What My Motion tells me about Your Pose:
Self-Supervised Fine-Tuning of Observed Vehicle Orientation Angle

Cédric Picron¹*, Punarjay Chakravarty², Tom Roussel¹, Tinne Tuytelaars¹

¹ESAT-PSI, KU Leuven  ²Ford Greenfield Labs, Palo Alto

Abstract. The determination of the relative 6 Degree of Freedom (DoF) pose of vehicles around the ego-vehicle from monocular cameras is an important aspect of the perception problem for Autonomous Vehicles (AVs) and Driver Assist Technology (DAT). Current deep learning techniques used for tackling this problem are data hungry, driving the need for unsupervised or self-supervised methods. In this paper, we consider the domain adaptation task of fine-tuning a vehicle orientation estimator on a new domain without labels. By leveraging the ego-motion consistencies obtained from a monocular SLAM method, we show that our self-supervised fine-tuning scheme consistently improves the accuracy of the resulting network. More specifically, when transitioning from Virtual Kitti to nuScenes, up to 70% of the performance is recovered compared to the 100% of a supervised method. Our self-supervised method hence allows us to safely transfer vehicle orientation estimators to new domains without requiring expensive new labels.

1 Introduction

The world stands at the cusp of a transportation revolution with the promise of Autonomous Vehicle (AV) technology that has been developing over the past decade. Technology that emerged out of university robotics labs is now visible on the streets of San Francisco and Guangzhou in the form of AV test vehicles. However, these cars are heavily laden with expensive sensors and compute. Most of them sport a ‘tiara’ on the roof, comprising of LIDAR, radar and surround-view cameras. The most expensive element of this sensor suite is the LIDAR, which though falling in price in recent years, still costs a significant fraction of the price of the vehicle. One of the most important tasks of this LIDAR is 3D bounding box detection. This is the task of determining the dimensions and relative poses of 3D bounding boxes for all vehicles observed around the AV. This spatial reasoning between the ego-vehicle and others around it is important for an AV to keep its distance from cars and plan collision-free paths around them, especially on crowded city streets. A major challenge in bringing AV technology to the masses is the price of the LIDAR sensor.

* Correspondence to cedric.picron@esat.kuleuven.be
There has been progress in monocular 3D bounding box detection \cite{1,4}, but there remains a performance gap between 2D vehicle detection and monocular 3D vehicle detection \cite{5}. 3D spatial reasoning solely based on single images remains difficult for neural networks. In addition to this performance gap, the manual annotation of massive amounts of data used to train these neural nets can be extremely costly and time-consuming, and is hence undesirable. Moreover, the data-driven nature of neural nets makes the models domain-dependent: they do not generalize well to new settings, making regular fine-tuning or retraining a necessity. If one were able to use data from these monocular cameras to self-supervise the training or to fine-tune these neural nets, one would be able to use the large amounts of unlabeled data available from existing vehicle fleets on city streets to improve the task of 3D bounding box detection.

In the last decade, there has been tremendous progress in the fields of Visual Odometry (VO) and visual Simultaneous Localization and Mapping (SLAM). Most notably, the problem of monocular SLAM has been tackled with great success \cite{6,7}. These SLAM/VO methods allow us to recover the path of the ego-vehicle from the forward facing camera in global coordinates, also called the ego-motion. When combined with detections of parked/stationary car instances, this type of information can be extremely valuable. As the stationary vehicle is fixed in global coordinates, orientation angle differences of this stationary car can be inferred in the ego-frame from ego-motion. In essence, the observed ego-motion allows us to obtain consistencies between pose estimates for different time instances of the same stationary car.

In this work, we utilize this consistency for self-supervising the fine-tuning of our neural net. Despite being broadly applicable on the full 6 DoF vehicle pose estimation task, we focus here on the vehicle orientation task.

A useful demonstration of our ego-motion based self-supervision is in the Domain Adaptation (DA) \cite{9} setting. DA comprises of techniques that help a neural network generalize to a different environment compared to the environment it was trained on. In our case, these environments are either simulated environments such as Virtual Kitti \cite{10} or real environments such as the specific cities from Kitti \cite{11} and nuScenes \cite{12}. Our goal comprises of training a vehicle orientation estimator on one environment in a fully supervised fashion, and subsequently fine-tuning it in a self-supervised manner on a different environment. An overview of our self-supervised method is found in Figure \ref{fig:overview}.

