

Time-to-Label: Temporal Consistency for Self-Supervised Monocular 3D Object Detection

Issa Mouawad¹, Nikolas Brasch², Fabian Manhardt³, Federico Tombari^{2,3}, Francesca Odone¹

Abstract—Monocular 3D object detection continues to attract attention due to the cost benefits and wider availability of RGB cameras. Despite the recent advances and the ability to acquire data at scale, annotation cost and complexity still limit the size of 3D object detection datasets in the supervised settings. Self-supervised methods, on the other hand, aim at training deep networks relying on pretext tasks or various consistency constraints. Moreover, other 3D perception tasks (such as depth estimation) have shown the benefits of temporal priors as a self-supervision signal. In this work, we argue that the temporal consistency on the level of object poses, provides an important supervision signal given the strong prior on physical motion. Specifically, we propose a self-supervised loss which uses this consistency, in addition to render-and-compare losses, to refine noisy pose predictions and derive high-quality pseudo labels. To assess the effectiveness of the proposed method, we finetune a synthetically trained monocular 3D object detection model using the pseudo-labels that we generated on real data. Evaluation on the standard KITTI3D benchmark demonstrates that our method reaches competitive performance compared to other monocular self-supervised and supervised methods.

I. INTRODUCTION

Enabling computer systems to perceive the 3D world has received a lot of attention in the recent years, as it serves numerous applications in autonomous driving, robotics and many more [1], [2], [3], [4]. In particular, monocular approaches are growing in interest due to the advantages in availability and cost of RGB cameras, in comparison with actual 3D sensors. As for autonomous driving, lidar sensors are currently part of the standard equipment of every car, nevertheless, due to cost and availability advantages, in addition to the higher spatial resolution, cameras will most likely be the main sensor in future large-scale deployment. Hence, being able to perceive the 3D scene and detect all objects within it from RGB data alone is of high importance.

Unfortunately, the training of such models requires access to large-scale labeled datasets in order to meet generalization and accuracy standards. This is particularly challenging for 3D perception tasks central to autonomous driving and robotics applications, as it requires to tightly hand-label millions of 3D bounding boxes. Nonetheless, it is worth mentioning that, although labeling such samples is very expensive, recording of appropriate data is fairly cheap and can be achieved by simply driving around. Therefore, being

able to train on this data in a self-supervised fashion without the utilization of any labels would be highly beneficial, enabling to easily scale to larger volumes of training data and effectively pushing forward both accuracy and generalizability. While self-supervision has already been well explored in several disciplines [5], [6], only a handful of methods started very recently to apply self-supervision to the domain of 3D object detection [7], [8], [9]. These very recent works commonly leverage 3D shape priors and differentiable rendering pipelines to derive consistency losses as means of supervision, in the absence of annotations [9], [7], [10]. While these methods generally perform well, they completely neglect any temporal information, which can however serve as a very strong source of 3D supervision as demonstrated in many works from other domains [11], [12].

In this work we propose a novel pipeline that leverages raw lidar information together with temporal information to establish supervision for monocular 3D object detection without the need for any real pose labels. In particular, we train a state-of-the-art monocular 3D object detector completely in simulation. In the following step, we utilize the obtained model to label real data with pseudo labels. These pseudo labels are then refined based on establishing coherence with the 3D scene, in the form of a 3D lidar point cloud, and temporal information from neighboring frames. Our contributions can be summarized as follows. We propose

- a simple, yet effective, *motion classification module* which can estimate the motion state despite noise and outliers in the pose estimates,
- a novel *temporal consistency loss*, which builds on the temporal prior of objects trajectories and helps accurately refining noisy pose estimates,
- a self-supervised temporally-aware framework for generating 3D labels from image sequences and unannotated lidar pointclouds.

II. RELATED WORK

A. Monocular 3D Object Detection

Research in monocular 3D object detection has seen a significant surge recently. Recovering 6D pose of known objects, or, more recently, predicting rough 3D shapes (cuboids) in a category-based detection, represents an essential building block in 3D perception systems, for applications in Robotics, and Autonomous Driving, among others.

According to a recent taxonomy [13], monocular 3D object detection methods can be grouped into methods which lift 2D detections to 3D (referred to as result lifting-based

¹:MaLGA-DIBRIS, University of Genoa, Italy, issa.mouawad@edu.unige.it, francesca.odone@unige.it

²:Technical University of Munich, Germany; {nikolas.brasch, federico.tombari}@tum.de

