Leveraging Shape Completion for 3D Siamese Tracking

Silvio Giancola*, Jesus Zarzar*, and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
{silvio.giancola,jesusalejandro.zarzartorano,bernard.ghanem}@kaust.edu.sa

Abstract

*Point clouds are challenging to process due to their sparsity, therefore autonomous vehicles rely more on appearance attributes than pure geometric features. However, 3D LIDAR perception can provide crucial information for urban navigation in challenging light or weather conditions. In this paper, we investigate the versatility of Shape Completion for 3D Object Tracking in LIDAR point clouds. We design a Siamese tracker that encodes model and candidate shapes into a compact latent representation. We regularize the encoding by enforcing the latent representation to decode into an object model shape. We observe that 3D object tracking and 3D shape completion complement each other. Learning a more meaningful latent representation shows better discriminatory capabilities, leading to improved tracking performance. We test our method on the KITTI Tracking set using car 3D bounding boxes. Our model reaches a **76.94%** Success rate and **81.38%** Precision for 3D Object Tracking, with the shape completion regularization leading to an improvement of **3%** in both metrics.*

1. Introduction

Autonomous driving is changing the way we envision human transportation. Introducing fully autonomous vehicles into our cities implies sharing the roads with existing vehicles. Thus, it is imperative for autonomous vehicles to outperform humans in the task of driving. Understanding the urban environment and the human driving process is crucial for an agent to become capable of achieving and exceeding human driving performance. Accordingly, autonomous vehicles need to outperform human perception so to cope with an unbounded set of unpredictable situations.

An autonomous vehicle adapts its driving policy by understanding its environment. Modules for Road Detection [11, 6] and Road-sign Recognition [18, 53] indicate to the car where and how to drive. Object Detection methods [8, 41] constrain the vehicle's path in order to avoid collisions while Object Tracking algorithms [58, 46] predict their motion to anticipate danger. Autonomous vehicles need to sense both appearance and geometric components of the environment to extrapolate the semantic information required for driving. RGB cameras provide both appearance and geometric information by either inferring depth from single RGB cameras [54, 63] or by stereoscopy [60, 9]. Depth and shape completion [35, 22] are commonly used to improve the limited sensing capability of RGB sensors.

Alternatively, LIDAR systems directly sense geometry in a more accurate manner. LIDAR sensors are less sensitive to light and weather conditions, so they provide more reliable information in a much larger range of driving conditions. However, LIDARs generate sparse point clouds, not readily suitable for conventional CNN processing. Most current works pre-process 3D point clouds for use in CNNs by either voxelizing the 3D space [28, 20] or by projecting point clouds into a planar space [52, 26, 10, 50]. However, these methods lose fine-grained geometric details. It is worthwhile to note that only a few works deal directly

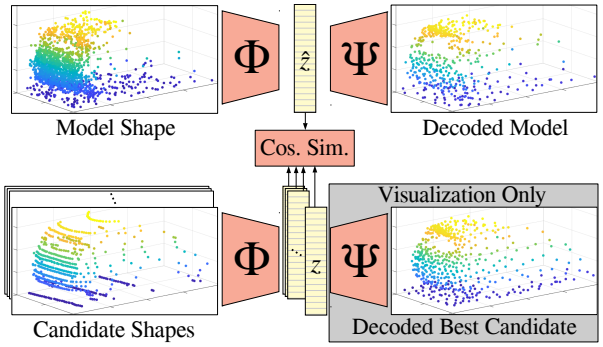

Figure 1. Our tracking model combines a Siamese network with an Auto-Encoder. The Siamese network encodes sparse 3D shapes into a latent representation z, in which shapes belonging to the same object have a high cosine similarity. By regularizing our tracker to auto-encode model shapes, we enforce that the encoder maps point clouds into meaningful representations. The effect of regularization is visualized by decoding a candidate shape.

* Both authors contributed equally to this work.

with point clouds [42, 1]. We believe appearance information is insufficient to reach better-than-human driving performance, especially in challenging driving environments.

In this work, we propose an online 3D Object Tracking method based purely on LIDAR. First, we leverage geometric features computed from sparse point clouds using the shape-completion network proposed by Achlioptas *et al*. [1]. These features are used in a Siamese network to create a latent representation in which a cosine similarity matches partial object point clouds to a model shape. Then, we regularize the encoding via an auto-encoder network to generate geometrically meaningful latent representations. We expect improved tracking performance by enriching the latent representation with semantic geometric information from the given object.

