ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

CICLO 33

SETTORE CONCORSUALE: 09/H1
SETTORE SCIENTIFICO-DISCIPLINARE: ING-INF/05

Deep Scene Understanding with Limited Training Data

Presentata da:
Pierluigi ZAMA RAMIREZ

Coordinatore di Dottorato:
Prof. Davide SANGIORGI

Supervisore:
Prof. Luigi DI STEFANO

ESAME FINALE ANNO 2021

Declaration of Authorship

I, Pierluigi ZAMA RAMIREZ, declare that this thesis titled, "Deep Scene Understanding with Limited Training Data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
Pierluigi Zama Ramirez

Date:
April 12, 2021

UNIVERSITÀ DI BOLOGNA

Abstract

Facoltà di Ingegneria ed Architettura
Dipartimento di Informatica - Scienza e Ingegneria

Dottorato di Ricerca

Deep Scene Understanding with Limited Training Data

by Pierluigi ZAMA RAMIREZ

Scene understanding by a machine is a challenging task due to the profound variety of nature. Nevertheless, deep learning achieves impressive results in several scene understanding tasks such as semantic segmentation, depth estimation, or optical flow. However, these kinds of approaches need a large amount of labeled data, leading to massive manual annotations, which are incredibly tedious and expensive to collect. In this thesis, we will focus on understanding a scene through deep learning with limited data availability. First of all, we will tackle the problem of the lack of data for semantic segmentation. We will show that computer graphics come in handy to our purpose, both to create a new, efficient tool for annotation as well to render synthetic annotated datasets quickly. However, a network trained only on synthetic data suffers from the so-called domain-shift problem, i.e. unable to generalize to real data. Thus, we will show that we can mitigate this problem using a novel deep image to image translation technique. In the second part of the thesis, we will focus on the relationship between scene understanding tasks. We argue that building a model aware of the connections between tasks is the first building stone to create more robust, efficient, performant models that need less annotated training data. In particular, we demonstrate that we can decrease the need for labels by exploiting the relationship between visual tasks. Finally, in the last part, we propose a novel unified framework for comprehensive scene understanding, which exploits the synergies between tasks to be more robust, efficient, and performant.

Author Publications

[1] Pierluigi Zama Ramirez, Matteo Poggi, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. "Geometry meets semantics for semi-supervised monocular depth estimation". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 298–313.

[2] Fabio Tosi*, Filippo Aleotti*, Pierluigi Zama Ramirez*, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mattoccia (*Equal Contribution). "Distilled semantics for comprehensive scene understanding from videos". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4654–4665.

[3] Pierluigi Zama Ramirez, Alessio Tonioni, and Luigi Di Stefano. "Exploiting semantics in adversarial training for image-level domain adaptation". In: *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*. IEEE. 2018, pp. 49–54.

[4] Pierluigi Zama Ramirez, Alessio Tonioni, Samuele Salti, and Luigi Di Stefano. "Learning across tasks and domains". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 8110–8119.

[5] Pierluigi Zama Ramirez, Claudio Paternesi, Luca De Luigi, Luigi Lella, Daniele De Gregorio, and Luigi Di Stefano. "Shooting Labels: 3D Semantic Labeling by Virtual Reality". In: *3rd International Conference on Artificial Intelligence and Virtual Reality*. IEEE. 2020.

[6] Pierluigi Zama Ramirez, Alessio Tonioni, and Federico Tombari. "Unsupervised Novel View Synthesis from a Single Image". In: *arXiv preprint arXiv:2102.03285* ().

Acknowledgments

I would like to thank T3Lab [1] for financing my PhD.

A huge thank to my supervisor Prof. Luigi Di Stefano for all the time spent in mentoring me, and for all the motivation and support he gave me through these years. His passion for research and Computer Vision is contagious, and it was the spark that prompted me to start this career.

I would like to thank my CVLab colleagues Matteo Poggi, Alessio Tonioni, Riccardo Spezialetti, Fabio Tosi, Filippo Aleotti, Luca De Luigi, Adriano Cardace, Gianluca Berardi, Daniele De Gregorio, Samuele Salti, Stefano Mattoccia. They are not only colleagues but friends which shared with me all the pains and joys of the PhD adventure. Going to the lab was always funny and exciting with them. Thank you guys!

I would like to thank Prof. Federico Tombari to welcoming me in his team and mentoring me during my 6 months experience as a Research Intern at Google. I would also like to thank Diego Martin Arroyo, and again Alessio Tonioni for all the help they gave me during these 6 months.

The rest of acknowledgments will be in italian for my family and friends.

Prima di tutto vorrei ringraziare la mia famiglia per essermi sempre stata vicino in questi anni. Ringrazio con tutto il mio cuore i miei genitori. Il vostro amore è stato fondamentale a superare i momenti più difficili, non ce l'avrai mai fatta senza di voi. Mi avete trasmesso la fiducia in me stesso di cui avevo bisogno per superare i molteplici fallimenti. Siete speciali, e non solo perchè sono vostro figlio. Grazie, vi voglio bene.

Grazie ai miei nonni, Maria, Gino, Paolo, Rosanna che sono sempre stati a fianco a me con dolcezza e tenerezza. Il vostro amore mi ha sempre donato la forza necessaria per andare avanti.

Grazie alla mia zia Anna, che con la sua ironia e dolcezza è sempre stata a fianco a me in questi anni.

Grazie a zio Claudio, che fin da quando ero bambino mi ha trasmesso la sua passione per l'informatica. Vorrei fossi qui a poter festeggiare con noi, ma sappi che sei sempre nel mio cuore.

[1] https://www.t3lab.it

Un ringraziamento speciale alla mia cuginetta. Sei la sorellina che non ho avuto, sappi che ti voglio un mondo di bene. Ma sappi anche che quando sarai fisioterapista sarai tu a doverti prender cura della mia schiena rotta per colpa della "scienza".

También quiero agradecer a mi familia en Colombia. Gracias abuelita Irene, tia Luz Irene, tio Ivan, tio Rafael, e gracias primos Mariana, Rafa y Santiago por todo el amor que siempre me das.

Grazie anche alla mia famiglia acquisita. Grazie a Mary e Claudio, per avermi accolto nella vostra casa come un figlio. Grazie Cri per la tua grande dolcezza e fiducia che hai sempre nei miei confronti.

Grazie alle cucciole, Carlotta e Gaia. Sappiate che lo zio vi vuole un mondo di bene. Sono molto fortunato ad avere due nipotine speciali e fantastiche come voi.

Grazie anche ai miei amici più cari. Grazie a Lore, Raul e Leo. Per quante idiozie possiamo dire tutto il giorno, insultandoci o (giustamente) insultando Raul, so che nel momento del bisogno posso sempre contare su di voi.

Grazie a Gala che fin dall'inizio dell'università è stato compagno di avventure e scemenze. Grazie per avermi fatto crescere ed educato a stare al mondo. Non penso che tu ci sia ancora riuscito completamente ma sento di aver fatto notevoli progressi da inizio università.

Grazie a Tom, compagno di giochi e passioni. Da quando ti sei trasferito per i big money si sente molto la tua mancanza, ma sappi che per quanto distanza ci sia tra noi due non ti libererai facilmente di me.

Grazie anche a Melissa, Lukas e la piccola Maya. Mi avete accolto nella vostra casa in un periodo molto difficile e probabilmente senza di voi sarei morto in Svizzera durante la pandemia.

Grazie a Mario, Scara e Bedo per essere stati dei gran coinquilini e amici e per tutte le serate di divertimento trascorse insieme.

Per ultimo, ma non sicuramente in importanza, ringrazio te, Leti. Le mie giornate, per quanto lunghe e faticose, sono sempre state piene di gioia e felicità. Tornare a casa la sera sapendo di poterti abbracciare è stato il motore più potente delle mie giornate. Questo dottorato è per metà anche tuo. Sei il mio amore meraviglioso. Grazie.

Infine ringrazio anche tutte le altre persone che mi hanno accompagnato in questi anni, i miei amici di infanzia, compagni di scuola e università, colleghi di lavoro.

Grazie a tutti!

Contents

List of Figures

List of Tables

To my family and Letizia...

Chapter 1

Introduction

1.1 Computer Vision and Deep Learning

What does it mean to see? Being aware of what is and where it is by looking. This would be the answer that most people would give to this question. In other words, Vision is the process of discovering from images what is present in the world and where it is [7]. If we think about it, it is incredible how our brain can instantaneously elaborate all the information coming from the world, in all its details, colors, and beauty, achieving a complete understanding of what and where things are. This is possible because our brain has the extraordinary capacity of finding extremely rich representation [8] of all this immense amount of information.

Computer vision is a multidisciplinary science that strives to give machines the ability to see [9]. As we can imagine, this is incredibly challenging due to the profound variety and complexity of the nature around us. In the last 70 years, thousands of scientists tried to approach this problem from different perspectives, from physiology, philosophy, psychology, engineering, computer science, and artificial intelligence.

Early approaches tried to find deductive models of nature, starting from strong assumptions of it. However, none of these techniques were able to achieve a high degree of understanding of the scene. On the other hand, in the last two decades, machine learning techniques [10] started to be a promising approach for designing systems with a human-level understanding of imagery. Machine learning systems can learn rich representations directly from data, which, in our case, are images, but they require a massive quantity of data to work. Nevertheless, nowadays, cameras are everywhere, on mobile phones, on personal computers, etc. Moreover, thanks to social media and the internet, we dispose of an immense number of images depicting the world around us. Thanks to this large availability of data, machines can

learn directly from the world, and they are moving towards a comprehensive understanding of it.

Among Machine Learning approaches, Deep Learning [11], thanks to Neural Networks, has revolutionized computer vision research and set forth a general framework to address a variety of visual tasks, to understand what is in the scene (e.g. semantic segmentation), to perceive where things are (e.g. depth estimation), to even understand how they are moving (e.g. optical flow) from images. In this thesis, we use deep learning approaches in several scene understanding tasks, with a special emphasis on semantic segmentation.

Despite these approaches' unquestionable power, most of them are supervised techniques, needing thousands of labels to be trained. In the last years, several approaches try to mitigate the need for labels. For instance, for depth estimation or optical flow, we devised self-supervised methods that can learn directly from images, leveraging on geometric constraints. However, we cannot rely on any geometric information for semantic segmentation, and we still need labels. In the next section, we present the semantic segmentation task and various solutions for addressing the data problem.

1.2 Semantic Segmentation and the Data Problem

Understanding what is inside the scene is the first fundamental step towards intelligent autonomous systems. Many technologies such as autonomous driving, medical imaging, and industrial robots need to know precisely what is in the world to take any action or decision.

In computer vision, semantic scene understanding is widely studied and addressed at different granularities [12]. As shown in Figure 1.1, given a reference image, we can perform:

- Image classification: understanding the objects in the image
- Object detection: classification and detection of each object
- Semantic segmentation: classification of each pixel of the image
- Instance segmentation: identification of pixel associated to different instances of the same class
- Panoptic segmentation: combination of the previous two

FIGURE 1.1: Semantic Scene Understanding at different granularities. From left to right: classification, object detection, semantic segmentation, instance segmentation, panoptic segmentation.

As stated above, nowadays, state-of-the-art approaches for semantic segmentation are based on deep learning and neural networks. These methods can achieve impressive results, reaching almost human-like performances. However, they suffer from the so-called domain-shift problem [13], where a model learned on a source data distribution (i.e. the training dataset) cannot to generalize to new unseen target data distributions. To give a practical example of this problem, let us imagine an autonomous driving car, whose vision system has been trained to drive only in California and then use the same system in a different country, for instance, Italy. There are many differences between the two countries, such as traffic signs and landscape. Thus, our system likely fail to recognize these elements, with catastrophic consequences, both for the driver and people around.

For this reason, although pixel-wise labeling is an extremely tedious and time-consuming process (e.g. approximately 3 hours for a 2048×1024 image [14]), many datasets have been created. For instance, we find Cityscapes [14], Apolloscapes [15], Mapillary [16] for autonomous driving, PASCAL VOC [17], ADE20K [18] for indoor scenes, [19] for human parsing, LiTS [20]for medical imaging. In light of these considerations, many recent works try to find ways to make the labeling process less painful and time-consuming or extract the knowledge from other sources with transfer learning, to reduce the amount of manually annotated data required.

In the following two sections, we describe two approaches investigated in this thesis to tackle the data problem. Firstly, we describe that it is possible to speed up data collection thanks to computer graphics techniques, by creating efficient annotation tools or by creating synthetic simulations of the real world. Moreover, we show that it is possible to mitigate the domain-shift problem of synthetic-data by using image-to-image translation techniques. Secondly, we describe that it is possible to exploit the relationship between visual tasks to decrease the need for labels and obtain more powerful models.

1.3 Creating Data with Computer Graphics

Computer graphics is the science that aims at generating images through a computer. Several computer vision works exploit computer graphics to produce an accurate simulation of the real world, to train and test algorithms. [21, 22, 23].

In the first part of this thesis, we show that computer graphics techniques may help us speed up collecting new labeled data. In chapter 4 we investigate a new technique that combines virtual reality and gamification with 3D reconstruction techniques to make labeling as easy and fun as playing a videogame. In chapter 5 we explore the possibility of obtaining labels for computer vision directly from the rendering of 3D synthetic models. In the last years, several synthetic datasets have been obtained in this way [24, 25]. However, though this strategy allows us to collect a large number of labeled images in a short amount of time, the domain-shift between synthetic and real data is significant. Though modern computer graphics can produce high-quality synthetic images that resemble the real world, renderings are still very different from real pictures in several aspects such as colors, textures, sensor noise, the shape of objects, and countless other factors. Therefore, in chapter 5 we propose to decrease the domain shift across real and synthetic data with image-to-image translation techniques. In particular, we propose to employ semantic information to improve the existing state-of-the-art approaches for image-to-image translation.

1.4 Transferring Knowledge Across Tasks

When we perceive a scene, our brain provides us a comprehensive understanding of it. We understand the semantics, the geometry, the motion of things jointly. These properties of the scene are tightly correlated one to another. For instance, the semantic of an object is strictly connected to its geometry and its way of moving. However, most state-of-the-art methods are approaching any single task, in isolation, ignoring the potentially beneficial relationship among them. Alternatively, a model aware of these relationships demands less supervision, uses less computation, and behaves in more predictable ways. Incorporating such a structure is the first stepping stone towards developing provably efficient comprehensive/universal perception models.

In the second part of the thesis, we show that it is possible to exploit this relationship across tasks to decrease the need for labels. Moreover, transferring knowledge across tasks is not helpful only for semantic segmentation, but it can be beneficial to a large set of visual tasks. For instance, performing depth estimation can decrease the need for semantic labels and vice-versa. In chapter 8 and chapter 9 we show a general framework to transfer knowledge across visual tasks to decrease the need for data leveraging on an auxiliary synthetic domain. Moreover, in chapter 10, we demonstrate the effectiveness of our framework in the standard benchmark for unsupervised domain adaptation for semantic segmentation achieving state-of-the-art results.

1.5 Comprehensive Scene Understanding and Multi-Task Learning

In the last part of the thesis, we make a further step towards the exploitation of visual task dependencies. We argue that jointly performing several tasks is key to boost performances and decrease computation requirements. We show that exploiting semantic information can be beneficial to several other visual tasks such as depth estimation, optical flow, camera pose estimation, or motion segmentation. In chapter 13 we show a first study showing that it is possible to exploit the synergies between depth and semantic to improve the former task, achieving state-of-the-art results with respect to previously published methods. In this preliminary investigation we still use some ground-truth labels for semantic segmentation. Finally, in chapter 14 we propose a unified framework for comprehensive scene understanding. Using the semantic information, we improve depth, optical flow, camera pose, and motion segmentation. Exploiting the synergy among these tasks allows the framework to be more robust, accurate, and lightweight (i.e., running in real-time on a standard GPU). Moreover our methodology can be trained without any manually annotated labels.

1.6 Structure of the thesis

To summarize the structure of the thesis is as follows:

Part 1 - Creating Data with Computer Graphics We present here some methodologies that employ computer graphics to generate annotations rapidly:

- chapter 4 - *3D Semantic Labeling with Virtual Reality*: we present here the novel tool for semantic segmentation with Virtual Reality
- chapter 5 - *Image to Image Translation for Synthetic to Real Adaptation*: we present here the novel methodology for image-to-image translation to decrease the domain-shift between synthetic and real data.

Part 2 - Transferring Knowledge Across Tasks for Domain Adaptation We propose our novel framework AT/DT that exploits the relationships between tasks to decrease the need for labels.

- chapter 8 - *Learning Across Tasks and Domains*: we present here the main AT/DT framework.
- chapter 9 - *Learning Good Features to Transfer Across Tasks and Domains*: We extend the AT/DT framework in case of structured tasks such as semantic and depth.
- chapter 10 - *AT/DT in Unsupervised Domain Adaptation for Semantic Segmentation*: we apply AT/DT in the standard unsupervised domain adaptation for semantic segmentation benchmark by transferring self-supervised depth to semantic.

Part 3 - Comprehensive Scene Understanding with Multi-Task Learning In this part we deepen the idea of exploiting the relationships among tasks to obtain robust, accurate and light models.

- chapter 13 - *Geometry and Semantics*: we report the preliminary study on self-supervised depth and semantic segmentation in a multi-task learning framework showing that semantic helps depth estimation.
- chapter 14 - *Geometry, Semantics and Motion*: we extend the previous chapter considering more scene understanding tasks. We show that we can build a robust, accurate, real-time model that can be trained without any manually annotated label by exploiting the synergies between different tasks.

Chapter 2

Deep Scene Understanding

Scene understanding is the process, often real-time, of perceiving, analyzing, and elaborating an interpretation of a 3D dynamic scene observed through sensors, typically cameras. Examples of scene understanding tasks from RGB images are visualized in Figure 2.1. As already outlined in chapter 1, nowadays, deep networks are the standard approach to address these tasks. In this section, we will review the essential works concerning scene understanding with deep neural networks. In particular, we will review the state-of-the-art methods for semantic segmentation, depth estimation, and optical flow, three tasks addressed in this thesis.

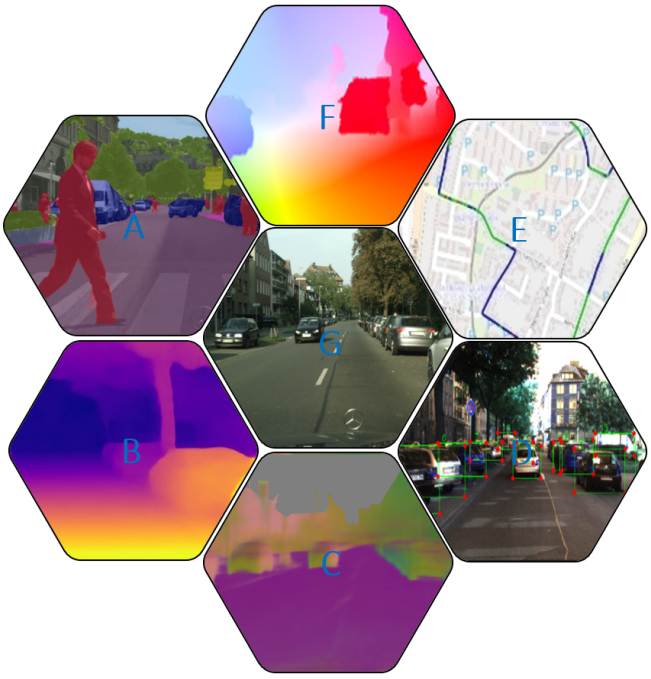

FIGURE 2.1: Example of scene understanding tasks from RGB images (G): semantic segmentation(A), depth estimation(B), normal estimation(C), Object Detection(D), visual odometry(E), optical flow(F)

2.1 Semantic Segmentation

Semantic Segmentation is the task of estimating the class for each pixel of the image. It is a fundamental problem in scene understanding that was tackled since the the advent of machine learning in computer vision.

While most early proposals relied on hand-crafted features together with classifiers like Random Forests [26] or Support Vector Machines [27], nowadays pixel-level semantic segmentation approaches mainly exploit fully convolutional neural networks [28]. Compared to previous methods, the present-day strategy's key advantage concerns the ability to *automatically* learn a better feature representation, mainly focusing on contextual information. A popular trend in semantic segmentation is to employ encoder-decoder architectures. The encoder is in charge of extracting low-resolution features from high-resolution inputs while the decoder should recover fine object details from the feature representation to yield a high-resolution output map [28, 29, 30, 31]. Early deep learning methods attempted to encode different context information levels from images by leveraging on multi-scale prediction models [32, 33, 34], whereby the same architecture takes inputs at different scales so as to extract features at different contextual levels. Another popular trend of early methods was to refine results of the segmentation networks by encoding long-range context information exploiting Conditional Random Fields (CRF) either as a post-processing module [35] or as an integral part of the network [36]. On the other hand, following methods such as [37] try to extract context information at different levels and scales by relying on spatial pyramid pooling [38]. More recently, popular architecture such as Deeplab [35, 39, 40] deployed atrous-convolutions rather than the standard convolution operator to extract higher resolution features while keeping a large receptive field to capture long-range information [41, 42]. Even though all previous methods achieved already impressive performance in Semantic Segmentation, further improvements were realized thanks to recent developments in Auto Machine Learning (AutoML) [43, 44] by leveraging architectural search to achieve state-of-the-art accuracy. An alternative research path deals with real-time semantic segmentation networks. In this space, [45] deploys a compact and efficient network architecture, [46] proposes a two paths network to attain fast inference while capturing high-resolution details. DABNet [47] finds an effective combination of depth-wise separable filters and atrous-convolutions to reach a good trade-off between efficiency and accuracy. [48] employs cascaded sub-stages to refine results while

FCHardNet [49] leverages on a new harmonic densely connected pattern to maximize the inference performance of larger networks.

2.2 Depth Estimation

Depth estimation is the task of estimating the distance from the camera for each pixel of an image. It is an essential task in scene understanding, and it is key to unlock exciting applications such as autonomous driving, 3D scene reconstruction, and augmented reality. In robotics, depth is a crucial prerequisite to perform multiple tasks such as navigation and planning.

In particular, single view depth estimation [50, 51, 52, 53] has gained much more popularity recently thanks to the increasing availability of benchmarks [54, 55].

The work by Garg *et al.*[56] represents the first, pivotal step in this direction, proposing a network for monocular depth estimation by deploying, at training time, view reconstruction loss together with actual stereo pairs as supervision. Then, Godard *et al.*[57] introduced bilinear warping [58] alongside with more robust reconstruction losses, thereby achieving state-of-the-art performance for monocular depth estimation. This approach was extended to embedded systems [59], using a virtual trinocular setup at training time [60] or a GAN framework [61], Kuznietsov *et al.*[62] trained a network in a semi-supervised manner, by merging the unsupervised image reconstruction error with the contribution from sparse depth ground-truth labels. Other works improved results using proxy labels from SGM [63, 64] or guidance from visual odometry [65].

While the techniques mentioned above require rectified stereo pairs at training time, Zhou *et al.*[66] proposed to train a network to infer depth from video sequences. This network computes a reconstruction loss between subsequent frames and, at the same time, predicts the relative poses between adjacent frames. Therefore, this method enables a fully-monocular setup whereby stereo pairs are no longer required for training. However, this strategy comes to a price in performance [66], delivering less accurate depth estimations compared to [57] and predicting depth-maps up to a scale factor. More recent works aimed at improving the video-sequence supervision approach because of its easiness of use, introducing 3D point-cloud alignment [67], differentiable visual odometry [68], or normal consistency [69]. Nevertheless, none of them outperform the synergy of stereo supervision and network model deployed by Godard et al. [57].

A novel trend introduced by recent works [70, 71, 72, 73, 74, 75] model rigid and non-rigid components using the projected depth, relative camera transformations, and optical flow to handle independent motions, which can also be estimated independently in the 3D space [76, 77]. In [78], the authors show how to learn camera intrinsics together with depth and ego-motion to enable training on any unconstrained video. In [79, 80, 81], reasoned design choices such as a minimum reprojection loss between frames, self-assembled attention modules, and auto-mask strategies to handle the static camera or dynamic objects proved to be very effective.

Supervision from stereo and video has also been combined [82, 79], possibly improved by proxy supervision from stereo direct sparse odometry [74]. Uncertainty modeling for self-supervised monocular depth estimation has been studied in [83].

Finally, lightweight networks aimed at real-time performance on low-power systems have been proposed within self-supervised [59, 84] as well as supervised [85] learning paradigms.

2.3 Optical Flow Estimation

The optical flow problem concerns estimation of the apparent displacement of pixels in consecutive frames, and it is useful in various applications such as, e.g. , video editing [86, 87] and object tracking [88]. Initially introduced by Horn and Schunck [89], this problem has traditionally been tackled by variational approaches [90, 91, 92]. More recently, Dosovitskiy *et al.*[93] showed the supremacy of deep learning strategies in this field. Then, other works improved accuracy by stacking more networks [94] or exploiting traditional pyramidal [95, 96, 97] and multi-frame fusion [98] approaches. Unfortunately, obtaining even sparse optical flow labels is extremely challenging, which renders self-supervision from images highly desirable. For this reason, an increasing number of methods propose to use image reconstruction and spatial smoothness [99, 100, 101] as main signals to guide the training, while paying particular attention to occluded regions [102, 103, 104, 105, 106, 107].

Part I

Creating Data with Computer Graphics

Chapter 3

Initial Remarks

Training a neural network requires data. Generating ground-truth data is the main bottleneck when using this technology, both for research and industry. Indeed, when facing a new task or scenario, the first step is always collecting enough data to train the neural model properly. However, the annotation process is incredibly time-consuming in pixel-wise tasks such as semantic segmentation and even harder for geometric tasks such as depth estimation or optical flow. Computer graphics comes in handy for this, for instance, by helping in building new efficient and user-friendly annotation tools. Indeed, in chapter 4 we propose a new efficient labeling tool based on virtual reality to speed-up the annotation process. Given a 3D reconstruction of a scene we can load it into a virtual world and we can move around intuitively. Moreover, the practical game-style interface enlarge the pool of user being able to use our framework, paving the way for new kind of post-processing based on multi-player integration to automatically refine results.

Nevertheless, apart from the tool's efficiency, manually annotating data for each task and scenario is not scalable. Thus, a recent trend is to use computer graphics for creating a virtual simulation of the real world. From this simulation, we can produce thousands of ground-truths in few minutes of computation. Nowadays, computer graphics engines can create incredibly realistic simulations. However, synthetic data still differ from real ones for several factors such as illumination, the sensor noise, or unrealistic 3D models. Thus, training a neural network on synthetic data produces degraded results when tested on real scenarios because of the domain-shift problem. In chapter 5 we propose a novel technique to mitigate the domain-shift problem when training on simulated data (chapter 5). In particular, we employ style-transfer techniques to improve the realism of synthetic images. Peculiar to our approach is the use the semantic information to improve the style-transfer quality.

In the following sections we review some works relevant to this part of the thesis.

3.1 Semantic Segmentation Datasets

Obtaining a large amount of labeled data is crucial to achieving good performance in deep learning algorithms. Thus, even though annotating a dataset is a tedious and time-consuming process, several datasets featuring 2D images annotated with semantic labels are available. In the case of outdoor urban scenarios, the most popular are KITTI [108] and Cityscapes [109], which, yet, contain a relatively small number of images semantically annotated by hand. They were two of the first datasets proposed in this area (urban outdoor), so the focus was more on the quality of the data rather than the annotation process's scalability. The Mapillary dataset [16] includes many more images, though the labeling was still performed image by image by hand. The same is true for some indoor datasets, such as [110] and [111]. Although smart graphical tools are used to produce frame-by-frame annotations, the magnitude of the available images is only slightly higher. Indeed, annotating each 2D image by hand is not scalable.

Conversely, [112] shows an efficient pipeline for indoor environments. They propose a pipeline composed of three steps: RGBD scan, 3D reconstruction, and 3D labeling showing the increased speed by translating the annotation process in 3D. In [113] such procedure is formally extended with a projection module which, based on known camera poses, brings the 3D labels into 2D. The label projection approach was then exploited in other datasets, such as [114] and [115]. It can be observed that leveraging on 3D reconstruction and camera tracking to facilitate labeling may be thought of as shifting the cost of labeling each individual image toward the complexity of the requirements necessary to obtain a suitable dataset (tracked camera) and a 3D annotation tool. This benefit is even more evident in synthetic datasets, such as [23, 116, 117], where obviously both camera tracking and 3D reconstruction are no longer external elements but inherent to the rendering engine and we just need to annotate and render thee 3D model. Following this trend [15, 118, 119] have proposed large urban outdoor 2D-3D datasets. The annotation task is performed on point clouds, and semantically labeled images are attained by projection.

However, when dealing with 3D annotation, the most used tools are open-source software such as Blender or Meshlab, which require expertise in 3D

modeling. Some authors have expressly addressed this by proposing smart solutions. In [120], the authors have proposed an interactive procedure where the user can physically touch the object in the scene to label it. Moreover, they exploit region growing techniques to color large parts of the scene expeditiously. In [121], the authors build a physical device able to reproduce the pipeline where the user navigates the environment in Augmented Reality, using a laser pointer to identify the homogeneous areas of the scene and assign them a correct label. Differently, in chapter 4 we introduce a Virtual Reality framework to navigate the reconstructed environments. We provide the user with a series of intuitive gamification tools to label the scene expeditiously. To the best of our knowledge, ours is the first method that allows for labeling very-large-scale scenes in a short time by a VR approach.

Another strategy to collect labeled data is to leverage computer graphics engines to build a virtual simulation of the real world, and render synthetic images. In this way, in only a few minutes of computation, we can collect thousands of images and corresponding labels such as semantic, depth, or optical flow maps. Synthetic datasets such as Synthia [24] and Carla [25] were obtained by producing ad-hoc 3D simulations for an autonomous driving scenario. In [21] the authors extract render-layers information from the famous video-game Grand Theft Auto V (GTAV). Nevertheless, using only synthetic data for training a models lead to poor performances in real scenarios. For this reason we review essential works for Domain Adaptation that explicit address the domain-shift issue.

3.2 Domain Adaptation

Domain Adaptation is the research topic that address the domain-shift problem [122, 123, 124]. This field aims at learning models that turn out robust when tested on data sampled from a domain different from the training one.

Throughout the years, adaptation has been performed at different levels, addressing mainly the image classification task. The typical trend for domain adaptation for image classification is to learn a domain-invariant or domain-aligned feature space. Early approaches try to model a shared feature space across domains by relying on statistical metrics such as MMD [125, 126, 127]. Later, some works proposed to align domains by adversarial training [128, 129, 130, 131]. Recently [132] noticed that aligning feature norms to an arbitrarily large value results in better transferability across domains.

However, when dealing with dense tasks such as semantic segmentation, domain adaptation approaches designed for classification typically fail. Thus, several approaches address explicitly domain adaptation for dense tasks, such as semantic segmentation. We can divide most of these methods into two categories: feature-level and pixel-level. Feature-level methods [133, 127, 130, 134, 128, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144] try to align the feature representation extracted from CNN across different datasets, usually, again, by using adversarial training. On the other hand, pixel-level approaches [145, 146, 147] convert the source image into a target-style image relying on recent image-to-image translation generative networks [148, 149]. Subsequent works [150, 151, 152, 153, 154, 155, 156, 157, 158] take the best of the worlds and operate at both pixel and feature level.

More recently, a new research line focuses on Self-Training [159], where a semantic classifier is fine-tuned directly on the target domain, using its predictions as pseudo-labels. [160, 161, 162] cleverly set class-confidence thresholds to mask wrong predictions. [163, 164, 165] propose to use pseudo-labels with different regularization techniques to minimize both the inter-domain and intra-domain gap. On the other hand, [166] synthesizes new samples for the target domain by cropping objects from source images using ground truth labels and pasting them onto target images.

In chapter 5, we propose a novel pixel-level approach that exploits semantic discriminators to improve the image-to-image translation process.

Chapter 4

3D Semantic Labeling with Virtual Reality

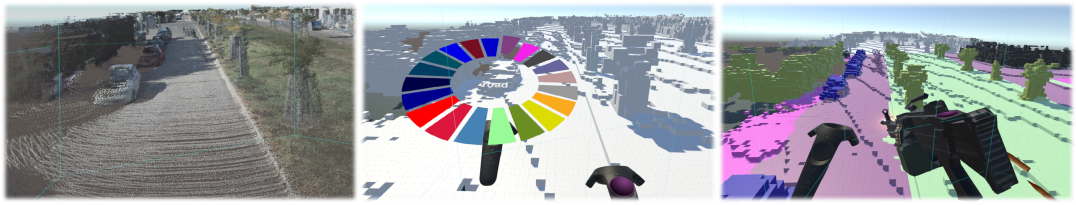

FIGURE 4.1: **V**irtual **R**eality view of a 3D reconstruction from KITTI sequence. Left image: RGB visualization of the virtual world. Middle image: label palette and empty voxelization. Right image: partially labeled environment.

In this chapter we propose Shooting Labels, a novel tool based on Virtual Reality (VR) to ease the dense 3D semantic labeling, so as to gather 3D and 2D data endowed with semantic annotations. To the best of our knowledge, ours is the first system which allows for handling efficiently large-scale 3D semantic labeling processes, such as labeling whole city blocks. Moreover, by exploiting Virtual Reality to make the task of labeling as easy and fun as playing a video-game, our approach remarkably reduces the expertise necessary to work with 3D semantic labeling tools. The immersive experience provided by VR technologies allows the user to physically move around within the scenario she/he is willing to label and interact with objects in a natural and engaging way. The user is transported into a large virtual environment represented as 3D meshes, where surfaces can be colored semantically in a highly captivating way (see. Figure 4.1 in-game visualizations).

The full fledged gamification of our tool empowers a larger community to undertake this type of activity and enables the possibility to obtain much more annotated data. For this reason, our tool features a multi-player post-processing procedure wherein we integrate results of several annotators to

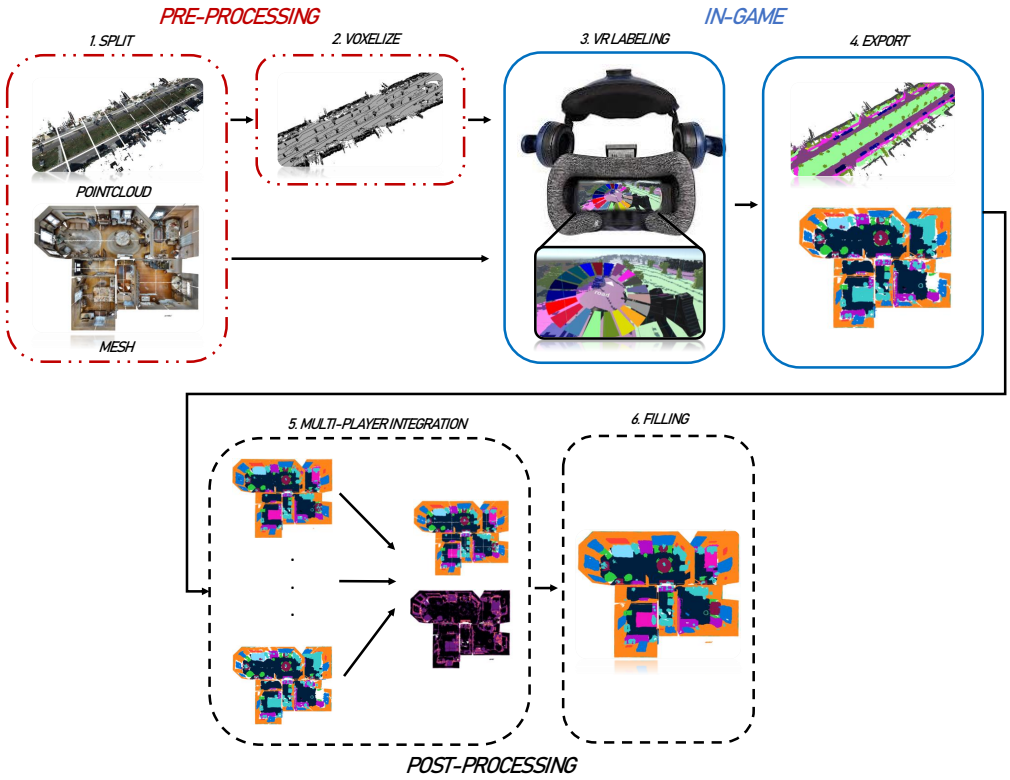

FIGURE 4.2: The six steps of the Shooting Labels pipeline. 1- Splitting 3D data (a mesh or point cloud) into chunks to optimize visualization in the VR environment. 2- In case of point clouds, voxelization enables real time rendering and reduces memory footprint 3- Semantic labeling by Virtual Reality. 4- Exporting annotated data into the original format 5- Integrating multi-player results to improve labeling and assess about the reliability of the labels (optional). 6- Filling of unlabeled elements (optional).

both improve accuracy and compute a labeling uncertainty map which provides information about the reliability of the produced ground truth.

Our open source framework is based on Unity[1], Blender[2] and open3D [167].

4.1 VR Labeling Tool

In this section we describe the key features of our tool. Shooting Labels works with the most popular 3D representations, such as point cloud and meshes, which can be obtained by any kind of 3D reconstruction technique. Moreover, with our tool we can also load a 3D scene pre-labeled by any other

[1] https://unity.com/
[2] https://www.blender.org/

technique (e.g. a CNN for 3D semantic segmentation) in order to refine it. As shown in Figure 4.2, our pipeline can be summarized into 6 main steps grouped into 3 stages, with the first stage dealing with *Pre-processing* of the input 3D Data, the second with *In-Game* labeling and the third with *Post-processing* of the labeled 3D data.

4.1.1 Pre-Processing of the Input 3D Data

Meshes and point clouds obtained by 3D reconstruction techniques typically consist of millions of vertices which can hardly be rendered in real-time in a VR environment. For this reason, with both meshes and point clouds we employ a Level Of Detail strategy to mitigate the computational demand. As illustrated in the first step of Figure 4.2, we split meshes and point clouds in several chunks, saving each chunk at 3 different resolutions. During a VR labeling session, objects closer to the player are loaded at a higher resolution than those farther away. Furthermore, as point clouds cannot be managed by the Unity gaming engine, we voxelize them (see Figure 4.2) in order to obtain a friendly visualization both in term of light computation and user-experience during navigation. To perform voxelization we set a discretization step and, for each position of the dense 3D grid, build a cube mesh if that volume contains a minimum number of points (e.g. > 5).

4.1.2 In-Game Labeling

3D meshes are loaded into the Virtual World and the user can explore and label the environment. The player can teleport or physically move around the scene to reach each portion of the environment. The following features have been implemented to enhance and simplify the user experience:

- Geometric and RGB visualization
- Unlabeled Face Visualization
- Level of Detail (LOD)
- Labeling Granularity
- Export of Final Results

Geometric and RGB Visualization To assign a semantic label to a mesh, the user paints on the geometric view of the object (Figure 4.1 central picture). However, in the 3D reconstruction objects may be difficult to disambiguate without color cues. To address this, in case of meshes, we directly visualize

the RGB version of the mesh if available. As for point clouds, we noticed that coarse RGB voxelizations can lead the user to misunderstand the scene. Thus, we visualize directly the RGB point cloud, building a mesh object for each point to enable visualization of this type of data also within the Unity rendering system (Figure 4.1 left image). We did not employ this kind of visualization during labeling because the interaction with this type of data can be extremely slow. However, we obtain smooth rendering performances for only the visualization.

Unlabeled Face Visualization Reaching some portion of the 3D space can be hard (e.g small hidden faces), or the user might wish to visualize the progress of its labeling. Thus, we keep track of the faces labeled by the user and, at any moment, allow the user to visualize only the faces still unlabeled.

Level of Detail As already mentioned, we implement a Level Of Detail (LOD) optimization to enable real-time rendering of large-scale scenarios. For each chunk obtained by splitting the mesh we keep 3 versions at different LOD and dynamically load at high resolution only the meshes within an action range. The user can interact only with the meshes at highest resolution, those closer to him, thereby significantly alleviating the overall computational burden.

Labeling Granularity A user may require different labeling resolution degrees so as to, e.g., colour either large surfaces or small details. Therefore, she/he can choose between a pool of different *weapons* which feature different action ranges, thereby enabling either a more precise or faster labeling. The user chooses the current label from a color palette (Figure 4.1, central picture) and when shooting toward a direction we color each face within the weapon action range of the first hit face. When hitting a face, in Unity we know only the hit face and we must find each face up to a range. As analyzing all faces of the scene can be extremely slow, thus impractical for real time rendering, we search between the faces belonging only to the the same object chunk.

Exporting Final Results At the end of VR Labeling phase we can save the progress or export the annotated mesh. During export chunks are merged together into the original mesh. In case the source data was a point cloud, we assign to each 3D point the label of the corresponding voxel.

4.1.3 Post-Processing of the Labeled 3D Data

Once the labeled data have been exported, the tool offers some optional post-processing step.

Multi-Player Integration The gamification process potentially enlarge the pool of possible users of our tool. Thus, we can exploit the redundancy of labeling to predict the most confident label for each face or point. Let us denote as $\mathcal{S}$ our mesh or point cloud, composed by several elements, e, i.e. faces or points respectively. For each element $e \in \mathcal{S}$ we define as $\mathcal{H}$ its corresponding histogram of labels, as assigned to it by different players. We can assign to each element e its most confident label by simply finding the most frequent label:

$$e = argmax(\mathcal{H}), e \in \mathcal{S} \tag{4.1}$$

Label Uncertainty Since the annotation process by a single user may contain errors, we might wish to know the uncertainty associated with each label.

