In: *Conference on Computer Vision and Pattern Recognition*. under review.

1 INTRODUCTION

The management of grocery stores or supermarkets is a challenging task that requires many people constantly busy with the supervision of shelves and of the whole shop. Technology advances may help by providing more reliable information in real time to the store manager, so that he may coordinate in a more targeted way the human resources at his disposal. New technologies may not only lead to better and more effective management of the store, but also to improvements for the customers by giving to store owners the ability to provide better services without additional fixed costs like new employees. Since understanding an observed scene is one of the crucial premises for most tasks, in this work we are going to study how to develop, deploy and tune state of the art computer vision techniques to ease and partially automate some management tasks for stores. Some examples of such tasks are automatic inventory, store surveillance, aided shopping for visually impaired people, customer tracking and analytics. . . . In the following, we are going to address some of these tasks by developing computer vision systems based on state of the art techniques. We will spend quite some time discussing in details how we overcome some of the several problems arising from a naive deployment of general solutions in real applications. All the works featured in this thesis will share some common traits that we are going to briefly discuss here.

Firstly we work under the assumption of modifying existing stores and commercial practices as little as possible. By assuming this constraint we hope to achieve solutions viable from an economic perspective even for small-medium operators, that don't modify dramatically the current customer experience and that are more interesting from a research perspective. This assumption is, however, in direct contrast with the way industrial practices are currently evolving: the most successful proposals so far try to ease the management of a store by completely changing the way we build and organize them. Some practical examples of these solutions have already reached the commercial phase and are available across stores all around the world. For example, commercial outlets belonging to the *Decathlon* retail chain already includes on almost all the products on sale a disposable

RFID tag both to protect items from being stolen as well as to perform automatic recognition for inventory and fast checkout. This solution is very effective and fail-safe but can hardly be applied to grocery stores that may feature goods for which the selling price would be lower than the cost of the disposable RFID tag itself. Another famous example of disruptive technology in retails is the *Amazon Go* project[1], the first, and currently only, implementation of an almost completely autonomous grocery store. Inside, customers can benefit from a *checkout free* experience with no queue or cashier. By taking an item from a shelve and walking out of the store the customer agrees to purchase the item and automatically charge the cost on its bill. While potentially disruptive and intriguing, technological solutions like these require huge investments that are often not viable for the vast majority of retail chains. Moreover covering the whole store with camera and sensors to track the customers, as in the *Amazon Go* project, can be perceived as a quite invasive technology that can lead to potential loss of customers due to serious concerns regarding the user privacy during shopping. In contrast, all our proposals will not require ad-hoc stores or packaging nor additional sensors besides a camera. We believe that our proposals could be easily integrated with the current job of sales clerk blending unnoticed and without threatening the customer experience. Besides these extremes, in recent years some research and industrial solutions which rely on assumption closer to ours have started to emerge to solve some of the aforementioned problems, however, to the best of our knowledge, established scientific approaches have not emerged yet while the few commercial solutions seems either at a prototype stage or in the very early part of their life cycle.

The second common feature among all the researches presented in this thesis will be to solve problems arising in the retail scenario with the hope of proposing solutions to more general problems. For example one of the recurrent topics across this work will be the loss in performance due to *domain shift* between training and testing data for machine learning based algorithms. We faced this kind of issue when applying state of the art deep learning based algorithms to the solution of retail problems, however, the same problem arises across countless other applications and research fields. As such for most of the works we have tried to keep the presentation and the proposed solutions as general as possible, using the retail environment only as an exemplary test case rather than developing more ad-hoc and specific

1 https://www.amazon.com/b?ie=UTF8&node=16008589011

solutions. We believe that most of our proposals can be interesting from a more broad research audience rather than only for practitioners searching for a solution to specific retail related problems.

Among all the possible in-store problems addressable through computer vision techniques, we are going to focus mainly on two: automatic recognition of items on store shelves and obtaining a reliable 3D reconstruction of an environment for navigation purpose. We will now briefly introduce the two problems that will be more carefully analyzed in Part i and Part ii respectively.

1.1 AUTOMATIC DETECTION AND RECOGNITION OF ITEMS ON STORE SHELVES

The first task that we are going to analyze concerns the detection and recognition of all visible products on an image of a store shelf. To perform the recognition we assume the availability beforehand of one or more reference images depicting each one of the possible items on sale. We addressed the solution of this problem as it can be crucial to automate a series of low-level repetitive management tasks, like visual shelves inspections to detect out or low in stock products, while laying the foundations for a completely automatic inventory. At the same time, automate the recognition of products can be quite beneficial to offer new services to customers visiting the store, for example, allowing assisted shopping for visually impaired and being one of the building blocks for checkout free technologies. From a research point of view the recognition of products on shelves can be traced back to the more general case of object recognition, where we have one or more reference images of the sought object and a scene image where the objects may appear in different poses, such as rotated, translated, scaled, or even partially covered by other objects and under diverse illumination conditions. Analysis of products on shelves, though, may be regarded as a quite challenging case of the general visual object recognition problem. Firstly, it requires the simultaneous detection and recognition of the many different instances typically exposed on a store shelf; moreover, recognition must be carried out choosing among the thousands of possible items on sale on a store at any given times and at a quite fine instance classification level (*e.g.*, two different types of pasta, even from the same brand, must be recognized as different items). Finally, the *reference* images for each product are usually produced for marketing purpose and, therefore, exhibit ideal, but quite unrealistic, lighting and acquisition conditions. *In store* images fed to the recognition engine, instead, are usually acquired with lower quality equipment (*e.g.*, smartphone cameras) and in conditions far from ideal. As such a viable solution for the problem must somehow address this huge shift between the image available offline (*i.e.*, the high-quality *reference* ones) and the low-quality one acquired *in stores* where recognition should be carried out. The problem became even more prominent when relying on state of the art object detectors based on machine learning that will obviously face problems coming from the domain shift between train and test data.

In Chap. 3 we will first address a constrained version of this problem where we assume to know in advance the expected product disposition on store shelves, which is usually called **planogram**. This arrangement is carefully planned to maximize sales and keep customers happy, currently, however, verifying compliance of real shelves to the ideal layout is still mostly performed by store personnel. We try to automate this step by deploying an object recognition pipeline to verify if the observed portion of a shelve is compliant with the planned disposition or if there are missing and/or misplaced products, a task we will reefer as **planogram compliance check**. For the solution of this task, we deploy local invariant features together with a novel formulation of the product recognition problem as a sub-graph isomorphism between the items appearing in the given image and the ideal layout. This allows for auto-localizing the given image within the aisle or store and improving recognition dramatically.

In Chap. 4 we will address the more general problem of detecting items on store shelves without any assumption. To this end, we will deploy state of the art object detectors based on deep learning to obtain an initial product-agnostic items detection. Then, we pursue product recognition through a similarity search between global descriptors computed on *reference* product images and cropped (*query*) images extracted from the shelf picture. To maximize performance, we learn an ad-hoc global descriptor by a CNN[2] trained on *reference* images through an image embedding loss. Our system is computationally expensive at training time but can perform recognition rapidly and accurately at test time. We will also advocate why direct deployment of state of the art object recognition system based on classification is impractical, if not completely infeasible, for this specific task.

2 Convolutional Neural Network

Finally, in Chap. 5 we will focus on the recognition phase of the pipeline introduced above and explicitly address problems arising from the domain shift between *reference* images for products available beforehand and *query* images acquired in store. Inspired by recent advances in image retrieval, one-shot learning and semi-supervised domain adaptation, we propose an end-to-end architecture comprising a GAN[3] to address the domain shift at training time and a deep CNN trained on the samples generated by the GAN to learn an embedding for product images that enforces a hierarchy between product categories. At test time, we perform recognition by means of K-NN[4] search against a database consisting of one reference image per product. We will also show experimentally how our proposed solution generalizes fairly well to domains different than the retail one where might be useful to explicitly address the distribution shift between training and testing data.

1.2 RELIABLE 3D RECONSTRUCTION OF UNSEEN ENVIRONMENTS

While not immediately related to problems typically addressed in retail environments, the ability to correctly sense the 3D structure of an environment is one of the crucial building blocks for any applications that want to rely on autonomous agents moving inside an environment. Example of such agents inside a store could be automatic shopping carts, personalized robotic shopping assistants or autonomous platform for aisle inspection and management. All these systems should be able to correctly sense the 3D structure of the environment that surrounds them to navigate safely inside the store and to ease the interaction with customers or items exposed on the shelves. More in general, the availability of reliable 3D information could be helpful for other retail-oriented tasks such as the detection of out of stocks products on shelves or to allow virtual shopping and tour in a realistic 3D reconstruction of stores.

Among the available technologies for 3D sensing, we have mostly investigated the use of algorithms which relies only on one or two passive RGB cameras and ignored all the active alternatives. We have made this choice both due to the well-known shortcoming of active sensors when dealing with dark or highly reflective surfaces, being both quite common among the pattern on product packages, and for the cheapest cost and higher flexibility

3 Generative Adversarial Network
4 K Nearest Neighbour

offered by passive sensors. State-of-the-art to obtain accurate and dense depth measurements out of RGB images currently consists of training deep CNN models in end-to-end fashion on large amounts of data. Despite the outstanding performance achievable, these frameworks suffer from drastic drops in accuracy when running on unseen imagery different from those observed during training either in terms of appearance (synthetic vs real) or context (indoor vs outdoor). The effect due to this domain shift is usually softened by fine-tuning on smaller sets of images with depth labels, but in practice, it is not always possible, or it is extremely costly, to acquire such kind of data required to supervise the network. For example, one possible way to acquire reliable 3D labels mandate the use of costly Lidar[5] sensors, a careful calibration between sensors and, finally, a noise removal step from the sensor raw output. Unfortunately, as we will see, only a few selected environments and applications offer publicly available annotated dataset suitable for supervised fine-tuning, and none of these concerns a supermarket like environments. To overcome these shortcomings, we have mainly worked on methods to perform these adaptation steps effectively without requiring ground truth information. We have mostly tested our solutions on autonomous driving environments due to the public availability of datasets with precise ground-truth data for evaluation, but all the proposed solution can be generalized to any environments without changes.

In Chap. 8 we will introduce an effective off-line and unsupervised domain adaptation technique enabling to overcome the domain shift problem without requiring any ground-truth label. Relying on easier to obtain stereo pairs we deploy traditional non-learned stereo algorithms to produce disparity/depth labels and confidence measures to assess their degree of reliability. With these cues, we can fine-tune deep models by means of a confidence-guided loss function, neglecting the effect of possible outliers outcome of the stereo algorithms. Experimental results prove that our technique is effective to adapt models for both stereo and monocular depth estimations.

In Chap. 9 we address the domain shift problem by performing a continuous and unsupervised on-line adaptation of a deep stereo network to preserve good performance in challenging environments. Performing continuous training of a network can be extremely demanding on computational resources, however, we mitigate the requirements by introducing a new light-

5 Laser Imaging Detection and Ranging, active sensors that measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor.

weight yet effective deep stereo architecture and by developing an algorithm to train independently only sub-portions of the network. We will show how using both our new architecture and our fast adaptation schema we are able to continuously fine-tune a network to unseen environments at ~ 25FPS on a GPU, opening the way to widespread applications.

Finally in Chap. 10 we will present some ongoing work on how to deploy meta-learning techniques to speed up and make more efficient the adaptation process for depth prediction models. We will introduce a novel training schema to condition the initial weight configuration of a network to be more suitable to be adapted once exposed to unseen environments, *i.e.* it will require fewer adaptation steps to get optimal performance in an unseen environment. This will obviously help in the deployment of the online adaptation schema described before.

Part I

RECOGNITION OF PRODUCTS ON STORE SHELVES

2 INITIAL REMARKS

The arrangement of products in supermarket shelves is planned very carefully in order to maximize sales and keep customers happy. Shelves void, low in stock or misplaced products renders it difficult for the customer to buy what she/he needs, which, in turn, not only leads to unhappy shoppers but also to a significant loss of sales. As pointed out in [5], 31% of customers facing a void shelf purchase the item elsewhere and 11% do not buy it at all. The planned layout of products within shelves is called *planogram*: it specifies where each product should be placed within shelves and how many *facings* it should cover, that is how many packages of the same product should be visible in the front row of the shelf. Keeping shelves full as well as compliant to the planogram is a fundamental task for all types of stores. The key step to verify the compliance between the planned and observed layout is the recognition of products displayed on store shelves. If properly solved this task could be used for many different applications ranging from fast store management to improved customer experience inside the store. For these reasons, in this chapter we are going to address the problem of **visual shelf monitoring** through computer vision techniques and in particular the *automatic detection and recognition* of packaged products exposed on store shelves. A visual representation of the task we are trying to solve is depicted in Fig. 2.1

The seminal work on product recognition dates back to [13], where Merler *et al*. highlights the particular issues to be addressed in order to achieve a viable approach. First of all, the number of different items to be recognized is huge, in the order of several thousand for small to medium shops, well beyond the usual target for current state-of-the-art object detector based on image classifier. Moreover, product recognition can be better described as a hard instance recognition problem, rather than a classification one, as it deals with lots of objects looking remarkably similar but for small details (*e.g.*, different flavors of the same brand of cereals). Then, any practical methodology should rely only on the information available within existing commercial product databases, *i.e.* at most just one high-quality image for each side of the package, either acquired in studio settings or rendered (see

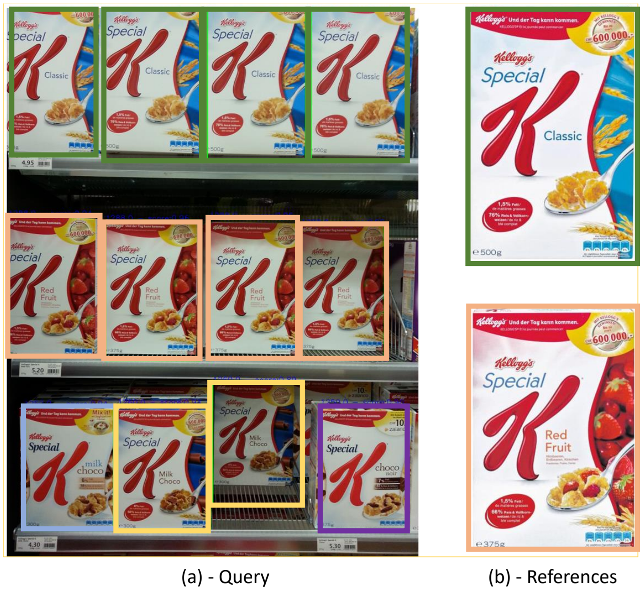


Figure 2.1: Illustration of the product recognition task with images from the Grocery Product dataset [39]. Given a *query* image featuring multiple products (a) identify regions associated with individual items and recognize the product enclosed in each region based on a database featuring only few (usually one) *reference* images per product (two examples are shown in (b)). Example of correct recognition are showed in (a) with bounding boxes colored according to the recognized classes.

Fig. 2.1-(b)). *Query* images for product recognition are, instead, taken in the store with cheap equipment (*e.g.*, a smart-phone) and featuring many different items displayed on a shelf (see Fig. 2.1-(a)).

To address all these problems we will propose two different object detection and recognition pipeline. Our first proposal will be discussed in Chap. 3 and leverage on local feature matching and geometric verification as the first step of a more complex pipeline to solve the specific task of *planogram* compliance check. In Chap. 4 we will present, instead, a more generic deep learning based pipeline for unconstrained product detection and recognition, while in Chap. 5 we will focus on the product recognition problem and we will introduce a deep learning framework to tackle some of the challenges it poses. But first in Sec. 2.1 we will cover most of the related work in the field of automatic detection and recognition of products on store shelves.

2.1 RELATED WORK

Grocery products recognition was firstly investigated in the already mentioned paper by Merler *et al.* [13]. Together with a thoughtful analysis of the problem, the authors propose a dataset and a system based on local invariant features to realize an assistive tool for visually impaired customers. They assume that no information concerning product layout may be deployed to ease detection. Given these settings, the performance of the proposed systems turned out quite unsatisfactory in terms of both precision and efficiency. Further research has then been undertaken to ameliorate the performance of automatic visual recognition of grocery products [21], [64], [36]. In particular, Cotter *et al.* [36] report significant performance improvements by leveraging on machine learning techniques, such as HMAX and ESVM, together with HOG-like features. Yet, their proposal requires many training images for each product, which is unlikely feasible in real settings and deploys a large ensemble of example-specific detectors, which makes the pipeline rather slow at test time. Moreover, adding a new type of sought product is rather cumbersome as it involves training a specific detector for each exemplary image, thereby also further slowing down the whole system at test time. The approach proposed in [36] was then extended in [48] through a contextual correlation graph between products. Such a structure can be queried at test time to predict the products more likely to be seen given the last k detections, thereby reducing the number of ESVM computed at test time and speeding up the whole system. A similar idea of deploying contextual information to ease the recognition phase was also deployed by [66]. More recently, Franco *et al.* [91] proposed a hierarchical multi-stage recognition pipeline that seems to be able to obtain remarkably good performance but has only been tested on small-scale problems regarding the detections of few tens of different items. Yet, all these recent papers focus on a relatively small-scale problem, *i.e.* recognition of a few hundred different items at most, whilst usually, several thousand products are on sale in a real shop George *et al.* [39] address more realistic settings and propose a multi-stage system capable of recognizing ~ 3400 different products based on a single model image per product. First, they carry out an initial classification to infer the macro categories of the observed items to reduce the recognition search space. Then, following detection, they run an optimization step based on a genetic algorithm to detect the most likely combination of products from a series of proposals. Despite the quite complex pipeline, when relying on only one model image per product the overall precision of the system is below 30%. The paper proposes also a publicly available dataset, referred to as *Grocery Products*, comprising 8350 product images classified into 80 hierarchical categories together with 680 high-resolution images of shelves.

The same large-scale realistic problem has been subsequently tackled by [86] using a standard local feature based recognition pipeline and an optimized Hough Transform to detect multiple object instances and filter out inconsistent matches, which brings in a slight performance improvement. More recently, [98] have shown how it is possible to improve detection and recognition performance on the same dataset relying on a probabilistic model based on local feature matching and refinement by a deep network.

3 PLANOGRAM COMPLIANCE CHECK

In this section, we will describe a computer vision pipeline that, given an expected product disposition (referred herein as **planogram**) and an image depicting supermarket shelves, can correctly localize each product, check whether the real arrangement is compliant to the planned one and detect missing or misplaced items. Key to our approach is a novel formulation of the problem as a sub-graph isomorphism between the product detected in the given image and those that should ideally be found therein given the planogram. Accordingly, our pipeline relies on a standard feature-based object recognition step, followed by a novel graph-based consistency check and a final refinement step to improve the overall product recognition rate.

3.1 RELATED WORKS

Beside works related to unconstrained product recognition already discussed in Sec. 2.1, it is worth mentioning the closely related work by Marder et al. [56] since it addresses the very same problem of checking *planogram compliance* through computer vision. Their approach relies on detecting and matching SURF features [11] followed by visual and logical disambiguation between similar products. To improve product recognition the authors deploy information dealing with the known product arrangement through specific hand-crafted rules, such as 'conditioners are placed on the right of shampoos'. Differently, we propose to deploy automatically these kinds of constraints by modeling the problem as a sub-graph isomorphism between the items detected in the given image and the planogram. Unlike ours, their method mandates a-priori categorization of the sought products into subsets of visually similar items. Systems to tackle the *planogram compliance* problem are described also in [38], [52] and [29]. These papers delineate solutions relying either on large sensor/camera networks or mobile robots monitoring shelves while patrolling aisles. In contrast, our proposal would require just an off-the-shelf device, such as a smartphone, tablet or hand-held computer.

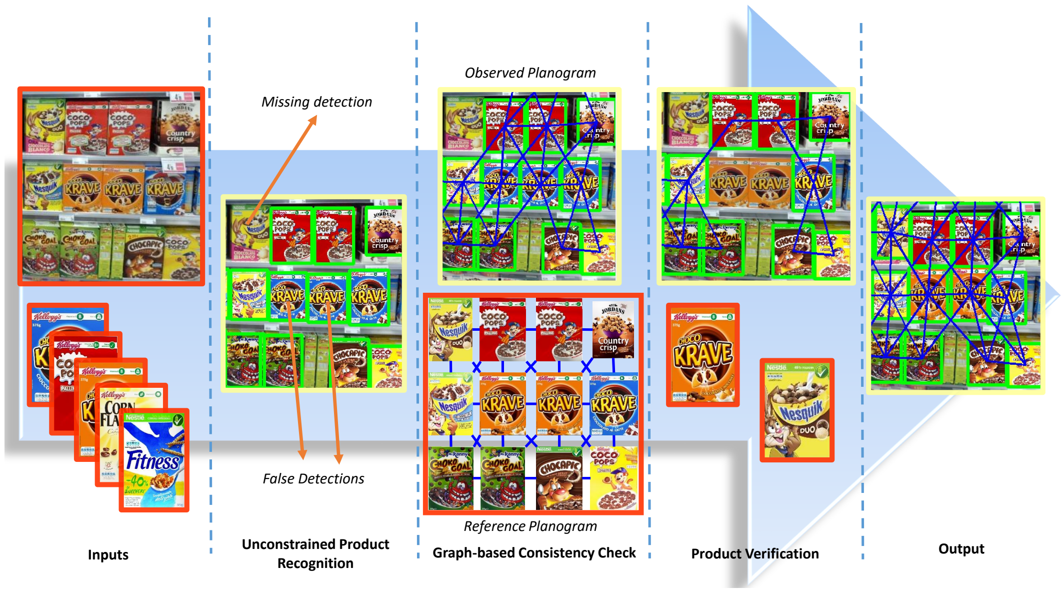


Figure 3.1: Overview of our pipeline. For each step we highlight the inputs and outputs through red and yellow boxes, respectively. Product detections throughout stages are highlighted by green boxes, while blue lines show the edges between nodes on the graphs enconding both the *Reference* and the *Observed* planograms.

3.2 PROPOSED PIPELINE

We address the typical industrial settings in which at least one model image per product together with a general schema of the correct disposition of items (the planogram) are available. At test time, given one image featuring products on shelves, the system would detect and localize each item and check if the observed product layout is compliant to the given planogram. As depicted in Fig. 3.1, we propose to accomplish the above tasks by a visual analysis pipeline consisting of three steps. We provide here an overview of the functions performed by the three steps, which are described in more detail in the following.

The first step operates only on model images and the given shelves image. Indeed, to pursue seamless integration with existing procedures, we assume that the information concerning which portion of the aisle is observed is not available together with the input image. Accordingly, the first step cannot deploy any constraint dealing with the expected product disposition and is thus referred to as **Unconstrained Product Recognition**. As most product packages consist of richly textured piecewise planar surfaces, we obtained promising result through a standard object recognition pipeline based on local invariant features (as described, *e.g.*, in [7]). Yet, the previously highlighted nuisances cause both missing product items as well as false detections due to similar products. Nonetheless, the first step can gather enough correct detections to allow the successive steps to identify the observed portion of the aisle in order to deploy constraints on the expected product layout and improve product recognition dramatically. The output of the first step consists of a set of bounding boxes corresponding to detected product instances (see Fig. 3.1).

From the second step, dubbed **Graph-based Consistency Check**, we start leveraging on the information about products and their relative disposition contained in planograms. We choose to represent a planogram as a *grid-like fully connected graph* where each node corresponds to a product facing and is linked to at most 8 neighbors at 1-edge distance, *i.e.* the closest facings along the cardinal directions. We rely on a graph instead of a rigid grid to allow for a more flexible representation; an edge between two nodes does not represent a perfect alignment between them but just proximity along that direction.

This abstract representation, referred to as *Reference Planogram*, encodes information about the number of facings related to each product and the items placed close together in shelves. An example of *Reference Planogram* is shown in Fig. 3.1. The detections provided by the first step are used in the second to automatically build another *grid-like graph* having the same structure as the *Reference Planogram* and referred to as *Observed Planogram*. Then, we find the *sub-graph isomorphism* between the *Observed* and *Reference* planograms, so as to identify local clusters of self-consistent detected products, *e.g.*, sets of products placed in the same relative position in both the *Observed* and *Reference* planograms. As a result, the second step ablates away inconsistent nodes from the *Observed Planogram*, which typically correspond to false detections yielded by the first step. It is worth pointing out that, as the *Observed Planogram* concerns the shelves seen in the current image while the *Reference Planogram* models the whole aisle, matching the former into the latter implies localizing the observed scene within the aisle[1].

After the second step the *Observed Planogram* should contain true detections only. Hence, those nodes that are missing compared to the *Reference Planogram* highlight items that appear to be missing w.r.t. the planned product layout. The task of the third step, referred to as **Product Verification**, is to verify whether these product items are really missing in the

1 More generally, matching the *Observed* to a set of *Reference* planograms does localize seamlessly the scene within a set of aisles or, even, the whole store.

scene or not. More precisely, we start considering the missing node showing the highest number of already assigned neighbors, for which we can most reliably determine a good approximation of the expected position in the image. Accordingly, a simpler computer vision problem than in the first step needs to be tackled, *i.e.* verify whether or not a known object is visible in a well-defined ROI (Region of Interest) within the image. Should the verification process highlight the presence of the product, the corresponding node would be added to the *Observed Planogram*, so to provide new constraints between found items; otherwise, a planogram compliance issue related to the checked node is reported (*i.e.*, missing/misplaced item). The process is iterated till all the facings in the observed shelves are either associated with detected instances or flagged as compliance issues.

3.2.1 *Unconstrained Product Recognition*

For the initial detection step, we rely on the classical multi-object and multi-instance object recognition pipeline based on local invariant features presented in [7], which is effective with planar textured surfaces and scales well to database comprising several hundreds or a few thousands models, *i.e.* comparable to the number of different products typically sold in grocery stores and supermarkets. Accordingly, we proceed through feature detection, description, and matching, then cast votes into a pose space by a Generalized Hough Transform that can handle multiple peaks associated with different instances of the same model in order to cluster correspondences and filter out outliers. In our settings, it turns out reasonable to assume the input image to represent an approximately frontal view of shelves, so that both in-plane and out-of-plane image rotations are small. Therefore, we estimate a 3 DOF pose (image translation and scale change).

Since the introduction of SIFT [7], a plethora of other feature detectors and descriptors have been proposed in the literature. Interestingly, the object recognition pipeline we implemented may be deployed seamlessly with most such newer proposals. Moreover, it turns out just as straightforward to rely on multiple types of features jointly to pursue higher sensitivity by detecting diverse image structures. Purposely, our implementation of the standard object recognition pipeline can run in parallel several detection/description/matching processes based on different features and have them eventually cast vote altogether within the same Hough pose space. In

Sec. 3.3, we will report an extensive experimental investigation to establish which features (or combination) would yield the best performance.

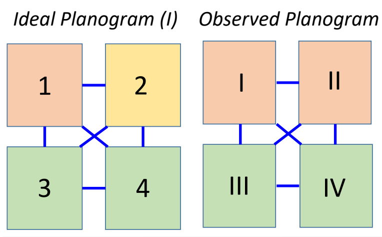
3.2.2 *Graph-based Consistency Check*

To build the *Observed Planogram* we first instantiate a node for each item detected in the previous step, then perform a loop over all detections to seek for link existing between detected instances. For each node, the search is performed along 8 cardinal directions (N, S, E, W, NW, NE, SW, SE) and, if another bounding box is found at a distance less than a dynamically determined threshold, an edge is created between the two nodes. In each node, the edge is labeled according to the search direction (*e.g.*, N) and oppositely in the found neighbor node (*e.g.*, S). The graph is kept self-coherent, *e.g.* if node B is the *North* node of A, then A must be the *South* node of B. In case of ambiguity, *e.g.* both A and C are found to be the *South* node of B, we retain only the edge between the two closest bounding boxes.

Once built, we compare the *Observed* to the *Reference Planogram* so to determine whether and how the two graphs overlap one to another. In theoretical computer science this problem is referred to as *subgraph isomorphism* and known to be NP-complete[10]. A general formulation may read as follows: given two graphs G and H, determine whether G contains a subgraph for which does exist a bijection between the vertex sets of G and H. However, given our strongly bounded graphs, we choose not to rely on one of the many general algorithms, like *e.g.* [20], and, instead, devised an ad hoc heuristic algorithm that, casting ours as a constraint satisfaction problem, works fairly well in practice.

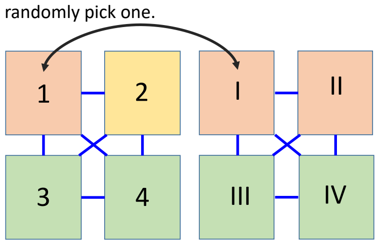
We formulate our problem as follows: given two graphs I (*Reference Planogram*) and O (*Observed Planogram*), find an isomorphism between a subset of nodes in I and a subset of nodes in O such that the former subset has the maximum feasible cardinality given product placements constraints. Each node in I can be associated with a node in O only if they both refer to the same product instance and exhibit coherent neighbors. In other words, we find the maximum set of nodes in graph O that turn out self-consistent, *i.e.* their relative positions are the same as in the reference graph I.

As illustrated in Algorithm Alg. 1, the process starts with procedure *CreateHypotheses*, which establishes the initial set of hypotheses, $\mathcal{H} = \{\dots h_i \dots\}$, $h_i = \{n_I, n_O, c(n_I, n_O)\}$, with n_I and n_O denoting, respectively, a node in I



$H = \{ [n_I^1, n_O^I, {}^2/_8], [n_I^3, n_O^{III}, {}^2/_8], [n_I^4, n_O^{IV}, {}^2/_8], [n_I^1, n_O^{II}, {}^1/_8], [n_I^3, n_O^{IV}, {}^1/_8], [n_I^4, n_O^{III}, {}^0/_8] \}$

a) Pick the best hypothesis and add it to the solution. In case more hypotheses have equal score randomly pick one.



$S = \{[n_I^1, n_O^I, {}^2/_8]\}$

b) Remove hypotheses and increase scores.

$H = \{ [n_I^1, n_O^I, {}^2/_8], [n_I^3, n_O^{III}, 1 + {}^2/_8], [n_I^4, n_O^{IV}, 1 + {}^2/_8], [n_I^1, n_O^{II}, {}^1/_8], [n_I^3, n_O^{IV}, {}^1/_8], [n_I^4, n_O^{III}, {}^0/_8] \}$

c) Compute B_C. If $B_c < C_{max}$ return C $= B_c$.

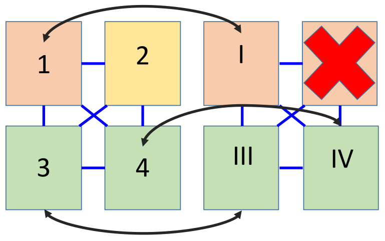
$H = \{ [n_I^3, n_O^{III}, {}^{10}/_8], [n_I^4, n_O^{IV}, {}^{10}/_8], [n_I^3, n_O^{IV}, {}^1/_8], [n_I^4, n_O^{III}, {}^0/_8] \}$

$S = \{ [n_I^1, n_O^I, {}^2/_8]\}$

$B_c = B(S, H) = 3$

d) Restart from step (**a**) until $H = \emptyset$ or $c(n_I, n_O) < \tau$ for all remaining hypotheses.

e) Compute C and return.



$S = \{ [n_I^1, n_O^I, {}^2/_8], [n_I^3, n_O^{III}, {}^{10}/_8], [n_I^4, n_O^{IV}, {}^{18}/_8]\}$

$C = C(S) = 3$

$h_0 = [n_I^1, n_O^I, {}^2/_8]$

Figure 3.2: Toy example concerning two small graphs with 4 nodes used to describe procedure *FindSolution*. The color of each node denotes the product the numbers within squares identify the different nodes in the text.

Algorithm 1 Find *sub-graph isomorphism* between I and O

```
$C_{max} \leftarrow 0$
$S_{best} \leftarrow \emptyset$
$\mathcal{H} \leftarrow \mathtt{CreateHypotheses}(I,O)$
while $\mathcal{H} \neq \emptyset$ do
    $C, \mathcal{S}, h_0 \leftarrow \mathtt{FindSolution}(\mathcal{H}, C_{max}, \tau)$
    if $C > C_{max}$ then
        $S_{best}, C_{max} \leftarrow S, C$
    $\mathcal{H} \leftarrow \mathcal{H} - h_0$
return $S_{best}, C_{max}$
```

and O related to the same product and $c(n_I, n_O) = \frac{nn_c}{nn_t}$ with nn_c number of coherent neighbors (*e.g.*, refering to the same product both in O and I) and nn_t number of neighbors for that node in I. *CreateHypotheses* iterates over all $n_I \in I$ so to instantiate all possible hypotheses. An example of the hypotheses set determined by *CreateHypotheses* given I and O is shown in the first row of Fig. 3.2. Then, procedure *FindSolution* finds a solution, $\mathcal{S}$, by iteratively picking the hypothesis featuring the highest score. The first hypothesis picked in the considered example is shown in Fig. 3.2-a). Successively, $\mathcal{H}$ is updated by removing the hypotheses containing either of the two nodes in the best hypothesis and increasing the scores of hypotheses associated with coherent neighbors (Fig. 3.2-b)). Procedure *FindSolution* returns also a confidence score for the current solution, C, which takes into account the cardinality of $\mathcal{S}$, together with a factor which penalizes the presence in O of disconnected sub-graphs that exhibit relative distances different than those expected given the structure of I[2] which instead is always fully connected. *FindSolution* takes as input the score of the current best solution, C_{max}, and relies on a branch-and-bound scheme to accelerate the computation. In particular, as illustrated in Fig. 3.2-c), after updating $\mathcal{H}$ (Fig. 3.2-b)), *FindSolution* calculates an upper-bound for the score, B_C, by adding to the cardinality of $\mathcal{S}$ the number of hypotheses in $\mathcal{H}$ that are not mutually exclusive, so as to early terminate the computation when the current solution can not improve C_{max}. The iterative process continues with picking the new best hypothesis until $\mathcal{H}$ is found empty or containing only hypotheses with confidence lower than a certain threshold (τ) (Fig. 3.2-d). The found solution, $\mathcal{S}$, contains all the hypotheses that are self-consistent and such that each node n_I is either associated with a node n_O or to none, as shown in the last row of Fig. 3.2. In the last step ((Fig. 3.2-e)), the procedure computes C and returns also the

2 In the toy example in Fig. 3.2, O does not contain disconnected sub-graphs.

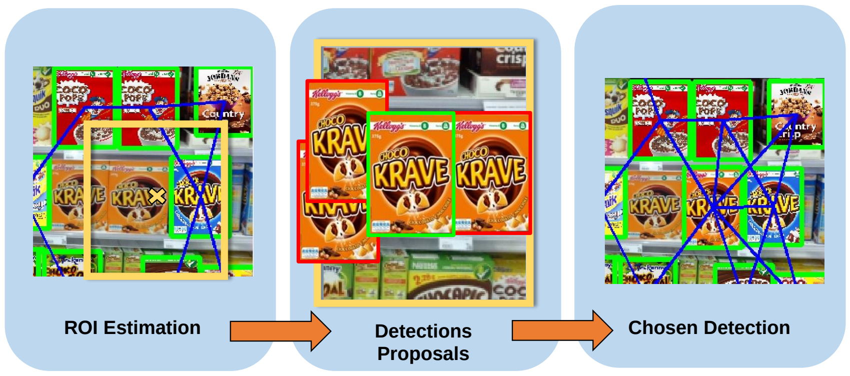


Figure 3.3: One iteration of our proposed Product Verification step. The estimated ROI is drawn in yellow. The correct proposal is highlighted in green while others are in red.

first hypothesis, h_0, that was added into $\mathcal{S}$, *i.e.* the one with the highest score $c(n_I, n_O)$ (Fig. 3.2.-a)). Upon returning from *FindSolution*, the algorithm checks whether or not the new solution $\mathcal{S}$ improves the best one found so far and removes h_0 from $\mathcal{H}$ (see Algorithm Alg. 1) to allow evaluation of another solution based on a different initial hypotheses.

As a result, Algorithm Alg. 1 finds self-consistent nodes in O given I, thereby removing inconsistent (*i.e.* likely false) detections and localizing the observed image wrt to the planogram. Accordingly, the output of the second steps contains information about which items appear to be missing given the planned product layout and where they ought to be located within the image.

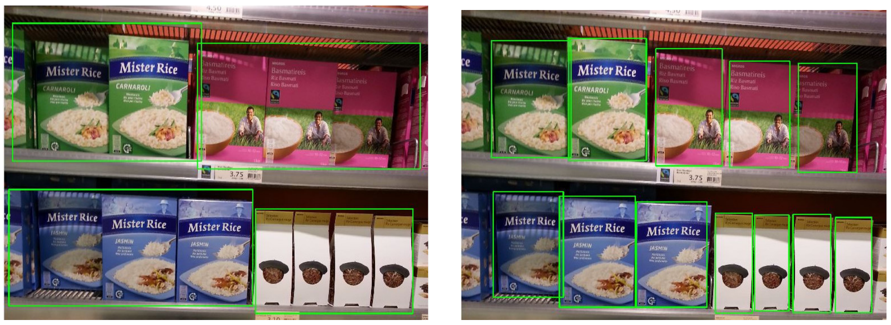
3.2.3 *Product Verification*

We use an iterative procedure whereby each iteration tries to fill the observed planogram with one seemingly missing object. As illustrated in Fig. 3.3, each iteration proceeds through three stages. We start with the missing element featuring the highest number of already detected neighbors. The positions of these neighbors provide clues on where the missing product should appear in the image. In particular, the position and size of each neighbor, together with the average edge length in the *Observed Planogram*, provide an estimation of the center of the missing element: averaging estimations across the neighbors yields a good approximate position. Then, we define a coarse image ROI centered at this position by estimating the size of the missing element[3] and allowing for some margin on account of possible localization inaccuracies.

Given the estimated ROI, the second stage attempts to find and localize the missing product therein. As already pointed out, unlike the initial step of our pipeline, here we now know exactly which product is sought as well as its approximate location in the image. To look for the sought product within the ROI, we have experimented with template matching techniques as well as with a similar pipeline based on local features as deployed for Unconstrained Product Recognition (Sec. 3.2.1). The latter, in turn, would favorably reuse the image features already computed within the ROI in the first step of our pipeline, so as to pursue matching versus the features associated with the model image of the sought product only and, accordingly, cast votes in the pose space. Both approaches would provide a series of Detection Proposals (see Fig. 3.3).

Detection proposals are analyzed in the last stage of an iteration by first discarding those featuring bounding boxes that overlap with already detected items and then scoring the remaining ones according to the coherence of the position within the (*Observed Planogram*) and the detection confidence. As for the first contribution to the score, we take into account the error between the center of the proposal and that of the ROI estimated in the first stage (so to favor proposals closer to the approximated position inferred from already detected neighbors); the second component of the score, instead, depends on the adopted technique: for template matching methods we use the correlation score while for approaches based on local features we rely on the number of correct matches associated with the proposal. Both terms are normalized to 1 and averaged out to get the final score assigned to each Detection Proposal. Based on such a score, we pick the best proposal and add it to the *Observed Planogram*, so as to enforce new constraints that may be deployed throughout successive iterations to select the best-constrained missing item as well as improve ROI localization. If either all detection proposals are discarded due to the overlap check or the best one exhibits too low a score, our pipeline reports a planogram compliance issue related to the currently analyzed missing product. We have not investigated yet on how to disambiguate between different issues such as low in stock items and misplaced items. In real settings, however, such different issues would both be dealt with by manual intervention of sales clerks. The iterative procedure

3 Store databases contain product sizes: the image size of a missing product can be estimated from those of the detected neighbors and the known metric sizes.



