`SeMoLi`: What Moves Together Belongs Together
Supplementary Material

Jenny Seidenschwarz[1,2] Aljoša Ošep[2] Franceso Ferroni[2] Simon Lucey[3] Laura Leal-Taixé[2]

[1]Technical University of Munich [2]NVIDIA [3]University of Adelaide

Abstract

In this appendix we first discuss limitations as well as computational costs in Sec. 1, detail the datasets we evaluate `SeMoLi` on in Sec. 2, as well as the computation of our SegIoU metric in Sec. 3. We then detail our point cloud filtering as well as the impact on `SeMoLi`'s performance in Sec. 4 and give a deeper analysis of our graph construction approaches in Sec. 5. Afterwards, we give deeper insights on our cluster post-processing with correlation clustering Sec. 6, bounding box extraction Sec. 7, as well as bounding box inflation Sec. 8. We compare the latter with the registration introduced in [8] in Sec. 9 and give details on our implementation of [8] in Sec. 10. Then, we detail the training of `SeMoLi` in Sec. 11 and PointPillars (PP) in Sec. 12, followed by a deeper discussion on the performance PP trained with different pseudo-labels in Sec. 13.

1. Limitations and Computational Cost.

Since `SeMoLi` relies on pre-processed point clouds and predicted motion as input, the quality of both influences the final performance (see Sec. 4 of supplementary and Tab. 3 of the main paper). For `SeMoLi` itself we observe two main limitations. (i) Our focus on close, moving PL introduces a bias to PointPillars towards regions where typically moving objects are found which leads to a lower recall (see Tab. 6 All vs. Moving) since during training correct predictions for which no pseudo-labels exist are penalized. This could be addressed by data augmentation. (ii) `SeMoLi` is based on learning to group points, and deriving enclosing modal boxes. Hence, this is different to the amodal ground truth bounding boxes of the datasets under investigation and leads to PointPillars predicting too tight bounding boxes. This issue is especially notable for strict evaluation thresholds (see @0.7 Tab. 6 in the main paper) and large objects (see Tab. 4 and 5 in the main paper). It could be addressed by utilizing sequential data to obtain amodal estimates as in [8, 11]. However, we observed that especially the approach of [8] suffers from point cloud pre-processing and the quality of the predicted motion (see Sec. 9 of supplementary).

Computational costs: Especially per-timestamp motion optimization (∼7 min, 12GB GPU) increases computational costs. Recently, faster self-supervised methods like [6] allow for speed-ups. To train `SeMoLi`, a single 32GB GPU suffices due to the enormous PC reduction during filtering. For PointPillars we utilize 8x16GB GPUs.

2. Dataset details

As intoduced in Sec. 4.1 of the main paper, we ablate and evaluate our method using the large-scale Waymo Open dataset [9] and only utilize Lidar sensory data. The dataset consists of 6.4 hrs of calibrated image and Lidar sensory data, recorded at 10 Hz in Phoenix, San Francisco, and Mountain View.

Additionally, we assess the generalization of our method on Argoverse2 dataset [10] recorded at 10 Hz in six U.S. cities, namely Austin, Detroit, Miami, Pittsburgh, Palo Alto, and Washington, D.C. Argoverse2 provides significantly finer-grained semantic labels (30 object classes). In Tab. 5 of the main paper we show the performance on those classes that occur as moving objects in our `val_gnn` dataset of Argoverse2.

Waymo Open dataset and Argoverse2 were recorded with different types of (proprietary) Lidar sensors. While Argoverse2 dataset was recorded with two roof-mounted VLP-32C Lidar sensors, *i.e.*, 64 beams in total, Waymo Open dataset was recorded with five Lidar sensors – one mid-range Lidar (top) and four short-range Lidar sensors (front, side left, side right, and rear). This leads to a denser representation in close-range. We show in our experiments, that despite the representations being different `SeMoLi` is able to generalize between the datasets.

3. SegIoU Computation

To compute our SegIoU metrics, in each frame we utilize the extracted pseudo-label bounding boxes $b_c \in B_c$ as well

* Correspondence to j.seidenschwarz@tum.de.

as the ground truth bounding boxes $g_c \in G_c$ to find their interior points in the filtered point cloud $\tilde{P} \in R^{M \times 3}$. This leaves us with binary instance segmentation masks $I_G \in R^{M \times |G_c|}$ and $I_B \in R^{M \times |B_c|}$ for ground truth and pseudo-label bounding boxes, respectively, where the masks indicate point membership to a bounding box. Finally, we compute the intersection over union between I_B and I_G to obtain out SegIoU matrix $S \in R^{|B_c| \times |G_c|}$.