In our experiments, we show consistent improvements during fine-tuning with our self-supervised method, dramatically closing the gap with supervised methods. We thereby show that it is not necessary to reacquire costly human annotations in new environments, while maintaining excellent performance. Our main contributions can be summarized as follows:

1. We introduce the idea of using ego-motion as a self-supervisory signal for 6 DoF vehicle pose estimation, or more specifically vehicle orientation estimation.

2. We show how naive use of the self-supervisory signal can be detrimental, and how this issue can be solved.
3. We show how the important task of DA can be addressed in the vehicle orientation estimation setting by using our method.

2 Related Work

Monocular 6 DoF vehicle pose estimation. With the popularization of deep learning in the computer vision community, it has not taken very long since the first monocular 6 DoF vehicle pose estimators have appeared [1, 2, 13]. Of particular interest to us, is which representation was chosen for the vehicle orientation. While early works [1, 13] relied on the yaw angle as measured from the ego-frame, it was found that a decomposition in ray and local angle was more beneficial [2] (see Figure 2). This was again emphasized in [14] where the difference between the egocentric orientation and allocentric orientation was explained. Here the yaw angle as measured from the ego-frame equals to the egocentric orientation and the local angle to the allocentric orientation. This decomposition has been used by multiple works ever since [5, 15, 16]. In some works, the orientation is found implicitly, by instead finding the 8 bounding box corners [17]. Finally, note that some works also chose to represent the 3D vehicle orientation as a quaternion [5].

Self-supervision in computer vision. Unsupervised visual learning remains a challenging and mostly unresolved task [18]. Among the vast amount of unsu-
Supervised methods proposed during the years, are self-supervised methods. These techniques make use of a so-called pretext task. Many such pretext tasks have been formulated over the years, both for still images \cite{18,23} and videos \cite{24,27}. Most similar to our work are \cite{28,30} that also use ego-motion as a pretext task, but instead use it as a representation learning tool \cite{28,29} or for single image depth estimation \cite{30}.

Domain Adaptation in computer vision. Domain Adaptation (DA) is a longstanding research topic in computer vision \cite{31}. With the rise of deep learning, \cite{32} were one of the first to observe that the DA problem persists for deeper networks. Since, many deep unsupervised DA methods have been devised. A simple DA method \cite{33} consists of adapting the batch normalization statistics for each environment. For the autonomous driving setting, the DA methods have mostly been focused on the 2D semantic segmentation task \cite{34,35} and the 2D object detection task \cite{36,39}. Many of these methods rely either on synthetic data or make use of adversarial feature matching \cite{40} to minimize the domain gap in the input or feature space. Recently, the DA task for the full 3D object detection setting has also been considered in \cite{41} for one of the first times.

3 Method

Our method consists of different components, some of which are readily available and some of which require specific attention. In Subsection 3.1 we give a general overview of the method where we explain where everything fits in. In Subsection 3.2 and Subsection 3.3 we then turn to how the ego-motion consistencies provided by the SLAM method can best be used. Finally, in Subsection 3.4 we present two simple extensions that are added to our base method.

3.1 A general overview

The goal of our method is to determine the orientations, i.e. yaw angles, of the different vehicles visible in the scene. For this purpose, we take a deep learning approach where we use a deep network to estimate the orientations of the different vehicles. The question is now what the input and output of the network should be. Intuitively, we would like to only provide the cropped image of the vehicle as input, from which the yaw angle as seen from the ego-frame or global angle $\theta_g$ would be estimated as output. There is a problem however. As observed in \cite{2}, it is not possible to infer the global angle $\theta_g$ from the cropped vehicle image only. For the same global angle, the cropped view of the car can appear very different, and on the other hand, for different global angles, the cropped car can appear the same \cite{14}.

Global angle decomposition. A decomposition of the global angle $\theta_g$ is hence required. Following \cite{2}, we decompose the global angle $\theta_g$ as

$$\theta_g = \theta_r + \theta_l,$$

(1)
Self-Supervised Fine-Tuning of Observed Vehicle Orientation Angle

with $\theta_r$ the ray angle and $\theta_l$ the local angle. If we define the $Z$-axis as the axis orthogonal to the camera image plane and the $X$-axis pointing left, then the ray angle $\theta_r$ is defined as

$$\theta_r = \text{atan} \left( \frac{X}{Z} \right),$$

with $(X, Y, Z)$ the position of the vehicle in the ego-frame. These coordinates can be found up to an unknown scale using the inverse camera intrinsic matrix as follows:

$$\begin{pmatrix} X/Z \\ Y/Z \\ 1 \end{pmatrix} = K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix},$$

with $K$ the camera intrinsic matrix and $(u, v)$ the projection of the vehicle center onto the camera frame expressed in image coordinates. We approximate this with the center of the 2D vehicle crop. We circumvent the scale ambiguity issue, as we only need the fraction $\frac{X}{Z}$ as shown in (2). A visual example of the decomposition is found in Figure 2.