Currently, the main challenges faced in tracking relate to **(a)** similarity metrics, **(b)** model updates, and **(c)** occlusion handling. Our 3D tracker tackles these three aspects by **(a)** using Siamese networks, which have been shown to achieve state-of-the-art performance on 2D visual object tracking, adapted for processing 3D LIDAR point clouds, **(b)** leveraging the shape invariance in rigid bodies to generate a more complete model by aggregating its shape in time, and **(c)** enforcing our model to understand shape regardless of occlusions through shape completion.

Contributions: Our contributions are three-fold. **(i)** To the best of our knowledge, we propose the first 3D Siamese tracker applied to point clouds rather than images. **(ii)** We propose to regularize the Siamese network's latent space such that it resembles the latent space of a shape completion network. **(iii)** We show that regularizing our network with semantic information results in better discrimination and tracking. To ensure reproducibility and to promote future research, all source code, trained model weights, and dataset results will be made publicly available.

2. Related Work

Our work takes insights from Object Tracking based on Siamese networks, Shape Representation and Completion based on Auto-encoders, and Search Strategy.

Visual Object Tracking. Tracking is the task of identifying the trajectory of an object through time, either in images [27, 36] or in 3D space [33, 47]. Visual tracking focuses on image *patches* across consecutive frames, that represent visual attributes [27], objects [38], people [33] or vehicles [16]. The problem is commonly tackled by *tracking-by-detection*, where a *model representation* is built after the first frame and a *search space* is constructed to trade off computational costs and dense space sampling. Early works on tracking were based on Correlation Filtering [2], but current better performing methods rely on deep CNNs [23] and Siamese networks [5]. Bertinetto *et al*. [3] introduced Siamese networks for visual object tracking. They proposed a fully-convolutional Siamese network and achieved state-of-the-art performance on the VOT benchmark [27]. Recent Siamese trackers estimate boundary flows [30], use contextual structure [19], attention [56], distraction [64], semantic information [62], triplet losses [13] and region proposal networks [31] to improve tracking performance. To the best of our knowledge, our work is the first 3D adaptation of Siamese networks for 3D point cloud tracking.

3D Object Tracking. 3D Object Tracking tackles tracking from a geometric perspective. Instead of following appearance attributes using 2D bounding boxes (BBs), it computes the position of targets in the 3D world using geometry contained in 3D BBs. 3D object tracking either focuses on RGB-D information [47], by mimicking the 2D object tracking methods but with an additional depth channel [4, 32], or it focuses on purely geometric features [48, 33]. Recent work tackles 3D tracking using Bird Eye Views (BEV) of LIDAR point clouds [34, 59]. Luo *et al*. [34] input multiple BEV frames to a deep CNN to perform detection, tracking, and motion forecasting. Yang *et al*. [59] used up to 35-channel BEV frames. However, these methods lose fine-grained shape information by projecting the point cloud in the BEV. LIDARs sense the environment from a single point of view inducing self-occlusion. As compared to images where occlusion leads to noisy observation, self-occlusion in point clouds leads to incomplete observations. Despite these occlusions, 3D object tracking expects amodal 3D BBs covering the complete 3D shape. However, it assumes rigid body objects, whose 3D shape and size do not change with time.

Shape Representation. 3D shapes are complex entities to manage as they are usually sparse and lying in a continuous space, unlike images that are stored in dense and discrete matrices. Several works focus on finding efficient geometric representations [51] such as occupancy grids and TSDF cubes. They are commonly used for 3D reconstruction [40, 17] but suffer from large-scale memory inefficiency and require a space discretization which loses fine-grained details. Recent works compress 3D shapes using auto-encoders to efficiently handle geometric information [57, 55, 12]. They typically encode-decode shapes into different representations. Those auto-encoders provide a compact latent shape representation of down to 10 dimensions. Alternatively, Kundu *et al*. [29] used RGB information to decode dense 3D meshes of vehicles using Fast RCNN [43] and a differentiable Render-and-Compare loss. Achlioptas *et al*. [1] proposed to solve shape completion using an efficient auto-encoder based on PointNet [42] for point cloud to point cloud auto-encoding. They regress partial point clouds into full shapes. Alternatively, Stutz *et al*. [49] proposed an occupancy grid shape completion network based on a two-stage training process. Also, Engel-

mann *et al*. [14] proposed an energy minimization method that aligns shape and pose concurrently in stereo images.

Search Strategy. Search spaces used in visual object tracking are generally dense (exhaustive). Bertinetto *et al*. [3] used correlation filtering methods to obtain a similarity score for the whole search space. However, exhaustive search space strategies are not realistically transferable to the continuous and denser 3D space. This is commonly solved by relying on Kalman filters, Particle filters, or Gaussian mixture models to reduce the search space by providing candidate object proposals [44, 61]. At each frame, particles are sampled according to a probability distribution. Only the selected particles are observed and the probability distribution is updated according to the observation. Recently, Karkus *et al*. [24] proposed a learnable particle filter network. In our experiments, we choose to disentangle the search space and the similarity function, a common practice done in 2D tracking, by using an approximation of the exhaustive search detailed in the experiments.