4. Point Cloud Filtering

In this section, we specify the point cloud filtering we apply before predicting point trajectories introduced in Sec. 3.1.1 of the main paper. We then discuss its impact on SeMoLi's performance as well as the amodal and modal oracle performance we can obtain with the filtered point cloud.

Details on Point Cloud Filtering. We follow a similar filtering pipeline as [8], where we compute an approximate velocity magnitude for each point via Chamfer Distance between a point cloud at timestamp t and $t+4$ where the point clouds are ego-motion-compensated. We then filter points with $|v_i^t| < 0.2m/s$. We also remove ground points utilizing a ground plane fitting algorithm [5] and only retain points within a range of $80m$ around the ego vehicle and lower than $4m$. On Waymo Open dataset the average full point cloud has 175000 points while the average filtered point cloud has 10500 points.

Impact on SeMoLi's Performance. The point cloud filtering step is delicate: it impacts the quality of the estimated point trajectories τ_i – the more stationary points we filter, the better the point trajectories. On the other hand, by removing too many non-stationary points we may filter out several moving instances and hamper recall. The quality of filtering static points can be measured by precision and recall, where a true positive is defined as a static point that we successfully remove, while a false positive is a moving point that we accidentially remove. As in [8] we define static points as points with a ground truth velocity $< 1m/s$. Despite closely following the point cloud filtering in [8], with additional feedback of the authors, we were not able to achieve the recall and precision given in [8] – they report precision and recall of 97.2% and 62.2% while we achieve 88.3% and 97.5%. This leaves us with a more noisy point cloud than used in [8], where more moving points are filtered out and more static points kept.

Amodal Oracle Performance of SeMoLi In Tab. 1 we discuss the number of *static* and *moving* ground truth bounding boxes before (*original*) and after (*filtered*) point cloud filtering with $\geq x_f$ interior points in the rectangular region of $100x40m$ around the ego vehicle on the val_pseudo dataset. We observe that a certain amount of ground truth bounding boxes already have no interior points even in the original point cloud (see difference between $x_f = 0$ and $x_f = 1$). After filtering, the percentage of static objects is significantly reduced, at the cost of losing some moving objects. However, we retain a high number of boxes of moving objects, *e.g.*, 90% with at least 10 points, and 67% with at least 50. This amount of moving objects present in the filtered point cloud determines the *amodal* upper performance bound, *i.e.*, the quality upper bound of our pseudo-labels.

# Interior	*static objects* original	*static objects* filtered	*moving objects* original	*moving objects* filtered
x_f = 0	100	100	100	100
x_f = 1	97.8	71.1	96.3	95.4
x_f = 10	94.9	21.8	92.4	90.4
x_f = 30	89.5	9.7	84.3	79.5
x_f = 50	82.6	6.3	76.2	67.7

Table 1. **Point cloud filtering.** In the pre-filtering step, we remove points that appear static. We report the percentage of labeled bounding boxes that have more $\geq x_f$ interior before and after filtering. A few observations: (i) even when not filtering the point cloud, a certain percentage of boxes have no interior points; those are probably occluded and extrapolated. (ii) filtering causes a big drop for static objects, as expected. (iii) after filtering we retain a high number of boxes of moving objects: 90% with at least 10 points, and 67% with at least 50. Those are those that we can obtain in the pseudo-labeling process.