![Fig. 2. The ray and local angles, $\theta_r$ & $\theta_l$, corresponding to the blue (ego) and red (observed) vehicle make up the global (yaw) orientation angle of the observed vehicle relative to the ego-vehicle. The ego-motion of the observing vehicle relative to a parked (observed) vehicle constrains the global angle difference between the two observations.]

The main objective of this decomposition was to separate the global angle $\theta_g$ into an angle that accounts for the vehicle position, i.e. the ray angle $\theta_r$, and one angle that accounts for the vehicle appearance, i.e. the local angle $\theta_l$. Hence the goal of the network is now to estimate the local angle $\theta_l$ instead of the global angle $\theta_g$, which can be estimated from the cropped vehicle image only as it is position independent. At inference, once the local angle $\theta_l$ is estimated by the network, we can easily compute the desired global angle $\theta_g$ by applying (3), (2) and (1) respectively.

**Self-supervised targets based on ego-motion.** For the supervised setting the story would end here. The network would be trained by using ground-truth local angles, which can readily be computed by using (1), (2) and (3) based
on the ground-truth global angles and the ground-truth 2D detections. For our self-supervised setting however, matters are slightly more complicated.

In order to obtain the self-supervised target local angles, we make use of consistencies provided by the ego-motion. The main observation here is that the ego-motion tells us what the global angle difference $\Delta \theta_g$ should be between any two time instances of a same stationary car instance. After conversion of the global angle difference $\Delta \theta_g$ to the local angle difference $\Delta \theta_l$, this local difference is used as a self-supervisory signal in our method.

More formally, consider a sequence of yaw angles $\theta_s(t_n)$ for the ego-vehicle which describes the ego-motion as measured by the monocular SLAM method. We denote by $t_n$ the time instance at which the frame was shot, indexed in chronological order by the positive integer $n$. Note that these yaw angles $\theta_s(t_n)$ are computed with respect to a fixed reference frame chosen by the SLAM method. Consider now a stationary car instance that was recorded between $t_a$ and $t_b$, with $a$ and $b$ positive integers such that $b > a$. The goal in a first place is to compute the global angle differences $\Delta \hat{\theta}_g(t_i, t_j)$ of this stationary vehicle as seen from the ego-frame, for every $i, j \in \{n \in \mathbb{N} : a \leq n \leq b\}$. Here, the global angle difference $\Delta \hat{\theta}_g(t_i, t_j)$ is defined as

$$
\Delta \hat{\theta}_g(t_i, t_j) = \hat{\theta}_g(t_j) - \hat{\theta}_g(t_i).
$$

Due to the stationary nature of the vehicle, we now find that

$$
\Delta \hat{\theta}_g(t_i, t_j) = -\Delta \theta_s(t_i, t_j),
$$

where the minus sign originates from the fact that when the ego-vehicle turns left in the fixed SLAM reference frame, a stationary car appears to turn right as seen from the ego-frame. When combining (5) with the definition of $\Delta \theta_s(t_i, t_j)$ which is similar to (4), we obtain

$$
\Delta \hat{\theta}_g(t_i, t_j) = \theta_s(t_i) - \theta_s(t_j),
$$

where the yaw angles $\theta_s(t_i)$ and $\theta_s(t_j)$ are obtained from the SLAM method. Note that the hat symbol in $\Delta \hat{\theta}_g(t_i, t_j)$ reminds the fact that the differences are only approximate due to the approximate nature of the yaw angles $\theta_s(t_i)$ and $\theta_s(t_j)$ obtained from SLAM. Finally, these global angle differences $\Delta \hat{\theta}_g(t_i, t_j)$ are converted into the desired local angle differences $\Delta \hat{\theta}_l(t_i, t_j)$ by using (1), (2) and (3). In our method, these local angle differences $\Delta \hat{\theta}_l(t_i, t_j)$ are used as a form of self-supervision during training. An intuitive visualization of our method is found in Figure 3.