3. Methodology

Herein, we propose a 3D Siamese tracker with a regularization on its latent space. The tracker is regularized to learn an encoding containing semantically meaningful information. An overview of our network is shown in Figure 1.

3.1. Siamese Tracker:

Our 3D Siamese tracker takes as input a sequence of point clouds (tracklet), in which a given object exists, along with an initial 3D BB corresponding to the position of the object in the first frame. For a frame at time t, a set of candidate shapes $\{\mathbf{x}_c^t\}$ are encoded into latent vectors $\{\mathbf{z}_c^t\}$ and compared with the latent vector $\hat{\mathbf{z}}^t$ from a model shape $\hat{\mathbf{x}}^t$. The best candidate is selected to be the object in the current frame, and the model shape $\hat{\mathbf{x}}$ is updated accordingly.

Encoding. Our encoder $\Phi(\cdot)$ is inspired from previous work on shape completion by Achlioptas *et al*. [1]. This encoder consists of 3 layers of 1D-convolutions followed by ReLU layers [39] and BN layers [21] with filter size [64, 128, K], as shown in Figure 2. The output of the last BN layer is followed by a max pooling across the points to obtain a K-dimensional latent vector. We found $K = 128$ to be a suitable size for the latent vector, as it provides the best trade-off between computational efficiency, latent space compactness, and tracking performance. The input to our network is pre-processed to have N points by randomly discarding or duplicating points, so to use mini-batches in training. We set $N = 2048$ as this is a large enough number of points to maintain a good representation of an object's shape. As compared to the network of [1], we leverage a more compact yet efficient latent space and a shallower network to reduce the size of the overall model from $\sim$ 140K to $\sim$ 25K parameters.

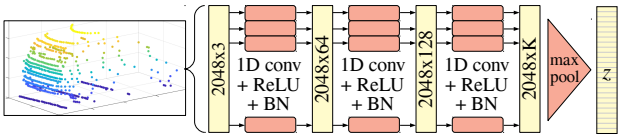

Figure 2. Our encoder takes as input a point cloud with $N = 2048$ points. Point clouds are encoded into a K-dimensional ($K = 128$) latent vector $\mathbf{z}$ using 3 layers of 1D CNN with ReLU and BN.

Similarity Metric. The encoder $\Phi(\cdot)$ calculates a latent representation from a point cloud. To compare a pair of shapes $\mathbf{x}$ and $\hat{\mathbf{x}}$, we measure the cosine similarity between their respective latent vectors $\mathbf{z} = \Phi(\mathbf{x})$ and $\hat{\mathbf{z}} = \Phi(\hat{\mathbf{x}})$ as defined in Equation (1).

$$CosSim(\mathbf{z}, \hat{\mathbf{z}}) = \frac{\mathbf{z}^\top \hat{\mathbf{z}}}{\|\mathbf{z}\|_2 \|\hat{\mathbf{z}}\|_2} \tag{1}$$

Tracking Loss. For any given frame being used in training, we designate $\mathbf{x}$ to be the tracked object's point cloud and $\hat{\mathbf{x}}$ to be the ground truth model obtained by concatenating the object's point clouds across all frames in the tracklet. We train our Siames network to regress a function of the distance between a candidate shape $\mathbf{x}$ and the model shape $\hat{\mathbf{x}}$, according to Equation (2). The poses of $\mathbf{x}$ and $\hat{\mathbf{x}}$ are parameterized by the 3 degrees of freedom of an object on a plane (t_x,t_y,α). The distance $d(\cdot,\cdot)$ is taken to be the L2-norm $\|\cdot\|_2$ of the difference between the parameterized poses. The angle α, given in degrees, is weighted with a factor of $\frac{1}{5}$ to have the same scale as t_x and t_y which are given in meters. We chose the differentiable function $\rho(\cdot)$ to be a Gaussian with $\mu = 0$, $\sigma = 1$. The purpose of $\rho(\cdot)$ is to soften the distance between positive and negative samples. $\rho(\cdot)$ takes a value of one when the distance is zero and decays as the distance increases. We then regress our similarity metric $CosSim(\cdot,\cdot)$ using an MSE loss as shown in Equation (2). Minimizing this loss for the network parameters encourages our encoder to encode partial and complete model shapes belonging to the same object into a latent space, in which they have a high cosine similarity.

$$\mathcal{L}_{tr} = \frac{1}{n} \sum_{\mathbf{x}} \Big(CosSim\big(\phi(\mathbf{x}), \phi(\hat{\mathbf{x}})\big) - \rho\big(d\,(\mathbf{x}, \hat{\mathbf{x}})\,\big)\Big)^2 \tag{2}$$