Modal Oracle Performance of SeMoLi. However, our pseudo-labels are *modal*, *i.e.*, the extent of the bounding boxes is defined by the smallest enclosing cuboid around the points present in the filtered point cloud. To obtain a modal upper bound, we compute an *Oracle* performance where we assign each point its ground truth identity and extract bounding boxes given those ground truth identities. This gives us the performance we would obtain if SeMoLi would be able to segment all points correctly. In Tab. 2 we report this oracle performance utilizing SegIoU- and 3DIoU-based evaluation if we discard segments with $\leq x_f$ interior points. We see that SegIoU performance corresponds to the amodal upper bound – the slight differences are caused by our bounding box extraction. 3DIoU-based performance shows that even if SeMoLi would segment a given point cloud perfectly, comparing our modal bounding boxes to amodal ground truth bounding boxes would lead to a heavy drop in performance compared to amodal oracle performance. We further observe, the larger x_f gets, the lower the recall but the higher the precision and F1 score. This means, that if SeMoLi would wrongly classify edges of clusters with a small amount of interior points as negative classes, 3DIoU-based performance would increase. Hence, SegIoU is better suited than 3DIoU to evaluate the clustering quality of SeMoLi and we only use 3DIoU-based evaluation to evaluate the final pseudo-label quality and PP performance.

Method	*SegIoU* Pr 0.4	Re 0.4	F1 0.4	*3DIoU* Pr 0.4	Re 0.4	F1 0.4
$x_f = 2$	95.1	94.8	95.0	40.6	40.6	40.6
$x_f = 30$	97.9	81.2	88.8	48.8	40.4	44.2
$x_f = 50$	98.1	68.8	80.8	56.5	39.6	46.6

Table 2. **SeMoLi [10] ablation modal upper bound:** We report upper bound via **oracle**, *i.e.*, the achievable performance with segments with at least x_f interior points from GT labels.

Method	Pr 0.7	Re 0.7	F1 0.7	Pr 0.4	Re 0.4	F1 0.4
Oracle Velocity kNN	35.7	67.1	46.6	39.4	74.1	51.4
Oracle Position kNN	85.0	87.9	86.4	89.4	92.5	90.9
Velocity kNN	35.9	41.7	38.6	45.6	52.9	49.0
Position kNN	63.3	53.0	57.7	77.9	65.1	70.9

Table 3. **SeMoLi [10] ablation graph construction (SegIoU):** We discuss the oracle performance of SeMoLi utilizing different strategies on **graph construction** as well as the performance when utilizing those strategies for training.

5. Deeper analysis of our graph construction

In the main paper, we report the oracle performance, *i.e.*, the maximum achievable performance given an underlying graph construction strategy when utilizing a kNN graph in Tab. 1. We showed, that the oracle performance of position-based kNN-graph construction is significantly higher than the oracle performance when utilizing velocity. In this section, we discuss those results more in detail as well as the performance when training SeMoLi on the different graph construction approaches.

Oracle Graph Construction Performance. For velocity-based graph construction, closest points can be situated anywhere in space (see also Fig 1b) which leads to several negative edges being added to the graph. This is mirrored in the low performance in recall, *i.e.*, many false negative segments, as well as precision, *i.e.*, many false positive segments as reported in Tab. 1 in the main paper. Utilizing position as an initial inductive bias assures that close-by points are considered first as belonging to the same object (see Fig 1a). Hence, computing the oracle performance leads to significantly better performance.

SeMoLi's Graph Construction Performance. Remarkably, utilizing velocity as a graph construction method for SeMoLi leads to a higher precision than when computing the oracle performance (see Tab. 3). This seems counterintuitive since the oracle performance should lead to the best possible performance achievable. However, the oracle approach leads to scattered objects and more false positive pseudo-labels due to the edge hypothesis' connecting distant points and not connecting close points. During the learning process, SeMoLi fails to correctly classify edges of heavily scattered objects leading to them being discarded due to being singleton points or being filtered out by the size

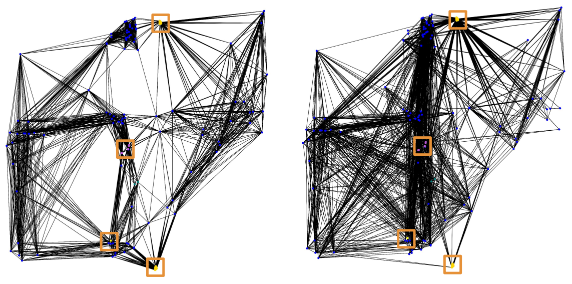
(a) Position-based construction. (b) Velocity-based construction.