**Naive self-supervised training approach.** A naive self-supervised training approach would proceed as follows. First, pick a pair of 2D detections of the

---

\[1\] Here the subscript $s$ denotes that these angles were obtained by the SLAM method and hence are only approximate, as opposed to the ground-truth ego-motion angles which may be written as $\theta_e(t_n)$, such that $\theta_s(t_n) \approx \theta_e(t_n)$.
same stationary car at two different time instances $t_i$ and $t_j$. Then, feed both cropped images through a siamese network \cite{42} and obtain the estimates $\tilde{\theta}_l(t_i)$ and $\tilde{\theta}_l(t_j)$. Finally, update the deep network by backpropagating gradients based on the loss

$$\mathcal{L} = d\left(\hat{\theta}_l(t_j) - \hat{\theta}_l(t_i), \Delta \hat{\theta}_l(t_i, t_j)\right),$$

with $\Delta \hat{\theta}_l(t_i, t_j)$ the target local angle difference and $d(x, y)$ some distance metric.

This naive approach however does not work in practice. The poor performance of the approach is best explained with an example. Suppose the ground-truth target angles at two time instances are $10^\circ$ and $40^\circ$ while the network predicts $0^\circ$ and $20^\circ$ respectively. As in practice only the target difference of $30^\circ$ is known, the network is told that the difference of $20^\circ$ between its predictions is too small and should be increased. Hence the network is told to mistakenly decrease its $0^\circ$-prediction and to correctly increase its $20^\circ$-prediction. On average, we have observed that 40% of the gradients flow in the wrong direction! This shows the difficulty of learning from differences as it is unknown which of the two predictions should be trusted more, if any. To overcome this issue, we design a more robust approach in Subsection 3.2 and Subsection 3.3.

### 3.2 Sequence offset computation

In this and the following subsection, we design a more robust approach to leverage the information obtained by the ego-motion on stationary car instances. Our base method (i.e. without extensions) consists of three components. In this subsection, we first look at a general technique to compute the offset of a stationary

\footnote{Network estimates are denoted with a tilde.}
car sequence. In next subsection, we then increase the method’s robustness by pruning entries within sequences for the offset computation, and by even removing full sequences from the entire self-supervised training process.

**Offset computation.** Consider again a stationary car that was recorded for $N$ consecutive time instances. If we simplify the notation with

$$ s_i = -\theta_s(t_i), \quad (8) $$

then we can rewrite (6) as

$$ \Delta \hat{\theta}_g(t_i, t_j) = s_j - s_i. \quad (9) $$

Hence the ego-motion provides us a sequence of $s_i$’s that plays a similar role as the true global angles $\theta_g(t_i)$ as per construction

$$ \theta_g(t_j) - \theta_g(t_i) = \Delta \theta_g(t_i, t_j) \approx \Delta \hat{\theta}_g(t_i, t_j) = s_j - s_i, \quad (10) $$

with the difference originating from SLAM inaccuracies. The sequence of $s_i$’s is therefore approximately the same as the sequence of true global angles $\theta_g(t_i)$ up to a constant offset $\alpha$, or

$$ \theta_g(t_i) \approx s_i + \alpha. \quad (11) $$

In order to leverage the ego-motion data, it is hence required to find the best possible estimate $\alpha^*$ for every sequence of stationary car instances.

For this purpose, we consider the DA setting. We assume that a network model $M$ is available which was trained for the same task, but on a different domain. Consider now the same sequence of stationary car instances as above. By feeding every cropped image of the sequence through the model $M$, we can find the rough network estimates $\tilde{\theta}_g(t_i)$, which for simplicity we here write as

$$ r_i = \tilde{\theta}_g(t_i). \quad (12) $$

If we now assume that the original model $M$ is good at capturing the rough magnitude of the different global angles of the sequence, but not the fine details, then we could estimate the offset $\alpha^*$ by solving following minimization problem

$$ \alpha^* = \arg\min_{\alpha} \left( \sum_{i=1}^{N} [r_i - (s_i + \alpha)]^2 \right). \quad (13) $$

By setting the derivative w.r.t. $\alpha$ of above cost function to zero, we find that the optimal offset $\alpha^*$ is given by

$$ \alpha^* = \frac{1}{N} \sum_{i=1}^{N} (r_i - s_i). \quad (14) $$

This offset $\alpha^*$ can then be used to compute the global self-supervised target angles $\tilde{\theta}_g(t_i)$ by applying

$$ \tilde{\theta}_g(t_i) = s_i + \alpha^*, \quad (15) $$
which should be a good approximation of the true global angles \( \theta_g(t_i) \). Finally, these global self-supervised target angles \( \bar{\theta}_g(t_i) \) can then be converted into local self-supervised target angles \( \bar{\theta}_l(t_i) \) by using (1), (2) and (3) as per usual.