(a)-Customer[39] (b)-Management

Figure 3.4: Ground-truth dealing with product types provided with the *Grocery Products* dataset (a) and our instance-specific bounding boxes (b). In (a) a system should identify at least one instance for each product type, while in (b) it should find and correctly localize all the displayed product instances. We will refer to the two task in Chap. 4 as Customer use case and Management use case.

stops when all the seemingly missing products have been either detected or labeled as compliance issues.

3.3 EXPERIMENTAL RESULTS

To assess the performance of our pipeline we rely on the *Grocery Products* dataset [39]. However, as the ground-truth available with shelves images concerns product types while we aim at detecting each individual instance, we have manually annotated a subset of images with item-specific bounding boxes (see Fig. 3.4-b). Moreover, for each image, we have created an ideal planogram encoded in our graph-like representation for the perfect disposition of products (*e.g.*, if the actual image contains voids or misplaced items they will not be encoded on the ideal planogram that instead will model only the correct product disposition). The annotation used are available at our project page [4].

Our chosen subset consists of 70 images featuring box-like packages and dealing with different products such as rice, coffee, cereals, tea, juices, biscuits.... Each image depicts many visible products, for a total of 872 instances of 181 different products, that is on average ≈ 12 instances per image. According to the metric used in the PASCAL VOC challenge, we judge a detection as correct if the intersection-over-union between the detected and ground-truth bounding boxes is > 0.5. For each image we compute *Pre-*

4 vision.disi.unibo.it/index.php?option=com_content&view=article&id=111&catid=78

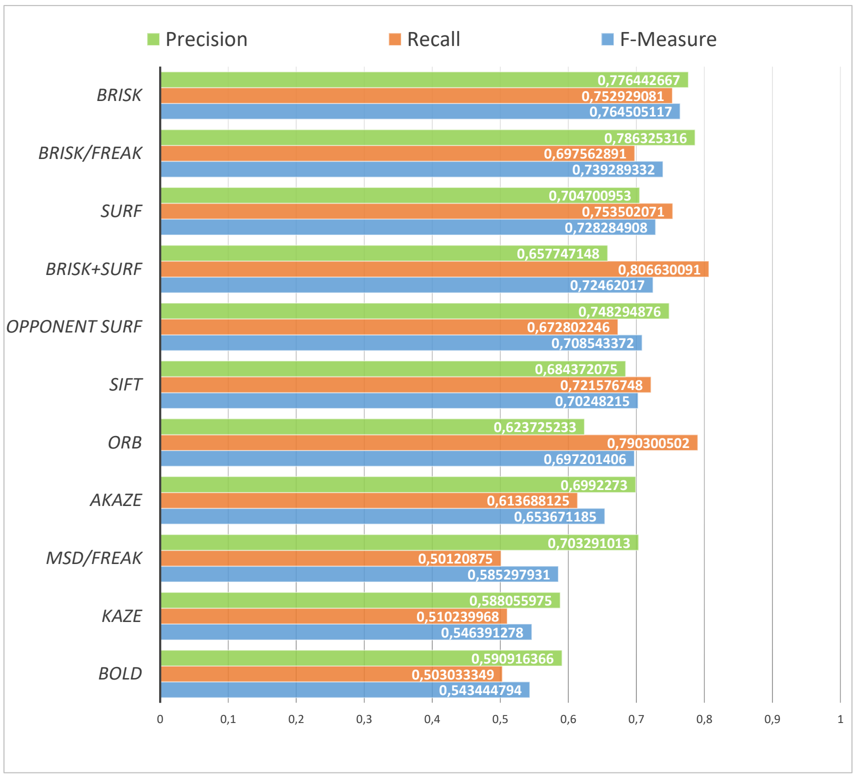


Figure 3.5: Evaluation of different features for **Unconstrained Product Recognition**. Results ordered from top to bottom alongside with *F-Measure* scores.

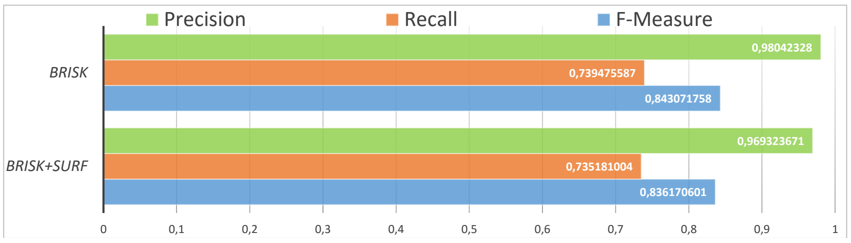


Figure 3.6: Results after **Graph-based Consistency Check** when using either BRISK or BRISK+SURF in the first step.

cision (number of correct detections over total number of detections), *Recall* (number of correctly detected products over number of products visible in the image) and *F-Measure* (harmonic mean of *Precision* and *Recall*). Then, we provide charts reporting average figures across the dataset.

We will now follow the processing flow along our pipeline so as to evaluate performance gain upon execution of each step. We start with evaluating the **Unconstrained Product Recognition** step, in order to find the best suitable local features to be used in this scenario.

We have tested all the detectors and descriptors available in OpenCV, *i.e.* SIFT [7], SURF [11], ORB [24], BRISK [23], KAZE [26], AKAZE [22], STAR [14], MSD [46], FREAK [25], DAISY [19], LATCH [72], Opponent Color

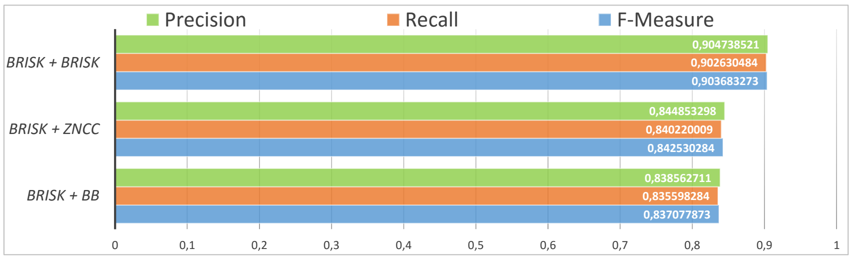


Figure 3.7: Evaluation of different choices for the final **Product Verification** step of our pipeline, with BRISK features used in the first step.

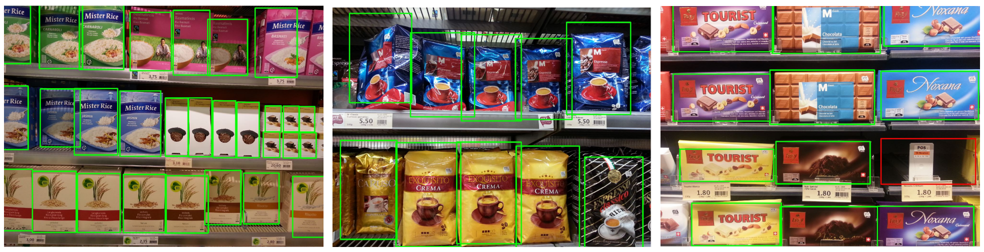


Figure 3.8: Qualitative results obtained by our pipeline: detected items are enclosed by green boxes while planogram compliance issues are highlighted by red boxes.

Space Descriptors [15], as well as the the line segments features known as BOLD[35] (original code distributed by the authors for research purposes). We have considered features providing both the detector and descriptor (*e.g.* SIFT) as well as many different detector/descriptor pairs (*e.g.* MSD/FREAK) and multiple feature processes voting altogether in the same pose space (*e.g.* BRISK+SURF). A summary of the best results is reported in Fig. 3.5. As it can be observed, binary descriptors, such as BRISK and FREAK performs fairly well in the addressed product recognition scenario, yielding the highest *Precision* and best *F-Measure* scores. SURF features provide good results alike, in particular as concerns *Recall*. It is also worth noticing how the use of multiple features, such as BRISK + SURF, to capture different image structures may help to increase the sensitivity of the pipeline, as vouched by the highest *Recall*. ORB features may yield a comparably high *Recall*, but at expense of a lower *Precision*. The use of color descriptors (Opponent SURF), instead, does not seem to provide significant benefits. As the second step is meant to prune out the false detections provided by the first, one would be lead to prefer those features yielding higher *Recall*. Yet, it may turn out hard for the second step to solve the *sub-graph isomorphism* problem in presence of too many false positives. Thus, a good balance between the two types of detection errors turns out preferable, rather. As such, we will consider both BRISK and BRISK+SURF features within the **Unconstrained Product**

Recognition step in order to further evaluate the results provided by our pipeline after the **Graph-based Consistency Check** step.

For the second step we fixed $\tau = 0.25$ and deployed the algorithm proposed in Sec. 3.2.2, the results are displayed in Fig. 3.6. First, the boost in *Precision* attained with both types of features compared to the output provided by the first step (Fig. 3.5) proves that the proposed *sub-graph isomorphism* formulation described in Sec. 3.2.2 is very effective to improve product recognition by removing false detections arising in unconstrained settings. In particular, when using BRISK features, *Precision* raises from $\approx$ 78% to $\approx$ 98% and with BRISK+SURF from $\approx$ 66% to $\approx$ 97%. Alongside, though, we observe a decrease in *Recall*, such as from $\approx$ 75% to $\approx$ 74% with BRISK and from $\approx$ 81% to $\approx$ 74% with BRISK+SURF. This is mostly due to items that, although detected correctly in the first step, cannot rely on enough self-coherent neighbors to be validated (*i.e.* $c(n_I, n_O) < \tau$). Overall, the **Graph-based Consistency Check** does improves performance significantly, as the *F-Measure* increases from $\approx$ 76% to $\approx$ 84% and from $\approx$ 72% to $\approx$ 84% with BRISK and BRISK+SURF, respectively.

Given that in Fig. 3.6 BRISK slightly outperforms BRISK+SURF according to all the performance indexes and requires less computation, we pick the former features for the first step and evaluate different design choices as regards the final **Product Verification**. In particular, as mentioned in Sec. 3.2.3, we considered different template matching and feature-based approaches. The best results, summarized in Fig. 3.7, concern template matching by the ZNCC (Zero-mean Normalized Cross Correlation) in the HSV color space, the recent *Best-buddies Similarity* method [51] in the RGB color space and a feature-based approach deploying the same features used for the first step, that is BRISK. As shown in Fig. 3.7, using BRISK features in both the first and last step does provide the best results, all the three performance indexes getting now as high as $\approx$ 90%.

Eventually, in Fig. 3.8 we present some qualitative results obtained by our pipeline both in case of compliance between the observed scene and the planogram as well as in the case of missing products.

4 UNCONSTRAINED PRODUCT DETECTION

After having addressed the problem of verifying planogram compliance, in this section, we are going to focus on the more general task of unconstrained product detection and recognition on store shelves. Due to the already discussed peculiarities of this scenario, unfortunately, the deployment of state of the art multi-class object detectors based on deep learning [59, 96, 111] cannot offers a trivial solution since all of them will require a large corpus of annotated images as similar as possible to the deployment scenario in order to provide good performance. Even acquiring and manually annotating with product labels a huge dataset of in-store images is not a viable solution due to the products on sale in stores, as well as their appearance, changing frequently over time, which would mandate continuous gathering of annotated in-store images and retraining of the system. Conversely, a practical approach should be trained once and then be able to handle seamlessly new stores, new products and/or new packages of existing products (*e.g.*, seasonal packages).

We propose to address product recognition by a pipeline consisting of three stages. Given a shelf image, we perform first a class-agnostic object detection to extract region proposals enclosing the individual product items. This stage relies on a deep learning based object detector trained to localize product items within images taken in the store; we will refer to this network as to the *Detector*. In the second stage, we perform product recognition separately on each of the region proposal provided by the *Detector*. Purposely, we carry out K-NN (K-Nearest Neighbours) similarity search between a global descriptor computed on the extracted region proposal and a database of similar descriptors computed on the *reference* images available in the product database. Rather than deploying a general-purpose global descriptor (*e.g.*, Fisher Vectors [17]), we train a CNN using the *reference* images to learn an image embedding function that maps RGB inputs to n-dimensional global descriptors amenable for product recognition; this second network will be referred to as to the *Embedder*. Eventually, to help prune out false detections and improve disambiguation between similarly looking products, in the third stage of our pipeline we refine the recognition output by re-ranking the first K proposals delivered by the similarity search.

It is worth pointing out how our approach needs samples of annotated in-store images only to train the product-agnostic *Detector*, which, however, does not require product-specific labels but just bounding boxes drawn around items. In Sec. 4.3 we will show how the product-agnostic *Detector* can be trained once and for all so to achieve remarkable performance across different stores despite changes in shelves disposition and product appearance. Therefore, new items/packages are handled seamlessly by our system simply by adding their global descriptors (computed through the *Embedder*) in the *reference* database. Besides, our system scales easily to the recognition of thousands of different items, as we use just one (or few) *reference* images per product, each encoded into a global descriptor in the order of a thousand float numbers.

Finally, while computationally expensive at training time, our system turns out light (*i.e.*, memory efficient) and fast at deployment time, thereby enabling near real-time operation. Speed and memory efficiency do not come at a price in performance, as our system compares favorably with respect to previous work on the standard benchmark dataset for product recognition.

4.1 RELATED WORKS

CNN-based systems are widely recognized as state-of-the-art across many object detection benchmarks. The different proposals can be broadly subdivided into two main families of algorithms based on the number of stages required to perform detection. On one hand, we have the slower but more accurate two stage detectors [59], which decompose object detection into a region proposal followed by an independent classification for each region. On the other hand, fast one stage approaches [96, 111] can perform detection and classification jointly. A very recent work has also addressed the specific domain of grocery products, so as to propose an ad hoc detector [107] that analyzes the image at multiple scales to produce more meaningful region proposals.

Besides, deploying CNNs to obtain rich image representations is an established approach to pursue image search, both as a strong off-the-shelf baseline [43] and as a key component within more complex pipelines [69].

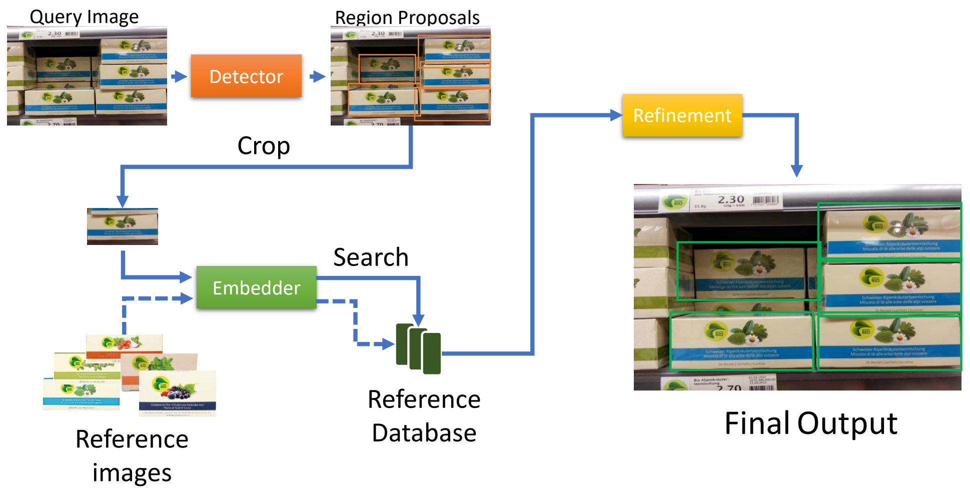


Figure 4.1: Schematic structure of our proposed product recognition pipeline. Dashed arrows denotes elaboration that can be performed offline just once since are not related to the *query* images.

Inspiration for our product recognition approach came from [49, 61]. In [61], Schroff *et al*. train a CNN using triplets of samples to create an embedding for face recognition and clustering while in [49] Bell *et al*. rely on a CNN to learn an image embedding to recognize the similarity between design products. Similarly, in the related field of fashion items recognition, relying on learned global descriptor rather than classifiers is an established solution shared among many recent works [47, 53, 115].

4.2 PROPOSED APPROACH

Fig. 4.1 shows an overview of our proposed pipeline. In the first step, described Sec. 4.2.1, a CNN (*Detector*) extracts region proposals from the *query* image. Then, as detailed in Sec. 4.2.2, each region proposal is cropped from the *query* image and sent to another CNN (*Embedder*) which computes an ad-hoc image representation. These will then be deployed to pursue product recognition through a K-NN similarity search in a database of representations pre-computed off-line by the *Embedder* on the *reference* images. Finally, as illustrated in Sec. 4.2.3, we combine different strategies to perform a final refinement step which helps to prune out false detections and disambiguate among similar products.

4.2.1 *Detection*

Given a *query* image featuring several items displayed in a store shelf, the first stage of our pipeline aims at obtaining a set of bounding boxes to be used as region proposals in the following recognition stage. Ideally, each bounding box should contain exactly one product, fit tightly the visible package and provide a confidence score measuring how much the detection should be trusted.

State-of-the-art CNN-based object detectors may fulfill the above requirements for the product recognition scenario, as demonstrated in [107]. Given an input image, these networks can output several accurate bounding boxes, each endowed by confidence and a class prediction. To train CNN-based object detectors, such as [59, 96, 111], a large set of images annotated with the position of the objects alongside with their class labels is needed. However, due to the ever-changing nature of the items sold in stores, we do not train the *Detector* to perform predictions at the fine-grained class level (*i.e.*, at the level of the individual products), but to carry out a product-agnostic item detection. Accordingly, the in-store training images for our *Detector* can be annotated for training just by drawing bounding-boxes around items without specifying the actual product label associated with each bounding-box. This formulation makes the creation of a suitable training set and the training itself easier and faster. Moreover, by training the *Detector* to recognize *generic products* from everything else we hope to achieve a solution that is more general therefore deployable across different stores and products. Training a CNN to directly perform detection and recognition would require a much more expensive and slow image annotation process which should be carried out, again and again, to keep up with changes of the products/packages to be recognized. This continuous re-training of the *Detector* is just not feasible in any practical settings.

4.2.2 *Recognition*

Starting from the candidate regions delivered by the *Detector*, we perform recognition by means of K-NN similarity search between a global descriptor computed on each candidate region and a database of similar descriptors (one for each product) pre-computed off-line on the *reference* images. Recent works (*e.g.*, [43]) have shown that the activations sampled from layers of pre-trained CNNs can be used as high quality global image descriptors. [47] extended this idea by proposing to train a CNN (*i.e.*, the *Embedder*) to learn a function $E : \mathcal{I} \rightarrow \mathcal{D}$ that maps an input image $i \in I$ into a k-dimensional descriptor $d^k \in \mathcal{D}$ amenable to recognition through K-NN search. Given a set of images with associated class labels, the training is performed sampling triplets of different images, referred to as *anchor* (i_a), *positive* (i_p) and *negative* (i_n), such that i_a and i_p depict the same class while i_n belongs to a different one. Given a distance function in the descriptor space, $d(\mathbf{X}, \mathbf{Y})$, with $X, Y \in \mathcal{D}$, and denoted as $E(i)$ the descriptor computed by the the *Embedder* on image i, the network is trained to minimize the so called *triplet ranking loss*:

$$\mathcal{L} = \max(0, d(E(i_a), E(i_p)) - d(E(i_a), E(i_n)) + \alpha) \tag{4.1}$$

with α a fixed margin to be enforced between the pair of distances. Through minimization of this loss, the network learns to encode into nearby positions within $\mathcal{D}$ the images depicting items belonging to the same class, whilst keeping items of different classes sufficiently well separated.

We use the same formulation and adapt it for the context of grocery product recognition where different products corresponds to different classes (*e.g.* the two reference images of Fig. 2.1-(b) belong to different classes and could be used as i_p and i_n). Unfortunately, we can not sample different images for i_a and i_p due to available commercial datasets featuring just a single exemplar image per product (*i.e.*, per class). Thus, to create the required triplet, at each training iteration we randomly pick two products and use their *reference* images as i_p and i_n. Then, we synthesize a new i_a from i_p by a suitable data augmentation function $A : \mathcal{I} \rightarrow \mathcal{I}$, to make it similar to *query* images (*i.e.*, $i_a = A(i_p)$)[1].

To perform recognition, firstly, the *Embedder* network is used to describe each available *reference* image i_r by a global descriptor $E(i_r)$ and thus create the *reference* database of descriptors associated with the products to be recognized. Then, when a *query* image is processed, the same embedding is computed on each of the candidate regions, i_{pq}, cropped from the *query* image, i_q, so to get $E(i_{pq})$. Finally, for each i_{pq} we compute the distance in the embedding space with respect to each *reference* descriptor, denoted as $d(E(i_{pq}), E(i_r))$, in order to sift-out the first K-NN of $E(i_{pq})$ in the *reference* database. These are subject to further processing in the final refinement step.

1 The details concerning the adopted augmentation function are reported in Sec. 4.3

4.2.3 *Refinement*

The aim of the final refinement is to remove false detections and re-rank the first K-NN found in the previous step in order to fix possible recognition mistakes.

Since the initial ranking is obtained comparing descriptors computed on whole images, a meaningful re-ranking of the first K-NN may be achieved by looking at peculiar image details that may have been neglected while comparing global descriptors and yet be crucial to differentiate a product from others looking very similar. Therefore, we describe both the *Query* and each of the first K-NN *reference* images by a set of local features $F_1, F_2, ..., F_k$, each consisting in a spatial position (x_i, y_i) within the image and a compact descriptor f_i. Given these features, we look for similarities between descriptors extracted from *query* and *reference* images, to compute a set of matches. Matches are then weighted based on the distance in the descriptor space, $d(f_i, f_j)$ and a geometric consistency criterion relying on the unit-norm vector, $\vec{v}$, from the spatial location of a feature to the image center. In particular, given a match, $M_{ij} = (F_i^q, F_j^r)$, between feature i of the *query* image and feature j of the *reference* image, we compute the following weight:

$$W_{ij} = \frac{(\vec{v}_i^q \cdot \vec{v}_j^r) + 1}{d(f_i^q, f_j^r) + \epsilon} \tag{4.2}$$

where $\cdot$ marks scalar products between vectors and ϵ is a small number to avoid potential division by zero. Intuitively W_{ij} is bigger for matching features which share the same relative position with respect to the image center (high $(\vec{v}_i^q \cdot \vec{v}_j^r)$) and have descriptors close in the feature space (small $d(f_i^q, f_j^r)$). Finally, the first K-NN are re-ranked according to the sum of the weights W_{ij} computed for the matches between the local features. In Sec. 4.3.2 we will show how good local features can be obtained at zero computational cost as a by-product of our learned global image descriptor. This refinement technique will be referred to as ***+lf***.

A simple additional refinement step consists in filtering out wrong recognitions by the *distance ratio* criterion [7] (*i.e.*, by thresholding the ratio of the distances in feature space between the *query* descriptor and its 1-NN and 2-NN). If the ratio is above a threshold, τ_d, the recognition is deemed as ambiguous and discarded. In the following, we will denote this refinement technique as ***+th***.

Finally, as commercial product databases typically provide a multilevel classification of the items (*e.g.*, at least instance, and category level), we propose a re-ranking and filtering method specific to the grocery domain where, as pointed out by [39], products belonging to the same macro category are typically displayed close one to another on the shelf. Given the candidate regions extracted from the *query* image and their corresponding sets of K-NN, we consider the 1-NN of the region proposals extracted with high confidence (> 0.1) by the *Detector* in order to find the main macro category of the image. Then, in case the majority of detections votes for the same macro category, it is safe to assume that the pictured shelf contains almost exclusively items of that category thus filter the K-NN for all candidate regions accordingly.It is worth observing how this strategy implicitly leverages on those products easier to identify (*i.e.*, the high-confidence detections) to increase the chance to correctly recognize the harder ones. We will refer to this refinement strategy as to ***+mc***.

4.3 EXPERIMENTAL RESULTS

To validate the performance of our product recognition pipeline we take into account two possible use cases dealing with different final users:

- **Customer**: the system should be deployed for a guided or partially automated shopping experience (*e.g.*, product localization inside the shop, augmented reality overlays or support to visually impaired). As proposed in [39] the goal is to detect *at least one* instance of each visible type of product displayed in a shelf picture.
- **Management**: the system will be used to partially automate the management of a shop (*e.g.*, automatic inventory, and restocking). Here, the goal is to recognize *each* product instance displayed on the shelve.

4.3.1 *Datasets and Evaluation Metrics*

For our experimental evaluation, we rely on the publicly available *Grocery Products* dataset [39], which features more than 8400 grocery products organized in hierarchical classes and with each product described by exactly one *reference* image. The dataset contains also 680 in-store (*query*) images that display items belonging to the *Food* subclass of the whole dataset. The annotations released by the authors for the *query* images allow for evaluating performance in the *Customer* use case, as they consist in bounding boxes drawn around spatial clusters of instances of the same products. To test our system also in the *Management* use case, we deploy the annotations that we have presented in Sec. 3.3. Fig. 3.4 shows examples of the two kinds of annotations used to evaluate the system in the two different use cases.

To compare our work with previously published results we use the metrics proposed by the authors in [39]: mean average precision-*mAP* (the approximation of the area under the Precision-Recall curve for the detector) and Product Recall-*PR* (average product recall across all the test image). As for scenario (a), we report also mean average multi-label classification accuracy-*mAMCA*.

To train the *Detector* we acquired multiple videos with a tablet mounted on a cart facing the shelves and manually labeled a subset of sampled frames to create a training set of 1247 images. Thus, our videos are acquired in different stores with respect to those depicted in the *Grocery Products* dataset and feature different products on different shelves, vouching for the generalization ability of our system.

4.3.2 *Implementation Details*

For all our tests we have used as *Detector* the state-of-the-art one-stage object detector known as yolo_v2 (shortened in *yolo*) [111]. We choose this network as it grants real-time performance on a GPU and for the availability of the original implementation. Starting from the publicly available weights[2], we have fine-tuned the network on our 1247 in-store images for 80000 steps keeping the hyperparameters suggested by the original authors.

The backbone network for our *Embedder* is a VGG_16 [62] pre-trained on the Imagenet-1000 classification task (weights publicly available[3]). From this network we obtain global image descriptors by computing *MAC* features [83] on the conv4_3 convolutional layer and applying L2 normalization to obtain unit-norm embedding vectors. To carry out the comparison between descriptors, both at training and test time, we used as distance function

2 https://github.com/pjreddie/darknet
3 https://github.com/tensorflow/models/tree/master/research/slim

$d(X, Y) = 1 - X \cdot Y$ with $X, Y \in \mathcal{D}$ (*i.e.*, 1 minus the cosine similarity between the two descriptors). The motivation for the choice of network, layers, and encoding will be discussed in details in Sec. 5.4. To better highlight the advantage of learning an ad-hoc descriptor in the following we will report experiments using general purpose descriptors obtained without fine tuning the *Embedder* with the suffix $\_gd$ (general descriptor), while descriptors obtained after fine-tuning as described in Sec. 4.2.2 will be denoted by the suffix $\_ld$ (learned descriptor). To train the *Embedder* we use the *reference* images of *Grocery Products* dealing with the products belonging to the *Food* subclass (*i.e.*, 3288 different product with exactly one training image each). To enrich the dataset and create the anchor images, i_a, we randomly perform the following augmentation functions A: blur by a Gaussian kernel with random σ, random crop, random brightness and saturation changes. These augmentations were engineered so to transform the *reference* images in a way that renders them similar to the proposals cropped from the *query* images. The hyper-parameters obtained by cross-validation for the training process are as follows: $\alpha = 0.1$ for the triplet loss, learning rate $lr = 0.000001$, ADAM optimizer and fine-tuning by 10000 steps with batch size 24.

We propose a novel kind of local features for the *+lf* refinement: as *MAC* descriptors are obtained by applying a max-pool operation over all the activations of a convolutional layer, by changing the size and stride of the pool operation it is possible to obtain a set of local descriptor with the associated location being the center of the pooled area reprojected into the original image. By leveraging on this intuition, we can obtain in a single forward computation both a global descriptor for the initial K-NN search (Sec. 4.2.2) as well as a set of local features to be deployed in the refinement step (Sec. 4.2.3). For our test we choose kernel size equal 16 and stride equals 2 obtaining 64 features per *reference* image.

4.3.3 *Customer Use Case*

In this section, we evaluate the effectiveness of our system in the *Customer* scenario. To measure performance we rely on the annotations displayed in Fig. 3.4-(a) and score a correct recognition when the product has been correctly identified and its bounding box has a non-empty intersection with that provided as ground-truth. We compare our method with already published work tested on the same dataset: FV+RANSAC (Fisher Vector classifica-

Method	**mAP(%)**	**PR(%)**	**mAMCA(%)**
FV+RANSAC[39]	11.26	23.14	6.41
RF+PM+GA[39]	23.49	43.13	21.19
FM+HO[86]	23.71	41.60	**32.50**
yolo_gd	21.49	47.03	13.34
yolo_ld	27.84	53.17	16.32
yolo_ld+th	30.46	37.88	28.74
yolo_ld+lf	32.34	**58.41**	17.72
yolo_ld+mc	30.15	55.25	21.09
yolo_ld+lf-mc-th **(full)**	**36.02**	57.07	31.57

Table 4.1: Product recognition on the *Grocery Products* dataset in the *Customer* scenario. Best result highlighted in bold. Our proposals (in italic) yields large improvements in terms of both mAP and PR with respect to previously published results.

tion re-ranked with RANSAC) [39], RF+PM+GA (Category prediction with Random Forests, dense Pixel Matching and Genetic Algorithm) [39] and FM+HO (local feature matching optimized with Hough) [86]. We report the results obtained by the different methods according to the tree metrics in Tab. 4.1. As [86] does not provide the mAP figure but only the values of precision and recall, for FM+HO we report an approximate mAP computed by multiplying precision and recall.

Using our trained *yolo* network for product detection, in Tab. 4.1 we report the results obtained by either deploying a general purpose VGG-based descriptor ($yolo_{gd}$) or learning an ad-hoc embedding for grocery products $yolo_{ld}$. Moreover, we report the results achieved with the different refinement strategies presented in Sec. 4.2.3.

Tab. 4.1 shows that our pipeline can provide a higher recall than previously proposed methods even with a general purpose image descriptor (*yolo_gd*), although with a somehow lower precision, as demonstrated by the slightly inferior mAP score. However, our complete proposal relies on learning an ad-hoc descriptor for grocery products (*yolo_ld*), which yields a significant performance improvement, as vouched by an average gain of about 6% in terms of both Recall and mAP. Wrongly classified proposals can be discarded to further improve accuracy by the threshold refinement strategy (*yolo_ld* + *th* - with $\tau_d = 0.9$), thereby increasing the mAMCA from 16.32% to 28.74%. Re-ranking based on the proposed local features (*yolo_ld+lf*)

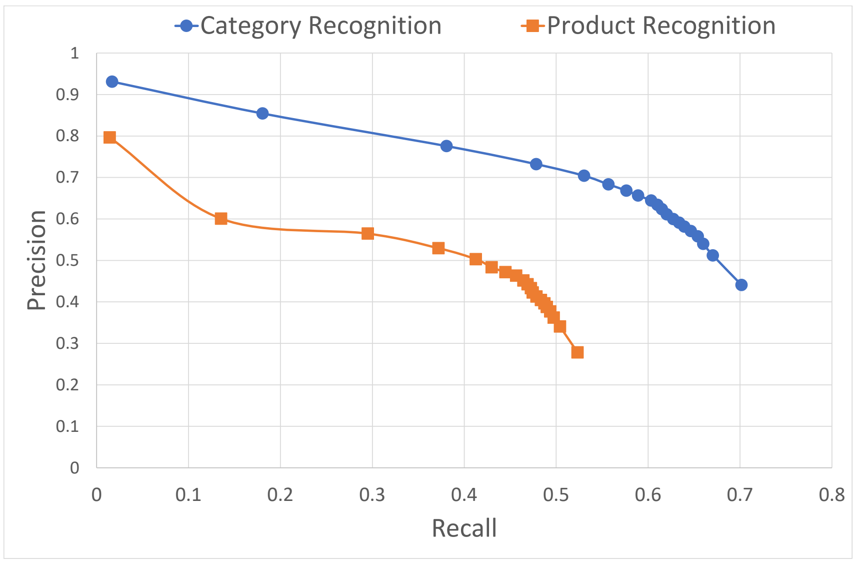


Figure 4.2: Precision-recall curves obtained in the *customer* use case by the *yolo_ld* system when trying to recognize either the individual products or just their category.

turns out an effective approach to ameliorate both precision and recall, as demonstrated by a gain of about 5% in mAP and PR with respect to the pipeline without final refinement (*yolo_ld*). The category-based re-ranking strategy (*yolo_ld* + *mc*) seems to fix some of the recognition mistakes and improve the recognition rate with respect to (*yolo_ld*), providing gains in all metrics. Finally, by mixing all the refinement strategies to obtain our overall pipeline (*yolo_ld+lf-mc-th*), we neatly get the best trade-off between precision and recall, as vouched by the 57.07% PR and 36.02% mAP, *i.e.* about 14% and 12.5% better than previously published results, respectively, with a mAMCA turning out nearly on par with the best previous result.

We found that casting recognition as a similarity search through learned global descriptors can provide a multi-level classification for free. For example, even when the 1-NN does not correspond to the right product, it usually corresponds to items belonging to the correct macro category(*i.e.* cereals, coffee,. . .). We believe this behavior being due to items belonging to the same macro class sharing similar peculiar visual patterns that are effectively captured by the descriptor and help to cluster nearby items belonging to the same categories (*e.g.*, coffee cups often displayed on coffee packages or flowers on infusions). To highlight this generalization property, we perform here an additional test in the Customer scenario by considering a recognition as correct if the category of the 1-NN match is the same as those of the annotated bounding box. Accordingly, we compare the performance

Method	mAP(%)	PR(%)
FS	66.37	75.0
yolo_gd	66.95	78.89
yolo_ld	74.32	84.75
yolo_ld+th	75.62	81.55
yolo_ld+lf	76.37	**86.56**
yolo_ld+mc	74.80	85.28
yolo_ld+lf-mc-th **(full)**	**76.93**	85.71

Table 4.2: Product recognition for *Management* use case. Our proposal highlighted (in italic), best results in bold.

Method	mAP(%)	PR(%)
FS	47.32	57.0
yolo_gd	60.17	73.66
yolo_ld	67.88	80.27
yolo_ld+th	69.70	76.01
yolo_ld+lf	70.69	**82.83**
yolo_ld+mc	69.01	81.55
yolo_ld+lf-mc-th **(full)**	**73.50**	82.66

Table 4.3: Results in the *Management* use case performing recognition against all the items belonging to the Food subclass of the *Grocery Products* dataset (~ 3200). Our proposals highlighted (in italic), best results in bold.

of *yolo_ld* when trying to recognize either the individual products or their category. The results of this experiment are reported as Precision-Recall curves in Fig. 4.2. The large difference between the two curves proves that very often the system mistakes items at the product level but correctly recognizing their category. Eventually, it is worth pointing out that our method not only provides a significant performance improvement with respect to previously published results but turns out remarkably fast. Indeed, our whole pipeline can be run on a GPU in less than one second per image.

Management Use Case

The experiments presented in this Section concern the *Management* use case where we are required to correctly detect and recognize all the individual

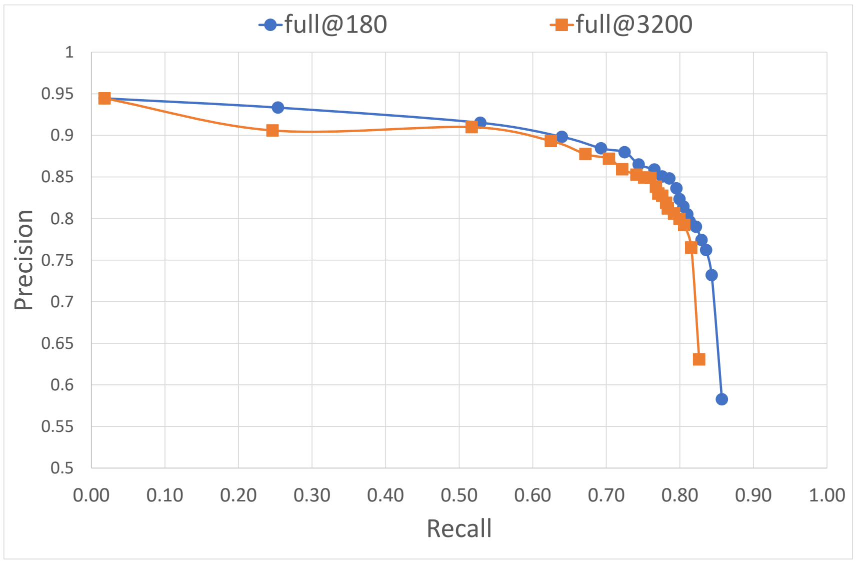
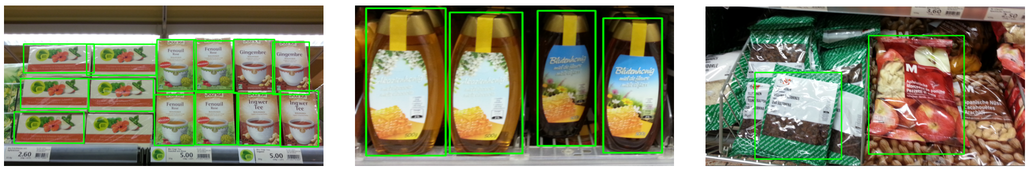


Figure 4.3: Precision-Recall curves for our full pipeline in the *Management* use case. full@180 denotes performing recognition on the small *reference* database of Chap. 3 (~ 180 entries), full@3200 against all the products in the *Food* category of *Grocery Products* (~ 3200).



(a) (b) (c)

Figure 4.4: Examples of correct product recognitions in *query* images from *Grocery Products*.

products displayed on shelves. Thus, we rely on the annotations pictured in Fig. 3.4-(b) and consider recognition as correct when the item has been correctly recognized and the intersection over union (IoU) between the predicted and ground truth bounding boxes is higher than 0.5.

We compare our new proposal to the most effective configuration of the first stage of the previous pipeline presented in Chap. 3, referred to hereinafter as *FS*, which is based on matching BRISK local features[23] followed by Hough Voting and pose estimation through RANSAC. We compare our new proposal only to the first stage of FS because we are interested only on unconstrained product recognition without considering the availability of planogram to ease the detection. To compare our two pipelines, we perform recognition against the smaller *reference* database of 182 products used in Chap. 3. The results are reported in Tab. 4.2. Firstly, it is worth pointing out how, despite the task being inherently more difficult than in the *Customer* use case, we record higher recognition performance. We ascribe this mainly to the smaller subset of in-store images used for testing, (*i.e.*, 70 vs. 680) as well as to these images featuring mainly rigid packaged products, which are easier to recognize. Once again, the use of a learned descriptor (*yolo_ld*) provides a substantial performance gain with respect to a general purpose descriptor (*yolo_gd*), as the mAP improves from 66.95% to 74.32% and the PR from 78.89% to 84.75%. The different refinement strategies provide advantages similar to those discussed in Sec. 4.3.3, the best improvement yielded by re-ranking recognitions based on the local features extracted from the *Embedder* network (*yolo_ld+lf*). The optimal trade-off between precision and recall is achieved again by deploying together all the refinement strategies (*yolo_ld+lf-mc-th*), which provided a mAP and PR as high as 76.93% and 84.75%, respectively (*i.e.*, both about 10% better than the previously published results on this dataset).