Figure 1. **Comparison of different graph construction approaches:** We compare three different graph construction approaches as initial hypothesis for SeMoLi: position-based, velocity-based and a combination of both, where we first build a graph based on position and then cut edges if node velocities are highly different. Blue points represent nodes *i.e.* points in the point cloud, edges the initial hypothesis of connection to be refined by our GNN. Position-based graph construction utilizes the inductive-bias of proximity, velocity-based hypothesis yields edges spanning the entire scene since points can potentially have a similar velocity if they are are far in space.

filter (any dimension $< 0.1m$). This in turn leads to less false positive predictions hence a higher precision but also a lower recall. Since (i) this behavior is not desirable during our learning process and (ii) the performance is still significantly worse than when utilizing position-based graph construction, we utilized the latter.

6. Cluster Postprocessing

As introduced in Sec. 3.1.2 in the main paper, we obtain edge scores $\tilde{h}_{i,j}^{(L)}$ for our edges from a binary classifier on the edge features after the last layer. We cut negative edges, *i.e.*, edges for which we predict $\tilde{h}_{i,j}^{(L)} < 0.5$. Then, we remove singleton nodes and apply correlation clustering on the remaining node and edge set utilizing our edge scores as edge weights. For correlation clustering, we utilize the implementation of [1] with default settings.

7. Bounding Box Extraction

As mentioned in Sec. 3.1.1 of the main paper, we discard singleton nodes that are left without edges after our cluster postprocessing and correlation clustering. Additionally to the bounding box extraction defined in Sec. 3.2 of the main paper, we discard bounding boxes where either of the dimensions is $< 0.1m$.

8. Bounding Box Inflation

SeMoLi generates compact, modal bounding boxes that are tightly fitted around the clustered points by extracting to smallest enclosing cuboid. Due to point cloud filtering

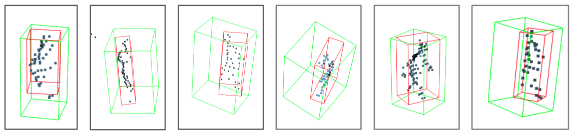
Figure 2. **Visualizations of Pedestrian Boudning Boxes:** We show visualizations of pedestrian clusters in our filtered point cloud, our extracted bounding boxes (red) as well as ground truth bounding boxes (green). We can see, that `SeMoLi` clusters points correctly, but the extracted bounding boxes are significantly smaller than their corresponding ground truth bounding boxes.

and possibly wrongly filtering moving points, pseudo labels are more compact when extracted on the filtered point cloud. However, ground truth bounding boxes are amodal representations of the objects and typically more loosely placed around the object. This leads to the fact that despite `SeMoLi` clustering points correctly, our extracted pseudo-labels are not considered as true positives. Particularly for pedestrians, we observed our pseudo-labels being perfectly fit around the points but being considered as false positive detections. We show several visualizations in Fig. 2 where green represents ground truth bounding boxes and red our pseudo-labels fitted around a point cluster of the filtered point cloud.

To enhance our pseudo-labels to correspond to the underlying ground truth data to train the detector, we inflate our bounding boxes to have a minimum size of $(1m, 1m, 2m)$ for Waymo Open dataset and $(0.75m, 0.75m, 1.75m)$ for Argoverse2. We show the impact on the performance on Waymo Open Dataset in Tab. 2 of the main paper. The performance evaluation based on SegIoU does not change significantly indicating that `SeMoLi` clusters points correctly. When evaluating on 3DIoU, our simple bounding box inflation leads to a drastic performance increase indicating the better correspondence.

9. Registration

For a fair comparison with our baseline DBSCAN++† [8], we also implemented the registration [3] algorithm provided in the paper. The authors state that they track bounding boxes with a BEV IoU threshold of 0.1, *i.e.*, a small overlap is enough for two bounding boxes to be matched together. Then they register bounding boxes over all tracks to obtain a more complete representation of the underlying object. Contrary to the performance reported in [8], adding registration to our pipeline worsened our performance (see Tab. 4). We assume that this is due to our more noisy point cloud filtering. As soon as one detection in a track is noisy, due to the registration all bounding boxes in a track will be noisy. We also show the performance of our best registration setting, where we: (i) utilize a threshold of 0.5, (ii) only apply constrained ICP to obtain transformations between detections but propagate the dimensions of the most dense point cloud, (iii) only apply constrained ICP on clusters with more than 50 points and tracks with a length ≥ 5. However this also does not improve our performance. Hence, we rely on bounding box inflation as our only post-processing step due to its simplicity and high efectiveness.