³:Google Inc. fabianmanhardt@google.com

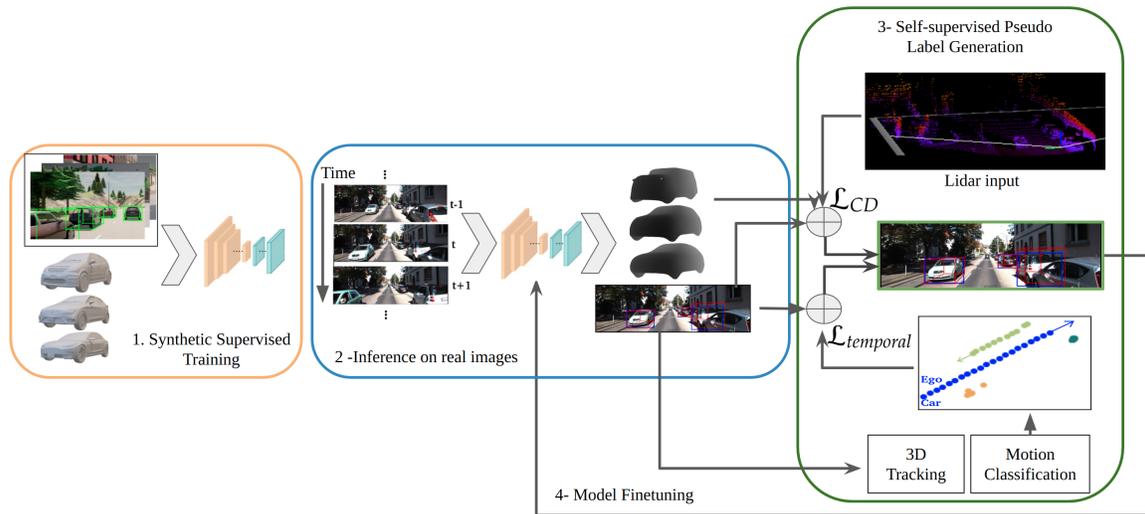


Fig. 1. Schematic overview of our proposed method. (1) We first train a monocular 3D object detector completely in simulation in a supervised fashion. (2-4) Using the 3D scene geometry together with temporal data, we create high-quality pseudo labels that we use to finetune the model.

methods) and methods which process 3D features derived from the 2D image plane.

Early methods are dominantly from the first category. Such works first estimate 2D properties such as the 2D location, dimensions and orientation. Then, the 3D center is retrieved by estimating the center’s depth z and then back-projecting the 2D center according to the regressed depth to 3D. Several of those detectors are built on top of region-proposal 2D detectors [14] and use the RPN features to estimate 3D properties of objects [15], [16], [1], [17]. While others exploit the low computational cost of single-stage 2D detectors to deliver an efficient 3D inference [18], [19], [20], [21], [22].

Other recent methods, falling in the second category, propose to handle 3D aspects early in the pipeline such as pseudo-lidar methods [23] which transform depth to a lidar point cloud, and methods which directly transform the CNN features from the 2D image plane to the world 3D reference system (or directly to the bird’s eye view) [24].

Despite the recent progress of monocular methods, learning 3D object detection from images is an ill-posed problem and requires access to a large number of examples. In a supervised setting, manually produced labels require an expensive and tedious procedure to annotate. On the other hand, using a limited amount of labeled data carries the risks of over-fitting and poor performance.

B. Self-Supervised Learning

In order to exploit the potential of large-scale data, learning without the need for annotations has gained a lot of attention recently. Self-supervised representation learning (SSRL) [25] is used to extract meaningful representations for solving various visual tasks, such as image classification [26], object detection [27], and visual tracking [28], to name a few. Furthermore, self-supervised learning has been used to directly train models on the downstream task without supervision, harnessing additional geometric and consistency

priors. This direction has witnessed a growing interest lately with methods addressing monocular depth estimation [29], [30], optical flow [31], and scene flow prediction [32].

In 3D object detection, the self-supervision typically requires a strong prior on the scene, which is commonly encoded in the form of initial predictions generated by a pre-trained model [7], or a latent variable decoders [33] in addition to 3D shape object priors [34]. Exploiting Differentiable Learning pipelines [35], and render-and-compare losses [2], different consistency losses are proposed both in 3D [33], [9], [10] and in 2D using mask, appearance and other photometric similarities [10], [33], [7]. To improve learning stability and provide fairer comparisons, a recent direction frames the self-supervision as a pseudo-label generation. In this framework, pseudo-labels are generated using initial hypotheses of 6D poses, these labels are then refined and subsequently utilized to train a 3D object detection network in a supervised manner. Recent works either use a combination of RGB images and lidar point cloud during training [9], or rely solely on lidar point clouds to generate 3D pseudo-labels [36], [37], [38].

Nonetheless, although achieving great results despite the lack of labels, none of these approaches make use of temporal consistency across frames as a prior on object pose estimation to further strengthen the pseudo label generation.

C. Temporal Consistency for Self-supervision

Temporal consistency across time in videos represents a rich source of prior information. Recently, several visual tasks try to explicitly model and exploit this consistency. In visual tracking [28] temporal consistency is used to self-supervise visual correspondence representation learning. In tasks where time is a key component, such as scene flow, a good temporal prior is essential to capture meaningful representations [39]. Notice that also pixel-wise tasks, not directly related to time evolution, can benefit from temporal

consistency both during training and inference, as for example depth estimation [29], [30] and semantic segmentation [40].

Temporal Consistency remains a largely unexplored direction for 3D object detection supervision, despite recent attempts to use it in a supervised framework [41] or in domain adaptation for lidar object detection [42].

III. METHODOLOGY

We aim to train a monocular 3D object detector from unlabelled data sequences. Unfortunately, this is a very challenging and highly ill-posed problem due to the nature of the perspective projection onto the image plane. Therefore, we slightly relax the problem, assuming to have access to a lidar and IMU/GPS sensor at training time. We believe that this is a fair assumption as almost all common benchmark datasets or test vehicles provide the respective data. Notice that at inference time, our network is capable of estimating the 6D object pose from a single RGB image alone.