Tab. 4.3 reports results aimed at assessing the scalability of the methods with respect to the number of products in the *reference* database. We carried out an additional experiment by performing the recognition of each item detected within the 70 *query* images against all the 3200 products of the "Food" category in *Grocery Products* rather than the smaller subset of 182 products considered before. By comparing the values in Tab. 4.2 and Tab. 4.3, we can observe how, unlike *FS*, this pipeline can scale nicely from few to thousands of different products: our full method *yolo_ld+lf-mc-th* looses only 3.43% mAP upon scaling-up the *reference* database quite significantly, whilst the performance drop for *FS* is as relevant as 19.05%. In Fig. 4.3 we also plot the precision-recall curves obtained by our full pipeline (*yolo_ld+lf-mc-th*) using the smaller (full@180) and larger (full@3200) sets of reference products. The curves show clearly how our pipeline can deliver almost the same performance in the two setups, which vouches for the ability of our proposal to scale smoothly to the recognition of thousands of different products. As far as recognition time is concerned, our pipeline can scale fairly well regardless of the size of the *reference* database, due to the NN search, even if extensive, amounting to a negligible fraction of the overall computation: the difference in inference time between recognizing 180 and 3200 product is less than a tenth of a second.

4.3.4 *Qualitative Results*

Fig. 4.4 reports some qualitative results obtained by our pipeline. Picture (a) shows the recognition results on an image taken quite far from the shelf and featuring a lot of different items; (b) deal with some successful recognition in a close-up *query* image, where only a few items are visible at once. Finally, (c) refers to recognition of products featuring deformable and highly reflective packages, which are quite challenging to recognize due to the appearance of the items within the *query* images turning out significantly different than in the available *reference* images. Yet, in (c) our system was able to find at least one item for each product type (*i.e.*, as required in the *Customer* use case).

5 DOMAIN INVARIANT HIERARCHICAL EMBEDDING FOR GROCERY PRODUCTS RECOGNITION

After describing in Chap. 4 a complete computer vision pipeline to solve the problem of unconstrained product recognition on store shelves, in this section we will explore more in detail how to properly address the *recognition* phase. As in Chap. 4 we still rely on a global image descriptor learned to disentangle grocery products and to pursue recognition through K-NN search within a database featuring one reference image per sought product. We rely on learned image embeddings as this approach allows seamlessly to perform recognition both on products *seen* or *unseen* at training time. For example, should a new product be put on sale in the store, our system would just require to add its image into the reference database without the need to perform new costly training. Moreover, K-NN search is quite amenable to product recognition from a computational standpoint alike. Indeed, compared to typical image retrieval settings, the database is very small, *i.e.* in the order of several thousand images rather than millions or even billions of images. Thus, a global image descriptor of about a few hundred entries turns out viable in terms of time and memory efficiency.

In Chap. 4 we have already show how a standard triplet loss can be deployed to learn an embedding suitable for product recognition. Here, instead, we wish to go one step further and directly address some criticality of the product recognition task: the domain shift between data acquired in-store and those available during training (*e.g.*, see Fig. 5.1) and the similarity between products belonging to the same category. We propose to learn an image embedding through a deep CNN trained by a loss function that forces both similar looking items as well as items belonging to the same high-level category to map close one to another in the descriptor space. Moreover, to tackle the domain shift between images available at training time and those acquired in stores, and to increase the training set size we propose to deploy an image-to-image translation GAN together with the embedding CNN and to optimize the whole architecture end-to-end. In particular, some of the training samples for the embedding network are generated by a GAN that learns without supervision to transform images taken in studio

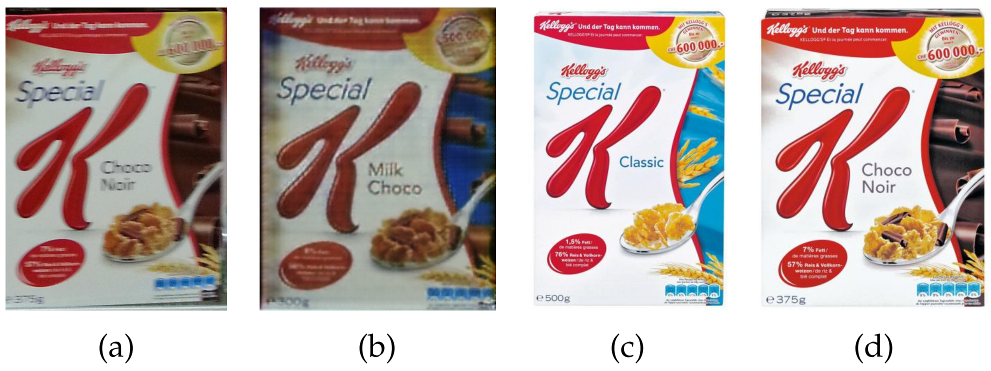


Figure 5.1: Exemplar images for the grocery recognition task: *reference* images (c-d) carefully acquired in studio (available at training time), *query* images: (a) captured in the store (*query* at test time) and (b) *synthetic query* generated by our GAN (used at training time).

settings into in-store images without introducing excessive modifications to product appearance (Fig. 5.1 (b) shows an exemplary image synthesized by the GAN). These training samples force the embedding CNN to learn robustness to domain shift; moreover, the GAN can be trained to produce samples that are particularly hard to embed, thereby allowing the CNN to learn a stronger embedding function thanks to these adversarial samples. Despite the use of multiple networks, the overall architecture can be trained end-to-end effortlessly via simple gradient descent.

5.1 RELATED WORK

Using CNNs to obtain rich image representations is nowadays an established approach to pursue image retrieval, both as a strong off-the-shelf baseline ([43]) and as a key component within more complex pipelines ([69]). Schroff *et al.* [61], firstly proposed to train a CNN using triplets of samples to create an embedding for face recognition and clustering. This approach has, since then, been extensively used to learn representations for a variety of different tasks, with more recent works advocating smart sampling strategies ([124]) or suitable regularizations ([125]) to ameliorate performance. Similarly to our proposal, [88] extend the idea of triplets by a novel formulation amenable to embed label structure and semantics at training time based on tuples. Unlike [61, 88], in this section we propose to embed label structure within the learning process using only standard triplets; moreover, our method uses only one exemplary image per class and augment the training set by a GAN trained jointly together with the embedding network.

The grocery product recognition problem shares commonalities with the *exact street to shop* task addressed in [53, 85, 115], which consists in recognizing a real-world example of a garment item based on the catalog of an online shop. Similarly to ours, these works rely on matching and retrieval using deep features extracted from CNN architectures. However, the *exact street to shop* task can leverage on labeled paired couples of samples depicting the same item in the *Street* and *Shop* domain to learn a cross-domain embedding at training time; while our proposal leverages only on labeled images from one domain, thereby vastly relaxing the applicability constraint.

Few-shot learning has been addressed successfully in [94] through classifiers trained on top of a fixed feature representation by artificially augmenting a small training set with transformations in the feature space. Yet, in the grocery product recognition scenario the items to be recognized at test time change quite frequently, which would mandate frequent retraining of new classifiers. Besides, as product packages exhibit very low intra-class variability, the generalization ability of a classifier may not be needed. Thus, we prefer to learn a strong image embedding and rely on K-NN similarly to perform recognition. Our approach shares commonalities with [84], where the authors address few-shot learning by a matching network that computes K-NN similarity in a learned embedding space.

Starting from the pioneering works of Goodfellow *et al.* [40] and Redford *et al.* [157], GANs have received ever-increasing attention in the computer vision community as they enable to synthesize realistic images with few supervision. Recently GAN frameworks have been successfully deployed to accomplish image-to-image translation, with ([97]) and without ([117, 128]) direct supervision, as well as to tackle domain shift issues by forcing a classifier to learn invariant features [121]. We draw inspiration from these works and deploy a GAN at training time to pursue domain adaptation as well as to improve the effectiveness of the learned embedding. A related idea is proposed in [108] though, unlike [108], a) we explicitly deploy the GAN while learning the embedding to attain domain adaptation, b) use only one sample per class and c) train the GAN to produce realistic though hard to embed training samples, *i.e.* the *generator* of our GAN not only plays an adversarial game against the *discriminator* but also against the *encoder* network that learns the embedding.

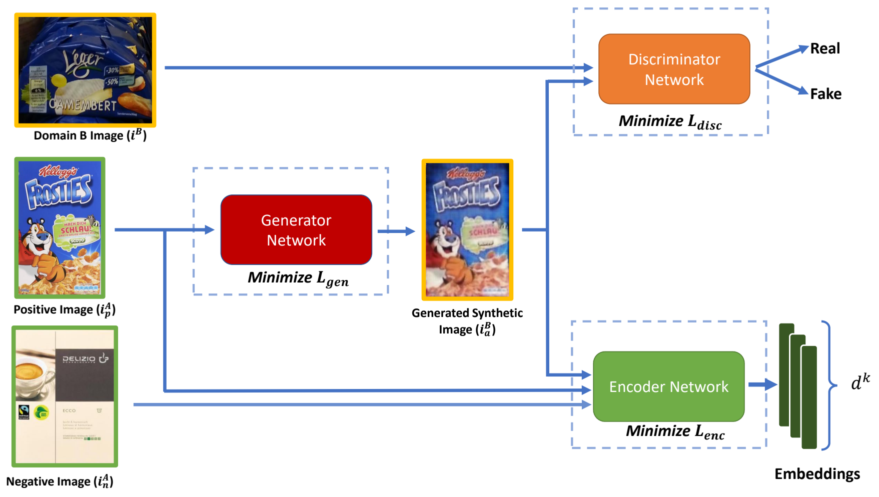


Figure 5.2: Overview of DIHE at training time. Each training sample consists of three images, two from domain A (enclosed in green) and one from domain B (enclosed in yellow). The *generator* and *discriminator* implements a classic GAN for domain translation from A to B. The *encoder* network uses two images from domain A alongside with the generated one to learn an image embedding by a modified triplet ranking loss. $i_a^{\mathcal{B}}$ is generated to be both indistinguishable from images sampled from domain B as well as hard to encode.

5.2 DOMAIN INVARIANT HIERARCHICAL EMBEDDING

An overview of our Domain invariant hierarchical embedding (DIHE) is depicted in Fig. 5.2. We use a deep CNN (*encoder*) to learn an embedding function $E : \mathcal{I} \rightarrow \mathcal{D}$ that maps an input image $i \in I$ to a k-dimensional descriptor $d^k \in \mathcal{D}$ amenable to pursue recognition through K-NN similarity search. During training, we exploit, if available, a taxonomy of classes by means of a novel loss function that forces descriptors of different items to be closer if they share some portion of the taxonomy, distant otherwise. To learn a descriptor robust to domain shifts between training and testing data, we use an image-to-image translation GAN, consisting of a *generator* and a *discriminator*, which augments the training set with samples similar to those belonging to the test domain while simultaneously producing hard examples for the embedding network. The three networks can be trained jointly by standard gradient descent in order to minimize the three loss functions described in the following sections.

5.2.1 *Hierarchical Embedding*

As already discussed in Sec. 4.2.2 an effective image embedding for product recognition can be obtained by training a deep CNN according to the *triplet ranking loss* ([47]). We propose to modify the original formulation presented in Eq. 4.1 for domains that feature a hierarchical structure between classes (*e.g.*, ImageNet classes taxonomy), so as to mimic this structure within the learned descriptor space. This is a quite common scenario for problems where a multi-level classification is available, for instance, grocery products included in existing commercial databases feature both labels at instance as well as multiple category levels (*e.g.*, for the product depicted in Fig. 5.1 (c) we would have three different classification labels with increasing generality: Kellog's Special K Classic →Cereal→Food). Our aim is to force the network to embed images nearby in the descriptor space not only based on their appearance but also on higher level semantic cues, like those shared between items belonging to the same macro-class. We argue that doing so will help to produce a stronger image descriptor and may provide better generalization to products whose reference images are *unseen* at training time.

Using the notation of Sec. 4.2.2 and assuming a taxonomy of classes encoded in a tree like structure, we propose to impose a hierarchy in $\mathcal{D}$ by rendering α inversely proportional to the *amount of hierarchy* shared between the classes of i_a, i_p and i_n. Each image sample i in the training set has a fine class c (foil level in the taxonomy) and a set of higher level classes $\mathcal{H}(i)$ (all the parent nodes in the class tree excluding the common root). Using this notation and defining the minimum and maximum margin, α_{min} and α_{max} respectively, our hierarchical margin, $\alpha \in [\alpha_{min}, \alpha_{max}]$, can be computed as:

$$\alpha = \alpha_{min} + \left(1 - \frac{|\mathcal{H}(i_a) \cap \mathcal{H}(i_n)|}{|\mathcal{H}(i_a)|}\right) \cdot (\alpha_{max} - \alpha_{min}) \tag{5.1}$$

where $|\cdot|$ is the cardinality operator for sets. Thus, if i_a and i_b share all the parent nodes $\alpha = \alpha_{min}$, whilst the margin is proportionally increased until completely disjoint fine classes will produce $\alpha = \alpha_{max}$.

5.2.2 *Domain Invariance*

A common trait across many computer vision tasks is that easily available labeled training data (*e.g.*, tagged images published online) are usually sampled from a different distribution than the actual test images. Thus, machine learning models straightly trained with such samples, such as embedding networks, will typically perform poorly at test time due to domain shift issues. However, annotating samples from the test distribution, even if possible, is usually very expensive and time-consuming. We propose to address this problem by dynamically transforming the appearance of the available labeled training images to make them look similar to samples from the unlabeled test images. This transformation is carried out by two CNNs, referred to in Fig. 5.2 as *generator* and *discriminator*, realizing an image-to-image translation GAN which is trained end-to-end together with the embedding network (*encoder*).

Given two image domains $\mathcal{A}, \mathcal{B} \subset \mathcal{I}$ consisting of $i_k^{\mathcal{A}} \in \mathcal{A}$ and $i_k^{\mathcal{B}} \in \mathcal{B}$, the standard image-to-image GAN framework can be summarized as a *generator* network that tries to learns a generative function $G : \mathcal{A} \rightarrow \mathcal{B}$ by playing a two player min-max game against a *discriminator* network $D : \mathcal{I} \rightarrow \mathbb{R}$ that tries to classify examples either as real images from $\mathcal{B}$ or fake ones produced by G. In the following we will denote with $G(i_k^{\mathcal{A}}) \in \mathcal{I}$ the output of the *generator* network given the input image $i_k^{\mathcal{A}}$ and with $D(i) \in \mathbb{R}$ the output of the *discriminator* network for image i. To generate samples similar to the images from domain $\mathcal{B}$ without drastically changing the appearance of the input image i_k^{A}, we introduce an additional term in the generator loss function ($\mathcal{L}_{reg}$) that, similarly to the self regularization term deployed in [117], forces $G(i_k^{A})$ to be visually consistent with i_k^{A}. In our architecture, $\mathcal{A}$ is the training set while $\mathcal{B}$ is a set of unlabeled images from the test data distribution. Thus, in the grocery product recognition scenario, $\mathcal{A}$ will be the training images, either rendered or studio quality, while $\mathcal{B}$ a set of unlabeled product images acquired in the store.

During each training iteration of the whole architecture we sample one image $i^{B} \in \mathcal{B}$ to train the *discriminator* and two from the other domain $i_p^{\mathcal{A}}, i_n^{\mathcal{A}} \in \mathcal{A}$ to train the *encoder* and *generator*. As mentioned in Sec. 5.2.1, the *encoder* needs triplets of samples to compute its loss, so we synthesize the missing image using the *generator* $i_a^{\mathcal{B}} = G(i_p^{\mathcal{A}})$. With this architecture the triplet used to calculate Eq. 4.1 consists of two images from domain $\mathcal{A}$ and one from the simulated domain $\mathcal{B}$, thereby mimicking the test conditions where the query images to be recognized will came from $\mathcal{B}$ and the reference images to perform K-NN similarity from $\mathcal{A}$.

The *encoder* is trained to minimize Eq. 4.1 with the margin defined in Eq. 5.1. The *discriminator* tries to minimize a standard cross entropy loss:

$$\mathcal{L}_{disc} = \log(D(i^B)) + \log(1 - D(G(i_p^A))) \tag{5.2}$$

while the *generator* minimizes a loss consisting of three terms:

$$\begin{aligned} \mathcal{L}_{gen} = L_{adv} + \lambda_{reg} \cdot L_{reg} + \lambda_{emb} \cdot L_{emb} \\ L_{adv} = -\log(D(G(i_p^{\mathcal{A}}))) \\ L_{reg} = \phi(i_p^{\mathcal{A}}, i_a^{\mathcal{B}}) \\ L_{emb} = -d(E(i_p^{\mathcal{A}}), E(G(i_p^{\mathcal{A}}))) \end{aligned} \tag{5.3}$$

with $\phi(x, y)$, $x, y \in \mathcal{I}$ a similarity measure between the appearance of image x and y, either at pixel level (*e.g.*, mean absolute difference...) or at image level (*e.g.*, SAD, ZNCC...), and $\lambda_{reg}, \lambda_{emb}$ two hyper parameters that weigh the different terms of the loss function. The contribution of the three terms can be summarized as follows. L_{adv} is the standard adversarial loss for the generator network that forces the synthesized images to be indistinguishable from those sampled from domain $\mathcal{B}$; L_{reg} is aimed at synthesizing images that preserve the overall structure of the input ones (avoiding thereby the mode collapse issue often occurring in unconstrained GAN generators); L_{emb} forces an additional adversarial behaviour against the *encoder*, so as to create hard to embed samples.

At training time, given a minibatch of M different triplets of samples $(i_p^{\mathcal{A}}, i_n^{\mathcal{A}}, i^B)$, the three networks are trained jointly to minimize their average loss on the M samples.

5.3 IMPLEMENTATION DETAILS

For the implementation of DIHE we have used tensorflow[1] as our deep learning framework. For the initialization of the *encoder* network on the fine tuning tests we have used the weights publicly available in the tensorflow/-models repository[2]. The three networks that compose DIHE can easily fit in a single GPU, so training our system, once implemented, in a deep learning framework is straightforward. For all our test we used as *generator* U-Net ([60]) and as *discriminator* PatchGAN ([97]), the latter producing a dense grid

1 https://www.tensorflow.org/
2 https://github.com/tensorflow/models/tree/master/research/slim

of predictions for each input image. For the *encoder* we tested different available CNN model with or without pre-trained weights on the ImageNet-1000 classification task. We will show how for the grocery product recognition scenario, the best performance can be obtained using as embedding the maximum activation of convolution features (MAC [83]) extracted from different layers, which are concatenated and L2-normalized to get a final representation laying on the unit hypersphere. For all our tests, we used as distance function $d(X, Y) = 1 - X \cdot Y$ with $X, Y \in \mathcal{D}$ (*i.e.*, one minus the cosine similarity between the two descriptors). As for ϕ in L_{reg}, we tried the pixel-wise L1 or L2 norms, the Structured Similarity Index (SSIM) ([8]) and the Zero Mean Normalized Cross Correlation (ZNCC) and found out the last to work best in all our tests, in the following $\phi(x, y) = ZNCC(x, y)$. The weights of each network are trained to minimize their specific loss functions as introduced in Sec. 5.2. We use Adam ([54]) as optimizer at different learning rates across the different tests. Concerning data preprocessing, we use as input color images with a fixed size of 256X256X3 and intensities rescaled between $[-1, 1]$. To obtain the input dimension the original images are rescaled to the target resolution preserving the aspect ratio and filling the extra pixels with 0s. The only additional data augmentation is a random crop with size at least 80% of the original image to attain the input of the *generator* network performed before rescaling the images to input resolution.

At test time the *encoder* network is extracted from the whole DIHE and used as a stand-alone global image descriptor. Our implementation is quite efficient and can easily encode, on GPU, more than 200 images per second taking into account also the time needed to load the images from disk and rescale them to the 256x256 input size. Finally, given that usually our *reference* database only provide a single image per product and the descriptor dimension is relatively low (between 256 and 1024 floats across different tests), we perform the K-NN similarity search extensively without any kind of approximation. The biggest descriptor database considered in our tests is the one obtained from *Product8600* using a 1024 float dimensional embedding vector (*i.e.*, the *reference* database is a matrix of float with 8600 rows and 1024 column). Even on this kind of database given a query descriptor the whole similarity search can be solved in a tenth of a second using brute force search, nevertheless, the search could potentially be speeded up using KD-Tree or approximate search technique.

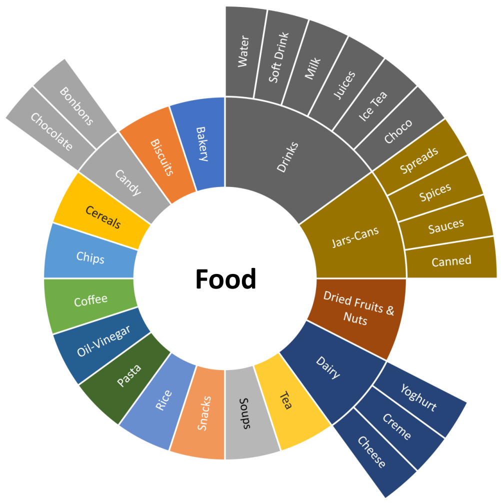


Figure 5.3: Visualization of the hierarchy of categories of the *Grocery_Food* dataset used as training set throughout our experiments. Each outermost category contains several different fine classes (products) not depicted for clarity.

5.4 EXPERIMENTAL RESULTS

To evaluate the effectiveness of DIHE in recognizing grocery products we rely on two products datasets comprising thousands of items: the publicly available *Grocery Products* dataset ([39]) and a standard commercial database, referred to here as *Product8600*. Both datasets include more than 8500 grocery products, each described by exactly one studio-quality (*reference*) image of the frontal face of the package, and feature a multi-level class hierarchy in the categorization of products. As already discussed, at test time we pursue recognition from a different set of images (*query*). To create this set, for *Grocery Products* we automatically cropped individual items from the available shelf images according to the annotation used in Chap. 3, thereby obtaining a total of 938 *query* images. As for *Product8600*, we cropped and annotated individual items from shelf videos that we acquired in a grocery store by a tablet camera, for a total number of 273 *query* images. As the shelf images available in *Grocery Products* concern only items belonging to the *Food* macro class, which accounts for 3288 products, we consider also this smaller subset of products, which will be referred to as *Grocery_Food*, whilst *Grocery_Full* will denote all the products of the *Grocery Product* dataset. We depict in Fig. 5.3 the taxonomy of macro categories that compose the *Grocery_Food* dataset which we are going to use as the training set, each category features several fine-grained classes (one for each product) not depicted in the figure. As for the samples from domain $\mathcal{B}$ needed to train the *discriminator* and *generator* of our architecture (see Sec. 5.2.2), we have used 547 additional images cropped from the shelf images available in *Grocery Products*, picked as to have no overlap with the previously mentioned 938 *query* images used at test time. We wish to point out how our formulation requires images from domain $\mathcal{B}$ only for the discriminator of the GAN system. Therefore, few samples without any kind of annotation are sufficient to learn the appearance of products on the shelf. Moreover, we can use images from domain $\mathcal{B}$ that depict any kind of product, even items not in $\mathcal{A}$.

NETWORK ARCHITECTURES For the implementation of DIHE we have used tensorflow[3] as our deep learning framework. For all our test we used as *generator* U-Net ([60]) and as *discriminator* PatchGAN ([97]), the latter producing a dense grid of predictions for each input image. For the *encoder* we tested different available CNN model with or without pretrained weight on the ImageNet-1000 classification task. For the initialization of the *encoder* network on the fine tuning tests we have used the weights publicly available in the tensorflow/models repository[4]. The three networks that compose DIHE can easily fit in a single GPU, so training our system, once implemented, in a deep learning framework is straightforward.

DESCRIPTOR COMPUTATION We will show how for the grocery product recognition scenario, the best performance can be obtained using as embedding the maximum activation of convolution features (MAC [83]). We extract these descriptors from different layers, concatenate them and finally perform L2-normalization to get a final representation laying on the unit hypersphere. For all our tests, we used as distance function $d(X, Y) = 1 - X \cdot Y$ with $X, Y \in \mathcal{D}$ (*i.e.*, one minus the cosine similarity between the two descriptors).

TRAINING DETAILS As for ϕ in L_{reg}, we tried the pixel-wise L1 or L2 norms, the Structured Similarity Index (SSIM) ([8]) and the Zero Mean Normalized Cross Correlation (ZNCC) and found out the last to work best in all our tests, in the following $\phi(x, y) = \text{ZNCC}(x, y)$. The weights of each network are trained to minimize their specific loss functions as introduced in Sec. 5.2. We use Adam ([54]) as optimizer with different learning rates for the different tests. Concerning data preprocessing, we use as input color

3 `https://www.tensorflow.org/`
4 `https://github.com/tensorflow/models/tree/master/research/slim`

images with a fixed size of 256×256 and intensities rescaled between $[-1, 1]$. To obtain the input dimension the original images are rescaled to the target resolution preserving the aspect ratio and filling the extra pixels with 0s. The only additional data augmentation is a preliminary random crop with size at least 80% of the original image to attain the input of the *generator* network.

EVALUATION PROTOCOL To test our embedding network we encode all the *reference* images of the considered dataset to create a reference database, then compute the same encoding for the *query* images. For each query vector we perform similarity search against all the reference vectors and retain the K most similar database entries; if the reference image for the product depicted in the query does belong to this set of nearest neighbors we consider recognition to be successful. As a measure of the effectiveness of the embedding, we report the accuracy (number of successful recognitions over the number of queries) for different K values.

Based on these premises, we train once and for all our architecture using only the reference images belonging to *Grocery_Food*, (*i.e.*, one reference image for each of 3288 different products organized in a multi-level hierarchy of products categories). We then use the trained embedding model to address three different test scenarios:

(a) *Grocery_Food*: we recognize the 938 *query* images from *Grocery Products* based on the 3288 reference images from *Grocery_Food*. Thus, all the *reference* images were deployed at training time.

(b) *Grocery_Full*: we recognize the 938 *query* images from *Grocery Products* based on the 8403 reference images from *Grocery_Full*. Thus, only 40% of the *reference* images were deployed at training time.

(c) *Product8600*: we recognize the 273 *query* images cropped from our videos based on the 8597 reference images from *Product8600*. Thus, none of the *reference* images was deployed at training time.

Among the three, (b) is the most likely to happen in practical settings as product appearance changes frequently over time and it is infeasible to constantly retrain the embedding network, although perhaps a portion of the reference images dealing with the items actually on sale in the store had been used at training time

	(a) Grocery_Food		(b) Grocery_Full		(c) Product8600	
Training loss	K=1	K=5	K=1	K=5	K=1	K=5
(1) triplet	0.301	0.430	0.277	0.390	0.351	0.490
(2) hierarchy	0.325	0.491	0.302	0.433	0.355	0.553
(3) triplet+GAN	0.454	0.626	0.418	0.586	0.512	0.706
(4) hierarchy+GAN	0.479	0.660	0.455	0.621	0.538	0.699
(5) triplet+GAN+adv	0.470	0.648	0.431	0.595	0.548	0.717
(6) hierarchy+GAN+adv (**DIHE**)	**0.481**	**0.688**	**0.463**	**0.642**	**0.553**	**0.732**

Table 5.1: Ablation study for DIHE. Recognition accuracy for 1-NN and 5-NN similarity search in the three considered scenarios. Best results highlighted in bold.

At test time the *encoder* network is used as a stand-alone global image descriptor. Our implementation is quite efficient and can easily encode, on GPU, more than 200 images per second taking into account also the time needed to load the images from disk and rescale them to the 256×256 input size. Finally, given that usually our *reference* database only provides a single image per product and the descriptor dimension is relatively low (between 256 and 1024 floats across different tests), we perform the K-NN similarity search extensively without any kind of approximation. The biggest descriptor database considered in our tests is the one obtained from *Product8600* using a 1024 float dimensional embedding vector (*i.e.*, the *reference* database is a matrix of float with 8600 rows and 1024 column). Even on this kind of database given a query descriptor the whole similarity search can be solved in a tenth of a second using brute force search, nevertheless, the search could be speeded up using KD-Tree or approximate search technique.

5.4.1 *Ablation Study*

To understand the impact on the performance of the different novel components proposed in our architecture, we carry out a model ablation study using as *encoder* PatchGAN ([97]) with MAC features ([83]) extracted from the last convolutional layer before the output. We use this randomly initialized small network to better highlight the gains provided by the different kind of proposed losses. We will show how to obtain the best performance we rely on a larger pre-initialized network. We train this architecture on the *reference* images of *Grocery_Food* according to six different training losses and report the accuracy dealing with the three test scenarios presented in

Sec. 5.4 in Tab. 5.1. In particular, with reference to the first column, *triplet* denotes training by triplet loss with fixed margin ($\alpha = 0.3$ obtained by cross-validation); *hierarchy* denotes training by our triplet loss with variable margin introduced in Sec. 5.2.1 ($\alpha_{min} = 0.1, \alpha_{max} = 0.5$); entries with *+GAN* denote deploying the image translation GAN to generate the anchor image $i_a^{\mathcal{B}}$ (Eq. 5.3: $\lambda_{reg} = 1, \lambda_{emb} = 0$); finally, entries with *+GAN+adv* concerns introducing also the adversarial term in the loss of the GAN generator (Eq. 5.3: $\lambda_{reg} = 1, \lambda_{emb} = 0.1$). For all the models that do not use GANs, we rely on standard data augmentation techniques (*e.g.*, crop, gaussian blur, color transformation. . .) to obtain the anchor image given the positive one. In all the tests the networks are randomly initialized and trained for the same number of steps and identical learning rates.

The results reported in Tab. 5.1 show how each individual novel component proposed in our DIHE architecture provides a significant performance improvement with respect to the standard triplet ranking loss. Indeed, by comparing rows (2),(4) and (6) to (1),(3) and (5), respectively, it can be observed that modifying the fixed margin of the standard triplet loss into our proposed hierarchically adaptive margin can improve accuracy with all models and in all scenarios, with a much larger gain in (c)(*i.e.*, completely unseen *reference* images). This proves that embedding a hierarchy into the descriptor space is an effective strategy to help to learn an embedding amenable to generalize to unseen data. The main improvements are clearly achieved by methods featuring a GAN network to pursue domain adaptation by generating training samples similar to the images coming from the test domain. Indeed, comparison of (3) and (4) to (1) and (2), respectively, highlights how performance nearly double across all models and scenarios, testifying that the domain shift between test and train data and the lack of multiple training samples are indeed the key issues in this task that the proposed image-to-image translation GAN can help to address very effectively. Finally, comparing (5) and (6) to (3) and (4), respectively, vouches that training the *generator* to produce anchor images not only realistic but also hard to embed turns out always beneficial to performance. Indeed, the adversarial game played by the *generator* and *encoder* may be thought of as an online and adaptive hard-mining capable of dynamically synthesize hard to embed samples that help training a more robust embedding. Performance of our overall DIHE architecture are reported in the row (6) and show a dramatic improvement with respect to the standard triplet loss, row (1).

5.4.2 *Product Recognition*

MODEL AND DESCRIPTOR SELECTION To ameliorate product recognition performance we can rely on larger networks pre-trained on the ImageNet classification benchmark. To chose the best CNN model as our *encoder* network we downloaded the public available weights of different models trained on ImageNet-1000 classification and test them as general purpose off-the-shelf feature extractors on our datasets without any kind of fine-tuning. We considered three different popular CNN models (VGG_16 [62], resnet_[50/101/152] [70] and inception_v4 [119]) and compute three kind of different descriptors from activations extracted at various layers:

- **Direct**: directly use the vectorized activation of a given layer. The dimension of the descriptor is the number of elements in the feature maps for that layer.
- **AVG**: perform average pooling on the feature map with a kernel with width and height equals those of the map. Therefore, we use as descriptor the average activation of each convolutional filter for a given layer. The dimension of the descriptor is the number of convolutional filters in the selected layer.
- **MAC [[83]]**: perform max pooling on the feature map with a kernel with width and height equals those of the map. Therefore, we use as descriptor the maximum activation of each convolutional filter for a given layer. The dimension of the descriptor is the number of convolutional filters in the selected layer.

We applied the three different descriptors above at different layers of the three networks and report in Tab. 5.2 the 1-NN accuracy using the test protocol described in Sec. 5.4.1. For all our tests the descriptors were L2 normalized to the unit norm and the similarity search is performed using cosine similarity. In Tab. 5.2 we report only some of the best performing layers for each network and additional tests where the descriptors are obtained by concatenation of representations extracted at different depths in the CNN (*e.g.*, *conv4_3+conv5_3* is the concatenation of representations extracted at layers conv4_3 and conv5_3) with L2 normalization performed after concatenation.

Looking at the results in Tab. 5.2 we can observe how, in our settings, newer and more powerful CNN, like inception_v4 or resnet_152, fail to

CNN	Layer	Type	Grocery_Food	Grocery_Full	Product8600
VGG_16	conv4_3	MAC	0.789	0.785	0.717
		AVG	0.515	0.510	0.538
	conv5_3	MAC	0.724	0.720	0.611
		AVG	0.406	0.398	0.395
	conv4_3+conv5_3	MAC	**0.792**	**0.787**	**0.725**
		AVG	0.501	0.493	0.523
	fc6	Direct	0.560	0.549	0.549
	fc7	Direct	0.444	0.433	0.432
inception_v4	Mixed_7a	MAC	0.610	0.603	0.509
		AVG	0.673	0.670	0.560
	Mixed_7b	MAC	0.652	0.641	0.512
		AVG	0.675	0.668	0.5091
	Mixed_7a+Mixed_7b	MAC	0.655	0.646	0.534
		AVG	0.690	0.685	0.542
resnet_50	Block3	MAC	0.731	0.729	0.703
		AVG	0.441	0.433	0.432
	Block4	MAC	0.654	0.646	0.509
		AVG	0.571	0.558	0.465
	Block3+Block4	MAC	0.723	0.720	0.644
		AVG	0.547	0.538	0.545
resnet_101	Block3	MAC	0.737	0.735	0.695
		AVG	0.389	0.388	0.432
	Block4	MAC	0.636	0.662	0.490
		AVG	0.570	0.556	0.417
	Block3+Block4	MAC	0.714	0.708	0.626
		AVG	0.535	0.524	0.520
resnet_152	Block3	MAC	0.708	0.703	0.655
		AVG	0.345	0.337	0.446
	Block4	MAC	0.571	0.561	0.435
		AVG	0.571	0.561	0.435
	Block3+Block4	MAC	0.678	0.671	0.542
		AVG	0.506	0.500	0.504

Table 5.2: 1-NN accuracy for different descriptors obtained from layers of network pre-trained on the ImageNet-1000 classification dataset without any kind of additional fine-tuning. Best results are higlighted in bold.

		(a) ***Grocery_Food***		**(b)** ***Grocery_Full***		**(c)** ***Product8600***	
	Training loss	K=1	K=5	K=1	K=5	K=1	K=5
(1)	triplet	0.799	0.922	0.775	0.894	0.765	0.915
(2)	hierarchy	0.812	0.933	0.816	0.926	0.805	0.952
(3)	triplet+GAN	0.829	0.941	0.821	0.937	0.816	0.945
(4)	hierarchy+GAN	0.832	0.943	0.826	0.933	0.819	0.952
(5)	triplet+GAN+adv	0.833	**0.948**	0.821	0.937	0.816	0.945
(6)	hierarchy+GAN+adv (**DIHE**)	**0.853**	**0.948**	**0.842**	**0.942**	**0.827**	**0.959**

Table 5.3: Ablation study for DIHE on a VGG-16 network pretrained on ImageNet-1000. Recognition accuracy for 1-NN and 5-NN similarity search in the three considered scenarios. Best results highlighted in bold.

achieve the same instance-level distinctiveness of VGG_16. We conjecture that deeper architectures, trained on Imagenet, tend to create more abstract representations that may not provide out-of-the-box features distinctive enough to tell apart many items looking almost identical as required by our problem. Some evidence to support this conjecture may be found in Tab. 5.2 due to deeper layers providing typically inferior performance when compared to shallower ones (e.g., VGG_16: conv5_3 vs conv4_3, resnet_50,resnet_101,resnet_152: Block4 vs Block3) Concerning the type of descriptor to use, from Tab. 5.2 it seems quite clear that MAC descriptor is the best choice for grocery recognition with respect to AVG or direct activations of fully connected layers.

Given these results, we selected for our fine tuning tests the VGG_16 network with MAC descriptor computed on the concatenation of conv4_3 and conv5_3 layers, and train the overall architecture according to our losses. As the *encoder* is already pre-trained, we perform 5000 iterations of pre-training for the *generator* and *discriminator* (Eq. 5.3: $\lambda_{reg} = 1, \lambda_{emb} = 0$) before training jointly the whole DIHE architecture. The chosen hyper-parameters obtained by cross validation for the training process are as follows. Learning rates 10^{-5},10^{-5} and 10^{-6} for *generator*, *discriminator* and *encoder*, respectively; $\lambda_{emb} = 0.1$, $\lambda_{reg} = 1$, $\alpha_{min} = 0.05$ and $\alpha_{max} = 0.5$.

Before comparing our proposal to other embedding losses, it is interesting to verify whether the improvements provided by the different components (Sec. 5.4.1) are valid even when relying on the VGG-16 network pretrained on Imagenet. Purposely, we carry out the same ablation study as in Tab. 5.1 and report the results in Tab. 5.3. Indeed, the ranking of performances among the different training modalities turns out coherent, although, as

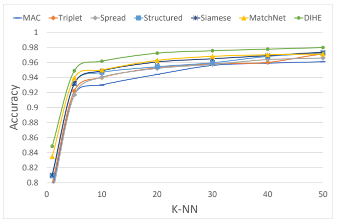
	(a) *Grocery_Food*		**(b) *Grocery_Full***		**(c) *Product8600***	
Training Loss	K=1	K=5	K=1	K=5	K=1	K=5
MAC	0.792	0.917	0.787	0.9093	0.725	0.908
Triplet [47]	0.799	0.922	0.775	0.894	0.765	0.915
Spread [125]	0.784	0.916	0.764	0.893	0.758	0.923
Structured [88]	0.809	0.931	0.804	0.926	0.750	0.912
Siamese [12]	0.810	0.931	0.805	0.928	0.733	0.926
MatchNet [84]	0.834	0.939	0.810	0.929	0.820	0.948
DIHE	**0.853**	**0.948**	**0.842**	**0.942**	**0.827**	**0.959**
	+0.02	+0.01	+0.03	+0.02	+0.007	+0.01

Table 5.4: Recognition accuracy for 1-NN and 5-NN similarity search in the three considered scenarios. Best results highlighted in bold, differences between DIHE and best performing competitor reported in the last line.

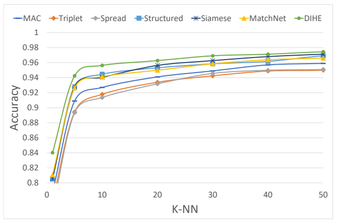
expected, the margins are smaller due to the higher performance provided by the baseline.

COMPARISON WITH OTHER EMBEDDING LOSSES We compare our architecture to the already mentioned concatenation of MAC descriptors without fine tuning (*MAC*) and to our implementation of different embedding learning methods: [47] fine tuning using the classic triplet ranking loss (*Triplet*); [12] fine tuning using Siamese networks and the contrastive loss (*Siamese*); Matching networks [84] without the *full context embedding* which does not scale to thousands of classes (*MatchNet*); [88] tuplet loss to embed label structure (*Structured*); and [125] triplet loss regularized by a spread out term (*Spread*). Similarly to DIHE, all methods are trained on the *reference* images of *Grocery_Food* starting from the very same VGG16 pre-trained on ImageNet-1000 and using the same concatenation of MAC descriptors as embedding function.