Given n annotators we can easily get the label probability distribution $\mathcal{P} = \frac{\mathcal{H}}{n}$ for each element e. From the probability distribution we can calculate its entropy:

$$\mathcal{E} = -\sum p \log p \tag{4.2}$$

The entropy of that distribution can be treated as the uncertainty of the labeling for that element, u_e. We can leverage this uncertainty to decide which points should be considered noisy ground truth in the annotation process. Moreover, we could exploit it to refine only the high entropy elements both manually or by means of suitable algorithms.

Filling Some users may decide to label only partially the whole scenario. Moreover, during the pre-processing we may lose information about few faces where no labels will be available. For these reasons, Shooting Labels provides a function for automatic filling missing elements based on their neighborhood. We define $\mathcal{S}_{unlabeled}$ the set of elements without any label assigned and $\mathcal{S}_{labeled}$ the set of elements with a label assigned such as $\mathcal{S} = \mathcal{S}_{labeled} \cup \mathcal{S}_{unlabeled}$. Given one point $s_{unlabeled} \in \mathcal{S}_{unlabeled}$, we can find its K closest elements $s_{labeled} \in \mathcal{S}_{labeled}$ and their labels, and build the histogram H of labels of its neighborhood. Then we can easily infer its label:

$$y = argmax(\mathcal{H}) \tag{4.3}$$

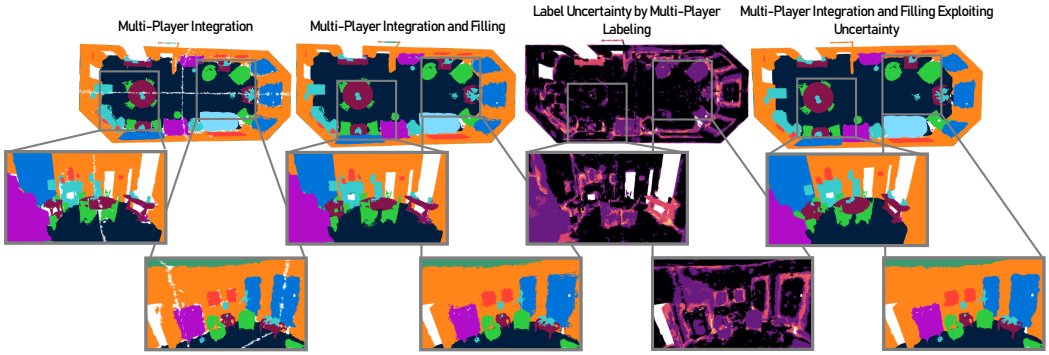

FIGURE 4.3: Qualitative comparison of different combination of post-processing steps. From left to right: results after only the multi-player integration step; multi-player integration and filling steps without label uncertainty; label uncertainty map; multi-player integration and uncertainty aware filling. We can notice that the fourth image have much smoother edges than the second one thanks to exploitation of the uncertainty map.

In a multi-player setting we can leverage the uncertainty information to further improve the precision and accuracy of the labeling by semantically filling also high uncertainty elements by means of their neighborhood. More precisely, given a fixed uncertainty threshold th_u, we consider unlabeled elements that have their uncertainty u_s above th_u:

$$\hat{\mathcal{S}}_{labeled} = \{s \in \mathcal{S}_{labeled} | u_s > th_u\} \tag{4.4}$$

$$\hat{\mathcal{S}}_{unlabeled} = \mathcal{S}_{unlabeled} \cup \hat{\mathcal{S}}_{labeled} \tag{4.5}$$

For each $\hat{s}_{unlabeled} \in \hat{\mathcal{S}}_{unlabeled}$ we can find its neighborhood composed by its K closest elements e_k. For each e_k we know its labels y_k and its uncertainty information u_k. Thus, we can build a weighted histogram of labels for the considered neighborhood, $\mathcal{H}_u$, collecting the votes for each label multiplied by their own uncertainty.

At this point the label of $\hat{s}_{unlabeled}$ will be $y = argmax(\mathcal{H}_u)$.

Obtaining 2D Segmentations We leverage the 3D segmentation to produce 2D segmentations of known RGB images. If a specific set of images comes along its intrinsic and extrinsic camera parameters we can seamlessly render the segmentation through the Blender render engine.

Player	Bed	Ceiling	Chair	Floor	Furniture	Object	Picture	Sofa	Table	Wall	Window	Perc.Labeled	mIoU
1	74.63	**95.19**	69.35	84.16	78.83	58.66	74.74	85.99	55.14	86.86	71.84	**97.04**	75.94
2	76.37	90.29	72.45	**88.16**	**80.87**	60.76	**77.36**	82.39	**62.20**	85.65	69.43	94.56	76.90
3	74.15	93.53	71.19	80.28	67.14	50.12	66.80	80.43	56.56	86.31	70.02	94.31	72.41
4	-	90.07	70.74	80.12	58.78	44.89	64.32	80.60	38.79	84.72	68.93	88.44	62.00
5	77.69	89.10	35.99	77.76	56.09	41.87	46.90	33.36	40.49	80.95	44.25	91.76	56.77
Integr.	**81.46**	94.80	**77.34**	87.02	79.46	**64.27**	75.35	**89.68**	57.70	**89.83**	**74.93**	94.47	**79.26**

TABLE 4.1: Comparison between single player and multi-player results on the Matterport 3D dataset [115]. Best results in bold.

4.2 Experimental Results

4.2.1 Efficiency and Accuracy of the Tool

To evaluate the efficiency and performance of our tool we tested it on the Matterport 3D dataset [168]. To perform the evaluation we considered the labeling provided by Matterport as our ground truth. Their labeling has been attained through a series of refinement steps based on several different tools and expertises. They first produced a coarse annotation with a first tool for planar surface labeling, then they used the ScanNet crowd-sourcing interface by Dai et al. [113] to "paint" triangles and name all object instances of the house. Finally a team of 10 expert annotators refined, fixed and verified the quality of the final labeling. In ours tests we labeled an entire Matterport house[3] made out of 6 rooms based on the 13 classes of objects defined in [169] (eigen13 categories). We exploit the mapping provided by Matterport from their labeling to eigen13 to obtain ground truths used for testing. They provide face-wise labels since ground truth are meshes. Therefore, we evaluated our results using a mean intersection over union weighted on the area of the faces:

$$IoU^{c}_{faces} = \frac{A_{TP}}{A_{TP} + A_{FN} + A_{FP}}, c \in Cl \tag{4.6}$$

$$mIoU_{faces} = \frac{1}{N} \sum_{c \in Cl} IoU^{c}_{faces} \tag{4.7}$$

where Cl is the set of classes, N is the total number of classes, A_{TP}, A_{FN}, A_{FP} represent the total area of the true positive, false negative and false positive faces respectively for a class c. Furthermore, we provide the percentage of

[3]Matterport House ID: `2t7WUuJeko7`

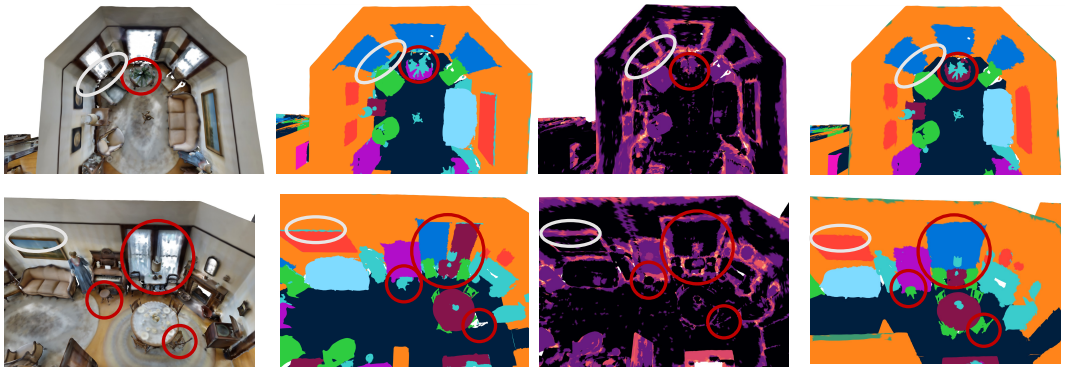

FIGURE 4.4: Matterport 3D dataset. From left to right: RGB mesh, ground-truth provided by Matterport, Uncertainty map by multi-user integration, best results obtained with our tool. Red circles: errors in the Matterport ground-truth (table as furniture, windows as table) avoided by labeling with our tool. White circles: high uncertainty labels (e.g. 3D boundaries).

area of faces annotated $A_{labeled}$ over the total area of the labeled ground truth A_{total}.

$$Perc.Area = \frac{A_{labeled}}{A_{total}}\% \tag{4.8}$$

Single Player and Multi-Player Integration Results

We evaluated the annotation of 5 different players without any expertise in 3D modeling. We compared their results with the ground truth provided by Matterport. The results are shown in Table 4.1. The average time needed for the labeling was about of 2.5 hours. Even though there are users who achieved low labeling performances (player 4 and 5), we notice that integrating results of all players yield the best overall performances of 79.26% mIoU, surpassing the accuracy of each single user. As we wanted to analyze what are the most common errors in labeling, we inspected qualitatively each single player results noticing that most frequent errors are correlated with 3D object boundaries and ambiguous object. Therefore, we manually investigate also the GT provided by Matterport finding the same types of errors. In Figure 4.4, we circled in white the errors on object boundaries while in red the completely mismatching object between our labeling and the Matterport ground truth. In second row we note that a window (Blue Label) was labeled as a Table (Red Label) in the Matterport GT while with our tool we did not encounter that error. We also found a case of an ambiguous object where a stool has been labeled as an Object (Light Blue Label) by the Matterport ground truth while as a chair by our users (Green Label). These ambiguities

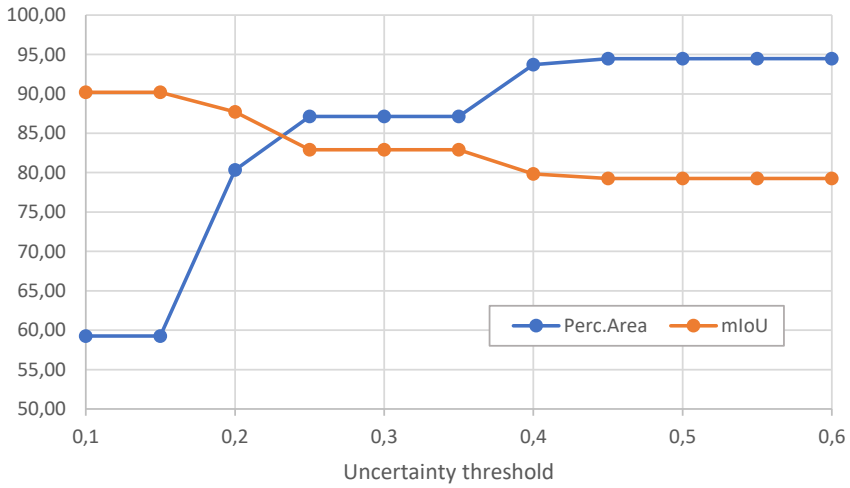

FIGURE 4.5: Evaluation at different threshold of uncertainty. The more noisy labels we discard the higher mIoU wrt Matterport3D ground truth.

and errors in both labeling might be the main cause of achieving lower mIoU score in our labeling and therefore, discarding them with our uncertainty information should lead to better overall performances.

Uncertainty Map Evaluation By integrating the results of several users we computed the label uncertainty map shown in the third column of Figure 4.3 and Figure 4.4. As we wish to evaluate the quality of our labeling, we analyzed the performances of our labeled mesh at different uncertainty thresholds. In Figure 4.5 we show the mIoU and the Perc.Area labeled at different uncertainty thresholds. For each threshold we evaluated the mIoU only on the elements with lower uncertainty than the threshold. We see that the higher the threshold the lower the mIoU, symptom of a good uncertainty map. In Figure 4.4 the third column are the uncertainty maps of the labeling where warmer color represents higher uncertainty. We notice that while white circled error are always correlated to high uncertainty, red circled errors might have low uncertainty. This happens because there are object that the majority of the annotators labeled in the same way while they are labeled different in Matterport ground truth which is correlated to an error in the Matterport ground truth (Windows labeled as a Table).

Filling Results w/ or w/o Uncertainty Table 4.2 shows the results of the filling step in various setting. The first five rows report the results obtained by applying filling immediately after the single player annotation. These rows highlight that we were able to fill all the unlabeled faces obtaining Perc.Area of 100% and slightly lower performance. We can see a similar trend also in multi-player integration result where we score a 76.11% mIoU while

Player	Bed	Ceiling	Chair	Floor	Furniture	Object	Picture	Sofa	Table	Wall	Window	Perc.Labeled	mIoU
1	73.77	**95.18**	68.09	82.94	78.04	56.28	73.79	**85.56**	54.29	86.13	71.58	100	75.06
2	75.90	90.28	65.55	**84.69**	**78.31**	55.44	**76.00**	67.00	**58.60**	84.05	68.19	100	73.09
3	73.21	93.23	67.00	77.54	65.27	48.08	65.57	75.25	52.64	84.46	68.68	100	70.09
4	-	88.86	57.47	73.92	54.73	39.56	59.50	68.01	30.52	80.93	65.67	100	56.29
5	71.56	88.82	35.13	73.79	54.49	38.55	46.13	32.30	38.50	78.59	43.89	100	54.71
Integr.	**80.33**	94.70	**70.27**	83.00	77.90	**57.98**	74.51	82.94	53.86	**87.85**	**73.85**	100	**76.11**
Integr.0.5	80.36	94.71	70.30	83.02	77.94	57.99	74.58	82.86	53.95	87.87	73.86	100	76.13
Integr.0.65	77.97	94.85	70.15	82.13	77.84	57.24	74.71	85.28	50.65	87.91	74.78	100	75.77
Integr.0.8	75.17	94.40	68.62	81.10	77.91	55.62	73.21	81.91	47.40	87.13	70.27	100	73.89

TABLE 4.2: Comparison between filling single users, multi-player integration and multi-player integration based on label uncertainty on the Matterport 3D dataset [115]. Best results of filling without uncertainty in bold. Best result of filling integration using uncertainty in red.

gaining a +5.53% on the Perc.Area labeled and losing only the 3.15% in mIoU with respect to the results without filling. The last three rows report the results of filling by exploiting the uncertainty map and using different threshold levels, i.e. 0.5, 0.65 and 0.8 respectively, with the best results of 76.13% mIoU attained with threshold 0.5. The decreasing trend with higher threshold can be explained thinking that using an higher threshold corresponds with considering good a lot of uncertain elements making filling too difficult and noisy. Figure 4.3 depicts a qualitative comparison between filling strategies. From left to right we can see the labeled mesh before the filling, the mesh filled without uncertainty, the uncertainty map and filling exploiting uncertainty. We highlight how, though the increase in performance in uncertainty aware filling is small, the qualitative results highlight a much smoother labeling.

3D to 2D Projection Figure 4.6 illustrate qualitative results of the 2D labels obtained by projecting 3D labels. Given availability of the RGB image (top left image) and its associated camera intrinsic and extrinsic parameters, we can configure the Blender rendering engine and position the virtual camera in order to obtain the 2D render of the scene. Peculiarly to our tool, we can also provide a 2D uncertainty map by projecting the 3D uncertainty map. Bottom left and right images are rendered from the filled mesh with (right) or without (left) exploiting uncertainty. We can notice that the right image has smoother edges and several blobs are less noisy. The render took place in approximately 1 second on a GTX 1080 Ti, much less than the hours needed by manual pixel-wise annotation.

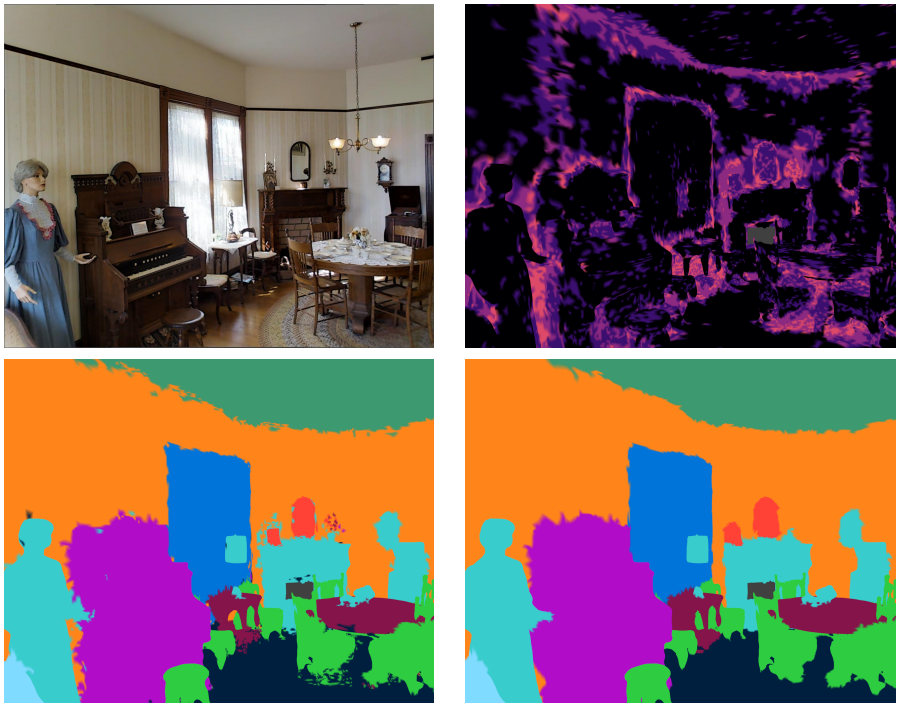

FIGURE 4.6: Matterport3D: projection of 3D labels into 2D labels. Top Left: RGB image with known camera pose. Bottom Left: 2D labels from a semantically filled mesh without exploiting uncertainty. Top Right: 2D uncertainty map. Bottom Right. 2D labels from a semantically filled mesh by exploiting uncertainty.

Large Scale Outdoor Labeling We evaluated the effectiveness of our tool in a challenging outdoor scenario: the Kitti Odometry Dataset [108]. We used the provided 3D Lidar data of a static sequence[4], consisting of more than 1000 images equipped with ground truth camera poses. We reconstructed the point cloud, then voxelized and labeled it by our tool. Then, we were able to annotate the whole sequence in approximately 8 hours, a much shorter time compared to other non-VR tool such as [119] which needed about 51 hours for each sequence. Moreover, we could obtain the 2D semantic segmentation associated with the 1000 RGB input images in a few minutes of rendering. Figure 4.7 reports qualitative results dealing with our 3D labeling and examples of projected 3D labels.

4.3 Conclusions and Future Works

We have proposed the first 3D semantic labeling tool based on Virtual Reality (VR). Our tool exploits VR alongside with gamification to ease and expedite

[4]Kitti Sequence `2011_09_30_drive_0020_sync`

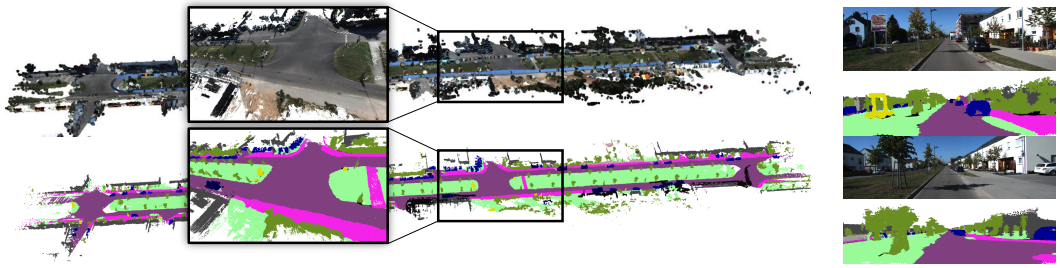

FIGURE 4.7: 3D and 2D Labeling from a Kitti sequence. Top Left: RGB point cloud. Bottom Left: labeled point cloud obtained by using our tool. Right: RGB images and projected semantic labels

3D semantic labeling of large scale scenarios and enlarge the pool of possible annotators to people without any knowledge about 3D modeling. The tool works with the most popular 3D data structures, such as meshes and point clouds. Moreover, we have shown how to integrate results from multiple users in order to achieve an overall better performance as well as uncertainty map of the labeling process. We have also demonstrated how to integrate the uncertainty map in the labeling process in order to further improve the results.

We argue that the label uncertainty information may also be leveraged while training deep neural networks for semantic segmentation, e.g. so as to weight the labels in the loss based on their associate uncertainty, as proposed in some recent works dealing with stereo vision [170]. Moreover, availability of uncertainty maps may foster the design of novel performance evaluation metrics which would take into account the uncertainty of labels.

Chapter 5

Image to Image Translation for Synthetic to Real Adaptation

Many recent works [21, 171, 24, 25] have proposed to deploy synthetic training images generated by state-of-the-art computer graphics techniques to obtain for free, during the rendering process, different kinds of annotations. Yet, such synthetic training samples turn out significantly different from the real images processed at test time, which implies a well-known issue, referred to in the machine learning literature as *domain shift*.

Promising works like [130, 129, 127] try to learn models which extract the same kind of features across the two domains. While this strategy seems successful for tasks like classification, it does not scale to dense *structured domain adaptation* [172] where the improvement gained by feature alignment is still modest. Alternatively, [145, 173] work directly on the training data trying to shrink the gap between synthetic and real images by transforming the first to make them look real using image-to-image generative adversarial networks. However, since they do not enforce any kind of constraint on the geometric consistency between input and output, these approaches can easily produce artifacts and distortions. Beside harming the realism of the generated images, artifacts could easily render annotations created for the synthetic images useless, especially for pixel-level labeling task where even a few pixels shift may invalidate the annotation.

In this chapter we propose a novel approach based on image-to-image domain translation by GANs while explicitly training the system to keep the semantic structure of the scene. The intuition behind our formulation is that forcing the *generator* network to keep the semantic structure of the image acts as a regularizer enforcing overall consistency of image appearance and producing images that look more realistic and exhibit less artifacts. For example, according to our formulation a "tree" can change its appearance but it should still be recognizable as a "tree" across domains. To enforce

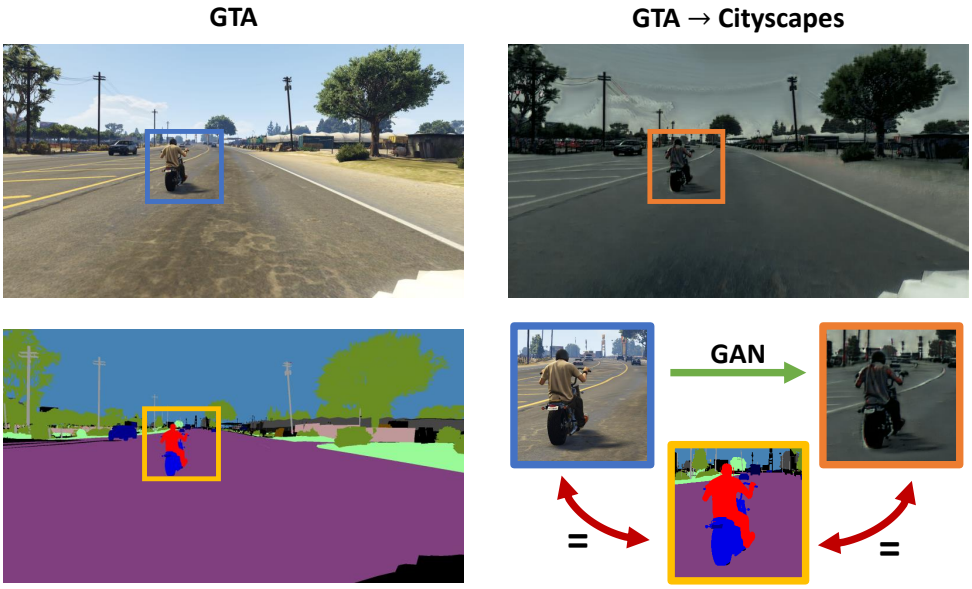

FIGURE 5.1: On the right an image generated applying our semantically aware GAN on a synthetic image from the GTA dataset [21] (left) to make the latter look more realistic. Lower right corner: zoomed crops to highlight how our semantically aware GAN can transform images across domains preserving the semantic structure of the scene.

the semantic constraint we train a *discriminator* network not only to classify the domain (real/fake) but also to solve the task of semantic segmentation on the synthetic domain (i.e. , we do not need labels in the real target domain). Moreover, we introduce an appearance reconstruction loss to further regularize the generation process and help preserving small details. To asses upon effectiveness of our proposal we transform synthetic images obtained from the synthetic GTA datasets [21] to look similar to the real images of the Cityscapes [14] dataset. Figure 5.1 shows on the right column a qualitative example generated by our method using as input the corresponding synthetic images depicted on the left. We will show how those images can be used to train a model to solve the problem of semantic segmentation yielding promising result with respect to the use of synthetic images.

5.1 Proposed Method

In this section we present our proposal for domain adaptation exploiting semantic information. We consider the problem of unsupervised and unpaired pixel-level domain adaptation from a source to a target domain. We define as X_s , Y_s the provided source data and associated semantic labels whilst as X_t the provided target data, but without any available target labels. Our goal is

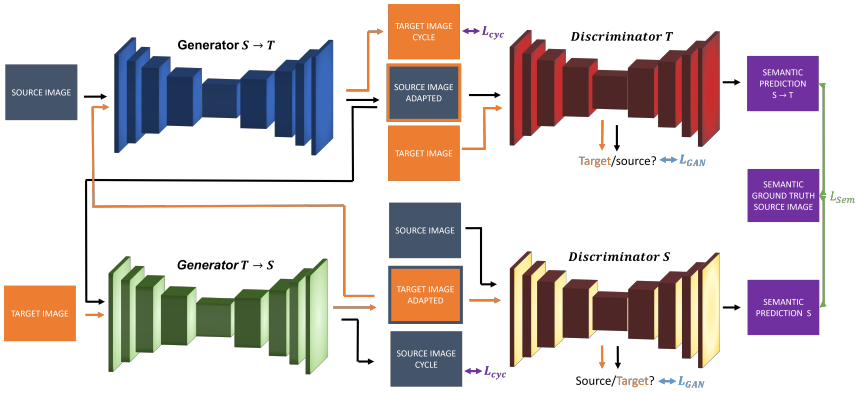

FIGURE 5.2: Schematic representations of the proposed network architecture. In dark blue and orange images from the source domain and target domain respectively. Dual color framed images are obtained by our adaptation method. In purple the use of the semantic maps.

to transform source images so to resemble target images while maintaining the semantic content of the scene during the generation process. A schematic representation of our method is shown in Figure 5.2.

5.1.1 Architecture

Inspired by [149], we adopt a cycle architecture consisting of two generators and two semantic discriminators. The first generator, $G_{S \rightarrow T}$, introduce a mapping from the source to the target domain and produces target samples which should deceive the discriminator D_T. The discriminator D_T, instead, learns to distinguish between adapted source and true target samples. On the other hand, the second generator, $G_{T \rightarrow S}$, learns the opposite mapping from source to target data, while the second discriminator D_s distinguish between adapted target and true source samples. Furthermore, peculiarly to our work, both *semantic* discriminators act not only as classifiers but also as semantic segmentation networks. Thus, we add a second decoder to D_T and D_S obtaining $D_{S_{sem}}$ and $D_{T_{sem}}$. The features extracted by the last encoder layer of the discriminators are used to generate both the semantic map and the domain classification score.

5.1.2 Training

We train our system to minimize multiple losses:

Adversarial Loss

We apply adversarial losses [174] to both mapping functions $S \rightarrow T$ and $T \rightarrow S$. To be concise, we define here only source to target adversarial loss, being equivalent to its inverse.

$$\mathcal{L}_{adv} = \mathbb{E}_{x_t \sim X_T}[log(D_t(x_t))] \tag{5.1}$$

$$\mathbb{E}_{x_s \sim X_s}[log(1 - D_t(G_{s \rightarrow t}(x_s)))] \tag{5.2}$$

$G_{S \rightarrow T}$ tries to generate images that look similar to images from domain T while D_T tries to distinguish between adapted source samples $G_{S \rightarrow T}(x_S)$ and real target samples X_T. $G_{S \rightarrow T}$ seek to minimize this objective against D_T which instead tries to maximize it.

Semantic Discriminator Loss

We train both discriminators, $D_{S_{sem}}$ and $D_{T_{sem}}$, to perform semantic segmentation employing source labels. $D_{T_{sem}}$ will be trained on adapted source images, while $D_{S_{sem}}$ will be trained directly on source images. We used a pixel-wise cross entropy loss $H(p, q)$ as in standard segmentation networks:

$$\mathcal{L}_{sem} = H(D_{S_{sem}}(G_{S \rightarrow T}(X_S)), Y_S) + H(D_{S_{sem}}(X_S), Y_S) \tag{5.3}$$

Weighted Reconstruction Loss

We exploit the cyclic L1 reconstruction loss proposed in [149] for target samples where we do not have any label. Regarding source samples, we weight each pixel proportionally to the probability of not belonging to its semantic class. Our weighting term acts as a regularization where the network usually fail adaptation introducing artifacts, forcing the least frequent classes to be reconstructed preserving input appearance:

$$\mathcal{L}_{rec} = ||G_{S \rightarrow T}(G_{T \rightarrow S}(x_T))) - x_T||_1 \tag{5.4}$$

$$(1 - w)||G_{T \rightarrow S}(G_{S \rightarrow T}(x_S))) - x_S||_1 \tag{5.5}$$

w is a weight mask with the same resolution of the source image. Defined C as the set of possible classes, each weight $w_{i,j}$ represents the likelihood of a class among all synthetic dataset:

$$w_{i,j} = \frac{n_{pixel} \in c}{n_{pixel}}, c \in C \tag{5.6}$$

.

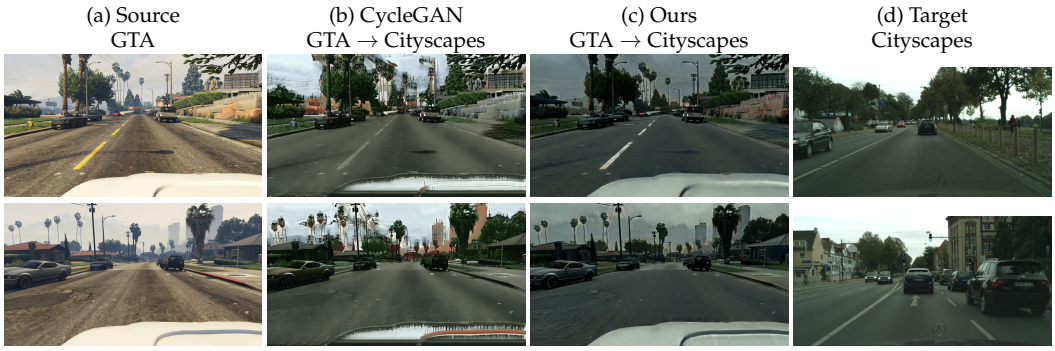

FIGURE 5.3: Image generated by CycleGan [149](b) and our semantics-aware GAN (c) for the GTA to Cityscapes domain alignement task

Final Loss

We train our discriminators and generators to minimize the following losses:

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{sem}\mathcal{L}_{sem} \tag{5.7}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{sem}\mathcal{L}_{sem} + \lambda_{rec}\mathcal{L}_{rec} \tag{5.8}$$

λ_{sem} and λ_{rec} are hyper-parameters that control the relative importance of domain classification, weighted reconstruction and semantic segmentation. Across all our experiments we will use λ_{sem} = 1 and λ_{rec}= 3.

5.2 Experimental Results

We conduct a series of tests to assess the effectiveness of our method in producing realistic images and verify if they are suitable for training deep learning models.

5.2.1 Datasets Creation

We have used as synthetic source domain the GTA dataset [21], that features 22K realistic synthetic images obtained from the Grand Theft Auto videogame enriched with perfect pixel level annotations for semantic segmentation. As target real images we have used the Cityscapes dataset [14] featuring 5000 images acquired during real driving sessions around Germany and annotated with precise pixel level labels for semantic segmentation. Among all available images we have used the *training* split as our target samples during training, while we have kept the *validation* split to measure performance of different semantic segmentation networks. We did not

use the *test* split since the labels are not publicly available. We chose these two datasets as they provide annotations for the same set of semantic classes and feature domains where the biggest difference concern the shift from synthetic to real images. We used ResNet as our generator networks and U-Net [30] as our discriminator. Using the loss formulation described in subsection 5.1.2 we have trained our GAN to transform images from the GTA [21] to the Cityscapes [14] domain for 300k steps using Adam as optimizer, 0.0001 for learning rate and batch size 2. We cropped our input images to 512x512. During the training process we have used images and labels from GTA and only images from Cityscapes, i.e. , our method does not require annotations from the real/target domain but only from the source one. Once trained, we used the generator to transform synthetic images from the training dataset to produce an *aligned* GTA dataset that should resemble images from the real Cityscapes domain. On Figure 5.3 we depict some qualitative example of images produced by our GAN (column (c)) together with the corresponding input from the GTA dataset (column (a)) and some exemplar images from the Cityscapes, target, dataset (column (d)). To better show the effectiveness of our semantic aware GAN, we also report images obtained by training a CycleGAN network [149] that does not use any semantic clues at training time (column (b)). By comparing our images (column (c)) with those produced by CycleGAN (column (b)) it turns out clearly that, unlike previous approaches (i.e. , column (b)), our novel formulation can preserve the semantic content and avoid introduction of artifacts. Moreover, the introduction of semantic constraints during the training process helps to produce sharper edges in the final image, which increases the quality of the images compared to CylceGAN. We have also applied our GANs to entire video sequences from the GTA domain and verified that the network can easily maintain temporal consistency even if it has only been trained on individual frames. [1]

5.2.2 Semantic Segmentation

Figure 5.3 shows how our network can produce visually appealing images. In the following we demonstrate that our adapted images can be used to train a neural network to obtain much better performance on the target domain w.r.t the corresponding synthetic ones.

Focusing on semantic segmentation, we have trained a standard FCN8s [28] on the original GTA synthetic images and on our *aligned* dataset. We

[1] https://youtu.be/wIpFcKLviYQ

Method	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU	Acc.
Source [133]	31.9	18.9	47.7	7.40	3.10	16.0	10.4	1.00	76.5	13.0	58.9	36.0	1.00	67.1	9.50	3.70	0.00	0.00	0.00	21.2	-
[133]	70.4	32.4	62.1	14.9	5.40	10.9	14.2	2.70	79.2	21.3	64.6	44.1	4.20	70.4	8.00	7.30	0.00	3.50	0.00	27.1	-
Source [135]	18.1	6.80	64.1	7.30	8.70	21.0	14.9	16.8	45.9	2.40	64.4	41.6	17.5	55.3	8.40	5.0	6.90	4.30	13.8	22.3	-
[135]	74.9	22.0	71.7	6.00	11.9	8.40	16.3	11.1	75.7	13.3	66.5	38.0	9.30	55.2	**18.8**	18.9	0.00	**16.8**	**16.6**	28.9	-
Source [175]	26.0	14.9	65.1	5.50	12.9	8.90	6.00	2.50	70.0	2.90	47.0	24.5	0.0	40.0	12.1	1.50	0.0	0.0	0.0	17.9	54.0
[175]	83.5	**38.3**	76.4	20.6	**16.5**	22.2	**26.2**	**21.9**	80.4	**28.7**	65.7	49.4	4.2	74.6	16.0	**26.6**	2.0	8.0	0.0	**34.8**	82.8
Source Ours	43.3	11.9	54.3	3.42	11.96	9.63	10.74	5.23	68.3	6.39	46.84	30.02	2.07	33.1	7.72	0.00	0.00	0.00	0.00	18.2	60.4
Ours	**85.4**	32.8	**78.0**	**21.0**	9.35	**26.1**	18.0	8.71	**82.2**	22.1	**71.2**	**51.4**	**13.4**	**79.5**	16.0	13.5	**7.83**	10.1	0.03	34.2	**84.4**

TABLE 5.1: Comparison between domain adaptation methods for semantic segmentation on the Cityscapes validation set. Middle section reports mIoU score per class, final two columns aggregated performance across the whole dataset, best results highlighted in bold.

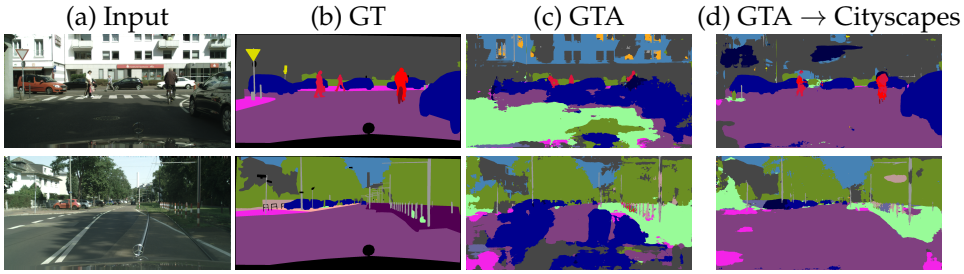

FIGURE 5.4: Segmentation results on the Cityscapes dataset for a FCN8s network trained only on synthetic data from the GTA dataset (c) and on our GTA *adapted* dataset (d).

tested both on the validation set of Cityscapes and reported the result in Table 5.1. For all our tests, we have initialized the feature extractor of the FCN8s with the publicly available VGG16 weights trained on the Imagenet dataset, then performed 100000 training iterations using batch 4, Adam optimizer and 0.0001 as learning rate. We trained networks on 1024x1024 cropped images.

To compare the networks we report two different metrics: the mean intersection-over-union (from now on shortened *mIoU*) computed following the guidelines of the Cityscapes benchmark [14] and the overall pixel accuracy (shortened *acc*), i.e. , the percentage of correctly predicted pixel labels. We also report detailed scores for each semantic class to highlight for which categories our image augmentation scheme is more effective. We compare the results obtained by our domain adaptation method with alternatives recently proposed in literature: the feature-level alignment method of [133], the curriculum style domain adaptation approach of [135] and the pixel level alignment introduced in [175]. In Table 5.1 for all methods we report the performance

achieved by training the very same FCN8s network [28] both before and after domain alignment, the former marked using *Source* in the method column. For each row we report per class *mIoU* and aggregated performance across the whole dataset (last two columns). Concerning aggregated *mIoU* score, we can see how our proposal can outperform both [135, 133] while being comparable with [175]. Moreover, considering pixel accuracy, our proposal compares favourably even to [175]. Considering the performance achieved before and after domain adaptation, our proposed pixel level alignment can provide an impressive +16.9 gain in *mIoU* and a +24 in *Acc.*, that, once again, compares favourably to [133, 135] and is comparable to [175]. Looking at class scores, we observe how our proposal can achieve the best absolute performances on 10 classes out of 19, including some key ones for autonomous driving like *road* (+42.1 gain between before and after alignment), car (+46.4) and *person* (+21.4). We still lose something compared to other proposals on less common classes (e.g. , *bus*, *motorcycle* and *bicycle*), we think that this might be due to the dataset used not having enough samples of the target classes to effectively teach to the generator how to realistically render them. Even though our proposal performs comparably to [175], we would like to stress out how our adaptation method can be trained end-to-end instead of relying on separate training steps for the different parts.

In Figure 5.4 we also report some qualitative examples of the improvement in segmentation attainable by training on our *adapted* GTA images (column (d)) compared to a purely synthetic training set (column (c)). Even if the results in column (d) are still far form optimal, most of the mistakes visible in column (c) are completely gone and the overall structure of the scene is more accurately segmented. Moreover, we can notice visually how the larger improvement concerns the segmentation of *road* (colored purple), *cars* (colored blue) and *people* (colored red).

5.2.3 Ablation Study

In subsection 5.2.2 we have proven that the images generated by our proposal can effectively be used to train a semantic segmentation network so as to nearly double its performance compared to using synthetic data only. In this section, instead, we investigate more in depth on how each component of our proposal contributes to the final result. Purposely, we trained different architectures, keeping the comparison as fair as possible by maintaining the same training protocol. We report the results of these tests in

Test	mIoU	Acc.
(a) Synthetic	18.23	60.43
(b) GAN+Sem.	29.45	78.13
(c) GAN+Sem+weight.	31.33	79.85
(d) Cycle [149]	29.43	79.20
(e) Cycle+sem+weight.	**34.27**	**84.48**

TABLE 5.2: Ablation study on the different component of our semantic aware GAN. Best results in bold.