	3DIoU			*SegIoU*		
	Pr 0.4	Re 0.4	F1 0.4	Pr 0.4	Re 0.4	F1 0.4
Initial	33.2	27.8	30.3	77.9	65.1	70.9
Registration as in [8]	2.1	0.9	1.2	20.5	8.6	12.2
Our best Registration	21.0	15.0	17.5	67.8	48.4	56.5
Inflated	59.1	48.3	53.1	80.9	66.2	72.8

Table 4. **Registration:** We implement constrained ICP [3] for bounding box amodalization as in [8]. Due to our more noisy point cloud and trajectory predictions we are not able to improve our performance. However, simple bounding box inflation leads to a significant performance improvement.

	Pr 0.7	Re 0.7	F1 0.7	Pr 0.4	Re 0.4	F1 0.4
Re-implementation as in [8]	0.4	5.2	0.7	2.8	37.7	5.3
Adapted re-implementation	1.6	6.2	2.5	10.2	40.3	16.2

Table 5. **Comparison of DBSCAN++† with and without minimum number of points per cluster:** We compare the performance of the re-implementation of our baseline DBSCAN++† as described in [8] where clusters are allowed to contain only one point. In our adapted re-implementation we define the minimum number of points per cluster for position- as well as scene flow-based clusterings as 10 and the minimum number of samples of their intersection as 20.

10. Details DBSCAN and DBSCAN++ re-implementation

For the vanilla DBSCAN baseline based on solely position, we set $\epsilon = 1$ and the minimum number of samples per cluster to 10. As introduced in the main paper, we also compare our approach to the DBSCAN++ approach introduced in [8]. Unfortunately, we were not able to access the implementation as well as the scene flow used and, therefore, report our best re-implementation. To be specific, we apply a DBSCAN clustering on the spatial positions of points in the point cloud and on the scene flow that we extract from our predicted trajectories. Then we utilize the intersection of both to obtain the final clusters. We follow [8] and set $\epsilon_{pos} = 1$ and $\epsilon_{flow} = 0.1$. To filter out noise, we define the minimum number of samples per cluster to 10 for both and to 20 for the intersection. Allowing any number of interior points, *i.e.*, also clusters with a single interior point leads to a significant drop in precision and F1 score – from 10.2 to 2.8 and from 16.2 to 5.3 for $T = 0.4$ on Waymo Open dataset (see Tab. 5). Since no minimum number of samples is mentioned in [8], we assume that either the point cloud filtering or the scene flow prediction were cleaner to begin with.

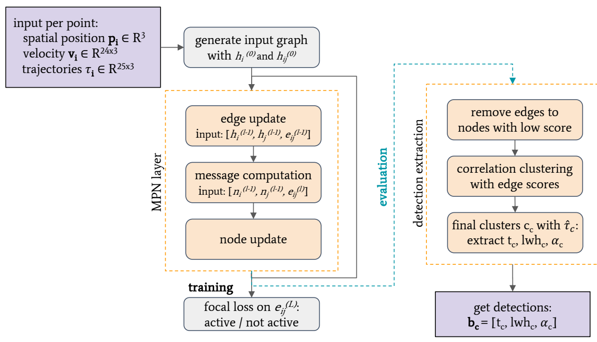
input per point:
spatial position $\mathbf{p_i} \in R^3$
velocity $\mathbf{v_i} \in R^{24x3}$
trajectories $\tau_\mathbf{i} \in R^{25x3}$
generate input graph with $h_i^{(0)}$ and $h_{ij}^{(0)}$
MPN layer
edge update input: $[h_i^{(l-1)}, h_j^{(l-1)}, e_{ij}^{(l-1)}]$
message computation input: $[n_i^{(l-1)}, n_j^{(l-1)}, e_{ij}^{(l)}]$
node update
training
focal loss on $e_{ij}^{(L)}$: active / not active
evaluation
detection extraction
remove edges to nodes with low score
correlation clustering with edge scores
final clusters c_c with $\hat{\tau}_c$: extract t_c, lwh_c, α_c
get detections: $\mathbf{b_c} = [t_c, lwh_c, \alpha_c]$

Figure 3. **Overview MPN during training and evaluation:** We visualize our MPN. It takes as input points from a point cloud with their corresponding spatial positions p_i, trajectory t_i, and velocities along the trajectory v_i. We then extract initial node and edge features $h_i^{(0)}$ and $h_{ij}^{(0)}$, apply L MPN layers and for training apply focal loss on the final edge features $h_{ij}^{(L)}$. During Evaluation, we prune edges based on the final edge scores, apply correlation clustering on the remaining, and extract bounding boxes b_c with translation t_c, dimensions lwh_c, and heading in xy-direction α_c.