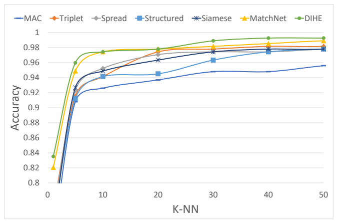
We use the same test scenarios presented in Sec. 5.4.1 and report the result in Tab. 5.4. As already shown in our model study, MAC activations have strong absolute performance without any need of fine tuning with accuracy at K=1 ranging from 0.72 in the worst case to 0.79 in the best. Starting from such a strong baseline the standard *Triplet* loss is able to only slightly increase performance in scenario (a) and (c) while being slightly penalized in (b), which testifies how in the cross-domain and low shot



(a) **Grocery_Food**



(b) **Grocery_Full**



(c) **Product8600**

Figure 5.4: Accuracy with increasing K in the three scenarios.

regime is quite hard to properly fine tune an embedding network. The additional regularization term introduced by *Spread* does not seems to help with a slight decrease in performance across all scenario compared with the standard *Triplet*. *Structured* is the first method to consistently improve performance across all scenario, vouching for the importance of deploying label structure in the embedding space for this type of recognition task. *Siamese* based on a simple contrastive loss is able to obtain performance comparable to *Structured* on the scenario (a) and (b) while performing slightly worse on the generalization to the dataset (c). *MatchNet* is definitely the best competing method for embedding learning in a few shot regime as testified by the good improvement obtained with respect to the initial MAC descriptors. Nevertheless, DIHE thanks to the combined use of GAN and hierarchical information is still able to improve the performance, obtaining recognition accuracies for K=1 consistently over 82% across all test. It is worth pointing out that the largest improvement, with respect to the initial MAC results, is obtained on the completely unseen products of the *Product8600*, which vouches for mixing GAN-based domain adaptation and hierarchical embedding to learn an embedding that can generalize very well to unseen items.

In Fig. 5.4 we report the accuracy of the methods while increasing K. In Fig. 5.4(a-b) almost all methods converge to more than 95% accuracy for $K > 30$. However, DIHE can provide substantially better results for lower values of K. In Fig. 5.4(c) DIHE still outperforms the competitors with performance almost equal to those achieved by Matching Networks but showing higher margin against the competitor trained with variants of *triplet ranking loss*.

	(a) Webcam		(b) DSLR	
Training Loss	K=1	K=5	K=1	K=5
FC6	0.470	**0.698**	0.489	0.712
MAC	0.382	0.640	0.393	0.660
Triplet [47]	0.596	0.675	0.628	0.710
Spread [125]	0.591	0.660	0.590	0.670
Siamese [12]	0.469	0.547	0.576	0.630
MatchNet [84]	0.522	0.579	0.566	0.618
DIHE	**0.628**	0.691	**0.662**	**0.742**

Table 5.5: Recognition accuracy for 1-NN and 5-NN similarity search on two subset of the *Office31* using as *reference* images the *Amazon* subset.

5.4.3 *Beyond product recognition*

To investigate the generality of our proposal as an improvement over established embedding learning methods, we perform additional tests on the *Office31* benchmark dataset for domain adaptation ([18]). This dataset consists of 4652 images dealing with 31 classes of office objects acquired in three different domains, namely *Amazon*, concerning ideal images downloaded from the web, *Webcam*, consisting of real images acquired by cheap cameras, and *DSLR*, featuring real images acquired by a high-quality camera. Akin to the setup of Grocery Recognition experiments, we use *Amazon* as the *reference* set, thus deploying these images to train the *encoder*, while images from *Webcam* and *DSLR* are used both as training samples for the GAN *discriminator* in DIHE and as two separate *query* sets for the tests. This scenario is compliant to the *full protocol setting* described by [32]: train on the entire labeled source and unlabeled target data and test on annotated target samples. Unfortunately, as the *Office31* dataset does not provide a taxonomy of classes, in these additional experiments DIHE can not leverage the hierarchical loss to improve the learned representation. However, unlike the Grocery Recognition scenario, all methods can be trained here by more than one sample per class.

Based on the above described experimental setup, we train different *encoders* starting, once again, from a VGG16 network pre-trained on the ImageNet-1000 dataset. We report the accuracy for 1-NN and 5-NN similarity search in Tab. 5.5. To asses the performance of DIHE we compare it once again against the methods considered in Sec. 5.4.2, with the exception of *Structured* due to *Office31* lacking a class taxonomy. The first two rows of Tab. 5.5 show that, differently from the Grocery Recognition setup, between activations extracted from the pretrained VGG16 network, FC6 outperforms MAC descriptors (computed as in Sec. 5.4.2). Coherently with the findings of Tab. 5.2, we believe this difference to be due to the office task concerning the recognition of the category of objects rather than instances. Accordingly, to obtain the *encoder* network for these tests we substitute the original FC6 layer with a smaller fully connected layer consisting of 512 neurons with randomly initialized weights and then perform fine tuning on the *Amazon* images.

In this different experimental settings wherein more training samples per class are available, both *Siamese*, *Spread Out* and *Matchnet* are outperformed by the plain *Triplet* ranking loss of [47], which turns out the best performing established method achieving a 1-NN accuracy of 59% and 62% for test datasets *Webcam* and *DSLR*, respectively. Yet, thanks to the introduction of the *Generator* in the training loop, DIHE can yield a significant performance improvement reaching a 1-NN accuracy of 62% (+3%) and 66% (+4%) for *Webcam* and *DSLR*, respectively, with best or comparable 5-NN accuracy when compared with competing methods using descriptors with the same dimension. Moreover, our proposal can outperform the descriptor extracted by FC6 that is twice larger on 3 experiments out of 4, obtaining comparable performance on the fourth.

Although performance on *Office31* turns out well below the state-of-the-art attainable by image recognition methods based on classifiers, amenable to handle a fixed and possible small number of classes, we argue that the experiments reported in this Section further highlight the advantages provided by our proposed DIHE architecture with respect to common feature learning approaches.

5.4.4 *Qualitative Results*

Fig. 5.5 and Fig. 5.6 report some successful recognitions obtained by K-NN similarity search based on DIHE. The upper portion of Fig. 5.5, dealing with the *Grocery_Food* scenario, shows query images for products quite hard to recognize due to both the *reference* database featuring several items looking remarkably similar as well as nuisances like shadows and partial occlusions (first query) or slightly deformed products (third query). The lower portion of Fig. 5.5 concerns the *Product8600* scenario: as clearly highlighted by the third and second query, despite differences between items belonging to the same brand and category (*e.g.*, the pasta boxes in the last row) being often subtle, DIHE can recognize products correctly. In Fig. 5.6 we report some results dealing with the *Office-31* dataset which vouches how DIHE can correctly retrieve images depicting objects belonging to the same category as the query. In Fig. 5.7, we highlight some failure cases. In the first row DIHE wrongly recognizes a stapler as a bookshelf, whilst in the second a ring binder is mistaken as a trash bin. In the third row DIHE seems to recognize the macro class and brand of the query product though missing the correct instance level label (*i.e.*, the correct NN, highlighted in green, is retrieved as the 3-NN). A similar issue pertains the fourth query image: all the first 5-NNs belong to the *Tea* macro class, but the correct item is not ranked among them.

Finally, in Fig. 5.8 we show some training images generated by our GAN framework to pursue domain invariance (Sec. 5.2.2). It is worth observing how the training samples created by our GAN seem to preserve both the overall structure and details of the input images, which is very important when addressing instance level recognition between many similar looking items, while modifying significantly the brightness and colors and injecting some blur, thereby realizing a domain translation between the input and output images.

5.4.5 *DIHE for product detection*

Finally we have tried to plug an *Embedder* trained using DIHE in the product detection and recognition pipeline described in Chap. 4. Considering the challenging test environment described in Sec. 4.3.3 we have measured a mAP of *38.54* (+2.52 with respect to the previous best result) and a PR of *60.82* (+2.41 with respect to the previous best result). Comparing these improvements with the difference between Triplet and DIHE in Tab. 5.4 it seems that the not perfectly cropped regions produced by the *Detector* harm the improvement achievable by DIHE without nevertheless making it pointless. We leave to future work the exploration of training the generator of DIHE to generate images similar to actual detection produced by a product detector.

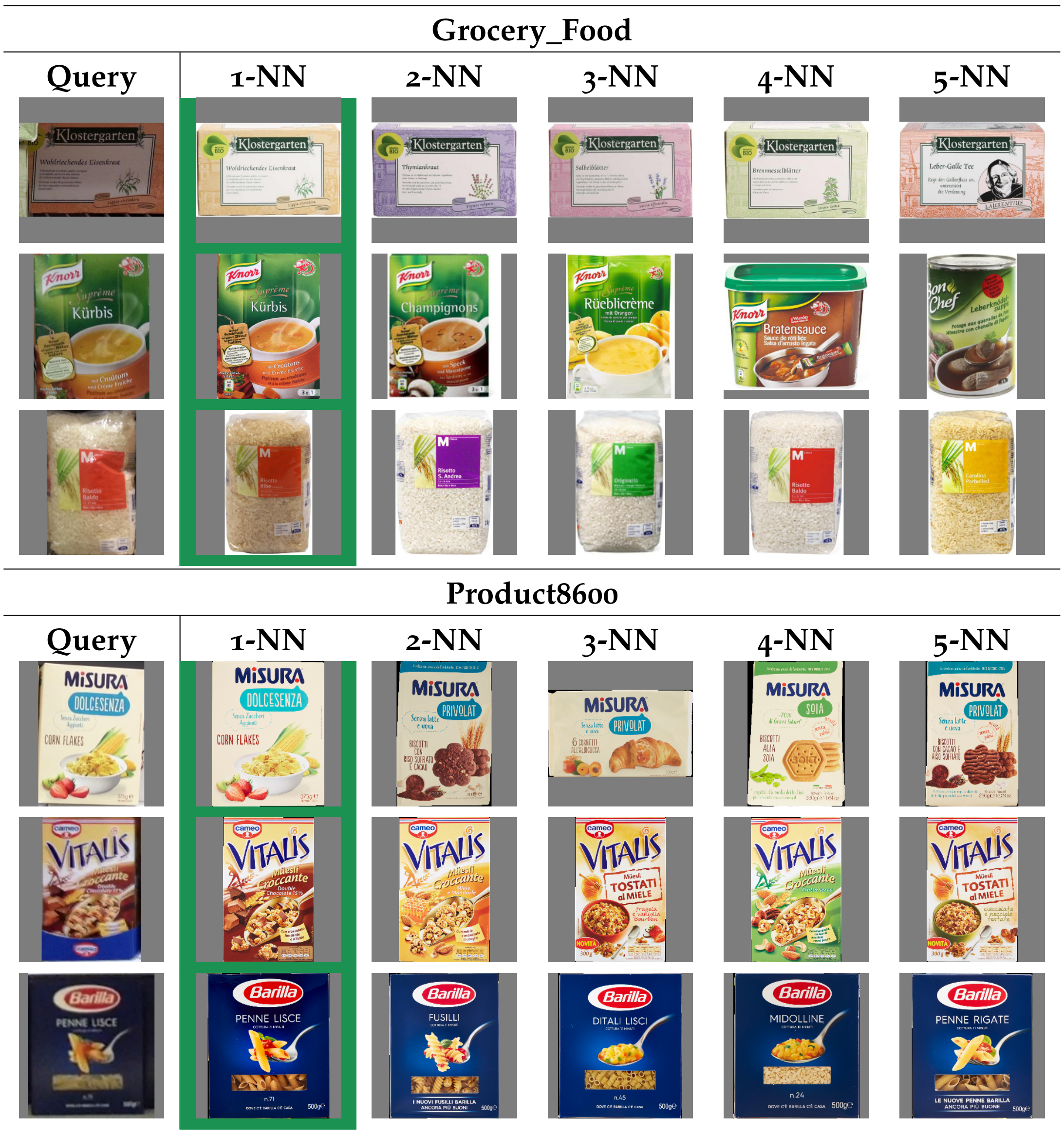


Figure 5.5: Qualitative results for K-NN similarity search by DIHE on *Grocery_Food* and *Product8600*. Correct results highlighted in green.

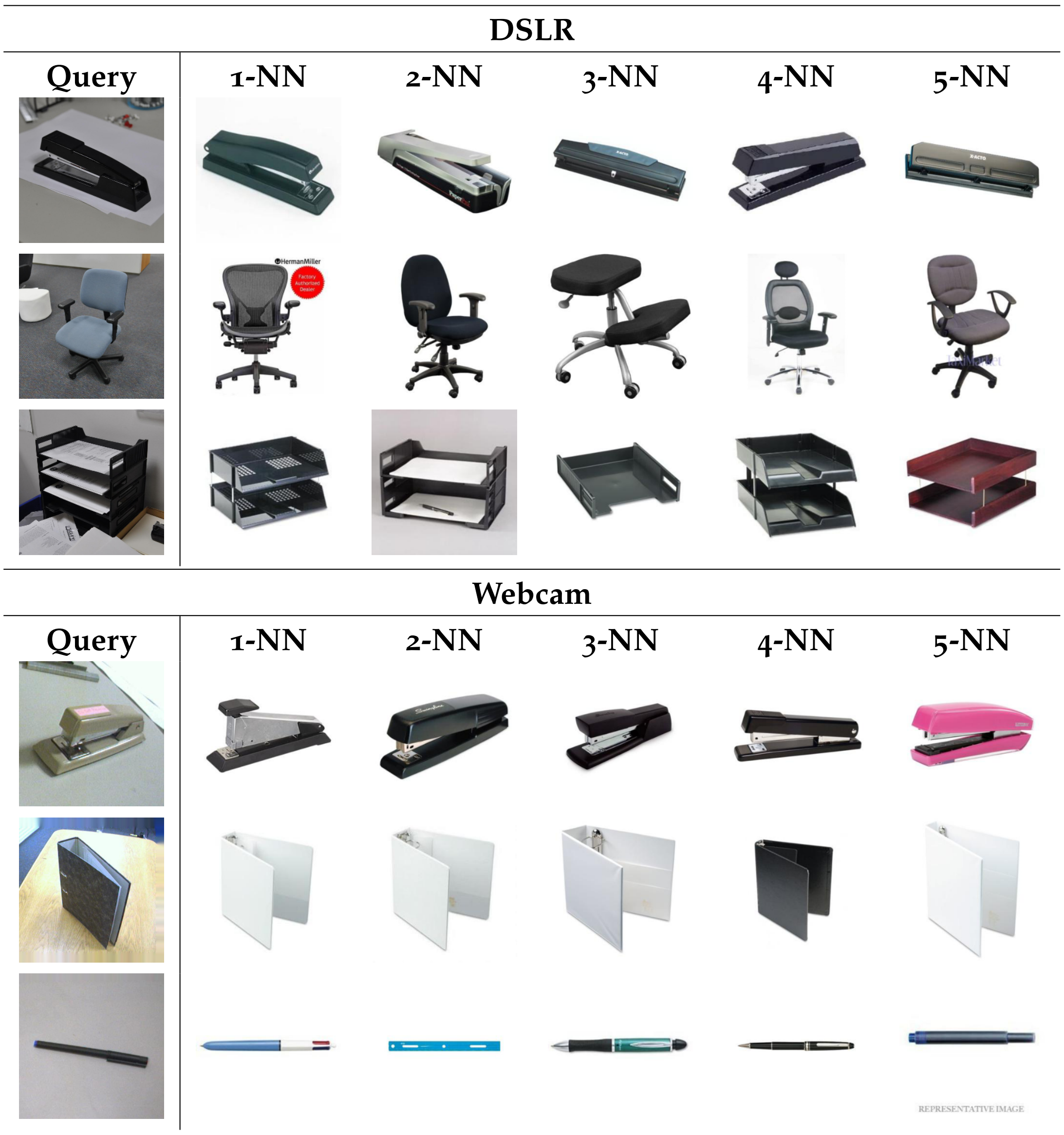


Figure 5.6: Qualitative results for K-NN similarity search by DIHE on the *Office31* dataset. Upper portion: *Amazon→DSLR* scenario, lower portion: *Amazon→Webcam*.

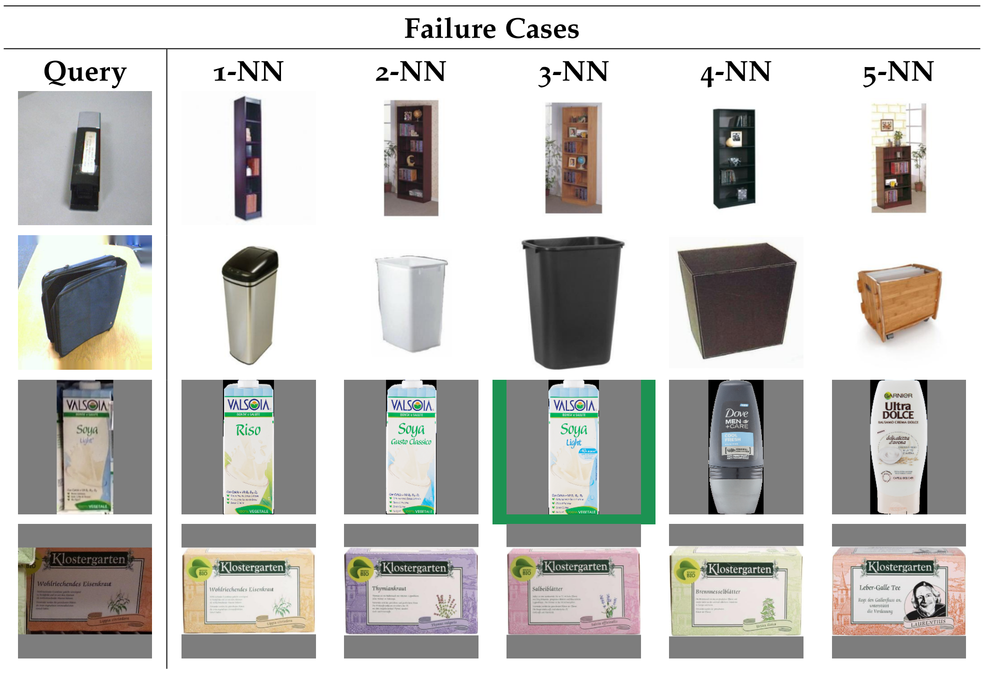


Figure 5.7: Some wrong recognitions yielded by DIHE on the different datasets.

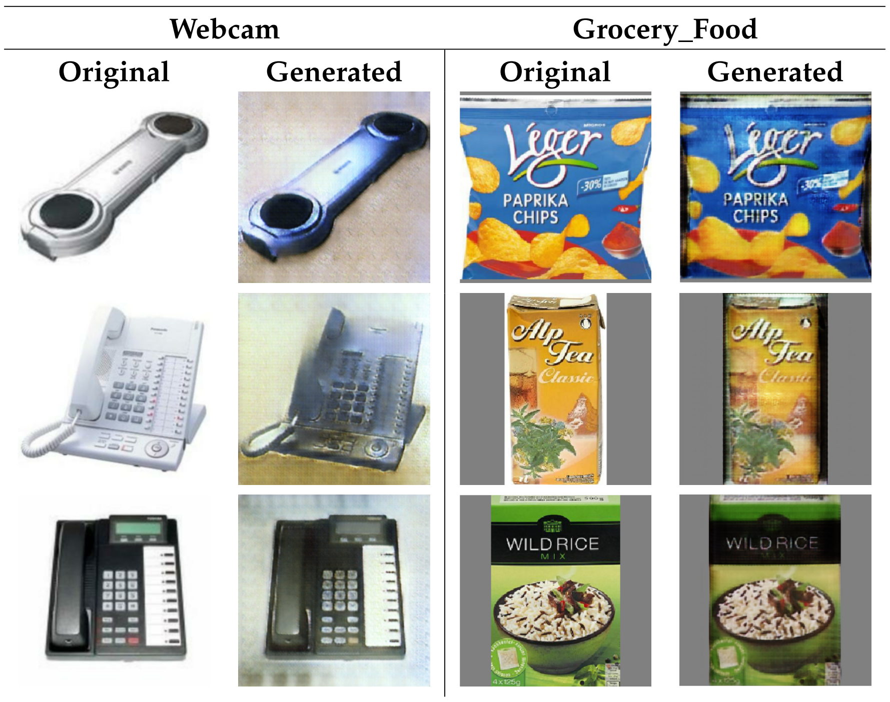


Figure 5.8: Images generated by the GAN trained jointly with the embedding network in DIHE for the *Office31-Amazon→Webcam* (left) and *Grocery_Food* (right) scenarios. Columns labeled as *Original* depict images provided as input to the *Generator* while those labeled as *Generated* show the corresponding outputs.

6 CONCLUSIONS

In this part, we have extensively addressed the recognition of products in store shelves either assuming the availability beforehand of the expected product layout (Chap. 3) or in a completely unconstrained setup (Chap. 4). For the solution of the first task, we relied on classic algorithms based on local features to achieve detection and recognition, while for the second we have shown how modern machine learning based methods can be successfully deployed. We have put a lot of effort into designing a system that does not require expensive manually annotated dataset, nor constantly retraining and relies only on information available in commercial databases. By self-imposing such limitations, we hope to produce an appealing solution for real stores. By comparing the results obtained by our two pipelines it is clear how machine learning based approaches can outperform feature based ones both in terms of accuracy and scalability. While features work remarkably well on textured packaged products, they tend to fail in presence of reflective surface or deformable packages, machine learning based method instead seems to be able to handle seamlessly even those situations (if properly trained). Finally in Chap. 5 we have shown how a CNN can be trained to directly address some of the criticality of the product recognition process such as the difference between images available during training and those to be recognized at test time.

Our system is still composed of multiple independent parts and features a lot of hyperparameters that need to be carefully tuned to achieve good performances. For the future we would like to investigate the possibility to deploy a unified CNN architecture acting as both *Detector* and *Embedder* for our machine learning based pipeline. Fusing the multiple stages in a single network will hopefully result in a lighter system that will be easier to deploy on mobile or resource-constrained platforms. However, the development of an all in one system for detection and image embeddings is non-trivial. Typically the solution of the former task involves a certain degree of invariance in learned image representation, while for the latter we wish to represent in a compact way peculiar image details. One possible solution that we wish to explore is to split the model into two parts, an initial feature extractor shared across the different problems and two separate final portions specialized for the solution of detection and image embedding respectively.

Part II

UNSUPERVISED ADAPTATION FOR DEEP DEPTH

7 INITIAL REMARKS

The focus of this part of the thesis will be on the problem of acquiring information about the 3D structure of an observed scene. This problem is one of the key steps to enable the solution of more complex tasks, like autonomous navigation, reconstruction and augmented reality and as such has always attracted huge research efforts. This technology has also wide applications in the supermarket environment where it could pave the way to augmented shopping experience for customers as well as be a fundamental building block for the development of autonomous agents moving inside the shop to monitor shelves or serve as personal shopping assistants. A more straightforward application concern the use of depth information to ease other tasks, for example, the *planogram* compliance check pipeline described in Chap. 3 would more easily detect missing product if depth information could be deployed by detecting discontinuities in the 3D structure of the items exposed on shelves. Moreover, even the object detection pipeline described in Chap. 4 could leverage on 3D information to improve the accuracy when detecting deformable objects or items with characteristic shapes. Other applications in the retail environment concern the 3D mapping and virtual reconstruction of real stores by using SLAM techniques. A 3D map of the shop would also be useful both for planning tasks as well as for interactive customer experiences like virtual tours or navigation aid to reach specific items. By combining the product detection capabilities shown in the previous chapter with the 3D sensing technologies that we are going to introduce here we can also think of reconstructing the shelves of a whole salespoint with the real product disposition in near real time, completely revolutionizing the way store are currently managed. For example, the concept of *planogram* could be completely redefined by allowing the head office to provide stores with a precise rendering of how the product should be arranged inside their specific shelves rather than a generic abstract representation.

The more commonly used technologies for depth sensing can be broadly categorized in *active* and *passive* sensors. The first family concerns devices that emits some signals (*e.g.*, lasers or infra-red light) and measure how they get perturbed by the environment to estimate the underlining 3D geometry

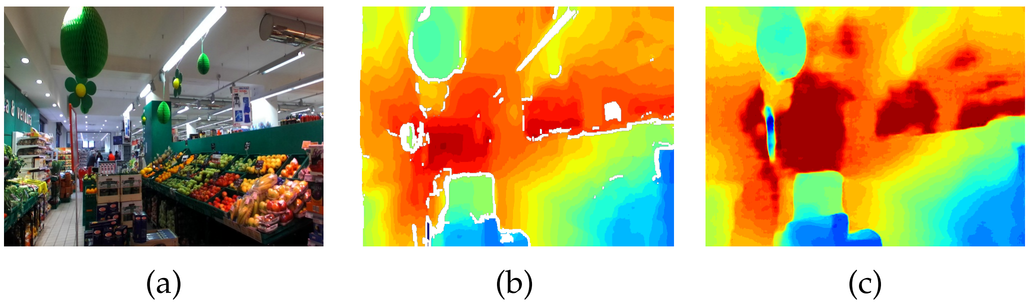


Figure 7.1: Example of depth sensing in a retail environment, (a) left rgb frame from a stereo couple acquired in a store, (b) depth map estimated using a commercial stereo sensor, (c) depth map estimated using DispNet stereo network [76]. White pixels correspond to unknown depth estimation.

of the scene; members of this family are laser scanners (*e.g.*, Velodyne LIDAR scanner[1]), time of flight device (*e.g.*, Microsoft Kinect 2) and active stereo sensors (*e.g.*, Microsoft Kinect 1). Devices belonging to this family have the advantage of being quite accurate and able to achieve a high frame rate, however, they suffer from some important drawbacks when applied to the retail environment. For example, noisy estimations when dealing with reflective or really dark surfaces that will deviate or absorb, respectively, the emitted rays; unfortunately, this kind of surfaces are quite common in stores. Other more general drawbacks concern the quite limited range of cheap sensors (*e.g.*, few meters for Kinect like sensors) and the high cost for the most accurate and versatile ones (*e.g.*, Velodyne LIDAR scanners). The *passive* family of sensors instead relies only on one or two RGB cameras and shift the focus of the problem from designing good sensing modalities to designing good algorithms that rely only on images to estimate 3D structures. By using only RGB cameras passive sensors are more cheap, compact and easy to deploy than their active counterpart, but, especially thanks to recent advantages in the field, they have also shown to be potentially as accurate without suffering from most of the limitations. Following the common nomenclature used in the field, we will refer to *depth-from-mono* when trying to estimate 3D information from a single RGB image and to *depth-from-stereo* when we will use couples of synchronized RGB frames acquired by two cameras with known relative positions. Since one of possible application we are targeting is autonomous indoor navigation we are not considering depth sensing modalities more targeted for offline 3D reconstruction like multi-view stereo and structure from motion pipelines.

1 https://velodynelidar.com/

We initially experimented with ready to use commercially available sensors like the ZED stereo camera[2] that allows to easily acquire synchronized stereo frames, rectify them and estimate the 3D structure of the scene using proprietary stereo algorithms. We opted for this sensor for the high accuracy, the easy deployment and the ability to produce depth estimation at 5 fps on a laptop GPU at 720p resolution. Using this sensor we carried out some preliminary acquisition in a real store but unfortunately, the results were not really satisfactory as can be observed in Fig. 7.1 -(b). While the general structure of the scene is estimated correctly, a lot of points do not have a valid depth estimation (white pixels in the image, *i.e.*, usually points near to objects boundaries where occlusions between left and right frames do not allow to compute a disparity). Even worse, some portion of the image features huge mistakes (*e.g.*, the skewed signboard on the left is completely missing in the depth map while there is a ghost flying blob slightly to the left of where the signboard is supposed to be).

Since we were not satisfied with the quality of the depth predictions produced by the black box development kit provided with the sensor, we tried to apply state of the art stereo algorithm on the acquired rectified stereo pairs to improve the results. Among all the possible algorithms we considered those that would provide depth estimation in timespan feasible to deployment on mobile platforms, *i.e.*, we do not consider stereo algorithms based on global energy minimization that required minutes to estimate a disparity map. At that time one of the best algorithm to fit our requirement of speed and accuracy was Dispnet [76] an end-to-end disparity estimation network that takes a couple of frames as input and directly outputs a disparity estimation. Dispnet had proven to be able to obtain remarkably accurate estimations on the challenging Kitti dataset [27, 57] while keeping an execution time of only $0.06s$ on a high-end GPU. Given these good premisses, we were pretty confident that it would be able to obtain good performance in our indoor environment as well. Unfortunately, we soon realized that without a proper fine-tuning the performance was meant to be low. Fig. 7.1 (c) display the prediction of a Dispnet network trained on synthetic data from the FlyingThings3D dataset [76] when tested on real in-store data. Once again the overall structure of the scene seems right and, compared to (b), we are also able to correctly detect the billboard on the left,

2 https://www.stereolabs.com/

however, the disparity map still features huge mistakes in the background area (that seems to cut off after a certain distance) and on the ceiling.

We ascribe those unsatisfactory result to the domain shift between training data (*i.e.*, rendered abstract synthetic images) and real ones (*i.e.*, our in-store acquisition). The usual practice to address domain shift is to fine tune the deep network on the target environment, however, in the depth estimation case, this procedure became quite challenging since it requires the acquisition of multiple frames from the new scenarios and a careful annotation of each pixel with its distance or disparity from the camera. The only way to generate such annotations, unfortunately, is to rely on costly high precision sensors like LIDAR laser scanners that, however, require careful calibration, can sense depth information only for a sparse subset of points in the image and still have to deal with possible mistakes coming from moving objects and reflective surfaces. Since the deployment of such sensors in our scenario was completely unrealistic we focused our research effort on possible way to *adapt* a deep depth estimation network to unseen environments without relying on ground truth data, *i.e.*, we wish to realize a *self-supervised adaptation*. Having ground truth information is, however, crucial to measure quantitatively if an algorithm is performing well or not. For this reason, and to compare our results with those achieved by other research groups, we are going to introduce and test our proposal in the context of autonomous driving where the KITTI datasets [27, 31, 57] offer rectified stereo frames coupled with sparse ground truth data coming from a LIDAR scanner that act as ground truth annotations.

In Chap. 8 we are going to describe a procedure to produce pseudo ground-truth labels by filtering noisy estimation using state of the art confidence measures and test it for offline fine tuning of a stereo network to unseen environments. In Chap. 9 we are going to introduce a new disparity estimation network and a procedure to adapt it online without the need of ground truth data to a new environment as soon as new frames are sensed. Finally in Chap. 10 we are going to present some ongoing work on how to use meta-learning techniques to make a depth estimation network able to adapt to unseen environments better and faster. But first, in Sec. 7.1 we are going to revise some of the recent literature on depth estimation from mono and stereo images.

7.1 RELATED WORK

We briefly review the literature for deep stereo matching and depth-from-mono.

Deep stereo matching. Since early works on stereo [6], traditional algorithms are thought as a number of subsequential steps, involving initial matching cost computation, local aggregation, disparity optimization and refinement. The firsts attempt to plug deep learning into stereo algorithms aimed at replacing single steps of the established pipeline, such as matching cost computation [50, 65, 74] and more recently optimization [82, 113] or refinement [92]. While these works substantially proved the superiority of learning based methods with respect to their non-learned counterparts on the single subtasks, in most cases they still required traditional optimization strategies, such as Semi Global Matching (SGM) [9], to reach top accuracy. The shift to end-to-end models started with Dispnet [76]. While previous works [50, 65, 74] processed small image patches to compute similarity scores, Dispnet works on a much larger receptive field and jointly extract features from two input frames, compute correlations between them and predict final disparities. Because of the large inputs processed, the few hundreds of images available from KITTI [27, 57] are not enough to train such model. To this aim a large synthetic dataset [76] is used for training, while KITTI images are used to address the domain shift when running on real imagery. Despite Dispnet did not reach the top of KITTI, it inspired many other end-to-end models [102, 139] able to assess this approach as state-of-the-art. Kendall *et al*. introduced the use of 3D convolutions [99] to explicitly exploit geometry and context, followed by [131, 152]. Despite the different architectural details, all of these techniques follow the same synthetic-to-real completely supervised training schedule. Only recently, this training paradigm has been questioned by proposing an unsupervised training schema. Pang *et al*. [143] propose a novel way of fine-tuning a deep network for stereo estimation by exploiting the output of the network at multiple resolutions combined within an iterative optimization problem with graph Laplacian regularization. Zhou *et al*. [126] use an iterative procedure based on the left-right check to train a network from scratch. Finally, Zhang *et al*. [155] propose a novel loss formulation based on image reprojection for depth estimation using an active stereo acquisition system.

Depth-from-mono. Deep learning dramatically boosted the quality of results obtained through depth-from-mono techniques. While most works addressed it as a supervised learning problem [16, 37, 41, 55, 71, 73, 123, 135, 150], an exciting trend concerns the possibility of self-supervising CNNs during training by solving an image reconstruction problem. These solutions earned a lot of attention from researchers thanks to the possibility of training them without requiring hard to source depth labels. They can be broadly classified into two main categories according to the cues used to replace ground-truth labels, that are respectively monocular [127] and stereo [93] self-supervision. In the first case, images are acquired by an unconstrained moving camera [127, 140, 149, 151]. Estimated depth is used to reconstruct views across the different frames by means of camera-to-world projection and vice-versa. To do so, the network also needs to estimate the relative camera pose between the different images, which is unknown. These approaches are weak in presence of dynamic objects, making reprojections in non-static regions of the frames not matching the real image content. The second category requires calibrated stereo pairs to be used as training dataset [68, 93, 129, 144, 153]. In this case, the relative pose is known thanks to stereo geometry, making the network only in charge of learning depth (actually, disparity) to minimize the reprojection between the two views. This solves the problem of moving objects, being images synchronized, but adds constraints for collecting data. Networks trained with this second approach usually results to be more accurate. This approach can be extended to trinocular setup [145] to compensate for occlusions inherited by the binocular stereo. Finally, we mention the semi-supervised frameworks proposed in [101, 134] and the joint use of these two supervisions [153].

8 UNSUPERVISED DOMAIN ADAPTATION FOR LEARNED DEPTH ESTIMATION

For years the problem of depth estimation from images has been tackled with hand-engineered algorithms, but recently deep learning proved to be much more effective and to provide superior accuracy. For the case of depth-from-stereo, initially, deep learning based methods replaced single steps of the traditional pipelines, afterward embodied the entire pipeline within deep end-to-end architectures. This latter approach currently represents the undisputed state-of-the-art in this field when enough training data is available. With these methods, training initially relies on large synthetic datasets [76], then fine-tuning on a few realistic images framing the target domain. For both training phases, stereo pairs with accurate ground-truth depth labels are mandatory. The popular KITTI online benchmarks [27, 57] witness the supremacy of these approaches, while it is less evident on Middlebury [42], where traditional, not learning-based algorithms, still keep the top rankings on the leaderboards due to the minimal amount of training images available. Deep learning also boosted the development of depth-from-mono systems, which requires only one image and thus can be potentially deployed on a broader range of devices equipped with a single camera. For monocular methods, deep learning enabled quite accurate results despite the strong ill-posed nature of the problem. Nevertheless, for stereo and monocular setups, deep architectures are affected by the *domain shift* curse, which drops the effectiveness of learning based frameworks when processing data (*i.e.*, images) much different from those on which the training process was carried out. This fact is particularly evident, for instance, when moving between indoor and outdoor environments or from synthetic to real data without any fine-tuning on the target environments. For example, Figure 8.1 (c) shows how DispNet [76] yields gross errors on a stereo pair of a dataset [30] lacking the ground-truth information to fine-tune the network. Unfortunately, besides a few research datasets, images paired with ground-truth depth information are quite rarely available as well as cumbersome and expensive to create in any practical settings. This state

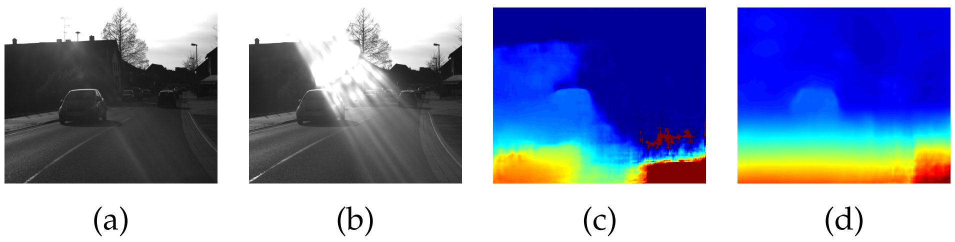


Figure 8.1: Effectiveness of unsupervised adaptation. (a),(b): Left and right images belonging to a challenging stereo pair of the dataset without ground-truth proposed in [30]. (c): Output provided by Dispnet [76]. (d): Output achieved after unsupervised adaptation of Dispnet.

of affairs may limit deployability of modern depth estimation architectures significantly.

To address the domain shift problem, in this paper we propose an unsupervised technique, enabling to fine-tune end-to-end architectures to new data domains without requiring any ground-truth measurement. To develop our method we build upon two key observations: i) computer vision researchers have pursued for decades the development of general-purpose stereo correspondence algorithms for depth estimation that does not require any adaptation to be deployed in different scenarios, ii) although traditional stereo algorithms exhibit well-known shortcomings in specific conditions (*e.g.*, occlusions, texture-less areas, photometric distortions ..), recent state-of-the-art confidence measures, more often than not relying on machine learning [58, 79, 80], can effectively highlight uncertain disparity assignments. Therefore we leverage on traditional stereo matching knowledge employing well-known algorithms from the literature to produce disparity/depth labels. Since these algorithms are sub-optimal solutions to the problem, some outliers occur among the obtained labels, and for this reason, we deploy confidence measures to attenuate or neglect their influence on the network employing a novel confidence-guided loss function. Following this strategy, with CNN-based depth estimation networks, we can tackle the domain shift problem by merely using synchronized stereo images without any depth annotation. Figure 8.1 (d) shows that our unsupervised adaptation approach can improve dramatically the output provided by DispNet [76] on a dataset lacking the ground-truth to fine-tune the network with supervision.

8.1 RELATED WORK

Confidence measures have been extensively reviewed by Hu and Mordohai [28] and by Poggi *et al*. [106] more recently including methods based on machine-learning. These latter methods can be broadly categorized into random-forest based [33, 45, 58, 79] and CNN based [80, 82, 136, 148]. While the first category, except [79], usually combines different cues available from the intermediate cost volume processed by traditional stereo algorithms [3, 9, 87], CNN based approaches only process disparity map and image content cues, making them better suited to work on custom systems or frameworks not explicitly providing a cost volume (*i.e.*, end-to-end CNNs). In general, this second category vouches for better outliers detection properties. Parallel works proposed an effective strategy to improve confidence measures by exploiting local consistency [103] and a method [105] to improve random forest-based approaches for confidence fusion [45, 58, 79] by using a CNN. Shaked and Wolf [114] embedded confidence estimation inside a deep model stereo matching, while Poggi et al. [104] propose an evaluation of conventional confidence measures and their simplifications when targeting embedded systems. Some works looked deeper into the learning process of confidence measures, by studying features augmentation [100] or by designing self-supervised techniques to train them on static video sequences [77] or stereo pairs [120]. In addition to otuliers detection and the unsupervised domain adaptation technique proposed in this paper, other applications of confidence measures aim at improving stereo accuracy [45, 58, 79, 82, 113], combine multiple algorithms [63, 78] and sensor fusion [75].