Inspired by [7], [8], we propose to pre-train a 6D pose and 3D shape prediction network on synthetic images (step 1 in Figure 1). Next, using the obtained trained base model, we can estimate initial hypotheses of the object pose on real data (step 2 in Figure 1). Furthermore, leveraging scene geometry and *temporal consistency*, we generate temporally-aware pseudo-labels (step 3), which can be harnessed to finetune the base model (step 4). Steps (2 to 4) are then iteratively repeated to further improve the pseudo-labels quality and thus the finetuned model, as discussed in Section IV-C.

A. Training in Simulation

As our self-supervision requires 3D shapes to enforce consistency with the 3D scene, we require the monocular 3D detector to predict additional object properties to encode the 3D shape. To this end, we utilize CAD models of various cars brands to obtain a low dimensional shape space with PCA. We then fully supervise the whole architecture (3D detection and 3D shape prediction) on synthetic data from the open-source CARLA [43] simulator adding an L2 loss term for shape supervision in the latent embedding of the PCA space. At inference, we can use the estimated latent encoding to reconstruct the underlying 3D shape *w.r.t.* the learned principal components. Exemplary training samples can be found in Figure 2.

B. Generation of Initial Pose Labels

After having trained our base model in simulation, we feed it with real RGB images to predict the 3D translation t_i , the rotation around the vertical axis yaw_i , and the shape embedding s_i for each object i . We then reconstruct the corresponding 3D point cloud $\hat{P}_i = D_{PCA}(s_i)$ using the learned principal components. To further improve the obtained pose labels, we apply an additional refinement step using lidar information. Therefore, we transform the estimated pointcloud to the lidar reference frame following

$$\hat{P}_{lidar} := (R_{yaw_i} \hat{P}_i + t_i), \quad (1)$$

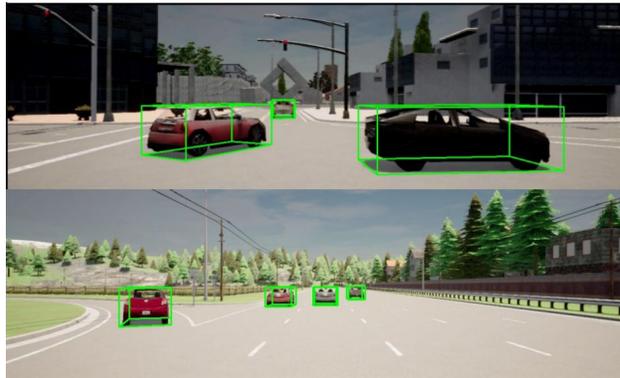


Fig. 2. Training samples generated with the CARLA simulator [43]

with R_{yaw_i} referring to the 3D rotation matrix *w.r.t.* the angle yaw_i . Thereafter, we employ Chamfer distance [44] to measure the misalignment between the observed lidar scan \tilde{P} and the predicted point cloud \hat{P}_{lidar} as follows:

$$\mathcal{L}_{CD} = \sum_{x \in \tilde{P}} \min_{y \in \hat{P}_{lidar}} \|x - y\|_2^2 + \sum_{y \in \hat{P}_{lidar}} \min_{x \in \tilde{P}} \|x - y\|_2^2, \quad (2)$$

which we optimize for our pose parameters R_i and t_i to improve their fit with the scene geometry.

As the observed lidar scan \tilde{P} is obtained for the whole scene, we need to filter-out points which do not belong to the object. This is accomplished by means of removing all points whose projection do not belong to the instance mask M_i as predicted by Mask R-CNN [45]. Note that our Mask R-CNN is pre-trained on a general purpose dataset (*i.e.* COCO [46]), thus, requiring no domain-specific supervision.

C. Supervision by Means of Temporal Consistency

Due to several sources of errors such as low lidar-coverage (due to occlusion for example) or weak initial pose estimates, the supervision from 3D points alone often generates a noisy signal. Therefore, we strengthen the refinement using temporal sequences. Thereby, we build a unified and temporally coherent world reference system across each video sequence to handle the presence of ego-motion, as well as the motion of all other cars in the scene.

Nevertheless, to build such world-reference system we first need to understand our own motion, which we obtain from IMU/GPS measurements. Secondly, we are also required to understand the motion of each other vehicle. To this end, we simply utilize the online 3D tracker [47] to build 3D trajectories from individual 3D predictions for each image in the sequence. Finally, to bring all objects to our temporally coherent world reference system, we project all observations t_i to the camera system of the initial frame. The respective 3D translation t_{global_i} is obtained by multiplying the observation t_i with $T_{i,0}$, the transformation from the reference system at frame i to that of frame 0, obtained from the IMU/GPS measurements, according to:

$$t_{global_i} = T_{i,0} \cdot t_i. \quad (3)$$

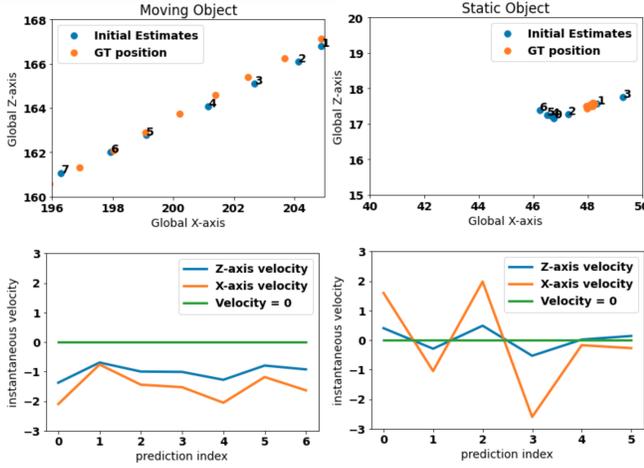


Fig. 3. World reference system observations of a static object (right) and a moving one (left) with the corresponding velocity profiles (bottom)

To bring the rotation into our world reference system yaw_{global_i} , we compute the object’s heading direction as

$$P_{heading} = Rot_{yaw_i}[1, 0, 0]^T \quad (4)$$

We then project $P_{heading}$ to the global reference system and calculate the angle of the vector $\overrightarrow{P_{heading}}$ with the horizontal axis, and employ it as the global object orientation.