3.2. Regularization

It is important to regularize the Siamese network in order to embed into the latent representation generative properties of shape that are useful in discrimination. Such an embedding helps in generalizing to cases which aren't seen in training. Our regularization enforces the Siamese network's latent space to lie within a shape representation space. Such

representation space embeds valuable semantic characteristics defining the object to track in a compact, meaningful, and efficient representation. We provide qualitative evidence that the representation space learned by our model holds the required semantic characteristics by decoding latent representations as shown in Figure 5 Quantitative evidence is given through the improved tracking performances obtained in Table 1.

Decoding. Our decoder $\Psi(\cdot)$ is inspired by the shape completion network employed by Achlioptas *et al*. [1]. Our decoder is composed of two fully connected layers that decode a $K = 128$-dimensional latent vector $\Phi(\mathbf{x})$ into Mx3 values representing M 3D points for a reconstructed shape $\tilde{\mathbf{x}} = \Psi(\Phi(\mathbf{x}))$. We use $M = 2048$ and a hidden layer of size 1024 for a total of ~ 6.4M parameters. Alternatively, Achlioptas *et al*. [1] decoded into a denser shape of 4096 points, which requires more than twice the number of parameters in our decoder network.

Completion Loss. Adding a completion loss as a regularizer for our Siamese network boosts the network's performance by enforcing the latent representation to hold semantic information of the tracked class. While other works use the Earth Mover's Distance [45] to compare the model shape $\hat{\mathbf{x}}$ and the decoded model shape $\tilde{\mathbf{x}} = \Psi(\Phi(\hat{\mathbf{x}}))$, we use the Chamfer distance [15] (according to Equation (3)), since it is simpler to compute [1]. The tracking loss enforces encoded partial shapes to be similar to their respective encoded model, and the completion loss enforces the encoded model to hold semantic information to enable its decoding. Thus, this regularization is used to enforce the latent space learned by the Siamese network to hold meaningful shape semantic information.

$$\mathcal{L}_{comp} = \sum_{\hat{\mathbf{x}}_i \in \hat{\mathbf{x}}} \min_{\tilde{\mathbf{x}}_j \in \tilde{\mathbf{x}}} \|\hat{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2 + \sum_{\tilde{\mathbf{x}}_j \in \tilde{\mathbf{x}}} \min_{\hat{\mathbf{x}}_i \in \hat{\mathbf{x}}} \|\hat{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2 \quad (3)$$

3.3. Training

We pre-train our encoder-decoder network $\Psi(\Phi(\cdot))$ using ShapeNet [7] by taking 5997 samples from the "car" class. Our model is then fine-tuned by minimizing both tracking and completion losses. First, we crop and center points lying inside an object's ground truth BBs $\{b^t\}_{t\in[1,..,T]}$ for all frames in a given tracklet. Then, we concatenate the cropped and centered object point clouds to generate an aligned model shape $\hat{\mathbf{x}}$. Around the ground truth object point cloud at time t, we crop a set of C candidate BBs in order to create the candidate shapes $\{\mathbf{x}_c^t\}_{c\in[1,..,C]}$. The candidate BBs are sampled from a multivariate Gaussian distribution for the three planar degrees of freedom (t_X, t_Y, α) centered around the current object's ground truth BB.

Both the model shape $\hat{\mathbf{x}}$ and the set of candidate shapes $\{\mathbf{x}_c^t\}_{c\in[1,..,C]}$ are encoded into their respective latent representations $\hat{\mathbf{z}}$ and $\{\mathbf{z}_c^t\}_{c\in[1,..,C]}$. The cosine similarity between the candidates' latent representations $\{\mathbf{z}_c^t\}_{c\in[1,..,C]}$ and the model latent representation $\hat{\mathbf{z}}$ is computed according to Equation (1). The similarity scores are regressed to their relative Gaussian distance according to Equation (2).

Simultaneously, the model shape $\hat{\mathbf{x}}$ is auto-encoded into $\tilde{\mathbf{x}}$ and the Chamfer loss between $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ is minimized as in Equation (3). Note that we auto-encode the model shape $\hat{\mathbf{x}}$ into itself, instead of encoding the candidate shapes, as is done for shape completion. This enforces the latent vector to decode into the most complete car shape we have available, *i.e*. the model shape $\hat{\mathbf{x}}$.