8.2 DOMAIN ADAPTATION FOR DEPTH SENSING

This section describes our domain adaptation framework, which is suited to both deep stereo as well as monocular depth estimation networks. To adapt a pre-trained model to face a new environment, we first acquire stereo pairs from the target domain. Then, we deploy a classical (*i.e.*, not learning-based) stereo algorithm to generate dense depth measurements together with a state-of-the-art confidence measure to estimate the reliability of the depth values calculated by the stereo algorithm. A key observation behind our method is that classical stereo algorithms, although affected by well-known shortcomings such as occlusions, poorly-textured regions and repetitive patterns, are substantially agnostic to the specific target environment and thus behave similarly across different scenarios. More importantly, they fail in the same predictable way, thereby enabling confidence measures to achieve remarkably good accuracy in detecting mistakes regardless of the sensed environment [106]. Based on the above observations, we advocate deploying the depths delivered by a classical stereo algorithm as noisy labels endowed with reliability estimations in order to fine-tune a network aimed at depth prediction. This is achieved through a novel per-pixel regression loss wherein the error between each model prediction and the corresponding depth measurement provided by the stereo algorithm is weighted according to the reliability estimated by the confidence measure, with higher weights associated to more reliable depth measurements. Thereby, the learning process is guided by the high-confidence depth measurements, *i.e.* those labels that appear to be more reliable, while the errors due to the shortcomings of the stereo algorithm have a negligible impact.

Thus, given a pre-trained depth estimation network, either stereo or monocular, and a set of stereo pairs, $(I^l, I^r) \in \mathcal{I}$, acquired from the target domain, for each pair we compute a dense disparity map, $D \in \mathcal{D}$, by means of a classical stereo algorithm, $f : (\mathcal{I}, \mathcal{I}) \to \mathcal{D}$, such as, *e.g.*, SGM [9] or AD-CENSUS[3]. Moreover, for each disparity map, D, we estimate a pixel-wise degree of reliability according to a confidence measure, $c : \mathcal{D} \to \mathcal{C}$. The resulting confidence map, $C \in \mathcal{C}$, encodes the reliability of the disparity calculated at each pixel as a score ranging from 0 (*not reliable*) to 1 (*reliable*).

We run f and c on each stereo pair available from the target domain so as to produce the training set deployed to perform fine-tuning of the pre-trained depth estimation network. Therefore, each sample, (S_i), in the training set is a tuple of four elements:

$$S_i = (I_i^l, I_i^r, D_i, C_i) = (I_i^l, I_i^r, f(I_i^l, I_i^r), c(f(I_i^l, I_i^r))) \tag{8.1}$$

Given the depth estimation network (either stereo or monocular), which takes input images and outputs per pixel disparities, we fine tune it toward the target domain by minimizing a loss function, L, consisting of three terms:

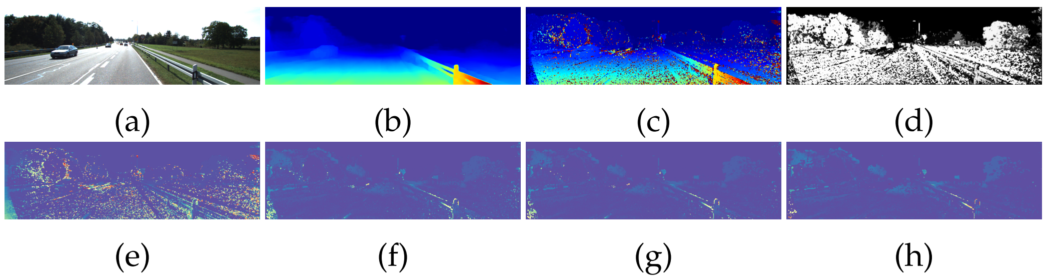


Figure 8.2: Visualization of our confidence guided loss: (a) left frame I^l; (b) Disparity map, $\tilde{D}$, predicted by the model; (c) Disparity map, D, estimated by a stereo algorithm; (d) Confidence map, C, on D; (e) L1 regression errors between (b) and (c), (f-h) same L1 errors weighted by C with $\tau = 0.00$ (f), $\tau = 0.50$ (g) and $\tau = 0.99$ (h). (e-h) Hotter colors marks bigger differences.

a *confidence guided loss* (L_c), a *smoothing loss* (L_s) and an *image reconstruction loss* (L_r):

$$L = L_c + \lambda_1 \cdot L_s + \lambda_2 \cdot L_r \tag{8.2}$$

with λ_1, λ_2 hyper-parameters aimed at weighting the contribution of the associated loss terms. All the three components of our loss can be applied seamlessly to deep learning models aimed either at depth-from-stereo or depth-from-mono (the latter one just need to convert disparities into depths). The structure of the three terms in Eq. 8.2 is detailed in the next sections, while in Sec. 8.3 we present model ablation experiments aimed at assessing their individual contribution to performance.

8.2.1 *Confidence Guided Loss*

The inspiration for the L_c term in the loss function of Eq. 8.2 comes from the observation that deep models can be successfully fine-tuned to new environments even by deploying only a few sparse ground truth annotations. This is vouched by the performance achievable on the KITTI datasets [27, 31, 57], where only a subset of pixels carries depth annotations (roughly $\frac{1}{3}$ of the image). The common strategy to account for the missing values consists simply in setting the loss function to 0 at those locations, thereby providing the network with meaningful gradients only at a subset of the spatial locations. Indeed, even in these sub-optimal settings, networks are able to adapt and ameliorate accuracy remarkably well. We build on these observations and leverage on the confidence measure, c, to obtain sparse and reliable depth labels from the noisy output D of the stereo algorithm. With reference to 8.1, denoting as $\tilde{D}$ the output predicted by the model at the current training iteration, we compute L_c as

$$L_c = \frac{1}{|P_v|} \sum_{p \in \mathcal{P}_v} \mathcal{E}(p) \tag{8.3}$$

$$\mathcal{E}(p) = C(p) \cdot |\tilde{D}(p) - D(p)| \tag{8.4}$$

$$\mathcal{P}_v = \{p \in \mathcal{P} : C(p) > \tau\} \tag{8.5}$$

where $\mathcal{P}$ is the set of all spatial locations on the image and $\tau \in [0, 1]$ a hyper-parameter that controls the sparseness and reliability of the disparity measurements provided by f that are actually deployed to update the model. A higher value of τ will mask out more mistakes in D though permitting less spatial locations to contribute to model update. Hence, points belonging to $\mathcal{P}_v$ define a set of sparse labels that, assuming the availability of a perfect confidence measure, may be used as if they were *groundtruth annotations*, *e.g.* akin to the LiDAR measurements deployed in the KITTI dataset. Yet, confidence measures are not perfect and often show some degree of uncertainty in the score assigned to disparity measurements. Thus, we weight the contribution at location p by $C(p) \in [0, 1]$, *i.e.* as much as the depth measurement, $D(p)$, can be trusted according to the confidence estimation, $C(p)$. We point out that, re-weighting the loss function in presence of noisy labels has been successfully exploited in supervised classification [137, 146]. Our formulation deploys a similar idea for a dense regression problem. Yet, we leverage on an external and highly accurate strategy to detect noise in the labels (*i.e.*, the confidence measure) and mask out those labels which, according to the adopted strategy, are very likely wrong, *i.e.*, $\{D(p) : p \notin \mathcal{P}_v\}$. In Sec. 8.3.2 we will show how both masking and re-weighting are crucial components to maximize performance in presence of noisy depth labels.

A graphical visualization of the errors that our L_c loss term tries to minimize can be observed in the bottom row of Fig. 8.2. On (e) we report the errors that will be minimized trying to directly regress the noisy depth labels of (c) given the model prediction on (b); on (f-g-h) we report, instead, the errors minimized by applying L_c with different τ values ($0, 0.5$ and 0.99 respectively). By tuning τ we can control the number of pixels, and therefore labels, taking part in the network adaptation process. Clearly, leveraging on more labels comes at the cost of injecting more noise in the process, which, in turn, may harm adaptation, even if their contribution will be attenuated by C, *e.g.* compare (f) to (e) where the only difference is the scaling of errors by $C(p)$ in (f). In (h) we can appreciate how even with $\tau = 0.99$ the amount of pixels considered during the optimization process is still quite high.

8.2.2 *Smoothing Term*

As L_c produces error signals to improve disparity prediction only at the subset of sparse image locations P_v, similarly to [34] we use an additional loss term, L_s, to propagate model update signals across neighbouring spatial locations. In particular, L_s tends to penalize large gradients in the predicted disparity map ($\partial\tilde{D}$) taking into acocunt the presence of gradients in pixel intensities (∂I):

$$L_s = \frac{1}{|P|} \sum_{p \in \mathcal{P}} \partial_x \tilde{D}(p) \cdot e^{-||\partial_x I(p)||} + \partial_y \tilde{D}(p) \cdot e^{-||\partial_y I(p)||} \tag{8.6}$$

Thus, based on the consideration that depth discontinuities occur in correspondence of edge, L_s constrains the predicted disparity map, $\tilde{D}$, to be smooth everywhere but at image edges. To efficiently compute gradients along x and y we use convolution with 3×3 Sobel filter.

8.2.3 *Image Reconstruction Loss*

To further compensate for the sparse model update information yielded by L_c, we include in the loss function a pixel-wise *image reconstruction* term, denoted as L_r in Eq. 8.2. Inclusion of this term in our loss has been inspired by [93], which has shown how deploying image re-projection between stereo frames can deliver a form of self-supervision to train a depth-from-mono network. Hence, given a stereo pair, I^l can be reconstructed from I^r according to the current disparity prediction $\tilde{D}$ by employing a bilinear sampler in order to render the process locally differentiable. Denoted as $\tilde{I}^l$

the re-projection of I^r according to $\tilde{D}$, we define the image reconstruction loss, L_r, as a weighted combination of the L1 norm and the single scale SSIM [8]:

$$L_r = \frac{1}{|P|} \sum_{p \in P} \alpha \frac{1 - \text{SSIM}(I^l(p), \tilde{I}^l(p))}{2} + (1 - \alpha)|I^l(p) - \tilde{I}^l(p)| \tag{8.7}$$

Similarly to [93], we use a simplified SSIM based on a 3×3 block filter and set $\alpha = 0.85$ throughout all our experiments.

8.3 EXPERIMENTAL RESULTS

In this section, we report exhaustive experiments showing the effectiveness of the proposed unsupervised domain adaptation technique. We run two main experiments i) adaptation of a stereo network and ii) adaptation of a depth-from-mono network. For the first case, we run extensive experiments on the entire KITTI raw dataset [31], counting about 40k images, thanks to a large amount of ground truth labels made recently available [122] in the official website. Concerning the second evaluation, we follow common protocols from the literature of self-supervised monocular depth estimation [93], splitting the KITTI raw data into train and test portions according to Eigen et al. [37].

To deploy the confidence guided loss described in Sec. 8.2.1, in our evaluation we consider two classical stereo algorithms (f): AD-CENSUS (shortened AD) [3] and Semi-Global Matching (shortened *SGM*) [9] and leverage the implementations of [44]. We have selected these two popular algorithms because they show quite different behaviors. While AD tends to generate prediction errors in the form of small spikes in the disparity maps, the errors generated by SGM can often cause over-smoothing. The effectiveness of our proposal with both types of error patterns may help testify its general validity. Besides, while SGM may turn out remarkably accurate, AD is notoriously significantly more prone to errors, which, in our framework, leads to fewer disparity measurements used at training time to compute L_c due to fewer pixels belonging to $\mathcal{P}_v$. To measure the confidence of the disparity measurements coming from the stereo algorithms, we rely on CCNN [80] as it can yield state-of-the-art performance and does require as input just the disparity map to be analyzed. Thanks to the latter trait,

	AD-CENSUS		SGM	
τ	gt $\cap$ τ (%)	bad3 (%)	gt $\cap$ τ (%)	bad3 (%)
0.00	100.00	38.64	100.00	16.53
0.50	61.89	7.83	87.87	6.58
0.80	53.16	2.90	83.64	4.37
0.90	48.71	1.70	80.58	3.40
0.95	44.49	1.06	77.48	2.67
0.99	32.15	0.35	68.01	1.40

Table 8.1: Intersection between confident points and ground-truth data as function of the threshold value τ and its error rate, for both AD-Census and SGM algorithms.

CCNN can be applied to any stereo system, even in case one has no access to the source code of the algorithm or is willing to employ an off-the-shelf external device. As CCNN consists of a network trained to classify each disparity pixel as reliable or not according to a small support region, it needs to be trained before deployment. To avoid reliance on expensive depth annotations, we used the original authors' implementation[1] and trained two variants of the network - one for AD and the other for SGM - on synthetic images taken from the Sceneflow dataset [76]. More precisely, we took six random stereo pairs from the Driving portion of such dataset (0040, 0265 forward from 15mm focal length set and 0075 forward, 0099, 0122, 0260 backward from 35mm set) and trained CCNN for 14 epochs, as suggested in [80].

8.3.1 *Effectiveness of Confidence Estimation*

Before testing our unsupervised adaptation loss we carried out some preliminary evaluation of the effectiveness of CCNN [80] to detect mistakes in disparity maps. We have produced disparity maps using AD and SGM on the KITTI 2012[27] training dataset and compute confidence maps on the predictions with the two properly trained variants of CCNN. Tab. 8.1 reports both the intersection between confident (*i.e.*, having a confidence value higher than the threshold τ) and ground-truth pixels as the percentage of the total amount of available ground-truth samples and the average error

1 https://github.com/fabiotosi92/CCNN-Tensorflow

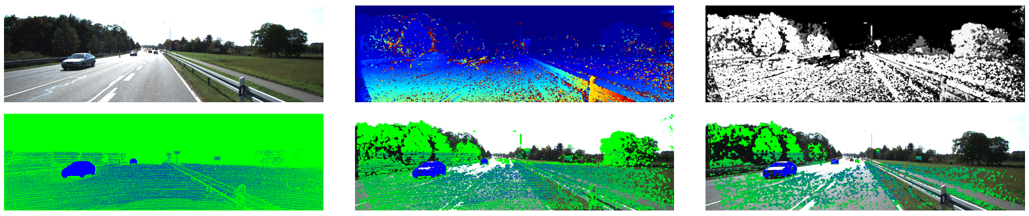


Figure 8.3: Spatial distribution of training samples on a stereo pair from KITTI dataset. Top row: reference image, disparity map yielded by AD-CENSUS algorithm and corresponding confidence map obtained by CCNN. Bottom row from left ot right, three colormaps obtained by thresholding the confidence map with with τ equal to 0, 0.5 and 0.99, respectively. The colormap depicts in green points above threshold and in blue their intersection with the available ground truth points obtained through a lidar scanner

rate (bad 3) on the intersection. As expected, increasing τ the intersection gets smaller as fewer pixels are considered, however, we can clearly see how the error decrease as well, going as low as 0.35% and 1.40% for 0.99 τ, thus testifying the effectiveness of CCNN. Our formulation, however, weights prediction mistakes according to the score produced by the confidence measure, so even using a $\tau < 0.99$ we can weaken the contribution to lose computation of pixel with low confidence. Another interesting observation is that the distribution of pixel with high confidence in the image is quite different from the distribution of points featuring gt labels commonly used for fine-tuning. For example using the threshold value of 0.99 and the AD-Census algorithm the subset of pixels that would be used by our confidence guided loss contains only 32% of the ground-truth data used by the common fine-tuning technique, while with the same threshold and the SGM algorithm this percentage rises to 68%. This means that all the remaining samples contributing to adaptation (*i.e.* 68 and 32% for, respectively, AD-CENSUS and SGM) encode patterns unseen using a traditional fine-tuning procedure. Thus, the network can learn from more varied and *generic* samples with respect to ground-truth which is, among other things, all contained in the lower part of the images. A visualization of how points with high confidence might be scattered across an image is depicted in Fig. 8.3

	Hyper parameters			Adaptation		Generalization	
Test	τ	λ_1	λ_2	bad3	MAE	bad3	MAE
(a) AD-CENSUS [3]	/	/	/	/	/	32.03	19.60
(b) No Adaptation	/	/	/	/	/	10.86	1.73
(c) Regression	/	/	/	11.73	2.49	12.23	2.47
(d) Weighted	0	0	0	3.66	1.03	4.57	1.12
(e) Masked	0.8	0	0	3.17	1.02	3.97	1.09
(f) Masked+Smoothness	0.8	0.1	0	3.17	0.98	3.78	1.05
(g) Masked+Reprojection	0.8	0	0.1	3.03	0.98	3.70	1.05
(h) Complete Adaptation	0.8	0.1	0.1	**2.96**	**0.96**	**3.66**	**1.04**

Table 8.2: Ablation study on the effectiveness of the different components of our adaptation formulation using AD-CENSUS as noisy label estimator. Results computed on the KITTI RAW dataset using a 4-fold cross validation schema, best results highlighted in bold.

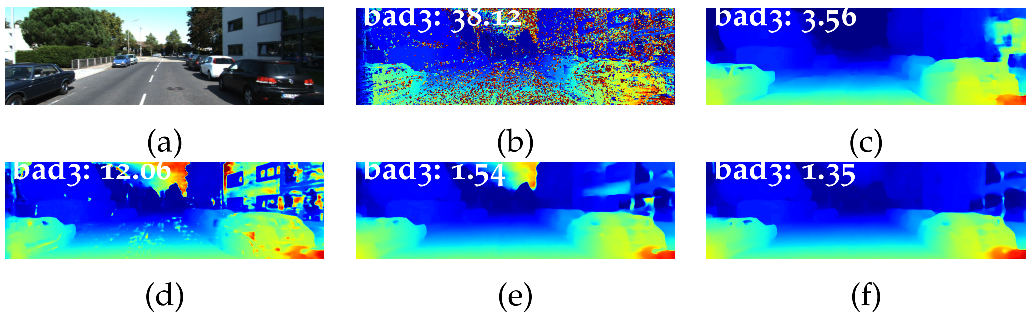


Figure 8.4: Ablation experiments: adaptation of DispNet using AD algorithm. (a) input image from KITTI, (b) disparity estimated using AD-CENSUS algorithm, (c) results before adaptation, (d) adapting by stereo algorithm only (*Regression*), (e) using confidence to weight the loss function (*Weighted*) and (f) running our full adaptation.

8.3.2 *Deep Stereo*

Our first experimental scenario is about the adaptation of a depth-from-stereo network to a new environment. The common training procedure for this kind of models consists of first training on the large synthetic FlyingThings3D dataset [76] and then fine-tuning on the target environment. In these settings, our proposal brings in the advantage of enabling fine-tuning without reliance on depth annotations from the target environment, which would be costly or prohibitive to collect. For all our tests we have used the DispNet-Corr1D [76] architecture, from now on shortened as DispNet. Following the authors' guidelines [76], we have trained a re-implementation of DispNet on FlyingThings3D by the standard and supervised L1 regression loss. Then, we have used these pre-trained weights as initialization for all the tests discussed hereinafter.

For our experiments we rely on the KITTI RAW [31] dataset, which features $\sim 43K$ images with depth labels [122] converted into disparities by known camera parameters. Images are taken from stereo video sequences concerning four diverse environments, namely *Road*, *Residential*, *Campus* and *City*, containing 5674, 28067, 1149 and 8027 frames, respectively. Although all images come from driving scenarios, each environment shows peculiar traits that would lead a deep stereo model to gross errors without suitable fine-tuning. For example, *City* and *Residential* often depict road surrounded by buildings, while *Road* mostly concerns highways and country roads where the most common objects are cars and vegetation. Using this data we wish to measure both *target domain* performance, *i.e.*, how the network performs on the target domain upon unsupervised adaptation without access to any ground-truth information, as well as *similar domains* performance, *i.e.*, how the network adapted without supervision generalizes to unseen images from similar domains. To analyze both behaviors, we have alternatively used one of the environments as the training set to perform fine-tuning, then tested the resulting model on all the four environments. In fact, this allows for assessing *target domain performance* by testing on the environment used for unsupervised fine-tuning and *similar domains performance* by testing on the other three. Since the environments amenable to perform fine-tuning are four, we can carry out 4-fold cross-validation in order to average performance figures. Hence, for each fold we average performance figures within an environment (*i.e.*, across all of its frames), obtaining, thereby, four sets of measurements. Then, we compute *target domain* performance by averaging the scores dealing with the four training sets in the corresponding four folds and *similar domains* performance by averaging across the other twelve scores.

As for the per-frame performance figures, we compute both the Mean Average Error (MAE) and the percentage of pixels with disparity error larger than 3 (bad3) as suggested in [27, 57]. Due to image formats being different across the KITTI RAW dataset, we extract a central crop of size 320×1216 from each frame, which matches to the downsampling factor of DispNet and allows for validating almost all pixels with respect to the available ground truth disparities.

Ablation

i) Can we simply use D as noisy ground truth without deploying C? ii) Is masking by τ really needed or could we just use C as a per-pixel weighting in L_c? iii) How important is the contribution of the additional loss terms L_s, L_r?

To answer the above questions, we set AD as the stereo algorithm, CCNN as confidence measure and run a set of experiments according to the cross-validation protocol described in Sec. 8.3.2. The resulting performance figures are reported in Tab. 8.2 as follows. Starting from the top row: (a) AD, *i.e.* the stereo algorithm providing us with the noisy labels, (b) DispNet trained only on synthetic data (*i.e.* the initial weights used for all the subsequent fine tunings), (c) DispNet fine-tuned to directly regress AD without deploying a confidence measure, (d) DispNet fine-tuned to minimize L_c with $\tau = 0$ (thereby weighting all disparities yielded by AD without masking out likely mistakes), (e-g) training to minimize different combinations of L_c, L_s and L_r using a fixed $\tau = 0.8$. The values for τ, λ_1 and λ_2 are obtained by preliminary cross-validation and will be kept fixed across all experiments. Since rows (a) and (b) do not need any kind of fine-tuning on KITTI we report the same performances for both *target* and *similar domains*.

To answer question i), we can compare results between rows (c) and (b). As expected. fine-tuning the network to directly regress the noisy measurements produced by AD is not a valid optimization strategy as it worsens the initial network performance both in the *target domain* as well as in *similar domains*. Interestingly, the network structure seems to behave as a regularizer and does not overfeat too much to the noise in the labels as testified by the huge performance gap between rows (c) and (a). To answer question ii), we can compare line (e) and (d), where the only difference is the value of τ. The presence of $\tau = 0.8$ in (e) helps to improve the performance by about 0.5% bad3 while obtaining comparable performance in MAE. These results testify how masking out disparity measurements that are likely wrong yields better performance even though it increases the sparsity of the gradient signal actually deployed to update the model. A possible explanation for the close performance gap between (d) and (e) may be ascribed to the confidence maps produced by CCNN being highly bi-modal, with the vast majority of pixels having confidence score either 0 or 1. Therefore, even without applying a fixed threshold, many completely mistaken labels will see their contribution masked out during loss computation. To answer question (iii) we can compare the performance reported in the last four rows. Adding L_s in the optimization process does not improve *target domain* performance but slightly helps in *similar domains*, as clearly observable by comparing rows (f) and (e). The introduction of L_r, instead, seems more effective and results in improvement across all metrics, as observable by comparing rows (g) and (e). Once again, larger improvements are obtained in case of unseen images from *similar domains*. Eventually, it is worth pointing out how our complete *Adaptation* loss yields the best results, as vouched by the performance figures reported in the row (h).

Fig. 8.4 shows qualitative results related to the ablation study proposed in this subsection. The top row depicts the reference image (a), the noisy disparities provided by AD (b) and the prediction produced by DispNet trained only on synthetic data (c). The bottom row, instead, reports three different predictions obtained by the three adaptation approaches referred to as *Regression* (d), *Weighted* (e) and *Complete* (f) in Tab. 8.2. By comparing (f) to (d) and (e) we can clearly verify that our adaptation schema can successfully mask out all the noise in the labels and learn only from good disparities. Moreover, we can perceive the effectiveness of our adaptation approach by comparing (f) to (c), for example by observing how it can significantly reduce the errors caused by the reflective surface on the right portion of the image, without at the same time introducing many artifacts, as unfortunately does happen in (c) and (d).

Comparison with other self supervised losses

In this section, we compare our proposal to other loss functions known in the literature that may be employed in order to fine-tune a deep stereo network without supervision. In particular, we consider two losses that, akin

	Target Domain		Similar Domains	
Loss	bad3	MAE	bad3	MAE
No Adaptation	/	/	10.86	1.73
GT Tuned (K12/15)	/	/	5.04	1.28
Godard et. al.[93]	4.01	1.07	4.20	1.09
Yinda et. al.[155]	3.59	1.00	5.15	1.14
Masked-AD + Smooth.	3.17	0.98	3.78	1.05
Masked-SGM + Smooth.	2.99	1.01	3.63	1.09
Adaptation-AD	2.96	0.96	3.66	1.04
Adaptation-SGM	**2.58**	**0.91**	**3.39**	**1.01**
Adaptation-AD-SGM	2.74	0.92	**3.39**	1.02

Table 8.3: Results obtained performing fine tuning of a pre-trained DispNet network using different unsupervised strategy. All results are computed on the KITTI raw dataset using a 4-fold cross validation schema, best results highlighted in bold, our proposals in italic.

to ours, rely only on stereo frames to achieve a form of self-supervision: the appearance based re-projection and smoothness loss by Godard et al. [93] and the local constraint normalization with window-based optimization loss of [155]. As the underlying principles and mechanisms are quite straightforward to reproduce, we have re-implemented the two losses following the authors' guidelines. Thus, we apply these alternative losses together with three variants of our proposal, relying either on AD or SGM or both stereo algorithms, in order to fine-tune DispNet upon pre-training on synthetic data. As an additional comparison, we also report results obtained by our loss formulation without the use of reprojection between frames. Again, we follow the same 4-fold cross validation protocol as in Sec. 8.3. Results are reported in Tab. 8.3 alongside with the performance of the pre-trained DispNet model (No Adaptation) and those attainable by fine-tuning the pre-trained model by the LiDAR groundtruth available for the 400 frames of the KITTI2012 [27] and KITTI2015 [57] training sets (GT Tuned), *i.e.* according to the standard training methodology adopted in the vast majority of works dealing with deep stereo. For the sake of fair comparison, all methods are evaluated based only on the disparity map predicted for the left frames of the stereo pairs and can not leverage additional external networks besides

DispNet (*i.e.*, as for [155] we do not deploy also an external Invalidation Network).

Tab. 8.3 shows that our proposal outperforms other approaches both in the *target domain* as well as *similar domain* experiments. In particular, *Adaptation-SGM* delivers the best performance across all metrics and scenarios, with gain as large as $\sim 1\%$ in the bad3 metric with respect to the closest alternatives known in literature beside our previous work (more than 25% relative improvement in the *target domain* and almost 20% in *similar domains*). The improvement is less substantial in the MAE figure, though our proposal still consistently outperforms alternative approaches. We also point out how even *Masked-AD + Smooth.* and *Masked-SGM + Smooth.* in Tab. 8.3, already outperforms competitors, which suggests the key component in our technique to be the confidence-guided loss. Yet, the novel *Adaptation* proposed here further ameliorates performance significantly. By comparing *Adaptation-AD* and *Adaptation-SGM* we can verify how a more accurate stereo algorithm (SGM vs AD) yields better performance. This can be ascribed to less noise in the disparities leading to a larger number of pixels scoring confidence $> \tau$ which, in turn, is conducive to denser and more accurate pseudo-groundtruths. Using both the disparity maps obtained by the two algorithms (*Adaptation-AD-SGM*) does not seem to help, yielding, indeed, slightly worse performance in the *target domain* and comparable performance in *similar domains* compared to *Adaptation-SGM*. We believe that mistakes made by AD and not detected by the confidence measure may explain the slight loss in performance. Finally, it is interesting to compare the performance achievable by fine tuning without supervision on many data (rows from 3 to 9) to those achievable by fine-tuning with supervision on few similar data (*i.e.*, *GT Tuned*). The large performance margin in favour of almost all the unsupervised approaches indicates that training on much more data with a sub-optimal objective turns out not only easier and cheaper but also beneficial to performance with respect to training on few, perfectly annotated samples (*e.g.*, 1.65 gain in bad3 and 0.27 in MAE by comparing *Adaptation-SGM* to *GT Tuned*).

8.3.3 *Depth-from-Mono*

To investigate the application of our approach to depth prediction from a single image, we run experiments based on the popular depth-from-

		Lower is better				Higher is better		
Supervision	**Encoder**	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard et al. [93]	VGG	0,124	1,076	5,311	0,219	0,847	0,942	0,973
Masked-AD	VGG	0.119	0.989	4.981	0.207	0.859	0.950	0.977
Masked-SGM	VGG	0.123	1.055	4.900	0.208	0.860	0.951	0.977
Godard et al. [93]	VGG+pp	0.118	0.923	5.015	0.210	0.854	0.947	0.976
Masked-AD	VGG+pp	0,111	0,871	4,852	0,199	0,858	0,952	0,980
Masked-SGM	VGG+pp	0,112	0,848	4,766	0,197	0,859	0,953	0,981
Godard et al. [93]	ResNet50+pp	0,114	0,898	4,935	0,206	0,861	0,949	0,976
Masked-AD	ResNet50+pp	**0,109**	0,867	4,810	0,197	0,866	0,953	0,979
Masked-SGM	ResNet50+pp	**0,109**	0,837	4,703	0,194	**0,867**	0,955	0,980
Adaptation-SGM	ResNet50+pp	**0,109**	**0,831**	**4,681**	**0,193**	**0,867**	**0,956**	**0,981**

Table 8.4: Experimental results on the KITTI dataset [31] on the data split proposed by Eigen et al. [37]. On even conditions, the proposed adaptation scheme outperforms the supervision by Godard et al. [93]. Combining the two produces the best results.

		Lower is better				Higher is better		
Configuration	**Encoder**	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Regression-AD	VGG+pp	0.209	2.121	7.788	0.402	0.639	0.818	0.900
Weighted-AD	VGG+pp	0.124	1.010	5.446	0.236	0.825	0.932	0.968
Masked-AD	VGG+pp	0.111	0.871	4.852	0.199	0.858	0.952	0.980
Regression-SGM	VGG+pp	0.136	1.697	5.540	0.220	0.848	0.942	0.973
Weighted-SGM	VGG+pp	0.117	0.983	4.987	0.202	0.857	0.951	0.979
Masked-SGM	VGG+pp	0.112	0.848	4.766	0.197	0.859	0.953	0.981
Masked-AD-SGM	VGG+pp	0.114	0.915	4.909	0.199	0.859	0.953	0.980
Regression-AD	ResNet50+pp	0.230	3.24	8.361	0.418	0.624	0.806	0.893
Weighted-AD	ResNet50+pp	0.120	0.952	5.288	0.225	0.836	0.937	0.971
Masked-AD	ResNet50+pp	0.109	0.867	4.810	0.197	0.866	0.953	0.979
Regression-SGM	ResNet50+pp	0.129	1.456	5.385	0.214	0.854	0.943	0.973
Weighted-SGM	ResNet50+pp	0.115	0.966	4.925	0.199	0.863	0.952	0.979
Masked-SGM	ResNet50+pp	0.109	0.837	4.703	0.194	0.867	0.955	0.980
Masked-AD-SGM	ResNet50+pp	0.110	0.866	4.775	0.195	0.867	0.955	0.980

Table 8.5: Ablation experiments on the KITTI dataset [31] on the data split proposed by Eigen et al. [37].

mono system developed by Godard et al. [93]. This choice is driven by two main factors i) despite a large number of works in this field [127, 140, 149, 151], it still represents one of the most effective solutions for unsupervised monocular depth estimation and ii) the image reconstruction loss proposed by Godard *et al*. represent the main competitor to our approach, thus comparison to [93] turns out the ideal test bench for our proposal.

The network proposed in [93], referred to here as *monodepth*, consists in a DispNet-like architecture featuring a backbone encoder followed by a decoder to restore the original input resolution and predict the final depth map. In [93], both VGG [62] and ResNet50 [70] were tested as encoders. The output is provided as disparity (*e.g.*, inverse depth), and used at training time to warp the stereo images. This also eases the use of our unsupervised adaptation technique, that could be deployed anyway also in case of architectures directly predicting depth by simply converting our disparity labels based on known camera parameters. Moreover, in [93] a post-processing step is proposed to deal with occlusions and artifacts inherited from stereo supervision, by producing both normal and flipped depth maps and combining them. We will run experiments with and without this optional step, referred to as '+pp'.

We start from the TensorFlow codebase provided by the authors of [93], adding our proposal therein and running experiments within the same framework to ensure perfectly fair test conditions.

Evaluation protocol:

We follow exactly the same protocol as reported in [93]. In particular, the KITTI raw dataset [31] is split into a training set and an evaluation set according to the guidelines by Eigen *et al*. [37]. Unlike the adopted stereo evaluation protocol [122], raw LiDAR measurements are usually assumed as groundtruth in the depth-from-mono literature despite their being sparse and noisy. Nonetheless, we adhere to the standard depth-from-mono evaluation protocol to ensure consistency with existing literature and enable a fair comparison with respect to [93].

Several works in this field [93, 127, 151] deploy pre-training on the City-Scapes dataset [67] before fine-tuning on the KITTI training split [37], [31]. Indeed, training only on KITTI leads to inferior accuracy due to the fewer training images, whilst training only on CityScapes let the network predicts depth maps of acceptable quality, but totally wrong in terms of the actual depth values. This scenario, thus, points out again how a domain shift severely affects the accuracy of depth-from-images networks, *i.e.* exactly

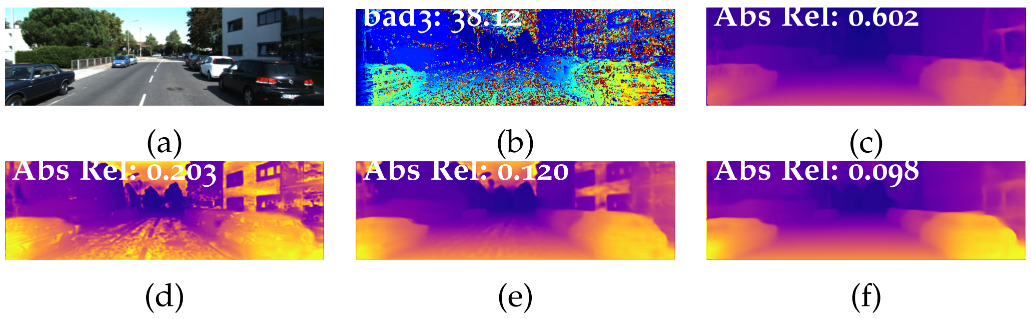


Figure 8.5: Ablation experiments: adaptation of monodepth (VGG encoder) using AD algorithm. a) input image from KITTI b) result from AD algorithm c) result before adaptation d) adapting with stereo algorithm only e) using confidence to weight the loss function f) running full adaptation.

the issue we aim to address by the general domain adaptation framework proposed in this paper. Therefore, to assess the effectiveness of our proposal also in depth-from-mono settings, we will start from models pre-trained on CityScapes in order to adapt them to KITTI. In particular, relying on the very same models pre-trained on CityScapes we compare the results attained on the KITTI test split by performing fine-tuning on the KITTI train split by either our approach or the reconstruction loss proposed in [93]. As for our method, we use the same stereo algorithms (AD and SGM), confidence measure (CCNN) and hyper-parameter settings as in depth-from-stereo experiments. Coherently to [93], we used the Adam optimizer and found that, while our competitor needs to run 50 epochs of training on KITTI, our method reaches convergence after only 5 epochs with a fixed learning rate of 0.001, thus resulting in faster and, as we shall see in the next section, more effective adaptation.

Results on KITTI:

We discuss here the outcomes of our experiments on the KITTI RAW dataset [31]. In particular, we report the standard error metrics, *i.e.* Absolute Relative error (Abs Rel), Square Relative error (Sq Rel), Root Mean Square Error (RMSE), logaritmic RMSE and the δ accuracy score computed as:

$$\delta = \tilde{D}_{i,j}\% : \max\left(\frac{\tilde{D}_{i,j}}{D_{i,j}}, \frac{D_{i,j}}{\tilde{D}_{i,j}}\right) < th \tag{8.8}$$

Tab. 8.4 reports a detailed comparison between the self-supervised loss proposed in [93] and our proposal, both applied to adapt the same pre-trained *monodepth* model. From top to bottom, we show the results dealing with a *monodepth* with a VGG encoder, a VGG encoder and the post-processing step (+pp) and finally a ResNet50 encoder with the post-processing step. Starting from the basic VGG-based model, we can observe that adapting by either AD or SGM with the masked configuration alone leads to better performance with respect to using the image reconstruction loss proposed in [93]. In general, adapting by SGM yields superior results, outperforming the model based on AD on nearly all metrics. This finding is confirmed when applying the post-processing step (i.e., VGG+pp), as our adaptation approach outperforms [93] under all evaluation metrics. Moreover, VGG+pp networks optimized by our technique can deliver better results than using a ResNet+pp network trained according to the image reconstruction loss of [93] despite the large difference in complexity between the two networks (VGG features about 31 millions learnable parameters, ResNet50 about 57 millions). By applying the considered adaptation approaches to the ResNet50-based *monodepth*, the margin turns out even higher. Finally, in the last row of Table 8.4, we report the results obtained by our approach when using SGM together with the complete adaptation loss of Eq. 8.2, the performance metrics scoring slightly better than those achieved by Masked-SGM. This, again, highlights how in Eq. 8.2 the confidence-guided loss term provides a more relevant contribution to performance than the image re-projection term inspired by [93].

Ablation experiments: Similarly to the previously addressed stereo settings, we report here an ablation study aimed at establishing upon the relative importance of the key ingredients deployed in our framework. Table 8.5 collects the results obtained in this evaluation. We comment about four main experiments, dealing with running our method with both AD and SGM in order to adapt VGG and ResNet50. The post-processing step is enabled in all tests, thereby solving most issues near occlusions and left border and highlighting how the full confidence-guided loss ameliorates results in many regions of the images where post-processing cannot operate. Three setups are considered in descending order in the Table for each of the four experiments: i) adaptation by minimization of the L1 loss with respect to the disparity maps estimated by the stereo algorithm (AD or SGM) "as is" (*Regression*) ii) adaptation by weighting the L1 loss with per-pixel confidence scores (*Weighted*) iii) full confidence-guided loss using threshold τ (*Masked*). We turn off additional terms to focus on the different key factors of the confidence-guided loss. In all experiments, we can notice how using the disparity labels alone leads to poor results, in particular when adapting the model by the AD algorithm, which is much more prone to outliers. This further highlights how, in our framework, deploying the confidence measure is crucial to avoid the impact of the wrong disparities possibly computed by the stereo algorithms. Formulating the confidence-guided loss as a simple weighting between confidence scores and loss signals reduces the impact of the outliers, but does not completely removes it as they can still contribute to the entire loss function with a lower weight and thus may lead, as reported, to worse performance. To better perceive this effect, Fig. 8.5 shows some qualitative results obtained by the three ablated configurations reported in the Table. In particular, we point out how on (c) the results from the original model trained on different environments look good qualitatively, but the range of the predicted depth values is totally wrong (Abs Rel of 0.620). We can observe how ablated configurations of our technique (d-e) do yield gradual improvements, whereas the full adaptation scheme (f) greatly ameliorates the quality of the estimated depth maps, *i.e.* so as to bring the error down to 0.098 Abs Rel.