11. SeMoLi training.

In this section, we detail the training of SeMoLi. As supervisory signal, we define edges as positive, if the two connected nodes belong to the same moving object. Every other edge we consider as negative. This leaves us with the vast majority of edges being negative. Hence, we use the focal loss function [7], well-suited for our imbalanced binary edge classification objective. For optimization, we use Adam optimizer [4] and step learning rate schedule with step size 15 and gamma value 0.7 for 30 epochs, batch size of 4 and a base learning rate of 0.003. We apply dense supervision, *i.e.*, we apply the loss function after each layer of the MPN and to all edges.

12. PointPillars Detector Training and Anchor Bounding Boxes

For our detector training on Waymo Open dataset as well as Argoverse2, we utilize the default implementation and parameters for PointPillars training on Waymo Open dataset of [2]. To be specific, we train for 24 epochs with a base learning rate of 0.001 and a batch size of 32. We apply a combination of linear and step learning rate decay. For augmentation, we utilize random horizontal and vertical flip, global rotation and scaling as well as point shuffling. We only load every 5-*th* frame. For the Waymo Open dataset we utilize spatial position, intensity, and elongation as input while for Argoverse2 we only use the first two since elongation is not provided.

To account for different object sizes we utilize three different anchor bounding box sizes on Waymo Open dataset: $(4.75, 2.0, 1.75), (0.9, 0.85, 1.75)$ and $(1.8, 0.85, 1.75)$. Due to the more diverse classes on Argoverse2 dataset, we utilize also a more diverse set of anchor bounding boxes: $(0.75, 0.75, 0.75)$, $(1.5, 0.75, 1.5)$, $(4.5, 2, 1.5)$, $(6.0, 2.5, 2)$, $(9.0, 3.0, 3.0)$, and $(13.0, 3.0, 3.5)$. We obtain those utilizing k-means algorithm on the val_gnn dataset.

13. PP Performance with different percentages of pseudo and labeled data

In this section, we detail PointPillars (PP) performance on different percentages of labeled and pseudo-labeled data (see Tab. 6). For this we compare different versions of SeMoLi, *i.e.*, SeMoLi $_{10}$, SeMoLi $_{50}$, and SeMoLi $_{90}$ with DBSCAN++† and our baselines trained on ground truth data.

Comparison of Different Versions of SeMoLi. We first compare the PP performance obtained when utilizing the different versions of SeMoLi but the same data set split of pseudo-labeled data. To be specific, we utilize the train_detector set containing 10% of our training data (please refer to Fig. 3 of the main paper for split definition).

Evaluation on static and moving objects. In Tab. 6 we show that the overall performance of SeMoLi $_{10}$ falls behind the two other versions. This shows that more training data indeed improves the performance of SeMoLi. However, a training split larger than $x = 50\%$ does not seem to improve the overall performance further when evaluating the performance with $T = 0.4$. For the more strict threshold evaluation of $T = 0.7$, we can observe slightly favorable behavior of SeMoLi $_{90}$. Interestingly, the precision of SeMoLi $_{50}$ falls behind the other models.

Evaluation on moving objects. The same observations hold when evaluating on moving objects only with all observations being more distinct. This shows that SeMoLi only extracts moving objects.

Training without Labeled Data. We compare the performance when we train PP with pseudo-labels generated using DBSCAN++† and the different versions of SeMoLi on different amounts of pseudo-labeled data.

Evaluation on static and moving objects. Interestingly, for SeMoLi as well as for DBSCAN++† without labeled data, the best performance is achieved utilizing 50% of the pseudo-labeled data. Precision as well as recall are always similar forDBSCAN++† with a constantly low precision. This indicates that the noisy training signal leads to PP predicting a large number of false positive bounding boxes. Pseudo-labels generated by SeMoLi lead to increased precision as we increase the amount of pseudo-labeled data mirroring the high pseudo-label quality. However, recall drops which indicates that pseudo-labels generated by SeMoLi indeed focus on moving objects. The higher the ratio of moving objects PP obtains as supervisory signal the less static objects it will predict.