The two losses are minimized jointly as in Equation (4), with the completion loss being weighted by λ_{comp}. We use the Adam optimizer [25] to train our model with an initial learning rate of $1e^{-4}$, β_1 of 0.9, and a batch size of 64. We reduce the learning rate at each plateau for the validation loss using a patience of 3 and a ratio of 0.1.

$$\mathcal{L} = \mathcal{L}_{tr} + \lambda_{comp}\mathcal{L}_{comp} \quad (4)$$

3.4. Testing

Since we are interested in online tracking, 3D tracklets are inferred frame-by-frame. The shape contained in the tracklet's first BB is used to initialize the model shape $\hat{\mathbf{x}}$. We track the object by looking over a set of candidate shapes in the frame at time t and comparing them to $\hat{\mathbf{x}}$ using our Siamese network. The candidate with maximum cosine similarity score is chosen to be the target object for the frame. The model shape $\hat{\mathbf{x}}$ is then updated by appending to it the chosen candidate shape. This update step makes the model sensitive to drift, as poorly selected candidates lead to a worse model which subsequently selects worse candidates. The same problem is encountered in 2D Siamese tracking, commonly solved by not updating the model at all. However, we show that our model performs better when the model is updated at each frame.

Exhaustively searching for candidates in the three degrees of freedom would incur very high computational cost. Thus, an approximation of an exhaustive search is leveraged to generate the candidate shapes. Approximating the exhaustive search allows us to assess the discriminative performance of our Siamese network by assuming the ground truth box will be included as one of the candidates as would be the case with an exhaustive search. This is a common practice in 2D trackers. Our exhaustive search is performed by generating candidates using a grid for the three degrees of freedom (t_X, t_Y, α) centered around the current ground truth. In our experiments, we compare different sampling methods such as Kalman Filters, Particle Filters, and Gaussian Mixture Models, which would be used to provide candidates for our tracker in a more realistic setting.

4. Experiments

We use the training set of the KITTI tracking dataset [16] for our experiments. It was split as follows: scenes 0-16 were used for training, scenes 17-18 for validation, and scenes 19-20 for testing. We adapt KITTI for 3D single object tracking by generating a tracklet for each instance of a car appearing in each of the scenes. Tracklets are created by concatenating the set of frames in a scene in which a given car instance appears. For each tracklet, only the first frame includes the ground truth BB. For our task, we evaluate for Single Object Tracking using the One Pass Evaluation (OPE) [27]. It defines the *overlap* as the IOU of a BB with its ground truth, and the *error* as the distance between both centers. The *Success* and the *Precision* metrics are defined using the overlap and error AUC. For our 3D object tracking purposes, we predict 3D BBs and so we estimate the precision as the AUC for 3D errors from 0 to 2m. We exhaustively generate candidates in a current frame by sampling over a grid of $[-3, 3]$m for t_x and t_y, and $[-10, 10]^o$ for α with a resolution of 1m and 10^o, respectively. The grid is centered around the current ground truth BB to approximate an exhaustive search. Experiments are run using PyTorch 0.4.1 on an NVidia GTX1080Ti with 11GB of memory, in a workstation with up to 64 CPU cores and 256GB of RAM.

4.1. Ablation Studies

We present an ablation study of our methodology in Table 1, highlighting the importance of the shape completion regularization for the 3D Siamese Tracker. Results are provided for five different cases: **(i)** an initialization of our network with random weights, **(ii)** our network pre-trained on ShapeNet, **(iii)** our network trained to minimize the completion loss only, **(iv)** our network trained as a regular Siamese tracker by using only our tracking loss, and **(v)** our network trained with both the tracking and completion losses. We observe that both training to minimize the completion loss and training to minimize the tracking loss provide better results than pre-training on ShapeNet and a random initialization. However, combining the shape completion regularizer with our tracking loss is able to improve the tracker's performance beyond either method isolated.

Table 1. Ablation study for different losses we are training with. We report the OPE Success/Precision metrics for different losses averaged over 5 runs. Best results shown in bold.

Ablation	Success	Precision
(i) Before Training (Random)	39.06	41.79
(ii) Pre-trained on ShapeNet	44.54	49.38
(iii) Ours – Completion only	65.36	70.62
(iv) Ours – Tracking only	73.96	78.68
(v) Ours – λ_{comp}@$1e^{-6}$	**76.94**	**81.38**

Completion Loss. Figure 3 shows detailed results obtained as the regularization parameter λ_{comp} is varied. As less weight is given to the completion loss, the performance moves from the results obtained with only the completion loss to those obtained with only the tracking loss. The best trade-off is obtained at a point where both losses are in the same order of magnitude. This occurs somewhere between $\lambda_{comp} = 1e^{-5}$ and $\lambda_{comp} = 1e^{-6}$, where we obtain peak performance.