We observe that a motion prior can provide an additional source of supervision which, depending on whether a car is parked or moving, can help recovering noisy initial estimates. Thereby we distinguish between two cases: (i) objects that are *static* relative to the scene, (ii) objects that are *moving*, while for objects which have less observations than $min_frames = 6$, we do not apply the temporal consistency. Unfortunately, due to noisy estimates from the trained base model, retrieving the motion state is not straightforward. Yet, we argue that temporal evolution of the object position, can provide insights about a plausible motion pattern or noise caused by pose estimation. Therefore, we employ velocity profiles to perform the motion state classification. First, we separately sum the instantaneous signed velocities across the X and Y axis. Moving objects yield large values, while static objects exhibit velocities with different signs, which cancel out and yield a small traversed distance. We use an empirical threshold of 3 meters (computed from the average vehicle dimension) to obtain the motion state. Thereby, whenever the traversed distance is less than the threshold we consider the object as static and moving otherwise. While we deem the previous condition sufficient to classify an object as static, we observe some situations where few outlier observations cause static objects to produce larger velocity values. Thus, for objects which fail the first condition, we further check zero-crossings of the velocity profile since the frequency of zero-crossings is not affected by the actual amplitude of the velocity values. We then classify as static, objects where zero-crossings events happen in at least 40% of the observations.

Eventually, to enforce temporal consistency as supervision,

we utilize two different formulations of our temporal consistency loss, depending on the motion state of the object. As for **static** objects, we first calculate the median scene position of all the observations t_{glob}^{median} , and consider it as an additional regularization for the refinement procedure (in addition to the Chamfer distance). Further, for the yaw angle of static cars, we express the observed yaw angles (of all the object instances) in our world reference system and then organize the angles in a 32-bin histogram. We pick the most frequent bin yaw_{glob}^{mean} as the best estimate of the yaw angle to regularize the rotation, combining measurements from different view points. Finally, to refine an observation i of a static object, we optimize the following loss function

$$\begin{aligned} \mathcal{L}_{static} = & \lambda_t \|t_i - t_i^{median}\|_2^2 \\ & + \lambda_r \|yaw_i - yaw_i^{mean}\|_2^2, \end{aligned} \quad (5)$$

where the translation t_i^{median} and the rotation yaw_i^{mean} are both expressed in the local reference system by projecting the median pose back to the respective time frame. Notice that given the increased stability of the optimized pose, we further propagate it to adjacent frames (in which the object is not detected) to account for false negatives induced by large distance or truncation.

As for **moving** cars, we model the motion of a vehicle using a piece-wise linear trajectory (in the global reference system), which is a simple, yet expressive model for motion (assuming adequate video frame-rate of at least 10 fps). We set the length of each segment to 10 frames, and fit a linear function using RANSAC. For each observation in the segment, we project the initial estimate on the RANSAC fitted motion model and obtain $t_{smoothed}$ as a regularization for the translation. We further assume the orientation to be roughly constant within the small time window and derive yaw_{line} as the direction of the fitted line, which we then use as a regularization for the rotation. This explicitly improves the orientation continuity and suppresses observation noise. Similar to Equation 5, the loss function to optimize a moving vehicle observation is formulated as:

$$\begin{aligned} \mathcal{L}_{moving} = & \lambda_t \|t_i - t_{smoothed}\|_2^2 \\ & + \lambda_r \|yaw_i - yaw_{line}\|_2^2. \end{aligned} \quad (6)$$

Summarizing, after training our model fully supervised in simulation using scenes generated from CARLA, we use it to compute noisy pseudo-labels for each frame. We then estimate the motion state of each vehicle and optimize the noisy observations accordingly to

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{CD} + \mathcal{L}_{temporal} \\ \mathcal{L}_{temporal} = & \begin{cases} \mathcal{L}_{static}, & \text{if static object} \\ \mathcal{L}_{moving}, & \text{if moving object.} \end{cases} \end{aligned}$$

Afterwards, we finetune our initial model on the obtained pseudo-labels using the losses from [18]. This process is then repeated until convergence.

IV. EXPERIMENTS

In the following we will first present our evaluation protocol before we demonstrate several ablations, proving the usefulness of our contributions. We then conclude by comparing our method to several state-of-the-art works for training with and without real pose labels.

A. Evaluation Protocol

a) *Datasets*: We use the **KITTI** dataset, which is composed of a number of driving sequences with the corresponding lidar scans, both annotated with object bounding boxes. To learn 3D object detection in a self-supervised manner, we do not make use of any ground truth information during training, and we exclusively use them for evaluation purposes. Specifically, we use the split proposed by [48] to train our model without ground truth labels, while we use the validation split, to report results and compare with other methods both in terms of average precision in 3D (AP 3D) and for the boxes projected on the ground plane or AP bird’s eye view (AP BEV).