Evaluation on moving objects. On moving objects only, the performance of DBSCAN++† when utilizing only 10% pseudo-labeled data is significantly worse than the performances utilizing $50\%, 90\%$ or 100% which are relatively stable. This indicates that DBSCAN++† pseudo-labels in general contain many noisy pseudo labels not belonging to moving objects and that the amount of moving objects is not enough in the 10% split. Afterwards the DBSCAN++† pseudo-labels with low singal-to-noise ratio do not add to the training signal. For `SeMoLi`, we again observe a constant increase of precision the more pseudo-labeled data we utilize, indicating that the overall higher quality of the bounding boxes can be transferred to training PP.

Training with Labeled Data. Finally, we compare the performance when we train PP on a combination of labeled data and pseudo-labeled data generated using DBSCAN++† and the different versions of `SeMoLi` on different data splits.

Evaluation on static and moving objects. `SeMoLi` profits significantly more from adding labeled data to PP training with respect to precision than DBSCAN++†. This indicates that the pseudo-labels generated by DBSCAN++† hamper the prediction of precise pseudo labels compared to when using `SeMoLi`'s pseudo-labels. This hypothesis is supported by the fact that the precision using DBSCAN++† does not significantly change depending on the amount of labeled data, while `SeMoLi` shows the highest precision when using the largest amount of labeled data. Since DBSCAN++† generates noisy pseudo-labels with many predictions not belonging to moving objects, the recall is highly similar to the recall when utilizing `SeMoLi`. This leads to an overall not significantly worse AP value of DBSCAN++†.

Evaluation on moving objects. Evaluating on moving objects only leads to similar observations, except that the recall when utilizing `SeMoLi`'s pseudo-labels is higher compared to when utilizing DBSCAN++† pseudo-labels. This again indicates that indeed `SeMoLi`'s pseudo-labels focus on moving objects while DBSCAN++†'s pseudo-labels also contain many pseudo-labels not belonging to moving objects – the vast majority belonging to noise.

References

[1] Ahmed Abbas and Paul Swoboda. Rama: A rapid multicut algorithm on gpu. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8193–8202, 2022. 3

[2] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020. 5

[3] Johannes Groß, Aljoša Ošep, and Bastian Leibe. Alignnet-3d: Fast point cloud registration of partially observed objects. In *3DV*, 2019. 4

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 5

[5] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In *IROS*, 2022. 2

[6] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. In *ICCV*, 2023. 1

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5

[8] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *ECCV*, 2022. 1, 2, 4, 7

[9] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1

[10] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Adv. Neural Inform. Process. Syst.*, 2021. 1

[11] Lunjun Zhang, Anqi Joyce Yang, Yuwen Xiong, Sergio Casas, Bin Yang, Mengye Ren, and Raquel Urtasun. Towards unsupervised object detection from lidar point clouds. In *CVPR*, 2023. 1