Table 5.2. We first investigated the performance of training a semantic segmentation network on images adapted by a simple GAN[174]. As we obtained results even worse than our baseline network (a), we decided to not report them in Table 5.2. We then trained a GAN framework enriched with our semantic discriminator. Comparing line (b) with (a) we can clearly see how adding our semantics-aware discriminator not only allows to successfully train the GAN system but also results in a +11.22% *mIoU*, thus testifying how semantic information can successfully regularize training. We then added our weighted L1 reconstruction loss between source and adapted image (c) slightly improving performances by a +1.88% *mIoU*. We then trained a standard CycleGAN [149] with no semantic clue demonstrating how having two pairs of generator and discriminator is extremely effective to stabilize training of a GAN framework, as shown by (d) reaching comparable results to (b). Finally (e) reports the performance achievable by our full proposal that deploys the CycleGAN network combined with the semantics-aware discriminator and our semantic weighting system, achieving remarkable performance: +16.04% *mIoU* and +24.05% *Acc.* with respect to our baseline(a). We argue that, our novel network structure and loss function allow us to produce realistic images with adversarial training, and at the same time preserve structural coherence between input and output thanks to the enforced semantic consistency leading to good domain adaptation performance.

Chapter 6

Final Remarks

In this part of the thesis, we have investigated the lack of data for training deep neural networks. Since manual annotation is a tedious, time-consuming, and not scalable job, we investigated strategies to address this problem, particularly for semantic segmentation.

In chapter 4 we have proposed a novel tool to efficiently gather 3D and 2D semantic labels by exploiting virtual reality. We showed that our tool is practical and easy to use, enlarging the user pool that can utilize it. Moreover, our tool employs a gamification strategy to ameliorate the labeling experience.

In chapter 5 we have addressed another possible solution: avoid manual annotation and using computer graphics simulations. We have have proposed a novel pixel-level domain adaptation technique to mitigate the domain-shift problem between real and synthetic data. Notably, we have employed a semantically aware image-to-image translation network to shrink the gap between synthetic and real data. Our novel network structure and loss function can successfully produce realistic images thanks to its adversarial component. Moreover, we can preserve structural coherence between input and output thanks to the enforced semantic consistency. We have tested our proposal for domain adaptation from synthetic to real images in the context of semantic segmentation. Nevertheless, we could use the same process to address different tasks, e.g. object detection, or different type of domain shifts, e.g. different seasons, different sensors, or different weather conditions.

Overall, we have shown that computer graphics is a powerful tool that can be used to gather annotation efficiently. In the following part of the thesis, we will deepen this strategy but from a different perspective. In particular, we aim at mitigating the domain-shift using the knowledge coming from synthetic data annotated for different tasks. We will show that a model aware of more tasks will be more robust across domains, more accurate and

memory efficient as well as computationally faster than a model trained only on a single task.

Part II

Transferring Knowledge Across Tasks for Domain Adaptation

Chapter 7

Initial Remarks

Deep learning has revolutionized computer vision research and set forth a general framework to address various visual tasks (e.g. , classification, depth estimation, semantic segmentation, ...). A common framework suggests a close relationship between different tasks that should be exploitable to alleviate the dependence on massive labeled training sets. Unfortunately, most state-of-the-art methods ignore these connections and instead focus on a single task by solving it in isolation through supervised learning on a specific domain (i.e. , dataset). Should the domain or task change, standard practice would require acquiring a new annotated training set followed by retraining or fine-tuning the model.

However, any deep learning practitioner can testify that the effort to annotate a dataset is usually quite substantial and does vary significantly across tasks, potentially requiring ad-hoc acquisition modalities. Hence, the question we try to answer is: *would it be possible to deploy the relationships between tasks to remove the dependence for labeled data on new domains?*

A partial answer to this question has been provided by [176], which formalizes the relationships between tasks within a specific domain into a graph referred to as *Taskonomy*. This knowledge can be used to improve performance within a fully supervised learning scenario, though it is not clear how well it may generalize to new domains and to which extent it may be deployed in a partially supervised scenario (i.e. , supervision on only some tasks/domains). The generalization to new domains is addressed in the domain adaptation literature [123], that, however, works under the assumption of solving a single task in isolation, therefore ignoring potential benefits from related tasks.

We fuse the two worlds by explicitly addressing a cross-domain and cross-task problem where on one domain (e.g. , synthetic data) we have annotations for many tasks, while in the other (e.g. , real data) annotations are available only for a specific task, though we wish to solve many.

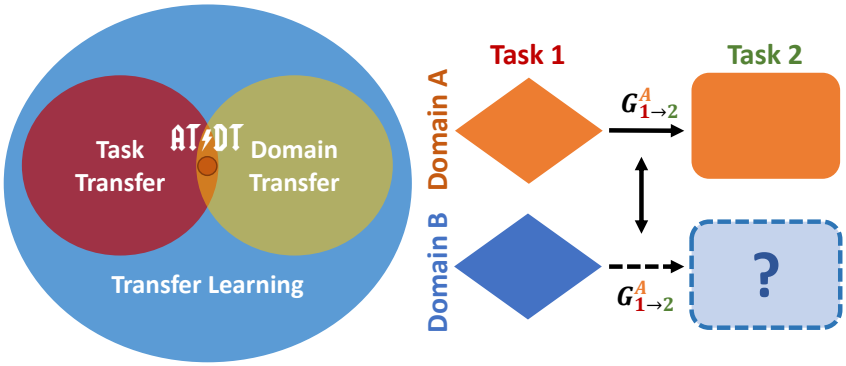

FIGURE 7.1: Our AT/DT framework transfers knowledge across tasks and domains. Given two tasks (1 and 2) and two domains (A and B), with supervision for both tasks in A but only for one task in B, we learn the dependency between tasks in A and exploit this knowledge in B to solve task 2 without the need of supervision.

Purposely, in this part of the thesis we will show a new 'Across Tasks and Domains Transfer framework' (shortened as AT/DT) which learns in a specific domain a function $G_{1\to 2}$ to transfer knowledge between a pair of tasks. We will see that, after the training phase, the same transfer function can be applied in a new domain to solve the second task while relying on supervision only for the first. A schematic representation of AT/DT is pictured in Figure 7.1.

In chapter 8 we will describe the main aspects of AT/DT, and we will prove its effectiveness on a challenging autonomous driving scenario where we address the two related tasks of depth estimation, and semantic segmentation [1]. We will show that our framework allows the use of fully supervised synthetic datasets (i.e. , Synthia [177], and Carla [25]) to drastically boost performance on partially supervised real data (i.e. , Cityscapes [14] and Kitti [108, 178]). Finally, we will also demonstrate how AT/DT is robust to sub-optimal scenarios where we use only a few annotated real samples or noisy supervision by proxy labels [179, 180, 170].

Then, in chapter 9, we will extend this framework, improving the transferability of learned features across tasks and domains in case of structured tasks such as semantic segmentation and depth estimation. We leverage two intuitions: 1. The transfer function still suffer from a domain-shift problem. Thus, we reduce the domain misalignment at the input level of the transfer network. 2. Some tasks can contain only partial information about the structure of the scene. Thus, we propose to train on an auxiliary task to enrich feature representation.

Finally, in chapter 10 we will show an application of AT/DT in the Unsupervised Domain Adaptation (UDA) benchmark for semantic segmentation (i.e. the same of previous chapter). In particular, we will highlight how plugging the depth information with AT/DT into any existing UDA method leads to superior performances.

In the following section we report some works relevant for this part of the thesis.

7.1 Transfer Learning

The existence of related representation within CNNs trained for different tasks has been highlighted since early works in the field [181]. These early findings have motivated the use of transfer learning strategy to bootstrap learning across related tasks. For example, object detection networks are typically initialized with ImageNet weights [182, 183, 184], although [185] has recently challenged this paradigm. Recently Zamir et. al. [176] have tried to formalize and deploy the idea of reusing information across training processes by proposing a computational approach to establish relationships among visual tasks represented in a taxonomy. Pal et. al. [186], propose to use similar knowledge alongside meta-learning to learn how to perform a new task within a zero-shot scenario. Both [176] and [186] assume a shared domain across the addressed task, while we directly target a cross-domain scenario. Moreover, [176] assumes full supervision to be available for all tasks while [186] zero supervision for the target task. Differently, AT/DT leverages full supervision for all tasks in one domain and only partial supervision in a different (target) domain.

7.1.1 Task and Domain Adaptation

Adapting the knowledge across tasks or domains are sub-cases of the general transfer learning literature. Most existing approaches address either task adaptation or domain adaptation independently. Nevertheless, a few works have proposed to tackle these two problems jointly. [134, 187] were the first papers to propose a cross-tasks and cross-domains adaptation approach, considering as tasks different image classification problems. On the contrary, in AT/DT we expand the concept of task adaptation by considering diverse visual tasks as done in [176].

Chapter 8

Learning Across Tasks and Domains

8.1 Across Task and Domain Transfer Framework

We wish to start with a practical example of the problem we are trying to solve and how we address it. Let us consider a synthetic and a real domain where we aim to solve the semantic segmentation task. Annotations come for free in the synthetic domain while are rather expensive in the real one. Domain adaptation comes handy for this; however, we wish to go one step further. May we pick a closely related task (e.g. , depth estimation) where annotations are available in both domains and use it to boost the performance of semantic segmentation on real data? To achieve this goal we train deep networks for depth and semantic segmentation on the synthetic domain and learn a mapping function to transform deep features suitable for depth estimation into deep features suitable for semantic segmentation. Then we apply the same mapping function on samples from the real domain to obtain a semantic segmentation model without the need of semantic labels in the real domain. In the remainder of this section, we formalize the AT/DT framework.

8.1.1 Common Notation

We denote with $\mathcal{T}_j$ a generic visual task defined as in [176]. Let us assume $\mathcal{X}^k$ to be the set of samples (i.e. , images) belonging to domain k and $\mathcal{Y}_j^k$ to be the paired set of annotations for task $\mathcal{T}_j$. In our problem we assumes to have two domains, $\mathcal{A}$ and $\mathcal{B}$, and two tasks, $\mathcal{T}_1$ and $\mathcal{T}_2$. For the two tasks we have complete supervision in $\mathcal{A}$, i.e. , $\mathcal{Y}_1^A$ and $\mathcal{Y}_2^A$, but labels only for $\mathcal{T}_1$ in $\mathcal{B}$, i.e. $\mathcal{Y}_1^B$. We assume each task $\mathcal{T}_j$ to be solvable by a deep neural network N_j, consisting in a feature encoder E_j and a feature decoder D_j, such that

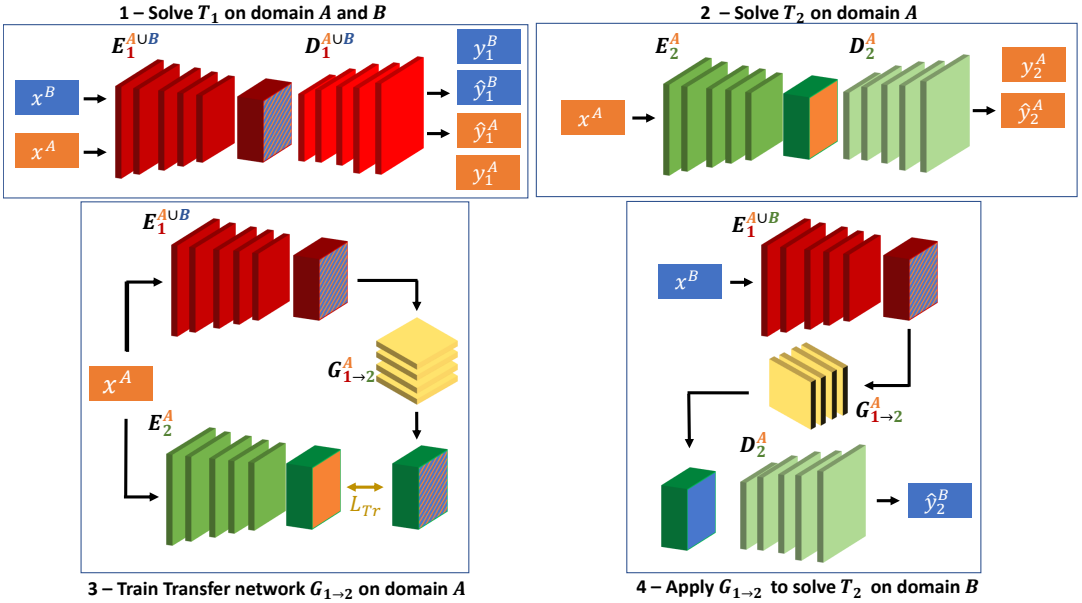

FIGURE 8.1: Overview of the AT/DT framework. (1) We train network $N_1^{A\cup B}$ to solve $\mathcal{T}_1$ (red) with supervision in domain $\mathcal{A}$ (orange) and $\mathcal{B}$ (blue) to obtain a shared feature representation across domains, highlighted by blue and orange strips. (2) We train a network N_2^A to solve $\mathcal{T}_2$ (green) on $\mathcal{A}$ where labels are available. (3) We learn a network $G_{1\to2}$ that transform features from $\mathcal{T}_1$ to $\mathcal{T}_2$ on samples from $\mathcal{A}$. (4) We apply the transfer network on $\mathcal{B}$ to solve $\mathcal{T}_2$ without the need for annotations.

$\hat{y}_j = N_j(x) = D_j(E_j(x))$. The network is trained on domain k by minimizing a task-specific loss on annotated samples $(x^k, y_j^k) \sim (\mathcal{X}^k, \mathcal{Y}_j^k)$. The result of this training is a network trained to solve $\mathcal{T}_j$ using samples from $\mathcal{X}^k$ that we denote as N_j^k.

8.1.2 Overview

Our work builds on the intuition that if two tasks are related there should be a function $G_{1\to2} : \mathcal{T}_1 \to \mathcal{T}_2$ that transfer knowledge among them. But what does transferring knowledge actually means? We will show that this abstract concept can be implemented by transferring representations in deep feature spaces. We propose to first train two task specific networks, N_1 and N_2, then approximate function $G_{1\to2}$ by a deep neural network that transforms features extracted by N_1 into corresponding features extracted by N_2 (i.e. , $G_{1\to2} : E_1(x) \to E_2(x)$). We train $G_{1\to2}$ by minimizing a reconstruction loss on $\mathcal{A}$, where we have complete supervision for both tasks, and use it on $\mathcal{B}$ to solve $\mathcal{T}_2$ having supervision only for $\mathcal{T}_1$.

Our method can be summarized into the four steps pictured in Figure 8.1 and detailed in the following sections:

1. Learn to solve task $\mathcal{T}_1$ on domains $\mathcal{A}$ and $\mathcal{B}$.
2. Learn to solve task $\mathcal{T}_2$ on domain $\mathcal{A}$.
3. Train $G_{1\to2}$ on domain $\mathcal{A}$.
4. Apply $G_{1\to2}$ to solve $\mathcal{T}_2$ on domain $\mathcal{B}$.

8.1.3 Solve $\mathcal{T}_1$ on $\mathcal{A}$ and $\mathcal{B}$

A network N_1 can be trained to solve task $\mathcal{T}_1$ on domain $\mathcal{X}^k$ by minimizing a task specific supervised loss

$$L_{\mathcal{T}_1}(\hat{y}_1^k, y_1^k);\ \hat{y}_1^k = N_1(x^k). \tag{8.1}$$

However, training one network for each domain would likely result in disjoint feature spaces; we, instead, wish to have similar representation to ease generalization of $G_{1\to2}$ across domains. Therefore, we train a single network, $N_1^{A\cup B}$, on samples from both domains, i.e. , $\mathcal{X}^k = \mathcal{X}^A \cup \mathcal{X}^B$. Having a common representation ease the learning of a task transfer mapping valid on both domains though training it only on $\mathcal{A}$. More details on the impact of having common or disjoint networks are reported in subsection 8.4.2.

8.1.4 Solve $\mathcal{T}_2$ on $\mathcal{A}$

Now we wish to train a network to solve $\mathcal{T}_2$, however, for this task we can only rely on annotated samples from $\mathcal{A}$. The best we can do is to train a N_2^A minimizing a supervised loss

$$L_{\mathcal{T}_2}(\hat{y}_2^A, y_2^A);\ \hat{y}_2^A = N_2(x^A). \tag{8.2}$$

8.1.5 Train $G_{1\to2}$ on $\mathcal{A}$

We are now ready to train a task transfer network $G_{1\to2}$ that should learn to remap deep features suitable for $\mathcal{T}_1$ into good representations suitable for $\mathcal{T}_2$. Given $N_1^{A\cup B}$ and N_2^A we generate a training set with pairs of features $(E_1^{A\cup B}(x^A), E_2(x^A))$ obtained feeding the same input x^A to $N_1^{A\cup B}$ and N_2^A. We use only samples from $\mathcal{A}$ for the training set as it is the only domain

where we are reasonably sure that the two networks perform well. We optimize the parameters of $G_{1\to 2}$ by minimizing the reconstruction error between transformed and target features

$$L_{Tr} = ||G_{1\to 2}(E_1^{A\cup B}(x^A)) - E_2^A(x^A)))||_2, \tag{8.3}$$

At the end of the training $G_{1\to 2}$ should have learned how to remap deep features from one space into the other.

Among all the possible splits (E, D) obtained cutting N at different layers, we select as input for $G_{1\to 2}$ the deepest features, i.e. , those at the lowest spatial resolution. We make this choice because deeper features tend to be less connected to a specific domain and more correlated to higher level concepts. Therefore, by learning a mapping at this level we hope to suffer less from domain shift when applying $G_{1\to 2}$ on samples from $\mathcal{B}$. A more in depth discussion on the choice of E is reported in subsection 8.4.1.

8.1.6 Apply $G_{1\to 2}$ to solve $\mathcal{T}_2$ on $\mathcal{B}$

Now we aim to solve $\mathcal{T}_2$ on $\mathcal{B}$. We can use the supervision provided for $\mathcal{T}_1$ on $\mathcal{B}$ to extract good image features (i.e. , $E_1^{A\cup B}(x_B)$). Then use $G_{1\to 2}$ to transform these features into good features for $\mathcal{T}_2$, and finally decode them through a suitable decoder D_2^A. The whole system at inference time corresponds to:

$$\hat{y}_2^B = D_2^A(G_{1\to 2}^A(E_1^{A\cup B}(x_B))) \tag{8.4}$$

Thus, thanks to our novel formulation, we can learn through supervision the dependencies between two tasks in a source domain and leverage on them to perform one of the two tasks in a different target domain where annotations are not available.

8.2 Experimental Settings

We describe here the experimental choices made when testing AT/DT.

Tasks. To validate the effectiveness of AT/DT, we select as $\mathcal{T}_1$ and $\mathcal{T}_2$ *semantic segmentation* and *monocular depth estimation*. In subsection 8.4.7, we report some promising results also using as $\mathcal{T}_1$ semantic segmentation and as $\mathcal{T}_2$ normal estimation. We minimize a cross entropy loss to train a network for semantic segmentation while we use a L_1 regression loss to train a network for monocular depth estimation. We choose these tasks to evaluate our

framework since they are closely related, as highlighted in recent works [1, 188, 189], and of clear interest in many practical settings such as, e.g. , autonomous driving. Moreover, as they require a structured output, they can be addressed by a similar network architecture with the only difference being the number of filters in the final layer: as many as the number of classes for semantic segmentation, just one for depth estimation and three in case of normal estimation.

Datasets. We consider four different datasets, two synthetic ones, and two real ones. We pick synthetic datasets as $\mathcal{A}$ to learn the mapping across tasks thanks to availability of free annotations. We use real dataset as $\mathcal{B}$ to benchmark the performance of AT/DT in challenging realistic conditions. As synthetic datasets we have used the six video sequences of the Synthia-SF dataset [177] (shortened as Synthia) and rendered several other sequences with the Carla simulator [25]. For both datasets, we have split the data into a train, validation, and test set by subdividing them at the sequence level (i.e. , we have used different sequences for train, validation, and test). As for the real datasets, we have used images from the Kitti [190, 108, 178] and Cityscapes [14] benchmarks. Concerning Kitti, we have used the 200 images from the Kitti 2012 training set [190] to benchmark depth estimation and 200 images from the Kitti 2015 training set with semantic annotations recently released in [191]. As for Cityscapes, we have used the validation split to benchmark semantic segmentation and all the images in the training split. When training depth estimation networks on Cityscapes, following a procedure similar to [170] we generate proxy labels by filtering SGM [192] disparities through confidence measures (left-right check).

Network Architecture. Each task network is implemented as a dilated ResNet50 [193] that compresses an image to 1/16 of the input resolution to extract features. Then we use several bilinear up-sample and convolutional layers to regain resolution and get to the final prediction layer. All the layers of the network feature batch normalization. We implement the task transfer network ($G_{1\to 2}$) as a simple stack of convolutional and deconvolutional layers that reduce the input to 1/4 of the input resolution before getting back to the original scale.

Evaluation Protocol. For each test we select two domains (i.e. , two datasets, referred to as $\mathcal{A}$ and $\mathcal{B}$) and one direction of task transfer, e.g. , from $\mathcal{T}_1$ to $\mathcal{T}_2$. We will use $Sem. \to Dep.$ when mapping features from semantics to depth and $Dep. \to Sem.$ when switching the two tasks. For each configuration of datasets and tasks we use AT/DT to train a cross-task network

($G_{1\to 2}$) following the protocol described in subsection 8.1.2, then measure its performance for $\mathcal{T}_2$ on $\mathcal{B}$. We compare our method against a *Baseline* obtained training a network with supervision for $\mathcal{T}_2$ in $\mathcal{A}$ (i.e. , N_2^A) and testing it on $\mathcal{B}$. Moreover, we report as a reference the performance attainable by a *Oracle* (i.e. , a network trained with supervision on $\mathcal{B}$). We perform all the evaluation at the original image resolution for Cityscapes, Carla and Synthia. Instead, for Kitti, we consider a central crop with size 320×1216 due to the varying size of images.

Metrics. Our semantic segmentation networks predict eleven different classes corresponding to those available in the Carla simulator plus one additional class for 'Sky'. To measure performance, we report two different global metrics: pixel accuracy, shortened *Acc.* (i.e. , the percentage of pixels with a correct label) and Mean Intersection Over Union, shortened *mIoU* (computed as detailed in [14]). To provide more insights on per-class gains we also report the *IoU* (intersection-over-union) score computed independently for each class.

When testing the depth estimation task we use the standard metrics described in [51]: Absolute Relative Error (Abs Rel), Square Relative Error (Sq Rel), Root Mean Square Error (RMSE), logarithmic RMSE and three δ accuracy scores (δ_α being the percentage of predictions whose maximum between ratio and inverse ratio with respect to the ground truth is lower than 1.25^α).

8.3 Experimental Results

In this section we report some experimental results made to evaluate AT/DT.

8.3.1 Depth to Semantics

Following the protocol detailed in section 8.2, we first test AT/DT when transferring knowledge from the *monocular depth estimation* task to the *semantic segmentation* task, and report the results in Table 8.1. In this setup, we have supervision for both tasks in $\mathcal{A}$ while only for depth estimation in $\mathcal{B}$. Therefore, for each configuration, we report the results obtained performing semantic segmentation on $\mathcal{B}$ without any domain-specific supervision.

We begin our investigation by studying the task transfer in a purely synthetic environment, where we can have perfect annotations for all tasks and domains, i.e. , we use Synthia and Carla as $\mathcal{A}$ and $\mathcal{B}$, respectively. The results obtained by AT/DT and a transfer learning baseline are reported in

	$\mathcal{A}$	$\mathcal{B}$	Method	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
(a)	Synthia	Carla	Baseline	63.94	54.87	15.21	**0.03**	13.55	12.78	**52.73**	27.34	4.88	50.24	79.73	34.12	73.36
	Synthia	Carla	AT/DT	**73.57**	**62.58**	**26.85**	0.00	**17.79**	**37.30**	35.27	**52.94**	**17.76**	**62.99**	**87.50**	**43.14**	**80.00**
(b)	Synthia	Cityscapes	Baseline	6.91	0.68	0.00	0.00	2.47	9.14	**3.19**	**8.90**	**0.81**	25.93	26.86	7.72	28.49
	Synthia	Cityscapes	AT/DT	**85.77**	**29.40**	**1.23**	0.00	**3.72**	**14.55**	1.87	8.85	0.38	**42.79**	**67.06**	**23.24**	**64.03**
(c)	Carla	Cityscapes	Baseline	71.87	**36.53**	3.99	**6.66**	24.33	22.20	66.06	**48.12**	7.60	60.22	69.05	37.88	74.61
	Carla	Cityscapes	AT/DT	**76.44**	32.24	**4.75**	5.58	**24.49**	**24.95**	**68.98**	40.49	**10.78**	**69.38**	**78.19**	**39.66**	**76.37**
	-	Cityscapes	Oracle	95.65	77.72	33.02	37.63	65.45	42.087	89.36	89.99	41.36	86.81	89.22	68.02	93.56

TABLE 8.1: Experimental results of *Dep.* $\rightarrow$ *Sem.* scenario. Best results highlighted in bold.

Table 8.1-(a). Comparing the two rows we can clearly see that our method boost performance by *+9.02%* and *+6,64%*, for mIoU and Acc, respectively, thanks to the additional knowledge transferred from the depth estimation task.

The same performance boost holds when considering a far more challenging domain transfer between synthetic and real data, i.e. , Table 8.1-(b) (Synthia $\rightarrow$ Cityscapes) and Table 8.1-(c) (Carla $\rightarrow$ Cityscapes). In both scenarios, our AT/DT improves the two averaged metrics (mIoU and Acc.) and most of the per class scores, with gain as large as *+78,86%* for the Road class in (b). Overall AT/DT consistently improves predictions for the more interesting classes in autonomous driving scenarios, e.g. , Road, Person.... The main difficulties for AT/DT seems to deal with transferring knowledge for classes where depth estimation is particularly hard (e.g. , Vegetation, where synthetic data have far from optimal annotations, or thin structures like Poles and Fences). Indeed our model in Table 8.1-(c) is still far from the performance obtainable by the same *Oracle* network trained with supervision on $\mathcal{B}$ for $\mathcal{T}_2$. However we wish to point out that in this scenario we do not use any annotation at all on the real Cityscapes data, since we automatically generate noisy proxy labels for depth from synchronized stereo frames following [170].

The top row of Figure 8.2 show qualitative results on Cityscapes where AT/DT produces clearly better semantic maps than the baseline network.

8.3.2 Semantics to Depth

Following the protocol detailed in section 8.2, we test AT/DT when transferring features from *semantic segmentation* to *monocular depth estimation*. In this setup, we have complete supervision for both tasks in $\mathcal{A}$ and only for semantic segmentation in $\mathcal{B}$. For each configuration we report in Table 8.2

	$\mathcal{A}$	$\mathcal{B}$	Method	Lower is better				Higher is better		
				Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
(a)	Synthia	Carla	Baseline	0.632	8.922	13.464	0.664	0.323	0.578	0.733
	Synthia	Carla	AT/DT	**0.316**	**5.485**	**11.712**	**0.458**	**0.553**	**0.785**	**0.880**
(b)	Carla	Cityscapes	Baseline	0.667	13.500	16.875	0.593	0.276	0.566	0.770
	Carla	Cityscapes	AT/DT	**0.394**	**5.837**	**13.915**	**0.435**	**0.337**	**0.749**	**0.899**
	Cityscapes	Cityscapes	Oracle	0.176	3.116	9.645	0.256	0.781	0.921	0.969
(c)	Carla	Kitti	Baseline	0.500	10.602	10.772	0.487	0.384	0.723	0.853
	Carla	Kitti	AT/DT	**0.439**	**8.263**	**9.148**	**0.421**	**0.483**	**0.788**	**0.891**
	-	Kitti	Oracle	0.265	2.256	5.696	0.319	0.672	0.859	0.939

TABLE 8.2: Experimental results of *Sem.* $\rightarrow$ *Dep.* scenario. Best results highlighted in bold.

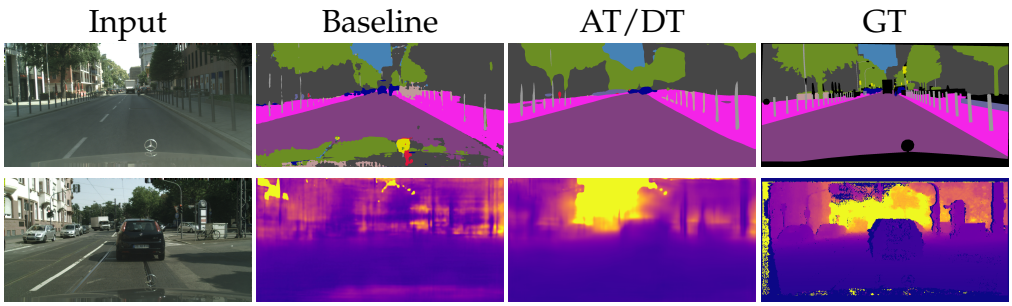

FIGURE 8.2: Qualitative results for $\mathcal{A}$: Carla to $\mathcal{B}$: Cityscapes. First row shows *Dep.* $\rightarrow$ *Sem.* scenario while second row shows *Sem.* $\rightarrow$ *Dep.* setting. From left to right RGB input, baseline predictions, AT/DT predictions, ground-truth images.

the results obtained performing monocular depth estimation on $\mathcal{B}$ without any domain-specific supervision.

The first pair of rows (i.e. , Table 8.2-(a)) reports results when transferring knowledge across two synthetic domains. The use of knowledge coming from semantic features helps AT/DT to predict better depths resulting in consistent improvements in all the seven metrics with respect to the baseline. The same gains hold for tests concerning real datasets (i.e. , Table 8.2-(b) with Cityscapes and Table 8.2-(c) with Kitti), where the deployment of AT/DT always results in a clear advantage against the baseline. We wish to point out how on Table 8.2-(c) we report a result where AT/DT use very few annotated samples from $\mathcal{B}$ (i.e. , only the 200 images annotated with semantic labels released by [191]). Comparing Table 8.2-(c) to Table 8.2-(b) we can see how the low data regime of Kitti results in slightly smaller gains, as also testified by the difference among oracle performances in the two datasets. Nevertheless AT/DT consistently yields improvements with respect to the baseline for all the seven metrics. We believe that these results provide some assurances on the effectiveness of AT/DT with respect to the amount of available data per task. Finally, the bottom row of Figure 8.2 shows qualitative results

	Method	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
(a)	Baseline	71.87	36.53	3.99	**6.66**	24.33	22.20	66.06	48.12	7.60	60.22	69.05	37.88	74.61
(b)	AT/DT	76.44	32.24	4.75	5.58	24.49	**24.95**	68.98	40.49	**10.78**	69.38	**78.19**	39.66	76.37
(c)	CycleGAN	81.58	39.15	**6.08**	5.31	**30.22**	21.73	**77.71**	50.00	8.33	68.35	77.22	42.33	80.93
(d)	AT/DT + CycleGAN	**85.19**	**41.37**	5.44	3.02	29.90	24.07	71.93	**58.09**	7.53	**70.90**	77.78	**43.20**	**81.92**

TABLE 8.3: Experimental results of integration with domain adaptation techniques. We show results of $\mathcal{A}$: Carla to $\mathcal{B}$: Cityscapes and *Dep.* $\rightarrow$ *Sem.* scenario. Best results highlighted in bold.

	Method	Lower is better				Higher is better		
		Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
(a)	Baseline	0.667	13.499	16.875	0.593	0.276	0.566	0.770
(b)	AT/DT	**0.394**	**5.837**	**13.915**	**0.435**	**0.337**	**0.749**	**0.899**
(c)	CycleGAN	0.943	27.026	21.666	0.695	0.218	0.478	0.690
(d)	AT/DT+CycleGAN	0.563	10.789	15.636	0.489	0.247	0.668	0.861

TABLE 8.4: Experimental results of comparison and integration with domain adaptation techniques. We show results of $\mathcal{A}$: Carla to $\mathcal{B}$: Cityscapes and *Sem.* $\rightarrow$ *Dep.* scenario. Best results highlighted in bold.

on monocular depth estimation on Cityscapes: we can clearly observe how AT/DT provides significant improvements over the baseline, especially on far objects.

8.3.3 Integration with Domain Adaptation

All the results of subsection 8.3.1 and subsection 8.3.2 are obtained learning a mapping function across tasks in a domain and deploying it in another one. Therefore, both the transfer network $G_{1\rightarrow 2}$ and the baseline we consider, can indeed suffer from domain shift issues. Fortunately, the domain adaptation literature provides several different strategies to overcome domain shifts that are complementary to our AT/DT. We provide here some preliminary results on how the two approaches may be combined together.

We consider a pixel-level domain adaptation technique, i.e. , CycleGan [149], that transforms images from $\mathcal{B}$ to render them more similar to those from $\mathcal{A}$. We train CycleGAN to transform images from Carla ($\mathcal{A}$) to Cityscapes ($\mathcal{B}$) and vice-versa. The network is trained using the original author implementation for 200k steps on random image crops of 400×400 pixels. We use the same hyper-parameters settings as proposed in the original paper.

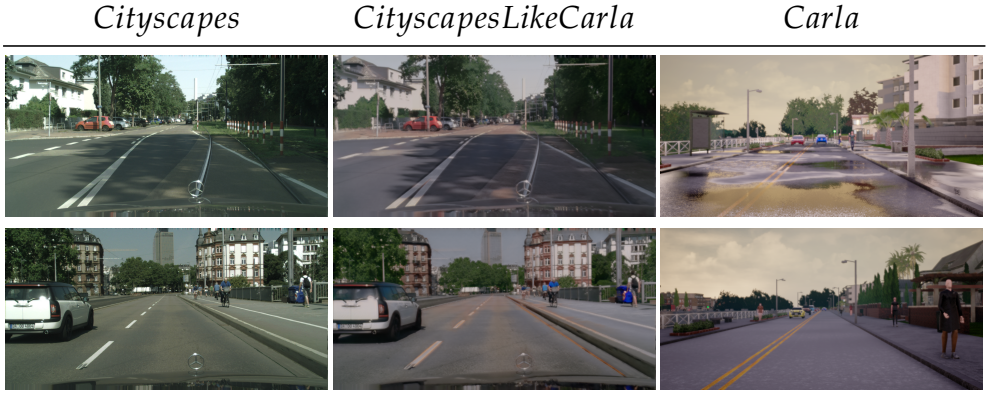

FIGURE 8.3: Images obtained applying CycleGAN to make Cityscapes samples similar to those of Carla. From left to right: samples from Cityscapes, corresponding image from *CityscapesLikeCarla* obtained by CycleGAN, similar samples from Carla

Once trained, we transform the Cityscapes dataset into the Carla style generating a new *CityscapesLikeCarla* dataset which we will call $\mathcal{B}like\mathcal{A}$ domain (see Figure 8.3). The baseline is then obtained by testing N_2^A with the validation set of $\mathcal{B}like\mathcal{A}$. To integrate AT/DT with CycleGAN, we train a $N_1^{A \cup \{BlikeA\}}$ on both $\mathcal{A}$ and $\mathcal{B}like\mathcal{A}$ at step 1 of AT/DT. Then, at step 4, to infer the predictions for $\mathcal{T}_2$ on $\mathcal{B}$, we employ the validation set of $\mathcal{B}like\mathcal{A}$ as done for the baseline. To summarize we train the shared source network on samples obtained from $\mathcal{A}$ and $\mathcal{B}like\mathcal{A}$, then we test all networks on the test set of $\mathcal{B}like\mathcal{A}$ (i.e. , Cityscapes images transformed to look like those from Carla).

In Table 8.3, we report results obtained for a *Dep.* $\rightarrow$ *Sem.* scenario using Carla as $\mathcal{A}$ and Cityscapes as $\mathcal{B}$. The pixel level domain alignment of CycleGAN (row (c)) proves particularly effective in this scenario, yielding a huge boost when compared to the baseline (row (a)), even greater then the gain granted by AT/DT (row (b)). However, we can see how the best average results (i.e. , mIoU and Acc.) can be obtained combining our cross task framework (AT/DT) with the pixel level domain adaptation provided by CycleGAN (row (d)). Considering the scores on single classes, instead, there is no clear winner among the four considered methods, with different algorithms providing higher accuracy for different classes. In Table 8.4 we report results obtained on a *Sem.* $\rightarrow$ *Dep.* scenario using the same pair of domains and the same four methods. Surprisingly, when targeting depth estimation CycleGAN (row (c)) is not as effective as before and actually worsen significantly the performance of the baseline (row (a)). Our AT/DT is instead

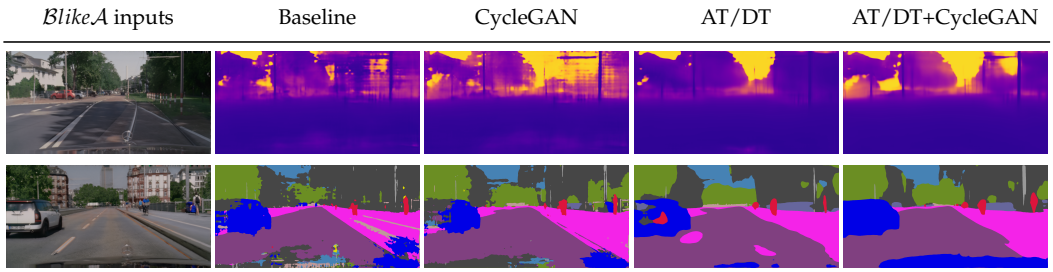

FIGURE 8.4: Qualitative results on the Cityscapes dataset in a *Sem.* $\rightarrow$ *Dep.* scenario (first row) and *Dep.* $\rightarrow$ *Sem.* scenario (second row). From left to right: $\mathcal{B}like\mathcal{A}$ inputs, predictions obtained by a transfer learning baseline, by a domain adaptation baseline (CycleGAN[149]), by our framework (AT/DT) and by our framework aided by domain adaptation (AT/DT+CycleGAN).

more robust to the task being addressed and in this scenario can improve the baseline when combined with CycleGAN (row (d)) and obtain the best overall results when applied alone (row (a)).

In Figure 8.4 we show some qualitative results obtained when combining AT/DT together with the pixel level domain adaptation obtained through CycleGAN. Comparing the results in the *Sem.* $\rightarrow$ *Dep.* scenario (first row) with those obtained in a *Dep.* $\rightarrow$ *Sem.* scenario (second row) we can see how CycleGAN is very effective when targeting the semantic segmentation tasks, much less effective when targeting a depth estimation task. AT/DT, instead, consistently produce better predictions than the baseline in both the considered tasks.

8.4 Additional Experiments

We report additional tests to shine light on some of the design choices made when developing AT/DT. Moreover, we show an experimental study focused on highlighting the importance of $G_{1\rightarrow 2}$, in particular comparing our proposal to an end-to-end multi-task network featuring a shared encoder and two task dependent decoders.

8.4.1 Study on the Transfer Level

For all the previous tests we split N between E and D at the layer corresponding to the lowest spatial resolution. We pick this split based on the intuition that deeper layers yield more abstract representations, thus less correlated

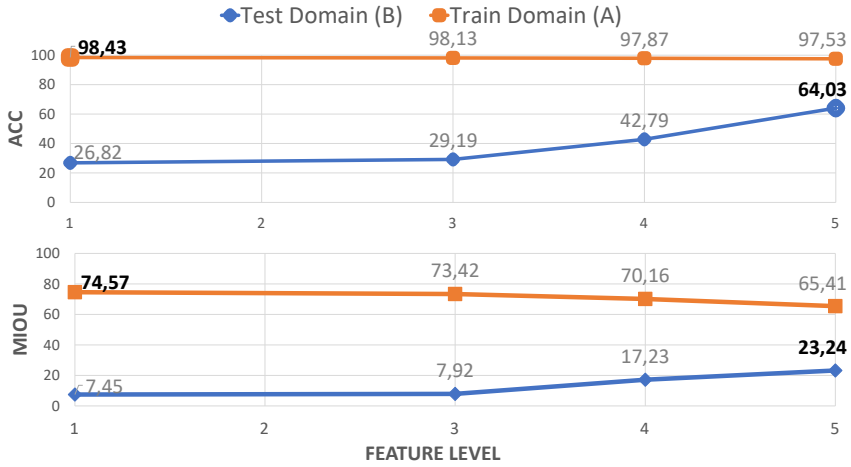

FIGURE 8.5: Study on feature level for task transfer from Synthia to Cityscapes and *Dep.* $\rightarrow$ *Sem.* scenario. Deeper levels correspond to higher generalization performances.

to specific domain information, while lower level features are more domain dependent. Therefore, learning $G_{1\rightarrow 2}$ between shallower layers should lead to less generalization ability across domains. To validate this intuition, we run experiments aimed at measuring performance for the *Dep.* $\rightarrow$ *Sem.* scenario (Synthia $\rightarrow$ Cityscapes) when varying the network layer at which we split N into E and D. We consider four different feature levels corresponding to residual blocks at increasing depth in ResNet50. For each of them we train a transfer network on domain $\mathcal{A}$ and then measure *mIoU* and *Acc.* testing on unseen images from $\mathcal{A}$ (i.e. $D_2^A(G_{1\rightarrow 2}^A(E_1^{AuB}(x_A)))$) and $\mathcal{B}$ (i.e. $D_2^A(G_{1\rightarrow 2}^A(E_1^{AuB}(x_B)))$). The results are plotted in Figure 8.5.

Considering *Acc.* (top plot) we can see how in-domain performance are almost equivalent at the different feature levels (orange line), while cross-domain performance increase when considering deeper feature levels (blue line). This pattern is even more pronounced when considering *mIoU* (bottom plot), where in-domain performance actually decreases alongside with deeper feature, whilst cross-domain performance increases. These results validate our intuition that deeper features are less domain specific and may lead to better generalization to unseen domains.