B. Synthetic Data Training

We generate an urban dataset with Carla Simulator [43] having camera viewpoint and intrinsics similar to those in the KITTI dataset, comprising 50K images with 6D pose labels. We use 12 different car models in the training set, for which we generate PCA shape encodings.

As the choice of the backbone 3D object detector is not part of our main contribution, we simply adopt the state-of-the-art MonoFlex [18] detector for all of our experiments. MonoFlex is built on top of CenterNet [49] and conducts single-stage anchor-free 2D object detection together with the estimation of object-wise properties, including 3D translation and horizontal plane rotation (yaw). Notice that MonoFlex only predicts one angle of the 3D rotation as it assumes that all objects stand on the ground plane, which is a common practice in the autonomous driving scenario. As mentioned in Section I, we extend MonoFlex to predict 3D shape encodings supervised by the encodings of the ground truth CAD models. We further employ DLA-34 [50] as backbone initialized with Imagenet weights. After freezing the first three blocks of the backbone, we train the rest of the network for 10 epochs on the tasks of interest, *i.e.* 2D and 3D detection, and shape encoding regression. During training, we incorporate horizontal flip and color jitter augmentations, to minimize overfitting on the synthetic domain.

C. Pseudo-label Generation

We run pseudo-label generation routine for three iterations (using $\lambda_t = 0.25$ and $\lambda_r = 2$) and for 100 pose refinement steps, where we alternate between generating pseudo-labels and finetuning of the initial model on those pseudo-labels. When finetuning the model on the pseudo-labels we freeze an additional block in the backbone and we finetune all the heads as before, training for 4 epochs during each iteration.

In Table I, we show the quality of our pseudo-labels on the train split after each iteration. Despite the large domain gap

between CARLA and KITTI, and the poor initial estimates, results demonstrate that the pseudo-labels improve as the model is finetuned and produces better initial estimates in the subsequent iterations, surpassing pseudo-labels generated by Autolabeling [9] (which even uses ground truth boxes) in the AP BEV metric, whilst also achieving a decent 2D accuracy (AP 2D). Figure 4 visualizes our improving pseudo-labels of the train split across the three iterations, while Figure 5 depicts the inference results on the validation set achieved by MonoFlex trained on our pseudo-labels.

Iteration	AP 2D %			AP BEV %		
	Easy	Mod	Hard	Easy	Mod	Hard
1	84.5	63.2	56.0	66.7	45.0	37.9
2	91.5	67.3	57.6	87.2	60.5	50.8
3	91.9	69.8	60.1	89.9	63.1	53.4
Autolabeling [9]	Ground truth boxes			77.8	59.7	N/A

TABLE I

EVALUATION OF THE PSEUDO LABELS GENERATED DURING THE THREE ITERATIONS THAT WE PERFORM ON THE TRAIN SPLIT

D. Comparison with state-of-the-art

After the end of the third iteration, we evaluate our model on the validation split to compare with the state-of-the-art. Our results from Table II demonstrate the ability of our pseudo-labels to achieve competitive results without using any supervision, even when only training on the train split.

As we can learn without labels, we can easily make use of a huge amount of data to further boost our performance. To demonstrate the usefulness of this, we generate pseudo-labels for KITTI Raw sequences, which are video sequences acquired with synchronized lidar point clouds but without any labeling. After discarding videos overlapping with the train or validation split, we generate pseudo-labels for 20 sequences and add them to the labels pool created for the train split. We again finetune the model trained on synthetic data on this larger set of sequences, using as supervision the generated pseudo-labels, for three iterations. Results reported in Table II demonstrate the capacity of our pipeline to improve when more data is available outperforming other unsupervised methods, and matching performance obtained by recently proposed fully supervised methods (such as [20]), underlining the advantage of unsupervised approaches which require no labeling efforts.

V. ABLATION STUDY

We first want to demonstrate the impact of our main contribution, being the use of temporal information. To this end, we test our pseudo-label generation without using any temporal prior, but instead solely employing the Chamfer distance for self-supervision. In Table III, we demonstrate the respective results for pseudo-labels generated by our pipeline, and pseudo-labels generated by the Chamfer baseline, and compare both with the initial model estimates. Our proposed temporal consistency proves to be less sensitive to noisy initial estimates, and maintains a considerable advantage both