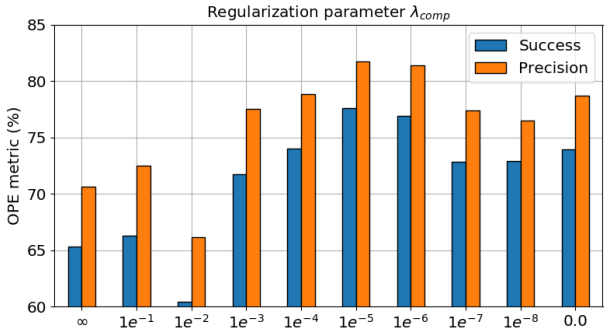

Figure 3. Ablation study for different value for the regularization of the shape completion. We report the OPE Success/Precision metrics for different values of λ_{comp} averaged over 5 runs.

Latent representation dimension. Figure 4 shows how varying the size of our latent representation $\mathbf{z}$ affects the performances. It can be observed that a larger latent representation generally performs better. This is due to the fact that larger latent representations encode more expressive capabilities. However, this reaches a maximum at a size of around $K = 128$ dimensions. Larger latent representations require more expensive computations, but the difference is not significant when comparing a latent representation of 32 dimensions against a 128-dimensional representation. Thus, it is best to use the representation which provides the best tracking performance, *i.e.* a 128-dimensional representation.

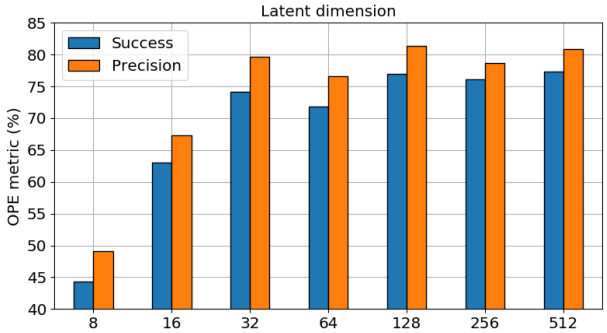

Figure 4. Ablation study for the latent representation size K. We report the OPE Success/Precision metrics for different values of K averaged over 5 runs.

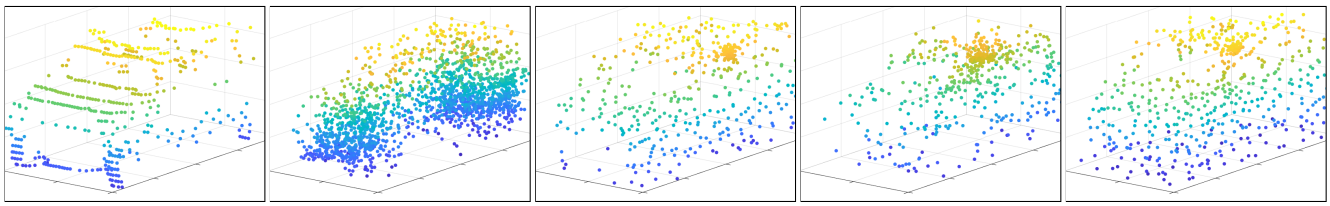

Figure 5. Example of model completion (from left to right): **(i)** Candidate point cloud, **(ii)** Decoded candidate point cloud when it is pre-trained with ShapeNet, **(iii)** Decoded candidate point cloud when it is trained with completion loss only ($\lambda_{comp} = \infty$), **(iv)** Decoded candidate point cloud when it is trained with tracking loss only ($\lambda_{comp} = 0$) (the decoder trained for completion is used for fair comparison), **(v)** Decoded candidate point cloud when it is trained with both tracking and completion losses ($\lambda_{comp} = 1e^{-6}$).

Qualitative results. Figure 5 shows qualitative results regarding decoded shapes $\phi(\mathbf{x})$. We can observe that training for tracking only results in a decoded point cloud containing a large amount of noise, and the model pre-trained with ShapeNet provides a reconstruction which resembles a general car but not the specific candidate car. Training for shape completion only provides a shape reconstruction which is a more complete version of the original candidate shape. Regularizing tracking with shape completion by using $\lambda_{comp} = 1e^{-6}$ provides a reconstruction similar to that using shape completion only. However, the model trained for shape completion only follows the candidate shape more closely. A regularized loss is able to improve tracking results while conserving enough class information as to reconstruct the encoded shape from its latent vector.

Figure 6 illustrates the activations obtained from the cosine similarity for a set of samples obtained using an exhaustive search. We observe that a randomly initialized model generates high scores everywhere, hence providing a bad discrimination. A model pre-trained on ShapeNet is able to better discriminate the shape to track than random initialization, but is still distracted by the environment, confusing other shapes for the car. Our model is able to discriminate fairly well between the ground truth car and the surrounding areas; there are high activations only in the vicinity of the ground truth box. The ideal shape we expect to obtain for the activations is a Gaussian centered at the ground truth BB, as regressing to in training.