8.4.2 Shared vs Non-Shared N_1

Throughout this work we have always trained a single network for $\mathcal{T}_1$ with samples from $\mathcal{A}$ and $\mathcal{B}$. The rationale behind this choice is to have a single feature extractor for both domains such that $G_{1\rightarrow 2}$ trained only on samples

Shared	Domain	mIoU	Acc.
✗	$\mathcal{A}$	61.73	97.02
✓	$\mathcal{A}$	65.41	97.53
✗	$\mathcal{B}$	6.42	29.36
✓	$\mathcal{B}$	23.24 **(+16.82)**	64.03 **(+34.67)**

TABLE 8.5: Study on Shared vs Not-Shared $N_1^{A \cup B}$. We show a $\mathcal{A}$: Synthia to $\mathcal{B}$: Carla and *Dep.* $\rightarrow$ *Sem.* scenario. Performance improvement highlighted in bold.

Batchnorm	Domain	mIoU	Acc.
✗	$\mathcal{A}$	72.48	98.09
✓	$\mathcal{A}$	65.41	97.53
✗	$\mathcal{B}$	22.75	58.29
✓	$\mathcal{B}$	23.24 **(+0.49)**	64.03 **(+5.74)**

TABLE 8.6: Ablation Study on Batch Normalization. We show a $\mathcal{A}$: Synthia to $\mathcal{B}$: Cityscapes and *Dep.* $\rightarrow$ *Sem.* scenario. Performance improvement highlighted in bold.

from $\mathcal{A}$ would be able to generalize well to samples from $\mathcal{B}$ as they are sampled from a similar distribution.

Here we experimentally validate this intuition by comparing a shared $N_1^{A \cup B}$ against the use of two separate networks, one trained on samples from $\mathcal{A}$ (N_1^A) and the other with samples from $\mathcal{B}$ (N_1^B). We consider a *Dep.* $\rightarrow$ *Sem.* scenario where we use Synthia as domain $\mathcal{A}$ and Cityscapes as $\mathcal{B}$. In Table 8.5 we report the *mIoU* and *Acc.* achieved on unseen samples from the two domains. On the training domain $\mathcal{A}$ both methods are able to obtain good results, slightly better for the shared network, probably thanks to the higher variety of data used for training. However, when moving to the completely different domain $\mathcal{B}$, it is clear that maintaining the same feature extractor is of crucial importance to be able to use the same $G_{1 \rightarrow 2}$. This test suggests the interesting findings that feature extracted by the exact same network architecture trained for the exact same tasks in two different domains are quite different. Therefore to correctly apply $G_{1 \rightarrow 2}$ we need to take into account these difficulties.

8.4.3 Batch Normalization

We investigate the impact on performance of using task networks with or without batch normalization layers [194]. Our intuition is that the introduction of batch normalization yields more similar features across domains and

$\mathcal{A}$	Method	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
Synthia	N_2^A	**99.23**	**87.16**	**92.67**	**28.62**	**48.53**	**63.54**	**85.02**	**88.92**	**52.67**	**96.91**	**98.39**	**76.52**	**98.45**
Synthia	AT/DT	98.34	76.09	84.99	1.06	29.25	45.57	80.15	85.72	25.31	95.53	97.45	65.41	97.53

TABLE 8.7: Experimental results of *Dep.* $\rightarrow$ *Sem.* scenario using as domain $\mathcal{A}$ the Synthia dataset. Best results highlighted in bold.

		Lower is better				Higher is better		
$\mathcal{A}$	Method	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Synthia	N_2^A	0.138	**1.212**	**4.759**	0.825	**0.864**	0.952	0.970
Synthia	AT/DT	**0.135**	1.271	5.061	**0.634**	0.863	**0.958**	**0.977**

TABLE 8.8: Experimental results of *Sem.* $\rightarrow$ *Dep.* scenario using as domain $\mathcal{A}$ the Synthia dataset. Best results highlighted in bold.

smaller numerical values, making the training of $G_{1\rightarrow 2}$ easier and numerically more stable. In Table 8.6 we report results for the *Dep.* $\rightarrow$ *Sem.* scenario when employing Synthia as $\mathcal{A}$ and Cityscapes as $\mathcal{B}$. As expected, batch normalization yields representations more similar between domains, thus leading to better generalization performances on $\mathcal{B}$. Counter-intuitively, we also notice that results on $\mathcal{A}$ are worse with batch normalization, perhaps due to mapping features from $\mathcal{T}_1$ to $\mathcal{T}_2$ being harder when these lay within a more constrained space.

8.4.4 Train domain performance of $G_{1\rightarrow 2}$

Our framework has to overcome two nuisances to effectively address the lacking of supervision in the target task and domain: translation of features between tasks and change of domain. In this section, we are interested in isolating the impact of the first nuisance, which will also provide some hints on the importance of the second one. In other words, we are trying to answer the question: *How well are we effectively learning to translate deep representations?*

To focus only on the effectiveness in transferring representations, we consider a test set of images from $\mathcal{A}$ and compare AT/DT and N_2^A (the network trained on domain $\mathcal{A}$ for $\mathcal{T}_2$). As the test data are sampled from the same domain as the training data, we do not have errors due to the domain shift and can use the gap in performance between the two algorithms as a measure of the effectiveness of our framework in transferring representations.

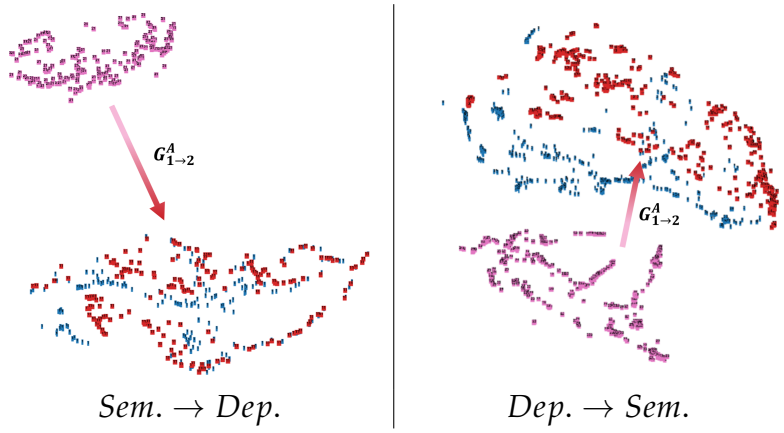

FIGURE 8.6: t-SNE [195] plots of deep features computed on $\mathcal{A}$. Pink denotes the features extracted for $\mathcal{T}_1$, i.e. $E_1^{A\cup B}(x_a)$. Blue features extracted for $\mathcal{T}_2$, i.e. $E_2^A(x_a)$. Red the prediction obtained by the feature transfer network $G_{1\to2}(E_1^{A\cup B}(x_a))$. Therefore, the red points are the transformations of the pink points according to $G_{1\to2}$. With an ideal $G_{1\to2}$ red and blue points would perfectly overlap, here we can see that unfortunately this is not the case. Nevertheless our transfer function successfully transform pink features to make them closer to blue ones.

As we wish to evaluate both semantic segmentation and depth estimation, we select the Synthia domain as $\mathcal{A}$, for which we have all labels available, and Cityscapes as $\mathcal{B}$. In Table 8.7 we report the results when transferring deep representations in the *Dep.* → *Sem.* scenario, while in Table 8.8 in the *Sem.* → *Dep.* scenario.

Table 8.7 shows how transferring deep representations from $\mathcal{T}_1$ to $\mathcal{T}_2$ with AT/DT results in a small loss in performance compared to N_2^A. In particular, the largest performance drops are related to classes dealing with small objects, like 'Fence', 'Poles' and 'Traffic Sign', that might get lost transferring features at the smallest spatial resolution in the network. These results suggest that a multi-scale transfer strategy would be a direction worth exploring in future work to better recover small details upon transferring representations. Nevertheless, the comparisin between the final pixel accuracy (Acc.) highlights that AT/DT loses only 1% though relying on a feature extractor trained for a different task.

In Table 8.8 AT/DT obtains again performance close to N_2^A. For some metrics, it even delivers better performance than N_2^A. This somewhat surprising result can be explained by the difference between the training sets: AT/DT uses as feature extractor $N_1^{A\cup B}$, which has been trained with samples from both $\mathcal{A}$ and $\mathcal{B}$, i.e. with a larger and more varied training set than that used by N_2^A. Therefore, the encoder of $N_1^{A\cup B}$ might learn a more general

	$\mathcal{A}$	$\mathcal{B}$	Method	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
	Carla	Cityscapes	Baseline	71.87	**36.53**	3.99	**6.66**	24.33	22.20	66.06	**48.12**	7.60	60.22	69.05	37.88	74.61
(c)	Carla	Cityscapes	No Transfer	**84.82**	33.15	1.00	1.79	6.30	14.26	**69.91**	40.32	1.84	65.67	73.49	35.69	**79.53**
(c)	Carla	Cityscapes	AT/DT	76.44	32.24	**4.75**	5.58	**24.49**	**24.95**	68.98	40.49	**10.78**	**69.38**	**78.19**	**39.66**	76.37

TABLE 8.9: Experimental results of *Dep.* $\rightarrow$ *Sem.* scenario. Best results highlighted in bold.

feature extractor than that of N_2^A, this resulting in better performance when applied on unseen data. AT/DT can successfully leverage on this better feature extractor and obtain slightly better performance when transferring them to $\mathcal{T}_2$.

The same reasoning may be applied to the results of Table 8.7. However, in this case, the shared encoder of $N_1^{A \cup B}$ has been partially trained with noisy ground truth depth labels on samples from $\mathcal{B}$. The introduction of noise in the training process might harm the learning of $N_1^{A \cup B}$ and explain the small gap in performance. Moreover, as stated above, due to the transferring of features at low resolution, AT/DT might struggle to transfer small image structures (e.g. , 'poles', 'traffic sign'...). However, wrong predictions on this kind of small structures do not arm much the depth estimation metrics (i.e. , few pixels are considered), though they have a larger impact on the mIoU metric considered for semantic segmentation. Finally, as stated in [1], the advantages yielded by semantic information to depth estimation are larger than the gains attainable going in the other direction, thus motivating the slight difference in performance across the two scenarios.

Overall, the results reported in Table 8.8 and Table 8.7 show that our framework is indeed learning to transfer deep representations effectively and that it is possible to approximate $G_{1 \rightarrow 2}$ by a neural network like that we propose in this work. This is further validated in Figure 8.6, where we report two t-SNE[195] plots of deep features extracted by $N_1^{A \cup B}$ (in pink), N_2^A (in blue) alongside with the features transformed by $G_{1 \rightarrow 2}$ (in red). All features are computed on image samples from the test set described above, i.e. samples unseen at training time. Therefore, $G_{1 \rightarrow 2}$ takes as input pink points and produces red points that should be as close as possible to the blue points. Indeed, the two plots show how our task transfer network can successfully produce features suitable for $\mathcal{T}_2$.

	$\mathcal{A}$	$\mathcal{B}$	Method	Lower is better				Higher is better		
				Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
(b)	Carla	Cityscapes	Baseline	0.667	13.500	16.875	0.593	0.276	0.566	0.770
(b)	Carla	Cityscapes	No Transfer	0.615	17.578	19.924	0.533	0.284	0.646	0.845
	Carla	Cityscapes	AT/DT	**0.394**	**5.837**	**13.915**	**0.435**	**0.337**	**0.749**	**0.899**

TABLE 8.10: Experimental results of *Sem.* $\rightarrow$ *Dep.* scenario. Best results highlighted in bold.

8.4.5 Importance of $G_{1\rightarrow 2}$

We report results of additional tests to further assess the importance of $G_{1\rightarrow 2}$ in our cross tasks and domains adaptation. Purposely, we consider a single network made out of one encoder, $E_{1,2}^{A\cup B}$ and two decoders, $D_1^{A\cup B}$ and D_2^A. $D_1^{A\cup B}$ is trained with samples from $\mathcal{A}$ and $\mathcal{B}$ for $\mathcal{T}_1$. D_2^A is trained with samples from $\mathcal{A}$ for $\mathcal{T}_2$. Finally, $E_{1,2}^{A\cup B}$ is trained together with the two heads with both tasks and domains. Therefore we consider a single feature extractor which yields a shared representation for both tasks and domains without the need to learn a transfer function between tasks. We will refer to this configuration as the *No Transfer* setting.

We evaluate *No Transfer* for both *Dep.* $\rightarrow$ *Sem.* and *Sem.* $\rightarrow$ *Dep.* settings from Carla to Cityscapes and compare it to AT/DT and the transfer learning baseline. Table 8.9 and Table 8.10 report results for *Dep.* $\rightarrow$ *Sem.* and *Sem.* $\rightarrow$ *Dep.* settings respectively. For *Sem.* $\rightarrow$ *Dep.* our method outperforms *No Transfer* for all metrics, and indeed this alternative is even worse than the baseline for Sq. Rel. and RMSE. On the other hand, for *Dep.* $\rightarrow$ *Sem.* our method achieves better performances in the majority of the classes and for the mIoU, while *No Transfer* provides the best pixel accuracy. We ascribe this result to *No Transfer* providing the highest IoU for the *road* class, which represents the vast majority of pixels in an autonomous driving scenario. However this good performance does not translate to other classes such that *No transfer* achieves the worst mIoU, even less than the baseline. These results confirm the importance of learning a mapping function (e.g. , $G_{1\rightarrow 2}$) between features to transfer representations between tasks.

8.4.6 Shared Decoder and Separate Encoders for N_1

In subsection 8.4.2 we highlighted how learning a common representation for $\mathcal{T}_1$ is crucial to learn a transfer function which generalize across domains. In this additional test we show that to learn a good shared representation across domains for one task, we need to share both encoders and decoders in $N_1^{A\cup B}$.

Shared Encoders	mIoU	Acc.
✗	11.55	56.79
✓	23.24 **(+11.69)**	64.03 **(+7,24)**

TABLE 8.11: Study on Shared Decoder with Non Shared Encoders for $N_1^{A \cup B}$. We show a $\mathcal{A}$: Synthia to $\mathcal{B}$: Carla and *Dep.* $\rightarrow$ *Sem.* scenario. Performance improvement highlighted in bold.

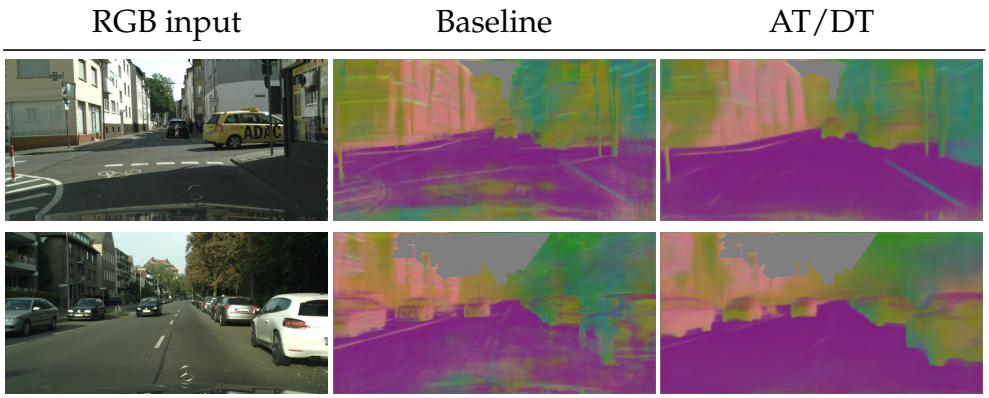

FIGURE 8.7: Qualitative results on Cityscapes dataset in a *Sem.* $\rightarrow$ *Norm.* scenario. From left to right: RGB input, prediction obtained by a transfer learning baseline and by our framework (AT/DT).

For this reason we train a different version of $N_1^{A \cup B}$ with a shared decoder but two encoders, one trained only on $\mathcal{A}$ and the other only on $\mathcal{B}$. Table 8.11 compares this architecture to AT/DT for Synthia to Cityscapes in the *Dep.* $\rightarrow$ *Sem.*scenario. Indeed training a shared encoder allows the representation to be more closely related resulting in better performance.

8.4.7 Additional tasks

In Figure 8.7 we report additional qualitative results when using as $\mathcal{T}_1$ semantic segmentation and as $\mathcal{T}_2$ normal estimation, with Carla as $\mathcal{A}$ and Cityscapes as $\mathcal{B}$. The results confirm the findings of the semantic to depth scenario, with AT/DT producing clearly better prediction than the baseline network. We report only qualitative results due to the lack of annotations to validate normal estimation on the real Cityscapes data.

Chapter 9

Learning Good Features to Transfer Across Tasks and Domains

In this chapter we expand and improve our original framework shown in chapter 8 by performing two types of feature alignment to ease the learning of a mapping function between them. We align feature representations across domains using a novel norm discrepancy alignment loss that constrain the feature space by penalizing features with very different norms in a spatially aware manner. At the same time, we align feature representations across tasks by using them as inputs to solve a common auxiliary task. This extra task can be very simple as it only acts as a bridge between the source and the target one by forcing the deep features extracted to solve them to share the same encoding and semantic content.

We test the effectiveness of this extension in the same tasks as the original framework, monocular depth estimation and semantic segmentation. We argue that both tasks need to reason about the borders of the objects. For instance, in semantic segmentation we need to reason about class borders, in depth estimation we need to estimate discontinuities related to objects at different depths. Thus, we select as auxiliary bridge task edge detection. Moreover, the generalization performance of state of the art models for this task [196] are good enough to allow to use them for proxy supervision avoiding the need for extra labels. We evaluate the extended framework in a similar setting to the previous one, using a fully supervised and completely synthetic domain (i.e. , the Carla simulator [25]) to improve the performance on a partially labeled real one (i.e. , Cityscapes [14]).

9.1 Extended ATDT

We describe here the additional improvement made to AT/DT in order to achieve better transferability across tasks and domains. We use the same

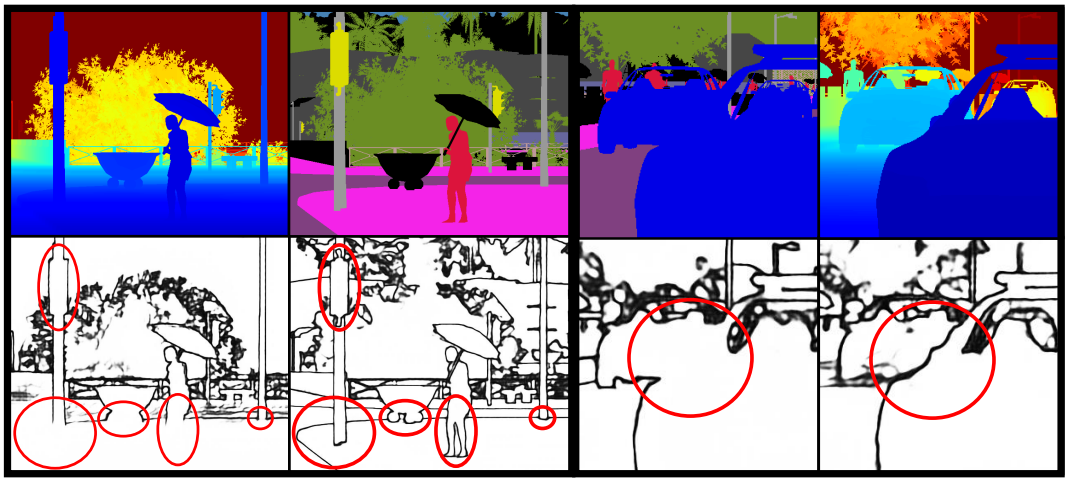

FIGURE 9.1: Two task transfer scenarios: on the left, the depth-to-semantic case; the opposite on the right. Red circles highlight image details that are needed to perform the second task but are not present in the first one. We propose to incorporate relevant details of the scene within deep features by exploiting an auxiliary edge detection task, with the aim of making these representations easier to transfer across tasks and domains.

mathematical notation as reported in subsection 8.1.1. A graphical overview of the additional improvements is depicted in Figure 9.2.

9.1.1 Feature Alignment Across Domains

In order to achieve good performances, it is crucial to have a $G_{1\to 2}$ which generalize well in a target unseen domain $\mathcal{B}$ even if trained only source data from $\mathcal{A}$. Domain Adaptation (DA) literature already offers several ways to accomplish this. One may act on the input space [147], on the feature space [130] or on the output space of the network [136]. In our case though, both input and output space of $G_{1\to 2}$ are high dimensional latent spaces and, as reported in [136], unsupervised domain adaptation techniques tend to fail when applied to such spaces for dense tasks. However, we can address the domain shift directly on the input space of $G_{1\to 2}$ since in our framework it boils down to the feature space of N_1, where partial alignment is already achieved by simultaneous supervised training on $\mathcal{A}$ and $\mathcal{B}$. We can further diminish the domain-shift between the two feature spaces by regularizing them, enforcing that f_1 features extracted from E_1 in $\mathcal{A}$ and $\mathcal{B}$ have similar L_2 norms across channels. We preserve spatial information while calculating the norms assuming that the two domains contains scenes with similar structure (as it is the case for autonomous driving applications). Starting from features

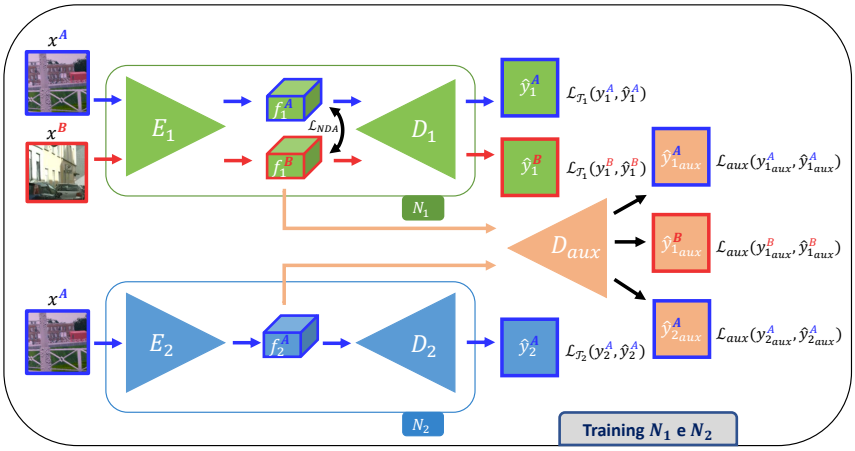

FIGURE 9.2: Features alignment strategies across tasks and domains. We train jointly the networks N_1, N_2 and a shared auxiliary decoder D_{aux}. We train N_1 to solve $\mathcal{T}_1$ on images from domains $\mathcal{A}$ and $\mathcal{B}$ using a supervised loss $\mathcal{L}_{\mathcal{T}_1}$ for $\mathcal{T}_1$ alongside a novel feature Norm Discrepancy Alignment loss $\mathcal{L}_{NDA}$ which helps better aligning the features computed by N_1 across the two domains. We train N_2 using a supervised loss $\mathcal{L}_{\mathcal{T}_2}$ for $\mathcal{T}_2$ on images from $\mathcal{B}$. D_{aux} is trained to solve an auxiliary task $\mathcal{T}_{aux}$ using the loss $\mathcal{L}_{aux}$ and based on the features computed by E_1 on images from $\mathcal{A}$ and $\mathcal{B}$ as well as by E_2 on images from $\mathcal{B}$.

f_1^A and f_1^B of dimensions $H \times W \times C$, where H, W and C are the height, width and channels of the feature maps respectively, we calculate the L_2 norm along the C axis and we minimize the absolute difference between each spatial location i, j along the H and W dimensions. Formally our NDA Loss is defined as follows:

$$L_{NDA} = \frac{1}{W \times H} \sum_{i=1}^{H} \sum_{j=1}^{W} |||f_{1_{i,j}}^A||_2 - ||f_{1_{i,j}}^B||_2| \tag{9.1}$$

9.1.2 Feature alignment across tasks

We have previously shown a practical solution to improve the generalization across domains of our mapping. However, we want to go a step further and align features also across tasks to ease the learning of a mapping function. We believe that, f_1 should contain as much information as possible, even if they are not strictly needed to solve $\mathcal{T}_1$, because they could be useful for $\mathcal{T}_2$. For this reason, while training networks for $\mathcal{T}_1$, we simultaneously train a decoder to solve an auxiliary task $\mathcal{T}_{aux}$ to enrich representations. However, though multi-task learning of $\mathcal{T}_1$ and $\mathcal{T}_{aux}$ can help to encode more relevant information in the T_1 features f_1, it does not guarantee that the decoder D_2

used at inference time on the transferred features from T_1 to T_2, $f_{1\to2}$, can make proper use of them if it has been trained only to solve $\mathcal{T}_2$ in isolation. $\mathcal{T}_{aux}$ can be used to this end and learn to solve it with the same decoder D_{aux} also from features f_2 computed by E_2.

In detail, given auxiliary task labels Y^A_{aux} and Y^B_{aux} for $\mathcal{A}$ and $\mathcal{B}$, we train N_1 and N_2 simultaneously with an auxiliary decoder D_{aux} using an auxiliary loss $\mathcal{L}_{aux}$. Therefore, we obtain auxiliary predictions in the following way: $y_{aux,k} = D_{aux}(E_k(x)), k \in [1,2]$. Again, we feed images of both domains through E_1, while we pass only images from $\mathcal{A}$ through E_2. We do not pass images belonging to $\mathcal{B}$ through E_2 while training D_{aux} since this would be the only supervision for E_2 on $\mathcal{B}$ and it may skew E_2 output to be more effective on $\mathcal{T}_{aux}$ than on $\mathcal{T}_2$.

9.2 Experimental Settings

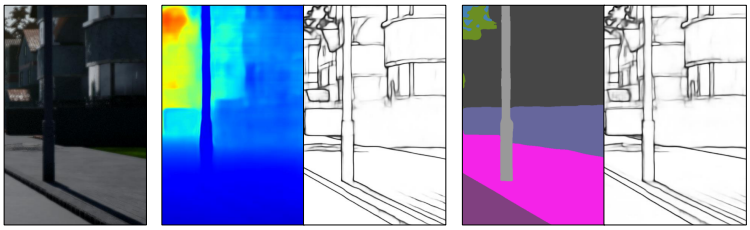

FIGURE 9.3: From left to right: RGB input image of domain $\mathcal{A}$, depth prediction from N_1, edges from f_1, semantic segmentation from N_2 and edges from f_2. Task features f_1 and f_2 encode richer details than strictly needed to solve either task as we can recover all edges from both of them.

Tasks. We fix $\mathcal{T}_1$ and $\mathcal{T}_2$ to be monocular depth estimation or semantic segmentation. These two visual tasks can be addressed using the same base architecture and changing only the final layer. Semantic segmentation is solved by minimizing a cross entropy loss, monocular depth estimation by minimizing a L_1 loss. We select edge detection as our $\mathcal{T}_{aux}$ since this task has many advantages. First, from an implementation point of view, it can be solved using again the same decoder as $\mathcal{T}_1$ and $\mathcal{T}_2$. Second, since our main goal is to improve transferability of features among tasks, we force features for $\mathcal{T}_1$ to contain as many details as possible of the scene, even if they are not strictly needed to solve $\mathcal{T}_1$. To make a concrete example, we can think about the case of $\mathcal{T}_1$ being depth estimation and $\mathcal{T}_2$ semantic segmentation. Features f_1 used to compute depth can ignore boundaries between semantically

distinct regions of the image that are not needed to correctly predict depth, as shown in Fig. 9.1 (e.g. , legs or tyres touching the ground, or between street signs and poles). Therefore, even if fed to a perfect $G_{1\to 2}$, f_1 may not contain all the information needed to restore the semantic structure of the image. By solving edge-detection, instead, we force the network to extract additional information from the image, not normally encoded when training depth features in isolation. We define L_{aux} as a L_2 loss for training the edge decoder.

Datasets. We set $\mathcal{A}$ and $\mathcal{B}$ to be the synthetic and real datasets, respectively. We use as $\mathcal{A}$ a collection of images generated with the Carla simulator [25], while as $\mathcal{B}$ the Cityscapes dataset [14]. We generate a new version of the Carla dataset w.r.t. the one used to evaluate the original AT/DT to reduce the gap between synthetic and real scenes, using FOV of 100° and collecting 3500, 500, and 1000 images for training, validation, and testing respectively. For each image, we store the associated depth and semantic labels easily provided by the simulator. Finally, we use a pre-trained state-of-the-art neural network[196] (trained on datasets different from $\mathcal{A}$ and $\mathcal{B}$) as an off-the-shelf black-box edge detector to extract the edges from both $\mathcal{A}$ and $\mathcal{B}$ to be used as proxy labels when learning $\mathcal{T}_{aux}$.

Architecture. We use the same architecture as described in section 8.2 to solve each task. The two encoders are also used to capture good features for edge detection, which is solved using D_{aux}, that shares the same architecture as the decoders used in N_1 and N_2. $G_{1\to 2}$ is a simple CNN made out of 6 pairs of convolutional and batch normalization layers with kernel size 3×3 which, differently from the original AT/DT version, do not perform any downsampling or upsampling operation.

Evaluation protocol. During the first step of the training, i.e. , training N_1 and N_2, we found to be extremely relevant to monitor the performance of the two networks. Indeed, we have observed that the more effective is N_2 on the downstream task, the higher is the final performance of our method and that the same reasoning may be applied on N_1 when trained on $\mathcal{A}$ and $\mathcal{B}$: better results lead to a superior domain adaptation method.

During the training phase of the transfer network, the model is evaluated on the validation set of Carla. Of course, it is possible that the global optimum for Carla is not the global optimum for Cityscapes. Yet, we cannot use data from the target domain neither for hyper-parameters tuning nor for

early stopping, because these data would not be available in a real case scenario. Therefore, the Cityscapes validation set is only used at test time to measure the final performances of our adaptation method.

We use the same metrics described in section 8.2 to evaluate the extended framework.

$\mathcal{A}$	$\mathcal{B}$	Method	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
Carla	CS	Source	78.99	38.81	1.34	5.80	24.02	24.47	71.98	52.23	5.57	65.17	59.10	38.86	78.58
Carla	CS	**ATDT**	**90.57**	**48.46**	**7.37**	**12.27**	**41.16**	**31.90**	**81.96**	**72.77**	**23.44**	**77.85**	**76.33**	**51.28**	**87.57**
CS	CS	Transfer Oracle	89.69	48.05	11.46	29.58	59.68	35.84	85.83	85.57	34.03	78.17	85.54	58.50	88.84
-	CS	Oracle	96.74	78.28	29.26	40.78	72.39	51.28	90.69	91.94	58.92	86.33	89.23	71.44	93.90

TABLE 9.1: Experimental results of *Dep.* $\rightarrow$ *Sem.* scenario. Source stands for N_2 trained on $\mathcal{A}$ and tested on $\mathcal{B}$, Transfer Oracle represents $G_{1\rightarrow 2}$ trained only on $\mathcal{B}$, Oracle refers to N_2 trained and tested on $\mathcal{B}$. Best results highlighted in bold.

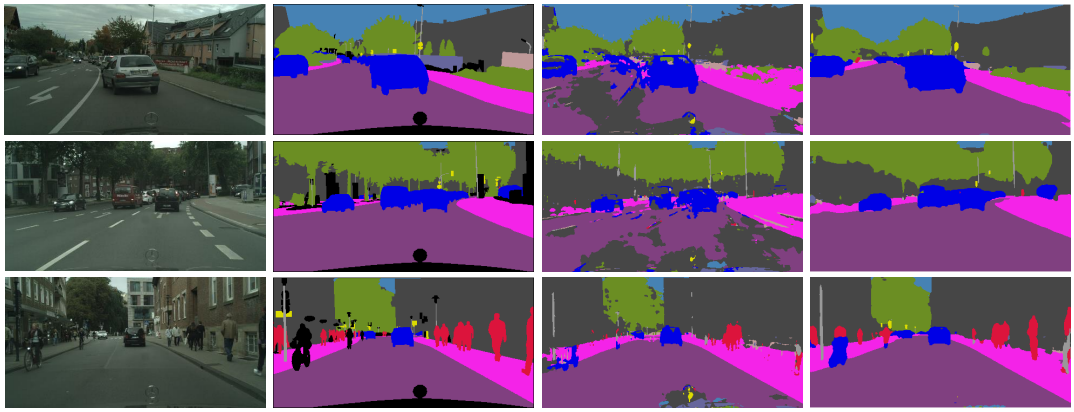

FIGURE 9.4: Qualitative results of the *Dep.* $\rightarrow$ *Sem.* scenario. From left to right: RGB image, ground truth, baseline trained only on domain $\mathcal{A}$, ours.

9.3 Experimental Results

We provide results for two different settings: transferring features from depth estimation to semantic segmentation (Sec. 9.3.1) as well as from semantic segmentation to depth estimation (Sec. 9.3.2).

In both scenarios, as already mentioned, we used edge detection as auxiliary task, motivated by the idea that either semantic segmentation and depth estimation can benefit from edge information. Fig. 9.3 shows that with our multi-task learning protocol we are able to restore all the details of the scene

from both f_1 and f_2, proving that N_1 and N_2 learned to encode richer information than strictly needed to solve $\mathcal{T}_1$ and $\mathcal{T}_2$.

$\mathcal{A}$	$\mathcal{B}$	Method	Lower is better				Higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Carla	CS	Source	0.7398	15.169	14.774	0.641	**0.406**	0.650	0.781
Carla	CS	**ATDT**	**0.3928**	**4.9094**	**12.363**	**0.444**	0.372	**0.757**	**0.923**
CS	CS	Transfer Oracle	0.2210	2.2962	9.032	0.275	0.669	0.914	0.972
-	CS	Oracle	0.1372	1.6214	8.566	0.244	0.816	0.938	0.976

TABLE 9.2: Experimental results of *Sem.* $\rightarrow$ *Dep.* scenario. Source stands for N_2 trained on $\mathcal{A}$ and tested on $\mathcal{B}$, Transfer Oracle represents $G_{1\rightarrow 2}$ trained only on $\mathcal{B}$, Oracle refers to N_2 trained and tested on $\mathcal{B}$. Best results highlighted in bold.

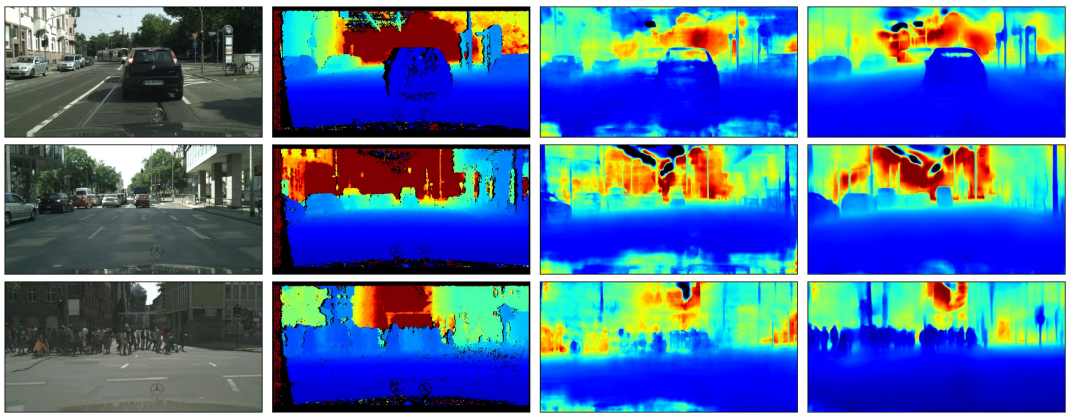

FIGURE 9.5: Qualitative result of the *Sem.* $\rightarrow$ *Dep.* scenario. From left to right: RGB image, ground truth, baseline network trained only on domain $\mathcal{A}$, ours.

9.3.1 Depth to Semantics

In this setup, denoted as *Dep.* $\rightarrow$ *Sem.*, the goal of our framework is to transform depth features into semantic segmentation features. This mapping is learned using Carla as domain $\mathcal{A}$ and Cityscapes as domain $\mathcal{B}$. We report results in Tab. 9.1: the first row shows results obtained with no adaptation (i.e. , training N_2 on Carla and testing it directly on Cityscapes), while from the second row we can see that our final framework yields 51.28% mIoU and 87.57% Acc with an improvement of +12.48% and +8.99% respectively in terms of mIoU and Acc wrt to the baseline.

Furthermore, as we are transferring features from another task, it is worth trying to investigate on the upper bound in performance due to the inherent

transferability of the features between the two tasks. Thereby, we train $G_{1\to2}$ using only Cityscapes to learn a mapping function in a supervised fashion as explained in Sec. 9.1.2 on $\mathcal{B}$ and testing on the validation set of $\mathcal{B}$. These results are shown in the third row of the table (denoted as Transfer Oracle): given a transfer architecture, there seems to be an upper bound in performance due to the nature of the two tasks, which in the considered setting amounts to a 58.5% mIoU. Thus, our proposal shows a gap that is only about -7.2% mIoU. We also report the performance of N_2 trained on $\mathcal{B}$ and tested on $\mathcal{B}$ to show the absolute upper bound (last row of the table, denoted as Oracle).

Some qualitative results dealing with the *Dep.* $\to$ *Sem.* scenario are depicted in Fig. 9.4. It is possible to appreciate the overall improvement of our method wrt the baseline, either in flat areas (e.g. , roads, sidelwalks and walls), in objects shapes (e.g. , cars and persons) or in fine-grained details (e.g. , poles and traffic signs).

9.3.2 Semantics to Depth

In this setup, which we define as *Sem.* $\to$ *Dep.*, the goal of our framework is to transform semantic features into depth features. This mapping is learned using Carla as domain $\mathcal{A}$ and Cityscapes as domain $\mathcal{B}$.

Results are reported in Tab. 9.2. Similarly to the *Dep.* $\to$ *Sem.* scenario, in the first row we show results with no adaptation (denoted as Source), while the second row presents the ones obtained with our framework. In Fig. 9.5, we show some qualitative results of the *Sem.* $\to$ *Dep.* scenario. While predictions look quite noisy in the background, we can see a good improvement in the foreground area thanks to our method. Shapes are recovered almost perfectly, both for big and small objects, even with difficult subjects like the crowd in the bottom row. Additionally, our method enables a remarkable enhancement of the prediction smoothness.

9.4 Additional Experiments

In the following sections, we study the effectiveness of each implementation choice.

$\mathcal{A}$	$\mathcal{B}$	Edge	NDA	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
Carla	CS			89.95	46.77	5.16	10.21	28.93	28.92	77.50	71.37	19.24	75.29	75.12	48.04	85.90
Carla	CS	✓		90.12	48.90	4.18	11.63	37.40	31.98	**82.34**	71.50	15.11	**78.04**	**80.61**	50.16	87.21
Carla	CS		✓	**91.21**	**50.16**	5.14	**13.78**	36.99	**32.10**	77.72	**73.38**	**23.47**	76.67	72.67	50.30	86.77
Carla	CS	✓	✓	90.57	48.46	**7.37**	12.27	**41.16**	31.90	81.96	72.77	23.44	77.85	76.33	**51.28**	**87.57**

TABLE 9.3: Ablation study in the *Dep.* $\rightarrow$ *Sem.* scenario. Best results highlighted in bold. Edge refers to the framework trained with our details-aware features. NDA refers to the framework trained with our NDA loss.

9.4.1 Contribution of $\mathcal{T}_{aux}$ and NDA Loss

We start by studying the effect of introducing in our framework the auxiliary task and the NDA Loss, analyzing their contribution either when used separately or combined together. The second and the third row of Tab. 9.3 report results obtained in the *Dep.* $\rightarrow$ *Sem.* setting respectively when integrating in our method exclusively the auxiliary task (i.e. , edge detection) or the NDA loss. We can see that both techniques bring in an improvement of about +2% in terms of mIoU wrt to the plain version of AT/DT (first row). Interestingly, though, from the last row of the table we can see that edge detection and the NDA loss result complementary when combined, providing an overall improvement of +3.34% mIoU.

Fig. 9.6 presents some zoomed-in qualitative results: we can see how small details such as poles or car shapes are recovered with our method wrt results obtained without $\mathcal{T}_{aux}$ and NDA loss.

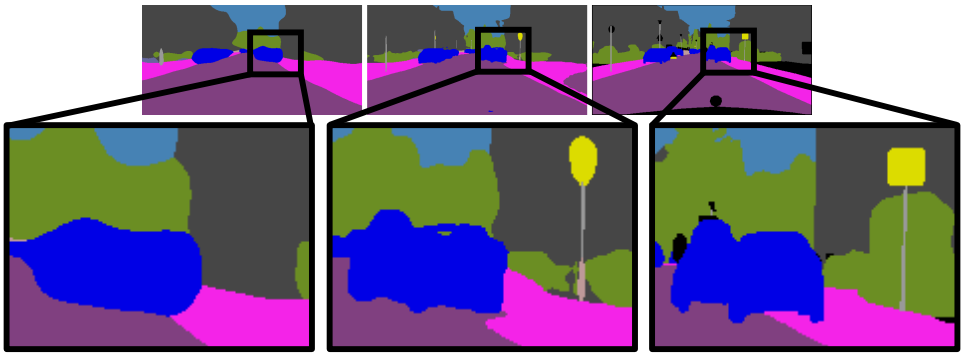

FIGURE 9.6: Zoomed results in a *Dep.* $\rightarrow$ *Sem.* scenario. From left to right: plain AT/DT without edge and NDA, our complete framework, ground truth. We notice how our method is able to recover fine-grained details of the output.