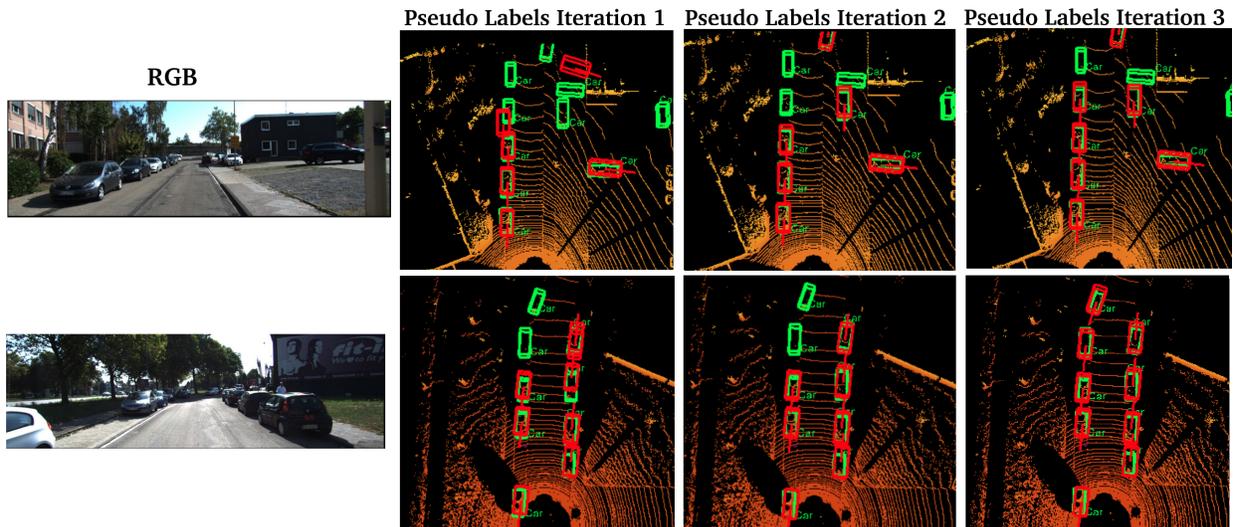


Fig. 4. Pseudo-Labels (red boxes) qualitative assessment against the ground truth (green boxes) across different iterations (best viewed in color)

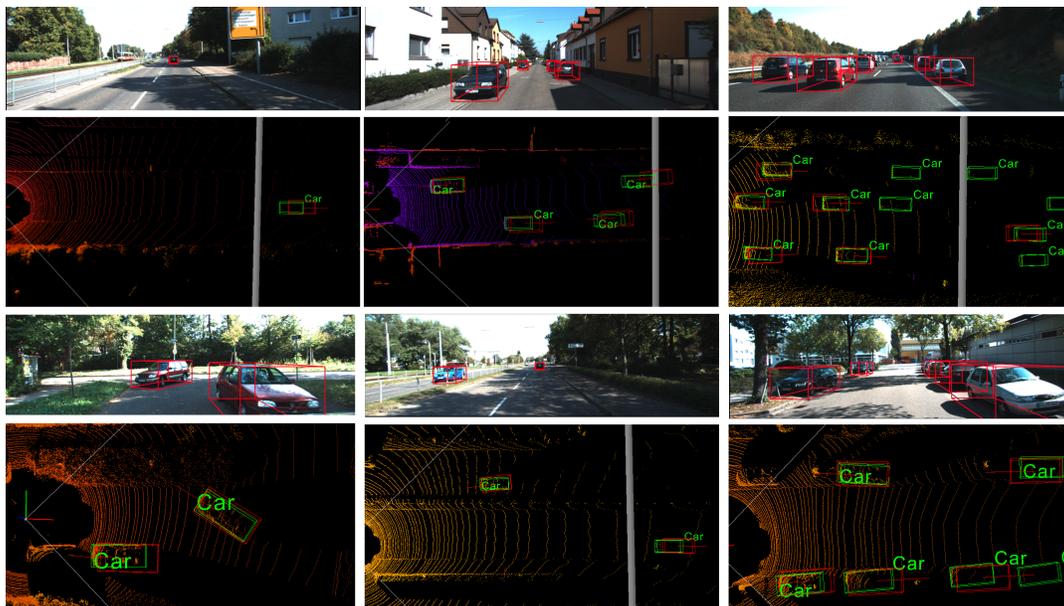


Fig. 5. Examples of detections (red boxes) generated by MonoFlex trained on our pseudo-labels (best viewed in color)

in AP3D (Easy 45.45% compared to 38.61) and AP BEV (Easy 89.90% compared to 87.23).

A. Motion trajectory modeling

Modeling the motion of other cars plays an important role in generating accurate pseudo-labels. Noisy initial estimates can complicate the optimization procedure and converge to sub-optimal poses. Using priors on plausible movements of vehicles can potentially alleviate this issue and suppresses noise caused by motion blur or occlusions. We test different motion trajectory models, with different complexities and modeling power. First, we apply robust linear fitting (using RANSAC) to the entire trajectory, which assumes that the entire motion is linear within the tracking period. While this model applies to many situations in straight roads,

it clearly fails to adapt to other cases such as turns and crossings. As a more complex model, we also consider splines, piece-wise polynomial functions, with an adaptive degree selection (based on MSE between the candidate fit and the original data). Such models, however, require, in addition to the polynomial degree, an adequate smoothing parameter selection (related to the number of knots), which is hard to tune given different motion patterns. Our experiment shows that simple models (like the linear one) are still expressive when applied on local segments of the trajectory, while requiring less parameter tuning.

In Table IV, we generate the first iteration of pseudo-labels of the train split using different motion models, and we evaluate the moving cars against the ground-truth (since these are of main interest for this ablation).

