Received September 18, 2020, accepted October 11, 2020, date of publication October 19, 2020, date of current version November 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032024

Co-Training for On-Board Deep Object Detection

GABRIEL VILLALONGA AND ANTONIO M. LÓPEZ PEÑA, (Member, IEEE)
Computer Vision Center (CVC), Universitat Autònoma de Barcelona (UAB), 08193 Bellaterra, Barcelona, Spain
Computer Science Department, Universitat Autònoma de Barcelona (UAB), 08193 Bellaterra, Barcelona, Spain

Corresponding author: Gabriel Villalonga (gvillalonga@cvc.uab.es)

The authors acknowledge the financial support received for this work from the Spanish TIN2017-88709-R (MINECO/AEI/FEDER, UE) project. Antonio acknowledges the financial support to his general research activities given by ICREA under the ICREA Academia Program. The authors acknowledge the support of the Generalitat de Catalunya CERCA Program as well as its ACCIO agency to CVC's general activities.

ABSTRACT Providing ground truth supervision to train visual models has been a bottleneck over the years, exacerbated by domain shifts which degenerate the performance of such models. This was the case when visual tasks relied on handcrafted features and shallow machine learning and, despite its unprecedented performance gains, the problem remains open within the deep learning paradigm due to its data-hungry nature. Best performing deep vision-based object detectors are trained in a supervised manner by relying on human-labeled bounding boxes which localize class instances (*i.e.* objects) within the training images. Thus, object detection is one of such tasks for which human labeling is a major bottleneck. In this article, we assess co-training as a semi-supervised learning method for self-labeling objects in unlabeled images, so reducing the human-labeling effort for developing deep object detectors. Our study pays special attention to a scenario involving domain shift; in particular, when we have automatically generated virtual-world images with object bounding boxes and we have real-world images which are unlabeled. Moreover, we are particularly interested in using co-training for deep object detection in the context of driver assistance systems and/or self-driving vehicles. Thus, using well-established datasets and protocols for object detection in these application contexts, we will show how co-training is a paradigm worth to pursue for alleviating object labeling, working both alone and together with task-agnostic domain adaptation.

INDEX TERMS Co-training, domain adaptation, vision-based object detection, ADAS, self-driving.

I. INTRODUCTION

Since more than two decades ago, machine learning (ML) has been the enabling technology to solve computer vision tasks. In the last decade, traditional ML, *i.e.* based on relatively shallow classifiers and hand-crafted features, has given way to deep learning (DL). Thanks to DL models based on convolutional neural networks (CNNs), DL approaches significantly outperformed traditional ML in all kinds of computer vision tasks, such as image classification, object detection, semantic segmentation, etc. A major key for the usefulness of DL models is to train them in a supervised way. In other words, the raw data (still images and videos) need to be supplemented by ground truth; nowadays, usually collected via crowd-sourced labeling. In practice, due to the data-hungry nature of CNNs, data labeling is considered a major bottleneck. Therefore, approaches to minimize labeling effort are of high interest; or put in another way, approaches to automatically leverage the large amounts of available unlabeled raw data must be pursued. Thus, we can find different families of algorithms, or paradigms, which address human-labeling minimization under different working assumptions. We continue this introduction by reviewing them in subsection I-A, while subsection I-B highlights how domain shifts exacerbate human labeling requirements. With the human-labeling bottleneck problem introduced, subsection I-C summarizes our focus when addressing such a problem. In particular, we outline our application context and the proposed approach for reducing human annotation effort in such a context. Finally, subsection I-D highlights the main contributions presented in this article as well as the organization of the rest of its contents.

The associate editor coordinating the review of this manuscript and approving it for publication was Yongqiang Zhao.

A. PARADIGMS TO MINIMIZE HUMAN LABELING

For instance, *active learning* (AL) approaches [1], [34], [35] assume an initial model and an unlabeled dataset to be labeled by a human (oracle) following an interactive procedure. In particular, the current model processes the unlabeled data providing so-called *pseudo-labels* (at image, object, or pixel level); then, these results are inspected, either automatically or directly by the oracle, to select the next data to be labeled. Afterwards, the model is trained again, and the process repeated until fulfilling a stop criterion. AL assumes that with less labeling budget than the required to have the oracle labeling data at random, one can train models that perform at least similarly. In practice, it is usually hard to clearly outperform the oracle's random labeling strategy. In contrast to AL, other approaches do not assume human oracles in the labeling loop. For instance, this is the case of semi-supervised learning (SSL) algorithms [6], [38], [40], which assume the availability of a large number or raw unlabeled data together with a relatively small number of labeled data. Then, a model must be trained using the unlabeled and labeled data (without human intervention), with the goal of being more accurate than if only the labeled data were used. The so-called *self-training* and *co-training* algorithms, which are of our main interest in this article, fall in the SSL paradigm [40]. In AL and SSL, the most common idea is to efficiently label the unlabeled data, either via an oracle (AL) or automatically (SSL). *Self-supervised learning* (SfSL) follows a different approach which consists in providing supervision in the form of additional (relatively simple) tasks, known as pretext tasks, for which automatic ground truth can be easily generated (*e.g.* solving jigsaw puzzles [12], [19], [21]).

B. THE DOMAIN ADAPTATION PROBLEM

Within these approaches, the most classical assumption is that the labeled and the unlabeled data are drawn from the same distribution and one aims at using the unlabeled data (via annotations or pretext tasks) to solve the same tasks for which the labeled data was annotated. However, in practice, we may require to solve new tasks, leading to *transfer learning* (TL) [46], [54], or solving the same tasks in a new domain, leading to *domain adaptation* (DA) [10], [43], [47]. Beyond specific techniques to tackle TL or DA, we can leverage solutions/ideas from AL, SSL, or SfSL. For instance, AL [41], SSL [20], [55], and SfSL [49] algorithms have been used to address DA problems. In this article, we use SSL (comparing self-training and co-training) to address DA too.

C. THE FOCUS OF THIS WORK

From the application viewpoint, in this article, we focus on vision-based on-board deep object detection. Note that this is a very relevant visual task for driving, since detecting objects (*e.g.* vehicles, pedestrians, etc.) along the route of a vehicle is a key functionality for perception-decision-action pipelines in the context of both advanced driver assistance systems (ADAS) and self-driving vehicles. Moreover, nowadays, most accurate vision-based models to detect such objects are based on deep CNN architectures [23], [24], [30], [31]. In addition, in this application context, it is possible to acquire innumerable quantities of raw images, for instance, from cameras installed in fleets of cars. Thus, methods to minimize the effort of manually labeling them are of great relevance.

Among the SSL approaches, co-training [5], [15] has been explored for deep image classification [8], [29] with promising results. However, up to the best of our knowledge, it remains unexplored for deep object detection. Note that, while image classification aims at assigning image-level class/attribute labels, object detection is more challenging since it requires to localize and classify objects in images, *i.e.* placing a bounding box (BB) per object, together with the class label assigned to it. Moreover, object detection from on-board images, *i.e.* detection of vehicles, pedestrians, etc., is especially difficult since, to the inherent intra-class differences (*e.g.* due to vehicle models, pedestrian clothes, etc.), acquisition conditions add a vast variability because the objects appear in a large range of distances to the camera (resolution and focus variations), under different illumination conditions (from strong shadows to direct sunlight), and they usually move (blur, occlusion, view angle, and pose variations).

Originally, co-training relies on the *agreement* of two trained models performing predictions from different *features* (views) of the data [5]. These predictions are taken as pseudo-labels to improve the models incrementally. Later, prediction *disagreement* was shown to improve co-training results in applications related to natural language processing [15], [39]. Thus, in this article, we propose a co-training algorithm inspired by prediction *disagreement*. On the other hand, since co-training was proposed as a SSL method aiming at avoiding the drift problem of self-training [5], which incrementally re-trains a single model from its previously most confident predictions, we also elaborate a strong self-training baseline for our deep object detection problem.

We assess the effectiveness of the two self-labeling methods, *i.e.* self-training and co-training, in two different practical situations. First, following most classical SSL, we will assume access to a small dataset of labeled images, $\mathcal{X}^l$, together with a larger dataset of unlabeled images $\mathcal{X}^u$; where here the labels are object BBs with class labels. Second, we will address a relevant setting that resorts DA. In particular, we will assume access to a dataset of virtual-world images with automatically generated object labels. Therefore, eventually, $\mathcal{X}^l$ can be larger than $\mathcal{X}^u$, since labeling these images does not require human intervention; however, $\mathcal{X}^u$ is composed of real-world images, thus, between $\mathcal{X}^l$ and $\mathcal{X}^u$ there is a domain shift [41]. In this case, we also compare self-labeling techniques with task-agnostic DA, in particular, using GAN-based image-to-image translation [53]. As especially relevant cases, we will focus on on-board detection of vehicles and pedestrians using a deep CNN architecture. Our experiments will show how, indeed, our co-training algorithm is a good SSL alternative for on-board deep object detection. Co-training clearly boosts detection accuracy in regimes where the size of $\mathcal{X}^l$ is just the 5%/10% of the labeled data used to train an upper-bound object detector. Moreover, under the presence of domain shift, we will see how image-to-image GAN-based translation and co-training complement each other, allowing to reach almost upper-bound performances without human labeling.

D. CONTRIBUTIONS AND ORGANIZATION

Hence, the main contributions of this article are: (1) proposing a co-training algorithm for deep object detection; (2) designing this algorithm to allow addressing domain shift via GAN-based image-to-image translation; (3) showing its effectiveness by developing a strong self-training baseline and relying on publicly available evaluation standards and on-board image datasets. Alongside, we will also contribute with the public release of the virtual-world dataset we generated for our experiments. To show these achievements we organize the rest of the paper as follows. Section II reviews the most related works to ours, highlighting commonalities and differences. Section III details our self-training and co-training algorithms. Section IV draws our experimental setting, and discuss the obtained results. Finally, Section V summarizes the presented work, suggesting lines of continuation.

II. RELATED WORK

Our main goal is to train a vision-based deep object detector without relying on human labeling. Following our work line [26], we propose to leverage labeled images from virtual worlds and unlabeled ones from the real world, in such a way that we can automatically label the real-world images by progressively re-training a deep object detector that substitutes human annotators. Note that the labeling step is needed due to the domain shift between virtual and real-world images, otherwise, we could just use the virtual-world images to train object detectors and, afterwards, deploy them in the real world expecting a reliable performance. In particular, we explore the combination of the co-training idea for self-labeling objects and GAN-based image-to-image translation in the role of task-agnostic DA. On the other hand, our proposal is agnostic to the object detection architecture supporting self-labeling.

Accordingly, in the remaining of this section, we first review related works on self-labeling (subsection II-A), including self-training and co-training; next, we review related works on DA (subsection II-B). In both sections, we relate our proposal to the reviewed works.

A. SELF-LABELING

Self-labeling algorithms are examples of SSL *wrappers* [40], which work as meta-learners for supervised ML algorithms. The starting point consists of a labeled dataset, $\mathcal{X}^l$, and an unlabeled dataset, $\mathcal{X}^u$, where it is supposed that the cardinality of $\mathcal{X}^u$ is significantly larger than the cardinality of $\mathcal{X}^l$, and both datasets are drawn from the same domain, $\mathcal{D}$. The goal is to learn a predictive model, ϕ, whose accuracy would be relatively poor by only using $\mathcal{X}^l$ but becomes significantly higher by also leveraging $\mathcal{X}^u$. Briefly, self-labeling is an incremental process where ϕ is first trained with $\mathcal{X}^l$; then $\mathcal{X}^u$ is processed using ϕ in a way that the predictions are taken as a new pseudo-labeled dataset $\mathcal{X}^{\hat{l}}$, which in turn is used to retrain ϕ. The pseudo-labeling/retraining cycle is repeated until reaching some stop criterion, when ϕ is expected to be more accurate than at the beginning of the process. The main differences between self-labeling algorithms arise from how $\mathcal{X}^{\hat{l}}$ is formed and used to retrain ϕ in each cycle.