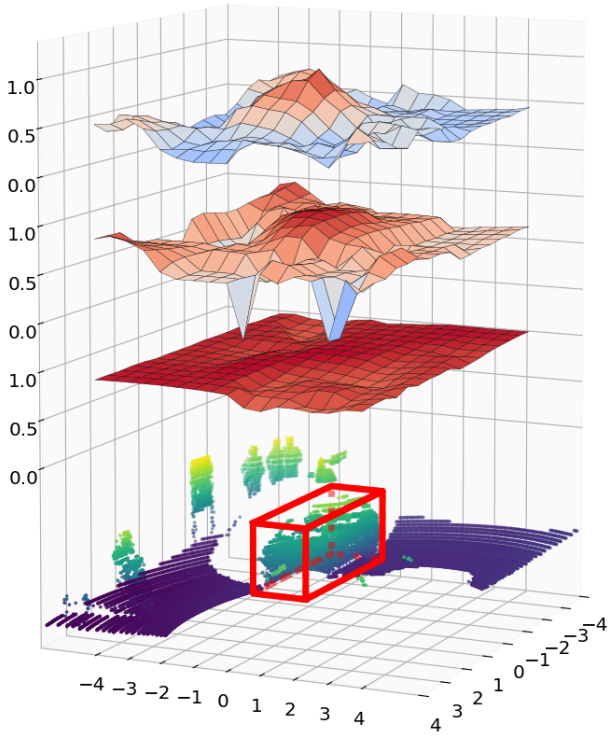

Figure 6. Heatmap of model cosine similarity scores on an exhaustive search space grid: From bottom to top: **(i)** activation using random weights model, **(ii)** activation on pre-trained model (ShapeNet), **(iii)** our model.

4.2. Model Fusion and Shape Aggregation

We construct and update a model $\hat{\mathbf{x}}$ for the target object's shape as we track it. By default, the model is maintained as a point cloud. Our update step for the model after iteration t consists of concatenating the points of the tracked shape in frame x_t with the current model. An alternative is to maintain a model by averaging the latent representations. We investigate the effects of fusing either point clouds or latent representations as well as the effects of different types of aggregation in time for both representations. We test the different shape fusions and aggregations in our method. We report the main results in Table 2.

Early/Late Fusion. We update the model by either concatenating the shape point clouds $\mathbf{x}^t$ (*Early Fusion*) or aggregating the latent shape representations $\mathbf{z}^t$ (*Late Fusion*). Early Fusion requires a larger amount of memory to the store the model shape. Late Fusion allows for a more memory-efficient representation for point clouds, since we only need to keep a latent vector to represent a whole shape. It is also more computationally efficient since the model is not encoded during testing.

Shape Aggregation. We investigate different types of shape aggregation. In particular, we try using the shape in the first frame only, the previous shape only, an aggregation

of the the first and previous shapes, and an aggregation of all the previous shapes. We also investigate aggregating the latent representations by either computing the average, the median, or the max of the vectors across time.

Analysis. As shown in Table 2, concatenating point clouds (Early Fusion) performs generally better than fusing the latent vectors (Late Fusion). This is mainly due to our completion loss designed to handle arbitrary shapes sampled at 2048 points. We do not include any loss that would train our network to aggregate latent vectors. As a result, late fusion does not perform as well as concatenating point clouds.

Using an aggregation of only the first or the previous frame does not perform well. In particular, the number of points belonging to the object in question in a single frame can be significantly small, which impedes a proper shape representation. Should this happen in the first frame, it will imply a bad initial representation. A low point count when tracking using the previous frame will induce drifting.

Fusing the first and previous frames performs surprisingly well and provides the best precision. We believe that the two distant representations complement each other, particularly by limiting the amount of translational drift in the first frames given an initial bad representation. The shapes in the first frames typically contain a limited number of points, since they are sensed from a large distance. They provide a very incomplete shape information, but are still helpful to localize roughly its position but not its orientation. The full model will inevitably drift while the fusion of first and previous frame avoids initial drifts to a certain extent, thus having an improved precision.

For the latent representation, median pooling is less effective than average pooling, but max pooling provides the best performance. We argue that it interacts well with the max pooling layer at the end of our encoder network. By consecutively max pooling over the shape's point features (last layer of our encoder) and over all the previous latent vectors, we actually pool over all the shape's point feature in a tracklet, which provide a more global model latent representation. Still, this is not as effective as Early Fusion.

Table 2. OPE Success/Precision for different Data Fusion and Model Aggregation. All results are averaged over 5 runs. Best representation aggregation shown in bold.