Aux Task	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
None	89.95	46.77	5.16	10.21	28.93	28.92	77.50	71.37	**19.24**	75.29	75.12	48.04	85.90
Autoencoder	**90.68**	**50.12**	**7.45**	9.08	31.40	29.43	78.72	68.51	12.95	74.67	75.68	48.07	86.31
Edge detection	90.12	48.90	4.18	**11.63**	**37.40**	**31.98**	**82.34**	**71.50**	15.11	**78.04**	**80.61**	**50.16**	**87.21**

TABLE 9.4: Comparison between autoencoder and edge detection as auxiliary tasks in the *Dep.* $\rightarrow$ *Sem.* scenario. Best results highlighted in bold.

9.4.2 Effectiveness of edge detection as auxiliary task

In this section, we show empirically that the choice of the proper auxiliary task is key for the performance of our framework.

In both the *Dep.* $\rightarrow$ *Sem.* and the *Sem.* $\rightarrow$ *Dep.* scenarios, we proposed to use edge detection as an auxiliary task because it captures information about the shapes of the objects in the input images, and allows for the straightforward computation of proxy labels. To validate this design choice, we tested our framework in the *Dep.* $\rightarrow$ *Sem.* setting, using D_{aux} to simply reconstruct the input images both from f_1 and f_2, i.e. , the classical autoencoder setting (results in Tab. 9.4). Interestingly, using a reconstruction task as the auxiliary task achieves comparable performances in terms of mIoU over plain AT/DT. We believe that the autoencoder is a trivial task which does not require the extraction of informative features about the image, therefore not providing any additional cues to the downstream task.

9.4.3 Importance of simultaneous training of N_1, N_2 and D_{aux}

In our experiments we use edge detection as auxiliary task and train a shared decoder D_{aux} to reconstruct edges of the input image from features extracted by both E_1 and E_2. In fact, we argue that this procedure should force E_1 to encode in the extracted features also edges that are not necessary for $\mathcal{T}_1$ but could be relevant for $\mathcal{T}_2$. Besides, we believe that simultaneous training of N_1, N_2 and D_{aux} is crucial to encourage features coming from E_1 and E_2 to encode edge information in the same way, making it easier to learn $G_{1\rightarrow 2}$.

In Tab. 9.5 we report the ablation study conducted to validate these intuitions. We consider the *Dep.* $\rightarrow$ *Sem.* scenario using the Carla dataset as domain $\mathcal{A}$ and Cityscapes as domain $\mathcal{B}$. The four rows of the table represent the following training schemes:

method	Road	Sidewalk	Walls	Fendce	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
plain AT/DT	89.95	46.77	5.16	10.21	28.93	28.92	77.50	71.37	**19.24**	75.29	75.12	48.04	85.90
Separate (N_1 + edge), N_2	87.24	43.30	3.08	10.17	41.77	29.04	81.81	72.35	16.58	77.10	73.10	48.69	85.89
Separate (N_1 + edge), (N_2 + edge)	88.83	47.31	**7.10**	8.59	**44.53**	30.99	**83.24**	**73.54**	18.05	**78.10**	69.66	49.99	86.72
Simultaneous (N_1 + N_2 + edge)	**90.12**	**48.90**	4.18	**11.63**	37.40	**31.98**	82.34	71.50	15.11	78.04	**80.61**	**50.16**	**87.21**

TABLE 9.5: Ablation study on the use of edge detection as auxiliary task. Best results highlighted in bold. See text for a detailed explanation of the training protocol used in each row.

1. We report the results obtained by *plain* AT/DT (i.e. , trained without $\mathcal{T}_{aux}$ and NDA loss) as a baseline.

2. We first train N_1 and D_{aux} on both $\mathcal{A}$ and $\mathcal{B}$. Then, we train N_2 on $\mathcal{A}$. Finally, we train $G_{1\to2}$ on features extracted by E_1 and E_2 on domain $\mathcal{A}$.

3. We train N_1 and a first D^1_{aux} on both $\mathcal{A}$ and $\mathcal{B}$. Then, we train N_2 and a second D^2_{aux} on $\mathcal{A}$. Finally, we train $G_{1\to2}$ on features extracted by E_1 and E_2 on domain $\mathcal{A}$

4. Our proposed method with a simultaneous training of N_1, N_2 and a shared D_{aux}.

The introduction of edge detection as auxiliary task helps in every scenario. In fact, if we use D_{aux} only during training of N_1 (second row), we already see an increase of 0.6% in the overall mIoU. We believe that this is explained by the presence of edge details (not strictly necessary to solve $\mathcal{T}_1$ but relevant for $\mathcal{T}_2$) in features extracted by E_1. However, $G_{1\to2}$ can have difficulties in adapting f_1 into f_2 when the edge information is not explicitly present in f_2. This is confirmed by the result in the third row of the table, where an additional increase of 1.3% in the overall mIoU is attained by using two different D_{aux} (one during training of N_1 and one during training of N_2). Finally, the best results in terms of mIoU and Acc are achieved by our method, i.e. , when simultaneously training N_1, N_2 and a shared D_{aux}. This shows the benefit of encoding in the same way the edge information in f_1 and f_2 to enforce feature alignment across tasks.

9.4.4 Alignment strategies for N_1

An alternative way to align N_1 features between domains to ease the transfer process and favor the generalization of $G_{1\to2}$, is to apply the widely used adversarial training in feature space. In our settings, this can be done by

E_1 Align.	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
None	89.95	46.77	**5.16**	10.21	28.93	28.92	77.50	71.37	19.24	75.29	**75.12**	48.04	85.90
Adv.	89.89	46.01	4.22	11.89	**38.20**	30.65	77.00	63.68	12.99	74.35	81.16	48.19	85.42
NDA	**91.21**	**50.16**	5.14	**13.78**	36.99	**32.10**	**77.72**	**73.38**	**23.47**	**76.67**	72.67	**50.30**	**86.77**

TABLE 9.6: Comparison between NDA loss and adversarial training to align E_1 features. Best results highlighted in bold.

adding a critic that must discriminate whether the features produced by E_1 come from $\mathcal{A}$ or $\mathcal{B}$. Thus, the encoder E_1 not only has to learn a good feature space for its task, but it is also asked to fool the critic. Afterward, we can proceed to learn a mapping function $G_{1\to 2}$ among tasks as usual. In Tab. 9.6 we compare this standard DA methodology to our NDA loss. Adversarial training (second row) does not introduce significant improvements over not performing DA for $\mathcal{T}_1$ (first row, it even lowers the pixel-wise accuracy), while constraining the features extracted by E_1 in a norm aligned space (third row) significantly increases both metrics with respect to the baseline. Our intuition is that although adversarial training can be useful for domain alignment, it alters the learned feature space with the goal of fooling the critic, and due to the instability of adversarial learning, this training objective can lead to worse performances on the current task. Our NDA loss on the other hands, acts as a simple regularizer that tries to favor the generalization of the network, thereby it directly helps the network to perform better in the specific tasks.

9.4.5 Aligning N_2 features

E_2 Align.	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
plain AT/DT	**89.95**	**46.77**	5.16	**10.21**	28.93	**28.92**	**77.50**	71.37	**19.24**	**75.29**	75.12	**48.04**	**85.90**
Adv.	89.36	46.03	**5.59**	8.22	**36.45**	25.44	75.15	**72.29**	12.69	74.12	**75.79**	47.38	85.31
NDA	44.94	23.82	3.81	2.09	30.74	24.21	42.08	68.84	11.69	35.67	11.10	27.18	56.17

TABLE 9.7: Results of aligning output space of E_2 in a *Dep.* $\to$ *Sem.* scenario. Best results highlighted in bold.

We tried to perform feature alignment across domains also on the features f_2 extracted by E_2, by either deploying adversarial training or imposing our

NDA loss. The idea is to favor the generalization of $G_{1\to2}$ by making not only alignment in its input space (i.e. , the features produced by E_1, aligned with our NDA loss) more homogeneous, but also its output space, i.e. , the features produced by E_2. However, the setting is not completely symmetric: when learning E_2, we do not have supervision available for $\mathcal{B}$, and the only loss shaping the feature space for its images is the alignment loss. We believe this to be the reason why aligning N_2 features turned out always detrimental to performance, as shown in Tab. 9.7 and discussed below.

In the first row, we report the results provided by *plain* AT/DT (no NDA, no aux), in the second those obtained by adversarial training on the features f_2 using the same procedure as described in the previous section for f_1. We can observe that not only adversarial training does not improve (like adversarial training applied to E_1) but it even decreases the overall mIoU of 0.7% compared to the baseline. Finally, in the third row, we report the results obtained by our NDA loss on f_2, like we did successfully on f_1: the NDA loss destroys the feature space of $\mathcal{T}_2$ when applied in this context, as vouched by the drop of 20% in the overall mIoU wrt to plain AT/DT.

We formulate the following hypothesis to explain these results: both adversarial training and NDA perform a comparison between $f_2^{\mathcal{A}}$ and $f_2^{\mathcal{B}}$. While $f_2^{\mathcal{A}}$ are shaped also by the supervision of $\mathcal{T}_2$, $f_2^{\mathcal{B}}$ are evolved only according to the additional loss we impose, as we do not have supervision for $\mathcal{T}_2$ on $\mathcal{B}$. However, E_2 is shared across domains, and therefore may be pushed to produce worse representations for both domains while it tries to accomplish the adversarial objective or the NDA loss minimization for $\mathcal{B}$. If this happens, mappings learned by $G_{1\to2}$ from $f_1^{\mathcal{A}}$ to $f_2^{\mathcal{A}}$ will hallucinate worse features for $\mathcal{T}_2$ on $\mathcal{B}$. To understand why adversarial training leads to a small decrease in performances compared to the use of NDA loss, we ought to consider that adversarial training implies a discriminator that cannot be easily fooled by totally degenerated features, while, without any additional constrain from task supervision, the NDA loss can yield totally collapsed representations.

9.4.6 Aligning $G_{1\to2}$ features

Although feature alignment didn't turn out beneficial when training N_2, one may still expect to obtain better hallucinated features if the representations obtained when transferring $f_1^{\mathcal{A}}$ and $f_1^{\mathcal{B}}$ are aligned. We empirically found out that even though output space aligning strategies deployed when training $G_{1\to2}$ can lead to improvements in performance, input space alignment using

Input Align.	Output Align.	Road	Sidewalk	Walls	Fence	Person	Poles	Vegetation	Vehicles	Tr. Signs	Building	Sky	**mIoU**	**Acc**
-	NDA	42.97	19.60	2.31	1.36	4.21	15.74	18.42	11.77	7.19	36.72	38.99	18.12	43.63
-	Adv	90.80	48.91	**6.16**	11.84	35.32	30.29	**78.78**	71.17	18.51	75.66	75.03	49.32	86.43
-	NDA + Adv	91.03	48.93	6.14	12.24	35.91	31.05	77.93	70.28	16.65	75.50	74.47	49.10	86.28
NDA	Adv	90.67	49.49	5.54	12.29	36.73	28.49	78.28	70.19	22.05	76.47	**76.35**	49.69	86.73
NDA	-	**91.21**	**50.16**	5.14	**13.78**	**36.99**	**32.10**	77.72	**73.38**	**23.47**	**76.67**	72.67	**50.30**	**86.77**

TABLE 9.8: Results of aligning input and output space of $G_{1\to2}$ in a *Dep.* $\to$ *Sem.* scenario. Best results highlighted in bold.

our NDA loss deployed when training N_1 is more effective than them. Moreover, combining input and output space alignment techniques does not lead to further improvements. We performed this ablation study in the *Dep.* $\to$ *Sem.* scenario using Carla as $\mathcal{A}$ and Cityscapes as $\mathcal{B}$. Results of these experiments are reported in Tab. 9.8.

First, we applied our NDA loss to the output-space of $G_{1\to2}$. Similarly to previous section, we notice that, without having supervision on $\mathcal{B}$, the representations extracted from $G_{1\to2}$ for its images collapsed into a single value, yielding a drastic drop in transfer performance (row 1). We also tried to align the output-space features by training $G_{1\to2}$ alongside a discriminator in an adversarial fashion. We wanted to fool the discriminator in order to generate indistinguishable features from $\mathcal{A}$ or $\mathcal{B}$. We noticed that this strategy allow us to reach good overall performances with a 49.32 mIoU on Cityscapes (second row). Moreover, we thought that as adversarial training provides a supervision on $\mathcal{B}$, using the NDA loss in combination with adversarial loss could avoid feature collapse for $\mathcal{B}$ while reaching a better overall alignment between $\mathcal{A}$ and $\mathcal{B}$. However, we notice that the combination of the two losses lead us to slightly worse results than adversarial training only (rows 2 vs 3). Finally, since we noticed that using adversarial loss on the output space lead us to good overall performances, we tested the combination of input space alignment, through NDA loss applied when training N_1, and output space alignment, through adversarial training for $G_{1\to2}$. However, the combination of these two methods achieves worse performance than using only NDA alignment loss on input space (rows 5 vs 6).

Chapter 10

AT/DT in Unsupervised Domain Adaptation for Semantic Segmentation

In previous chapters, we showed that, with AT/DT, it is possible to train a CNN to hallucinate deep features learned to address one task into features amenable to another task related to the former.

Inspired by previous findings, we argue that monocular depth estimation could be an excellent task to gather additional knowledge useful to address semantic segmentation in Unsupervised Domain Adaptation (UDA) settings. First of all, a monocular depth estimation network makes predictions based on 3D cues dealing with the appearance, shape, relative sizes, and spatial relationships of the stuff and things observed in the training images. This suggests that the network has to predict geometry by implicitly learning to understand the scene semantics. Indeed, chapter 13 as well as other works [197, 198, 199] show that a monocular depth estimation network obtains better performances if forced to learn a semantic segmentation task jointly. Moreover, previous experiments in chapter 8 show that depth can help semantics alike. Indeed, it is possible to learn a mapping in both directions between features learned to predict depth and per-pixel semantic labels. It is also worth observing how depth prediction networks tend to extract accurate information for regions characterized by repeatable and simple geometries, such as roads and buildings, which feature strong spatial and geometric priors (e.g. the road is typically a plane in the bottom part of the image) [57, 79, 63, 64]. Therefore, on the one hand, predicting accurately the semantics of such regions from depth information alone should be possible. On the other, a semantic network capable of reasoning on the scene geometry should be less prone to mistakes caused by appearance variations between synthetic and real images, the key issue in UDA for semantic segmentation.

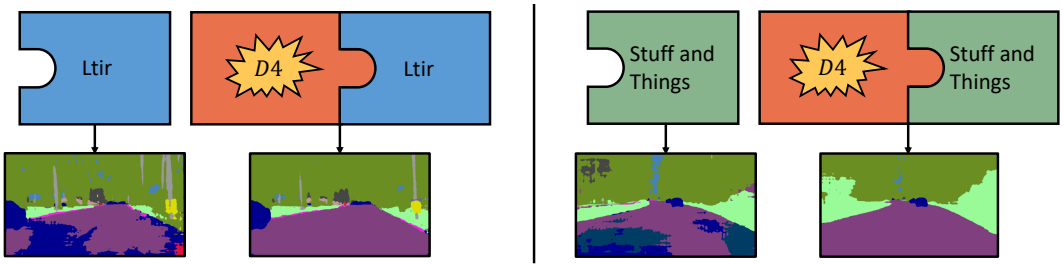

FIGURE 10.1: D4 can be plugged seamlessly into any existing method to improve UDA for Semantic Segmentation. Here we show how the introduction of D4 can ameliorate the performance of two recent methods like Ltir [157] and Stuff and Things [158].

Despite the above observations, injection of geometric cues into UDA frameworks for semantic segmentation has been largely unexplored in literature, except for a few proposals, which either assume the availability of depth labels in the real domain [200], a very restrictive assumption, or can leverage on depth information only in the synthetic domain due to availability of cheap labels [201, 202, 203]. In this respect, we set forth an additional consideration: nowadays, effective self-supervised procedures allow for training a monocular depth estimation network without the need for ground-truth labels [79, 56, 204].

Based on the above intuitions and considerations, in this chapter, we show that, thanks to self-supervision, we can deploy depth information from both synthetic and *unlabelled real* images in order to inject geometric cues in UDA for semantic segmentation. Purposely, we adapt the knowledge learned to pursue depth estimation into a representation amenable to semantic segmentation with AT/DT. As the geometric cues learned from monocular images yield semantic predictions that are often complementary to those attainable by current UDA methods, as illustrated in Figure 10.1 we realize our proposal as a depth-based add-on, dubbed D4 (Depth For), which can be plugged seamlessly into any UDA method to boost its performance.

Finally, we also follow a recent trend in UDA for semantic segmentation, the Self-Training (ST), which consists of further fine-tuning the trained network by its predictions [160, 161, 165, 205, 164, 162]. We propose a novel Depth-Based Self-Training (DBST) approach, which deploys once more the availability of depth information for real images to build a large and varied dataset of plausible samples to be deployed in the Self-Training (ST) procedure.

We will show that our framework can improve many state-of-the-art methods by a large margin in two UDA for semantic segmentation benchmarks, where networks are trained either on GTA5 [21] either or SYNTHIA VIDEO SEQUENCES [24] and tested on Cityscapes [14]. Moreover, we show that our DBST procedure enables us to distill the whole framework into a single ResNet101 [206] and achieve state-of-the-art performance.

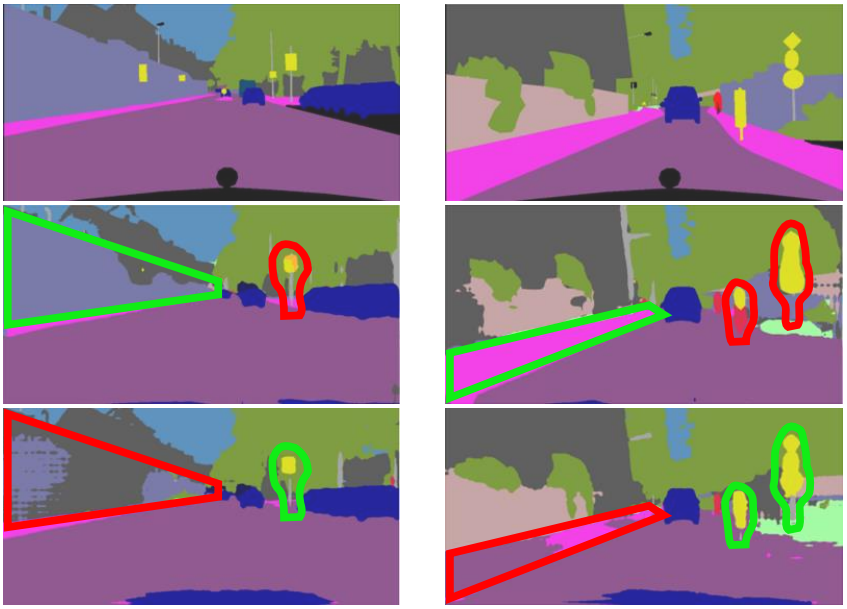

FIGURE 10.2: From top to bottom: ground truth, semantics from depth, semantics by LTIR [157]. The semantic labels predicted from depth are more accurate than those yielded by UDA methods in regularly-shaped objects (such as the *wall* in the left image and the *sidewalk* in the right one), whilst UDA approaches tend top perform better on small objects (see the *traffic signs* in both images).

10.1 Method

In Unsupervised Domain Adaptation (UDA) for semantic segmentation one wishes to solve semantic segmentation in a target domain, $\mathcal{B}$, though labels are available only in another domain, referred to as source domain $\mathcal{A}$. In the following we describe the two ingredients to better tackle this problem. In subsection 10.1.1 we show how to use AT/DT to transfer information from self-supervised monocular depth to semantic segmentation and merge this knowledge with any UDA method (D4-UDA, Depth For UDA). Then, in subsection 10.1.2 we introduce a Depth-Based Self-Training strategy (DBST) to further improve semantic predictions while distilling the whole framework

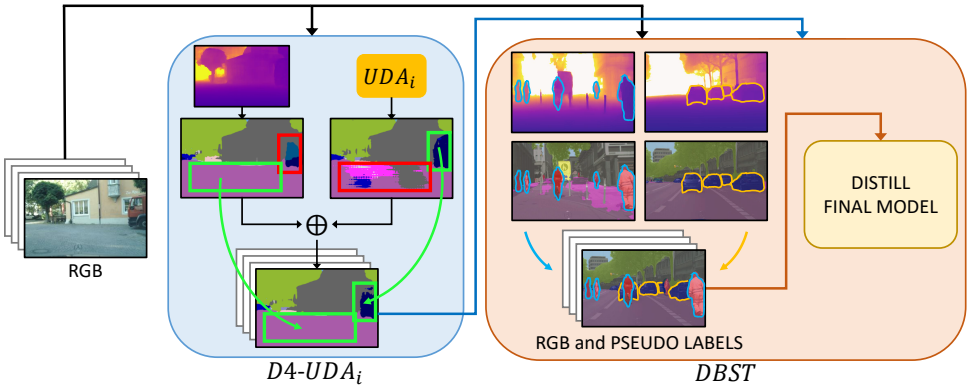

FIGURE 10.3: Overview of D4. RGB images are first processed by two different semantic segmentation engines to produce complementary predictions that are then combined by a weighted sum which accounts for the relative strengths and weaknesses of the two engines (Equation 10.3). During the next step, referred to as DBST, predictions from D4-UDA$_i$ are used to synthesize augmented samples by mixing portions of different images according to depth and semantics. The augmented samples are exploited to train a final model, so as to distill the whole pipeline into a single network.

into a single CNN. We follow the mathematical notation reported in subsection 8.1.1.

10.1.1 D4 (Depth For UDA)

Semantics from Depth. As explained above we want to use AT/DT to transfer knowledge from depth to semantic. However, AT/DT assumes availability of ground-truth labels for the first task (depth estimation in this setting) also in $\mathcal{B}$ (real images). As pointed out previously, this assumption dose not comply with the standard UDA for semantic segmentation problem formulation, which pertains availability of semantic labels for source images ($\mathcal{A}$) alongside with unlabelled target images ($\mathcal{B}$). To address this issue we propose here to rely on *depth proxy-labels* attainable from images belonging to both $\mathcal{A}$ and $\mathcal{B}$ without the need of any ground-truth information. In particular, we propose to deploy one of the recently proposed deep neural networks, such as [79], that can be trained to perform monocular depth estimation based on a self-supervised loss that requires availability of raw image sequences only, i.e. without ground-truth depth labels. Thus, we will follow the following protocol. First, we train a self-supervised monocular depth estimation network on both $\mathcal{A}$ and $\mathcal{B}$. Then, we use this network to generate *depth proxy-labels* for both domains. Finally, we train AT/DT following the

protocol exposed in section 8.1 using the previously computed *depth proxy-labels* to train N_1. In the following, we will refer to such predictions as *semantics from depth* because they concern semantic information extracted from features amenable to perform monocular depth estimation.

Combine with UDA. Figure 10.2 compares semantic predictions obtained from depth by the protocol described in the previous sub-section and from a recent UDA method. The reader may observe a clear pattern: predictions from depth tend to be smoother and more accurate on objects with large and regular shapes, like *road*, *sidewalk*, *wall* and *building*. However, they turn out often imprecise in regions where depth predictions are less informative, like thin things partially overlapped with other objects or fine-grained structures in the background. As UDA methods tend to perform better on such classes (see Figure 10.2), our D4 approach is designed to *combine* the semantic knowledge extracted from depth with that provided by any chosen UDA method in order to achieve more accurate semantic predictions.

Depth information helps on large objects with regular shapes, which usually account for the majority of pixels in an image, and a whole dataset alike. On the contrary, UDA methods perform well in predicting semantic labels for categories that typically concern much smaller fractions of the total number of pixel in an image and dataset, like e.g. the *traffic signs* in Figure 10.2. This orthogonality suggests that a simple yet effective way to combine the semantic knowledge drawn from depth with that provided by UDA methods consists in a weighted sum of predictions, with weights computed according to the frequency of classes in $\mathcal{A}$ (the domain where semantic labels are available). As weights given to UDA predictions ($\mathbf{w}_{uda}$) should be larger for rarer classes, they can be computed as:

$$\mathbf{w}_{uda} = [w^1_{uda}, \dots, w^C_{uda}] \quad \text{where} \quad w^i_{uda} = \frac{1}{ln(\delta + f^i)} \tag{10.1}$$

where C denotes the number of classes and $f^i = \frac{n^i}{tot}$ denotes their frequencies at the pixel level, i.e. the ratio between the number n^i of pixels labelled with class i in $\mathcal{A}$ and the total number tot of labelled pixels in $\mathcal{A}$. Akin to common practice, we set the constant δ to 1.02 in our experiments. Equation 10.1 is the standard formulation introduced in [45] to compute bounded weights inversely proportional to the frequency of classes. Accordingly, weights applied to semantic predictions drawn from depth ($\mathbf{w}_{dep}$) are given by:

$$\mathbf{w}_{dep} = [w^1_{dep}, ..., w^C_{dep}] \quad \text{where} \quad w^i_{dep} = 1 - w^i_{uda}. \tag{10.2}$$

FIGURE 10.4: The rightmost column is a training sample synthesized by copying pixels from left column into the central one. Pixels are chosen according to their semantic class and stacked according to their depths (third row). For example, the two small persons in the left pair are copied behind the one in the middle column. The white pixels in the depth maps represent areas that cannot be copied into other samples due to their depth being too large.

Thus, at each pixel of a given image we propose to combine semantics from depth and predictions yielded by any chosen UDA method as follows:

$$\widehat{\mathbf{y}}_f = \mathbf{w}_{dep} \cdot \phi_T(\widehat{\mathbf{y}}_{dep}) + \mathbf{w}_{uda} \cdot \phi_T(\widehat{\mathbf{y}}_{uda}), \tag{10.3}$$

where $\widehat{\mathbf{y}}_f$ is the final prediction, $\widehat{\mathbf{y}}_{dep}$ and $\widehat{\mathbf{y}}_{uda}$ are the logits associated with semantics from depth and the selected UDA method, respectively, ϕ_T denotes the *softmax* function with a temperature term T that we set to 6 in our experiments.

As illustrated in Figure 10.3, the formulation presented in Equation 10.3 and symbolized as $\bigoplus$ can be used seamlessly to plug semantic information extracted from self-supervised monocular depth into any existing UDA method. We will refer to the combination of a given UDA method with our D4 with the expression D4-UDA. Experimental results reported in subsection 10.2.3 show that, indeed, all recent s.o.t.a. UDA methods do benefit significantly from the complementary geometric cues brought in by D4.

method	Road	Sidewalk	Building	Walls	Fence	Pole	T-light	T-sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	**mIoU**	**Acc**
AdaptSegNet [136]	86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.6	32.5	35.4	3.9	30.1	28.1	42.4	85.6
D4-AdaptSegNet + DBST	93.1	53.0	85.1	42.8	27.3	35.8	43.9	18.5	85.9	39.0	89.9	63.0	31.6	86.6	39.8	36.7	0	42.4	35.0	**50.0**	**90.3**
MaxSquare [207]	88.1	27.7	80.8	28.7	19.8	24.9	34.0	17.8	83.6	34.7	76.0	58.6	28.6	84.1	37.8	43.1	7.2	32.2	34.5	44.3	86.9
D4-MaxSquare + DBST	92.9	51.2	84.7	43.5	22.2	35.7	42.5	20.0	86.2	42.0	90.0	63.7	33.0	86.9	45.5	50.9	0	42.2	41.4	**51.3**	**90.3**
BDL [156]	88.2	44.7	84.2	34.6	27.6	30.2	36.0	36.0	85.0	43.6	83.0	58.6	31.6	83.3	35.3	49.7	3.3	28.8	35.6	48.5	89.2
D4-BDL + DBST	93.2	52.6	86.4	44.1	31.2	36.5	42.4	36.1	86.3	41.0	89.8	63.3	37.4	86.3	42.8	57.8	0	40.3	37.9	**52.9**	**90.7**
MRNET [163]	90.5	35.0	84.6	34.3	24.0	36.8	44.1	42.7	84.5	33.6	82.5	63.1	34.4	85.8	32.9	38.2	2.0	27.1	41.8	48.3	88.3
D4-MRNET + DBST	93.2	51.6	86.1	45.9	24.5	37.9	47.4	40.4	85.3	37.5	89.6	64.7	39.8	85.8	41.1	53.2	8.9	17.1	33.4	**51.7**	**90.0**
Stuff and things* [158]	90.2	43.5	84.6	37.0	32.0	34.0	39.3	37.2	84.0	43.1	86.1	61.1	29.9	81.6	32.3	38.3	3.2	30.2	31.9	48.3	88.8
D4-Stuff and things + DBST	93.3	54.0	86.5	46.4	32.3	37.7	45.2	39.5	85.5	39.4	90.0	63.7	32.8	85.5	32.0	39.5	0	37.7	35.5	**51.4**	**90.5**
FADA [139]	92.5	47.5	85.1	37.6	32.8	33.4	33.8	18.4	85.3	37.7	83.5	63.2	39.7	87.5	32.9	47.8	1.6	34.9	39.5	49.2	88.9
D4-FADA + DBST	93.9	58.2	86.4	45.9	29.6	36.9	44.6	27.0	86.3	39.4	90.0	64.9	41.0	85.8	34.6	51.2	9.9	24.2	37.3	**52.0**	**90.7**
LTIR [157]	92.9	55.0	85.3	34.2	31.1	34.4	40.8	34.0	85.2	40.1	87.1	61.1	31.1	82.5	32.3	42.9	3	36.4	46.1	50.2	90.0
D4-LTIR + DBST	94.2	59.6	86.9	43.9	35.3	36.9	45.7	36.1	86.2	40.6	90.0	65.9	38.2	84.4	33.3	52.4	13.7	46.2	51.7	**54.1**	**91.0**

TABLE 10.1: Results on GTA5→Cityscapes. When available, checkpoints provided by authors are used. * denotes method retrained by us.

10.1.2 DBST (Depth-Based Self-Training)

We describe here our proposal to further improve semantic predictions and distill the knowledge of the entire system into a single network easily deployable at inference time. First, we predict semantic labels for every image in $\mathcal{B}$ by our whole framework (i.e. D4 alongside a selected UDA method, referred to as D4-UDA); then, we use these labels to train a new model on $\mathcal{B}$. This procedure, also known as Self-Training [159], has become popular in recent UDA for semantic segmentation literature [160, 161, 165, 205, 164, 162] and consists in training a model by its own predictions, referred to as *pseudo-labels*, sometimes through multiple iterations. The novelty of our approach concerns the peculiar ability to leverage on the depth information available for the images in $\mathcal{B}$ to generate plausible new samples.

Running D4-UDA on $\mathcal{B}$ yields semantic pseudo-labels for every image in $\mathcal{B}$. Yet, as described in subsection 10.1.1 (*Semantics from Depth*), each image in $\mathcal{B}$ is also endowed with a depth prediction, provided by a self-supervised monocular depth estimation network. We can take advantage of this information to formulate a novel depth-aware data augmentation strategy whereby portion of images and corresponding pseudo-labels are *copied* onto others so as to synthesize samples for the Self-Training procedure. The crucial difference between similar approaches presented in [208, 166] and ours consists in the deployment of depth information to steer the data augmentation procedure towards generating plausible samples. Indeed, a first intuition behind our method deals with semantic predictions being less accurate for objects distant from the camera: as such predictions play the role of labels in Self-Training, we are lead to prefer picking closer rather than distant regions in order to generate training samples. Moreover, we reckon certain kinds of objects, like e.g. persons, vehicles, poles, traffic signs, to be more plausibly transferable across different images as they tend to be small and less bound to specific spatial locations. For example, a piece of road or building from another image would more unlikely merge seamlessly into a given one with respect to a pedestrian or vehicle.

Given N randomly selected images x^n from $\mathcal{B}$, with $n \in \{1, \dots, N\}$, paired with semantic pseudo-labels s^n and depth predictions d^n, we augment x^1, by copying on it pixels from the set $\mathcal{X}^{src} = \{x^2, \cdots, x^N\}$. For each pixel of the augmented image we have N possible candidates, one from x^1 itself and $N-1$ from the images in $\mathcal{X}^{src}$. We filter such candidates according to two criteria: the predicted depth should be lower than a threshold t and the semantic prediction should belong to a predefined set of classes, C^*. Hence, we

define the set of depths of the filtered candidates at each spatial location p as:

$$\mathcal{D}_p = \{d_p^n \mid d_p^n < t \wedge s_p^n \in C^*\} \quad n \in \{1, \dots, N\}. \tag{10.4}$$

In our experiments, for each image the depth threshold t is set to the 80^{th} percentile of the depth distribution, so as to avoid selecting pixels from the farthest objects in the scene, while C^* contains classes: *pole*, *traffic light*, *traffic sign*, *person*, *car*, *rider*, *truck*, *bus*, *train*, *motorbike*, *bicycle*, *wall* and *fence*, which we found more amenable to synthesize plausible training samples. Then, we synthesize a new image x^z and corresponding pseudo-labels s^z, by assigning at each spatial location p the candidate with the lowest depth, so that objects belonging to different images do overlap plausibly into the synthesized one:

$$x_p^z = x_p^k \quad s_p^z = s_p^k \tag{10.5}$$

$$k = \begin{cases} 1, & \mathcal{D}_p = \emptyset \\ n \text{ s.t. } d_p^n = \min \mathcal{D}_p, & \mathcal{D}_p \neq \emptyset \end{cases} \tag{10.6}$$

In Figure 10.4 we depict our depth-based procedure to synthesize new training samples, considering, for the sake of simplicity, the case where N is 2.

Hence, with the procedure detailed above, we synthesize an augmented version of $\mathcal{B}$, used to distill the whole D4-UDA framework into a single model by a Self-Training process. This dataset is much larger and exhibits more variability than the original $\mathcal{B}$. Due to its reliance on depth information, we dub our novel technique as DBST (Depth-Based Self-Training). The results reported in subsection 10.2.3 prove its remarkable effectiveness, both when used as the final stage following D4 as well as when deployed as a standalone Self-Training procedure applied to any other UDA for semantic segmentation method.

10.2 Experiments

10.2.1 Implementation Details

Network Architectures. We use Monodepth2 [79] to generate depth proxy-labels for the procedure described in Sec. 10.1.1. We change the AT/DT

Method	Sky	Building	Road	Sidewalk	Fence	Vegetation	Pole	Car	T-Sign	Person	Bicycle	T-Light	**mIoU**	**Acc**
AdaptSegNet* [136]	75.6	78.0	89.7	28.5	3.4	76.0	28.5	85.1	27.2	55.3	46.6	0	49.5	86.9
D4-AdaptSegNet + DBST	88.0	80.2	95.1	66 .8	5.7	80.4	33.2	87.3	33.2	60.9	52.4	0	**56.9**	**90.7**
MaxSquare* [207]	72.4	79.2	89.2	36.0	4.6	75.7	31.5	84.9	30.7	55.8	45.8	8.6	51.2	87.3
D4-MaxSquare + DBST	88.1	80.1	95.0	66.6	6.0	79.4	34.4	86.7	36.3	60.8	47.2	8.4	**57.4**	**90.6**
MRNET* [163]	84.6	79.7	93.9	56.3	0	80.5	35.4	88.9	27.2	59.4	56.3	0	54.5	90.0
D4-MRNET + DBST	88.3	79.0	95.0	67.0	5.9	78.6	36.2	86.7	31.0	60.6	47.5	0	**56.3**	**90.2**

TABLE 10.2: Results on the SYNTHIA-SEQ→Cityscapes benchmark. * denotes method retrained by us.

framework respect to the one used in chapter 8 to this new setting by deploying the popular Deeplab-v2 [209] for depth estimation and semantic segmentation networks. Both networks consist of a backbone and an ASPP module [209], which substitute, respectively, the encoder and decoder used in chapter 8. The backbone is implemented as a dilated ResNet50 [193]. We also use the transfer function without the downsampling and upsampling operations as in section 9.2. More precisely, the transfer function is realized as a simple 6-layers CNN with kernel size 3×3 and Batch Norm [194]. Following the recent trend in UDA for semantic segmentation [136, 207, 156, 163, 158, 139, 157], during DBST we train a single Deeplab-v2 [209] model, with a dilated ResNet101 pre-trained on Imagenet [210] as backbone.

Training Details. Our pipeline is trained on one NVIDIA Tesla V100 GPU with 16GB of memory. In every training and test phase we resize input images to 1024×512, with the exception of DBST, when we first perform random scaling and then random crop with size 1024×512. During DBST we use also color jitter to avoid overfitting on the pseudo-labels. The depth and the transfer network are optimized by Adam [211] with batch size 2 for 70 and 40 epochs, respectively, while the semantic segmentation network is trained by SGD with batch size 2 for 70 epochs.The final model obtained by DBST is trained again with SGD, batch size 3 and for 30 epochs. We adopt the One Cycle learning rate policy [212] in every training, setting the maximum learning rate to 10^{-4} but in DBST, where we use 10^{-3}.

Method	UDA	D4-UDA	UDA + DBST	D4-UDA + DBST
AdaptSegNet [136]	42.4	46.7	46.0	**50.0**
MaxSquare [207]	44.3	48.0	48.1	**51.3**
BDL [156]	48.5	49.6	51.7	**52.9**
MRNET [163]	48.3	49.6	50.0	**51.7**
Stuff and Things* [158]	48.3	49.1	50.4	**51.4**
FADA [158]	49.3	49.9	51.4	**52.0**
LTIR [157]	50.2	51.1	53.1	**54.1**

TABLE 10.3: Impact on performance of the two components of our proposal (D4, DBST) when applied separately or jointly to selected UDA methods on GTA5→Cityscapes. * indicates that the method was retrained by us. Results are reported in mIoU.

10.2.2 Datasets

We briefly describe the datasets adopted in our experiments, pointing the reader to the Supplementary Material for additional details. We use two synthetic datasets: GTA5 [21, 171] and SYNTHIA [24]. Since our method requires video sequences to train Monodepth2 [79], we use the split SYNTHIA VIDEO SEQUENCES (SYNTHIA-SEQ) in the experiments involving the SYNTHIA dataset. As for real images, we leverage on the Cityscapes dataset [14].

10.2.3 Results

We report here experimental results obtained in two domain adaptation benchmarks, i.e. GTA5→Cityscapes and SYNTHIA-SEQ→Cityscapes, which show how the combination with our D4 method allows to boost performance of recent UDA for semantic segmentation approaches.

GTA5→Cityscapes. Table 10.1 reports results on the most popular UDA benchmark for semantic segmentation, i.e. GTA5→Cityscapes, where methods are trained on GTA5 and tested on Cityscapes. We selected the most relevant UDA approaches proposed in the last years [136, 207, 156, 163, 158, 139, 157], using training checkpoints provided by authors whenever available. We report per-class and overall results in terms of mean intersection over union (mIoU) and pixel accuracy (Acc), when each method is either used stand-alone or deployed within our proposal (i.e. D4 + DBST). The reader may notice how every recent UDA method does improve considerably if combined with our proposal, despite the variability of their stand-alone performances. Indeed, Adaptsegnet [136], which yields about 42% in

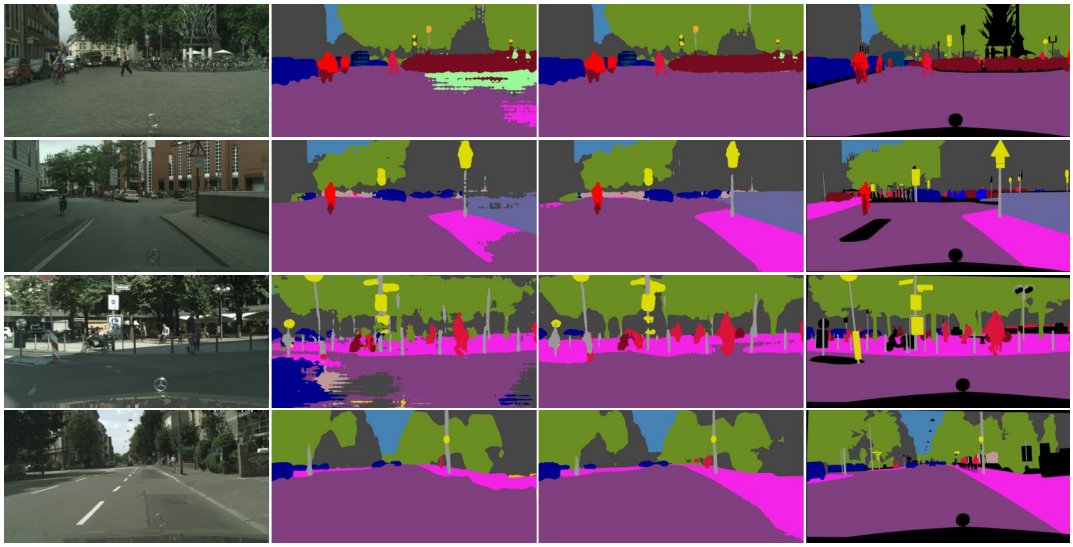

FIGURE 10.5: From left to right: RGB image, prediction from UDA method, prediction from D4-UDA + DBST, GT. The top two rows deal with GTA5→Cityscapes, the other two with SYNTHIA-SEQ→Cityscapes. Selected methods are, from top to bottom: LTIR [157], BDL [156], MaxSquare [207] and MRNET [163]. In all these examples our proposal can ameliorate dramatically the output of the given stand-alone method, especially on classes featuring large and regular shapes, like *road* in rows 1-3, *sidewalk* in rows 2-4 and *wall* in row 2.

terms of mIoU, reaches 50% when embedded into our framework. Likewise, LTIR [157], currently considered one of the s.o.t.a. UDA methods, improves in mIoU from 50.2% to 54.1%. Analyzing Table 10.1 more in detail, we can observe that our method produces a general improvement for all classes, although we experience a certain performance variability for some of them (such as *train*, *motorbike* and *bicycle*), probably due to noisy pseudo-labels used during DBST. Conversely, our method yields consistently a significant gain on classes characterized by large and regular shapes, namely *road*, *sidewalk*, *building*, *wall* and *sky*, which validates our intuition on the effectiveness of a) the geometric cues derivable from depth to predict the semantics of these kind of objects and b) the methodology we propose to leverage on these additional cues in UDA settings. This behavior is also clearly observable from a qualitative perspective in Fig. Figure 10.5. Finally, we point out that, to the best of our knowledge, the performance figure obtained by D4-LTIR + DBST, i.e. 54.1% mIoU (last row of Table 10.1) establishes the new state-of-the-art for the GTA5→Cityscapes benchmark.