1) SELF-TRAINING

In self-training, introduced by Yarowsky [50] for word sense disambiguation, $\mathcal{X}^{\hat{l}}$ is formed by collecting the most confident predictions of ϕ in $\mathcal{X}^u$, updating $\mathcal{X}^u$ to contain only the remaining unlabeled data. Then, ϕ is retrained using supervision from $\mathcal{X}^l \cup \mathcal{X}^{\hat{l}}$, *i.e.* pseudo-labels are taken as ground truth. Before the DL era, Rosenberg *et al.* [33] already showed promising results when applying self-training to eye detection in face images. More recently, Jeong *et al.* [18], proposed an alternative to collect $\mathcal{X}^{\hat{l}}$ for deep object detection, which consists of adding a consistency loss for training ϕ as well as eliminating predominant backgrounds. If I_i^u is an unlabeled image, the consistence loss is based on the idea that $\phi(\mathrm{I}_i^u)$ and $\phi((\mathrm{I}_i^u)^{\Lsh})$, where $()^{\Lsh}$ stands for horizontal mirroring, must provide corresponding detections. Experiments are conducted on PASCAL VOC and MS-COCO datasets, and results are on pair with other state-of-the-art methods combining AL and self-training [42]. The reader is referred to [28] for a review on loss-based SSL methods for deep image classification. On the other hand, note that this SSL variations are not agnostic to ϕ, since its training loss is modified. This is also the case in [25], where, in the context of deep image classification, the activation functions composing ϕ must be replaced by Hermite polynomials. In this article, we use a self-training strategy as SSL baseline for on-board deep object detection, thus, keeping agnosticism regarding ϕ.

2) CO-TRAINING

Co-training was introduced by Blum and Mitchell [5] in the context of web-page classification, as alternative to the self-training of Yarowsky [50], in particular, to avoid model drift. In this case, two models, ϕ_1 and ϕ_2, are trained on different conditionally independent features of the data, called *views*, assuming that each view is sufficiently good to learn an accurate model. Each model is trained by following the same idea as with self-training, but in each cycle the data samples self-labeled by ϕ_1 and ϕ_2 are aggregated together. Soon, co-training was shown to outperform other state-of-the-art methods including self-training [27], and the conditional independence assumption was shown not to be essential in practice [4], [44], [45]. Later, in the context of sentence segmentation, Guz *et al.* [15] introduced the *disagreement* idea for co-training, which was refined by Tur [39] to jointly tackle DA in the context of natural language processing. In this case, samples self-labeled with high confidence by ϕ_i but with low confidence by ϕ_j, $i, j \in \{1, 2\}, i \neq j$, are considered as part of the new pseudo-labeled data in each cycle. In fact, disagreement-based SSL became a subject of study on its own at that time [52].

Before the irruption of DL, Levin *et al.* [22] applied co-training to detect vehicles in video-surveillance images, so removing background and using different training data to generate different views. More recently, Qiao *et al.* [29] used a co-training setting for deep image classification, based on several views. Each view corresponds to a different CNN, ϕ_i, trained by including samples generated to be mutually adversarial. The idea is to use different training data for each ϕ_i to prevent them to prematurely collapse in the same network. This implies to link the training of the ϕ_i's at the level of the loss function. In this article, we force the use of different training data for each ϕ_i, without linking their training at the level of loss functions, again, keeping co-training agnostic to the used ϕ_i. Moreover, we address objected detection, which involves not only predicting class instances as in image classification, but also localizing them within the images, so that the background becomes a large source of potential false positives.

Finally, to avoid confusion, it is worth to mention the so-called *co-teaching*, recently introduced by Han *et al.* [16], and its variant *co-teaching+* introduced by Yu *et al.* [51]. These algorithms are designed to address situations with noisy labels on $\mathcal{X}^l$, both demonstrated on deep image classification. Indeed, these algorithms stem ideas from co-training (the classical one from [16], the disagreement-based one from [51]), however, reproducing the words of Han *et al.* [16], *co-training is designed for SSL, and co-teaching is for learning with noisy (ground truth) labels (LNL); as LNL is not a special case of SSL, we cannot simply translate co-training from one problem setting to another problem setting*.

B. DOMAIN ADAPTATION

ML algorithms assume that training and testing data are drawn from the same domain, $\mathcal{D}$. When this is not the case, the trained models suffer from *domain shift*. In other words, we have data, $\mathcal{X}_S$, drawn from a *source domain*, $\mathcal{D}_S$, as well as data, $\mathcal{X}_T$, drawn from a *target domain*, $\mathcal{D}_T$, $\mathcal{D}_T \neq \mathcal{D}_S$. We can assume that $\mathcal{X}_S = \mathcal{X}_S^{tr} \cup \mathcal{X}_S^{tt}$, $\mathcal{X}_S^{tr} \cap \mathcal{X}_S^{tt} = \emptyset$, where $\mathcal{X}_S^{tr}$ is used to train some predictive model ϕ_S. It turns out that the prediction accuracy of ϕ_S in $\mathcal{X}_S^{tt}$ is much higher than in $\mathcal{X}_T$, a phenomenon known as domain shift. Addressing this problem is the goal of DA techniques, under the assumption that there is some (unknown) correlation between $\mathcal{D}_S$ and $\mathcal{D}_T$, since DA is not possible otherwise.

The core idea is to use $\mathcal{X}_T$ to obtain a new model, ϕ_T, being clearly more accurate than ϕ_S in $\mathcal{D}_T$. While doing this, the human-based labeling effort in $\mathcal{X}_T$ must be minimized. *Supervised DA* (SDA) assumes access to a relatively small set of labeled target-domain data $\mathcal{X}_T^{l,tr} \subset \mathcal{X}_T$. If we do not have access to $\mathcal{X}_S^{l,tr}$, then the challenge is to leverage from ϕ_S and $\mathcal{X}_T^{l,tr}$ to obtain ϕ_T. Otherwise, we can combine $\mathcal{X}_S^{l,tr}$ and $\mathcal{X}_T^{l,tr}$ to train ϕ_T. In *unsupervised DA* (UDA), $\mathcal{X}_T$ is unlabeled; thus, we address the more challenging situations of using $\mathcal{X}_T$ with either ϕ_S or $\mathcal{X}_S^{l,tr}$ to train ϕ_T. For a review of the DA corpus we advise the reader to consult [10], [43], [47].

In this article, we assume an UDA setting. Moreover, our source data comes from a virtual world with automatically generated labels, so we have $\mathcal{X}_S^{l,tr}$. Since we aim at assessing self-labeling by co-training to address the UDA problem, we have $\mathcal{X}^l = \mathcal{X}_S^{l,tr}$, $\mathcal{X}^u = \mathcal{X}_T$, thus, $\mathcal{X}^{\hat{l}} \subseteq \mathcal{X}_T$.

Following this line of work, Kim *et al.* [20] addressed USD for deep object detection by proposing a combination of a weak self-training and a special treatment of backgrounds via a loss component used during the training of the object detector, with PASCAL VOC as $\mathcal{X}_S$ and art-style datasets (Clipart1K, Watercolor2K, Comic2K) as $\mathcal{X}_T$. Zou *et al.* [56] also use a self-training strategy for UDA in the context of deep image classification and semantic segmentation (including a virtual-to-real setting), where the core idea is to perform confidence and model regularization of the trained classifiers. Since in our experimental setting we will use Faster R-CNN to obtain ϕ, it is also worth to mention the work by Chen *et al.* [9], where an UDA method was specifically designed for Faster R-CNN and demonstrated on car detection under a virtual-to-real setting too. Since Faster R-CNN is a two-stage classifier, the proposed UDA involves an image-level adaptation for the region proposal stage, and an instance-level adaptation for the BB prediction stage. Finally, focusing on traditional ML and *Amazon reviews* datasets, Chen *et al.* [7] showed that co-training is a promising algorithm for UDA, providing better performance than self-training.

Accordingly, beyond self-training-style and model-specific strategies for UDA, in this article, we are interested in assessing co-training as a meta-learning UDA strategy. To the best of our knowledge, this is an under-explored and relevant setting. Moreover, even we are going to use Faster R-CNN because its outstanding accuracy, our proposal neither is specific for it, nor requires modifying its losses. In other words, our co-training-based UDA works at the training-data level. This allows to complement it with other UDA working at the same level. In particular, we combine it with GAN-based image-to-image translation, since such task-agnostic approach can transform $\mathcal{X}_S^{l,tr}$ to be more similar to $\mathcal{X}_T$ before starting co-training. By using the CycleGAN implementation of Zhu *et al.* [53], we will see how, indeed, GAN-based image-to-image translation combined with co-training outperforms the use of each method in isolation.

III. METHODS

In this section, we detail our self-training and co-training meta-learning proposals as Algorithms 1 and 2, respectively. In addition, Figures 1 and 2 provide a visual representation of these algorithms, highlighting their main components with the corresponding data flow. Our main interest is to assess the performance of co-training in vision-based object detection, but we need also to develop a strong self-training baseline.

Algorithm 1 Self-Labeling by Self-Training

```
input  : Labeled images: $\mathcal{X}^l = \{< \mathrm{I}_i^l, \mathcal{Y}_i^l >\}$
         Unlabeled images: $\mathcal{X}^u = \{\mathrm{I}_i^u\}$
         Object detection architecture: $\Phi$
         $\Phi$ Training hyper-par.: $\mathcal{H}_\Phi$
         Slf-tr. hyp.-p.: $\mathcal{H}_{sl} = \{T, N, n, \mathcal{H}_{stp}[, \mathcal{H}_{seq}]\}$
output: New labeled images: $\mathcal{X}^{\hat{l}} = \{< \mathrm{I}_i^{\hat{l}}, \mathcal{Y}_i^{\hat{l}} >\}$
$< \mathcal{X}^{\hat{l}}, k > \leftarrow < \emptyset, 0 >$;
$\phi \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l, \mathcal{X}^{\hat{l}}$);
$\mathcal{X}^{\hat{l}}_{new} \leftarrow$ RunDetector($\phi, \mathcal{X}^u, T$);
repeat
    $\mathcal{X}^{\hat{l}}_{old} \leftarrow \mathcal{X}^{\hat{l}}_{new}$;
    $\mathcal{X}^{\hat{l}}_{\uparrow} \leftarrow$ Select($\uparrow, n,$ Rand($\mathcal{X}^{\hat{l}}_{new}, N[, \mathcal{H}_{seq}, k]$));
    $\mathcal{X}^{\hat{l}} \leftarrow$ Fuse($\mathcal{X}^{\hat{l}}, \mathcal{X}^{\hat{l}}_{\uparrow}$);
    $\phi \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l, \mathcal{X}^{\hat{l}}$);
    $\mathcal{X}^{\hat{l}}_{new} \leftarrow$ RunDetector($\phi, \mathcal{X}^u, T$);
until Stop?($\mathcal{H}_{stp}, \mathcal{X}^{\hat{l}}_{old}, \mathcal{X}^{\hat{l}}_{new}, k$++);
$\mathcal{X}^{\hat{l}} \leftarrow \mathcal{X}^{\hat{l}}_{new}$;
return $\mathcal{X}^{\hat{l}}$;
```

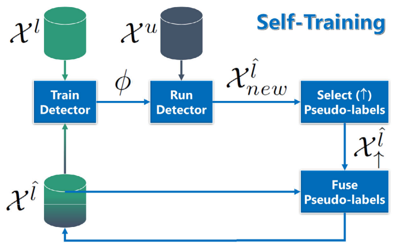

FIGURE 1. Self-training main components. We use the same notation as in Algorithm 1. The data in green is labeled, the one in dark grey is unlabeled, the transition of both colors represents pseudo-labeled data. The blue boxes correspond to the main components involved in self-training according to Algorithm 1.

Since they share functional components, we first introduce those and then detail how they are used for both self-training and co-training. Finally, we will see how, depending on the input data, these SSL algorithms can be used even in a context where there is a domain shift between already existing labeled images and the images to be labeled automatically. To refer to both, self-training and co-training indistinctly, we will use the term *self-labeling* in the rest of this section.