As an additional experiment, and similarly to the stereo experiments, we tried to adapt the model using labels from both AD and SGM by simply computing the loss function according to both disparity hypotheses. This leads to slightly worse results, possibly because some outliers not correctly masked out from the one algorithm may interfere with the correct assignments suggested by the other.

8.3.4 *Qualitative Results*

Finally we show some qualitative results, concerning both stereo and depth-from-mono networks, on the Middlebury v3 [42] and ETH3D [112] datasets. Fig. 8.6 shows examples of depth maps obtained by *monodepth* pre-trained on CityScapes [67] before and after adaptation by our technique. The overall quality of the maps is greatly improved by the adaptation step, which is also vouched by the drastic drop of the absolute error reported in the Figure. We show similar results for DispNet on Fig. 8.7: the column labeled as *No Adaptation* concerns predictions obtained by the model pre-trained on FlyingThings3D while the *Adaptation* column deals with the results obtained after fine-tuning by our unsupervised adaptation approach. Results indicate clearly how our proposal can successfully correct the prediction range and drastically reduce the percentage of wrong pixels.

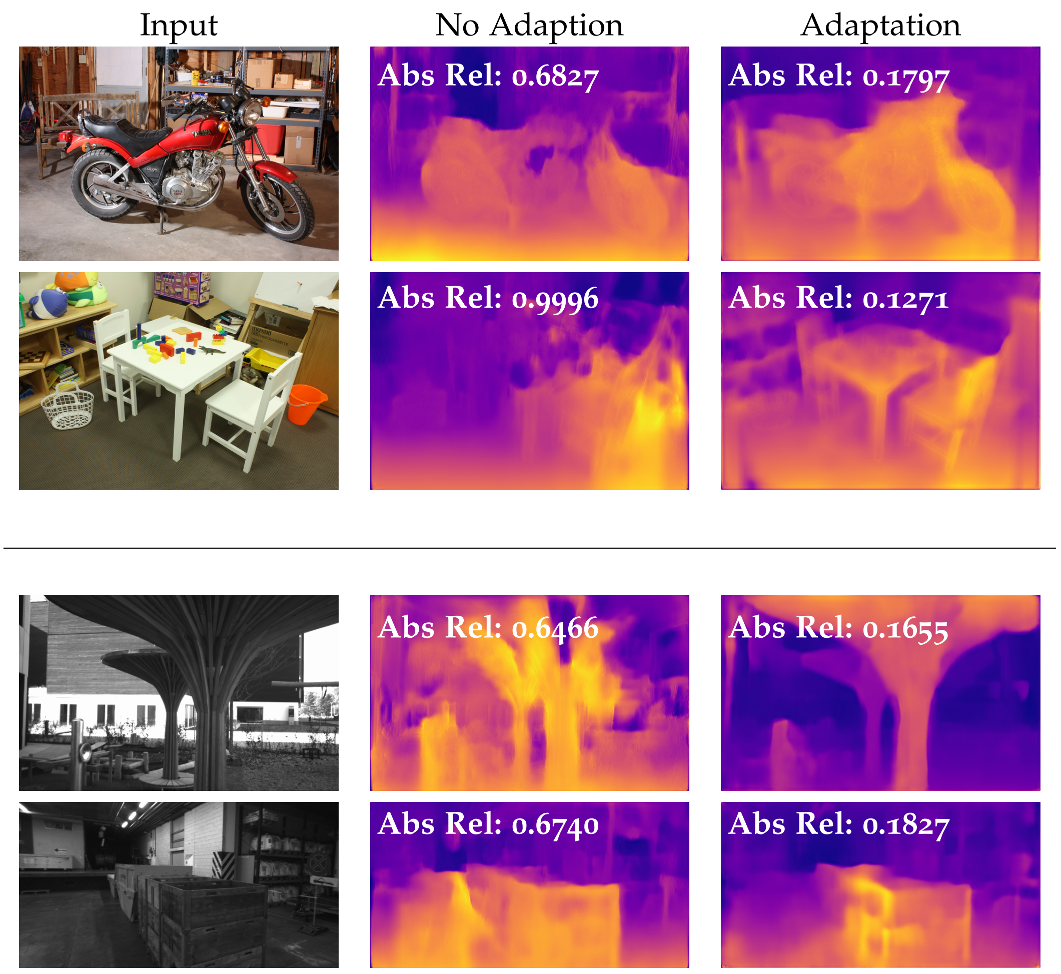


Figure 8.6: Adaptation results for depth-from-mono on Middlebury v3 [42] (top) ETH3D dataset [112] (bottom). From left to right: input (left) image, depth maps from network before adaptation and after fine tuning with our adaptation technique. The absolute error rate is overimposed on each depth map.

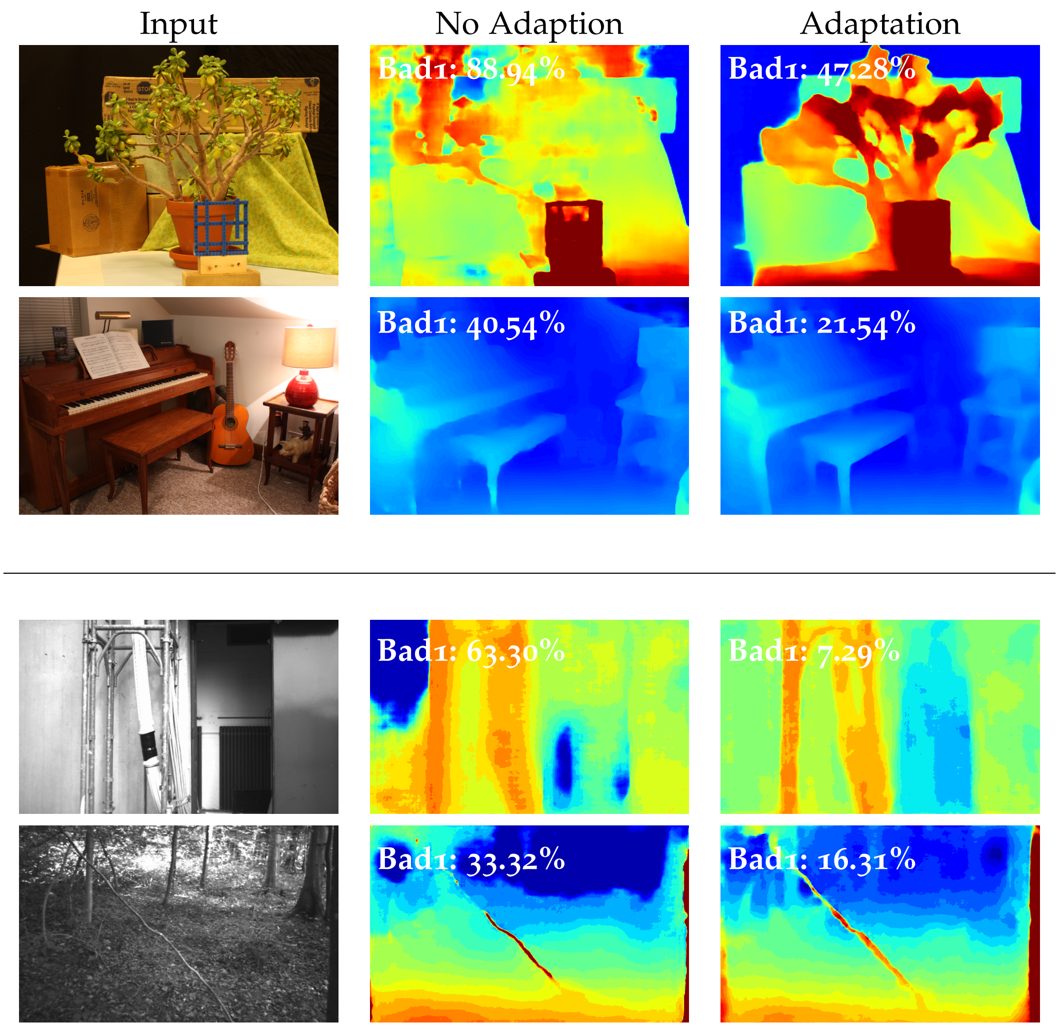


Figure 8.7: Adaptation results for Dispnet on Middlebury v3 [42] (top) ETH3D dataset [112] (bottom). From left to right input (left) image, disparity maps predicted from network before any adaptation and after fien tuning with our adaptation technique. The bad1 error rate is overimposed on each disparity map.

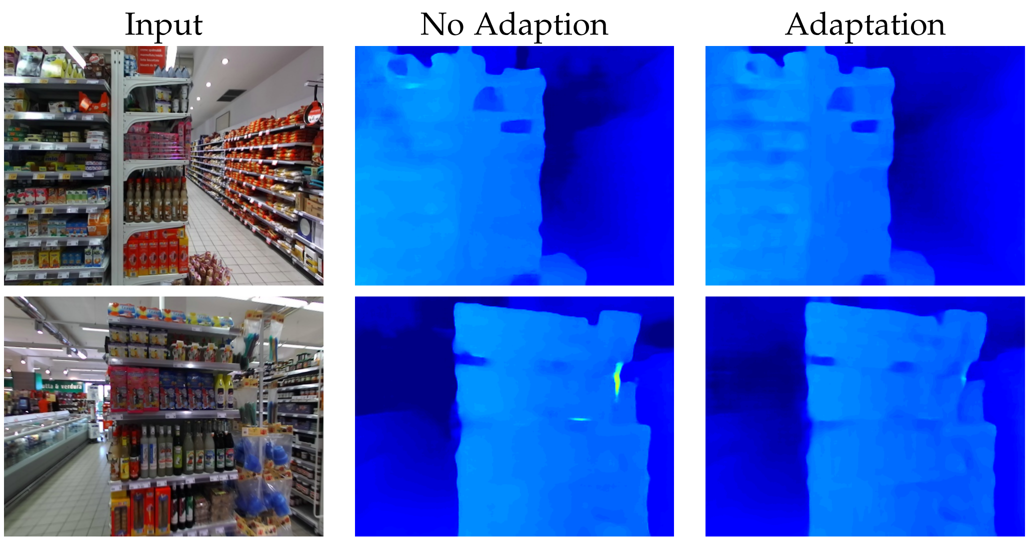


Figure 8.8: Adaptation results for Dispnet on stereo frames acquired in a supermarket with a ZED stereo camera

8.3.5 *Qualitative evaluation on Supermarket Environment*

We are interested in testing the effectiveness of our adaptation schema in the supermarket scenario, therefore we acquired video sequences using a ZED stereo camera inside a store and split them in a training and test set. For each acquisition this kind of commercial sensor provides left and right RGB frames together with disparities and confidences estimated using proprietary algorithms. Therefore instead of relying on the stereo algorithm and confidence measures used in all the other tests we directly used this kind of measurements provided by the sensor to fine tune a stereo network according to the loss formulation described in Sec. 8.2. We report a couple of qualitative results in Fig. 8.8. First we would like to point out how the disparity predicted from DispNet without any kind of adaptation are relatively good except for the presence of errors due to reflective surface (*e.g.*, the image on the second row) or mistakes for points far from the camera (*e.g.*, the area on the right of the shelf in the first row seems truncated after a certain distance). After the adaptation both types of mistakes seems mitigate with the reflection being mostly solved and the predictions for points far from the camera being way more smooth.

9 ONLINE UNSUPERVISED DOMAIN ADAPTATION FOR DEEP STEREO

In Chap. 8 we have proposed a novel loss function to address the domain shift issue for depth estimation networks by means of unsupervised offline fine tuning. In this section, instead, we want to go one step further and cast *adaptation* as a *continuous unsupervised learning* process whereby a depth estimation network evolves *online* based on the images gathered by a camera during its real deployment. We believe that the ability to constantly adapt itself in real-time is key to any deep learning machinery meant to work in relevant practical scenarios, like autonomous driving, where gathering training samples to be deployed offline from all possible surroundings turns out definitely unfeasible.

We will show how continuous online adaptation can be achieved by leveraging on one of the unsupervised losses proposed in the literature to compute the loss on the current frames, update the whole network by back-propagation (from now on shortened *back-prop*) and move to the next couple of input frames. Adapting, however, comes with the side effect of reducing inference speed to $\sim \frac{1}{3}$ at best, as we will show experimentally, a price far too high to pay for most modern state of the art deep stereo systems which are relatively slow. To keep a high enough frame rate we have developed a novel <u>M</u>odularly <u>AD</u>daptive <u>Net</u>work (*MADNet*) designed to be lightweight, fast and modular. Our architecture exhibits accuracy comparable to DispNet yet use $\frac{1}{10}$ of the parameters, runs at ~ 40FPS for disparity inference and performs online adaptation of the whole network at ~ 15FPS. Moreover, to allow a higher frame rate during the adaptation at the cost of slightly worse accuracy, we have developed <u>M</u>odular <u>AD</u>aptation (*MAD*) an algorithm that leverage on the modular architecture of *MADNet* and train independently sub-portion of the whole network. Using *MADNet* together with *MAD* we are able to adapt our system without supervision to unseen environment at ~ 25FPS.

Fig. 9.1 shows the disparity maps predicted by *MADNet* on three successive frames of a video from the KITTI dataset [31] either without undergoing any adaptation (b) or by adapting online the *whole* network (c) or by our

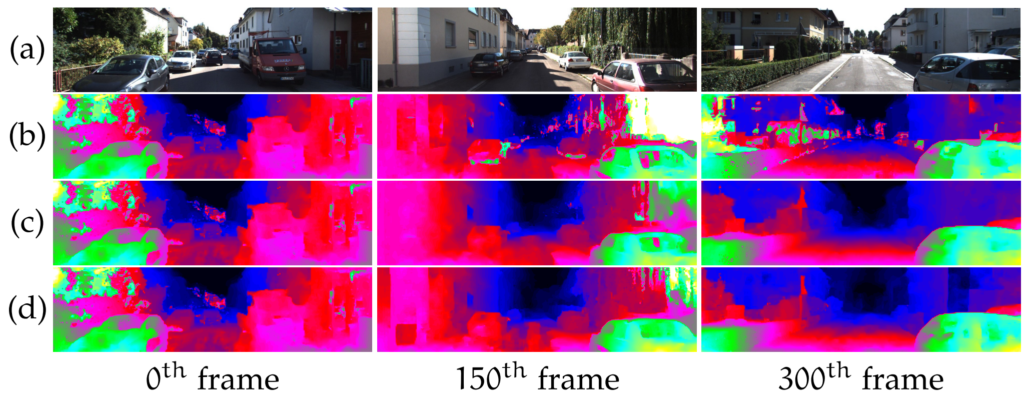


Figure 9.1: Disparity maps predicted by *MADNet* on a KITTI sequence [31]. Left images (a), no adaptation (b), online adaptation of the *whole* network (c), online adaptation by *MAD* (d).

computationally efficient approach *MAD* (d). Thus, (c) and (d) show how online adaptation can improve significantly the quality of the predicted disparity maps in as few as 150 frames, *i.e*. with a latency of about 10 and 6 seconds for whole adaption and *MAD*, respectively. Extensive experimental results on KITTI raw dataset [31] will evidence that i) our network is more accurate than models with similar complexity (*e.g*., [138]) ii) it can dramatically boost its accuracy in few hundred frames (*i.e*., a couple of seconds) to be comparable to offline fine-tuning over ground truth. To the best of our knowledge, *MADNet* and *MAD* realize the first-ever real-time, self-adapting, deep stereo system.

9.1 RELATED WORK

A recent trend to train deep depth estimation networks in an unsupervised manner relies on image reconstruction losses. In particular, for monocular depth estimation this is achieved by warping different views, coming from stereo pairs or image sequences, and minimizing the reconstruction error [68, 93, 127, 144, 145, 155]. This principle has been used also for optical flow [141] and stereo [126]. For the latter task, an alternative unsupervised learning approach consists in: deploying traditional stereo algorithms and confidence measures (see Chap. 8) or by combining with iterative optimization prediction obtained at multiple resolutions [143]. However, we point out that all these works have addressed offline training only, while we propose to address the very same problem casting it as an online (thus fast) adaptation to an unseen environment. Concurrently with our work Zhong *et* *al.* [156] propose to directly train from scratch a deep stereo system on video sequences in an unsupervised manner, but this process takes several minutes limiting its effectiveness in case of sudden changes. In contrast, we maintain the offline training phase on synthetic data and limit the online training to solve domain adaptation dramatically reducing runtime requirements.

9.2 ONLINE DOMAIN ADAPTATION

Modern machine learning models can suffer from a loss in accuracy when tested on data significantly different from the training set, an issue commonly referred to as *domain shift*. Despite all the research work done to soften this issue, the most effective practice still relies on additional offline training on samples from the target environments to get a new model tuned to that scenario. The domain shift curse is inherently present in deep stereo networks due to most training iterations being performed on synthetic images quite different from real ones. Then, adaptation can be effectively achieved by offline fine-tuning the model on samples from the target domain either relying on expensive annotations or using unsupervised loss functions (Chap. 8 or [68, 93, 155]).

In this chapter we move one step further and argue that adaptation can be effectively performed online as soon as new frames are available, thereby obtaining a deep stereo system capable to adapt itself dynamically. For our online adaptation strategy we do not rely on the availability of ground truth annotations and instead, use one of the proposed unsupervised loss. To adapt the model we perform a single train iteration (forward and backward pass) for each incoming stereo pair on-the-fly. Therefore, our model is always in training mode and continuously fine-tuning to the sensed environment.

9.2.1 MADNet - *Modularly ADdaptive Network*

We believe that one of the main limitations that have prevented exploration of online adaptation is the computational cost of performing a full train iteration for each incoming frame. Indeed, we will show experimentally how it roughly corresponds to a reduction of the inference rate of the system to roughly one third, a price far too high to be paid with most modern architectures. To address this issue, we have developed Modularly ADdaptive

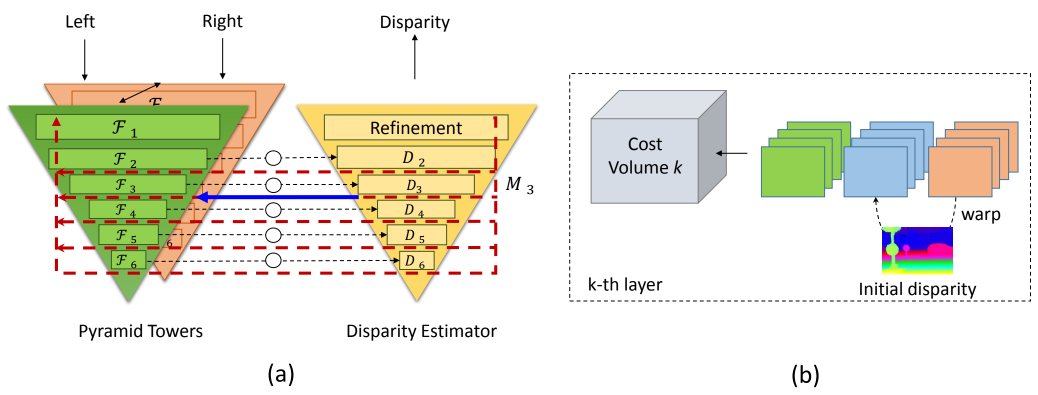


Figure 9.2: Schematic description of the proposed *MADNet* architecture (a), each circle between an F_k and the corresponding D_k representing a warp and correlation layer (b). Each pair ($\mathcal{F}_i$,$\mathcal{D}_i$) composes a module $\mathcal{M}_i$, adaptable independently by means of *MAD* (blue arrow) much quickly compared to performing full back-prop (red arrow)

.

Network (*MADNet*), a novel lightweight model for depth estimation inspired by very recent fast, yet accurate, architectures for optical flow [109, 147].

We deploy a pyramidal strategy for dense disparity regression for two key purposes: i) maximizing speed and ii) obtaining a modular architecture as depicted in Fig. 9.2. Two pyramidal towers extract features from the left and right frames through a cascade of independent modules sharing the same weights. Each module consists of convolutional blocks aimed at reducing the input resolution by two 3×3 convolutional layers, respectively with stride 2 and 1, followed by Leaky ReLU non-linearities. According to Fig. 9.2, we count 6 blocks providing us with feature representations $\mathcal{F}$ from half resolution to $1/64$, namely $\mathcal{F}_1$ to $\mathcal{F}_6$, respectively. These blocks extract 16, 32, 64, 96, 128 and 192 features.

At the lowest resolution (*i.e.*, $\mathcal{F}_6$), we forward the features from left and right images into a correlation layer [76] to get the raw matching costs. Then, we deploy a disparity decoder $\mathcal{D}_6$ consisting of 5 additional 3×3 convolutional layers, with 128, 128, 96, 64, and 1 output channels. Again, each layer is followed by Leaky ReLU, except the last one, which provides the disparity map at the lowest resolution.

Then, D_6 is up-sampled to level 5 by bilinear interpolation and used both for warping right features towards left ones before computing correlations and as input to $\mathcal{D}_5$. Thanks to our design, from $\mathcal{D}_5$ onward, the aim of the disparity decoders $\mathcal{D}_k$ is to refine and correct the up-scaled disparities coming from the lower resolution. In our design, the correlation scores computed between the original left and right features aligned according to the lower resolution disparity prediction should provide the network with guidance for the refinement process. We compute all correlations inside our network along a [-2,2] range of possible shifts.

This process is repeated up to quarter resolution (*i.e.*, $\mathcal{D}_2$), where we add a further refinement module consisting of 3×3 dilated convolutions [147], with, respectively 128, 128, 128, 96, 64, 32, 1 output channels and 1, 2, 4, 8, 16, 1, 1 dilation factors, before bilinearly upsampling to full resolution.

MADNet has a smaller memory footprint and delivers disparity maps much more rapidly than other more complex networks such as [**liang2018learning**, 99, 132] with a small loss in accuracy. Concerning efficiency, working at decimated resolutions allows for computing correlations on a small horizontal window [147], while warping features and forwarding disparity predictions across the different resolutions enables to maintain a small search range and look for residual displacements only. With a 1080Ti GPU, *MADNet* runs at about 40 FPS at KITTI resolution and can performs online adaptation with full back-prop at 15 FPS.

9.2.2 MAD - *Modular ADaptation*

As we will show, *MADNet* is remarkably accurate while performing the online adaptation of the whole network at 15 FPS. However, for some applications, it might be desirable to have a higher frame rate without losing the ability to tune the network to unseen environments automatically. Most of the time needed to perform online adaptation is effectively spent executing back-prop and weights update across the whole network layers. A naive way to speed up the process will be to *freeze* the initial part of the network and fine tune only a subset of k final layers, thus realizing a shorter back-prop that would yield a higher frame rate. However, there is no guarantee that these last k layers are indeed those that would benefit most from online fine-tuning. One might reasonably guess that other portions of the network should be adapted as well, in particular, the initial layers, which directly interacts with the images from a new, *unseen*, domain. Indeed, we will show in Sec. 9.3.3 that training only the final layers is not enough for handling drastic domain changes that typically occur in practical applications.

Thus, following the key intuition that to keep up with fast inference we should pursue a partial, though effective, online adaptation, we developed Modular ADaptation (*MAD*) an approximate online adaptation algorithm tailored for *MADNet*, but extensible to any multi-scale inference network. The key intuition for our method is to take a network $\mathcal{N}$ and subdivide it into p non-overlapping portions, each referred to as n, such that $\mathcal{N} = [n_1, n_2, ..n_p]$. Each portion n_i consists of k connected layers with the last one (the deeper) able to output a disparity estimation of y_i. Given as input a couple of stereo frames x, $\mathcal{N}$ will output $p+1$ disparity estimation $[y, y_1, \dots, y_p] = \mathtt{forward}(\mathcal{N}, x)$, with y being the final prediction of the model at full resolution. Due to its modular structure, *MADNet* naturally allows such subdivision by assigning to the same n_i layers working at the same resolution both in the convolutional towers and among the disparity estimators (*e.g.*, referring to Fig. 9.2 $n_i = [F_i, D_i]$). Each n_i can be independently trained by computing a loss directly on the disparity y_i and back-prop only across the layers belonging to n_i, thereby skipping the costly backward pass through lower resolution layers in between. This process can be figured out as an approximation of the standard full back-prop. In fact, by alternating the portion n_i of the network that is trained, it sequentially allows updating all layers. Such a strategy drastically reduces the time needed to compute and apply gradients without sacrificing too much accuracy as confirmed by experimental results.

Offline, we statically subdivide the network into different portions, that with *MADNet* correspond to different layers of the pyramids. Online, for each incoming stereo pair, first we perform a forward pass to obtain all the disparity predictions at multiple scales $[y, y_1, \dots, y_p]$, then, we choose a portion $\theta \in [1, \dots, p]$ of the network to train according to some heuristic and update the weights of layers belonging to n_θ according to a loss computed on y_θ. We consider a valid heuristic any function that outputs a probability distribution among the p trainable portions of N from whom we could perform sampling.

Among different functions, we obtained effective results using a simple reward/punishment mechanism detailed in Algorithm Alg. 2. We start by creating a histogram $\mathcal{H}$ with p bins all initialized at 0. For each stereo pair we perform a forward pass to get the disparity predictions (line 5) and measure the performance of the model by computing the loss $\mathcal{L}_t$ using the full resolution disparity y and potentially the input frames x, *e.g.*, reprojection error between left and right frames [93] (line 6). Then, we sample the portion to train $\theta \in [1, \dots, p]$ according to the distribution $\mathtt{softmax}(\mathcal{H})$ (line 7) and compute one optimization step for layers of n_θ with respect

Algorithm 2 Independent online training of *MADNet*

```
1: Require: $\mathcal{N} = [n_1, \dots, n_p]$
2: $\mathcal{H} = [h_1, \dots, h_p] \leftarrow 0$
3: while not stop do
4:     $x \leftarrow \mathtt{readFrames}()$
5:     $[y, y_1, \dots, y_p] \leftarrow \mathtt{forward}(\mathcal{N}, x)$
6:     $\mathcal{L}_t \leftarrow \mathtt{loss}(x, y)$
7:     $\theta \leftarrow \mathtt{sample}(\mathtt{softmax}(\mathcal{H}))$
8:     $\mathcal{L}_t^\theta \leftarrow \mathtt{loss}(x, y_\theta)$
9:     $\mathtt{updateWeights}(\mathcal{L}_t^\theta, n_\theta)$
10:    if firstFrame then
11:        $\mathcal{L}_{t-2} \leftarrow \mathcal{L}_t, \mathcal{L}_{t-1} \leftarrow \mathcal{L}_t, \theta_{t-1} \leftarrow \theta$
12:    $\mathcal{L}_{exp} \leftarrow 2 \cdot \mathcal{L}_{t-1} - \mathcal{L}_{t-2}$
13:    $\gamma \leftarrow \mathcal{L}_{exp} - \mathcal{L}_t$
14:    $\mathcal{H} \leftarrow 0.99 \cdot \mathcal{H}$
15:    $\mathcal{H}[\theta_{t-1}] \leftarrow \mathcal{H}[\theta_{t-1}] + 0.01 \cdot \gamma$
16:    $\theta_{t-1} \leftarrow \theta_t, \mathcal{L}_{t-2} \leftarrow \mathcal{L}_{t-1}, \mathcal{L}_{t-1} \leftarrow \mathcal{L}_t$
```

to the loss L_t^θ computed on the lower scale prediction y_θ (line 8-9). We have now partially adapted the network to the current environment. Next, we update $\mathcal{H}$ increasing the probability of being sampled for the n_i that have proven effective. To do so, we can compute a noisy expected value for $\mathcal{L}_t$ by linear interpolation of the losses at the previous two time step: $L_{exp} = 2 \cdot \mathcal{L}_{t-1} - \mathcal{L}_{t-2}$ (line 13). By comparing it with the measured $\mathcal{L}_t$ we can assess the impact of the network portion sampled at the previous step (θ_{t-1}) as $\gamma = L_{exp} - \mathcal{L}_t$, and then increase or decrease its sampling probability accordingly (*i.e.*, if the adaptation was effective $\mathcal{L}_{exp} > \mathcal{L}_t$, thus $\gamma > 0$). We found out that adding a temporal decay to $\mathcal{H}$ helps increasing the stability of the system, so the final update rule for each step is: $\mathcal{H} = 0.99 \cdot \mathcal{H}, \mathcal{H}[\sigma_{\tau-1}] + = 0.01 \cdot \gamma$ (lines 15 and 16). We will show how with the proposed *MAD* we can online adapt *MADNet* at $\sim$ 25FPS without losing much accuracy.

9.3 EXPERIMENTAL RESULTS

To properly address practical deployment scenarios in which there is no ground truth data available for fine-tuning in the actual testing environments, we train our stereo network using only synthetic data [76].

To test the online adaptation we use those weights as a common initialization and carry out an extensive evaluation on the large and heterogeneous

KITTI raw dataset [31] with depth labels [122] converted into disparities by knowing the camera parameters. Overall, we assess the effectiveness of our proposal on 43k images. Specifically, according to the KITTI classification, we evaluate our framework in four heterogeneous environments, namely *Road*, *Residential*, *Campus* and *City*, obtained by concatenation of the available video sequences and resulting in 5674, 28067, 1149 and 8027 frames respectively. Although these sequences are all concerned with driving scenarios, each has peculiar traits that would lead a deep stereo model to gross errors without a suitable fine-tuning in the target domain. For example, *City* and *Residential* often depict road surrounded by buildings, while *Road* mostly concern highways and country roads where the most common objects depicted are cars and vegetation.

By processing stereo pairs within sequences, we can measure how well the network adapts, either with full back-prop or *MAD*, to the target domain compared to an offline trained model. For all experiments, we analyze both average End Point Error (EPE) and the percentage of pixels with disparity error larger than 3 (D1-all). Due to image format being different for each sequence, we extract a central crop of size 320×1216 from each frame, which suits to the downsampling factor of our architecture and allows for validating almost all pixels with available ground-truth disparities.

Finally, we highlight that for both full back-prop and *MAD*, we compute the error rate on each frame *before* applying the model adaptation step. That is, we measure performances achieved by the current model on the stereo frame at time t and *then* adapt it according to the current prediction. Therefore, the model update carried out at time t will affect the prediction only from frame $t+1$ and so on. As unsupervised loss for online adaptation, we rely on the photometric consistency between the left frame and the right one reprojected according to the predicted disparity. Following [93], to compute the reprojection error between the two images we combine the Structural Similarity Measure (SSIM) and the L1 distance, weighted by 0.85 and 0.15, respectively. We selected this unsupervised loss function as it is the fastest to compute among those proposed in literature (Chap. 8 and [126, 143] and does not require additional information besides a couple of stereo pairs.

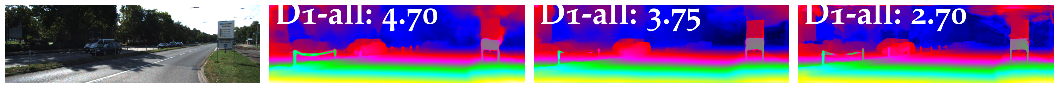


Figure 9.3: Qualitative comparison between disparity maps from different architectures. From left to right, reference image from KITTI 2015 online benchmark and disparity map by DispNetC [76], StereoNet [138] and *MADNet*.

Model	D1-all	Runtime
DispNetC [76]	4.34	0.06
StereoNet [138]	4.83	0.02
MADNet (Ours)	4.66	0.02

Table 9.1: Comparison between fast disparity regression architectures on the KITTI 2015 test set without online adaptation.

9.3.1 *MADNet performance*

Before assessing the performance obtainable through online adaptation, we test the effectiveness of *MADNet* by following the canonic two-phase training using synthetic [76] and real data. Thus, after training on synthetic data, we perform fine-tuning on the training sets of KITTI 2012 and KITTI 2015 and submit to the KITTI 2015 online benchmark. On Tab. 9.1 we report our result compared to other (published) fast inference architectures on the leaderboard (runtime measured on nVidia 1080Ti). At the time of writing, our method ranks 90th. Despite the mid-rank achieved in terms of absolute accuracy, *MADNet* compares favorably to StereoNet [138] ranked 92nd, the only other high frame rate proposal on the KITTI leaderboard. Moreover, we get close to the performance of the original DispNetC [76] while using $\frac{1}{10}$ of the parameter and running at more than twice the speed. Figure 9.3 also reports a qualitative comparison between the output of the three architectures, showing how *MADNet* better maintains thin structures compared to StereoNet.

9.3.2 *Online Adaptation*

We will now show how online adaptation is an effective paradigm, comparable to or better than offline fine-tuning. Tab. 9.2 reports extensive experiments on the four different KITTI environments. We report results achieved by i) DispNetC [76] implemented in our framework and trained,

		City		Residential		Campus		Road		
Model	**Adapt.**	D1-all(%)	EPE	D1-all(%)	EPE	D1-all(%)	EPE	D1-all(%)	EPE	FPS
DispNetC	No	8.31	1.49	8.72	1.55	15.63	2.14	10.76	1.75	15.85
DispNetC	Full	4.34	1.16	3.60	1.04	8.66	1.53	3.83	1.08	5.22
DispNetC-GT	No	3.78	1.19	4.71	1.23	8.42	1.62	3.25	1.07	15.85
MADNet	No	37.42	9.96	37.41	11.34	51.98	11.94	47.45	15.71	39.48
MADNet	Full	2.63	1.03	2.44	0.96	8.91	1.76	2.33	1.03	14.26
MADNet	*MAD*	5.82	1.51	3.96	1.31	23.40	4.89	7.02	2.03	25.43
MADNet-GT	No	2.21	0.80	2.80	0.91	6.77	1.32	1.75	0.83	39.48

Table 9.2: Performance on the *City*, *Residential*, *Campus* and *Road* sequences from KITTI [31]. Experiments with DispNetC [76] (top) and *MADNet* (bottom) with and without online adaptations.

from top to bottom, on synthetic data following authors' guidelines, using online adaptation or fine-tuned on ground-truth and ii) *MADNet* trained with the same modalities and, in addition, using *MAD*. These experiments, together to Sec. 9.3.1, support the three-fold claim of this work.

DispNetC: Full adaptation. On top of Tab. 9.2, focusing on the D1-all metric, we can notice how running full back-prop online to adapt DispNetC [76] decimates the number of outliers on all scenarios compared to the model trained on the synthetic dataset only. In particular, this approach can consistently halve D1-all on *Campus*, *Residential* and *City* and nearly reduce it to one third on *Road*. Alike, the average EPE drops significantly across the four considered environments, with improvement as high as a nearly 40% relative improvement on the Road sequences. These massive gains in accuracy, though, come at the price of slowing the network down significantly to about one-third of the original inference rate, *i.e.* from nearly 16 to 5.22 FPS. As mentioned above, the Table also reports the performance of the models fine-tuned offline on the 400 stereo pairs with ground-truth disparities from the KITTI 2012 and 2015 training dataset [27, 57]. It is worth pointing out how online adaptation by full back-prop turns out competitive with respect to fine-tuning offline by ground-truth, even more accurate in the Residential environment. This fact may hint at training without supervision by a more considerable amount of data possibly delivering better models than supervision by fewer data.

***MADNet*: Full adaptation.** On bottom of Tab. 9.2 we repeat the aforementioned experiments for *MADNet*. Due to the much higher errors yielded by the model trained on synthetic data only, full online adaptation turns out even more beneficial with *MADNet*, leading to a model which is more accurate than DispNetC with Full adaptation in all sequences but *Campus* and can run nearly three times faster (*i.e.* at 14.26 FPS compared to the 5.22 FPS of DispNetC-Full). These results also highlight the inherent effectiveness of the proposed *MADNet*. Indeed, as vouched by the rows dealing with *MADNet*-GT and DispNetC-GT, using for both our implementations and training them following the same standard procedure in the field, *MADNet* yields better accuracy than DispNetC while running about 2.5 times faster.

***MADNet*: *MAD*.** Having proven that online adaptation is feasible and beneficial, we show that *MADNet* employing *MAD* for adaptation (marked as *MAD* in column *Adapt.*) allows for effectively and efficient adaptation. Since the proposed heuristic has a non-deterministic sampling step, we have run the tests regarding *MAD* five times each and reported here the average performance. We refer the reader to Sec. 9.3.3 for analysis on the standard deviation across different runs. Indeed, *MAD* provides a significant improvement in all the performance figures reported in the table compared to the corresponding models trained by synthetic data only. Using *MAD*, *MADNet* can be adapted paying a relatively small computational overhead which results in a remarkably fast inference rate of about 25 FPS. Overall, these results highlight how, whenever one has no access to training data from the target domain beforehand, online adaptation is feasible and definitely worth. Moreover, if speed is a concern *MADNet* combined with *MAD* provides a favorable trade-off between accuracy and efficiency.

Short-term Adaptation. Tab. 9.2 shows also how all adapted models perform significantly worse on *Campus* with respect to the other sequences. We ascribe this mainly to *Campus* featuring fewer frames (1149) compared the other sequences (5674, 28067, 8027), which implies a correspondingly lower number of adaptation steps being executed online. Indeed, a key trait of online adaptation is the capability to improve performance as more and more frames are sensed from the environment. This favourable behaviour, not captured by the average error metrics reported in Tab. 9.2, is highlighted in Fig. 9.4, which plots the D1-all error rate over time for *MADNet* models in the four modalities. While without adaptation the error keeps being always large, models adapted online clearly improve over time such that, after a certain delay, they become as accurate as the model that could have been obtained by offline fine-tuning had ground-truth disparities been available. In particular, full online adaptation achieves performance comparable to fine-tuning by the ground-truth after 900 frames (*i.e.*, about 1 minute) while for *MAD* it takes about 1600 frames (*i.e.*, 64 seconds) to reach an almost equivalent

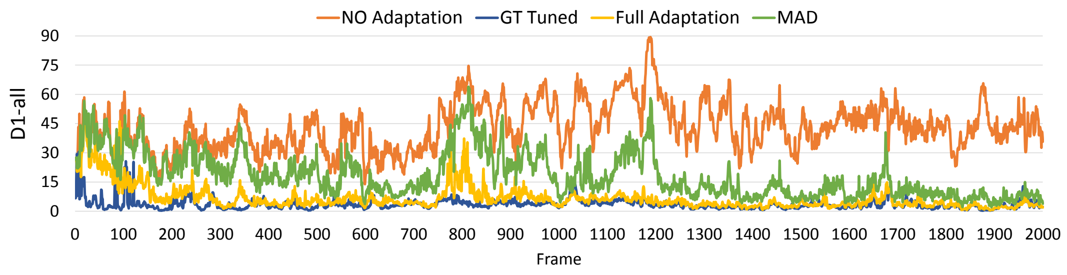


Figure 9.4: *MADNet*: error across frames in the *2011_09_30_drive_0028_sync* sequence (KITTI dataset, *Residential* environment).

performance level while providing a substantially higher inference rate (~ 25 vs ~ 15).