### 3.3 Sequence pruning and removal

For some sequences the assumption that the original model \( M \) correctly estimates the rough magnitude of the different local angles (and by extension global angles), is incorrect. In some cases this assumption is only correct for a particular subset of entries within the sequence, while for some other sequences this assumption is incorrect altogether. In this first scenario, we perform **sequence pruning**, which removes entries with low confidence from the sequence. In practice, these might for example be heavily occluded or truncated entries within a sequence. In the second scenario however, no rough estimate within the sequence is reliable and hence the full sequence must be removed. We call this process **sequence removal**. Note that this need for reliable rough estimates explains why our self-supervised method could not be used to train a network from scratch and hence why the DA setting was chosen.

**Sequence pruning.** Consider again a sequence of stationary car instances of length \( N \), where for each possible entry \( i \) we have rough estimates \( r_i \) from the original model and estimates \( s_i \) obtained from the ego-motion data. The goal is to find the indices of entries with poor rough estimates. To do this, we make use of the consistencies provided by \( s_i \). Good rough estimates \( r_i \) and \( r_j \) should approximately satisfy \( r_i - r_j \approx s_i - s_j \), conversely this is no longer true once either \( r_i \) or \( r_j \) are poor estimates. Given a set of indices \( S_n \), its most inconsistent index \( i_{\text{max}} \) is then found by

\[
i_{\text{max}} = \arg \max_{i \in S_n} \mathcal{I}_i = \arg \max_{i \in S_n} \left( \sum_{j \in S_n} |(r_i - r_j) - (s_i - s_j)| \right).
\] (16)

Next, its most consistent index \( i_{\text{min}} \) is computed in a similar way. If now following condition is satisfied

\[
\frac{\mathcal{I}_{i_{\text{max}}}}{\mathcal{I}_{i_{\text{min}}}} > t_p
\] (17)

for a predetermined pruning threshold \( t_p \), and with \( \mathcal{I}_i \) the inconsistency at index \( i \), we decide to prune entry \( i_{\text{max}} \) from \( S_n \) and repeat the same process for the new set \( S_{n-1} = S_n \setminus \{i_{\text{max}}\} \). When starting from \( S_N = \{1, \ldots, N\} \), this procedure is repeated until the condition (17) is no longer satisfied or when only two entries remain within the set. Finally, only the indices within the resulting set are used during the minimization problem (13) for determining the sequence offset \( \alpha^* \).

Pseudo-code of the sequence pruning method is found in Algorithm 1.


**Algorithm 1:** Pseudo-code of sequence pruning method

- **Input:** Sequence length $N$; Prune threshold $t_p$
- **Output:** Pruned list of indices $S$

Initialize $S = \{1, \ldots, N\}$;

\[\text{while } |S| > 2 \text{ do} \]
\[\quad \text{Compute } i_{\text{max}} \text{ according to (16);} \]
\[\quad \text{Compute } i_{\text{min}} \text{ in similar fashion;} \]
\[\quad \text{if } \text{Condition (17) then} \]
\[\quad \quad S = S \setminus \{i_{\text{max}}\} \]
\[\quad \text{else} \]
\[\quad \quad \text{break;} \]

**Sequence removal.** For sequence removal, instead of relying onto a relative condition (17) as during sequence pruning, we look at following absolute condition

\[
\sum_{i,j \in S_3} |(r_i - r_j) - (s_i - s_j)| > 6 t_r, \tag{18}
\]

with $S_3$ the set containing the three most confident indices and with $t_r$ a predetermined remove threshold. When above absolute condition is satisfied, we decide to remove the sequence as a whole because of its inconsistencies with the egomotion data. Note that sequences containing two entries or less are automatically removed as their quality cannot be inferred in a robust way.

### 3.4 Extensions

**Adding cycles.** Our first extension is the introduction of cycles. Before explaining the cycle mechanism, we first introduce some terminology. As highlighted in Figure 1, the DA task we consider consists in first the supervised training of the network on the labeled source domain, and secondly the self-supervised fine-tuning of the network on the unlabeled target domain. In future discussions, we will call these two phases P1 and P2 and their resulting models P1 model and P2 model respectively.