A. SELF-LABELING FUNCTIONAL COMPONENTS

1) INPUT AND OUTPUT PARAMETERS

Since we follow a SSL setting [40], we assume access to a set of images (*e.g.* acquired on-board a car) with each object of interest (*e.g.* vehicles and pedestrians) labeled by a BB and a class label, as well as to a set of unlabeled images. The former is denoted as $\mathcal{X}^l = \{< \mathrm{I}_i^l, \mathcal{Y}_i^l >\}$, where I_i^l is a particular image of the labeled set, being $\mathcal{Y}_i^l$ its corresponding labeling information; *i.e.*, for each object of interest in I_i^l, $\mathcal{Y}_i^l$ includes its BB and its class label. The unlabeled set is $\mathcal{X}^u = \{\mathrm{I}_i^u\}$, where I_i^u is a particular unlabeled image. We assume also a given object detection architecture to perform self-labeling, denoted as Φ, with corresponding training hyper-parameters denoted as $\mathcal{H}_\Phi$. After self-labeling,

Algorithm 2 Self-Labeling by Co-Training

```
input  : Labeled images: $\mathcal{X}^l = \{< \mathrm{I}_i^l, \mathcal{Y}_i^l >\}$
         Unlabeled images: $\mathcal{X}^u = \{\mathrm{I}_i^u\}$
         Object detection architecture: $\Phi$
         $\Phi$ Training hyper-par.: $\mathcal{H}_\Phi$
         Co-tr. hyp.-p.: $\mathcal{H}_{sl} = \{T, N, n, m, \mathcal{H}_{stp}[, \mathcal{H}_{seq}]\}$
output: New labeled images: $\mathcal{X}^{\hat{l}} = \{< \mathrm{I}_i^{\hat{l}}, \mathcal{Y}_i^{\hat{l}} >\}$
$< \mathcal{X}^{\hat{l}}_1, \mathcal{X}^{\hat{l}}_2, k > \leftarrow < \emptyset, \emptyset, 0 >$;
$< \mathcal{X}^l_1, \mathcal{X}^l_2 > \leftarrow < \mathcal{X}^l, (\mathcal{X}^l)^{\dagger} >$;
$\phi_1 \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_1, \mathcal{X}^{\hat{l}}_1$);
$\phi_2 \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_2, \mathcal{X}^{\hat{l}}_2$);
$\mathcal{X}^{\hat{l}}_{1,new} \leftarrow$ RunDetector($\phi_1, \mathcal{X}^u, T$);
$\mathcal{X}^{\hat{l}}_{2,new} \leftarrow$ RunDetector($\phi_2, \mathcal{X}^u, T$);
repeat
    $\mathcal{X}^{\hat{l}}_{old} \leftarrow \mathcal{X}^{\hat{l}}_{1,new}$;
    $\mathcal{X}^{\hat{l}}_{1,\uparrow} \leftarrow$ Select($\uparrow, m,$ Rand($\mathcal{X}^{\hat{l}}_{1,new}, N[, \mathcal{H}_{seq}, k]$));
    $\mathcal{X}^{\hat{l}}_{2,\uparrow} \leftarrow$ Select($\uparrow, m,$ Rand($\mathcal{X}^{\hat{l}}_{2,new}, N[, \mathcal{H}_{seq}, k]$));
    $\mathcal{X}^{\hat{l}}_{1,\downarrow} \leftarrow$ Select($\downarrow, n,$ RunDetector($\phi_1, \mathcal{X}^{\hat{l}}_{2,\uparrow}, T$));
    $\mathcal{X}^{\hat{l}}_{2,\downarrow} \leftarrow$ Select($\downarrow, n,$ RunDetector($\phi_2, \mathcal{X}^{\hat{l}}_{1,\uparrow}, T$));
    $\mathcal{X}^{\hat{l}}_1 \leftarrow$ Fuse($\mathcal{X}^{\hat{l}}_1, \mathcal{X}^{\hat{l}}_{1,\downarrow}$);
    $\mathcal{X}^{\hat{l}}_2 \leftarrow$ Fuse($\mathcal{X}^{\hat{l}}_2, \mathcal{X}^{\hat{l}}_{2,\downarrow}$);
    $\phi_1 \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_1, \mathcal{X}^{\hat{l}}_1$);
    $\phi_2 \leftarrow$ TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_2, \mathcal{X}^{\hat{l}}_2$);
    $\mathcal{X}^{\hat{l}}_{1,new} \leftarrow$ RunDetector($\phi_1, \mathcal{X}^u, T$);
    $\mathcal{X}^{\hat{l}}_{2,new} \leftarrow$ RunDetector($\phi_2, \mathcal{X}^u, T$);
until Stop?($\mathcal{H}_{stp}, \mathcal{X}^{\hat{l}}_{old}, \mathcal{X}^{\hat{l}}_{1,new}, k$++);
$\mathcal{X}^{\hat{l}} \leftarrow \mathcal{X}^{\hat{l}}_{1,new}$;
return $\mathcal{X}^{\hat{l}}$;
```

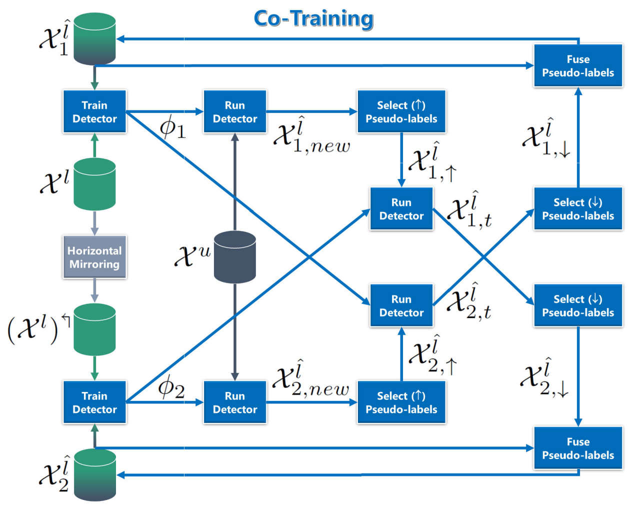
Co-Training
$\mathcal{X}^{\hat{l}}_1$
Fuse Pseudo-labels
Train Detector
ϕ_1
Run Detector
$\mathcal{X}^{\hat{l}}_{1,new}$
Select (↑) Pseudo-labels
$\mathcal{X}^{\hat{l}}_{1,\downarrow}$
$\mathcal{X}^{\hat{l}}_{1,\uparrow}$
$\mathcal{X}^l$
Run Detector
$\mathcal{X}^{\hat{l}}_{1,t}$
Select (↓) Pseudo-labels
Horizontal Mirroring
$\mathcal{X}^u$
Run Detector
$\mathcal{X}^{\hat{l}}_{2,t}$
Select (↓) Pseudo-labels
$(\mathcal{X}^l)^{\dagger}$
$\mathcal{X}^{\hat{l}}_{2,\uparrow}$
$\mathcal{X}^{\hat{l}}_{2,\downarrow}$
Train Detector
ϕ_2
Run Detector
$\mathcal{X}^{\hat{l}}_{2,new}$
Select (↑) Pseudo-labels
Fuse Pseudo-labels
$\mathcal{X}^{\hat{l}}_2$

FIGURE 2. Co-training main components. We use the same notation as in Algorithm 2, but we have introduced two dummy variables for the sake of clarity ($\mathcal{X}^{\hat{l}}_{1,t}, \mathcal{X}^{\hat{l}}_{2,t}$). The data in green is labeled, the one in dark grey is unlabeled, the transition of both colors represents pseudo-labeled data. The blue boxes correspond to the main components involved in co-training according to Algorithm 2. The light grey bounding box is executed just once.

we expect to obtain a new set of automatically labeled images, which we denote as $\mathcal{X}^{\hat{l}} = \{< \mathrm{I}_i^{\hat{l}}, \mathcal{Y}_i^{\hat{l}} >\}$, where $\mathrm{I}_i^{\hat{l}} \in \mathcal{X}^u$ and has $\mathcal{Y}_i^{\hat{l}}$ as so-called pseudo-labels, which in this case consists of a BB and a class label for each detected object in $\mathrm{I}_i^{\hat{l}}$. The variables introduced so far are generic in SSL, *i.e.* they are not specific of our proposals. We denote as $\mathcal{H}_{sl}$ the specific set of hyper-parameters we require for self-labeling. In fact, $\mathcal{H}_{sl}$ includes $\{T, N, n, \mathcal{H}_{stp}, \mathcal{H}_{seq}\}$ as common parameters for both self-training and co-training, but the latter requires an additional parameter m that we will explain in the context of co-training. If K is the number of classes to be considered, then $T = \{t_1, \ldots, t_K\}$ is a set of per-class detection thresholds, normally used by object detectors to ensure a minimum per-class confidence to accept potential detections. Since self-labeling must perform object detection on unlabeled images, these thresholds are needed. During a self-labeling cycle, N self-labeled images are randomly selected, from which n are kept to retrain the object detector. Self-training and co-training use different criteria to select such n self-labeled images. $\mathcal{H}_{stp}$ consists of the parameters required to evaluate whether self-labeling should stop or not. Finally, $\mathcal{H}_{seq}$ is an optional set of parameters, only required if $\mathcal{X}^u$ consists of sequences rather than isolated images. Note that, in this case, we can easily avoid to introduce too similar training samples coming from the same object instances in consecutive frames, which has shown to be useful in AL approaches [2].

2) TrainDetector($\Phi, \mathcal{H}_\Phi, \mathcal{X}^l, \mathcal{X}^{\hat{l}}$) : ϕ

This function returns an object detector, ϕ, by training a CNN architecture Φ (*e.g.* Faster R-CNN [31]) according to the training hyper-parameters $\mathcal{H}_\Phi$ (*e.g.* optimizer, learning rate, mini-batch size, training iterations, etc.). Note that this is just the standard manner of training ϕ, but as part of our self-labeling procedures, we control the provided training data. In particular, we use labels (in $\mathcal{X}^l$) and pseudo-labels (in $\mathcal{X}^{\hat{l}}$) indistinctly. However, we only consider background samples based on $\mathcal{X}^l$, since, during self-labeling, $\mathcal{X}^{\hat{l}}$ can contain false negatives which could be erroneously taken as hard negatives when training ϕ. Despite this control over the training data, note that we neither require custom modifications of the loss function used to train ϕ, nor architectural modifications of Φ.

3) RunDetector($\phi, \mathcal{X}^u, T$) : $\mathcal{X}^{\hat{l}}$

This function returns a set of self-labeled images, $\mathcal{X}^{\hat{l}}$, obtained by running the object detector ϕ on the unlabeled set of images $\mathcal{X}^u$; in other words, the pseudo-labels correspond to the object detections. Each detection D consists of a BB B and a class label c; moreover, being a detection implies that the confidence $\phi(D)$ fulfils the condition $\phi(D) \geq t_c, t_c \in T$.

4) Rand($\mathcal{X}^{\hat{l}}, N[, \mathcal{H}_{seq}, k]$) : $\mathcal{X}_s^{\hat{l}}$

This function returns a set $\mathcal{X}_s^{\hat{l}}$ of N self-labeled images randomly chosen from $\mathcal{X}^{\hat{l}}$. $\mathcal{H}_{seq}$ and k are optional parameters. If they are provided, it means that $\mathcal{X}^{\hat{l}}$ consists of sequences and we want to ensure not to return self-labeled consecutive frames, since they may contain very similar samples coming from the same object instances and we want to favor variability in the samples. Moreover, in this way we can minimize the inclusion of spurious false positives that may persist for several frames. In this case, $\mathcal{H}_{seq} = \{\Delta t_1, \Delta t_2\}$; where Δt_1 is the minimum distance between frames with pseudo-labels generated in the current cycle, k, and Δt_2 is the minimum distance between frames with pseudo-labels generated in cycle k and frames with pseudo-labels generated in previous cycles ($< k$). Δt_1 condition is applied first, then Δt_2 one. The final random selection is performed on the frames remaining after applying these distance conditions.

5) Select($s, n, \mathcal{X}^{\hat{l}}$) : $\mathcal{X}_s^{\hat{l}}$

This function returns a sub-set $\mathcal{X}_s^{\hat{l}} \subset \mathcal{X}^{\hat{l}}$ of m self-labeled images, selected according to a criterion s. If $s = \uparrow$, the top n most confident images are selected; while for $s = \downarrow$, the n least confident images are selected. We resume the confidence of an image from the confidences of its detections. For the sake of simplicity, since $\mathcal{X}^{\hat{l}} = \{< \mathrm{I}_i^{\hat{l}}, \mathcal{Y}_i^{\hat{l}} >\}$, we can assume that $\mathcal{Y}_i^{\hat{l}}$ not only contains the BB and class label for each detected object in $\mathrm{I}_i^{\hat{l}}$, but also the corresponding detection confidence. Then, the confidence assigned to $\mathrm{I}_i^{\hat{l}}$ is the average of the confidences in $\mathcal{Y}_i^{\hat{l}}$.

6) Fuse($\mathcal{X}_{old}^{\hat{l}}, \mathcal{X}_{new}^{\hat{l}}$) : $\mathcal{X}^{\hat{l}}$

The goal of this function is to avoid redundant detections between the self-labeled sets $\mathcal{X}_{old}^{\hat{l}}$ and $\mathcal{X}_{new}^{\hat{l}}$ and preserving different ones, producing a new set of self-labeled images, $\mathcal{X}^{\hat{l}}$. Thus, on the one hand, $\mathcal{X}^{\hat{l}}$ will contain all the self-labeled images in $\mathcal{X}_{old}^{\hat{l}} \cup \mathcal{X}_{new}^{\hat{l}} - \mathcal{X}_{old}^{\hat{l}} \cap \mathcal{X}_{new}^{\hat{l}}$; on the other hand, for those images in $\mathcal{X}_{old}^{\hat{l}} \cap \mathcal{X}_{new}^{\hat{l}}$, only the detections in $\mathcal{X}_{new}^{\hat{l}}$ are kept, so the corresponding information is added to $\mathcal{X}^{\hat{l}}$.