Method	Images	$AP_{BEV} / AP_{3D} (AP_{R11} @ 0.5 \text{ IoU})$			$AP_{BEV} / AP_{3D} (AP_{R40} @ 0.5 \text{ IoU})$		
		Easy	Mod	Hard	Easy	Mod	Hard
Supervised							
Deep3DDBBox [17]	trainsplit	30.02/27.04	23.77/20.55	18.83/15.88	-	-	-
Mono3D [15]	trainsplit	30.50/25.19	22.39/18.20	19.16/15.52	-	-	-
M3D-RPN [20]	trainsplit	55.37/48.96	42.49/39.57	35.29/33.01	53.35/48.53	39.60/35.94	31.76/28.59
MonoGRNet [51]	trainsplit	-	-	-	48.53/47.59	35.94/32.28	28.59/25.50
LPCG-M3D-RPN [36]	trainsplit	67.66/61.75	52.27/49.51	46.65/ 44.70	-	-	-
MonoFlex [18]	trainsplit	68.62/65.33	51.61/ 49.54	49.73/43.04	67.08/61.66	50.54/46.98	45.78/41.38
Unsupervised							
MonoDIS- SDFLabel [9]	trainsplit	51.10/32.90	34.50/22.10	-	-	-	-
Ours w/ MonoFlex	trainsplit	52.43/36.71	37.55/26.74	31.21/22.09	48.59/32.10	31.45/21.12	24.40/15.92
MonoDR [33]	-	51.13/45.76	37.29/32.31	30.20/26.19	48.53/43.37	33.90/29.50	25.85/22.72
LPCG-M3D-RPN[36]	Raw data	52.06/47.58	35.37/29.06	28.61/26.58	-	-	-
Ours w/ MonoFlex	Raw data	<u>63.94/51.90</u>	<u>42.29/33.24</u>	<u>35.31/30.39</u>	<u>59.63/46.95</u>	<u>38.31/30.08</u>	<u>30.62/24.41</u>

TABLE II

AVERAGE PRECISION ON KITTI VALIDATION SET OF OUR METHOD AND OTHER SUPERVISED AND UNSUPERVISED METHODS (BEST IS IN BOLD, BEST UNSUPERVISED IS UNDERLINED)

	AP 3D % @0.5IoU			AP BEV % @0.5IoU		
	Easy	Mod	Hard	Easy	Mod	Hard
Initials	15.12	9.91	8.18	35.51	21.95	19.70
1st Iteration						
baseline	20.09	11.72	9.62	54.40	33.00	28.26
Ours	33.82	18.78	16.90	66.70	45.02	37.92
2nd Iteration						
baseline	42.62	25.07	20.28	84.64	55.46	48.23
Ours	45.13	29.44	24.54	87.19	60.51	50.82
3rd Iteration						
baseline	38.61	22.90	20.24	87.23	60.54	50.86
Ours	45.45	29.38	25.94	89.90	63.12	53.40

TABLE III

QUANTITATIVE EVALUATION OF THE PSEUDO-LABELS GENERATED FOR THE TRAIN SPLIT USING THE CHAMFER BASELINE AND OUR PROPOSED TEMPORAL CONSISTENCY

	AP 3D % @0.5IoU			AP BEV % @0.5IoU		
	Easy	Mod	Hard	Easy	Mod	Hard
Moving						
linear	32.65	21.08	19.54	60.68	45.80	43.36
ada. spline	37.92	23.96	22.22	68.23	50.71	45.90
p.w linear	41.40	26.41	24.58	68.25	50.92	48.47

TABLE IV

EVALUATION OF MOVING CARS IN THE PSEUDO-LABELS GENERATED USING DIFFERENT MOTION MODELS: GLOBAL ROBUST LINEAR FIT (USING RANSAC)- FIRST ROW, ADAPTIVE SPLINE FITTING- SECOND ROW, PIECE-WISE LINEAR FIT (USING RANSAC)- THIRD ROW

B. Motion state classification

The proper use of the temporal prior requires an accurate understanding of the state of the object (in motion, static). By discarding the temporal context of short lived tracklets, we avoid wrongly assigned motion classes and maintain high precision for both motion states. In Table V, we compare our tracklets with the ground-truth (of the trainsplit) and measure the precision of our motion classification module across

different iterations. We observe a consistent improvement of both the precision and the support, which is achieved by an increasing detection accuracy. Additionally, this is reflected in the support of short tracks due to better matching.

To further analyze the contribution of each of the temporal priors (static and moving objects), we perform an evaluation of our pseudo-labels considering only one motion class at a time and compare with the baseline approach. Table VI demonstrates the effectiveness of the temporal consistency in both of the motion states, with a stronger improvement in the case of static cars.

Iteration	Static		Moving		undecided
	P % \uparrow	#boxes \uparrow	P % \uparrow	#boxes \uparrow	#boxes \downarrow
1st	93.97	3291	96.37	3257	175
2nd	99.11	3157	91.85	4129	118
3rd	99.68	3730	96.40	4369	78

TABLE V

CLASSIFICATION OF OBJECT MOTION STATE IN DIFFERENT ITERATIONS OF PSEUDO-LABELS.

	AP 3D % @0.5IoU			AP BEV % @0.5IoU		
	Easy	Mod	Hard	Easy	Mod	Hard
Static						
baseline	7.79	5.31	4.32	39.78	20.43	17.95
Ours	23.28	14.84	12.57	57.76	35.89	29.29
Moving						
baseline	31.63	20.19	18.64	60.79	45.83	43.38
Ours	41.40	26.41	24.58	68.25	50.92	48.47

TABLE VI

EVALUATION RESULTS BY MOTION CLASS OF BOTH OUR PROPOSED METHOD AND THE BASELINE (COMPUTED ON THE TRAINSPLIT)

VI. CONCLUSIONS

We have proposed a self-supervised framework to train monocular 3D object detection methods without the use of ground truth labels. Starting off by an initial training

on simple synthetic data, we then use the noisy initial pose and shape estimates to formulate our self-supervised loss terms. Assuming the availability of lidar point clouds during training, and harnessing temporal prior in video sequences, we optimize the model estimates and derive high-quality pseudo-labels. Using these in a finetuning procedure, we obtain results competitive with the state-of-the-art and demonstrate the potential of using large-scale un-annotated data in pushing the model performance. In the future, the pipeline can be extended to support less accurate depth data coming from stereo-pairs or monocular depth estimation.