So far, we used the P1 model to compute our sequence offsets as elaborated in previous two subsections. The quality of the offsets and hence of the used self-supervised targets, rely on the quality of the P1 model. A better model namely requires fewer sequences to be pruned or removed, adding more data variety to the P2 phase. After a first self-supervised training cycle, we could therefore reuse the obtained P2 model and use it to compute a new set of offsets and self-supervised targets. Using this enhanced set, we can then further fine-tune our P2 model. This process can then be repeated multiple times, every time further fine-tuning the P2 model. We call this process cycling and every offset computation followed by fine-tuning is defined as a fine-tuning cycle or cycle for short. Its effectiveness is showcased in Section 4.
Adding moving vehicles. Our second extension consists of also adding moving vehicles to the P2 phase. As moving vehicles turn, it is not possible to infer their global angle differences simply from the ego-motion of the ego-vehicle, which is why we currently restricted ourselves to stationary cars. However, most of the time moving cars drive in straight lines and hence the inferred global angle differences from ego-motion are still valid. Occasionally, the moving car will turn and the inferred global angle differences will be wrong. In many cases however, this anomaly will be detected by our sequence pruning and removal mechanism. In Section 4, we show that adding moving vehicles generally does not harm the performance, while substantially simplifying the method by not requiring to differentiate between moving and stationary cars.

4 Experiments

Datasets. For P1, we consider three datasets: Kitti [11], nuScenes [12] and Virtual Kitti [10]. Here, Virtual Kitti is a simulated environment, while both Kitti and nuScenes are real environments. Note that for the Virtual Kitti dataset, we train by making use of all available renderings in order to obtain the most robust P1 model. After P1, we will hence end up with three distinct P1 models, each trained on a different domain. Next, each P1 model is fine-tuned on each real and different environment during P2. Note that we only fine-tune on the real environments, as in practice one is interested in good performing models on real environments as opposed to simulated environments.

Evaluation metric. During both phases, we split the available training data of the domain into a training and validation set. We chose to place the first 80% of each video sequence in the training set and the remaining 20% in the validation set. The validation set of the target domain (P2 dataset) will then be used for evaluation. As absolute evaluation metric, we chose the median orientation angle error for simplicity. This is our main metric and will be featured in the tables containing our experiment results. In addition, we will also use a relative evaluation metric in text which goes from 0% to 100% corresponding respectively to the P1 model performance and the P2 model performance which was obtained by supervised training. Our goal is hence to obtain a P2 model in a self-supervised way with a relative performance as close as possible to 100%. In what follows, we will call this relative metric relative gain or relative recovery.

Baselines. As DA baselines, we tried both the batchnorm adaptation approach from [33] and the adversarial feature matching approach from [40]. However, neither of them was able to successfully fine-tune the P1 model, with resulting P2 models with similar or even worse performance than its corresponding P1 model. A potential reason for their poor performance might be related to the fact that these methods were mainly designed for classification tasks. As baseline, we therefore take the performance of the P1 model instead.

3 We consider the day scenes of the 1.0 version of the dataset.
Implementation details and methodology. As backbone model, we consider the ImageNet-pretrained ResNeXt-50 32x4d model from [43]. The final fully connected layer is replaced with a new linear layer yielding the network’s local angle estimate. During P1, this model is trained using ground-truth 2D detections and vehicle orientations. During P2, the 2D vehicle detections are obtained from a COCO pretrained Mask R-CNN model with a ResNet-50-FPN backbone from [44], while the ego-motion is obtained by monocular ORB-SLAM2 [8]. Note that we use ground-truth information for the matching of the 2D detections. This restriction can easily be removed by replacing the pretrained 2D detector by a pretrained 2D tracker. One could for example pretrain a 2D tracker on simulated data such as on Virtual Kitti.

As P2 hyperparameters, we use pruning threshold $t_p = 1.0$ and remove threshold $t_r = 1.0^\circ$ consistently throughout all our experiments. Hence none of these hyperparameters were tuned towards a specific domain change, proving their general nature. If desired, these hyperparameters can be tuned towards a specific domain change using a small validation set.

We use the Smooth-$L_1$ loss with a quadratic region of width 20$^\circ$. The models are optimized using SGD with momentum 0.9 and weight decay $10^{-4}$. We use a batch size of 32 with a learning rate of $2 \cdot 10^{-5}$ (where losses within a batch are summed instead of averaged). We train for 30 epochs on all domains, with learning rate decrease by a factor 10 after 20 epochs. During P2, we fine-tune for 5 cycles and we use both stationary and moving cars. Here every cycle consists of 30 epochs on Kitti with learning rate decrease after 20 epochs, and of 10 epochs on nuScenes with decrease after 7 epochs. As data augmentation, we only use random horizontal flipping at train time during both phases.

Main results. The main experiment results are found in Table 1. Here the rows correspond to the obtained models and the columns to the considered domain changes. In the first two rows our two reference models are found, with the P1 model (model before fine-tuning) operating as baseline and the supervised P2 model as upper bound. In the five rows below, we can find our self-supervised models obtained after each of the five cycles. Ideally, we would like our self-supervised models to be as close as possible to the supervised performance.