7) Stop?($\mathcal{H}_{stp}, \mathcal{X}_{old}^{\hat{l}}, \mathcal{X}_{new}^{\hat{l}}, k$) : Boolean

This function decides whether to stop self-labeling or not. The decision relies on two conditions. The first one is if a minimum number of self-labeling cycles, K_{min}, have been performed, where current number is k. The second condition relies on the similarity of $\mathcal{X}_{old}^{\hat{l}}$ and $\mathcal{X}_{new}^{\hat{l}}$, computed as the mAP (mean average precision) [11] by using $\mathcal{X}_{old}^{\hat{l}}$ in the role of ground truth and $\mathcal{X}_{new}^{\hat{l}}$ in the role of results to be evaluated. We compute the absolute value difference between the mAP at current cycle and previous one, keeping track of these differences in a sliding window fashion. If these differences are below a threshold, $T_{\Delta_{mAP}}$, for more than a given number of consecutive cycles, ΔK, self-labeling will stop. Therefore, $\mathcal{H}_{stp} = \{K_{min}, T_{\Delta_{mAP}}, \Delta K\}$.

B. SELF-TRAINING

Algorithm 1 describes self-training based on the functional components introduced in Section III-A. In the following, we highlight several points of this algorithm.

As Select(, ,) is run in the loop, it implies that in each cycle the n most confident self-labeled images are kept for retraining ϕ. Moreover, potentially redundant object detections arising from different cycles are resumed by running the Fuse(,) function. As is called Fuse(,) in the loop, when a previous detection and a new one overlap enough, the new one is kept since it is based on the last trained version of ϕ, which is expected to be more accurate than previous ones. When calling Select(, ,), not all the available self-labeled images are considered, but just a random set of them of size N; which are chosen taking into account the selection conditions introduced in case of working with sequences. This prevents the same highly accurate detections to be systematically selected across cycles, which would prevent the injection of variability when retraining ϕ. The random selection over the entire $\mathcal{X}^u$ was also proposed in the original co-training algorithm [5]. As we have mentioned before, when running TrainDetector(, , ,), background samples are only collected from $\mathcal{X}^l$ to avoid introducing false positives. Moreover, we set $\mathcal{H}_\Phi$ to ensure that all the training samples are visited at least once (mini-batch size $\times$ number of iterations $\geq$ number of training images). Therefore, we can think of $\mathcal{X}^l \cup \mathcal{X}^{\hat{l}}$ as a mixing of data, where $\mathcal{X}^l$ acts as a regularizing factor during training, which aims to prevent ϕ to become an irrecoverably bad detector. Thinking in a virtual-to-real UDA setting, we note that from traditional ML algorithms to modern deep CNNs, mixing virtual and real data has been shown to be systematically beneficial across different computer vision tasks [17], [32], [36], [41]. Finally, we can see that to run the stopping criterion we rely on the full self-labeled data available at the beginning and end of each cycle, which results from applying the last available version of ϕ to the full $\mathcal{X}^u$ set.

C. CO-TRAINING

Algorithm 2 describes our co-training proposal based on the functional components introduced in Section III-A. In the following, we highlight several points of this algorithm.

Our co-training strategy tries to make ϕ_1 and ϕ_2 different by training them on different data. Note how each ϕ_i is trained on $\mathcal{X}^l_i$ and $\mathcal{X}^{\hat{l}}_i$, $i \in \{1, 2\}$, at each cycle. The $\mathcal{X}^l_i$'s do not change across cycles and we have $\mathcal{X}^l_1 = \mathcal{X}^l$ and $\mathcal{X}^l_2 = (\mathcal{X}^l)^{\dashv}$, thus, one of the labeled sets is a horizontal mirroring of the other. Moreover, the $\mathcal{X}^{\hat{l}}_i$'s are expected to be different from each other and changing from cycle to cycle. On the other hand, even technically not using the same exact data to train ϕ_1 and ϕ_2, there will be a cycle when they converge and are not able to improve each other. This is what we check to stop, *i.e.* as for self-training, but focusing on the performance of ϕ_1 since it is the detector using $\mathcal{X}^l$ (the original labeled dataset).

Another essential question is how each $\mathcal{X}^{\hat{l}}_i$ is obtained. As we have mentioned before, we follow the idea of disagreement. Thus, from the random set of images self-labeled by ϕ_i, *i.e.* $\mathcal{X}^{\hat{l}}_{i,new}$, $i \in \{1, 2\}$, we set to $\mathcal{X}^{\hat{l}}_{i,\uparrow}$ the m with overall higher confident pseudo-labels; then, the images in $\mathcal{X}^{\hat{l}}_{i,\uparrow}$ are processed by ϕ_j, $j \in \{1, 2\}$, $i \neq j$, and the n with the overall lower confident pseudo-labels are set to $\mathcal{X}^{\hat{l}}_{j,\downarrow}$. Finally, each $\mathcal{X}^{\hat{l}}_{i,\downarrow}$ is fused with the $\mathcal{X}^{\hat{l}}_i$ accumulated from previous cycles to update $\mathcal{X}^{\hat{l}}_i$. Therefore, in each cycle, each ϕ_i is retrained with the images containing the less confident pseudo-labels for current ϕ_i, selected among the images containing the most confident pseudo-labels for current ϕ_j. In this way, the detectors trust each other, and use the samples that are more difficult for them for improving.

Note also that the most costly processing in self-training and co-training cycles is the TrainDetector(, , ,) procedure. It is called once per cycle for self-training, and twice for co-training. However, in the latter case the two executions can run totally in parallel, therefore, with the proper hardware resources, self-training and co-training cycles can be performed in a very similar time.

D. SELF-LABELING FOR UDA

Once introduced our self-training and co-training algorithms, we see how using them for UDA is just a matter of providing the proper input parameters; *i.e.*, $\mathcal{X}^l = \mathcal{X}^{l,tr}_S$, $\mathcal{X}^u = \mathcal{X}_T$, being $\mathcal{X}^{l,tr}_S$ the source-domain labeled dataset, and $\mathcal{X}_T$ the target-domain unlabeled one. We will draw the former from a virtual world, and the latter from the real world.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

As real-world datasets we use KITTI [11] and Waymo [37], in the following denoted as $\mathcal{K}$, $\mathcal{W}$, respectively. The former is a well-established standard born in the academia; the latter is a new contribution from the industry. To work with $\mathcal{K}$, we follow the splits introduced in [48], which are based on isolated images, and guarantee that training and testing images are not highly correlated; *i.e.*, training and testing images are not just different frames sampled from the same sequences. On the other hand, $\mathcal{W}$ dataset contains video sequences. Following the advice for its use, we have randomly selected part of the sequences for training and the rest for testing. Taking $\mathcal{K}$ as reference, we focus on daytime and non-adverse weather conditions in all the cases. In addition, we have accommodated $\mathcal{W}$ images to the resolution of $\mathcal{K}$ ones, *i.e.* 1240×375 pixels, by just cropping away their upper part (which mainly shows sky) and keeping the vertically centered 1240 columns. Following KITTI benchmark moderate settings, we set the minimum BB height to detect objects to 25 pixels and, analogously, 50 pixels for $\mathcal{W}$. Moreover, since $\mathcal{W}$ contains 3D BBs, as for the virtual-world objects (Figure 3), we obtain 2D BBs from them. In other words, the resulting 2D BBs account for occluded object areas, which is not the case for the 2D BBs directly available with $\mathcal{W}$.

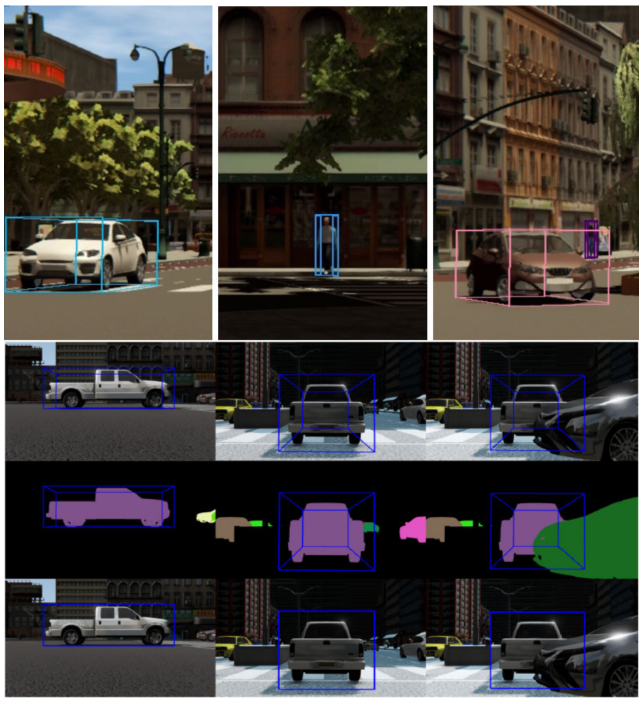

FIGURE 3. Top images: virtual-world patches showing 3D BBs framing vehicles and pedestrians. Bottom image: projecting 3D BBs as 2D BBs for different views of a pickup, with instance segmentation as visual reference.

In order to perform our experiments, as set of virtual-world images ($\mathcal{V}$), we have used a dataset that we generated internally around two years ago as a complement for our former SYNTHIA dataset [32]. We did not have the opportunity to release it at that moment, but it will be publicly available to complement this article. We generated this data following KITTI parameters: same image resolution, daytime and non-adverse weather, and isolated images. As in [32], we did not focus on obtaining photo-realistic images. However, for generating $\mathcal{V}$, we included standard visual post-effects such as anti-aliasing, ambient occlusion, depth of field, eye adaptation, blooming and chromatic aberration. Later, we will see how the domain shift between $\mathcal{V}$ and $\mathcal{K}/\mathcal{W}$, is similar to the one between $\mathcal{K}$ and $\mathcal{W}$; thus, being $\mathcal{V}$ a proper virtual-world dataset for our study. Figure 3 shows virtual-world images with 3D BBs framing vehicles and pedestrians, illustrating also how the 3D BBs are projected as 2D BBs covering occluded object areas.

For each dataset, Table 1 summarizes the number of images/frames and objects (vehicles and pedestrians) used for training and testing our object detectors.

Since our proposal does not assume a particular object detection architecture, for performing the experiments we have chosen Faster R-CNN [31] because it provides a competitive detection accuracy and is very well known by computer vision practitioners [23]. For the sake of completeness, Figure 4 shows the main components of Faster R-CNN, namely, the Feature Extractor (FE), the region proposal network (RPN), and the region-based CNN (RCNN). We use the implementation available in the *Detectron* framework [13], with VGG16 as FE. The RPN component generates object candidates from the so-called bottleneck of FE (*i.e.* ConvBlock5). Conceptually, these candidates can be understood as BBs which can potentially contain objects of interest. Using these candidates and the same bottleneck, the RCNN component performs the final object classification and BB regression (a refinement of the BBs proposed by RPN). For comprehensive details the reader can refer to [13]. At training time, we always initialize VGG16 (FE) with ImageNet weights since it is a recommended best practice. The weights of the RPN and RCNN components are randomly initialized. We run each Faster R-CNN training for 40,000 iterations using SGD optimizer. The learning rate starts at 0.001 and we have used a decay of 0.1 at iterations 30,000 and 35,000. Each iteration uses a mini batch of two images. In terms of Algorithms 1 and 2, these settings correspond to $\mathcal{H}_\Phi$.

As GAN-based image-to-image translation method we use the CycleGAN implementation in [53]. A GAN training runs for 40 epochs using a weight of 1.0 for the identity mapping loss. In order to be able to use full resolution images, the training of the GAN has been done patch-wise, with patch sizes of 300 × 300 pixels. The rest of parameters have been set

TABLE 1. Datasets ($\mathcal{X}$): train ($\mathcal{X}^{tr}$) and test ($\mathcal{X}^{tt}$) information, $\mathcal{X} = \mathcal{X}^{tr} \cup \mathcal{X}^{tt}$, $\mathcal{X}^{tr} \cap \mathcal{X}^{tt} = \emptyset$. We show the number of images/frames (sequences), vehicle BBs, pedestrian BBs, and whether the datasets consists of video sequences or not.