SYNTHIA-SEQ→Cityscapes. Akin to common practice in literature we present results also on the popular SYNTHIA dataset. Due to our pipeline

Method	D4-UDA	Self-Training	DBST
D4-BDL [156]	49.6	50.1	**52.9**
D4-MRNET [163]	49.6	50.3	**51.7**
D4-Stuff and Things [158]	49.1	49.4	**51.4**
D4-FADA [139]	49.9	50.0	**52.0**
D4-LTIR [157]	51.1	51.5	**54.1**

TABLE 10.4: Comparison between DBST and baseline Self Training. Results are reported in terms of mIoU on GTA5→Cityscapes.

requiring video sequences to train the self-supervised monocular depth estimation network, we select the SYNTHIA VIDEO SEQUENCES split for training and the Cityscapes dataset for testing. We will refer to this setting as to SYNTHIA-SEQ→Cityscapes. To address SYNTHIA-SEQ→Cityscapes we re-trained the UDA methods for which the code is available and the training procedure is more affordable in terms of memory and run-time requirements, namely AdaptSegNet [136], MaxSquare [207] and MRNET [163]. The results in Table 10.2 show that all the selected UDA approaches exhibit a substantial performance gain when coupled with our proposal, with a general improvement in all classes. In particular, similarly to the results obtained in GTA5→Cityscapes, we observe a consistent improvement for classes related to objects with large and regular shapes (as depicted also in Figure 13.4), with the only exception of a slight performance drop for the class *building* when using MRNET [163] (last row of Table 10.2). We argue that our approach is relatively less effective with MRNET [163] as, unlike AdaptSegNet [136] and MaxSquare [207], it yields already satisfactory results in those classes which are usually improved by the geometric clues injected by D4.

10.2.4 Ablation study

In this section we report the main ablation studies of our work. Additional analysis in the supplementary material.

Impact of the individual contributions. In Table 10.3, we analyse the impact on the performance yielded by the two main contributions of this approach, i.e. injection of geometric cues into UDA methods by D4 and DBST. Purposely, we select the GTA5→Cityscapes benchmark and, for each of the UDA methods considered in Table 10.1, we report the mIoU figures obtained

by a) using the stand-alone UDA method, b) combining it with D4, c) applying DBST directly on the stand-alone method and d) embedding the method into our full pipeline. Thus, we can observe that each of our novel contributions allow for improving the performance of the most recent UDA methods by a large margin. Moreover, as shown in the last column of Table 10.3, when deployed jointly so as to realize our whole proposal, D4 and DBST further enhance the performances of the any selected method suggesting that they are complementary.

Effectiveness of DBST. In Table 10.4 we compare our DBST against an alternative Self-Training procedure. We select five UDA methods with appealing performances on GTA5→Cityscapes and combine each of them with our D4, reporting the obtained mIoU figures in the second column. Seeking for a viable Self-Training algorithm, we considered those originally proposed together with each method. Yet, we found experimentally that none of the original Self-Training procedure could yield a further performance improvement. This can be explained by the fact that these Self-Training algorithms rely on modeling the uncertainty of the original method for which they are designed and this tends to fail when UDA methods get merged with D4. This leads us to consider a baseline Self-Training procedure, which consists in simply fine-tuning the model by its own predictions on the images of the target domain. Thus, in Table 10.4 we compare this baseline Self-Training to DBST (third and fourth column respectively). The figures in Table 10.4 highlight how our DBST procedure consistently provides a much larger performance improvement than the considered baseline Self-Training. As the only difference between the two procedures concerns the dataset employed in the fine-tuning process, the results in Table 10.4 prove the effectiveness of DBST in generating a large and varied set of plausible training samples more amenable to Self-Training than the original images belonging to the target domain.

Chapter 11

Final Remarks

In this part of the thesis, we have shown that visual tasks are tightly correlated one to another, and this correlation can be used to ameliorate the need for labeled data. In chapter 8 we proposed AT/DT showing that it is possible to learn a mapping function to transform deep representations suitable for specific tasks into others amenable to different ones. AT/DT allows for leveraging on easy to annotate domains to solve tasks in scenarios where annotations would be costly. We have demonstrated the validity of our framework with thorough experiments in several tasks and domains.

Then, in chapter 9 we extended the base version of AT/DT addressing its main weaknesses. In particular, to improve AT/DT performance, we introduced two novel feature alignment strategies operating at the tasks and domains level. Precisely, we have shown that for dense tasks such as semantic segmentation and depth estimation, encoding the scene structure in the latent space via an auxiliary edge detection task is key to ease generalization and obtain sharp predictions.

Finally, in chapter 10 we showed a practical application of AT/DT demonstrating that it is complementary to the whole domain adaptation literature and it can be integrated with it. In particular, we have shown that it is possible to exploit self-supervised monocular depth estimation in UDA problems to obtain accurate semantic predictions for objects with strong geometric priors (like roads and buildings). These findings highlight the possibility of transferring the knowledge gathered by auxiliary tasks learned by self-supervision to better tackle UDA for semantic segmentation, paving the way for novel research directions.

In conclusion, we think that these results can be extended in generality. We believe that the relationships among visual tasks to reduce the need for data is a novel and appealing research direction. Indeed, we can create more robust and general deep perception models, extending the applicability of these methods in the real world.

Part III

Comprehensive Scene Understanding with Multi-Task Learning

Chapter 12

Initial Remarks

In the last part of the thesis, we showed that visual tasks are tightly connected and that we can leverage these correlations to decrease the need for training labels. Nevertheless, here, we want to make a further step by asking ourselves a new question: *Can we deploy tasks' relationships to build more robust and performant models for comprehensive scene understanding?* In other words, by merging more tasks, how much can we gain in terms of accuracy, memory efficiency, computational time, and data requirements?

To address this question, in chapter 13 we report a preliminary study on monocular depth estimation and semantic segmentation. In previous chapters, we have already shown that these two tasks are tightly correlated one to another, and we can leverage this connection to improve the generalization of neural networks. However, we would like to investigate if we can build a better architecture by exploiting these synergies in a single domain scenario. We consider monocular self-supervised depth estimation because it can be trained efficiently without any ground-truth. We presuppose to have stereo-pairs as supervision. On the other hand, we assume to have few labels for semantic segmentation for this early study. From the results of this chapter, we can evince that we can boost depth estimation performances thanks to semantic segmentation. Indeed, the depth network enriched by semantic information can reason about the high-level content of the image, mitigating problems such as texture-less areas or ambiguous object's borders.

However, we would like to further investigate the advantages deriving from multi-task awareness. Thus, in chapter 14 we make some further steps starting from previous findings. First of all, we move to self-supervised depth estimation from video sequences because it is easier to utilize in real scenarios respect to stereo pairs (i.e. we need a traditional monocular camera). In this type of scenario comes straightforward to reason dynamic scenes

and motion. Hence, we consider the main pixel-wise tasks that can be performed from a monocular video sequence: depth estimation, semantic segmentation, motion segmentation, camera pose estimation, and optical flow. Thus, we propose a new framework to address jointly all these scene understanding tasks. We demonstrate that by leveraging on the dependencies among tasks, we can obtain a more robust, efficient, and accurate model than those trained on each task in isolation. Moreover, by leveraging self-supervision for geometry and distillation for semantics, our multi-task framework can be trained without using even a single manually annotated sample.

In the following section, we discuss some related works for this part of the thesis.

12.1 Multi-task Learning

The goal of multi-task learning is to solve many tasks simultaneously. By pursuing this rather than solving the tasks independently, a neural network should use more information to obtain more robust and reliable predictions. Moreover, we obtain models that can produce several outputs in a single run [213, 214, 215].

Many works try to tackle several tasks jointly [213, 189]. For example, [189] showed that by learning to weigh each task loss correctly, multi-task learning methods could outperform separate models trained individually. Recently, [216] proposes a technique to improve the performances of multiple single task networks imposing consistency across them during training. Moreover, to effectively accomplish multi-task learning, the relationship between the deployed tasks must be considered. Again Taskonomy [176] provides hints on this topic, building a correlation graph among visual tasks. In this part of the thesis, we address the possibility of exploiting the synergy between the geometry and semantics (chapter 13) or geometry, semantics, and motion of a scene (chapter 14).

Semantic segmentation and depth estimation. We can infer a scene's depth by a single image mostly because of context and prior semantic knowledge. Recent works explored the possibility to learn both tasks with either full supervision [217, 32, 218, 219, 220, 197, 221] or supervision concerned with semantic labels only [222]. Unlike other works, in chapter 13 we propose the first architecture trained by self-supervision on stereo pairs and exploiting semantic labels.

Semantic segmentation and optical flow. Joint learning of semantic segmentation and optical flow estimation has been already explored [223]. Moreover, scene segmentation [224, 225] is required to disentangle potentially moving and static objects for focused optimizations. Differently, [226] leverages on optical flow to improve semantic predictions of moving objects. Peculiarly w.r.t. previous work, in chapter 14 we propose a novel self-distillation training procedure guided by semantics to improve occlusion handling in a comprehensive scene understanding multi-task framework.

Chapter 13

Geometry and Semantics

In this chapter we propose to train a CNN architecture to perform both semantic segmentation and depth estimation from a single image. By optimizing our model jointly on the two tasks, we enable it to learn a more effective feature representation which yields improved depth estimation accuracy. We rely on the unsupervised image re-projection loss [57] to pursue depth prediction whilst we let the network learn semantic information from the observed scene by supervision signals from pixel-level ground-truth semantic maps. Thus, with respect to recent work [57], our proposal requires semantically annotated imagery, thereby departing from a totally unsupervised towards a semi-supervised learning paradigm (*i.e.* unsupervised for depth and supervised for semantics). Yet, though manual annotation of per-pixel semantic labels is tedious, it is much less prohibitive than collecting groundtruth depths. Besides, while the former task may be performed off-line after acquisition, as recently proposed for some images of the KITTI dataset [227], one may very unlikely obtain depth labels out of already collected frames.

Thus, we propose the integration of unsupervised monocular depth estimation with supervised semantic segmentation. By applying this novel paradigm, we improve a state-of-the-art encoder-decoder depth estimation architecture [57]. We introduce an additional decoder stream based on the same features as those deployed for depth estimation and trained for semantic segmentation; thereby, the overall architecture is trained to optimize both tasks jointly. Moreover, we also propose a novel loss term, the *cross-domain discontinuity* loss $\mathcal{L}_{cdd}$, aimed at enforcing spatial proximity between depth discontinuities and semantic contours.

Experimental results on the KITTI dataset prove that tackling the two tasks jointly does improve monocular depth estimation. For example, Figure 13.1 suggests how recognizing objects like cars (c) can significantly ameliorate depth estimation (d) with respect to a depth-from-mono approach

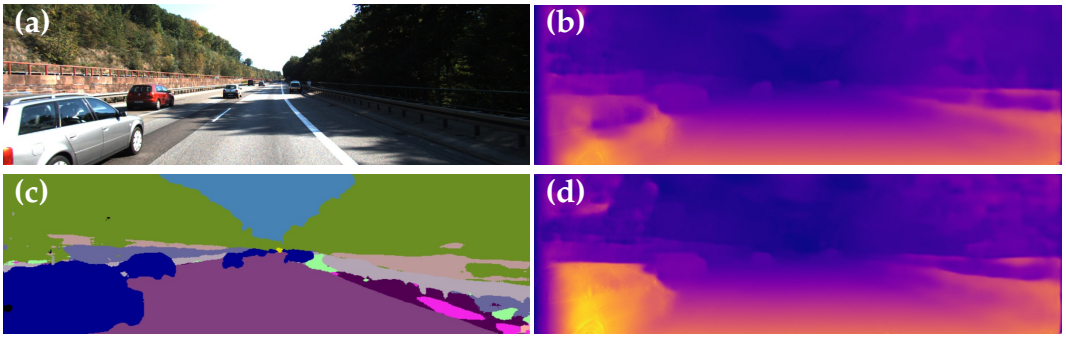

FIGURE 13.1: Joint depth from mono and semantic segmentation. (a) Input image, (b) depth map by state-of-the-art method [57], (c) semantic and (d) depth maps obtained by our network.

lacking any awareness about scene semantics (b). It is also worth highlighting that, unlike all previous unsupervised frameworks in this field, our proposal delivers not only the depth map (Figure 13.1 (d)) but also the semantic segmentation of the input image (Figure 13.1, (c)).

13.1 Method

Estimating the distance of objects from a camera through a single acquisition is an ill-posed problem. While other techniques can effectively measure depth based on features extracted from different view points (e.g. , binocular stereo allows for triangulating depth from point matches between two synchronized frames), monocular systems cannot rely on geometry constraints to infer distance unambiguously. Despite this lack of information, modern deep learning monocular frameworks achieved astounding results by learning effective feature representations from the observed environment. Common to latest work in this field [50, 51, 57, 66] is the design of deep encoder-decoder architectures, with a first contractive portion progressively decimating image dimensions to reduce the computational load and increase the receptive field, followed by an expanding portion which restores the original input resolution. In particular, the encoding layers learn a high level feature representation crucial to infer depth. Although it is hard to tell what kind of information the network is actually learning at training time, we argue semantics to play an important role. Recent works like [57, 66] somehow support this intuition. Indeed, although the authors trained and evaluated their depth estimators on the KITTI dataset [178], a preliminary training on CityScapes [109] turned out beneficial to achieve the best accuracy with

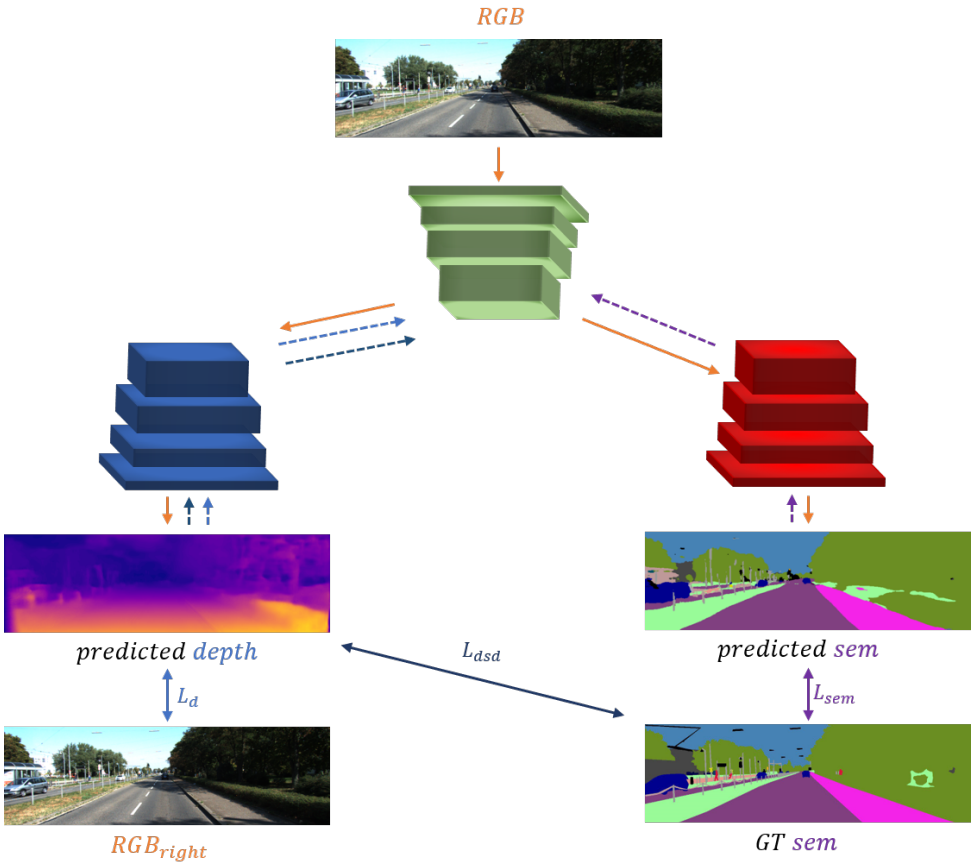

FIGURE 13.2: Schematic representations of the proposed network architecture and semi-supervised learning framework. A single encoder (green) is shared between a depth (blue) and a semantic (red) decoder. The depth decoder is optimized to minimize $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$, the semantic decoder to minimize $\mathcal{L}_s$.

both frameworks, despite the very different camera setup between the two datasets. Common to the datasets is, in fact, the kind of sensed environment and, thus, the overall semantics of the scenes under perception. This observation represents the main rationale underpinning our proposal. By explicitly training the network to learn the semantic context of the sensed environment we shall expect to enrich the feature representation resulting from the encoding module and thus obtain a more accurate depth estimation. This may be realized by a deep model in which a single encoder is shared between two decoders in charge of providing, respectively, a depth map and a semantic segmentation map. Accordingly, minimization of the errors with respect to pixel-level semantic labels provides gradients that flow back into the encoder at training time, thereby learning a shared feature representation aware of both depth prediction as well as scene semantics. According to our claim, this should turn out conducive to better depth prediction.

Inspired by successful attempts to predict depth from a single image, we design a suitable encoder-decoder architecture for joint depth estimation and semantic segmentation. The encoder is in charge of learning a rich feature representation by increasing the receptive field of the network while reducing the input dimension and computational overhead. Popular encoders for this task are VGG [228] and ResNet50 [229]. The decoder restores the original input resolution by means of up-sampling operators followed by 3×3 convolutions linked by means of skip connections with the encoder at the corresponding resolution. As illustrated in Fig. 13.2, to infer both depth and semantics we keep relying on a single encoder (green) and replicate the decoder to realize a second estimator. The two decoders (blue, red) do not share weights and are trained to minimize different losses, which deal with the depth prediction (blue) and semantic segmentation (red) tasks. While the two decoders are updated by different gradients flows, the shared encoder (green) is updated according to both flows, thereby learning a representation optimized jointly for the two tasks. We validate our approach by extending the architecture proposed by Godard *et al.*[57] for monocular depth estimation: the encoder produces two inverse depth (i.e., disparity) maps by processing the left image of a stereo pair. Then, the right image is used to obtain supervision signals by warping the left image according to the estimated disparities, as explained in the following section.

Figure 13.3 shows how the shared representation used to jointly tackle both tasks enables to reconstruct better shapes when estimating depth (e) thanks to the semantic context (d) learned by the network compared to standalone learning of depth (c) as in [57].

13.1.1 Loss functions

To train the proposed architecture, we rely on the following multi-task loss function

$$\mathcal{L}_{tot} = \alpha_d \mathcal{L}_d + \alpha_s \mathcal{L}_s + \alpha_{cdd} \mathcal{L}_{cdd} \tag{13.1}$$

which consists in the weighted sum of three terms, namely the *depth* ($\mathcal{L}_d$), *semantic* ($\mathcal{L}_s$) and *cross-domain discontinuity* ($\mathcal{L}_{cdd}$) terms. As shown in in Fig. 13.2, each term back-propagates gradients through a different decoder: in particular, $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$ through the depth (blue) decoder whilst $\mathcal{L}_s$ through the semantic (red) decoder. All gradients then converge so to flow back into the shared (green) encoder.

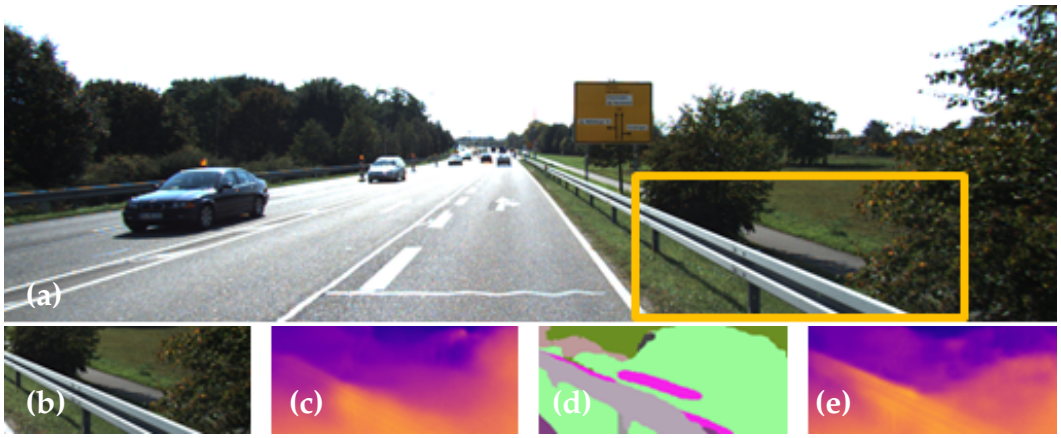

FIGURE 13.3: Example of improved depth estimation enabled by semantic knowledge. (a) input image, (b) region extracted from the scene, (c) depth map predicted by [57], depth (d) and semantic (e) maps predicted by our framework. We can clearly notice how the the structure of the guard rail is better preserved by our method (e) compared to [57] in (c).

Depth term

The depth term, $\mathcal{L}_d$, in our multi-task loss is computed according to the unsupervised training paradigm proposed by Godard *et al.*[57]:

$$\mathcal{L}_d = \beta_{ap}(\mathcal{L}^l_{ap} + \mathcal{L}^r_{ap}) + \beta_{ds}(\mathcal{L}^l_{ds} + \mathcal{L}^r_{ds}) + \beta_{lr}(\mathcal{L}^l_{lr} + \mathcal{L}^r_{lr}) \tag{13.2}$$

where the loss consists in the weighted sum of three terms, namely the *appearance*, *disparity smoothness* and *left-right consistency* terms. The first term measures the image re-projection error by means of the SSIM [230] and L1 difference between the original and warped images, I and $\tilde{I}$:

$$\mathcal{L}^l_{ap} = \frac{1}{N}\sum_{i,j}\gamma\frac{1 - SSIM(I^l_{i,j}, \tilde{I}^l_{i,j})}{2} + (1-\gamma)||(I^l_{i,j} - \tilde{I}^l_{i,j})|| \tag{13.3}$$

The smoothness term penalizes large disparity differences between neighboring pixels along the x and y directions unless these occur in presence of strong intensity gradients in the reference image I

$$\mathcal{L}^l_{ds} = \frac{1}{N}\sum_{i,j}|\delta_x d^l_{i,j}|e^{-||\delta_x I^l_{i,j}||} + |\delta_y d^l_{i,j}|e^{-||\delta_y I^l_{ij}||} \tag{13.4}$$

Finally, the left-right consistency enforces coherence between the predicted disparity maps, d^l and d^r, for left and right images:

$$\mathcal{L}^l_{lr} = \frac{1}{N}\sum_{i,j}|d^l_{i,j} - d^r_{i,j+d^l_{i,j}}| \tag{13.5}$$

As proposed in [57], in our learning framework $\mathcal{L}_d$ is computed at four different scales.

Semantic term

The semantic term $\mathcal{L}_s$ within our total loss is given by the standard cross-entropy between the predicted and groundtruth pixel-wise semantic labels:

$$\mathcal{L}_s = \mathcal{C}(p_t, \overline{p}_t) = H(p_t, \overline{p}_t) + KL(p_t, \overline{p}_t) \tag{13.6}$$

where H denotes the entropy and KL the $KL-$divergence. The semantic term, $\mathcal{L}_s$, is computed at full resolution only.

Cross-domain discontinuity term

We also introduce a novel cross-task loss term aimed at enforcing an explicit link between the two learning tasks by leveraging on the ground-truth pixel-wise semantic labels to improve depth prediction. We found that the most effective manner to realize this consists in deploying the observation that depth discontinuities are likely to co-occur with semantic boundaries. Accordingly, we have designed the following *cross-domain discontinuity*, $\mathcal{L}_{cdd}$, term:

$$\mathcal{L}_{cdd} = \frac{1}{N}\sum_{i,j} sgn(|\delta_x sem^l_{i,j}|)e^{-||\frac{\delta_x d^l_{i,j}}{d^l_{i,j}}||} + sgn(|\delta_y sem^l_{i,j}|)e^{-||\frac{\delta_y d^l_{i,j}}{d^l_{i,j}}||} \tag{13.7}$$

where sem denotes the ground-truth semantic map and d the predicted disparity map. Differently from the smoothness term $\mathcal{L}^l_{ds}$ in the disparity domain, the novel $\mathcal{L}_{cdd}$ term detects discontinuities between semantic labels encoded by the sign of the absolute value of the gradients in the semantic map. The idea behind this loss is that there should be a gradient peak between adjacent pixels belonging to different classes. Nevertheless, we do not care about its magnitude since the numeric labels do not have any mathematical meaning.

13.2 Experimental results

In this section, we compare the performance of our semi-supervised joint depth estimation and semantic segmentation paradigm with respect to the proposal by Godard *et al.*[57]. As discussed in section 13.1, our method as well as the baseline used in our experiments, *i.e.* [57], require rectified stereo pairs at training time. Suitable datasets for this purpose are thus CityScapes

[109] and KITTI [108], which provide a large number of training samples, *i.e.* about 23k and 29k rectified stereo pairs respectively. However, our method requires also pixel-wise groundtruth semantic labels at training time, which limits the actual amount of training samples available for our experiments. In particular, CityScapes includes about 3k finely annotated images, while the KITTI 2015 benchmark made available pixel-wise semantic groundtruths for about 200 images [227]. Therefore, to carry out a fair evaluation of the actual contribution provided by semantic information in the depth-from-mono task to the baseline fully unsupervised approach, we trained both methods based on the reduced datasets featuring stereo pairs alongside with semantically annotated left frames.

13.2.1 Implementation details

We adhere to the original training protocol by [57], scheduling 50 epochs on the CityScapes dataset and 50 further on the KITTI 2015 images. For quantitative evaluation, we split the KITTI 2015 dataset into train and test sets, providing more details in the next section. We train on 256×512 images using a batch dimension of 2, we set the previously introduced hyper-parameters as follows: $\alpha_d = 1$, $\alpha_s = 0.1$, $\alpha_{cdd} = 0.1$, $\beta_{ap} = 1$, $\beta_{lr} = 1$, $\beta_{ds} = \frac{1}{r}$ (being r the down-sampling factor at that resolution) and $\gamma = 0.85$. Models are trained using Adam optimizer [211], with $\beta_1 = 0.9$, $\beta_1 = 0.999$ and $\epsilon = 10^{-8}$. The initial learning rate is set to 10^{-4}, halved after 30 and 40 epochs. We perform data augmentation on input RGB images, in particular random gamma, brightness and color shifts sampled within the ranges [0.8,1.2] for gamma, [0.5,2.0] for brightness, and [0.8,1.2] for each color channel separately. Moreover we flip images horizontally with a probability of 50%. If the flip occurs, the right image in the stereo pair becomes the new reference image and we do not provide supervision signals from semantics (as right semantic maps are not available in the datasets). We implemented our network with both VGG and ResNet50 encoders, as in [57]. The semantic decoder adds about 20.5M parameters, resulting in nearly 50 and 79 million parameters for the two models (31 and 59, respectively, for [57]).

13.2.2 Monocular depth estimation: evaluation on KITTI 2015

We quantitatively assess the effectiveness of our proposal on the KITTI 2015 training dataset for stereo [178]. It provides 200 synchronized pairs of images together with groundtruth disparity and semantic maps [227]. As already

mentioned, to carry out a fair comparison between our approach and [57], we can use only these samples and thus the numerical results reported in our paper cannot be compared directly with those in [57]. Then, we randomly split the 200 pairs from KITTI into 160 training samples and 40 samples used only for evaluation[1]. We measure the accuracy of the predicted depth maps after training for 50 epochs on CityScapes and then fine-tuning for 50 more epochs on the samples selected from KITTI.

Table Table 13.1 reports quantitative results using VGG or ResNet50 as backbone encoder. Each model, one per row in the table, is trained with four different strategies:

- $\mathcal{L}_d$ uses only the depth term as loss (*i.e.*, equivalently to the baseline approach by Godard *et al.*[57]).
- $\mathcal{L}_d$+$\mathcal{L}_s$ adds the semantic term to the depth term.
- $\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$ minimizes our proposed total loss function (Equation 13.1).
- $\mathcal{L}_d$+$\mathcal{L}_{cdd}$ minimizes only the losses dealing with the depth decoder.

The table provides results yielded by the four considered networks according to standard performance evaluation metrics [57] computed between estimated depth d and groundtruth D.

This ablation highlights how introducing the second decoder trained to infer semantic segmentation maps, significantly improves depth prediction according to all performance metrics for both type of encoder. Moreover, adding the cross-domain discontinuity term, $\mathcal{L}_{cdd}$, leads in most cases to further improvements. On the other hand, minimizing $\mathcal{L}_d$ and $\mathcal{L}_{cdd}$ alone leads to inferior performance compared to the baseline method. We obtain the best configuration according to all metrics using ResNet50 when both $\mathcal{L}_s$ and $\mathcal{L}_{cdd}$ are minimized alongside with the depth term $\mathcal{L}_d$.

Moreover, we also evaluated the output obtained by all models after performing the post-processing step proposed by [57], that consists in forwarding both the input image I and its horizontally flipped counterpart $\hat{I}$. This produces two depth maps d_I and $d_{\hat{I}}$, the latter is flipped back obtaining $\hat{d}_{\hat{I}}$ and averaged with the former, in order to reduce artifacts near occlusions. We can notice that the previous trend is confirmed. In particular, the full loss

[1]The testing samples, belonging to the KITTI 2015 dataset, are: 000001, 000003, 000004, 000019, 000032, 000033, 000035, 000038, 000039, 000042, 000048, 000064, 000067, 000072, 000087, 000089, 000093, 000095, 000105, 000106, 000111, 000116, 000119, 000123, 000125, 000127, 000128, 000129, 000134, 000138, 000150, 000160, 000161, 000167, 000174, 000175, 000178, 000184, 000185 and 000193.

	Encoder	pp	Lower is better				Higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
$\mathcal{L}_d$ [57]	VGG		0.160	2.707	7.220	0.239	0.837	0.928	0.966
$\mathcal{L}_d$+$\mathcal{L}_s$	VGG		0.155	2.511	6.968	**0.234**	0.841	**0.931**	**0.968**
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	VGG		**0.154**	**2.453**	**6.949**	0.235	**0.844**	**0.931**	0.967
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	VGG		0.161	2.758	7.128	0.240	0.841	0.928	0.964
$\mathcal{L}_d$ [57]	VGG	✓	0.149	2.203	6.582	0.223	0.844	0.936	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$	VGG	✓	0.147	2.229	6.583	0.223	0.847	**0.938**	**0.972**
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	VGG	✓	**0.145**	**2.040**	**6.362**	**0.221**	**0.849**	**0.938**	0.971
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	VGG	✓	0.150	2.278	6.539	0.225	0.843	0.934	0.970
$\mathcal{L}_d$ [57]	ResNet		0.159	2.411	6.822	0.239	0.830	0.930	0.967
$\mathcal{L}_d$+$\mathcal{L}_s$	ResNet		0.152	2.385	6.775	0.231	0.843	0.934	0.970
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	ResNet		**0.143**	**2.161**	**6.526**	**0.222**	**0.850**	**0.939**	**0.972**
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	ResNet		0.155	2.282	6.658	0.232	0.840	0.932	0.968
$\mathcal{L}_d$ [57]	ResNet	✓	0.148	2.104	6.439	0.224	0.839	0.936	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$	ResNet	✓	0.144	2.050	6.351	0.220	0.849	0.938	0.972
$\mathcal{L}_d$+$\mathcal{L}_s$+$\mathcal{L}_{cdd}$	ResNet	✓	**0.136**	**1.872**	**6.127**	**0.210**	**0.854**	**0.945**	**0.976**
$\mathcal{L}_d$+$\mathcal{L}_{cdd}$	ResNet	✓	0.144	1.973	6.199	0.217	0.849	0.940	0.975

TABLE 13.1: Ablation experiments on KITTI 2015 evaluation split, using different configurations of losses, encoders and post-processing (pp). Best setup highlighted in bold for each configuration.

	Lower is better				Higher is better		
	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Zhou et al. [66]	0.286	7.009	8.377	0.320	0.691	0.854	0.929
Mahjourian et al. [67]	0.235	2.857	7.202	0.302	0.710	0.866	0.935
Yin et al. [231]	0.236	3.345	7.132	0.279	0.714	0.903	0.950
Godard et al. [57]	0.159	2.411	6.822	0.239	0.830	0.930	0.967
Ours	**0.143**	**2.161**	**6.526**	**0.222**	**0.850**	**0.939**	**0.972**

TABLE 13.2: Comparison with other self supervised method on KITTI 2015 evaluation split. Both [57] and our method use ResNet50 encoder.

$\mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_{cdd}$ leads to the best result on most scores. Furthermore, including the post-processing step allows the VGG model trained with our full loss to outperform the baseline ResNet50 architecture supervised by traditional depth losses only. This fact can be noticed in Table 13.1 comparing row 7 with row 13, observing that the former leads to better results except for δ_3 metric.

To further prove the effectiveness of our proposed method we compare it with other self-supervised approach as [231],[66],[67]. Thus, we have ran experiments with the source code available from [231],[66],[67] using the same testing data as for [57] and our method. Table 13.2 shows the outcome of this evaluation. We point out that we used the weights made available by the authors of [231],[66],[67], trained on a much larger amount of data (i.e., the entire Cityscapes and KITTI sequences, some of them overlapping with

the testing split as well) w.r.t. the much lower supervision provided to our network. Despite this fact, monocular supervised works [231],[66],[67] perform poorly compared to both [57] and our approach, confirming our semi-supervised framework to outperform them as well. We also point out that our test split relies on high-quality ground-truth labels for evaluation, available from KITTI 2015 stereo dataset, while the Eigen split used to validate [231],[66],[67] provides much worse quality depth measurements, as also argued by the authors of [57].

As our final test we also compare our method with the recent multi-task learning approach by Kendall et al. [232]. Differently from our approach, they jointly learn depth, semantic and instance segmentation in fully supervised manner. They run experiments Tiny Cityscapes, a split obtained by resizing the validation set of Cityscapes to 128×256 resolution. To compare our results to theirs we have taken our ResNet50 model trained on Cityscapes and validated it following the same protocol. Their depth-only model (trained supervised) achieves 0.640 inverse mean depth error, dropping to 0.522 when trained to tackle semantic and instance segmentation as well. Our ResNet50 network (trained unsupervised) starts with 1.705 error for depth-only, dropping to 1.488. Thus, the two approaches achieve 22% and 15 % improvement respectively. We point out that, besides relying on supervised learning for depth, [232] exploits both semantic and instance segmentation, requiring additional manually annotated labels, while we only enforce our cross-domain discontinuity loss.

Figure 13.4 depicts a qualitative comparison between the depth maps predicted by [57] and our semi-supervised framework. In the figure, from top to bottom, we consider images 000019 and 000095 belonging to our evaluation split. We can observe how explicitly learning the semantics of the scene helps to correct wrong depth estimations, especially on challenging objects. For example, we can notice how depth maps predicted by our frameworks provide better car shapes thanks to the contribution given by the semantic. This fact is particularly evident in correspondence of reflective or transparent surfaces like car windows as reported on image 000095. Moreover, the quality of thin structures like poles is improved as well, as clearly perceivable by looking at frame 000019.

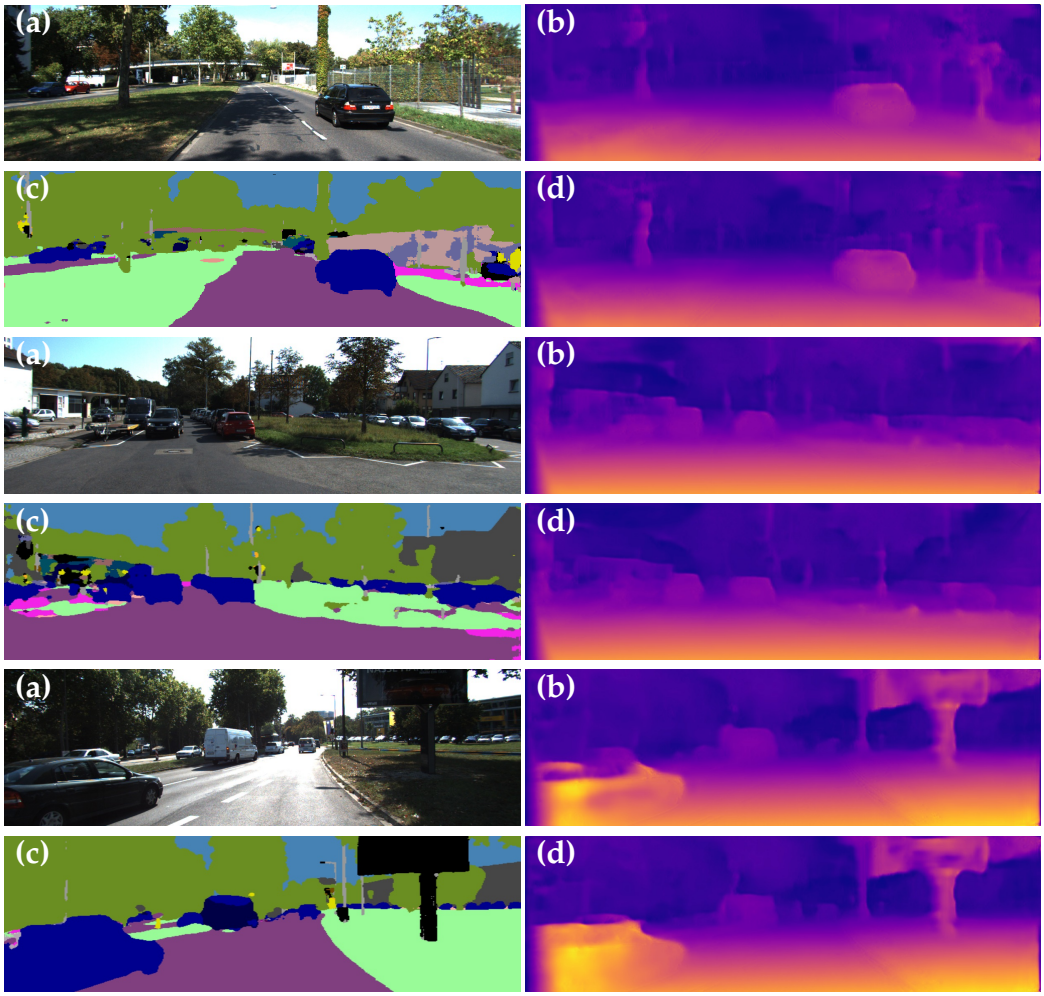

FIGURE 13.4: Qualitative comparison between [57] and our proposal on KITTI 2015 evaluation split [178]. (a) Input image, (b) depth map by [57], (c) and (d) semantic and depth maps by our approach. Both models use Resnet50 as encoder. From top to bottom, results concerning images 000019, 000087 and 000095 belonging to our evaluation split.

13.2.3 Semantic segmentation: evaluation on KITTI 2015

Although our proposal is aimed at ameliorating depth prediction by learning richer features exploiting semantics, our network also delivers a semantic segmentation of the input image. To gather hints about the accuracy of this additional outcome of our network, we evaluated the semantic maps generated on the same KITTI evaluation split defined before. Differently from the monocular depth estimation task, results concerning semantic segmentation are quite far from the state-of-the-art. In particular, we obtain 88.51% and 88.19% per-pixel accuracy, respectively, with models based on VGG and

ResNet50. We ascribe this to our architecture - inspired by [57] - being optimized for unsupervised depth prediction, whereas different design choices are often found in networks pursuing semantic segmentation (i.e., atrous convolutions, SPP layers ...). We also found that training the basic encoder-decoder for semantic segmentation only yields to 86.72% and 88.18% per-pixel accuracy with VGG and ResNet50, respectively. Thus, while semantics helps depth prediction inasmuch as to outperform the state-of-the-art within the proposed framework, the converse requires further studies as the observed improvements are indeed quite minor.