First, consider the domain changes ‘VK→N’ and ‘K→N’ in the left two columns. We can see that their P1 models perform relatively poorly on nuScenes with median angle errors of 41.2$^\circ$ and 25.1$^\circ$ still far from the supervised 2.7$^\circ$ error. There is hence a whole lot to be gained during the P2 fine-tuning phase. When looking at our self-supervised P2 models, we see in both domain changes the error significantly being reduced to the low tens. Note how both cases benefited from our cycle mechanism with the error steadily decreasing after every cycle. For the ‘VK→N’ domain change, we end up with an error of 13.9$^\circ$, which is a relative gain of approximately 70% without using any label. Note moreover that as the P1 model originates from the simulated Virtual Kitti environment, the whole model in fact was obtained without using a single human annotation.
Next, we look at the two right columns which correspond to the domain changes ‘VK→K’ and ‘N→K’. First, consider the ‘VK→K’ domain change which starts with a median angle error of 14.9° before fine-tuning. When looking at our self-supervised P2 models, we see the angle error drop, but saturate at approximately 10°. Considering that we also ended up with angle errors around 10° when fine-tuning on nuScenes, it hence seems that our method in general can fine-tune up till 10° and saturates from there on. Once around 10°, the cycle mechanism starts oscillating as evidenced by the errors obtained during the ‘VK→K’ domain change. Finally, consider the ‘N→K’ domain change which starts with a median angle error of 4.2° before fine-tuning. Given this extremely low error, our self-supervised P2 models can no further improve the performance, ending with the same 4.2° angle error as before fine-tuning. We can hence conclude that if the P1 model corresponding to a particular domain change has an error larger than 10°, this will be fine-tuned to an error of approximately 10° by our self-supervised method, while P1 models of domain changes with error smaller than 10° will be left unchanged (performance-wise) after fine-tuning.

**Ablation study.** Next, we perform an ablation study of the different components and mechanisms of our method. Here we keep the same settings as before, except that we fine-tune for one cycle only. The results of the ablation study are found in Table 2.

Here the first three rows are copied from Table 1 which operate as reference with which the different ablations will be compared. First, in the two middle rows, we ablate the pruning and removal mechanism, which were introduced in Subsection 3.3. When separately ablating both mechanisms, we consistently see the performance drop compared to our fully equipped P2 model, unambiguously showing the importance of both components. Here the performance loss is severe when removing the pruning mechanism, and more moderate when ablating the removal mechanism.

Next, we investigate in the bottom three rows of the ablation study whether imperfect 2D detections, imperfect angle differences and the addition of moving cars negatively affect the performance of our self-supervised method. First, when using ground-truth (gt.) detections, no improvements are observed compared to our Mask R-CNN detections showing that the imperfect detections did
Table 2. Ablation study results

|                      | VK → N | K → N | VK → K | N → K |
|----------------------|--------|-------|--------|-------|
| P1 model             | 41.2°  | 25.1° | 14.9°  | 4.2°  |
| P2 (supervised)      | 2.7°   | 2.7°  | 2.9°   | 2.9°  |
| P2 (cycle 1)         | 23.1°  | 14.1° | 10.4°  | 4.4°  |
| No pruning           | 37.9°  | 25.0° | 28.9°  | 10.0° |
| No removal           | 31.0°  | 16.9° | 13.0°  | 4.6°  |
| Gt. detections       | 23.5°  | 13.6° | 10.6°  | 4.1°  |
| Gt. angle diffs.     | 24.0°  | 14.8° | 8.6°   | 3.5°  |
| No moving            | 23.8°  | 14.4° | -      | -     |

not harm our method’s performance. Next, we investigate whether correct angle differences for both stationary and moving cars would improve our model. Recall that our method uses SLAM to infer these angle differences from its estimated ego-motion. Note that two type of errors are involved here. On the one hand these obtained angle differences are not applicable to moving cars and on the other hand these differences only approximately describe the ego-motion, also resulting in inaccuracies for stationary cars. Consider now the case of fine-tuning on nuScenes in the left two columns. Here we can see that using gt. angle differences does not improve the model, showing that both the use of SLAM as ego-motion estimator and the use of moving cars does not harm the performance. In the two leftmost entries of the last row, we confirm the latter by excluding the moving cars according to the gt. labels and still obtain similar performance. When we consider fine-tuning on Kitti in the right two columns, we see however that we would benefit from using gt. angle differences. This seems to show that if we want to go below the 10° error mark exhibited by our self-supervised method, some performance might still be found by using a more precise SLAM algorithm and/or by excluding the moving cars. Note that the latter could not be verified as, in contrast to nuScenes, Kitti does not provide labels whether a car is moving or stationary. This explains the dashed entries found in the two rightmost columns of the last row.