	$\mathcal{X}^{tr}$			$\mathcal{X}^{tt}$			
Dataset ($\mathcal{X}$)	Images (seq.)	Vehicles	Pedestrians	Images (seq.)	Vehicles	Pedestrians	Seq.?
VIRTUAL ($\mathcal{V}$)	19,791	43,326	44,863				No
KITTI ($\mathcal{K}$)	3,682	14,941	3,154	3,799	18,194	1,333	No
WAYMO ($\mathcal{W}$)	9,873 (50)	64,446	9,918	4,161 (21)	24,600	3,068	Yes

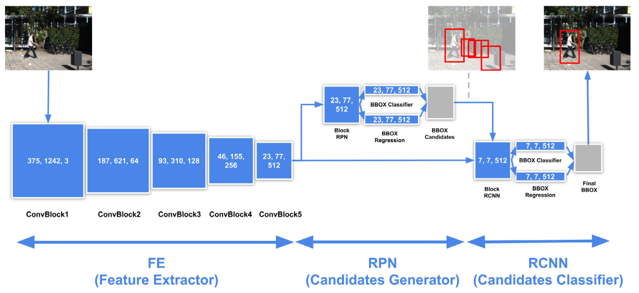

FIGURE 4. Main components of Faster R-CNN: feature extractor (FE), region proposal network (RPN), and region-based CNN (RCNN). Their responsibilities are outlined in parenthesis and elaborated in the main text. We use VGG16 as FE. Blue boxes are blocks of neural network layers with input dimensions indicated as <height, width, channels>. Grey boxes are algorithmic steps to return BBs, candidates (RPN) or detections (RCNN). This Faster R-CNN architecture is detailed in [13].

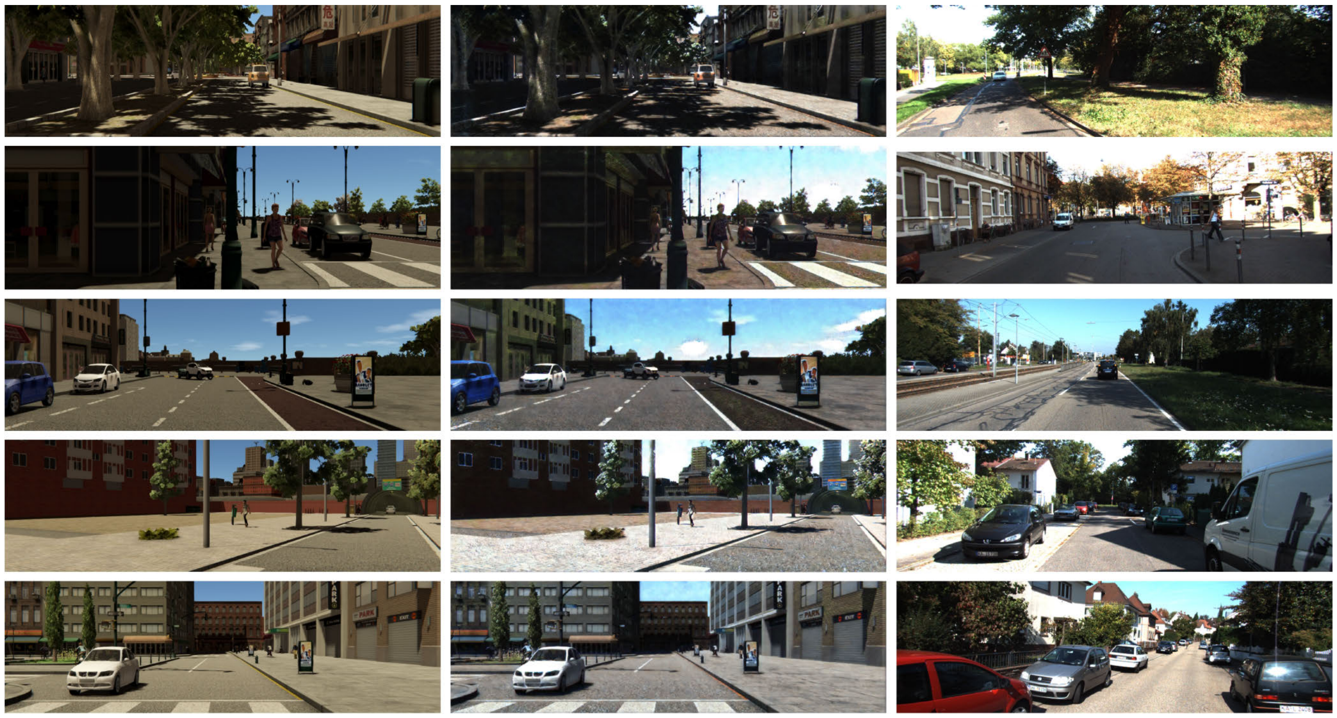

FIGURE 5. From left to right: images from $\mathcal{V}$, corresponding images in $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ (*i.e.* processed by the $\mathcal{V} \rightarrow \mathcal{K}$ GAN), and images from $\mathcal{K}$. Last column is just a visual reference since, obviously, there is no a one-to-one correspondence between $\mathcal{V}$ and $\mathcal{K}$.

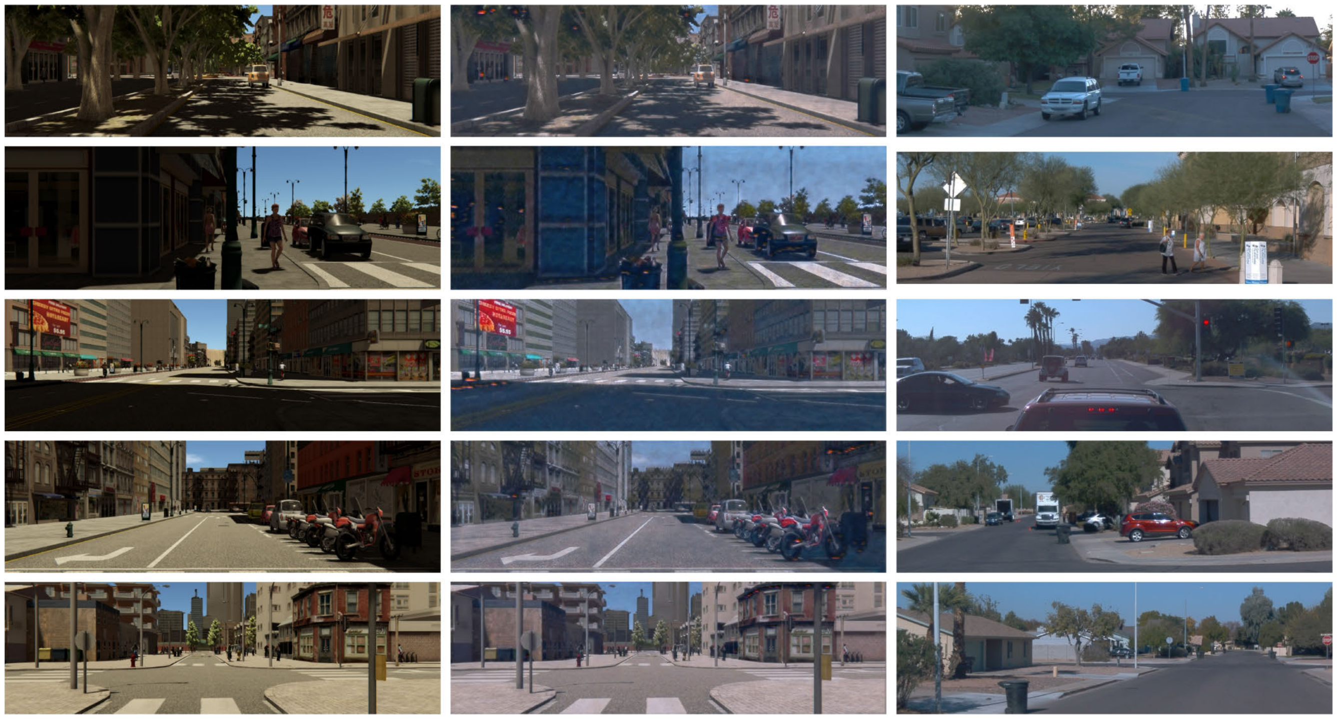

FIGURE 6. From left to right: images from $\mathcal{V}$, corresponding images in $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$ (*i.e.* processed by the $\mathcal{V} \rightarrow \mathcal{W}$ GAN), and images from $\mathcal{W}$. For visual comparison, the two top rows of this figure and those in Figure 5, start with the same images in $\mathcal{V}$.

as recommended in [53]. We train a $\mathcal{V} \rightarrow \mathcal{K}$ transforming GAN, $\mathcal{G}_{\mathcal{K}}$, and another for $\mathcal{V} \rightarrow \mathcal{W}$, $\mathcal{G}_{\mathcal{W}}$. Accordingly, $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}} = \mathcal{G}_{\mathcal{K}}(\mathcal{V})$ denotes the set of virtual-world images transformed by $\mathcal{G}_{\mathcal{K}}$ and, analogously, we have $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}} = \mathcal{G}_{\mathcal{W}}(\mathcal{V})$. We will use the notation $\mathcal{V}_{\mathcal{G}}$ to refer to any of these sets. The ground truth present in $\mathcal{V}$ is used as ground truth for $\mathcal{V}_{\mathcal{G}}$. For qualitative examples of image-to-image transformations, we refer to Figures 5 and 6, using $\mathcal{G}_{\mathcal{K}}$ and $\mathcal{G}_{\mathcal{W}}$, respectively.

Table 2 summarizes the rest of hyper-parameters used to perform our experiments. Note that, regarding our

TABLE 2. Self-training and co-training hyper-parameters as defined in Algorithms 1 and 2. We use the same values for both, as well as to work with KITTI ($\mathcal{K}$) and Waymo ($\mathcal{W}$) datasets, except for $\mathcal{H}_{seq}$ which only applies to $\mathcal{W}$. N, n, m, Δt_1, and Δt_2 are set in number-of-images units, K_{min} and ΔK in number-of-cycles, $T_{\Delta_{mAP}}$ runs in [0..100]. We use the same confidence detection threshold for vehicles and pedestrians, which runs in [0..1]. (*) Only used in co-training, however, for $m = \infty$, it has no effect.

				$\mathcal{H}_{stp}$			$\mathcal{H}_{seq}$	
T	N	n	$m^\star$	K_{min}	$T_{\Delta_{mAP}}$	ΔK	Δt_1	Δt_2
{0.8,0.8}	2,000	100	∞	20	2.0	5	5	10

TABLE 3. SSL results for $\mathcal{K}$ and $\mathcal{W}$. We assess vehicle (V) and pedestrian (P) detection, according to the mAP metric. From $\mathcal{X}^{tr} \in \{\mathcal{K}^{tr}, \mathcal{W}^{tr}\}$, we preserve the labeling information for a randomly chosen p% of its images, while it is ignored for the rest. We report results for $p = 100$ (all labels are used), $p = 5$ and $p = 10$. Slf-T and Co-T stand for self-training and co-training, resp., which refers to how images were self-labeled from the respective unlabeled training sets.

	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
Training set	V	P	V&P	V	P	V&P
100% Labeled (upper bound)	81.02	65.40	73.21	61.71	57.74	59.73
5% Labeled (lower bound)	59.81	36.49	48.15	51.69	41.92	46.81
5% Labeled + Slf-T	<u>68.94</u>	40.99	54.97	<u>**54.26**</u>	55.38	54.82
5% Labeled + Co-T	<u>**68.99**</u>	**55.07**	**62.03**	<u>54.00</u>	**56.34**	**55.17**
10% Labeled (lower bound)	72.13	49.03	60.58	49.53	49.83	49.68
10% Labeled + Slf-T	**76.32**	56.05	**66.19**	54.88	57.40	56.14
10% Labeled + Co-T	73.08	**58.53**	65.81	**56.15**	**60.20**	**58.18**

TABLE 4. Number of self-labeled vehicles and pedestrians applying self-training and co-training, for the SSL (5% & 10%) and UDA (Source & ASource) settings, for KITTI ($\mathcal{K}$) and Waymo ($\mathcal{W}$). In parenthesis we indicate the percentage of false positives. The top block corresponds to ground truth labels in the full training sets, and the percentages used for SSL. After removing false positives, in each block of rows, the corresponding Δ_X shows how many more objects are labeled by co-training compared to self-training.