Chapter 14

Geometry, Semantics and Motion

In this chapter, we propose the first-ever framework for comprehensive scene understanding from monocular videos. As highlighted in Figure 14.1, our multi-stage network architecture, named ΩNet, can predict depth, semantics, optical flow, per-pixel motion probabilities and motion masks. This comes alongside with estimating the pose between adjacent frames for an uncalibrated camera, whose intrinsic parameters are also estimated. Our training methodology leverages on self-supervision, knowledge distillation and multi-task learning. In particular, peculiar to our proposal and key to performance is distillation of proxy semantic labels gathered from a state-of-the-art pre-trained model [43] within a self-supervised and multi-task learning procedure addressing depth, optical flow and motion segmentation. Our training procedure also features a novel and effective self-distillation schedule for optical flow mostly aimed at handling occlusions and relying on tight integration of rigid flow, motion probabilities and semantics. Moreover, ΩNet is lightweight, counting less than 8.5M parameters, and fast, as it can run at nearly 60 FPS and 5 FPS on an NVIDIA Titan Xp and a Jetson TX2, respectively. Our model is able to achieve state-of-the-art performances in several tasks such as self-supervised monocular depth estimation, optical flow estimation among monocular multi-task frameworks and motion segmentation.

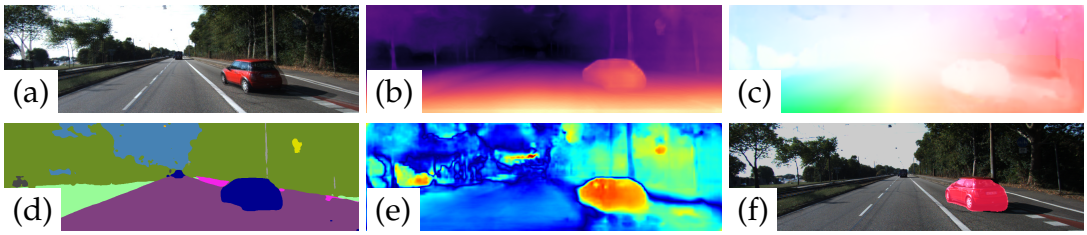

FIGURE 14.1: Given an input monocular video (a), our network can provide the following outputs in real-time: depth (b), optical flow (c), semantic labels (d), per-pixel motion probabilities (e), motion mask (f).

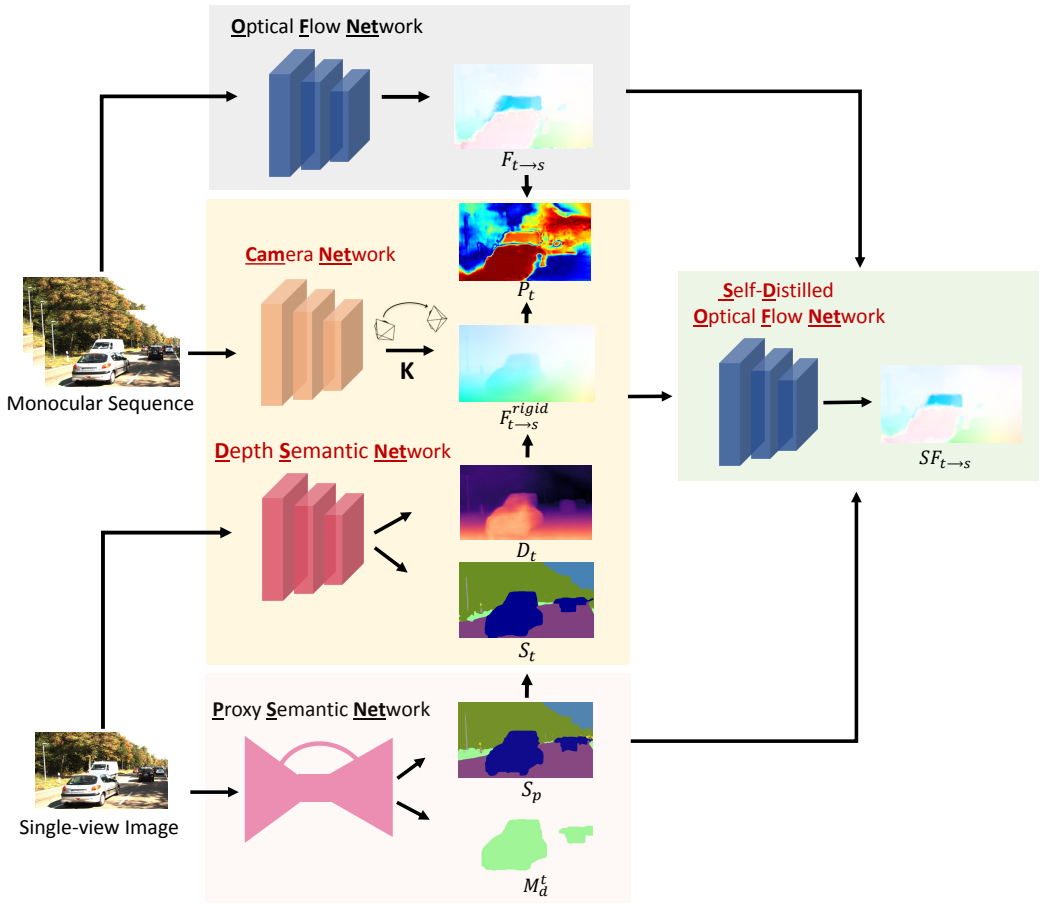

FIGURE 14.2: Overall framework for training ΩNet to predict depth, camera pose, camera intrinsics, semantic labels and optical flow. In **red** architectures composing ΩNet.

14.1 Overall Learning Framework

Our goal is to develop a real-time comprehensive scene understanding framework capable of learning strictly related tasks from monocular videos. Purposely, we propose a multi-stage approach to learn first geometry and semantics, then elicit motion information, as depicted in Figure 14.2.

14.1.1 Geometry and Semantics

Self-supervised depth and pose estimation. We propose to solve a self-supervised single-image depth and pose estimation problem by exploiting geometrical constraints in a sequence of N images, in which one of the frames is used as the target view I_t and the other ones in turn as the source image I_s. Assuming a moving camera in a stationary scene, given a depth map D_t aligned with I_t, the camera intrinsic parameters K and the relative pose $T_{t\to s}$

between I_t and I_s, it is possible to sample pixels from I_s in order to synthesise a warped image $\widetilde{I}_t$ aligned with I_t. The mapping between corresponding homogeneous pixels coordinates $p_t \in I_t$ and $p_s \in I_s$ is given by:

$$p_s \sim K T_{t \to s} D_{p_t} K^{-1} p_t \tag{14.1}$$

Following [66], we use the sub-differentiable bilinear sampler mechanism proposed in [58] to obtain $\widetilde{I}_t$. Thus, in order to learn depth, pose and camera intrinsics we train two separate CNNs to minimize the photometric reconstruction error between $\widetilde{I}_t$ and I_t, defined as:

$$\mathcal{L}_{ap}^{D} = \sum_{p} \psi(I_t(p), \widetilde{I}_t(p)) \tag{14.2}$$

where ψ is a photometric error function between the two images. However, as pointed out in [79], such a formulation is prone to errors at occlusion/disocclusion regions or in static camera scenarios. To soften these issues, we follow the same principles as suggested in [79], where a minimum per-pixel reprojection loss is used to compute the photometric error, an automask method allows for filtering-out spurious gradients when the static camera assumption is violated, and an edge-aware smoothness loss term is used as in [233]. Moreover, we use the depth normalization strategy proposed in [68]. See supplementary material for further details.

We compute the rigid flow between I_t and I_s as the difference between the projected and original pixel coordinates in the target image:

$$F_{t \to s}^{rigid}(p_t) = p_s - p_t \tag{14.3}$$

Distilling semantic knowledge. The proposed distillation scheme is motivated by how time-consuming and cumbersome obtaining accurate pixel-wise semantic annotations is. Thus, we train our framework to estimate semantic segmentation masks S_t by means of supervision from cheap proxy labels S_p distilled by a semantic segmentation network, pre-trained on few annotated samples and capable to generalize well to diverse datasets. Availability of proxy semantic labels for the frames of a monocular video enables us to train a single network to predict jointly depth and semantic labels. Accordingly, the joint loss is obtained by adding a standard cross-entropy term $\mathcal{L}_{sem}$ to the previously defined self-supervised image reconstruction loss $\mathcal{L}_{ap}^{D}$. Moreover, similarly to [1], we deploy a cross-task loss term, $\mathcal{L}_{edge}^{D}$ (see supplementary), aimed at favouring spatial coherence between depth edges and

semantic boundaries. However, unlike [1], we do not exploit stereo pairs at training time.

14.1.2 Optical Flow and Motion Segmentation

Self-supervised optical flow. As the 3D structure of a scene includes stationary as well as non-stationary objects, to handle the latter we rely on a classical optical flow formulation. Formally, given two images I_t and I_s, the goal is to estimate the 2D motion vectors $F_{t\to s}(p_t)$ that map each pixel in I_t into its corresponding one in I_s. To learn such a mapping without supervision, previous approaches [102, 105, 70] employ an image reconstruction loss $\mathcal{L}_{ap}^{F}$ that minimizes the photometric differences between I_t and the back-warped image $\widetilde{I}_t$ obtained by sampling pixels from I_s using the estimated 2D optical flow $F_{t\to s}(p_t)$. This approach performs well for non-occluded pixels but provides misleading information within occluded regions.

Pixel-wise motion probability. Non-stationary objects produce systematic errors when optimizing $\mathcal{L}_{ap}^{D}$ due to the assumption that the camera is the only moving body in an otherwise stationary scene. However, such systematic errors can be exploited to identify non-stationary objects: at pixels belonging to such objects the rigid flow $F_{t\to s}^{rigid}$ and the optical flow $F_{t\to s}$ should exhibit different directions and/or norms. Therefore, a pixel-wise probability of belonging to an object independently moving between frames s and t, P_t, can be obtained by normalizing the differences between the two vectors. Formally, denoting with $\theta(p_t)$ the angle between the two vectors at location p_t, we define the per-pixel motion probabilities as:

$$P_t(p_t) = \max\{\frac{1-\cos\theta(p_t)}{2}, 1-\rho(p_t)\} \tag{14.4}$$

where $\cos\theta(p_t)$ can be computed as the normalized dot product between the vectors and evaluates the similarity in direction between them, while $\rho(p_t)$ is defined as

$$\rho(p_t) = \frac{\min\{\|F_{t\to s}(p_t)\|_2, \|F_{t\to s}^{rigid}(p_t)\|_2\}}{\max\{\|F_{t\to s}(p_t)\|_2, \|F_{t\to s}^{rigid}(p_t)\|_2\}}, \tag{14.5}$$

i.e. a normalized score of the similarity between the two norms. By taking the maximum of the two normalized differences, we can detect moving objects even when either the directions or the norms of the vectors are similar. A visualization of $P_t(p_t)$ is depicted in Figure 14.3(d).

Semantic-aware Self-Distillation Paradigm. Finally, we combine semantic information, estimated optical flow, rigid flow and pixel-wise motion probabilities within a final training stage to obtain a more robust self-distilled optical flow network. In other words, we train a new instance of the model to infer a self-distilled flow $SF_{t\to s}$ given the estimates $F_{t\to s}$ from a first self-supervised network and the aforementioned cues. As previously discussed and highlighted in Figure 14.3(c), standard self-supervised optical flow is prone to errors in occluded regions due to the lack of photometric information but can provide good estimates for the dynamic objects in the scene. On the contrary, the estimated rigid flow can properly handle occluded areas thanks to the minimum-reprojection mechanism [79]. Starting from these considerations, our key idea is to split the scene into stationary and potentially dynamics objects, and apply on them the proper supervision. Purposely, we can leverage several observations:

1. **Semantic priors.** Given a semantic map S_t for image I_t, we can binarize pixels into static M_t^s and potentially dynamic M_t^d, with $M_t^s \cap M_t^d = \emptyset$. For example, we expect that points labeled as *road* are static in the 3D world, while pixels belonging to the semantic class *car* may move. In M_t^d, we assign 1 for each potentially dynamic pixel, 0 otherwise, as shown in Figure 14.3(e).

2. **Camera Motion Boundary Mask.** Instead of using a backward-forward strategy [71] to detect boundaries occluded due to the ego-motion, we analytically compute a binary boundary mask M_t^b from depth and ego-motion estimates as proposed in [234]. We assign a 0 value for out-of-camera pixels, 1 otherwise as shown in Figure 14.3(f).

3. **Consistency Mask.** Because the inconsistencies between the rigid flow and $F_{t\to s}$ are not only due to dynamic objects but also to occluded/inconsistent areas, we can leverage Equation 14.4 to detect such critical regions. Indeed, we define the consistency mask as:

$$M_t^c = P_t < \xi, \xi \in [0, 1] \tag{14.6}$$

   This mask assigns 1 where the condition is satisfied, 0 otherwise (i.e. inconsistent regions) as in Figure 14.3(g).

Finally, we compute the final mask M, in Figure 14.3(h), as:

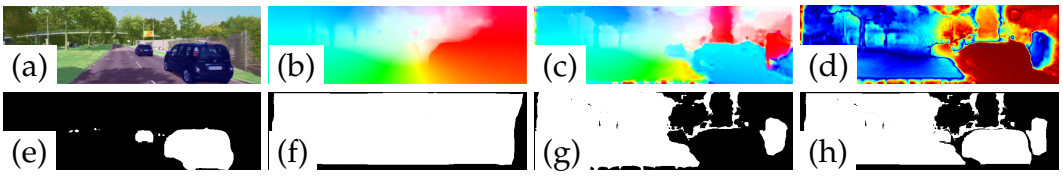

FIGURE 14.3: Overview of our semantic-aware and self-distilled optical flow estimation approach. We leverage semantic segmentation S_t (a) together with rigid flow $F^{rigid}_{t\to s}$ (b), teacher flow $F_{t\to s}$ (c) and motion probabilities P_t (d), the warmer the higher. From a) we obtain semantic priors M^d_t (e), combined with boundary mask M^b_t (f) and consistency mask M^c_t (g) derived from (d) as in Equation 14.6, in order to obtain the final mask M (h) as in Equation 14.7.

$$M = \min\{\max\{M^d_t, M^c_t\}, M^b_t\} \tag{14.7}$$

As a consequence, M will effectively distinguish regions in the image for which we can not trust the supervision sourced by $F_{t\to s}$, i.e. inconsistent or occluded areas. On such regions, we can leverage our proposed self-distillation mechanism. Then, we define the final total loss for the self-distilled optical flow network as:

$$\mathcal{L} = \sum \alpha_r \phi(SF_{t\to s}, F^{rigid}_{t\to s}) \cdot (1 - M) + \alpha_d \phi(SF_{t\to s}, F_{t\to s}) \cdot M + \psi(I_t, \widetilde{I}^{SF}_t) \cdot M \tag{14.8}$$

where ϕ is a distance function between two motion vectors, while α_r and α_d are two hyper-parameters.

14.1.3 Motion Segmentation

At test time, from pixel-wise probability P_t computed between $SF_{t\to s}$ and $F^{rigid}_{t\to s}$, semantic prior M^d_t and a threshold τ, we compute a motion segmentation mask by:

$$M^{mot}_t = M^d_t \cdot (P_t > \tau), \tau \in [0, 1] \tag{14.9}$$

Such mask allows us to detect moving objects in the scene independently of the camera motion. A qualitative example is depicted in Figure 14.1(f).

14.2 Architecture and Training Schedule

In this section we present the networks composing ΩNet (highlighted in red in Figure 14.2), and delineate their training protocol. We set $N = 3$, using 3-frames sequences.

14.2.1 Network architectures

We highlight here the key traits of each network.

Depth and Semantic Network (DSNet). We build a single model, since shared reasoning about the two tasks is beneficial to both [1, 222]. To achieve real-time performance, DSNet is inspired to PydNet [59], with several key modifications due to the different goals. We extract a pyramid of features down to $\frac{1}{32}$ resolution, estimating a first depth map at the bottom. Then, it is upsampled and concatenated with higher level features in order to build a refined depth map. We repeat this procedure up to half resolution, where two estimators predict the final depth map D_t and semantic labels S_t. These are bi-linearly upsampled to full resolution. Each conv layer is followed by batch normalization and ReLU, but the prediction layers, using reflection padding. DSNet counts 1.93M parameters.

Camera Network (CamNet). This network estimates both camera intrinsics and poses between a target I_t and some source views $I_s (1 \leq s \leq 3, s \neq t)$. CamNet differs from previous work by extracting features from I_t and I_s independently with shared encoders. We extract a pyramid of features down to $\frac{1}{16}$ resolution for each image and concatenate them to estimate the 3 Euler angles and the 3D translation for each I_s. As in [78], we also estimate the camera intrinsics. Akin to DSNet, we use batch normalization and ReLU after each layer but for prediction layers. CamNet requires 1.77M parameters for pose estimation and 1.02K for the camera intrinsics.

Optical Flow Network (OFNet). To pursue real-time performance, we deploy a 3-frame PWC-Net [96] network as in [105], which counts 4.79M parameters. Thanks to our novel training protocol leveraging on semantics and self-distillation, our OFNet can outperform other multi-task frameworks [73] built on the same optical flow architecture.

Table 14.1 and Table 14.2 report a detailed specification of the layers building up the DSNet and CamNet modules respectively. For each layer, we report kernel size (K), stride (S) and number of input/output channels. As for OFNet and the proxy semantic network, a thorough description can be found in [105] and [235] respectively.

Layer	K	S	In/Out	Input
Deep feature extractor (DSE)				
conv1a	3	2	3/16	input
conv1b	3	1	16/16	conv1a
conv2a	3	2	16/32	conv1b
conv2b	3	1	32/32	conv2a
conv3a	3	2	32/64	conv2b
conv3b	3	1	64/64	conv3a
conv4a	3	2	64/128	conv3b
conv4b	3	1	128/128	conv4a
conv5a	3	2	128/256	conv4b
conv5b	3	1	256/256	conv5a
Estimator (E)				
conv1	3	1	i_channels/64	input
conv2	3	1	64/48	conv1
conv3	3	1	48/32	conv2
conv4	3	1	32/16	conv3
Context (C)				
disp1	3	1	i_channels/64	input
disp2	3	1	64/32	disp1
disp3	3	1	32/16	disp2
disp	3	1	16/1	disp2
Disparity and Semantic Tower				
conv5	3	1	i_channels/16	**E**(conv5b)
disp5	3	1	i_channels//1	**C**(conv5)
conv4	3	1	i_channels/16	**E**(conv4b, disp5 ↑)
disp4	3	1	i_channels//1	**C**(conv4, conv5 ↑) + disp5 ↑
conv3	3	1	i_channels/16	**E**(conv3b, disp4 ↑)
disp3	3	1	i_channels//1	**C**(conv3, conv4 ↑) + disp4 ↑
conv2	3	1	i_channels/16	**E**(conv2b, disp3 ↑)
disp2	3	1	i_channels//1	**C**(conv2, conv3 ↑) + disp3 ↑
conv1	3	1	i_channels/16	**E**(conv1b, disp2 ↑)
disp1	3	1	i_channels//1	**C**(conv1, conv2 ↑) + disp2 ↑
sem	3	1	i_channels//1	**C**(conv1, conv2 ↑) + disp2 ↑

TABLE 14.1: Detailed structure of the DSNet modules in ΩNet. The symbol "," means concatenation, while ↑ indicates upsampling.

14.2.2 Training Protocol

Similarly to [70], we employ a two stage learning process to facilitate the network optimisation process. At first, we train DSNet and CamNet simultaneously, then we train OFNet by the self-distillation paradigm described in 14.1.2. For both stages, we use a batch size of 4 and resize input images to 640×192 for the KITTI dataset (and to 768×384 for pre-training on Cityscapes), optimizing the output of the networks at the highest resolution only. We also report additional experimental results for different input resolutions where specified. We use the Adam optimizer [211] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. As photometric loss ψ, we employ the same function defined in [233]. When training our networks, we apply losses using as

Layer	K	S	In/Out	Input
Deep feature extractor (DFE)				
conv1a	3	2	3/16	input
conv1b	3	1	16/16	conv1a
conv2a	3	2	16/32	conv1b
conv2b	3	1	32/32	conv2a
conv3a	3	2	32/64	conv2b
conv3b	3	1	64/64	conv3a
conv4a	3	2	64/128	conv3b
conv4b	3	1	128/128	conv4a
Pose Estimator				
conv1a	3	1	i_channels/128	DFE_t, DFE_s
conv1b	3	2	128/128	conv1a
conv2a	3	1	128/256	conv1b
conv2b	3	2	256/256	conv2a
pose	1	1	256/6*N	conv2b
Intrinsic Estimator				
focals	1	1	i_channels/2	conv2b
offsets	1	1	i_channels/2	conv2b

TABLE 14.2: Detailed structure of the CamNet modules in ΩNet. The symbol "," means concatenation, while ↑ indicates upsampling.

I_s both the previous and the next image of our 3-frame sequence. Finally, we set both τ and ξ to be 0.5 in our experiments.

Depth, Pose, Intrinsics and Semantic Segmentation. In order to train DSNet and CamNet we employ sequences of 3 consecutive frames and semantic proxy labels yielded by a state-of-the art architecture [44] trained on Cityscapes with ground-truth labels. We trained DSNet and CamNet for 300K iterations, setting the initial learning rate to 10^{-4}, manually halved after 200K, 250K and 275K steps. We apply data augmentation to images as in [233]. Training takes $\sim$ 20 hours on a Titan Xp GPU.

Method	M	A	I	CS	Lower is better Abs Rel	Sq Rel	RMSE	RMSE log	Higher is better δ <1.25	$\delta < 1.25^2$	$\delta < 1.25^3$
Godard *et al.*[79]					0.132	1.044	5.142	0.210	0.845	0.948	0.977
Godard *et al.*[79] (1024 × 320)			✓		**0.115**	0.882	4.701	0.190	**0.879**	**0.961**	0.982
Zhou *et al.*[80]			✓		0.121	0.837	4.945	0.197	0.853	0.955	0.982
Mahjourian *et al.*[234]				✓	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Wang *et al.*[68]				✓	0.151	1.257	5.583	0.228	0.810	0.936	0.974
Bian *et al.*[81]				✓	0.128	1.047	5.234	0.208	0.846	0.947	0.970
Yin *et al.*[70]	✓			✓	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Zou *et al.*[71]	✓			✓	0.146	1.182	5.215	0.213	0.818	0.943	0.978
Chen *et al.*[72]	✓		✓		0.135	1.070	5.230	0.210	0.841	0.948	0.980
Luo *et al.*[75]	✓				0.141	1.029	5.350	0.216	0.816	0.941	0.976
Ranjan *et al.*[73]	✓				0.139	1.032	5.199	0.213	0.827	0.943	0.977
Xu *et al.*[77]		✓	✓		0.138	1.016	5.352	0.217	0.823	0.943	0.976
Casser *et al.*[76]		✓			0.141	1.026	5.290	0.215	0.816	0.945	0.979
Gordon *et al.*[78]	✓	✓			0.128	0.959	5.230	-	-	-	-
ΩNet(640 × 192)	✓	✓			0.126	0.835	4.937	0.199	0.844	0.953	0.982
ΩNet(1024 × 320)	✓	✓			0.125	0.805	4.795	0.195	0.849	0.955	0.983
ΩNet(640 × 192)	✓	✓		✓	0.120	0.792	4.750	0.191	0.856	0.958	0.984
ΩNet(1024 × 320)	✓	✓		✓	0.118	**0.748**	**4.608**	**0.186**	0.865	**0.961**	**0.985**

TABLE 14.3: Depth evaluation on the Eigen split [51] of KITTI [108]. We indicate additional features of each method. M: multi-task learning, A: additional information (e.g. object knowledge, semantic information), I: feature extractors pre-trained on ImageNet [210], CS: network pre-trained on Cityscapes [14].

Optical Flow. We train OFNet by the procedure presented in 14.1.2. In particular, we perform 200K training steps with an initial learning rate of 10^{-4}, halved every 50K until convergence. Moreover, we apply strong data augmentation consisting in random horizontal and vertical flip, crops, random time order switch and, peculiarly, time stop, replacing all I_s with I_t to learn a zero motion vector. This configuration requires about 13 hours on a Titan Xp GPU with the standard 640×192 resolution. We use an L1 loss as ϕ. Once obtained a competitive network in non-occluded regions we train a more robust optical flow network, denoted as SD-OFNet, starting from pre-learned weights and the same structure of OFNet by distilling knowledge from OFNet and rigid flow computed by DSNet using the total mask M and 416×128 random crops applied to $F_{t\to s}$, $F^{rigid}_{t\to s}$, M and RGB images. We train SD-OFNet for $15K$ steps only with a learning rate of 2.5×10^{-5} halved after 5K, 7.5K, 10K and 12.5K steps, setting α_r to 0.025 and α_d to 0.2. At test-time, we rely on SD-OFNet only.

14.3 Experimental results

Using standard benchmark datasets, we present here the experimental validation on the main tasks tackled by ΩNet.

14.3.1 Datasets.

We conduct experiments on standard benchmarks such as KITTI and Cityscapes. We do not use feature extractors pre-trained on ImageNet or other datasets. For the sake of space, we report further studies in the supplementary material (e.g. results on pose estimation or generalization).

KITTI (K) [236] is a collection of 42,382 stereo sequences taken in urban environments from two video cameras and a LiDAR device mounted on the roof of a car. This dataset is widely used for benchmarking geometric understanding tasks such as depth, flow and pose estimation.

Cityscapes (CS) [14] is an outdoor dataset containing stereo pairs taken from a moving vehicle in various weather conditions. This dataset features higher resolution and higher quality images. While sharing similar settings, this dataset contains more dynamics scenes compared to KITTI. It consists of 22,973 stereo pairs with 2048×1024 resolution. 2,975 and 500 images come with fine semantic

Resolution	Learned Intr. [78]	Norm. [68]	Min. Repr. [79]	Automask [79]	Sem. [44]	Pre-train	Lower is better				Higher is better		
							Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	$\delta < 1.25^2$	$\delta < 1.25^3$
640 × 192	-	-	-	-	-	-	0.139	1.056	5.288	0.215	0.826	0.942	0.976
640 × 192	✓	-	-	-	-	-	0.138	1.014	5.213	0.213	0.829	0.943	0.977
640 × 192	✓	✓	-	-	-	-	0.136	1.008	5.204	0.212	0.832	0.944	0.976
640 × 192	✓	✓	✓	-	-	-	0.132	0.960	5.104	0.206	0.840	0.949	0.979
640 × 192	✓	✓	✓	✓	-	-	0.130	0.909	5.022	0.207	0.842	0.948	0.979
640 × 192 †	✓	✓	✓	✓	-	-	0.134	1.074	5.451	0.213	0.834	0.946	0.977
640 × 192	✓	✓	✓	✓	✓	-	0.126	0.835	4.937	0.199	0.844	0.953	0.980
416 × 128	✓	✓	✓	✓	✓	✓	0.126	0.862	4.963	0.199	0.846	0.952	0.981
640 × 192	✓	✓	✓	✓	✓	✓	0.120	0.792	4.750	0.191	0.856	0.958	0.984
1024 × 320	✓	✓	✓	✓	✓	✓	0.118	0.748	4.608	0.186	0.865	0.961	0.985

TABLE 14.4: Ablation study of our depth network on the Eigen split [51] of KITTI. †: our network is replaced by a ResNet50 backbone [70].

14.3.2 Monocular Depth Estimation

In this section, we compare our results to other state-of-the-art proposals and assess the contribution of each component to the quality of our monocular depth predictions.

Comparison with state-of-the-art. We compare with state-of-the-art self-supervised networks trained on monocular videos according to the protocol described in [51]. We follow the same pre-processing procedure as [66] to remove static images from the training split while using all the 697 images for testing. LiDAR points provided in [236] are reprojected on the left input image to obtain ground-truth labels for evaluation, up to 80 meters [56]. Since the predicted depth is defined up to a scale factor, we align the scale of our estimates by multiplying them by a scalar that matches the median of the ground-truth, as introduced in [66]. We adopt the standard performance metrics defined in [51]. Table 14.3 reports extensive comparison with respect to several monocular depth estimation methods. We outperform our main competitors such as [70, 71, 72, 73] that solve multi-task learning or other strategies that exploit additional information during the training/testing phase [76, 77]. Moreover, our best configuration, i.e. pre-training on CS and using 1024×320 resolution, achieves better results in 5 out of 7 metrics with respect to the single-task, state-of-the-art proposal [79] (and is the second best and very close to it on the remaining 2) which, however, leverages on a larger ImageNet pre-trained model based on ResNet-18. It is also interesting to note how our proposal without pretraining obtains the best performance in 6 out of 7 measures on 640×192 images (row 1 vs 15). These results validate our intuition about how the use of semantic information can guide geometric reasoning and make a compact network provide state-of-the-art performance even with respect to larger and highly specialized depth-from-mono methods.

Ablation study. Table 14.4 highlights how progressively adding the key innovations proposed in [78, 79, 68] contributes to strengthen ΩNet, already comparable to other methodologies even in its baseline configuration (first row). Interestingly, a large improvement is achieved by deploying joint depth and semantic learning (rows 5 vs 7), which forces the network to simultaneously reason about geometry and content within the same shared features. By replacing DSNet within ΩNet with a larger backbone [70] (rows 5 vs 6) we obtain worse performance, validating the design decisions behind our compact model. Finally, by pre-training on CS we achieve the best accuracy, which increases alongside with the input resolution (rows 8 to 10).

Method	Cap (m)	Abs Rel	Sq Rel	RMSE	RMSE log
Godard *et al.*[79]	0-8	**0.059**	**0.062**	0.503	**0.082**
ΩNet†	0-8	0.060	0.063	**0.502**	**0.082**
ΩNet	0-8	0.062	0.065	0.517	0.085
Godard *et al.*[79]	0-50	0.125	0.788	3.946	0.198
ΩNet†	0-50	0.127	0.762	4.020	0.199
ΩNet	0-50	**0.124**	**0.702**	**3.836**	**0.195**
Godard *et al.*[79]	0-80	0.132	1.044	5.142	0.210
ΩNet†	0-80	0.134	1.074	5.451	0.213
ΩNet	0-80	**0.126**	**0.835**	**4.937**	**0.199**

TABLE 14.5: Depth errors by varying the range. †: our network is replaced by a ResNet50 backbone [70].

Depth Range Error Analysis. We dig into our depth evaluation to explain the effectiveness of ΩNet with respect to much larger networks. Table 14.5 compares, at different depth ranges, our model with more complex ones [79, 70]. This experiment shows how ΩNet superior performance comes from better estimation of large depths: ΩNet outperforms both competitors when we include distances larger than 8 m in the evaluation, while it turns out less effective in the close range.

14.3.3 Semantic Segmentation

In Table 14.6, we report the performance of ΩNet on semantic segmentation for the 19 evaluation classes of CS according to the metrics defined in [14, 29]. We compare ΩNet against state-of-the art networks for real-time semantic segmentation [49, 47] when training on CS and testing either on the validation set of CS (rows 1-3) or the 200 semantically annotated images of K (rows 4-6). Even though our network is not as effective as the considered methods when training and testing on the same dataset, it shows greater generalization capabilities to unseen domains: it significantly outperforms other methods when testing on K for $\text{mIoU}_{\text{category}}$ and pixel accuracy, and provides similar results to [49] for $\text{mIoU}_{\text{class}}$. We relate this ability to our training protocol based on proxy labels (P) instead of ground truths (S).

Moreover, as we have already effectively distilled the knowledge from DPC [44] during pre-training on CS, there is only a slight benefit in training on both CS and K (with proxy labels only) and testing on K (row 7). Finally, although achieving 46.68 mIoU on fine segmentation, we obtain 89.64 mIoU

Method	Train	Test	mIoU Class	mIoU Cat.	Pix.Acc.
DABNet [47]	CS(S)	CS	69.62	87.56	94.62
FCHardNet [49]	CS(S)	CS	**76.37**	**89.22**	**95.35**
ΩNet	CS(P)	CS	54.80	82.92	92.50
DABNet [47]	CS(S)	K	35.40	61.49	80.50
FCHardNet [49]	CS(S)	K	**44.74**	68.20	72.07
ΩNet	CS(P)	K	43.80	**74.31**	**88.31**
ΩNet	CS(P) + K(P)	K	46.68	75.84	88.12

TABLE 14.6: Semantic segmentation on Cityscapes (CS) and KITTI (K). S: training on ground-truth, P: training on proxy labels.

for the task of segmenting static from potentially dynamic classes, an important result to obtain accurate motion masks.

In Table 14.7 and Table 14.8 we dig into the motivation behind our better generalization with thorough experiments. We train on CityScapes and test on KITTI, reporting in Table 14.7 the IoU for the 19 classes, the $\text{mIoU}_{\text{class}}$ and the pixel Acc. In Table 14.8 we report the IoU for the 7 categories and the $\text{mIoU}_{\text{category}}$. We refer with CS(S) methods trained on 2975 CityScapes images and with CS(P) methods trained on 22,973 proxy labels produced by [235]. To evaluate the performance of [47, 49] we used the official code and pre-trained weights available online. Our DSNet differs from other methods by three factors: 1) the architecture, 2) the training protocol exploiting proxy labels instead of ground truths and 3) the joint reasoning about geometry and semantics.

Regarding the tests on KITTI, our architecture trained only for semantic segmentation, namely Semantic Network or SNet, achieves good performance in Acc. but turns out worse than [49] for other metrics. On the other hand, it is worth to notice that training SNet with CS(P) allows our method to achieve a great performance boost in all metrics with respect to CS(P) (rows 8 vs 9). Finally, we can notice how DSNet achieves results comparable to SNet. This confirms the findings in [1], that joint reasoning about depth and semantics is more beneficial to the former task.

Method	Train	Test	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	$mIoU_{class}$	Acc.
DABNet [47]	CS(S)	CS	97.05	82.86	91.01	48.20	55.56	59.30	63.12	72.76	91.58	**61.24**	93.50	77.96	54.70	93.28	53.06	71.01	27.77	56.00	72.91	69.62	94.62
FCHardNet [49]	CS(S)	CS	**97.39**	**84.40**	**92.31**	**53.83**	**62.90**	**64.28**	**68.21**	**78.06**	**91.85**	59.82	**94.91**	**80.81**	**60.55**	**94.85**	**72.70**	**82.15**	**76.45**	**59.97**	**75.49**	**76.37**	**95.35**
ΩNet(SNet)	CS(S)	CS	93.69	65.66	83.46	23.57	20.57	40.11	35.32	47.77	86.62	44.22	89.94	56.00	23.00	84.98	17.22	1.22	0.00	17.11	52.82	46.49	89.56
ΩNet(SNet)	CS(P)	CS	95.97	77.23	87.96	38.37	42.62	47.82	48.15	60.44	89.73	54.97	92.62	65.87	36.96	90.57	25.19	0.06	0.00	25.53	61.06	54.80	92.45
ΩNet(DSNet)	CS(P)	CS	96.00	77.46	88.30	41.84	41.68	48.74	47.80	59.24	89.61	53.89	92.57	66.29	38.61	90.61	27.39	0.37	0.00	18.01	62.78	54.80	92.50
DABNet [47]	CS(S)	K	79.02	19.07	58.38	18.04	**30.73**	40.61	44.24	41.67	80.87	48.76	76.61	13.39	0.17	63.30	21.32	8.21	19.81	1.29	7.04	35.40	80.50
FCHardNet [49]	CS(S)	K	75.66	32.65	**78.51**	13.16	28.46	51.33	**57.16**	55.58	81.06	45.59	91.43	23.84	12.19	58.86	**24.91**	**34.89**	**68.71**	4.66	11.38	**44.74**	72.07
ΩNet(SNet)	CS(S)	K	83.31	33.39	66.57	12.15	20.18	44.20	37.76	32.35	84.46	58.79	88.70	24.66	13.55	76.09	12.62	2.09	0.10	1.15	12.64	37.09	84.94
ΩNet(SNet)	CS(P)	K	**88.73**	**47.85**	77.01	**19.72**	30.65	47.34	53.63	43.16	86.65	**67.97**	94.49	24.81	29.39	80.68	14.88	0.53	0.00	3.05	12.30	43.31	88.76
ΩNet(DSNet)	CS(P)	K	87.89	46.64	77.48	18.55	29.65	**48.73**	51.12	40.52	**86.66**	63.54	**95.06**	**29.79**	**34.74**	**82.03**	12.77	0.63	0.00	**7.60**	**18.82**	43.80	**88.31**

TABLE 14.7: IoU on 19 training classes, $mIoU_{class}$ and pixel accuracy (Acc.) results of ΩNet against state of the art method training on CS and tested on CS or K. Better generalization from CS to K thanks to our proxy labels training protocol.

Method	Train	Test	flat	construction	object	nature	sky	human	vehicle	$\text{mIoU}_{\text{category}}$
DABNet [47]	CS(S)	CS	97.93	91.69	65.90	92.03	93.50	79.59	92.25	87.56
FCHardNet [49]	CS(S)	CS	**98.19**	**92.55**	**70.77**	**92.27**	**94.91**	**82.31**	**93.54**	**89.22**
ΩNet(SNet)	CS(S)	CS	96.34	84.29	44.37	86.85	89.94	60.13	83.77	77.96
ΩNet(SNet)	CS(P)	CS	97.40	88.80	53.61	90.19	92.62	69.08	88.47	82.88
ΩNet(DSNet)	CS(P)	CS	97.38	88.76	53.91	89.93	92.57	69.27	88.61	82.92
DABNet [47]	CS(S)	K	83.41	59.07	46.41	84.30	76.61	17.05	63.61	61.49
FCHardNet [49]	CS(S)	K	80.89	75.35	58.68	88.11	91.43	24.62	58.33	68.20
ΩNet(SNet)	CS(S)	K	87.93	63.92	45.79	85.47	88.70	31.02	69.95	67.54
ΩNet(SNet)	CS(P)	K	**91.97**	**74.95**	52.29	**89.80**	94.49	29.28	**81.83**	73.52
ΩNet(DSNet)	CS(P)	K	91.42	74.84	**53.35**	89.36	**95.06**	**35.45**	80.69	**74.31**

TABLE 14.8: IoU on 7 training categories and, $\text{mIoU}_{category}$ results of ΩNet against state of the art method training on CS and tested on CS or K. Better generalization from CS to K thanks to our proxy labels training protocol.

14.3.4 Proxy Semantic Network

We evaluate the performance of the proxy semantic network. We employ DPC [235], pre-trained on CityScapes with the 2975 training ground truths. We report in Table 14.9 the testing results on the 500 and 200 images belonging to CityScapes validation set and the KITTI training datasets, respectively. Even though DPC [235] achieves impressive performance both on CityScapes as well as in generalizing to KITTI, it is a huge network unable to run in real-time (i.e. , it approximately delivers 3.5 fps on 768 $\times$ 384 images).

Method	Train	Test	$\text{mIoU}_{\text{class}}$	$\text{mIoU}_{\text{category}}$	Acc.
DPC[44] - Proxy	CS(S)	CS	80.22	90.73	95.99
DPC[44] - Proxy	CS(S)	K	58.75	81.30	90.21

TABLE 14.9: Semantic segmentation performances of the proxy semantic network [235] on CS and K datasets.

Priors Evaluation on KITTI

When we produce the priors used during training and, at prediction time, to create the M_t^{mot}, we split the 19 classes in static and potentially dynamic ones according to the following scheme:

1. **Static**: road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, sky

2. **Potentially dynamic**: person, rider, car, truck, bus, train

As among our objectives is to obtain a good motion segmentation mask, we are interested in evaluating the quality of our semantic segmentation predictions in terms of how they are amenable to producing good estimated priors according to the mapping defined above. We evaluate our DSNet trained on CityScapes+KITTI in the 200 KITTI images which provides semantic labels. We obtain a pixel accuracy of 98.50% while a 98.40% IoU for the static classes and a 80.99% for the potentially dynamic classes for a global 89.64% mIoU. It is worth noticing that, even though our segmentation is not able to perform a precise class segmentation, it yields excellent binary priors that turns out key to performance for motion segmentation.

14.3.5 Optical Flow

In Table 14.10, we compare the performance of our optical flow network with competing methods using the KITTI 2015 stereo/flow training set [108] as testing set, which contains 200 ground-truth optical flow measurements for

			train		test
Method	Dataset	Noc	All	F1	F1
Meister*et al.*[102] - C	SYN + K	-	8.80	28.94%	29.46%
Meister *et al.*[102] - CSS	SYN + K	-	8.10	23.27%	23.30%
Zou *et al.*[71]	SYN + K	-	8.98	26.0%	25.70%
Ranjan *et al.*[73]	SYN + K	-	5.66	20.93%	25.27%
Wang *et al.*[238] **	K	-	5.58	-	18.00%
Yin *et al.*[70]	K	8.05	10.81	-	-
Chen *et al.*[72] †	K	5.40	8.95	-	-
Chen *et al.*[72] (online) †	K	4.86	8.35	-	-
Ranjan *et al.*[73]	K	-	6.21	26.41%	-
Luo *et al.*[75]	K	-	5.84	-	21.56%
Luo *et al.*[75] *	K	-	5.43	-	20.61%
Ω**Net** (Ego-motion)	K	11.72	13.50	51.22%	-
OFNet	K	3.48	11.61	25.78%	-
SD-OFNet	K	**3.29**	**5.39**	**20.0**%	**19.47%**

TABLE 14.10: Optical flow evaluation on the KITTI 2015 dataset. †: pre-trained on ImageNet, SYN: pre-trained on SYNTHIA [237], *: trained on stereo pairs, **: using stereo at testing time.

evaluation. We exploit all the raw K images for training, but we exclude the images used at testing time as done in [71] , to be consistent with experimental results of previous self-supervised optical flow strategies [70, 71, 72, 73]. From the table, we can observe how our self-distillation strategy allows SD-OFNet to outperform by a large margin competitors trained on K only (rows 5-11), and it even performs better than models pre-initialized by training on synthetic datasets [237]. Moreover, we submitted our flow predictions to the online KITTI flow benchmark after retraining the network including images from the whole official training set. In this configuration, we can observe how our model achieves state-of-the-art $F1$ performances with respect to other monocular multi-task architectures.