5 Conclusion

This paper uses the ego-vehicle’s motion and the 2D tracking of cars to self-supervise the fine-tuning of a vehicle orientation estimator. The neural net outputs the vehicle’s yaw angle with respect to the ego-vehicle, which is an important intermediate step in the estimation of its full 6 DoF pose.

The potential of this approach is demonstrated in a Domain Adaptation (DA) setting, where supervised training is followed by self-supervised fine-tuning. We show that up to 70% of performance is recovered compared to supervised training. In addition, by leveraging simulated data from Virtual Kitti, we were able to obtain a median validation error of 13.9° on nuScenes without using a single human annotation, significantly reducing the gap with supervised methods.
References

1. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2147–2156
2. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 7074–7082
3. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9287–9296
4. Ding, M., Huo, Y., Yi, H., Wang, Z., Shi, J., Lu, Z., Luo, P.: Learning depth-guided convolutions for monocular 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. (2020) 1000–1001
5. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 1991–1999
6. Engel, J., Schöps, T., Cremers, D.: Lsd-slam: Large-scale direct monocular slam. In: European conference on computer vision, Springer (2014) 834–849
7. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics 31 (2015) 1147–1163
8. Mur-Artal, R., Tardós, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics 33 (2017) 1255–1262
9. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing 312 (2018) 135–153
10. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 4340–4349
11. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32 (2013) 1231–1237
12. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
13. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 2040–2049
14. Kundu, A., Li, Y., Rehg, J.M.: 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 3559–3568
15. Manhardt, F., Kehl, W., Gaidon, A.: Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2069–2078
16. Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 1057–1066
17. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3d object localization. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 33. (2019) 8851–8858
18. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. (2019) 1920–1929
19. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728 (2018)
20. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: Advances in neural information processing systems. (2014) 766–774
21. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision, Springer (2016) 69–84
22. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1422–1430
23. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3733–3742
24. Goroshin, R., Bruna, J., Tompson, J., Eigen, D., LeCun, Y.: Unsupervised learning of spatiotemporally coherent metrics. In: Proceedings of the IEEE international conference on computer vision. (2015) 4086–4093
25. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: European Conference on Computer Vision, Springer (2016) 527–544
26. Pathak, D., Girshick, R., Dollár, P., Darrell, T., Hariharan, B.: Learning features by watching objects move. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 2701–2710
27. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 2794–2802
28. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: Proceedings of the IEEE international conference on computer vision. (2015) 37–45
29. Jayaraman, D., Grauman, K.: Learning image representations tied to ego-motion. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1413–1421
30. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 1851–1858
31. Csurka, G.: Domain adaptation in computer vision applications. Volume 2. Springer (2017)
32. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 1 (2013)
33. Chang, W.G., You, T., Seo, S., Kwak, S., Han, B.: Domain-specific batch normalization for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 7354–7362
34. Saleh, F.S., Aliakbarian, M.S., Salzmann, M., Petersson, L., Alvarez, J.M.: Effective use of synthetic data for urban scene semantic segmentation. In: European Conference on Computer Vision, Springer (2018) 86–103
35. Tsai, Y.H., Hung, W.C., Schelter, S., Sohn, K., Yang, M.H., Chandraker, M.: Learning to adapt structured output space for semantic segmentation. In: Pro-
36. Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L.: Domain adaptive faster r-cnn for object detection in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 3339–3348
37. He, Z., Zhang, L.: Multi-adversarial faster-rcnn for unrestricted object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 6668–6677
38. Khodabandeh, M., Vahdat, A., Ranjbar, M., Macready, W.G.: A robust learning approach to domain adaptive object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 480–490
39. Rodriguez, A.L., Mikolajczyk, K.: Domain adaptation for object detection via style consistency. arXiv preprint arXiv:1911.10033 (2019)
40. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research 17 (2016) 2096–2030
41. Wang, Y., Chen, X., You, Y., Li, L.E., Hariharan, B., Campbell, M., Weinberger, K.Q., Chao, W.L.: Train in germany, test in the usa: Making 3d object detectors generalize. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 11713–11723
42. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a “siamese” time delay neural network. In: Advances in neural information processing systems. (1994) 737–744
43. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1492–1500
44. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. (2017) 2961–2969