Long-term Adaptation. As Fig. 9.4 hints at online adaptation delivering better performance when processing a higher number of frames, in Tab. 9.3 we report additional results obtained by concatenating together the four environments without network resets to simulate a stereo camera traveling across different scenarios for ~ 43000 frames. Firstly, Tab. 9.3 shows how both DispNetC and *MADNet* models adapted online by full back-prop yield much smaller average errors than in Tab. 9.2, as small, indeed, as to outperform the corresponding models fine-tuned offline by ground-truth labels. Hence, performing online adaptation through long enough sequences, even across different environments, can lead to more accurate models than offline fine-tuning on few samples with ground-truth, which further highlights the great potential of our proposed *continuous learning* formulation. Moreover, when leveraging on *MAD* for the sake of run-time efficiency, *MADNet* attains larger accuracy gains through *continuous learning* than before (Tab. 9.3 vs.Tab. 9.2) shrinking the performance gap between *MAD* and Full back-prop. We believe that this observation confirms the results plotted in Fig. 9.4: *MAD* needs more frame to adapt the network to a new environment, but given sequences long enough can successfully approximate full back propagation over time (*i.e.*, 0.20 EPE difference and 1.2 D1-all between the two adaptation modalities on Tab. 9.3) while granting nearly twice FPS. On long term (e.g., beyond 1500 frames on Fig. 9.4) running *MAD*, full adaptation or offline tuning on ground-truth grants equivalent performance.

9.3.3 *Different online adaptation strategies*

To assess the effectiveness of *MAD* for fast online adaptation we carried out additional tests on the whole KITTI RAW dataset [31] and compare

Model	**Adapt.**	**D1-all(%)**	**EPE**	**FPS**
DispNetC	No	9.09	1.58	15.85
DispNetC	Full	3.45	1.04	5.22
DispNetC-GT	No	4.40	1.21	15.85
MADNet	No	38.84	11.65	39.48
MADNet	Full	2.17	0.91	14.26
MADNet	*MAD*	3.37	1.11	25.43
MADNet-GT	No	2.67	0.89	39.48

Table 9.3: Results on the full KITTI raw dataset [31] (*Campus* $\rightarrow$ *City* $\rightarrow$ *Residential* $\rightarrow$ *Road*).

Adaptation Mode	**D1-all(%)**	**EPE**	**FPS**
No	38.84	11.65	39.48
Final-1	38.33	11.45	38.25
Final-7	31.89	6.55	29.82
Final-13	18.84	2.87	25.85
MAD-SEQ	3.62	1.15	25.74
MAD-RAND	3.56 ($\pm$0.05)	1.13 ($\pm$0.01)	25.77
MAD-FULL	**3.37** ($\pm$0.1)	**1.11** ($\pm$0.01)	25.43

Table 9.4: Results on the KITTI raw dataset [31] using *MADNet* trained on synthetic data and different fast adaptation strategies

performance obtainable deploying different fast adaptation strategies for *MADNet*. Results are reported on Tab. 9.4 together with those concerning a network that does not perform any adaptation.

First, we compare *MAD* keeping the weights of the initial portions of the network frozen and training only the final K layers (*Final-K*). Using the notations of Fig. 9.2, *Final-1* corresponds to training only the last prediction layer, *Final-7* to fine-tuning the last *Refinement* block, *Final-13* to training *Refinement* and D2. Then, since *MAD* consists in splitting the network in independent portion and choosing which one to train, we compare our full proposal (MAD-*FULL*) to: keeping the split in independent portions and choosing which one to train at each step either randomly (MAD-*RAND*) or using a round-robin schedule (MAD-*SEQ*). Since MAD-*FULL* and MAD-*RAND* feature non-deterministic sampling steps, we report their average performance obtained across 5 independent runs on the whole dataset with the corresponding standard deviations between brackets.

By comparing the *Final* entries with the ones featuring *MAD* we can clearly see how training only the final layers is not enough to successfully perform online adaptation. Even training as many as 13 last layers (at a comparable computational cost with our proposed *MAD*) we are at most able to halve the initial error rate, with performance still far from optimal. The three variants of *MAD* by training the whole network can successfully reduce the D1-all to $\frac{1}{10}$ of the original. Among the three options, our proposed layer selection heuristic provides the best overall performance even taking into account the slightly higher standard deviation caused by our sampling strategy. Moreover, the computational cost to pay to deploy our heuristic is negligible losing only 0.3 FPS compared to the other two options.

9.3.4 *Deployment on embedded platforms*

All the tests reported so far have been executed on a desktop PC equipped with a Nvidia 1080 Ti GPU. Unfortunately for many application like robotics or autonomous vehicles, it is unrealistic to rely on such high end and power-hungry hardware. One of the key benefits of *MADNet*, however, is its lightweight architecture conducive to easy deployment on lower-power embedded platforms. Thus, we evaluated *MADNet* on a Nvidia Jetson TX2 while processing stereo pairs at the full KITTI resolution and compared it to StereoNet [138] implemented using the same framework (*i.e.*, same level of optimization). We measured 0.26s for a single forward of *MADNet* versus 0.76-0.96s required by StereoNet, respectively with 1 or 3 refinement modules. Thus, *MADNet* runs faster and more accurately than existing fast architectures [138], making it an appealing alternative also for embedded applications.

10 LEARNING TO ADAPT FOR STEREO

In Chap. 9 we have shown how online adaptation can be deployed to adapt a deep stereo model to an unseen environment, in this chapter we are going to expand on the topic and propose a way of improving the effectiveness of the adaptation process. In detail, we propose to investigate the use of synthetic data to learn a model offline, which, when deployed, can quickly adapt to any unseen target domain online in an unsupervised manner, eliminating the need for expensive data collection.

We formulate this *learning-to-adapt* problem using a model agnostic meta-learning framework [90]. Specifically, we devise a meta-learning scheme for continuous adaptation. Our goal is to learn a model offline using synthetic data, which can continuously adapt to unseen real video sequences in an unsupervised manner at test time. This means our model is always in training mode and its parameters are automatically tuned to the current environment online without the need for supervision. Such an online adaptation scenario has been considered previously in the literature Chap. 9 and [156]. However, in this work, we explicitly *learn-to-adapt* which allows us to achieve superior performance.

Our meta-learning approach directly incorporates the online adaptation step into the learning objective, thus allowing us to obtain a base set of weights that are better suited for unsupervised online adaptation. However, since the adaptation is performed in an unsupervised manner (*e.g.*, based on re-projection loss [68, 93]), it is inherently noisy, causing an adverse effect on the overall algorithm. To alleviate this deficiency, we learn a confidence measure on the unsupervised loss and use the confidence weighted loss to update the network parameters. This effectively masks the noise in the adaptation step, preventing detrimental parameter updates. In our case, the confidence measures are predicted using a small CNN which is incorporated into the meta-learning framework, allowing the network to be trained end-to-end with no additional supervision.

In our experiments, we make use of a synthetic stereo dataset (Synthia [81]), a real-world dataset (KITTI-raw [122]), and generate a new synthetic dataset containing multiple sequences of varying weather and lighting conditions using the Carla simulator [89]. We evaluate our algorithm between two pairs of dataset domains: 1) Carla to Synthia; and 2) Carla or Synthia to KITTI-raw. In all experiments, our learning-to-adapt method consistently outperforms the previous unsupervised adaptation method (Chap. 9), validating our hypothesis that learning-to-adapt provides an effective framework for stereo.

10.1 RELATED WORKS

Meta-learning is a long-standing problem in machine learning [1, 2, 4] that tries to exploit structures in data to learn more effective learning rules or algorithms. Most of the recent development in meta-learning algorithms have been focused on the few shot classification task [84, 110, 118], with few exceptions like [90, 142] extending their models for simple function regression and reinforcement learning. In [90] the authors propose to constrain the learning rule for the model to be stochastic gradient descent and update the initial weight configuration of a network to make it more suited to learn new tasks. This simple formulation has been recently extended to address online adaptation in reinforcement learning using meta-learning to adapt to changing agents [133] or non-stationary and competitive environments [116]. Our work builds on [90] by modifying it to use structured regression, unsupervised loss functions and temporal consistency during the update process.

10.2 PROBLEM SETUP AND PRELIMINARIES

In this section, first we formalize online adaptation and discuss its advantages. We then briefly review a meta-learning algorithm which we will transpose into our continuous adaptation scenario.

10.2.1 *Online Adaptation for Stereo*

Let us denote two datasets of stereo video sequences: $\mathcal{D}_s$ (supervised), with available ground truth, and $\mathcal{D}_u$ (unsupervised), without ground truth. We would like to learn network parameters offline using $\mathcal{D}_s$, and use $\mathcal{D}_u$ as the target (or test) domain. However, in contrast to the standard evaluation

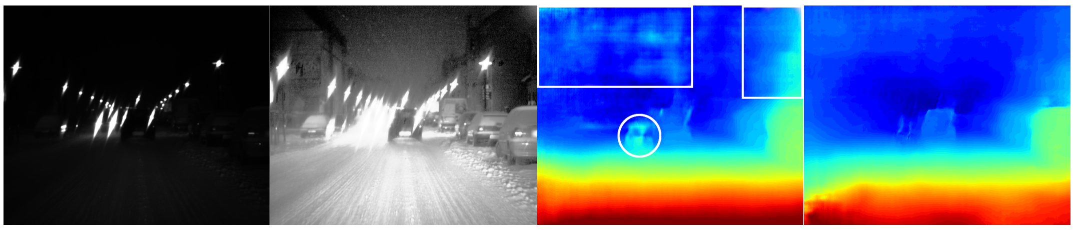

(a) Left frame (b) Equalized (c) KITTI tuned (d) Ours

Figure 10.1: We demonstrate the effectiveness of continuous adaptation on a challenging video sequence from [30]. (a) Left input frame, (b) histogram equalized frame for visualization purpose, (c) disparity map produced by a Dispnet-Corr1D [76] trained on annotated real data from KITTI, (d) disparity map produced by a Dispnet-Corr1D [76] trained on synthetic data using our learning-to-adapt framework and continuously adapted on this video sequence. As shown, the prediction of our method does not suffer from the same artifacts as (c) (highlighted in white), thus illustrating the advantage of continuous unsupervised adaptation.

setting and following the evaluation protocol of Chap. 9 and [156], the model is allowed to adapt to the target domain in an unsupervised manner.

Formally, let the parameters of the base model trained on $\mathcal{D}_s$ be θ. Given an unseen target video sequence $\mathcal{V} \in \mathcal{D}_u$, the adaptation is iteratively performed for each pair of frames in the sequence, using gradient descent on a predefined unsupervised loss function ($\mathcal{L}_u$). At iteration t, the online adaptation can be written as:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_\theta \mathcal{L}_u(\theta_{t-1}, i_t) \ , \tag{10.1}$$

where $\theta_0 = \theta$, $\alpha > 0$ is the learning rate and i_t denotes the stereo pair of t^{th} frame of the sequence $\mathcal{V}$. Note that the network parameters are *continuously* updated for the entire video in a sequential manner. This process is repeated for each video sequence starting from the learned parameters θ.

MOTIVATING EXAMPLE. To show that deep CNN based stereo networks are highly sensitive to domain-shift and that online adaptation is indeed necessary, we give a motivating example below. We select a video sequence from [30] as a test domain, where the environment is similar to that of KITTI but features extreme weather conditions (*e.g.*, night, snow, *etc.*). We compare the predicted disparities of a Dispnet-Corr1D network [76] for two training regimes. The first is fine-tuned on the KITTI training sets [27, 57], and the second is trained on synthetic data using our learning-to-adapt framework and performs unsupervised online adaptation for the target domain. The results are shown in Fig. 10.1. Here it is evident that fine tuning on real images is not sufficient to obtain reliable performance across all environments as evidenced by the major mistakes in (c) bordered in white. Indeed, (c) behaves worse than the network trained only on synthetic data and adapted online to the target domain in an unsupervised manner by our formulation (d).

10.2.2 *Model Agnostic Meta Learning*

Model Agnostic Meta Learning (MAML) [90] is a popular meta-learning algorithm designed for few-shot learning problems. The main idea is to replicate the procedure performed at test time throughout the training process. Specifically, since the authors focus on few-shot learning, they include the few-shot adaptation steps into the training procedure and measure the loss function on the adapted parameters. In short, the parameters of the model are trained so that they can quickly adapt to a given new task within a few gradient steps.

Assuming that a single gradient step descent is performed at each adaptation step. Let $\mathcal{T}$ be the set of tasks in the training set and for each $\tau \in \mathcal{T}$, the corresponding task-specific training and validation sets be $\mathcal{D}_\tau^{train}$ and $\mathcal{D}_\tau^{val}$, respectively. Now, the overall MAML objective can be written as:

$$\min_\theta \sum_{\tau \in \mathcal{T}} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_\tau^{train}), \mathcal{D}_\tau^{val}) \ , \tag{10.2}$$

where $\alpha > 0$ is the learning rate used for adaptation. This meta-objective function is optimized via a two-stage gradient descent algorithm. Specifically, at each optimization iteration, the inner-loop performs a gradient descent update for each task separately starting from a common base model θ (adaptation step). Then, the outer-loop performs an update on the common base model, where the gradient is the sum of task-specific gradients computed using the parameters updated in the inner loop. We refer the interested reader to the original paper for more detail [90].

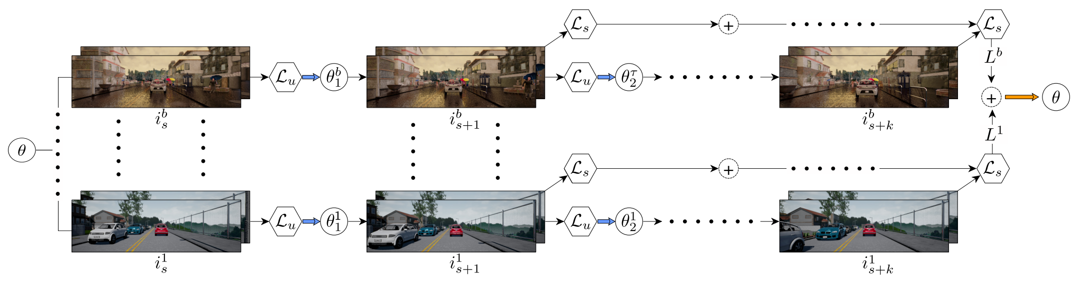


Figure 10.2: Schematic representation of one iteration of meta-learning of network parameters θ using a batch of b sequences and sampling k frames from each. We represent loss computation steps with hexagons and gradient descent steps with colored arrows. Blue and orange arrows denote adaptation steps and meta-learning steps, respectively. Starting from an initial set of parameters θ, the network is adapted independently on each sequence using loss function $\mathcal{L}_u$. Then, the adapted models are evaluated on the following frame of each sequence using loss function $\mathcal{L}_s$. Finally, the initial parameters θ are updated via a gradient descent step to minimize the sum of loss functions obtained by all evaluated models.

10.3 LEARNING TO ADAPT FOR STEREO

We first design a meta-learning algorithm for stereo adaptation by incorporating unsupervised continuous adaptation in the training paradigm. Then, we introduce a new mechanism to re-weight the pixel-wise errors estimated by the unsupervised loss function, making the adaptation more effective.

10.3.1 *Meta Learning for Stereo Adaptation*

Our hypothesis is that for any deep stereo network, before performing online adaptation to a target domain, it is beneficial to learn a base set of parameters (θ) that can be adapted effectively to unseen environments. We observe that our objective of learning-to-adapt to unseen video sequences is similar in spirit to that of MAML which tackled different tasks. Here, we perform the single task of dense disparity regression, but learn how to adapt to different environments and conditions.

We model an environment through a stereo video sequence $\mathcal{V}^\tau = [i_1^\tau, \dots, i_n^\tau]$[1]. At test time, the parameters are continuously adapted to the sequence $\mathcal{V}^\tau$ in an unsupervised manner according to Eq. 10.1. At training time, we mimic the same adaptation process on training sequences and evaluate the

1 For simplicity we assumed that all video sequences are of same length, but it is not a necessity.

Algorithm 3 Adaptation at training time for sequence $\mathcal{V}^\tau$

```
Require: $\theta, \mathcal{V}^\tau = [i_1^\tau, \dots, i_n^\tau]$
1: $\theta_0^\tau \leftarrow \theta$                                        ▷ Parameter initialization
2: for $t \leftarrow 1, \dots, n-1$ do
3:     $\theta_t^\tau \leftarrow \theta_{t-1}^\tau - \alpha \nabla_{\theta_{t-1}^\tau} \mathcal{L}_u \left(\theta_{t-1}^\tau, i_t\right)$        ▷ Adaptation
4:     $\mathcal{L}_s \left(\theta_t^\tau, i_{t+1}^\tau\right)$                          ▷ Supervised evaluation
```

performance of the model after each adaptation step on the subsequent frame. To measure the performance, we rely on a supervised loss function $\mathcal{L}_s$ (*e.g.*, L_1 or L_2 regression). This procedure for a single sequence $\mathcal{V}^\tau$ is given in Alg. 3.

During training we perform this adaptation on a supervised training set of video sequences $\mathcal{D}_s$ (*e.g.*, a set of rendered synthetic video sequences). The final objective of our problem is to maximise the measured performance across all frames and all sequences in $\mathcal{D}_s$. This can be written in a compact form as:

$$\min_\theta \sum_{\mathcal{V}^\tau \in \mathcal{D}_s} \sum_{t=1}^{n-1} \mathcal{L}_s(\theta_t^\tau, i_{t+1}^\tau) \,, \tag{10.3}$$

where θ_t^τ is obtained sequentially through updates as detailed in Alg. 3.

Note that this formula extends Eq. 10.2 (MAML) to the continuous and unsupervised adaptation scenario. Contrary to Eq. 10.2, we use two different loss functions: 1) an unsupervised loss ($\mathcal{L}_u$) to adapt a model to a video sequence; and 2) a supervised loss ($\mathcal{L}_s$) for the optimization of the set of parameters θ. We make this distinction to mimic the test time behaviour. Specifically, $\mathcal{L}_u$ (*i.e.*, some form of unsupervised loss function) will be used at test time, while $\mathcal{L}_s$ can use all the available annotations of the training set for optimization. Our intuition is that by using two different loss functions, θ can be optimized such that it is better suited to be adapted without supervision (*i.e.*, by $\mathcal{L}_u$), while the performance is measured with respect to the ground truth (*i.e.*, by $\mathcal{L}_s$).

Note that optimizing Eq. 10.3 on complete video sequences would be infeasible for long video sequences as the memory requirement grows linearly with n. To alleviate this, we approximate it by optimizing over batches of sequences of k randomly sampled consecutive frames. Our meta-learning algorithm is detailed in Alg. 4. After sampling a batch of sequences

Algorithm 4 Learning to Adapt for Stereo

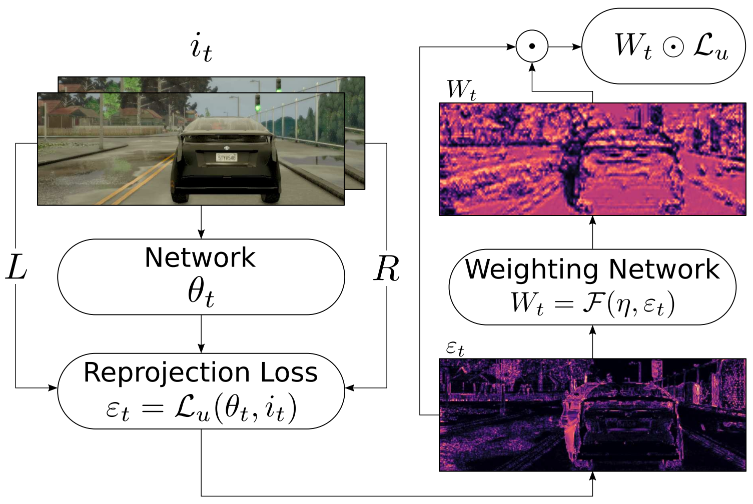
```
Require: Training set $\mathcal{D}_s$, and hyper-parameters $\alpha, \beta, k, b$
 1: Initialize $\theta$
 2: while not done do
 3:     $\mathcal{D}^b \sim \mathcal{D}_s$                                  ▷ Sample a batch of sequences
 4:     for all $\mathcal{V}^\tau \in \mathcal{D}^b$ do
 5:         $\theta^\tau \leftarrow \theta$                                  ▷ Initialize model
 6:         $L^\tau \leftarrow 0$                                            ▷ Initialize accumulator
 7:         $[i_s, \dots, i_{s+k}] \sim \mathcal{V}^\tau$                    ▷ Sample k frames
 8:         for $t \leftarrow s, \dots, s+k-1$ do
 9:             $\theta^\tau \leftarrow \theta^\tau - \alpha \nabla_{\theta^\tau} \mathcal{L}_u(\theta^\tau, i_t)$   ▷ Adaptation
10:             $L^\tau \leftarrow L^\tau + \mathcal{L}_s(\theta^\tau, i_{t+1})$                                  ▷ Evaluation
11:     $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{V}^\tau \in \mathcal{D}^b} L^\tau$           ▷ Optimization
```

(line 3) and k random frames from each sequence (line 7), we perform unsupervised adaptation on the current frame (line 9) and measure the effectiveness of this update on the following frame (line 10). This process is repeated for k frames. Finally, we optimize the initial model parameters θ to minimize the loss computed across all the sequences and all the frames (line 11). Here, α and β are the two learning rates used for online adaptation and for meta training, respectively. In Fig. 10.2, we illustrate one optimization iteration of the network parameters θ using a batch of b sequences and k frames from each.

By optimizing Eq. 10.3 we are able to learn a base parameter configuration θ suited for adaptation. However, the use of an imperfect unsupervised loss function ($\mathcal{L}_u$) for adaptation introduces mistakes in the optimization process that may have an adverse effect on the overall algorithm. To alleviate this issue, we introduce a mechanism to learn to recognize the noise (or mistakes) in the unsupervised loss estimation which can then be masked effectively.

10.3.2 *Confidence Weighted Adaptation*

Unsupervised loss functions for dense disparity estimation often compute some form of pixel-wise error map and minimize over the average mistake. Unfortunately, this process is not perfect and usually introduces errors in the optimization process. This may result in sub-optimal performance when compared to the use of supervised loss functions. For example, the left-right re-projection loss proposed in [68] is well-known to produce mistakes



i_t
L
Network
θ_t
R
Reprojection Loss
$\varepsilon_t = \mathcal{L}_u(\theta_t, i_t)$
$W_t \odot \mathcal{L}_u$
W_t
Weighting Network
$W_t = \mathcal{F}(\eta, \varepsilon_t)$
ε_t

Figure 10.3: Schematic representation of our weighted adaptation for a single stereo frame using an unsupervised re-projection based loss function $\mathcal{L}_u$ (bright colors indicate higher values). The system takes a stereo-pair (i_t) and computes a disparity map as well as a re-projection loss (ε_t). This loss is then weighted according to W_t effectively masking the mistakes.

in occluded areas and reflective surfaces. These mistakes are not due to bad predictions by the disparity estimation model, but instead are due to differences between the left and right frames. Ideally, we would like to have a confidence function to detect erroneous estimations of this loss such that they can be effectively masked. However, training such a confidence function might be difficult since there is no easy procedure to produce ground-truth annotations for this task. We propose to avoid explicit supervised training, and instead, automatically learn to detect noise in the loss estimations by incorporating this new objective into our meta-learning formulation.

In particular, we propose to learn a small CNN that takes a pixel-wise error map estimated by $\mathcal{L}_u$ as an input and produces a tensor W as an output, which has the same shape as the input and its elements are between 0 and 1. This output can be interpreted as a per pixel confidence on the reliability of the loss estimation with 1 corresponding to high reliability. We can now mask out potentially erroneous estimations by multiplying the loss values by their corresponding confidence values. The result is a cleaner measurement of the loss function that would reduce detrimental weight updates due to incorrect loss values. The idea of masking or weighting the contribution of single examples in the presence of noise or class imbalance in the labels has been previously studied for supervised classification in [137, 146]. In our case, we transpose the similar idea to pixel-wise loss, estimated for a dense regression task and directly predict a dense confidence map.

Let $W = \mathcal{F}(\eta, \varepsilon)$ be the mask produced by the re-weighting network parametrized by η and $\varepsilon = \mathcal{L}_u(\theta, i)$ be the estimated pixel-wise error map computed on the prediction of the disparity model with parameter θ on stereo frame i. We normalize the elements of W by dividing each one of them by the number of elements in W. Now, by modifying Eq. 10.1, the final weighted adaptation formula can be written as:

$$\begin{aligned} \widetilde{\theta}_t &\leftarrow \widetilde{\theta}_{t-1} - \alpha \nabla_\theta \left(W_t \odot \mathcal{L}_u(\widetilde{\theta}_{t-1}, i_t) \right) , \\ W_t &= \mathcal{F}(\eta, \mathcal{L}_u(\widetilde{\theta}_{t-1}, i_t)) , \end{aligned} \tag{10.4}$$

where $\widetilde{\theta}_0 = \theta$ and $\odot$ indicating the element-wise product between the matrices. We denote $\widetilde{\theta}$ as the network parameters which are updated according to the unsupervised loss weighted by W.

In Fig. 10.3, we show a schematic representation of our proposed weighted adaptation computed from a single stereo input frame i_t. On the bottom right corner we give a visualization of the error map produced by an unsupervised re-projection loss $\mathcal{L}_u$, while the top right corner shows a possible confidence mask W_t. In this example the weighting network is masking errors due to occlusions (*e.g.*, on the left side of the car) and due to reflections (*e.g.*, the puddles on the road).

Since supervision is not available for W, we indirectly train η by incorporating Eq. 10.4 inside the meta-learning objective described in Eq. 10.3. The final objective of our complete system becomes:

$$\min_{\theta, \eta} \sum_{\mathcal{V}_\tau \in \mathcal{D}_s} \sum_{t=1}^{n-1} \mathcal{L}_s(\widetilde{\theta}_t^\tau, i_t^\tau) . \tag{10.5}$$

Here, $\widetilde{\theta}_t^\tau$ are the parameter of the model updated according to the weighted unsupervised loss function on sequence $\mathcal{V}^\tau$. As such it depends on η and θ at the same time. The whole network can be trained end-to-end with the only supervision coming from the depth annotations used to compute $\mathcal{L}_s$. Both θ and η are tuned to maximize the network performances after few steps of optimization as measured by $\mathcal{L}_s$. By optimizing a single objective function we are able to learn the parameter (η) of the weighting network, and a set of base weights for the disparity estimation model (θ) which allow for fast adaptation.

10.4 Experiments

This section presents an evaluation of the quality of our proposed adaptation method. Firstly, we lay out our evaluation setup in Sec. 10.4.1. Secondly, in Sec. 10.4.2, we provide qualitative and quantitative results for two pairs of domains: 1) synthetic to real (*i.e.*, training on synthetic data and testing on real data from KITTI); and 2) synthetic to synthetic (*i.e.*, training on one synthetic dataset and testing on a different synthetic domain). Finally, in Sec. 10.4.3 we report qualitative results illustrating our confidence weighted loss.

10.4.1 *Experimental Setup*

Datasets. In our experimental evaluation we simulate realistic test conditions, in which no data from the target domain is available. We therefore use training and testing data sampled from two completely disjoint datasets. For the real dataset, we use the 71 different sequences of the KITTI-raw dataset [31] (denoted as KITTI) accounting for ~43K images with sparse depth annotations provided by [122].

For the synthetic dataset, we have used the FlyingThings3D dataset [76] (shortened F3D) to perform an initial training of the networks from random initialization. Then, we use the Synthia dataset [81] as a synthetic dataset containing scenarios similar to KITTI. The dataset is composed of 50 different video sequences rendered in different seasons and weather conditions for a total of ~45K images. For this dataset we scaled the image to half resolution to bring the disparity into the same range as KITTI.

Finally, using the Carla simulator[89], we have rendered a new dataset (referenced as Carla) composed of 25 different video sequences, each being a thousand frames long, with accurate ground truth data for each frame. Each video sequence is rendered in 15 different weather conditions to add variance to the dataset. Resulting in a total of 375K frames. During the rendering we set up the virtual cameras to match the geometry of the real KITTI dataset (*i.e.*, same baseline, field of view and similar image resolution).

Network architectures. For the experiments we have selected the Dispnet-Corr1D [76] architecture (shortened to Dispnet). For all the evaluation tests, we pretrain the networks on F3D to obtain a set of weights that will be used as an initialization across all the other tests. We implement the confidence function introduced in Sec. 10.3.2 as a small three layer fully convolutional CNN with batch normalization. The network takes the re-projection error scaled to quarter resolution as an input and produces an output at the same resolution. The prediction is then scaled to full resolution using bilinear upsampling.

EVALUATION PROTOCOL. After an initial offline training, we perform online adaptation and evaluate models on sequences of stereo frames. To test independent adaptations for each sequence, we reset the disparity network to its trained weight configuration at the beginning of each test sequence. Then, for each frame, first, we measure the performance of the current model and then we adapt it by a single step of back-propagation and weight update according to Eq. 10.1 before moving to the next frame. We do not measure the performance on frames already used for adaptation.

METRICS. We measure performance according to both average End Point Error (EPE) and percentage of pixels with disparity error larger than 3 (D1-all). Firstly, we measure both metrics independently for each frame to plot performance as a function of the number of frames processed for adaptation. Secondly, we can average both metrics over each sequence and finally average over all the dataset.

OFFLINE TRAINING. After the initial pretraining on F3D we fine tune the networks on a training set with our learning-to-adapt framework, we use $k = 3$ consecutive frames for each sample and set the learning rates $\alpha = 0.00001$ and $\beta = 0.0001$

ONLINE ADAPTATION. We use the left-right re-projected unsupervised loss [93] for the adaptation. Optimization is performed with SGD with momentum, where the momentum value is set to 0.9 and a learning rate set to 0.0001.

10.4.2 *Results*

We evaluate our learning-to-learn method between pairs of datasets, one for training, and one for evaluation. We consider two scenarios: 1) synthetic

		Method	Training set	D1-all (%)	EPE	ΔD1	ΔEPE
	(a)	**SL**	-	9.43	1.62	-	-
	(b)	**SL+Ad**	-	7.81	1.44	-1.62	-0.18
	(c)	**SL**	Carla	7.46	1.48	-	-
	(d)	**SL+Ad**	Carla	5.26	1.20	-2.20	-0.28
	(e)	**SL**	Synthia	8.55	1.51	-	-
	(f)	**SL+Ad**	Synthia	5.33	1.19	-3.22	-0.32
Ours	(g)	**L2A**	Carla	8.41	1.51	-	-
Ours	(h)	**L2A+WAd**	Carla	**4.49**	**1.12**	**-3.92**	**-0.39**
Ours	(i)	**L2A**	Synthia	8.22	1.50	-	-
Ours	(j)	**L2A+WAd**	Synthia	4.65	1.14	-3.57	-0.36
	(k)	**SL** (ideal)	KITTI	4.26	1.12	-	-

Table 10.1: Performance on KITTI for the Dispnet network trained according to different methods after initialization from F3D. It can clearly be seen that online adaptation (**+Ad**/**+WAd**) provides a significant improvement compared to when it is left out. The best results are obtained when the training is achieved using the **L2A+WAd** framework. Line (k) indicates an upper bound on how well Dispnet can perform when fine tuned on a subset of samples from the target domain. The last two columns indicate the performance improvement with adaptation, and as it is evident in the table, our **L2A+WAd** method obtains the largest increase in performance with adaptation.

to real and 2) synthetic to synthetic. We compare the results of our learning-to-adapt framework (**L2A**), and the method trained using a supervised L_1 regression loss (**SL**). Methods performing unsupervised online adaptation at test time are indicated by appending **+Ad** to the training method, and confidence weighted adaptation by **+WAd**.

Synthetic to Real

The most interesting scenario is the one where training on a synthetic domain is followed by testing on a real-life domain. Specifically, we train on Synthia or Carla and then evaluate on the KITTI dataset.

The results for the Dispnet architecture are provided in Tab. 10.1. Lines (a) to (f) report the performance when the network weights are obtained in a standard way (using a supervised L_1 loss function). As expected, the network performs poorly when tested on a different domain with respect to

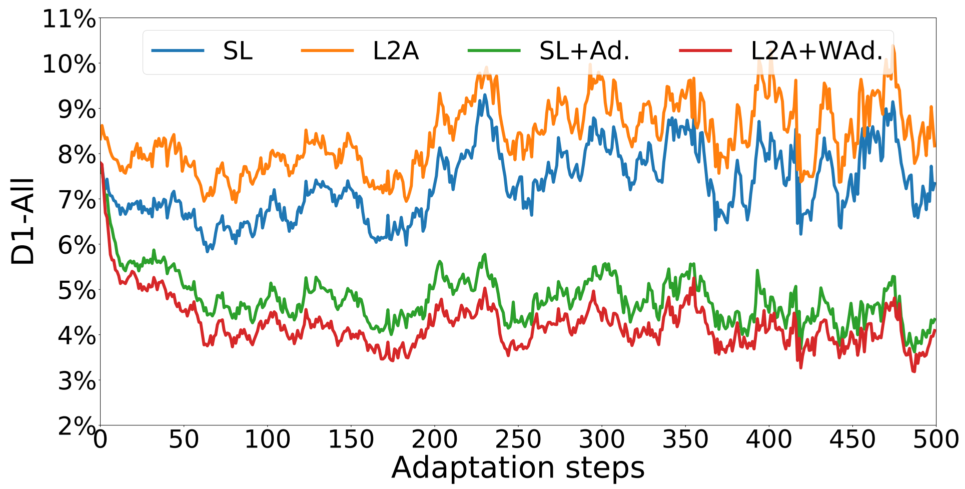


Figure 10.4: Average D1-all error with respect to the number of adaptation steps performed on the KITTI database by a Dispnet network trained according to supervised learning (**SL**) or our learning to adapt framework (**L2A**).

the one it was trained on - lines (a, c, e). The use of adaptation for this setup provides a significant improvement - lines (b, d, f) - further motivating the need to adapt to a new domain.

The two rows (h) and (j) report results obtained by learning to adapt on Carla or Synthia using the **L2A+WAd** framework. Our proposed framework clearly outperforms the baseline methods for both training datasets. Comparing lines (h) and (d) clearly shows that our training process is able to learn a model which is better suited for continuous adaptation. The same conclusions hold even for the results with Synthia (lines (j) and (f)). In the last two columns we can observe the relative improvement provided by adaptation for each method. In these results, it is evident that our **L2A+WAd** framework provides the largest increase in accuracy when performing adaptation. Line (k) provides the performance of Dispnet obtained by fine tuning the base model using a supervised L_1 regression loss on samples from the target domains (*i.e.*, KITTI2012 and KITTI2015 training sets). Our **L2A+WAd** framework obtains competitive results, even when compared to fine-tuning on a small dataset from the target domain with accurate ground truths.

ADAPTATION PERFORMANCE OVER TIME: To further highlight the difference of behaviour between models trained to regress and those trained to adapt, we plot the average D1-all error achieved by Dispnet on KITTI as a function of the number of adaptation steps in Fig. 10.4. The vertical axis

	Method	Training Set	Ad. Unsupervised		Ad. Supervised	
			D1-all (%)	EPE	D1-all (%)	EPE
(a)	**SL+Ad**	-	26.56	3.96	15.60	2.24
(b)	**SL+Ad**	Carla	25.07	3.62	13.89	1.97
(c)	**L2A+Ad**	Carla	22.69	3.08	**12.01**	**1.80**
(d)	**L2A+WAd**	Carla	**21.07**	**2.90**	✗	✗

Table 10.2: Comparison of the different training methods when evaluated on sequences from Synthia. It can be seen that the best performing training method is **L2A+WAd**. We also provide results for when we use a L_1 supervised adaptation loss. Best results in bold.

represents the average D1-all error of the k^{th} frame in all of the sequences in KITTI. Comparing the methods with and without online adaptation, it is clear that in both cases, adaptation drastically improves the performance. The comparison between **SL+Ad** (green line) and **L2A+WAd** (red line) shows how quickly our method adapts to the given video sequence. The poor results of **L2A** can easily be explained since our formulation never explicitly optimizes the base model for regression. Instead it optimizes the network to quickly learn-to-adapt, therefore the base model results can be sub-optimal, providing the performance can be improved in a few adaptation steps.

Synthetic to Synthetic

Here, we perform a more controlled synthetic-to-synthetic evaluation where we can measure the difference in performance more explicitly thanks to the availability of dense and accurate ground truth labels. The aim of the following series of tests will be to quantify the performance of the two key aspects of the learning-to-adapt framework, namely, learning to adapt through meta-learning and learning to weight noisy loss estimation. To further prove the generality of our learning to adapt formulation, we also provide results when the networks are trained to perform online adaptation using a supervised L_1 loss (*i.e.*, $\mathcal{L}_u \equiv \mathcal{L}_s$).

For these tests, we again use Dispnet trained on Carla but tested on all the sequences of the full Synthia dataset. Specifically, to show that we can adapt using different loss functions, we train for both unsupervised and supervised adaptation[2], and evaluate the performance of the following training scenarios: (a) Using the initial model trained using F3D; (b) Training on Carla using a supervised L_1 loss; (c) Using the learning-to-adapt framework **without** confidence weighted loss; (d) Using the learning-to-adapt framework **with** confidence weighted loss.

We report the results in Tab. 10.2, where it can be seen that explicitly training Dispnet to adapt using our learning-to-learn formulation (c), allows the network to exploit the online adaptation and greatly improve the performance both in the unsupervised and supervised adaptation setups. Finally, it can also be seen that weighting the unsupervised loss values results in superior performance (d). For this test set up, the results clearly demonstrate how our formulation is able to learn a weight configuration that is more inclined to be adapted to a new environment.

10.4.3 *Confidence Weighted Loss Function*

In Fig. 10.5, we show a visualization of the confidence masks and weighted errors optimized by our confidence guided adaptation loss described in Sec. 10.3.2. The predicted confidence maps effectively mask out occluded regions in the image while keeping the useful error signals in the rest of the image (low confidence areas are encoded as dark pixels). Errors on occluded regions, *e.g.*, to the left of the traffic sign in the left column or to the left of the car in the right column, are effectively masked out, producing a cleaner error estimation that will improve adaptation performances. We wish to highlight that the confidence network has been trained without any direct supervision and only on Carla, nevertheless, it seems to be able to generalize well to KITTI. We believe this ability to generalize is mainly due to the avoidance of working directly with RGB inputs, which inevitably change drastically between datasets. Instead, the confidence network relies on the estimated re-projection error, which is more consistent across different environments.

2 In online adaptation we use the L_1 loss between the predicted disparity and the ground truth annotations for each stereo pair.