14.3.6 Pose Estimation

We validate the performance of our framework on pose estimation on the KITTI odometry split, which provides ground-truth camera poses obtained with IMU/GPS readings for 11 driving sequences, indexed from 00 to 08 for training and 09-10 for testing purposes. As in [79], we have not changed our architecture for this specific task but simply trained it from scratch on new

training sequences without known intrinsic parameters. We compare our model with learned camera intrinsic parameters with several monocular self-supervised methods on the two sequences of KITTI odometry test split. All of the results, summarized in 14.11, are evaluated by optimizing the scaling factor to align with the ground-truth to address the inherent scale ambiguity.

Method	Frames	Sequence 09	Sequence 10
Zhou *et al.*[66]	5	0.021 ± 0.017	0.020 ± 0.015
Ranjan *et al.*[73]	5	0.012 ± 0.007	0.012 ± 0.008
Yin *et al.*[70]	5	0.012 ± 0.007	0.012 ± 0.009
ORB-Slam	3	0.014 ± 0.008	0.012 ± 0.011
Casser *et al.*[76]	3	0.011 ± 0.006	0.011 ± 0.010
Zou *et al.*[76]	3	0.017 ± 0.007	0.015 ± 0.009
Luo *et al.*[75]	3	0.013 ± 0.007	0.012 ± 0.008
Godard *et al.*[79]	2	0.017 ± 0.008	0.015 ± 0.010
Ours †	2	0.020 ± 0.013	0.017 ± 0.011

TABLE 14.11: Absolute Trajectory Error (ATE) of pose estimation evaluated on the KITTI odometry split sequences 09-10. †indicates strategies trained with unkwnown camera intrinsics.

14.3.7 Motion Segmentation

In Table 14.13 we report experimental results for the motion segmentation task on the KITTI 2015 dataset, which provides 200 images manually annotated with motion labels for the evaluation. We compare our methodology with respect to other state-of-the-art strategies that performs multi-task learning and motion segmentation [73, 75, 238] using the metrics and evaluation protocol proposed in [75]. It can be noticed how our segmentation strategy outperforms all the other existing methodologies by a large margin. This demonstrates the effectiveness of our proposal to jointly combine semantic reasoning and motion probability to obtain much better results. We also report, as upper bound, the accuracy enabled by injecting semantic proxies [44] in place of ΩNet semantic predictions to highlight the low margin between the two.

Moreover, in Figure 14.4, we present an ablation study dealing with the motion segmentation task. In Table 14.13, to be consistent with other methodologies, we set the threshold τ used for the evaluation to 0.5. However, we point out that a careful tuning of such threshold can improve the overall motion segmentation accuracy. In particular, we can notice how the best configuration for our predictions is obtained using a larger threshold. Indeed, we found out that the best trade-off between the mean accuracy and the mean

IoU is achieved by setting the threshold value to 0.7 (in this case the Mean Acc is 0.91 while Mean IoU is 0.77).

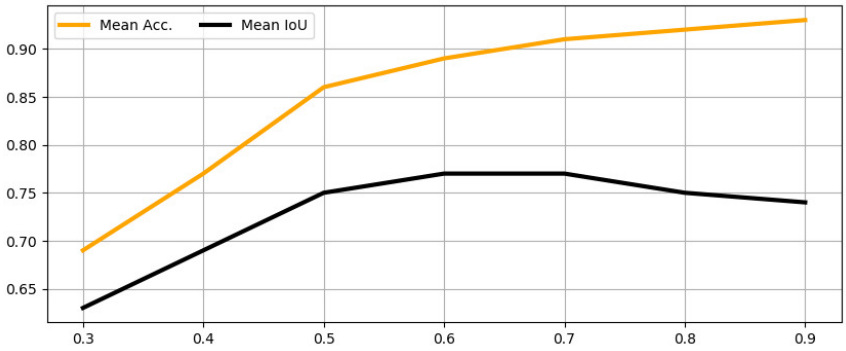

FIGURE 14.4: Mean Acc. and mIoU varying the threshold used to compute the motion segmentation M_t^{mot}.

Finally, we conduct an additional study to evaluate our motion segmentation masks only on pixels belonging to Cars, as proposed in [73]. In Table 14.12 we evaluate the IoU for static and dynamic cars yielded by ΩNet and [73] on the 200 KITTI images endowed with ground truth for the motion segmentation task. We notice that our M_t^{mot} outperforms [73] in all metrics (rows 1 vs 2 and 3) for all thresholds. Moreover, we point out that in this test configuration the contribution given to the motion segmentation by our estimated semantics is almost negligible as car regions are already extracted by using KITTI ground truths. Therefore, we test also our motion probability P_t alone, showing that it is superior to [73] even without the help provided by the estimated semantics.

Method	Threshold	Overall	Static Cars	Moving Cars
Ranjan [73]	-	56.94	55.77	58.11
ΩNet M_t^{mot}	0.5	**63.98**	**64.16**	63.79
ΩNet M_t^{mot}	0.7	63.97	64.15	63.79
ΩNet P_t	0.5	63.67	62.58	64.77
ΩNet P_t	0.7	62.66	58.42	**66.89**

TABLE 14.12: Motion Segmentation Results. IoU scores on KITTI 2015 training dataset images computed only over car pixels.

Method	Pixel Acc.	Mean Acc.	Mean IoU	f.w. IoU
Yang *et al.*[74] *	0.89	0.75	0.52	0.87
Luo *et al.*[75]	0.88	0.63	0.50	0.86
Luo *et al.*[75] *	0.91	0.76	0.53	0.87
Wang *et al.*[238] (Full) **	0.90	0.82	0.56	0.88
Ranjan *et al.*[73]	0.87	0.79	0.53	0.85
ΩNet	**0.98**	**0.86**	**0.75**	**0.97**
ΩNet (Proxy [44])	0.98	0.87	0.77	0.97

TABLE 14.13: Motion segmentation evaluation on the KITTI 2015 dataset. *: trained on stereo pairs, **: using stereo at testing time.

14.3.8 Runtime analysis

Finally, we measure the runtime of ΩNet on different hardware devices, i.e. a Titan Xp GPU, an embedded NVIDIA Jetson TX2 board and an Intel i7-7700K@4.2 GHz CPU. Timings averaged over 200 frames. Moreover, as each component of ΩNet may be used on its own, we report the runtime for each independent task. As summarized in Table 14.14, at 640 × 192, our network runs in real-time on the Titan Xp GPU and at about 2.5 FPS on a standard CPU. It also fits the low-power NIVIDA Jetson TX2, achieving 4.5 FPS to compute all the outputs. Moreover, it can be noticed how ΩNet achieves real-time results (i.e. 27.9) on the Titan Xp GPU even with the largest image size 1024 × 320, reaching about 2 FPS on the Jetson Tx2 embedded device with the same input configuration.

	416 × 128						640 × 192					1024 × 320				
	W	D	DS	OF	Cam	O	D	DS	OF	Cam	O	D	DS	OF	Cam	O
Jetson TX2	15	20.2	17.9	8.9	54.1	7.1	12.5	10.3	6.5	49.2	4.5	6.4	5.3	3.2	26.31	2.0
i7-7700K	91	10.9	9.1	11.0	60.1	5.5	5.0	4.2	4.9	31.4	2.4	1.9	1.6	1.8	13.2	0.9
Titan XP	250	250.7	212.4	152.6	550.7	90.5	170.2	134.1	94.1	446.7	57.4	86.0	64.5	44.5	251.0	27.9

TABLE 14.14: Runtime analysis on different hardware devices. For each device we report the power consumption in Watt and the FPS by varying input resolution. D: Depth, S: Semantic, OF: Optical Flow, Cam: camera pose, O: Overall architecture.

14.3.9 Results on a YouTube Video

Furthermore, to prove that our network can be trained on unconstrained monocular sequences with unknown camera parameters and without semantic ground-truth labels, we downloaded from YouTube an online video

captured by a moving camera consisting of 130K images depicting an urban scenario. Then, we generated proxy semantic labels using [235] and trained ΩNet(DSNet) to learn depth, pose, semantics and camera intrinsics. Figure 14.5, show qualitative results yielded by ΩNet on this unconstrained monocular video.

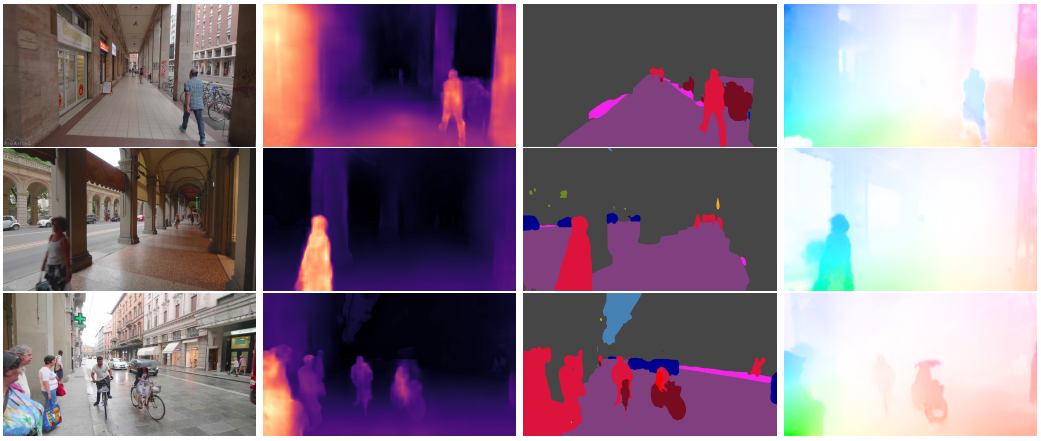

FIGURE 14.5: Qualitative results of ΩNet on a raw YouTube video. From left to right, we show the input images of a monocular sequence, the single-view depth and semantic predictions and, finally, the optical flow estimate.

Chapter 15

Final Remarks

In this part of the thesis, we have shown how leveraging task synergies in a single-domain multi-task framework fashion allows achieving better performance with lighter models and less supervision.

First of all, in chapter 13 we have focused on the joint learning of the geometry and semantics of the scene. We have proposed a deep learning architecture to improve unsupervised monocular depth estimation by leveraging semantic information. Moreover, we have shown how training our architecture end-end to infer semantics and depth jointly enables us to outperform the state-of-the-art approach for unsupervised monocular depth estimation [57].

Then, in chapter 14 we have extended our goal, by understanding the scene in a more comprehensive manner with a joint learning of geometry, semantics, and motion. In particular, we have proposed the first real-time network for comprehensive scene understanding from monocular videos. Our framework can estimate depth, optical flow, semantic segmentation, camera pose, and motion masks at about 60 FPS on high-end GPU and 5FPS on embedded systems. Moreover, we train our framework without any ground-truth data, using a mixture of self-supervision, a novel learning procedure based on semantic proxy labels distillation, and a semantic-aware self-distillation of optical-flow information. Thanks to this original paradigm, we have demonstrated state-of-the-art performance on standard benchmark datasets for depth, optical flow, and motion segmentation.

In conclusion, while in the second part of the thesis we have shown that we can exploit the correlation between visual tasks to achieve better generalization across domains, here we have shown that we can also boost performance and efficiency in a single domain scenario, still using no ground-truths for supervision. These findings support the hypothesis that leveraging on tasks dependencies is the key to unleash the power of deep learning. Indeed, we can craft models that are more robust, more accurate, and lighter

while requiring less supervision to be trained.

Part IV

Final Remarks

Chapter 16

Conclusions

In this thesis, we have investigated on the utilization of deep learning for scene understanding in situations where data are scarcely available.

In the first part, we have demonstrated that it is possible to employ computer graphics to speed up data collection for semantic segmentation by creating new efficient annotation tools or by synthetic renderings.

In chapter 4 we have proposed a novel tool based on virtual reality to make the annotation process less tedious and time-consuming. With our game-style interface, we enlarge the pool of users able to utilize our software. Thus, we can gather multiple annotations of the same scene in short-times. For the above reason, we also have proposed a novel post-process algorithm to auto-refine the annotated data.

In chapter 5, instead of using computer graphics as an efficient human-computer interface, we have investigated a different kind of approach. We explored the possibility of using synthetic data rendered from a simulation of the world and the domain-shift problem of using such data. We propose a novel image-to-image translation technique to mitigate this problem. Peculiarly to our work, we use semantic information to preserve the image content before and after the translation process. Notably, we have shown several experiments where we dramatically improve the style-transfer results, reducing the artifacts introduced by traditional approaches. Moreover, we have shown how training on our transformed data can decrease the performance gap between synthetic and real data.

In the last two parts of the thesis, we have investigated a novel research direction: employing the relationships between visual tasks (e.g. semantic segmentation and depth estimation) to obtain more general and accurate models.

In the second part of this manuscript, we have explored the possibility of using these relationships to decrease the need for labeled data. In chapter 8 we propose AT/DT, a novel framework that transfer the knowledge across

tasks and domains. As done in chapter 5 we consider two domains, synthetic and real. However, this time we consider multiple tasks at once. We have shown that we can learn the correlation between deep features learned by two task-specific CNNs (e.g. semantic and depth networks) by another network that we call transfer network. Surprisingly, the transfer network generalizes well across domains, showing that we can decrease the domain-shift by exploiting the relationship between visual tasks.

Based on these findings, we also propose an improved version of AT/DT, reported in chapter 9. Here we focus on the significant weaknesses of AT/DT in the particular scenario of having two dense tasks such as semantic segmentation and depth estimation. We propose two techniques to improve the transfer function quality by aligning its inputs across tasks and domains.

Finally, in chapter 10 we employ AT/DT also in the benchmark for unsupervised domain adaptation for semantic segmentation. We choose depth estimation as a pretext-task because we have already noticed its strict correlation with semantic segmentation and because it can be trained with self-supervision. We notice that transferring the knowledge from depth to semantics with AT/DT brings complementary information to standard domain adaptation techniques. Thus, we propose a technique to merge the knowledge transferred across task with AT/DT and domain adaptation methods. We can achieve state-of-the-art results in the benchmark through the proposed merging strategy and a novel self-training protocol.

In the third part of the thesis, we examined the possibility of exploiting the relationship between visual tasks to obtain more accurate and efficient models, still in the context of a low-data regime but in a single-domain scenario. In chapter 13 we have shown that learning depth and semantic together by a single neural network leads to improved performance in the depth estimation task. We already highlighted the existence of a correlation between the two tasks in previous experiments. However, we have shown here that we can use this relationship to improve the accuracy of the model when learning them jointly. Notably, we train our model self-supervised for depth estimation and only a few semantic segmentation labels.

In chapter 14, we extend the previous idea and we learn a single model for a comprehensive scene understanding, trying to exploit the synergy among several visual tasks. We consider a video sequence as inputs and the several tasks: depth, pose, and normal estimation jointly with motion and semantic segmentation. Our multi-task model can achieve state-of-the-art performances in depth, optical flow, and motion segmentation without training on

any manually annotated labels. Moreover, our framework is lightweight and can run in real-time on a standard GPU.

In conclusion, this thesis proposes several approaches for scene understanding, with only a few annotated data. We argue that the results also highlight another crucial finding: modeling the relationships between visual tasks is the first essential step to craft models for a comprehensive scene understanding that are more robust, more accurate, and need less supervision. We hypothesize that incorporating such multi-task knowledge is key to have models that can be safely applied to the real world. Therefore, we propose to deepen these discoveries in the future to find a universal framework for scene understanding hope that our findings may pave the way for further research towards a more systematic and unified approach aimed at solving many visual tasks jointly so as to fully leverage on their synergies and boost the ability of deep models to pursue comprehensive scene perception without any explicit supervision.

Bibliography

[7] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. 1982.

[8] H. B. Barlow. "Unsupervised learning. Neural computation". In: 1989.

[9] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 1848829345.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[12] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[13] Marco Toldo, Andrea Maracani, Umberto Michieli, and Pietro Zanuttigh. "Unsupervised Domain Adaptation in Semantic Segmentation: a Review". In: *arXiv preprint arXiv:2005.10876* (2020).

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R Benenson, U. Franke, S. Roth, and B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[15] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. "The ApolloScape Dataset for Autonomous Driving". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.

[16] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. "The mapillary vistas dataset for semantic understanding of street scenes". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4990–4999.

[17] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111.1 (2015), pp. 98–136.

[18] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. "Scene parsing through ade20k dataset". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 633–641.

[19] Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan. "Human parsing with contextualized convolutional neural network". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1386–1394.

[20] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *2016 fourth international conference on 3D vision (3DV)*. IEEE. 2016, pp. 565–571.

[21] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for data: Ground truth from computer games". In: *European Conference on Computer Vision*. Springer. 2016, pp. 102–118.

[22] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung. "A robust 3d-2d interactive tool for scene segmentation and annotation". In: *IEEE transactions on visualization and computer graphics* 24.12 (2018), pp. 3005–3018.

[23] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. "Understanding real world indoor scenes with synthetic data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4077–4085.

[24] 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: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3234–3243.

[25] 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.

[26] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation". In: *CVPR*. IEEE. 2008, pp. 1–8.

[27] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. "Class segmentation and object localization with superpixel neighborhoods". In: *ICCV*. IEEE. 2009, pp. 670–677.

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[29] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[30] 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.

[31] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation". In: *CVPR* (2017).

[32] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2650–2658.

[33] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. "Attention to scale: Scale-aware semantic image segmentation". In: *CVPR*. 2016, pp. 3640–3649.

[34] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. "Semantic image segmentation with deep convolutional nets and fully connected crfs". In: *ICLR*. 2015.

[35] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[36] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. "Conditional random fields as recurrent neural networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1529–1537.

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid scene parsing network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[39] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[40] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017).

[41] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. "Deformable convolutional networks". In: *CoRR, abs/1703.06211* 1.2 (2017), p. 3.

[42] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. "Understanding convolution for semantic segmentation". In: *WACV* (2018).

[43] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. "Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation". In: *CVPR*. 2019.

[44] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. "Searching for efficient multi-scale architectures for dense image prediction". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8699–8710.

[45] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation". In: *CoRR* abs/1606.02147 (2016). URL: http://arxiv.org/abs/1606.02147.

[46] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. "Bisenet: Bilateral segmentation network for real-time semantic segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 325–341.

[47] Gen Li and Joongkyu Kim. "DABNet: Depth-wise Asymmetric Bottleneck for Real-time Semantic Segmentation". In: *British Machine Vision Conference*. 2019.

[48] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. "Dfanet: Deep feature aggregation for real-time semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9522–9531.

[49] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. "HarDNet: A Low Memory Traffic Network". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3552–3561.

[50] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. "Deeper depth prediction with fully convolutional residual networks". In: *3DV*. 2016, pp. 239–248.

[51] 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.

[52] Xiaolong Wang, David Fouhey, and Abhinav Gupta. "Designing deep networks for surface normal estimation". In: *CVPR*. 2015, pp. 539–547.

[53] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. "Estimating depth from monocular images as classification using deep fully convolutional residual networks". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2017).

[54] Ashutosh Saxena, Min Sun, and Andrew Y Ng. "Make3d: Learning 3d scene structure from a single still image". In: *TPAMI* 31.5 (2009), pp. 824–840.

[55] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. "Sparsity Invariant CNNs". In: *3DV*. 2017.

[56] 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.

[57] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *CVPR*. 2017.

[58] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[59] 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.

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

[61] 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.

[62] 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.

[63] Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. "Learning monocular depth estimation infusing traditional stereo knowledge". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[64] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. "Self-Supervised Monocular Depth Hints". In: *ICCV*. 2019.

[65] Lorenzo Andraghetti, Panteleimon Myriokefalitakis, Pier Luigi Dovesi, Belen Luque, Matteo Poggi, Alessandro Pieropan, and Stefano Mattoccia. "Enhancing self-supervised monocular depth estimation with traditional visual odometry". In: *7th International Conference on 3D Vision (3DV)*. 2019.

[66] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. "Unsupervised Learning of Depth and Ego-Motion From Video". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[67] 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.

[68] 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.

[69] Zhenheng Yang, Peng Wang, Wang Yang, Wei Xu, and Nevatia Ram. "LEGO: Learning edge with geometry all at once by watching videos". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[70] 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.

[71] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. "DF-Net: Unsupervised Joint Learning of Depth and Flow using Cross-Task Consistency". In: *European Conference on Computer Vision (ECCV)*. 2018.

[72] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. "Self-supervised Learning with Geometric Constraints in Monocular Video: Connecting Flow, Depth, and Camera". In: *ICCV*. 2019.

[73] Ranjan Anurag, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. "Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[74] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. "Every Pixel Counts: Unsupervised Geometry Learning with Holistic 3D Motion Understanding". In: *The European Conference on Computer Vision (ECCV) Workshops*. 2018.

[75] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. "Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding". In: *PAMI* (2019).

[76] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. "Depth Prediction without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos". In: *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*. 2019.

[77] Haofei Xu, Jianmin Zheng, Jianfei Cai, and Juyong Zhang. "Region Deformer Networks for Unsupervised Depth Estimation from Unconstrained Monocular Videos". In: *IJCAI*. 2019.

[78] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. "Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras". In: *ICCV*. 2019.

[79] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Digging into self-supervised monocular depth estimation". In: *ICCV*. 2019.

[80] Junsheng Zhou, Yuwang Wang, Naiyan Wang, and Wenjun Zeng. "Unsupervised High-Resolution Depth Learning from Videos with Dual Networks". In: *Inter. Conf. on Computer Vision*. IEEE. IEEE, 2019.

[81] Jia-Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. "Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video". In: *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*. 2019.

[82] 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.

[83] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. "On the uncertainty of self-supervised monocular depth estimation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[84] Valentino Peluso, Antonio Cipolletta, Andrea Calimera, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. "Enabling Energy-Efficient Unsupervised Monocular Depth Estimation on ARMv7-Based Platforms".

In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*. 2019, pp. 1703–1708.

[85] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. "FastDepth: Fast Monocular Depth Estimation on Embedded Systems". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[86] Ya-Liang Chang, Zhe Yu Liu, and Winston Hsu. "VORNet: Spatio-temporally Consistent Video Inpainting for Object Removal". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[87] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. "Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[88] Yu Xiang, Alexandre Alahi, and Silvio Savarese. "Learning to track: Online multi-object tracking by decision making". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4705–4713.

[89] Berthold KP Horn and Brian G Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[90] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. "High accuracy optical flow estimation based on a theory for warping". In: *European conference on computer vision*. Springer. 2004, pp. 25–36.

[91] Michael J Black and Paul Anandan. "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields". In: *Computer vision and image understanding* 63.1 (1996), pp. 75–104.

[92] J. Revaud, P Weinzaepfel, Z. Harchaoui, and C. Schmid. "Epicflow: Edge-preserving interpolation of correspondences for optical flow". In: *CVPR*. 2019.

[93] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks". In: *ICCV*. 2015.

[94] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[95] 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.

[96] 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.

[97] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. "Lite-flownet: A lightweight convolutional neural network for optical flow estimation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[98] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik Sudderth, and Jan Kautz. "A fusion approach for multi-frame optical flow estimation". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 2077–2086.

[99] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.

[100] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Bin, and Hongyuan Zha. "Unsupervised deep learning for optical flow estimation". In: *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. 2017.

[101] Wei-Shi Zheng Shuosen Guan Haoxin Li. "Unsupervised Learning for Optical Flow Estimation Using Pyramid Convolution LSTM". In: *Proceedings of IEEE International Conference on Multimedia and Expo(ICME)*. 2019.

[102] Simon Meister, Junhwa Hur, and Stefan Roth. "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss". In: *AAAI*. New Orleans, Louisiana, 2018.

[103] Wang Yang, Yi Yang, Zhenheng Yang, Liang Zhao, and Wei Xu. "Occlusion aware unsupervised learning of optical flow." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[104] Pengpeng Liu, Irwin King, Michael R. Lyu, and Jia Xu. "DDFlow: Learning Optical Flow with Unlabeled Data Distillation". In: *AAAI*. 2019.

[105] Pengpeng Liu, Michael R. Lyu, Irwin King, and Jia Xu. "SelFlow: Self-Supervised Learning of Optical Flow". In: *CVPR*. 2019.

[106] Joel Janai, Fatma G"uney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. "Unsupervised Learning of Multi-Frame Optical Flow with Occlusions". In: *European Conference on Computer Vision (ECCV)*. Vol. Lecture Notes in Computer Science, vol 11220. Springer, Cham, 2018, pp. 713–731.

[107] Junhwa Hur and Stefan Roth. "Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5754–5763.

[108] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[109] 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.

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

[111] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[112] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. "Scenenn: A scene meshes dataset with annotations". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 92–101.

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

[114] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. "Joint 2d-3d-semantic data for indoor scene understanding". In: *arXiv preprint arXiv:1702.01105* (2017).

[115] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *International Conference on 3D Vision (3DV)* (2017).

[116] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. "Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2678–2687.

[117] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. "InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset". In: *British Machine Vision Conference (BMVC)*. 2018.

[118] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. 2017, pp. 91–98.

[119] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. "A Dataset for Semantic Segmentation of Point Cloud Sequences". In: *arXiv preprint arXiv:1904.01416* (2019).

[120] Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi, and Philip Torr. "Semanticpaint: Interactive 3d labeling and learning at your fingertips". In: *ACM Transactions on Graphics (TOG)* 34.5 (2015), p. 154.

[121] Ondrej Miksik, Vibhav Vineet, Morten Lidegaard, Ram Prasaath, Matthias Nießner, Stuart Golodetz, Stephen L Hicks, Patrick Pérez, Shahram Izadi, and Philip HS Torr. "The semantic paintbrush: Interactive 3d

mapping and recognition in large outdoor spaces". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 3317–3326.

[122] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. "Visual domain adaptation: A survey of recent advances". In: *IEEE signal processing magazine* 32.3 (2015), pp. 53–69.

[123] Mei Wang and Weihong Deng. "Deep visual domain adaptation: A survey". In: *Neurocomputing* 312 (2018). ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.05.083. URL: http://dx.doi.org/10.1016/j.neucom.2018.05.083.

[124] Gabriela Csurka. "A comprehensive survey on domain adaptation for visual applications". In: *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 1–35.

[125] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. "Geodesic flow kernel for unsupervised domain adaptation". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2066–2073.

[126] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. "Domain adaptation for object recognition: An unsupervised approach". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 999–1006.

[127] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. "Learning Transferable Features with Deep Adaptation Networks". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 97–105.

[128] Yaroslav Ganin and Victor Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *International Conference on Machine Learning*. 2015, pp. 1180–1189.

[129] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-adversarial training of neural networks". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030.

[130] 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.

[131] Ming-Yu Liu and Oncel Tuzel. "Coupled generative adversarial networks". In: *Advances in neural information processing systems*. 2016, pp. 469–477.

[132] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. "Larger Norm More Transferable: An Adaptive Feature Norm Approach for Unsupervised Domain Adaptation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1426–1435.

[133] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation". In: *arXiv preprint arXiv:1612.02649* (2016).

[134] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. "Simultaneous deep transfer across domains and tasks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4068–4076.

[135] Yang Zhang, Philip David, and Boqing Gong. "Curriculum domain adaptation for semantic segmentation of urban scenes". In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 5. 2017, p. 6.

[136] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. "Learning to Adapt Structured Output Space for Semantic Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00780. URL: http://dx.doi.org/10.1109/CVPR.2018.00780.

[137] Weixiang Hong, Zhenzhen Wang, Ming Yang, and Junsong Yuan. "Conditional Generative Adversarial Network for Structured Domain Adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1335–1344.

[138] Yi-Hsuan Tsai, Kihyuk Sohn, Samuel Schulter, and Manmohan Chandraker. "Domain Adaptation for Structured Output via Discriminative Patch Representations". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00154. URL: http://dx.doi.org/10.1109/ICCV.2019.00154.

[139] Haoran Wang, Tong Shen, Wei Zhang, Lingyu Duan, and Tao Mei. "Classes Matter: A Fine-grained Adversarial Approach to Cross-domain Semantic Segmentation". In: *The European Conference on Computer Vision (ECCV)*. 2020.

[140] Y. Zhang and Zilei Wang. "Joint Adversarial Learning for Domain Adaptation in Semantic Segmentation". In: *AAAI*. 2020.

[141] Jihan Yang, Ruijia Xu, Ruiyu Li, Xiaojuan Qi, Xiaoyong Shen, Guanbin Li, and Liang Lin. "An Adversarial Perturbation Oriented Domain Adaptation Approach for Semantic Segmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2020).

[142] Fabio Pizzati, Raoul de Charette, Michela Zaccaria, and Pietro Cerri. "Domain Bridge for Unpaired Image-to-Image Translation and Unsupervised Domain Adaptation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020.

[143] Matteo Biasetton, Umberto Michieli, Gianluca Agresti, and Pietro Zanuttigh. "Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

[144] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. "ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). DOI: 10.1109/cvpr.2019.00262. URL: http://dx.doi.org/10.1109/CVPR.2019.00262.

[145] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. "Learning from simulated and unsupervised images through adversarial training". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 3. 4. 2017, p. 6.

[146] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. "T2Net: Synthetic-to-Realistic Translation for Solving Single-Image Depth Estimation Tasks". In: *The European Conference on Computer Vision (ECCV)*. 2018.

[147] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). DOI: 10.1109/cvpr.2017.18. URL: http://dx.doi.org/10.1109/CVPR.2017.18.

[148] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-To-Image Translation With Conditional Adversarial Networks". In:

The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

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

[150] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. "Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00395. URL: http://dx.doi.org/10.1109/CVPR.2018.00395.

[151] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. *CyCADA: Cycle-Consistent Adversarial Domain Adaptation*. 2017. arXiv: 1711.03213 [cs.CV].

[152] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. "Fully Convolutional Adaptation Networks for Semantic Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00712. URL: http://dx.doi.org/10.1109/CVPR.2018.00712.

[153] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Peng, and Wei-Chen Chiu. "All About Structure: Adapting Structural Information Across Domains for Boosting Semantic Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). DOI: 10.1109/cvpr.2019.00200. URL: http://dx.doi.org/10.1109/CVPR.2019.00200.

[154] Jaehoon Choi, Taekyung Kim, and Changick Kim. "Self-Ensembling With GAN-Based Data Augmentation for Domain Adaptation in Semantic Segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00693. URL: http://dx.doi.org/10.1109/ICCV.2019.00693.

[155] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. "Image to Image Translation for Domain Adaptation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00473. URL: http://dx.doi.org/10.1109/CVPR.2018.00473.

[156] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. "Bidirectional Learning for Domain Adaptation of Semantic Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). DOI: 10.1109/cvpr.2019.00710. URL: http://dx.doi.org/10.1109/CVPR.2019.00710.

[157] Myeongjin Kim and Hyeran Byun. "Learning Texture Invariant Representation for Domain Adaptation of Semantic Segmentation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). DOI: 10.1109/cvpr42600.2020.01299. URL: http://dx.doi.org/10.1109/cvpr42600.2020.01299.

[158] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-mei Hwu, Thomas S. Huang, and Honghui Shi. "Differential Treatment for Stuff and Things: A Simple Unsupervised Domain Adaptation Method for Semantic Segmentation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). DOI: 10.1109/cvpr42600.2020.01265. URL: http://dx.doi.org/10.1109/cvpr42600.2020.01265.

[159] D. Lee. "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". In: *Workshop on challenges in representation learning, ICML*. 2013.

[160] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. "Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 289–305.

[161] Yang Zou, Zhiding Yu, Xiaofeng Liu, B.V.K. Vijaya Kumar, and Jinsong Wang. "Confidence Regularized Self-Training". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[162] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. "Instance Adaptive Self-Training for Unsupervised Domain Adaptation". In: *The European Conference on Computer Vision (ECCV)*. 2020.

[163] Zhedong Zheng and Yi Yang. "Unsupervised Scene Adaptation with Memory Regularization in vivo". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (2020). DOI: 10.24963/ijcai.2020/150. URL: http://dx.doi.org/10.24963/ijcai.2020/150.

[164] Fei Pan, Inkyu Shin, Francois Rameau, Seokju Lee, and In So Kweon. "Unsupervised Intra-Domain Adaptation for Semantic Segmentation Through Self-Supervision". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). DOI: 10.1109/cvpr42600.2020.00382. URL: http://dx.doi.org/10.1109/cvpr42600.2020.00382.

[165] Zhedong Zheng and Yi Yang. "Rectifying Pseudo Label Learning via Uncertainty Estimation for Domain Adaptive Semantic Segmentation". In: *International Journal of Computer Vision (IJCV)* (2020). DOI: 10.1007/s11263-020-01395-y.

[166] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. "DACS: Domain Adaptation via Cross-domain Mixed Sampling". In: *The European Conference on Computer Vision (ECCV)*. 2020.

[167] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[168] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 667–676.

[169] Camille Couprie, Clément Farabet, Laurent Najman, and Yann Lecun. "Indoor semantic segmentation using depth information". In: *International Conference on Learning Representations (ICLR2013), April 2013*. 2013.

[170] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. "Unsupervised adaptation for deep stereo". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1605–1613.

[171] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for benchmarks". In: *International Conference on Computer Vision (ICCV)*. 2017.

[172] Makoto Yamada, Leonid Sigal, and Yi Chang. "Domain adaptation for structured regression". In: *International journal of computer vision* 109.1-2 (2014), pp. 126–145.

[173] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping". In: *International Conference on Robotics and Automation*. 2018.

[174] 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.

[175] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Ya Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. "CyCADA: Cycle Consistent Adversarial Domain Adaptation". In: *International Conference on Machine Learning (ICML)*. 2018.

[176] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. "Taskonomy: Disentangling task transfer learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3712–3722.

[177] Daniel Hernandez-Juarez, Lukas Schneider, Antonio Espinosa, David Vazquez, Antonio M. Lopez, Uwe Franke, Marc Pollefeys, and Juan Carlos Moure. "Slanted Stixels: Representing San Francisco's Steepest Streets". In: *British Machine Vision Conference (BMVC), 2017*. 2017.

[178] Moritz Menze and Andreas Geiger. "Object Scene Flow for Autonomous Vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[179] Maria Klodt and Andrea Vedaldi. "Supervising the new with the old: learning SFM from SFM". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 698–713.

[180] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 817–833.

[181] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[182] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), 1137–1149. ISSN: 2160-9292. DOI: 10.1109/tpami.2016.2577031. URL: http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[183] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017). DOI: 10.1109/iccv.2017.322. URL: http://dx.doi.org/10.1109/ICCV.2017.322.

[184] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[185] Kaiming He, Ross Girshick, and Piotr Dollár. *Rethinking ImageNet Pre-training*. 2018. arXiv: 1811.08883 [cs.CV].

[186] Arghya Pal and Vineeth N Balasubramanian. *Zero-Shot Task Transfer*. 2019.

[187] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. "Label efficient learning of transferable representations acrosss domains and tasks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 165–177.

[188] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir Rawashdeh. *AuxNet: Auxiliary tasks enhanced Semantic Segmentation for Automated Driving*. 2019. arXiv: 1901.05808 [cs.CV].

[189] Roberto Cipolla, Yarin Gal, and Alex Kendall. "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00781. URL: http://dx.doi.org/10.1109/CVPR.2018.00781.

[190] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[191] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes". In: *International Journal of Computer Vision (IJCV)* (2018).

[192] Heiko Hirschmuller. "Accurate and efficient stereo processing by semi-global matching and mutual information". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 2. IEEE. 2005, pp. 807–814.

[193] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated Residual Networks". In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.

[194] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456. URL: http://dl.acm.org/citation.cfm?id=3045118.3045167.

[195] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* (2008).

[196] Xavier Soria, Edgar Riba, and Angel Sappa. "Dense Extreme Inception Network: Towards a Robust CNN Model for Edge Detection". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV '20)*. 2020.

[197] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. "Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 53–69.

[198] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. "Self-supervised Monocular Depth Estimation: Solving the Dynamic Object Problem by Semantic Guidance". In: *Lecture Notes in Computer Science* (2020), 582–600. ISSN: 1611-3349. DOI: 10.1007/978-3-030-58565-5_35. URL: http://dx.doi.org/10.1007/978-3-030-58565-5_35.

[199] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. "Semantically-Guided Representation Learning for Self-Supervised Monocular Depth". In: *Proceedings of the Eighth International Conference on Learning Representations (ICLR)*. 2020.

[200] Kohei Watanabe, Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. "Multichannel Semantic Segmentation with Unsupervised Domain Adaptation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.

[201] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez Perez. "DADA: Depth-Aware Domain Adaptation in Semantic Segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00746. URL: http://dx.doi.org/10.1109/ICCV.2019.00746.

[202] Kuan-Hui Lee, German Ros, Jie Li, and Adrien Gaidon. "Spigan: Privileged adversarial learning from simulation". In: *International Conference on Learning Representations*. 2019.

[203] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. "Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1841–1850.

[204] Chao Zhou, Hong Zhang, Xiaoyong Shen, and Jiaya Jia. "Unsupervised Learning of Stereo Matching". In: *ICCV*. 2017.

[205] Qing Lian, Lixin Duan, Fengmao Lv, and Boqing Gong. "Constructing Self-Motivated Pyramid Curriculums for Cross-Domain Semantic Segmentation: A Non-Adversarial Approach". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00686. URL: http://dx.doi.org/10.1109/ICCV.2019.00686.

[206] 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.

[207] Minghao Chen, Hongyang Xue, and Deng Cai. "Domain Adaptation for Semantic Segmentation With Maximum Squares Loss". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00218. URL: http://dx.doi.org/10.1109/ICCV.2019.00218.

[208] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. *ClassMix: Segmentation-Based Data Augmentation for Semi-Supervised Learning*. 2020. arXiv: 2007.07936.

[209] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[210] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[211] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[212] Leslie N. Smith and Nicholay Topin. "Super-convergence: very fast training of neural networks using large learning rates". In: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications* (2019). Ed. by TienEditor Pham. DOI: 10.1117/12.2520589. URL: http://dx.doi.org/10.1117/12.2520589.

[213] Iasonas Kokkinos. "UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). DOI: 10.1109/cvpr.2017.579. URL: http://dx.doi.org/10.1109/CVPR.2017.579.

[214] Carl Doersch and Andrew Zisserman. "Multi-task Self-Supervised Visual Learning". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017). DOI: 10.1109/iccv.2017.226. URL: http://dx.doi.org/10.1109/ICCV.2017.226.

[215] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. "Dynamic Task Prioritization for Multitask Learning". In: *The European Conference on Computer Vision (ECCV)*. 2018.

[216] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. "Robust Learning Through Cross-Task Consistency". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11197–11206.

[217] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. "Towards unified depth and semantic prediction from a single image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2800–2809.

[218] Arsalan Mousavian, Hamed Pirsiavash, and Jana Košecká. "Joint semantic segmentation and depth estimation with deep convolutional networks". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 611–619.

[219] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weight losses for scene geometry and semantics". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7482–7491.

[220] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. "Joint task-recursive learning for semantic segmentation and depth estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 235–251.

[221] Pier Luigi Dovesi, Matteo Poggi, Lorenzo Andraghetti, Miquel Martí, Hedvig Kjellström, Alessandro Pieropan, and Stefano Mattoccia. "Real-Time Semantic Stereo Matching". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[222] Po-Yi Chen, Alexander H. Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. "Towards Scene Understanding: Unsupervised Monocular Depth Estimation with Semantic-aware Representation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[223] Junhwa Hur and Stefan Roth. "Joint optical flow and temporally consistent semantic segmentation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 163–177.

[224] Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J Black. "Optical flow with semantic segmentation and localized layers". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3889–3898.

[225] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. "Exploiting semantic information and deep matching for optical flow". In: *European Conference on Computer Vision*. Springer. 2016, pp. 154–170.

[226] Hazem Rashed, Senthil Yogamani, Ahmad El-Sallab, Pavel Krizek, and Mohamed El-Helw. "Optical flow augmented semantic segmentation networks for automated driving". In: *arXiv preprint arXiv:1901.07355* (2019).

[227] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. "Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes". In: *BMVC*. 2017.

[228] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (2015).

[229] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR*. 2016, pp. 770–778.

[230] 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.

[231] Zhichao Yin and Jianping Shi. "Geonet: Unsupervised learning of dense depth, optical flow and camera pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1983–1992.

[232] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: (2018).

[233] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *CVPR*. 2017.

[234] 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.

[235] Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jonathon Shlens. "Searching for Efficient Multi-Scale Architectures for Dense Image Prediction". In: *NIPS*. 2018.

[236] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *CVPR*. 2012.

[237] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *CVPR*. 2016.

[238] Yang Wang, Peng Wang, Zhenheng Yang, Chenxu Luo, Yi Yang, and Wei Xu. "UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8071–8081.