REFERENCES

- [1] F. Manhardt, W. Kehl, and A. Gaidon, "Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape," in *CVPR*, 2019.
- [2] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *CVPR*, 2018.
- [3] P. Wang, F. Manhardt, L. Minciullo, L. Garattoni, S. Meier, N. Navab, and B. Busam, "Demograsp: Few-shot learning for robotic grasping with human demonstration," in *IROS*. IEEE, 2021.
- [4] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *CVPR*, 2017.
- [5] N. Komodakis and S. Gidaris, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.
- [6] T. Afouras, A. Owens, J. S. Chung, and A. Zisserman, "Self-supervised learning of audio-visual objects from video," in *ECCV*, 2020.
- [7] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6d object pose estimation," in *ECCV*. Springer, 2020.
- [8] G. Wang, F. Manhardt, X. Liu, X. Ji, and F. Tombari, "Occlusion-aware self-supervised monocular 6d object pose estimation," *IEEE PAMI*, 2021.
- [9] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3d objects with differentiable rendering of sdf shape priors," in *CVPR*, 2020.
- [10] L. Koestler, N. Yang, R. Wang, and D. Cremers, "Learning monocular 3d vehicle detection without 3d bounding box labels," *Pattern Recognition*, vol. 12544, p. 116, 2020.
- [11] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *ICCV*, 2019.
- [12] S. Pillai, R. Ambruş, and A. Gaidon, "Superdepth: Self-supervised, super-resolved monocular depth estimation," in *ICRA*, 2019.
- [13] X. Ma, W. Ouyang, A. Simonelli, and E. Ricci, "3d object detection from images for autonomous driving: A survey," *arXiv preprint arXiv:2202.02980*, 2022.
- [14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* 28, 2015.
- [15] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *CVPR*, 2016.
- [16] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," *NeurIPS*, vol. 28, 2015.
- [17] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *CVPR*, 2017.
- [18] Y. Zhang, J. Lu, and J. Zhou, "Objects are different: Flexible monocular 3d object detection," in *CVPR*, 2021.
- [19] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kotschieder, "Disentangling monocular 3d object detection," in *ICCV*, 2019.
- [20] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *ICCV*, 2019.
- [21] A. Simonelli, S. R. Bulo, L. Porzi, E. Ricci, and P. Kotschieder, "Towards generalization across depth for monocular 3d object detection," in *ECCV*. Springer, 2020.
- [22] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *ECCV*. Springer, 2020.
- [23] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR*, 2019.
- [24] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *ECCV*. Springer, 2020.
- [25] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *CVPR*, 2019.
- [26] N. Komodakis and S. Gidaris, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.
- [27] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*. Springer, 2016.
- [28] X. Wang, A. Jabri, and A. A. Efros, "Learning correspondence from the cycle-consistency of time," in *CVPR*, 2019.
- [29] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.
- [30] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *ICCV*, 2019.
- [31] P. Liu, M. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," in *CVPR*, 2019.
- [32] H. Mittal, B. Okorn, and D. Held, "Just go with the flow: Self-supervised scene flow estimation," in *CVPR*, 2020.
- [33] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Monocular differentiable rendering for self-supervised 3d object detection," in *ECCV*, 2020.
- [34] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.
- [35] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Differentiable rendering: A survey," *arXiv preprint arXiv:2006.12057*, 2020.
- [36] L. Peng, F. Liu, Z. Yu, S. Yan, D. Deng, and D. Cai, "Lidar point cloud guided monocular 3d object detection," *arXiv preprint arXiv:2104.09035*, 2021.
- [37] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *CVPR*, 2021.
- [38] B. Caine, R. Roelofs, V. Vasudevan, J. Ngiam, Y. Chai, Z. Chen, and J. Shlens, "Pseudo-labeling for scalable 3d object detection," *arXiv preprint arXiv:2103.02093*, 2021.
- [39] J. Hur and S. Roth, "Self-supervised multi-frame monocular scene flow," in *CVPR*, 2021.
- [40] S. Varghese, S. Gujamagadi, M. Klingner, N. Kapoor, A. Bar, J. D. Schneider, K. Maag, P. Schlicht, F. Huger, and T. Fingscheidt, "An unsupervised temporal consistency (tc) loss to improve the performance of semantic segmentation networks," in *CVPR*, 2021.
- [41] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova, "4d-net for learned multi-modal alignment," in *ICCV*, 2021.
- [42] C. Saltori, S. Lathuilière, N. Sebe, E. Ricci, and F. Galasso, "Sf-uda 3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection," in *3DV*, 2020.
- [43] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [44] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017.
- [45] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.
- [46] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014.
- [47] X. Weng and K. Kitani, "A baseline for 3d multi-object tracking," *arXiv preprint arXiv:1907.03961*, 2019.
- [48] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals using stereo imagery for accurate object class detection," *IEEE PAMI*, vol. 40, no. 5, pp. 1259–1272, 2017.
- [49] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.
- [50] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *CVPR*, 2018.
- [51] Z. Qin, J. Wang, and Y. Lu, "Monogrnnet: A geometric reasoning network for monocular 3d object localization," in *AAAI*, 2019.