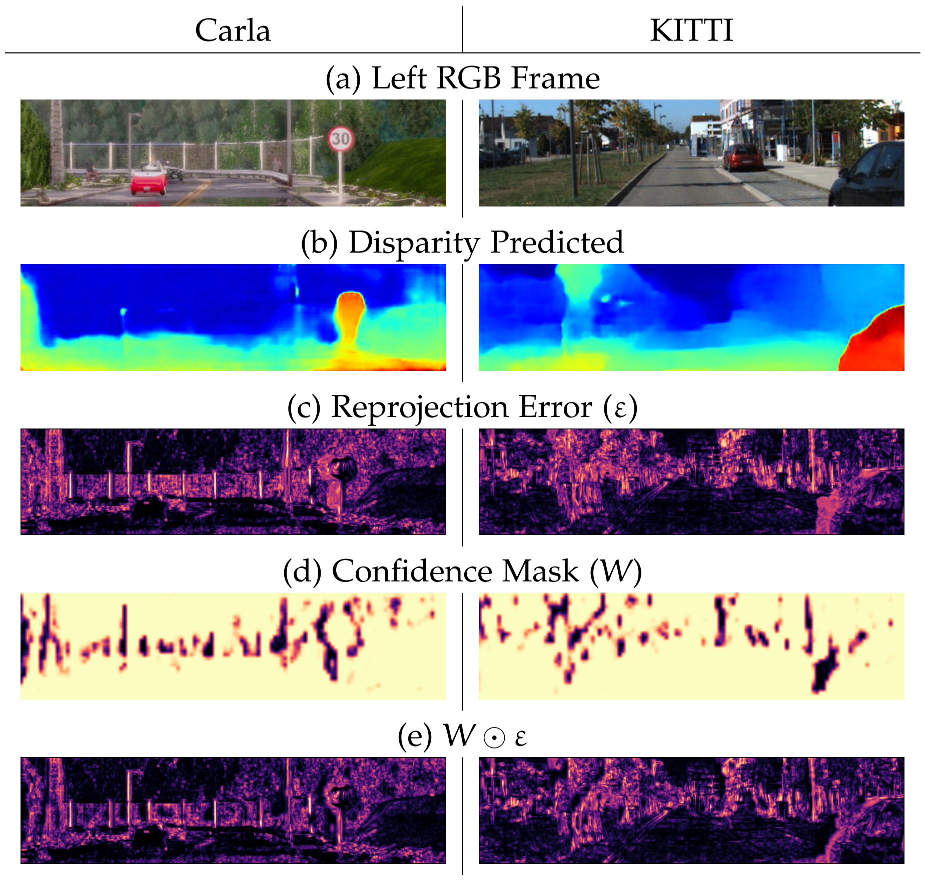


Figure 10.5: Visualization of the errors optimized to achieve unsupervised adaptation with reprojection based loss function and using our weighting function. Brighter colours encode higher values.

11 CONCLUSIONS

In this part, we have proposed different procedures to adapt depth estimation networks to unseen environments without the need for ground truth data. In Chap. 8 we have shown how classic stereo algorithms can be used to produce noisy label estimation and how confidence measure can be deployed to identify such mistakes. By combining the two source of information we were able to design a novel confidence guided loss able to outperform all the other self-supervised proposal in our extensive experimentation. While the focus of Chap. 8 was more on the achievable precision disregarding the training time and amount of data needed to achieve it, in Chap. 9 we tackle the adaptation problem from an *online and continuous learning* perspective and focus on the best trade-off between speed and accuracy. To do so we have introduced a new fast depth estimation architecture for stereo cameras (*MADNet*) and a novel way to adapt it online to unseen environments while keeping a high throughput (*MAD*). In terms of absolute performance, our online proposal is still worse than the offline procedure proposed in Chap. 8, but we believe this to be a key contribution for real-world applications where the deployment environment is unknown beforehand and dynamically changing (*e.g.*, weather conditions in an autonomous driving scenario). To furthermore boost the performance of our online formulation in Chap. 10 we have shown how it is possible to train a depth prediction model that is more amenable to be adapted effectively to new and unseen environments by changing the training paradigm and deploying meta-learning techniques. We believe this to be an exciting new direction to explore as for many applications having a reactive system able to compensate for sudden unexpected changes is key to success. We have mainly reported quantitative results only on an autonomous driving scenario due to the lack of properly annotated dataset in different environments. However, the qualitative results shown on our target supermarket environment seems to imply the effectiveness of our solutions even in this completely different setup.

All the test proposed in this work concern *adaptation* of a pre-trained deep neural network to new environments, however, we believe that both our unsupervised loss formulations could also be effective in training from scratch a deep neural network without any kind of supervision, we leave this to future research. Moreover, in Chap. 8 we have shown how combining naively noisy labels obtained from two very different stereo algorithms does not result in an improvement in performance. Recent works like [130], however, have shown how combining different disparity estimations taking into account their confidence maps can result in reliable disparity predictions. We plan to include in our work a similar procedure to obtain more reliable noisy labels that might be used as pseudo ground truth for off-line adaptation. We wish also to investigate a fast way to implement the loss function we introduced in Chap. 8 as to deploy it inside the online adaptation pipeline presented in Chap. 9 since in our experiments our confidence guided loss always outperformed the competing reprojection based methods. Regarding our online formulation, we plan to test and eventually extend *MAD* to be applicable to any end-to-end stereo system. We would also like to investigate more effective methods to select the portion of the network to be updated online at each step. In particular, we might deploy a small network trained by reinforcement learning in order to contextually choose which training action to perform based on the current frame and/or current network state. Finally, even with our lightweight *MADNet*, the computational cost to deploy the system on embedded low power device is still too high for most practical application as shown in Sec. 9.3.4, for this reason, we would like to focus some research efforts on ways to make our network even faster and lighter. Inspirations for this extension can come from works regarding the deployment of neural networks on mobile low power device such as [95, 154] where different paradigms for the implementation of efficient convolutional layers are deployed. Finally we believe that the use of more complex training paradigm, for example the meta-learning procedure described in Chap. 10, represent a still quite fresh research field that has not yet been properly addressed, for this reason in the future we plan to better explore the deployment of the learning to learn paradigm in the context of adaptation to unseen environments.

Part III

FINAL REMARKS

12 CONCLUSIONS

The works presented in this thesis have concerned how to effectively deploy state of the art computer vision techniques to ease and automate management tasks in retail shops or supermarket. We have mainly discussed two macro problems: the detection and recognition of products on store shelves (discussed in Part i) and the estimation of reliable 3D information on unknown environments (discussed in Part ii). For both sections, we have explained some of the peculiar shortcomings emerging from naively applying state of the art computer vision techniques and how to effectively overcome them.

For example, referring to Part i, we have shown how classic local feature based object detection pipeline provide satisfying results when dealing whit rigid boxed products but fails in presence of deformable packages and rotated or skewed products (see Chap. 3). We then moved on to more modern deep learning based object detectors and verified that they can indeed be trained to effectively detect and recognize products with remarkable accuracy. Such good performance comes at the cost of requiring huge annotated dataset at training time to tune the network behavior. Since in a common retail environment is unfeasible to think of keeping an up to date annotated dataset concerning all possible types of products exposed on shelves and constantly retraining an inference engine to keep up with changes, we propose in Chap. 4 to split the detection and recognition phases into two separate steps solved by two different algorithms. First, we carry out a class agnostic detection phase where the task is only to recognize individual instances of products on store shelves, then we perform recognition through similarity search in a learned embedding space for product images. We proved that our solution is able to outperform other competing methods in the commonly used public Grocery product dataset in terms of accuracy while keeping runtime efficiency. Finally in Chap. 5 we examined in depth the task of performing product recognition through similarity search and propose a novel metric learning network to directly address one of the main problems in the retail scenario: the huge domain gap between images acquired in-store and those available for training.

Moving on to Part ii, we have shown how the modern state of the art algorithms for depth estimation suffers from huge performance drops when exposed to environments different from those used at training time. To mitigate the negative effects of this domain shift we have introduced in Chap. 8 a novel way to adapt depth estimation networks to new environments without the need of annotated data that are usually too costly to acquire for dense 3D estimation. We have proven the effectiveness of our proposal both in indoor and outdoor environments, with stereo and mono depth estimation networks, and only assuming the availability of stereo images from the target domain at training time. In Chap. 9 we tackle the same problem but propose to solve the domain shift online performing continuous domain adaptation to new environments as soon as new frames are sensed. In our proposal we keep the deep network always in training mode, thus slowing it down considerably, for this reason we have introduced a novel lightweight depth estimation network (*MADNet*) and a modular update process (*MAD*) to improve network efficiency. Finally in Chap. 10 we have explored the idea of designing the training process of a network from the beginning to make it more suitable to be adapted quickly to new environments, we achieve so by deploying training techniques taken from the meta-learning literature.

In this thesis, we have analyzed how to apply state of the art computer vision techniques in real environments to solve two very different tasks. Even if the two applications, at first sight, do not seems closely related, most of the problems that we had to face to solve them are shared. For example, since almost all our proposed solutions are based on deep learning techniques, one of the main limitations that we had to overcome was how to effectively train them without relying on a huge annotated dataset that are often too costly to produce for many real applications. We tackled this problem in the product recognition scenario by deploying GANs to easily perform data augmentation and relying on embedding rather than classifier to perform product recognition, while in the dense depth estimation scenario we designed an ad hoc unsupervised loss function. Another common obstacle that we had to overcome across the two tasks was how to close the domain gap between train and test data. To address this problem in the product scenario we deployed GANs to solve the domain shift at training time by transforming images from one domain to the other, while for the dense depth estimation we have designed systems and loss functions to allow efficient continuous training and adaptation of deep learning based methods. We believe these problems to be more general than the single examples presented in this thesis and we are excited to test some of our proposed solutions in contexts different from those considered so far. For example, our online adaptation formulation could be extended to different tasks like semantic segmentation or optical flow as long as it is possible to define some kind of unsupervised loss function. At the same time, we have already shown some preliminary results obtained by applying our metric learning technique (Chap. 5) to scenario different than the retail ones. Finally, the preliminary results in Chap. 10 seems to suggest that it is possible to design more efficient learning algorithms able to converge to good solutions using only a few examples. Besides applications to dense depth regression, we would like to apply the same concept to object detection algorithm so as to obtain a flexible system able to learn how to recognize new categories based only on few annotated samples.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] Jürgen Schmidhuber. 'Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook'. PhD thesis. Technische Universität München, 1987 (cit. on p. 118).

[2] Devang K Naik and RJ Mammone. 'Meta-neural networks that learn by learning'. In: *Neural Networks, 1992. IJCNN., International Joint Conference on*. Vol. 1. IEEE. 1992, pp. 437–442 (cit. on p. 118).

[3] Ramin Zabih and John Woodfill. 'Non-parametric Local Transforms for Computing Visual Correspondence'. In: *Proceedings of the Third European Conference on Computer Vision (Vol. II)*. ECCV '94. Stockholm, Sweden: Springer-Verlag New York, Inc., 1994, pp. 151–158 (cit. on pp. 81, 82, 86, 89).

[4] Sebastian Thrun and Lorien Pratt. 'Learning to learn: Introduction and overview'. In: *Learning to learn*. Springer, 1998, pp. 3–17 (cit. on p. 118).

[5] Thomas W Gruen, Daniel Corsten and Sundrr Bharadwaj. 'Retail out of stocks: A worldwide examination of causes, rates, and consumer responses'. In: *Grocery Manufacturers of America, Washington, DC* (2002) (cit. on p. 11).

[6] Daniel Scharstein and Richard Szeliski. 'A taxonomy and evaluation of dense two-frame stereo correspondence algorithms'. In: *International journal of computer vision* 47.1-3 (2002), pp. 7–42 (cit. on p. 77).

[7] David G Lowe. 'Distinctive image features from scale-invariant keypoints'. In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. on pp. 16, 18, 25, 34).

[8] Zhou Wang, Alan C Bovik, Hamid R Sheikh and Eero P Simoncelli. 'Image quality assessment: from error visibility to structural similarity'. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612 (cit. on pp. 52, 54, 86).

[9] Heiko Hirschmuller. 'Accurate and efficient stereo processing by semi-global matching and mutual information'. In: *Computer Vision and Pattern Recognition*. Vol. 2. IEEE. 2005, pp. 807–814 (cit. on pp. 77, 81, 82, 86).

[10] Ingo Wegener. *Complexity theory: exploring the limits of efficient algorithms*. Springer Science & Business Media, 2005 (cit. on p. 19).

[11] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. 'Surf: Speeded up robust features'. In: *European conference on computer vision*. Springer. 2006, pp. 404–417 (cit. on pp. 15, 25).

[12] Raia Hadsell, Sumit Chopra and Yann LeCun. 'Dimensionality reduction by learning an invariant mapping'. In: *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE. 2006, pp. 1735–1742 (cit. on pp. 61, 63).

[13] Michele Merler, Carolina Galleguillos and Serge Belongie. 'Recognizing groceries in situ using in vitro training data'. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8 (cit. on pp. 11, 12).

[14] Motilal Agrawal, Kurt Konolige and Morten Rufus Blas. 'Censure: Center surround extremas for realtime feature detection and matching'. In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 102–115 (cit. on p. 25).

[15] Koen EA van de Sande, Theo Gevers and Cees GM Snoek. 'Color descriptors for object category recognition'. In: *Conference on Colour in Graphics, Imaging, and Vision*. Vol. 2008. 1. Society for Imaging Science and Technology. 2008, pp. 378–381 (cit. on p. 26).

[16] Ashutosh Saxena, Min Sun and Andrew Y Ng. 'Make3d: Learning 3d scene structure from a single still image'. In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2009), pp. 824–840 (cit. on p. 78).

[17] Florent Perronnin, Yan Liu, Jorge Sánchez and Hervé Poirier. 'Large-scale image retrieval with compressed fisher vectors'. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 3384–3391 (cit. on p. 29).

[18] Kate Saenko, Brian Kulis, Mario Fritz and Trevor Darrell. 'Adapting visual category models to new domains'. In: *European conference on computer vision*. Springer. 2010, pp. 213–226 (cit. on p. 63).

[19] Engin Tola, Vincent Lepetit and Pascal Fua. 'Daisy: An efficient dense descriptor applied to wide-baseline stereo'. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.5 (2010), pp. 815–830 (cit. on p. 25).

[20] Julian R Ullmann. 'Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism'. In: *Journal of Experimental Algorithmics (JEA)* 15 (2010), pp. 1–6 (cit. on p. 19).

[21] Tess Winlock, Eric Christiansen and Serge Belongie. 'Toward real-time grocery detection for the visually impaired'. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE. 2010, pp. 49–56 (cit. on p. 13).

[22] Pablo F Alcantarilla and TrueVision Solutions. 'Fast explicit diffusion for accelerated features in nonlinear scale spaces'. In: *IEEE Trans. Patt. Anal. Mach. Intell* 34.7 (2011), pp. 1281–1298 (cit. on p. 25).

[23] Stefan Leutenegger, Margarita Chli and Roland Y Siegwart. 'BRISK: Binary robust invariant scalable keypoints'. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2548–2555 (cit. on pp. 25, 41).

[24] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. 'ORB: an efficient alternative to SIFT or SURF'. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2564–2571 (cit. on p. 25).

[25] Alexandre Alahi, Raphael Ortiz and Pierre Vandergheynst. 'Freak: Fast retina keypoint'. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. Ieee. 2012, pp. 510–517 (cit. on p. 25).

[26] Pablo Fernández Alcantarilla, Adrien Bartoli and Andrew J Davison. 'KAZE features'. In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 214–227 (cit. on p. 25).

[27] Andreas Geiger, Philip Lenz and Raquel Urtasun. 'Are we ready for autonomous driving? the kitti vision benchmark suite'. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3354–3361 (cit. on pp. 75–77, 79, 83, 87, 91, 93, 112, 119).

[28] Xiaoyan Hu and Philippos Mordohai. 'A quantitative evaluation of confidence measures for stereo vision'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2012), pp. 2121–2133 (cit. on p. 81).

[29] Kunal Mankodiya, Rajeev Gandhi and Priya Narasimhan. 'Challenges and opportunities for embedded computing in retail environments'. In: *Sensor Systems and Software*. Springer, 2012, pp. 121–136 (cit. on p. 15).

[30] Stephan Meister, Bernd Jähne and Daniel Kondermann. 'Outdoor stereo camera system for the generation of real-world benchmark data sets'. In: *Optical Engineering* 51.2 (2012), p. 021107 (cit. on pp. 79, 80, 119).

[31] Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun. 'Vision meets Robotics: The KITTI Dataset'. In: *International Journal of Robotics Research (IJRR)* (2013) (cit. on pp. 76, 83, 86, 90, 95–97, 103, 104, 110, 112, 114, 115, 126).

[32] Boqing Gong, Kristen Grauman and Fei Sha. 'Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation'. In: *International Conference on Machine Learning*. 2013, pp. 222–230 (cit. on p. 63).

[33] R. Haeusler, R. Nair and D. Kondermann. 'Ensemble Learning for Confidence Measures in Stereo Vision'. In: *CVPR. Proceedings*. 1. 2013, pp. 305–312 (cit. on p. 81).

[34] Philipp Heise, Sebastian Klose, Brian Jensen and Alois Knoll. 'Pm-huber: Patchmatch with huber regularization for stereo matching'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2360–2367 (cit. on p. 85).

[35] Federico Tombari, Alessandro Franchi and Luigi Stefano. 'BOLD features to detect texture-less objects'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1265–1272 (cit. on p. 26).

[36] Matthew Cotter, Siddharth Advani, Jack Sampson, Kevin Irick and Vijaykrishnan Narayanan. 'A hardware accelerated multilevel visual classifier for embedded visual-assist systems'. In: *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press. 2014, pp. 96–100 (cit. on p. 13).

[37] David Eigen, Christian Puhrsch and Rob Fergus. 'Depth map prediction from a single image using a multi-scale deep network'. In: *Advances in neural information processing systems*. 2014, pp. 2366–2374 (cit. on pp. 78, 86, 95, 96).

[38] Emanuele Frontoni, Adriano Mancini, Primo Zingaretti and Valerio Placidi. 'Information management for intelligent retail environment: The Shelf Detector system'. In: *Information* 5.2 (2014), pp. 255–271 (cit. on p. 15).

[39] Marian George and Christian Floerkemeier. 'Recognizing products: A per-exemplar multi-label image classification approach'. In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 440–455 (cit. on pp. 12, 13, 24, 35, 36, 38, 53).

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. 'Generative adversarial nets'. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on p. 47).

[41] Lubor Ladicky, Jianbo Shi and Marc Pollefeys. 'Pulling things out of perspective'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 89–96 (cit. on p. 78).

[42] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang and Porter Westling. 'High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth.' In: *GCPR*. Ed. by Xiaoyi Jiang, Joachim Hornegger and Reinhard Koch. Vol. 8753. Lecture Notes in Computer Science. Springer, 2014, pp. 31–42. ISBN: 978-3-319-11751-5 (cit. on pp. 79, 99–101).

[43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan and Stefan Carlsson. 'CNN features off-the-shelf: an astounding baseline for recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 806–813 (cit. on pp. 30, 32, 46).

[44] Robert Spangenberg, Tobias Langner, Sven Adfeldt and Raúl Rojas. 'Large scale semi-global matching on the cpu'. In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE. 2014, pp. 195–201 (cit. on p. 86).

[45] Aristotle Spyropoulos, Nikos Komodakis and Philippos Mordohai. 'Learning to Detect Ground Control Points for Improving the Accuracy of Stereo Matching.' In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 1621–1628 (cit. on p. 81).

[46] Federico Tombari and Luigi Di Stefano. 'Interest Points via Maximal Self-Dissimilarities'. In: *Computer Vision–ACCV 2014*. Springer, 2014, pp. 586–600 (cit. on p. 25).

[47] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen and Ying Wu. 'Learning fine-grained image similarity with deep ranking'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1386–1393 (cit. on pp. 31, 33, 49, 61, 63, 64).

[48] Siddharth Advani, Brigid Smith, Yasuki Tanabe, Kevin Irick, Matthew Cotter, Jack Sampson and Vijaykrishnan Narayanan. 'Visual co-occurrence network: using context for large-scale object recognition in retail'. In: *Embedded Systems For Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*. IEEE. 2015, pp. 1–10 (cit. on p. 13).

[49] Sean Bell and Kavita Bala. 'Learning visual similarity for product design with convolutional neural networks'. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 98 (cit. on p. 31).

[50] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu and Chang Huang. 'A Deep Visual Correspondence Embedding Model for Stereo Matching Costs'. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 77).

[51] Tali Dekel, Shaul Oron, Michael Rubinstein, Shai Avidan and William T Freeman. 'Best-Buddies Similarity for robust template matching'. In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 2021–2029 (cit. on p. 27).

[52] Emanuele Frontoni, Adriano Mancini and Primo Zingaretti. 'Embedded Vision Sensor Network for Planogram Maintenance in Retail Environments'. In: *Sensors* 15.9 (2015), pp. 21114–21133 (cit. on p. 15).

[53] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg and Tamara L Berg. 'Where to buy it: Matching street clothing photos in online shops'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3343–3351 (cit. on pp. 31, 47).

[54] Diederik Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *Proceedings of the 3rd International Conference for Learning Representations*. 2015 (cit. on pp. 52, 54).

[55] Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel and Mingyi He. 'Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1119–1127 (cit. on p. 78).

[56] M Marder, S Harary, A Ribak, Y Tzur, S Alpert and A Tzadok. 'Using image analytics to monitor retail store shelves'. In: *IBM Journal of Research and Development* 59.2/3 (2015), pp. 3–1 (cit. on p. 15).

[57] Moritz Menze and Andreas Geiger. 'Object Scene Flow for Autonomous Vehicles'. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (cit. on pp. 75–77, 79, 83, 91, 93, 112, 119).

[58] Min Gyu Park and Kuk Jin Yoon. 'Leveraging Stereo Matching With Learning-Based Confidence Measures'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (cit. on pp. 80, 81).

[59] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. 'Faster R-CNN: Towards real-time object detection with region proposal networks'. In: *Advances in neural information processing systems*. 2015, pp. 91–99 (cit. on pp. 29, 30, 32).

[60] Olaf Ronneberger, Philipp Fischer and Thomas Brox. 'U-net: Convolutional networks for biomedical image segmentation'. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241 (cit. on pp. 51, 54).

[61] Florian Schroff, Dmitry Kalenichenko and James Philbin. 'Facenet: A unified embedding for face recognition and clustering'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823 (cit. on pp. 31, 46).

[62] Karen Simonyan and Andrew Zisserman. 'Very deep convolutional networks for large-scale image recognition'. In: (2015) (cit. on pp. 36, 58, 96).

[63] A. Spyropoulos and P. Mordohai. 'Ensemble Classifier for Combining Stereo Matching Algorithms'. In: *2015 International Conference on 3D Vision*. 2015 (cit. on p. 81).

[64] Gül Varol and Rıdvan S Kuzu. 'Toward retail product recognition on grocery shelves'. In: *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*. International Society for Optics and Photonics. 2015, pp. 944309–944309 (cit. on p. 13).

[65] Jure Zbontar and Yann LeCun. 'Computing the stereo matching cost with a convolutional neural network'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1592–1599 (cit. on p. 77).

[66] Ipek Baz, Erdem Yoruk and Mujdat Cetin. 'Context-aware hybrid classification system for fine-grained retail product recognition'. In: *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*. IEEE. 2016, pp. 1–5 (cit. on p. 13).

[67] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. 'The cityscapes dataset for semantic urban scene understanding'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223 (cit. on pp. 96, 99).

[68] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro and Ian Reid. 'Unsupervised cnn for single view depth estimation: Geometry to the rescue'. In: *European Conference on Computer Vision*. Springer. 2016, pp. 740–756 (cit. on pp. 78, 104, 105, 117, 123).

[69] Albert Gordo, Jon Almazán, Jerome Revaud and Diane Larlus. 'Deep image retrieval: Learning global representations for image search'. In: *European Conference on Computer Vision*. Springer. 2016, pp. 241–257 (cit. on pp. 30, 46).

[70] 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*. 2016, pp. 770–778 (cit. on pp. 58, 96).

[71] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari and Nassir Navab. 'Deeper depth prediction with fully convolutional residual networks'. In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, pp. 239–248 (cit. on p. 78).

[72] Gil Levi and Tal Hassner. 'LATCH: learned arrangements of three patch codes'. In: *2016 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2016, pp. 1–9 (cit. on p. 25).

[73] Fayao Liu, Chunhua Shen, Guosheng Lin and Ian Reid. 'Learning depth from single monocular images using deep convolutional neural fields'. In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2016), pp. 2024–2039 (cit. on p. 78).

[74] Wenjie Luo, Alexander G Schwing and Raquel Urtasun. 'Efficient deep learning for stereo matching'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5695–5703 (cit. on p. 77).

[75] Giulio Marin, Pietro Zanuttigh and Stefano Mattoccia. 'Reliable Fusion of ToF and Stereo Depth Driven by Confidence Measures'. In: *14th European Conference on Computer Vision (ECCV 2016)*. 2016, pp. 386–401 (cit. on p. 81).

[76] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy and Thomas Brox. 'A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 74, 75, 77, 79, 80, 87, 90, 106, 109, 111, 112, 119, 126).

[77] Christian Mostegel, Markus Rumpler, Friedrich Fraundorfer and Horst Bischof. 'Using Self-Contradiction to Learn Confidence Measures in Stereo Vision'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 4067–4076 (cit. on p. 81).

[78] Matteo Poggi and Stefano Mattoccia. 'Deep Stereo Fusion: combining multiple disparity hypotheses with deep-learning'. In: *Proceedings of the 4th International Conference on 3D Vision, 3DV*. 2016 (cit. on p. 81).

[79] Matteo Poggi and Stefano Mattoccia. 'Learning a general-purpose confidence measure based on O(1) features and a smarter aggregation strategy for semi global matching'. In: *Proceedings of the 4th International Conference on 3D Vision, 3DV*. 2016 (cit. on pp. 80, 81).

[80] Matteo Poggi and Stefano Mattoccia. 'Learning from scratch a confidence measure'. In: *Proceedings of the 27th British Conference on Machine Vision, BMVC*. 2016 (cit. on pp. 80, 81, 86, 87).

[81] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez and Antonio M. Lopez. 'The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016 (cit. on pp. 117, 126).

[82] Akihito Seki and Marc Pollefeys. 'Patch Based Confidence Prediction for Dense Disparity Map'. In: *British Machine Vision Conference (BMVC)*. 2016 (cit. on pp. 77, 81).

[83] Giorgos Tolias, Ronan Sicre and Hervé Jégou. 'Particular object retrieval with integral max-pooling of CNN activations'. In: (2016) (cit. on pp. 36, 52, 54, 56, 58).

[84] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra et al. 'Matching networks for one shot learning'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638 (cit. on pp. 47, 61, 63, 118).

[85] Xi Wang, Zhenfeng Sun, Wenqiang Zhang, Yu Zhou and Yu-Gang Jiang. 'Matching user photos to online products with robust deep features'. In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM. 2016, pp. 7–14 (cit. on p. 47).

[86] Erdem Yörük, Kaan Taha Öner and Ceyhun Burak Akgül. 'An efficient Hough transform for multi-instance object recognition and pose estimation'. In: *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE. 2016, pp. 1352–1357 (cit. on pp. 14, 38).

[87] Jure Zbontar and Yann LeCun. 'Stereo matching by training a convolutional neural network to compare image patches'. In: *Journal of Machine Learning Research* 17.1-32 (2016), p. 2 (cit. on p. 81).

[88] Xiaofan Zhang, Feng Zhou, Yuanqing Lin and Shaoting Zhang. 'Embedding label structures for fine-grained feature representation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1114–1123 (cit. on pp. 46, 61).

[89] 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*. 2017, pp. 1–16 (cit. on pp. 118, 126).

[90] Chelsea Finn, Pieter Abbeel and Sergey Levine. 'Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks'. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, Aug. 2017, pp. 1126–1135 (cit. on pp. 117, 118, 120).

[91] Annalisa Franco, Davide Maltoni and Serena Papi. 'Grocery product detection and recognition'. In: *Expert Systems with Applications* (2017) (cit. on p. 13).

[92] Spyros Gidaris and Nikos Komodakis. 'Detect, Replace, Refine: Deep Structured Prediction for Pixel Wise Labeling'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 77).

[93] Clément Godard, Oisin Mac Aodha and Gabriel J Brostow. 'Unsupervised monocular depth estimation with left-right consistency'. In: *CVPR*. Vol. 2. 6. 2017, p. 7 (cit. on pp. 78, 85, 86, 93, 95–98, 104, 105, 108, 110, 117, 127).

[94] Bharath Hariharan and Ross Girshick. 'Low-shot visual recognition by shrinking and hallucinating features'. In: *Proc. of IEEE Int. Conf. on Computer Vision (ICCV), Venice, Italy*. 2017 (cit. on p. 47).

[95] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. 'Mobilenets: Efficient convolutional neural networks for mobile vision applications'. In: *arXiv preprint arXiv:1704.04861* (2017) (cit. on p. 134).

[96] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama and Kevin Murphy. 'Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 3296–3297. DOI: 10.1109/CVPR.2017.351 (cit. on pp. 29, 30, 32).

[97] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A Efros. 'Image-to-Image Translation with Conditional Adversarial Networks'. In: *Computer Vision and Pattern Recognition, CVPR 2017*. 2017 (cit. on pp. 47, 51, 54, 56).

[98] Leonid Karlinsky, Joseph Shtok, Yochay Tzur and Asaf Tzadok. 'Fine-grained recognition of thousands of object categories with single-example training'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4113–4122 (cit. on p. 14).

[99] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach and Adam Bry. 'End-To-End Learning of Geometry and Context for Deep Stereo Regression'. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 77, 107).

[100] Sunok Kim, Dongbo Min, Seungryong Kim and Kwanghoon Sohn. 'Feature Augmentation for Learning Confidence Measure in Stereo Matching'. In: *IEEE Transactions on Image Processing* 26.12 (2017), pp. 6019–6033 (cit. on p. 81).

[101] Yevhen Kuznietsov, Jorg Stuckler and Bastian Leibe. 'Semi-Supervised Deep Learning for Monocular Depth Map Prediction'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 78).

[102] Jiahao Pang, Wenxiu Sun, Jimmy SJ. Ren, Chengxi Yang and Qiong Yan. 'Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching'. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2017 (cit. on p. 77).

[103] Matteo Poggi and Stefano Mattoccia. 'Learning to Predict Stereo Reliability Enforcing Local Consistency of Confidence Maps'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 81).

[104] Matteo Poggi, Fabio Tosi and Stefano Mattoccia. 'Efficient confidence measures for embedded stereo'. In: *19th International Conference on Image Analysis and Processing (ICIAP 2017)*. 2017 (cit. on p. 81).

[105] Matteo Poggi, Fabio Tosi and Stefano Mattoccia. 'Even More Confident predictions with deep machine-learning'. In: *12th IEEE Embedded Vision Workshop (EVW2017) held in conjunction with IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 81).

[106] Matteo Poggi, Fabio Tosi and Stefano Mattoccia. 'Quantitative Evaluation of Confidence Measures in a Machine Learning World'. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 81, 82).

[107] Siyuan Qiao, Wei Shen, Weichao Qiu, Chenxi Liu and Alan Yuille. 'Scalenet: Guiding object proposal generation in supermarkets and beyond'. In: *2017 IEEE International Conference on Computer Vision, ICCV*. 2017, pp. 22–29 (cit. on pp. 30, 32).

[108] Zhaofan Qiu, Yingwei Pan, Ting Yao and Tao Mei. 'Deep Semantic Hashing with Generative Adversarial Networks'. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 225–234 (cit. on p. 47).

[109] Anurag Ranjan and Michael J. Black. 'Optical Flow Estimation Using a Spatial Pyramid Network'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 106).

[110] Sachin Ravi and Hugo Larochelle. 'Optimization as a model for few-shot learning'. In: *In International Conference on Learning Representations (ICLR)*. 2017 (cit. on p. 118).

[111] Joseph Redmon and Ali Farhadi. 'YOLO9000: Better, Faster, Stronger'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition,CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690 (cit. on pp. 29, 30, 32, 36).

[112] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys and Andreas Geiger. 'A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos'. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 99–101).

[113] Akihito Seki and Marc Pollefeys. 'SGM-Nets: Semi-Global Matching With Neural Networks'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 77, 81).

[114] Amit Shaked and Lior Wolf. 'Improved Stereo Matching With Constant Highway Networks and Reflective Confidence Learning'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 81).

[115] Devashish Shankar, Sujay Narumanchi, HA Ananya, Pramod Kompalli and Krishnendu Chaudhury. 'Deep learning based large scale visual recommendation and search for E-Commerce'. In: *arXiv preprint arXiv:1703.02344* (2017) (cit. on pp. 31, 47).

[116] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch and Pieter Abbeel. 'Continuous adaptation via meta-learning in nonstationary and competitive environments'. In: *arXiv preprint arXiv:1710.03641* (2017) (cit. on p. 118).

[117] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang and Russ Webb. 'Learning from Simulated and Unsupervised Images through Adversarial Training'. In: *Computer Vision and Pattern Recognition, CVPR 2017*. 2017 (cit. on pp. 47, 50).

[118] Jake Snell, Kevin Swersky and Richard Zemel. 'Prototypical networks for few-shot learning'. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4077–4087 (cit. on p. 118).

[119] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke and Alexander A Alemi. 'Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.' In: *AAAI*. 2017, pp. 4278–4284 (cit. on p. 58).

[120] Fabio Tosi, Matteo Poggi, Alessio Tonioni, Luigi Di Stefano and Stefano Mattoccia. 'Learning confidence measures in the wild'. In: *Proceedings of the 28th British Machine Vision Conference (BMVC 2017), London, UK*. 2017, pp. 4–7 (cit. on p. 81).

[121] Eric Tzeng, Judy Hoffman, Kate Saenko and Trevor Darrell. 'Adversarial discriminative domain adaptation'. In: *Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 4 (cit. on p. 47).

[122] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox and Andreas Geiger. 'Sparsity Invariant CNNs'. In: *International Conference on 3D Vision (3DV)*. 2017 (cit. on pp. 86, 90, 96, 110, 117, 126).

[123] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy and Thomas Brox. 'Demon: Depth and motion network for learning monocular stereo'. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 5. 2017 (cit. on p. 78).

[124] Chao-Yuan Wu, R Manmatha, Alexander J Smola and Philipp Krähenbühl. 'Sampling matters in deep embedding learning'. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 46).

[125] Xu Zhang, Felix X Yu, Sanjiv Kumar and Shih-Fu Chang. 'Learning Spread-out Local Feature Descriptors'. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 46, 61, 63).

[126] Chao Zhou, Hong Zhang, Xiaoyong Shen and Jiaya Jia. 'Unsupervised learning of stereo matching'. In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 8. 2017 (cit. on pp. 77, 104, 110).

[127] Tinghui Zhou, Matthew Brown, Noah Snavely and David G Lowe. 'Unsupervised learning of depth and ego-motion from video'. In: *CVPR*. Vol. 2. 6. 2017, p. 7 (cit. on pp. 78, 96, 104).

[128] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A Efros. 'Unpaired image-to-image translation using cycle-consistent adversarial networks'. In: *International Computer Vision Conference, ICCV 2017*. 2017 (cit. on p. 47).

[129] Filippo Aleotti, Fabio Tosi, Matteo Poggi and Stefano Mattoccia. 'Generative Adversarial Networks for unsupervised monocular depth prediction'. In: *15th European Conference on Computer Vision (ECCV) Workshops*. 2018 (cit. on p. 78).

[130] Konstantinos Batsos, Changjiang Cai and Philippos Mordohai. 'CBMV: A Coalesced Bidirectional Matching Volume for Disparity Estimation'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 134).

[131] Jia-Ren Chang and Yong-Sheng Chen. 'Pyramid Stereo Matching Network'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 77).

[132] Jia-Ren Chang and Yong-Sheng Chen. 'Pyramid Stereo Matching Network'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5410–5418 (cit. on p. 107).

[133] Ignasi Clavera, Anusha Nagabandi, Ronald S Fearing, Pieter Abbeel, Sergey Levine and Chelsea Finn. 'Learning to Adapt: Meta-Learning for Model-Based Control'. In: *arXiv preprint arXiv:1803.11347* (2018) (cit. on p. 118).

[134] Arun CS Kumar, Suchendra M. Bhandarkar and Prasad Mukta. 'Monocular Depth Prediction using Generative Adversarial Networks'. In: *1st International Workshop on Deep Learning for Visual SLAM, (CVPR)*. 2018 (cit. on p. 78).

[135] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich and Dacheng Tao. 'Deep Ordinal Regression Network for Monocular Depth Estimation'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 78).

[136] Zehua Fu and Mohsen Ardabilian. 'Learning Confidence Measures by Multimodal Convolutional Neural Networks.' In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018 (cit. on p. 81).

[137] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li and Li Fei-Fei. 'MentorNet: Regularizing very deep neural networks on corrupted labels'. In: *Procedings of the International Conference on Machine Learning (ICML)*. 2018 (cit. on pp. 84, 124).

[138] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin and Shahram Izadi. 'StereoNet: Guided Hierarchical Refinement for Real-Time Edge-Aware Depth Prediction'. In: *15th European Conference on Computer Vision (ECCV 2018)*. 2018 (cit. on pp. 104, 111, 116).

[139] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou and Jianfeng Zhang. 'Learning for Disparity Estimation Through Feature Constancy'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 77).

[140] Reza Mahjourian, Martin Wicke and Anelia Angelova. 'Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D Geometric Constraints'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 78, 96).

[141] Simon Meister, Junhwa Hur and Stefan Roth. 'UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss'. In: *AAAI*. New Orleans, Louisiana, 2018 (cit. on p. 104).

[142] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel. 'A simple neural attentive meta-learner'. In: *In International Conference on Learning Representations (ICLR)*. 2018 (cit. on p. 118).

[143] Jiahao Pang, Wenxiu Sun, Chengxi Yang, Jimmy Ren, Ruichao Xiao, Jin Zeng and Liang Lin. 'Zoom and Learn: Generalizing Deep Stereo Matching to Novel Domains'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 77, 104, 110).

[144] Matteo Poggi, Filippo Aleotti, Fabio Tosi and Stefano Mattoccia. 'Towards real-time unsupervised monocular depth estimation on CPU'. In: *IEEE/JRS Conference on Intelligent Robots and Systems (IROS)*. 2018 (cit. on pp. 78, 104).

[145] Matteo Poggi, Fabio Tosi and Stefano Mattoccia. 'Learning monocular depth estimation with unsupervised trinocular assumptions'. In: *6th International Conference on 3D Vision (3DV)*. 2018 (cit. on pp. 78, 104).

[146] Mengye Ren, Wenyuan Zeng, Bin Yang and Raquel Urtasun. 'Learning to re-weight examples for robust deep learning'. In: *Procedings of the International Conference on Machine Learning (ICML)*. 2018 (cit. on pp. 84, 124).

[147] Deqing Sun, Xiaodong Yang, Ming-Yu Liu and Jan Kautz. 'PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 106, 107).

[148] Fabio Tosi, Matteo Poggi, Antonio Benincasa and Stefano Mattoccia. 'Beyond local reasoning for stereo confidence estimation with deep learning'. In: *15th European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 81).

[149] Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu and Simon Lucey. 'Learning Depth from Monocular Videos using Direct Methods'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 78, 96).

[150] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe and Elisa Ricci. 'Structured Attention Guided Convolutional Neural Fields for Monocular Depth Estimation'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 78).

[151] Zhichao Yin and Jianping Shi. 'GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 78, 96).

[152] Lidong Yu, Yucheng Wang, Yuwei Wu and Yunde Jia. 'Deep Stereo Matching With Explicit Cost Aggregation Sub-Architecture'. In: *AAAI Conference on Artificial Intelligence*. 2018 (cit. on p. 77).

[153] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal and Ian Reid. 'Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 78).

[154] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin and Jian Sun. 'ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 134).

[155] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser and Sean Fanello. 'ActiveStereoNet: End-to-End Self-Supervised Learning for Active Stereo Systems'. In: *15th European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 77, 93, 94, 104, 105).

[156] Yiran Zhong, Hongdong Li and Yuchao Dai. 'Open-World Stereo Video Matching with Deep RNN'. In: *15th European Conference on Computer Vision (ECCV 2018)*. 2018 (cit. on pp. 105, 117, 119).

[157] Alec Radford, Luke Metz and Soumith Chintala. 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks'. In: *International Conference on Learning Representation, ICLR 2016* () (cit. on p. 47).