	$\mathcal{K}$		$\mathcal{W}$	
Training set	V	P	V	P
100 % Labeled	14,941	3,154	64,446	9,918
5 % Labeled	647	83	3,520	472
10 % Labeled	1,590	367	6,521	959
5% Labeled + Slf-T	9,376 (0.8%)	837 (0.5%)	52,150 (1.7%)	5,554 (3.1%)
5% Labeled + Co-T	10,960 (1.8%)	1,591 (6.2%)	56,102 (4.0%)	6,948 (10.9%)
$\Delta_{5\%}$	+1,462	+660	+2,594	+809
10% Labeled + Slf-T	10,536 (0.7%)	1,409 (2.3%)	54,698 (1.6%)	6,519 (2.7%)
10% Labeled + Co-T	11,309 (1.7%)	1,552 (3.9%)	56,686 (3.1%)	7,384 (6.0%)
$\Delta_{10\%}$	+654	+115	+1,106	+598
Slf-T + Source	12,686 (4.6%)	1,378 (8.9%)	29,036 (23.8%)	837 (5.1%)
Co-T + Source	14,537 (6.5%)	1,964 (14.0%)	48,653 (30.4%)	3,481 (12.9%)
Δ_{Source}	+1,490	+434	+1,1737	+2,238
Slf-T + ASource	10,389 (1.7%)	1,268 (3.5%)	41,173 (24.7%)	1,928 (13.5%)
Co-T + ASource	12,864 (3.1%)	1,532 (5.2%)	53,955 (28.8%)	3,553 (16.0%)
$\Delta_{ASource}$	+2,253	+229	+7,413	+1,317

self-labeling approaches, we use the same values for $\mathcal{K}$ and $\mathcal{W}$, with the only exception that for $\mathcal{W}$ we also consider that it is composed of video sequences. Moreover, we also use the same values for hyper-parameters shared by self-training and co-training. We relied on the meaning of the hyper-parameters to set them with reasonable values. Then, during the experiments of the 5% with self/co-training (see Table 3) we did visual inspection of the self-labeled images to ensure the methods were working well. In this process, we noted that, since the confidence thresholds (T) were already avoiding too erroneous self-labeled images, for

TABLE 5. UDA results for $\mathcal{V} \rightarrow \{\mathcal{K}, \mathcal{W}\}$, *i.e.* virtual to real. ASource (*adapted source*) refers to $\mathcal{V}_\mathcal{G} \in \{\mathcal{V}_{\mathcal{G}_\mathcal{K}}, \mathcal{V}_{\mathcal{G}_\mathcal{W}}\}$. $\mathcal{X}^{l,tr}$ refers to the fully labeled target-domain training set. $\mathcal{X}^{\hat{l},tr}$ consists of the same images as $\mathcal{X}^{l,tr}$, but self-labeled by either self-taining (Slf-T) or co-training (Co-T). Just as reference, we also show the domain shift between $\mathcal{K}$ and $\mathcal{W}$. According to these results, as upper bound for $\mathcal{K}$ we take the detector based on $\mathcal{X}^{l,tr}$&$\mathcal{V}_\mathcal{G}$, while for $\mathcal{W}$ it is the detector based on $\mathcal{X}^{l,tr}$. We refer to the main text for more details.

	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
Training set	V	P	V&P	V	P	V&P
Source ($\mathcal{K}$)	-	-	-	37.79	49.80	43.80
Source ($\mathcal{W}$)	45.57	44.11	44.84	-	-	-
Source ($\mathcal{V}$) (lower bound)	62.27	61.28	64.28	38.88	53.37	46.13
ASource ($\mathcal{V}_\mathcal{G}$)	75.52	61.94	68.73	52.42	53.08	52.75
Target ($\mathcal{X}^{l,tr}$)	81.02	65.40	73.21	†61.71	†**57.74**	†**59.73**
Target + Source ($\mathcal{X}^{l,tr}$&$\mathcal{V}$)	81.68	**68.07**	74.88	60.65	54.23	57.44
Target + ASource ($\mathcal{X}^{l,tr}$&$\mathcal{V}_\mathcal{G}$)	†**84.08**	†66.00	†**75.04**	**62.02**	51.55	56.79
Slf-T + Source ($\mathcal{X}^{\hat{l},tr}$&$\mathcal{V}$)	70.25	67.59	68.92	<u>48.50</u>	44.63	46.57
Co-T + Source ($\mathcal{X}^{\hat{l},tr}$&$\mathcal{V}$)	**73.53**	**69.50**	**71.52**	<u>**48.56**</u>	**56.33**	**52.45**
Slf-T + ASource ($\mathcal{X}^{\hat{l},tr}$&$\mathcal{V}_\mathcal{G}$)	<u>79.62</u>	65.87	72.75	<u>59.19</u>	52.48	55.84
Co-T + ASource ($\mathcal{X}^{\hat{l},tr}$&$\mathcal{V}_\mathcal{G}$)	<u>**79.99**</u>	**69.01**	**74.50**	<u>**59.99**</u>	**55.39**	**57.69**
Δ(Co-T + ASource) vs Source	+17.72	+07.73	+10.22	+21.11	+02.02	+11.56
Δ(Co-T + ASource) vs ASource	+04.47	+07.07	+05.77	+07.57	+02.31	+04.94
Δ(Co-T + ASource) vs upp.-b.†	-04.09	+03.01	-00.54	-01.72	-02.35	-02.04
Δ(Co-T + Source) vs Source	+11.26	+08.22	+07.22	+09.68	+02.96	+06.32
ΔASource vs Source	+13.25	+00.66	+04.45	+13.54	-00.29	+06.62

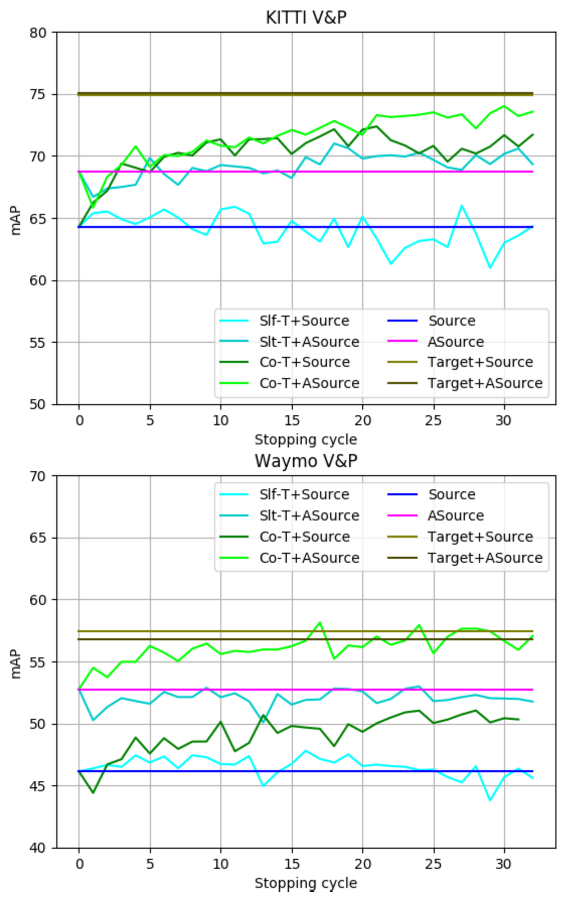

FIGURE 7. Eventual detection performance (mAP) of self-training and co-training as a function of the stopping cycle, in the UDA setting. Upper and lower bounds are included as visual reference. We refer to the main text for more details.

co-training it was better to send all the images self-labeled by the detector ϕ_i to detector ϕ_j, $i, j \in \{1, 2\}$, $i \neq j$, so that

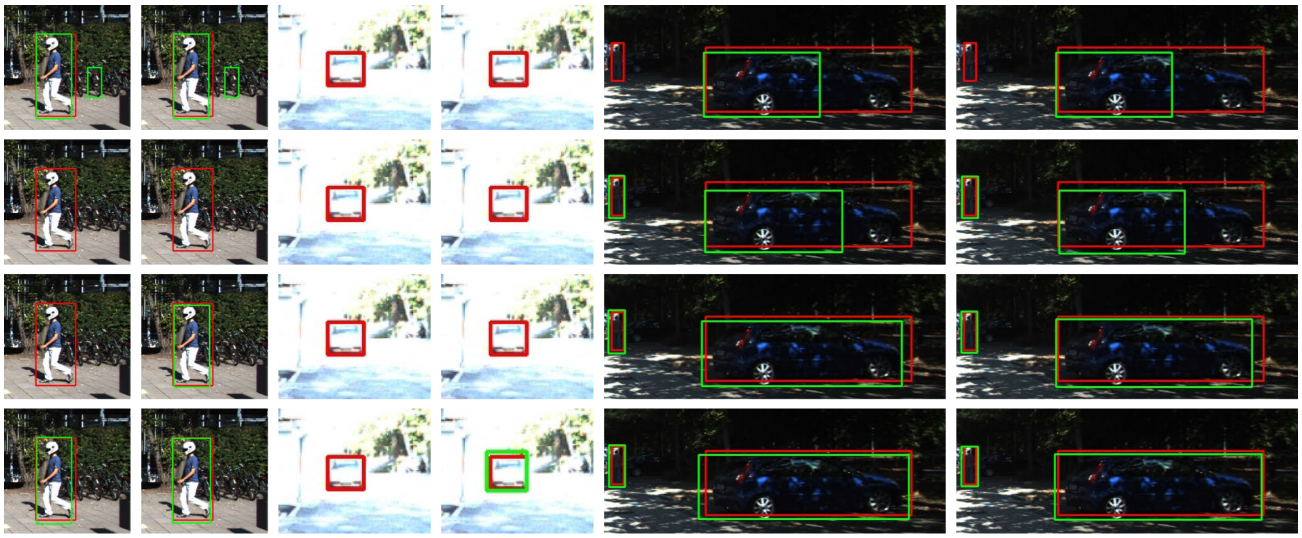

FIGURE 8. Examples of Self/Co-training + ASource. Red BBs are from the ground truth of $\mathcal{K}^{tr}$, and green BBs are predicted. Each block of two columns with the same underlying image compares self-training (left column of the block) and co-training (right column of the block). Top row corresponds to detections in Cycle 0, when, in these examples, the only available training data is $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ (so it is the same for self-training and co-training). The following rows, top to bottom, correspond to detections from cycles 1, 10, and 20, respectively, when self-labeled images are incrementally added to the training set.

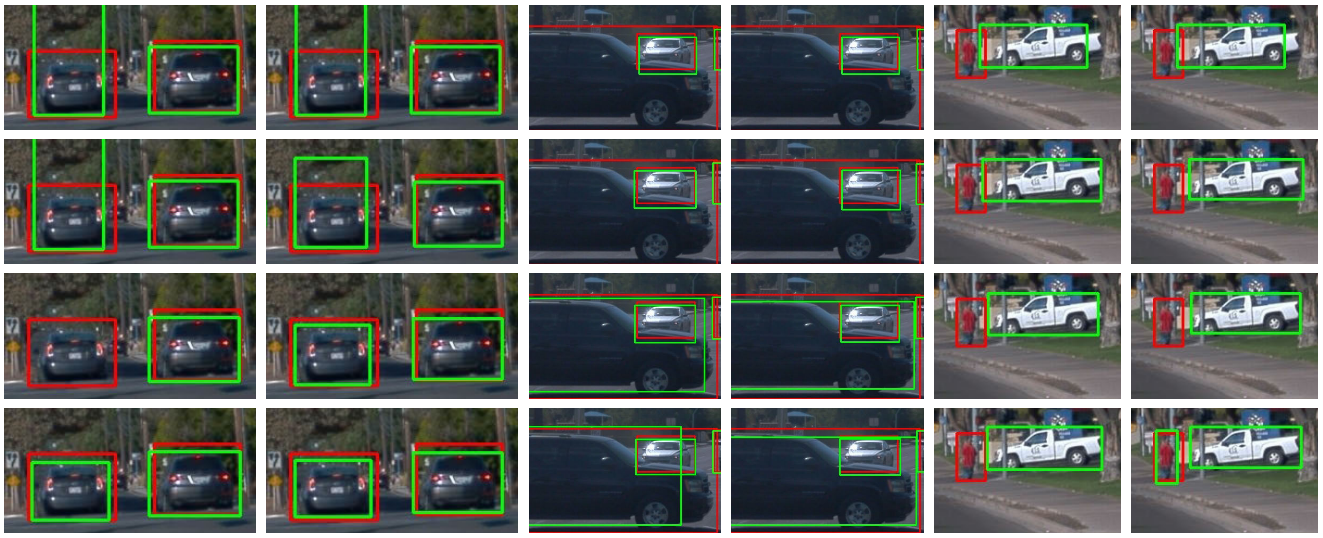

FIGURE 9. Analogous to Figure 8 for $\mathcal{W}^{tr}$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$.

ϕ_j can select the n most difficult for it among them. This is equivalent to set $m = \infty$ in Algorithm 2 (so nullifying the parameter).