		% Pseudo	% GT	Pr 0.7	Re 0.7	AP 0.7	mAP 0.7	Pr 0.4	Re 0.4	AP 0.4	mAP 0.4
All (Mov. + stat.)	*Ground Truth Baselines*										
	Stat. + Mov., class-specific	0	100	35.5	55.5	–	37.1	69.7	44.6	–	80.3
	Stat. + Mov., class-agnostic	0	100	31.8	41.2	36.1	–	51.2	66.5	64.7	–
	Mov-only, class agnostic	0	100	34.4	19.9	15.1	–	63.9	37.1	35.0	–
	Different versions of SeMoLi										
	SeMoLi$_{90}$	10	0	3.1	3.2	1.5	–	24.0	24.8	19.6	–
	SeMoLi$_{50}$	10	0	2.8	3.7	1.9	–	21.1	27.7	21.8	–
	SeMoLi$_{10}$	10	0	3.1	3.8	1.9	–	22.8	27.3	21.7	–
	Training with pseudo-labeled data only										
	DBSCAN++†	10	0	0.6	3.0	0.8	–	5.1	26.0	14.2	–
	DBSCAN++†	50	0	0.7	3.0	0.7	–	5.8	25.9	15.3	–
	DBSCAN++†	90	0	0.7	3.2	1.2	–	5.8	25.2	14.7	–
	DBSCAN++†	100	0	0.8	3.4	0.8	–	5.9	25.9	14.9	–
	SeMoLi$_{90}$	10	0	3.0	3.8	1.8	–	21.6	27.6	21.5	–
	SeMoLi$_{50}$	50	0	3.7	4.1	2.0	–	24.6	27.6	22.6	–
	SeMoLi$_{10}$	90	0	3.8	3.4	1.8	–	26.9	24.1	19.5	–
	Training with pseudo-labeled and labeled data										
	DBSCAN++†	10	90	7.2	35.2	31.3	–	11.4	55.2	52.2	–
	DBSCAN++†	50	50	6.9	35.1	31.2	–	10.9	54.9	51.7	–
	DBSCAN++†	90	10	7.4	35.1	31.1	–	11.5	55.0	52.0	–
	SeMoLi$_{90}$	10	90	49.0	35.8	32.4	–	70.5	51.5	50.4	–
	SeMoLi$_{50}$	50	50	29.4	36.0	32.3	–	46.9	57.4	55.0	–
	SeMoLi$_{10}$	90	10	25.4	35.4	31.8	–	40.7	56.8	54.6	–
Moving only	*Ground Truth Baselines*										
	Stat. + Mov., class-specific	0	100	30.5	34.5	–	43.2	36.4	41.1	–	85.6
	Stat. + Mov., class-agnostic	0	100	16.1	52.8	44.8	–	28.1	92.4	88.7	–
	Mov-only, class agnostic	0	100	33.9	53.7	44.3	–	57.6	91.2	89.0	–
	Different versions of SeMoLi										
	SeMoLi$_{90}$	10	0	2.9	9.6	3.4	–	19.4	64.2	54.3	–
	SeMoLi$_{50}$	10	0	2.6	11.0	4.4	–	16.1	68.3	59.2	–
	SeMoLi$_{10}$	10	0	2.8	10.6	4.5	–	17.6	67.4	58.5	–
	Training with pseudo-labeled data only										
	DBSCAN++†	10	0	0.5	8.7	2.4	–	3.0	56.4	39.7	–
	DBSCAN++†	50	0	0.6	9.2	2.1	–	3.5	58.7	44.2	–
	DBSCAN++†	90	0	0.6	9.5	1.1	–	3.6	58.4	43.5	–
	DBSCAN++†	100	0	5.9	9.7	2.4	–	3.6	58.6	43.2	–
	SeMoLi$_{90}$	10	0	2.6	10.7	4.3	–	17.6	67.7	58.5	–
	SeMoLi$_{50}$	50	0	3.4	12.1	4.8	–	19.7	69.9	62.7	–
	SeMoLi$_{10}$	90	0	3.8	10.7	4.2	–	23.3	66.0	57.5	–
	Training with pseudo-labeled and labeled data										
	DBSCAN++†	10	90	2.0	34.5	30.1	–	4.0	70.5	61.6	–
	DBSCAN++†	50	50	1.9	34.3	29.6	–	3.8	70.2	60.2	–
	DBSCAN++†	90	10	2.0	34.5	29.8	–	4.1	70.1	61.0	–
	SeMoLi$_{90}$	10	90	26.2	35.4	32.3	–	45.5	61.4	53.3	–
	SeMoLi$_{50}$	50	50	11.6	36.1	32.5	–	24.3	75.4	66.7	–
	SeMoLi$_{10}$	90	10	9.4	35.4	31.7	–	19.7	74.6	66.2	–
	DBSCAN++ [8]	100	0	–	–	–	–	–	–	–	40.4

Table 6. **Semi-supervised 3D object detection on Waymo Open Dataset:** We evaluate models on **all** (*top*) and **only moving** (*bottom*) on Waymo Open validation set. For each evaluation procedure we (i) show the ground truth baselines. Then we (ii) compare the different versions of SeMoLi utilizing the same train_detector training split. To compare SeMoLi to our baseline DBSCAN++† we (iii) show a comparison of both utilizing different percentages of pseudo-labeled data. Finally, we (iv) also show a comparison utilizing different combinations of pseudo-labeled and labeled data. % GT indicates the amount of labeled training data, % Pseudo indicates the amount of pseudo-labeled data.