Algorithms 1 and 2 return self-labeled images, *i.e.* $\mathcal{X}^{\hat{l}}$. For our experiments, we use $\mathcal{X}^{\hat{l}}$ together with the input labeled set, $\mathcal{X}^{l}$, to train a final Faster R-CNN object detector out of the self-labeling cycles but using the same training settings. This detector is the one used for testing. Finally, to measure object detection performance, we follow the KITTI benchmark mean average precision (mAP) metric [11].

B. RESULTS

We start by assessing the self-labeling algorithms in a pure SSL setting, working only with either the $\mathcal{K}$ or $\mathcal{W}$ dataset. Table 3 shows the results when using the 100% of the respective real-world training data, *i.e.* $\mathcal{X}^{l} = \mathcal{X}^{tr} \in \{\mathcal{K}^{tr}, \mathcal{W}^{tr}\}$, as well as when randomly selecting the 5% or the 10% of $\mathcal{X}^{tr}$ as $\mathcal{X}^{l}$ set, meaning that these subsets are created once and frozen for all the experiments. Table 4 shows the number of object instances induced by the random selection in each case. In Table 3, the 100% case shows the upper-bound performance, and the 5% and 10% act as lower bounds.

We can see that, in all the cases, the self-labeling methods outperform the lower bounds. For vehicles (V), self-training and co-training perform similarly for the 5% lower bound, while for the 10% self-training performs better than co-training in $\mathcal{K}$ but worse in $\mathcal{W}$. For pedestrians (P), co-training always performs better. Looking at the vehicle-pedestrian combined (V&P) performance, we see

TABLE 6. **Results for two new settings: (/FP) assuming we remove the self-labeled false positives; (/FP+BB) assuming that, in addition, for the self-labeled instances, we change the predicted BB by the corresponding one in the ground truth. The Δ_X rows show differences between these variants and the respective original one (*i.e.* neither removing the FP nor adjusting the BBs). The bottom block of rows remarks the differences between the best self-labeling (Co-T+ASource, including /FB and /FB+BB cases) and the upper bound.**

	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
Training set	V	P	V&P	V	P	V&P
Upper bound (UB)	†84.08	†66.00	†75.04	†61.71	†57.74	†59.73
5% Labeled + Co-T	68.99	55.07	62.03	54.00	56.34	55.17
5% Labeled + Co-T/FP	68.37	55.34	61.86	53.94	55.23	54.59
5% Labeled + Co-T/FP+BB	**82.98**	**59.24**	**71.11**	**62.72**	**58.54**	**60.63**
$\Delta_{5\%,\text{ FP}}$	-0.62	+0.27	-0.17	-0.06	-1.11	-0.58
$\Delta_{5\%,\text{ FP+BB}}$	+13.99	+4.17	+9.08	+8.72	+2.20	+5.46
10% Labeled + Co-T	73.08	58.53	65.81	56.15	60.20	58.18
10% Labeled + Co-T/FP	72.94	58.68	65.81	56.62	57.74	57.18
10% Labeled + Co-T/FP+BB	**83.16**	**61.29**	**72.23**	**63.17**	**60.13**	**61.65**
$\Delta_{10\%/\text{FP}}$	-0.14	+0.15	0.00	+0.47	-2.46	-1.00
$\Delta_{10\%/\text{FP+BB}}$	+10.08	+2.76	+6.42	+7.02	-0.07	+3.47
Co-T + Source	73.53	**69.50**	71.52	48.56	<u>56.33</u>	52.45
Co-T + Source/FP	71.64	**69.50**	70.57	51,06	<u>**56.94**</u>	54.00
Co-T + Source/FP+BB	**86.03**	68.97	**77.50**	**59.21**	<u>56.59</u>	**57.90**
$\Delta_{\text{Co-T + Source/FP}}$	-1.89	0.00	-0.95	+2.50	+0.61	+1.55
$\Delta_{\text{Co-T + Source/FP+BB}}$	+12.50	-0.53	+5.98	+10.65	+0.26	+5.45
Co-T + ASource	79.99	**69.01**	74.50	59.99	55.39	57.69
Co-T + ASource/FP	80,16	67.79	73.98	60.70	56,87	58,79
Co-T + ASource/FP+BB	**86.26**	66.69	**76.48**	**63.98**	**58.55**	**61.27**
$\Delta_{\text{Co-T + ASource/FP}}$	+0.17	-1.22	-0.52	+0.71	+1.48	+1.10
$\Delta_{\text{FP+FP+BB}}$	+6.27	-2.32	+1.98	+3.99	+3.16	+3.58
$\Delta_{\text{(Co-T + ASource) vs UB}}$	-4,09	+3,01	-0,54	-1,72	-2,35	-2,04
$\Delta_{\text{(Co-T + ASource/FP) vs UB}}$	-3.92	+1.79	-1.06	-1.01	-0.87	-0.94
$\Delta_{\text{(Co-T + ASource/FP+BB) vs UB}}$	+2.18	-0.69	+1.44	+2.27	+0.81	+1.54

significant improvements over the lower bounds. Interestingly, for the 10% setting, co-training even outperforms the upper bound for pedestrians in $\mathcal{W}$. In fact, in this case, the corresponding V&P performance is just 1.55 points below the upper bound. The same setting in $\mathcal{K}$, improves 5.3 points over the lower bound, but is 7.4 points below the upper bound. These experiments show that our self-labeling algorithms, especially co-training, are performing the task of SSL reasonably well, which encourages to address the UDA challenge with them.

Table 5 shows the UDA results for $\mathcal{V} \rightarrow \mathcal{K}$ and $\mathcal{V} \rightarrow \mathcal{W}$, thus, training with $\mathcal{V}$ acts as lower bound. Just as reference, we also show the results of training on $\mathcal{K}$ and testing on $\mathcal{W}$, and vice versa. The former case shows a similar domain shift as when training on $\mathcal{V}$. The latter case shows a significant lower shift from $\mathcal{V}$ to $\mathcal{K}$, than from $\mathcal{W}$ to $\mathcal{K}$. Thus, we think that $\mathcal{V}$ offers a realistic use case to assess virtual-to-real UDA. The Δ_X rows at the bottom block of Table 5 summarize numerically the main insights.

A first observation is that, by combining GAN-based image-to-image translation and co-training, we obtain significant performance improvements over the lower bounds in all cases; in terms of V&P, 10.22 points for $\mathcal{K}$ and 11.56 for $\mathcal{W}$. In fact, the gap to reach upper-bound performances, is relatively small compared to the improvement over the lower bounds; in terms of V&P, such gap is of 0.54 points for $\mathcal{K}$ and of 2.04 for $\mathcal{W}$. Note that for $\mathcal{K}$, the upper bound comes from the $\mathcal{X}^{l,tr}\&\mathcal{V}_G$ setting (*i.e.* training on the full labeled training set of $\mathcal{K}$ plus $\mathcal{V}_{G_\mathcal{K}}$); while for $\mathcal{W}$, the upper bound comes from the $\mathcal{X}^{l,tr}$ setting (*i.e.* training on the full labeled

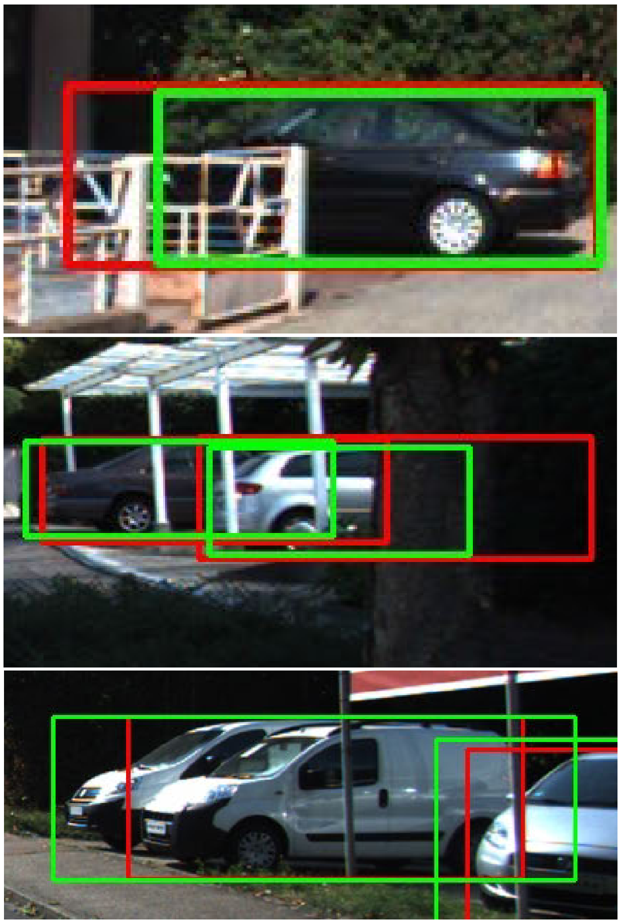
FIGURE 10. **Examples of misalignment between ground truth BBs (red) and self-labeled ones (green). Occlusion is the underlying problem, giving rise to shorter BBs (top and middle) or BBs fusing several instances in one (bottom).**

training set of $\mathcal{W}$). Thus, without any manual training data labeling, we are almost reaching upper-bound performance.

A second observation is that, indeed, co-training brings additional improvements on top of image-to-image translation; in terms of V&P, 5.77 points for $\mathcal{K}$ and 4.94 for $\mathcal{W}$. Note that, when applying them separately to the virtual-world images (source domain), both co-training and CycleGAN (ASource) show similar performance improvements for $\mathcal{W}$ but co-training outperforms CycleGAN in $\mathcal{K}$; in terms of V&P, co-training improves 7.22 points for $\mathcal{K}$ and 6.32 for $\mathcal{W}$, while CycleGAN improves 4.45 and 6.62 points, respectively. Interestingly, CycleGAN performs better than co-training for vehicles and it is the opposite for pedestrians. In any case, using both together outperforms them in isolation.

A third observation is that co-training consistently outperforms self-training. Moreover, in Figure 7 we can see that this is also consistent along self-labeling cycles (we show the UDA setting). It plots curves illustrating how the self-training and co-training strategies would perform as a function of the stopping cycle. We collect the respective self-labeled images at different cycles (x-axis) and train an

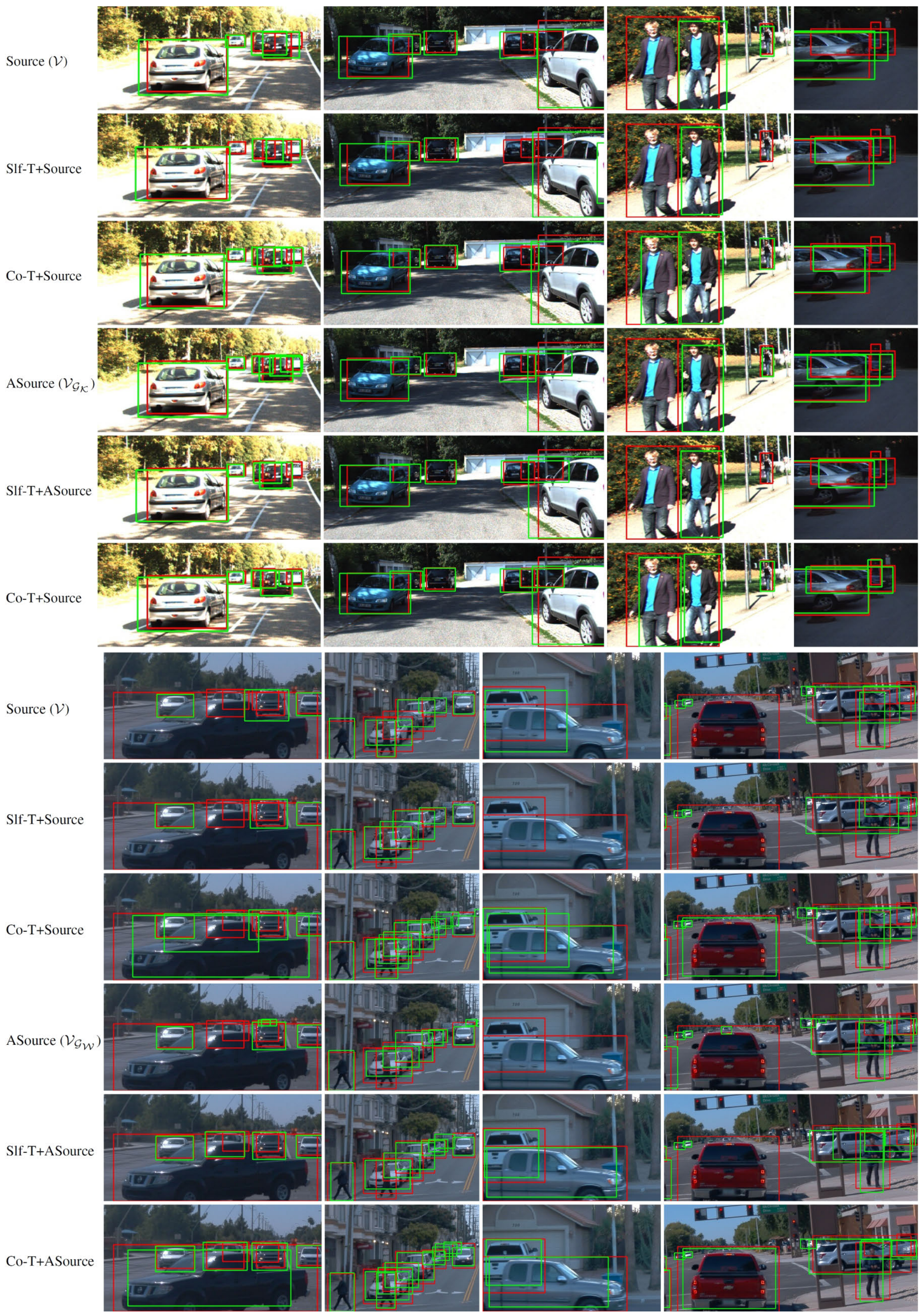

FIGURE 11. **Results for $\mathcal{K}^{tt}$ (top 'Source → Co-T + Asource' block) and $\mathcal{W}^{tt}$ (bottom 'Source → Co-T + Asource' block). Red BBs are the ground truth, and green ones are the detections done by the detector indicated at the first column of each row.**

object detector as these cycles were determined as the stopping ones. Then, we assess the performance of the detector in the corresponding testing set (either from $\mathcal{K}$ or $\mathcal{W}$), so collecting the corresponding mAP (y-axis) per each considered cycle. We see how self-training curves oscillate more around the respective lower bounds, while co-training ones keep improving performance as we train more cycles. Moreover, in Table 4, we can see how co-training systematically self-labels more correct object instances than self-training (Δ_X rows show the increment for SSL and UDA settings).

Figures 8 and 9 show qualitative examples of how self-training and co-training progress in $\mathcal{K}^{tr}$ and $\mathcal{W}^{tr}$ images, respectively; in this case, starting with $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$ as corresponding initial labeled training sets. Both self-labeling strategies improve over the starting point, since they can correct BB localization errors, remove false positives, and recovering from false negatives. In some cases, self-training and co-training show the same final right result, but co-training reaches it in earlier cycles, while in other cases co-training shows better final results.

As summarized in Table 6, with additional experiments, we have further analyzed co-training results. We repeat the training and evaluation of all the detectors developed for both the SSL and UDA settings, considering two variants in the respective training sets. In (/FP) we remove the false positives from the self-labeled data. In (/FP+BB), in addition, for those self-labeled instances that are true positives, we replace the BBs predicted during self-labeling by the corresponding BB ground truth. In this way, we can incrementally analyze the effect of false positives and BB adjustment accuracy.

Table 6 shows how the impact of having false positives as training data is not as strong as one may think a priori. For instance, in Table 4 we can see that co-training with CycleGAN (Co-T + ASource) has output a 28.8% of false vehicles in $\mathcal{W}$ and a 3.1% in $\mathcal{K}$, a large difference; however, Table 6 indicates that removing them results in a vehicle detection improvement of just 0.71 points in the former case, and 0.17 in the latter. This may be linked with the fact that SGD optimization in deep neural networks (DNNs) tend to prioritize learning patterns instead of noise [3].

In general, as Table 6 shows, a better BB adjustment has a higher impact than removing false positives, especially for vehicles. In fact, for the higher performing detector, *i.e.* co-training with CycleGAN, this adjustment would allow to even outperform the upper bound. This improvement is mainly coming from the detection of vehicles. For instance, Figure 10 shows examples of the typical BB misalignment we have found, mainly due to occluded vehicle instances. Note also that the (/FP+BB) results indicate that non-self-labeled objects (false negatives) do not cause any loss of performance, unless they would result in better BB adjustments.

We finish by showing qualitative results on $\mathcal{K}^{tt}$ and $\mathcal{W}^{tt}$ (Figure 11), for different object detectors. Comparing ground truth BBs and detections, we appreciate how it is confirmed what is expected from the quantitative results, *i.e.* co-training combined with GAN-based image-to-image translation is providing the most accurate results.

V. CONCLUSION

Motivated by the burden of manual data labeling when addressing vision-based object detection, we have explored co-training as SSL strategy for self-labeling objects in images. Moreover, we have focused on the challenging scenario where the initial labeled set is generated automatically in a virtual world; thus, co-training must actually perform UDA. We have proposed a specific co-training algorithm which is agnostic to the particular object detector used for self-labeling. We have devised a comprehensive set of experiments addressing the challenging task of on-board vehicle and pedestrian detection, using de facto standards such as KITTI and Waymo datasets, together with a virtual-world dataset introduced in this article. Our qualitative results allow us to conclude that co-training and GAN-based image-to-image translation complement each other up to allow the training of object detectors without manual labeling, while still reaching almost the same performance as by totally relying on human labeling for obtaining upper-bound performances. These results show that the self-labeled objects are sufficient to train a well-performing object detector, but also that improving BB adjustment is convenient to improve its performance. Accordingly, our future work will address this point, for instance, by developing a multi-modal co-training which jointly explores RGB images (as in this article) as well as depth information based on monocular depth estimation [14], where object borders may be better localized.

ACKNOWLEDGMENT

The authors would like to thank several past collaborators and visiting interns of the Computer Vision Center (CVC). They thank Dr. Hamed H. Aghdam for starting discussions on the use of co-training for computer vision tasks. They also thank the interns Mojtaba Valipour and, especially, Sahil Gupta, for helping in the early stages of the research, when they were programming exploratory ideas under their supervision. They also thank Jose L. Gómez for the help in the synthetic data generation.

REFERENCES

[1] Y. Abramson and Y. Freund, "SEmi-automatic VIsuaL LEarning (SEVILLE): A tutorial on active learning for visual object recognition," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2005.

[2] H. H. Aghdam, A. Gonzalez-Garcia, A. Lopez, and J. Weijer, "Active learning for deep detection neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3672–3680.

[3] D. Arpit, S. Jastrzebsk, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 233–242.

[4] M.-F. Balcan, A. Blum, and K. Yang, "Co-training and expansion: Towards bridging theory and practice," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2004, pp. 89–96.

[5] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory - COLT*, 1998, pp. 92–100.

[6] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[7] M. Chen, K. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2011, pp. 2456–2464.

[8] M. Chen, K. Weinberger, and Y. Chen, "Automatic feature decomposition for single view co-training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 953–960.

[9] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3339–3348.

[10] G. Csurka, "A comprehensive survey on domain adaptation for visual applications," in *Domain Adaptation in Computer Vision Applications* (Advances in Computer Vision and Pattern Recognition). Cham, Switzerland: Springer, 2017, ch. 1.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[12] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–16.

[13] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. (2018). *Detectron*. [Online]. Available: https://github.com/facebookresearch/detectron

[14] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3828–3838.

[15] U. Guz, D. Hakkani-Tür, S. Cuendet, and G. Tur, "Co-training using prosodic and lexical information for sentence segmentation," in *Proc. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2007, pp. 1–4.

[16] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 8527–8537.

[17] N. Jaipuria, X. Zhang, R. Bhasin, M. Arafa, P. Chakravarty, S. Shrivastava, S. Manglani, and V. N. Murali, "Deflating dataset bias using synthetic data augmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 772–773.

[18] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 10759–10768.

[19] D. Kim, D. Cho, D. Yoo, and I. S. Kweon, "Learning image representations by completing damaged jigsaw puzzles," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 793–802.

[20] S. Kim, J. Choi, T. Kim, and C. Kim, "Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6092–6101.

[21] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1920–1929.

[22] Levin, Viola, and Freund, "Unsupervised improvement of visual detectors using cotraining," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Oct. 2003, p. 626.

[23] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.

[25] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh, "Generating accurate pseudo-labels in semi-supervised learning and avoiding overconfident predictions via Hermite polynomial activations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11435–11443.

[26] A. M. López, G. Villalonga, L. Sellart, G. Ros, D. Vázquez, J. Xu, J. Marín, and A. Mozafari, "Training my car to see using virtual worlds," *Image Vis. Comput.*, vol. 68, pp. 102–118, Dec. 2017.

[27] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proc. 9th Int. Conf. Inf. Knowl. Manage. CIKM*, 2000, pp. 86–93.

[28] A. Oliver, A. Odena, C. Raffel, E. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 3235–3246.

[29] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, "Deep co-training for semi-supervised image recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 135–152.

[30] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[32] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3234–3243.

[33] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. 7th IEEE Workshops Appl. Comput. Vis. (WACV/MOTION)*, vol. 1, Jan. 2005, pp. 29–36.

[34] S. Roy, A. Unmesh, and V. Namboodiri, "Deep active learning for object detection," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, p. 91.

[35] B. Settles, "Active Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6. San Rafael, CA, USA: Morgan & Claypool, 2012, pp. 1–114.

[36] C. R. de Souza, A. Gaidon, Y. Cabon, N. Murray, and A. M. López, "Generating human action videos by coupling 3D game engines and probabilistic graphical models," *Int. J. Comput. Vis.*, vol. 128, no. 5, pp. 1505–1536, May 2020.

[37] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2446–2454.

[38] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 245–284, Feb. 2015.

[39] G. Tur, "Co-adaptation: Adaptive co-training for semi-supervised learning," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 2009, pp. 3721–3724.

[40] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, Feb. 2020.

[41] D. Vazquez, A. M. Lopez, J. Marin, D. Ponsa, and D. Geronimo, "Virtual and real world adaptation for pedestrian detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 797–809, Apr. 2014.

[42] K. Wang, X. Yan, D. Zhang, L. Zhang, and L. Lin, "Towards human-machine cooperation: Self-supervised sample mining for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1605–1613.

[43] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, Oct. 2018.

[44] W. Wang and Z.-H. Zhou, "Analyzing co-training style algorithms," in *Proc. Eur. Conf. Mach. Learn. (ECML)*, 2007, pp. 454–465.

[45] W. Wang and Z.-H. Zhou, "A new analysis of co-training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 1135–1142.

[46] K. Weiss, T. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 9, 2016.

[47] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 5, pp. 1–46, Sep. 2020.

[48] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3D voxel patterns for object category recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1903–1911.

[49] J. Xu, L. Xiao, and A. M. Lopez, "Self-supervised domain adaptation for computer vision tasks," *IEEE Access*, vol. 7, pp. 156694–156706, Nov. 2019.

[50] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proc. 33rd Annu. meeting Assoc. Comput. Linguistics*, 1995, pp. 189–196.

[51] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 7164–7173.

[52] Z.-H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowl. Inf. Syst.*, vol. 24, no. 3, pp. 415–439, Sep. 2010.

[53] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.

[54] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2019, *arXiv:1911.02685*. [Online]. Available: http://arxiv.org/abs/1911.02685

[55] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 289–305.
[56] Y. Zou, Z. Yu, X. Liu, B. V. K. V. Kumar, and J. Wang, "Confidence regularized self-training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5982–5991.

GABRIEL VILLALONGA received the B.Sc. degree in computer science from the Universitat Autònoma de Barcelona (UAB), in 2014, and the M.Sc. degree in computer vision jointly from UAB, UPC, UPF, and UOC universities, in 2015. He is currently pursuing the Ph.D. degree in computer science focusing on tasks related to perception for autonomous driving. He performed his B.Sc. final project on on-board 3-D pedestrian detection at UAB. He did his M.Sc. dissertation on pedestrian behaviour analysis.

ANTONIO M. LÓPEZ PEÑA (Member, IEEE) is currently the Principal Investigator of the Autonomous Driving Lab, Computer Vision Center (CVC), Universitat Autònoma de Barcelona (UAB). He has also a tenure position as an Associate Professor at the Computer Science Department, UAB. He has a long trajectory carrying research at the intersection of computer vision, computer graphics, machine learning, and autonomous driving. He has been deeply involved in the creation of the SYNTHIA dataset and the CARLA open-source simulator for democratizing autonomous driving research. He is actively working hand-on-hand with industry partners to bring state-of-the-art techniques to the field of autonomous driving. He was granted by the Catalan ICREA Academia Program.