Fusion Data Representation	Early PC	Late Latent
First shape only	54.6 / 64.2	54.6 / 64.1
Previous shape only	64.5 / 69.7	64.4 / 69.6
First and prev. shapes	75.4 / **82.7**	69.1 / **78.1**
All previous shapes	**76.9** / 81.4	63.9 / 73.2
Median Pooling	−− / −−	59.7 / 67.6
Max Pooling	−− / −−	**71.5** / 75.6

4.3. Search Space

Defining an efficient search space is extremely difficult in 3D due to the continuity and cubic nature of 3D space. Thus, an exhaustive search becomes infeasible when a very fine search space is required. To overcome this limitation, we use a Kalman Filter, Particle Filter, and Gaussian Mixture Model to generate candidates. We apply our network using more realistic search spaces, which do not use the ground truth BB, as opposed to the exhaustive search approximation. We argue that our model has good discriminating capabilities, but is limited by the quality of proposed candidates. To support our claim, we report the results obtained by scoring the candidates using their distance to the ground truth object BB – the best possible similarity metric – along with the results obtained using our best model with both early fusion and late fusion. Results are shown in Table 3. It can be observed that our model reaches performances similar to those obtained by selecting the candidate closest to the ground truth, which emphasize the effectiveness of our similarity metric for discrimination.

Table 3. OPE Success and Precision for different Search Space. All results are averaged over 5 runs.

Fusion Data Repres.	Early PC	Late Latent	Closest Space
Kalman Filter	**41.3 / 57.9**	**37.4 / 52.1**	**43.7 / 58.3**
Particle Filter	34.2 / 46.4	33.3 / 44.9	38.4 / 49.5
GMM(k= 25)	35.6 / 49.1	34.0 / 46.1	37.9 / 49.3

4.4. Comparison with Baselines

To compare our method for 3D tracking, we create two baselines due to the absence of 3D tracking methods for this specific task. We take as baselines a state-of-the-art 3D detection method as well as a 2D tracker. The results from these baselines are reported along with our best model using exhaustive search and our best model using a Kalman filter in Table 4. Evaluation metrics are reported using both the 3D IOU and the 2D BEV IOU. The 2D BEV IOU metrics are reported in the same fashion as the 3D IOU, but use the 2D BEV BBs instead of the 3D BBs. Our exhaustive model performs better than both baselines, while the model using a Kalman filter is able to outperform the 2D Tracker baseline.

3D Detection. For the 3D detection baseline, we pair the AVOD-FPN [28] detector with an online matching algorithm. AVOD-FPN utilizes both LIDAR point clouds and RGB images to obtain 3D detections. We use the detection for every frame in our tracklets and preform tracking-by-detection by matching objects frame-by-frame. The object in frame t is selected as the BB with the highest overlap with the BB tracked in frame $t-1$.

2D Tracker. We compare against the popularized 2D Sta-

ple_CA tracker [2, 37], when applied to BEV data. BEV images are extracted from point clouds in our tracklets by projecting points into the ground plane. The resulting 2D tracklets are then fed to the Staple_CA tracker. This method provides a LIDAR-only tracker as a fair baseline for our method, which also only relies on LIDAR input.

Table 4. Baseline comparison using the 3D OPE (3D BB) and the 2D OPE on BEV frames.

Test	OPE_{3D}	OPE_{2D}
STAPLE_CA	– / –	31.60 / 29.30
AVOD Tracking	63.16 / 69.74	67.46 / 69.74
Ours - Kalman Filter	40.09 / 56.17	48.89 / 60.13
Ours - Exhaustive	**76.94 / 81.38**	**76.86 / 81.37**

5. Discussions

Training on complete models. In our experiments, we auto-encode a complete model shape obtained by concatenating all the point clouds in a tracklet. We then enforce candidate shapes belonging to the same object as our model to encode into a vector with a high cosine similarity between itself and the latent representation of our model. An alternative would be to enforce partial shapes belonging to an object at different times to be similar to each other and partial shapes not belonging to the object to be dissimilar from those belonging to the object. In particular, we attempted to provide the object at time t as a target for our Siamese network in place of the full model. However, better results were obtained by using a full model as the target.

A natural extension to training using objects from the same time t is to concatenate different combinations of shapes from the same tracklet at different times. This augmentation is possible since we train our network to complete shape *i.e.* to be invariant to occlusions from different views. This is an intermediate step between training using a single frame as a model and using the whole tracklet to create a model shape for the Siamese network. However, this augmentation increased training time exponentially and did not provide further improvements to our tracking results. There are not enough points in every frame to learn the proper shape of the car without auto-encoding a full model.

Ground included in the car model. The model is pretrained on Shape net, which has complete shapes without noisy points such as a road. In our test, we scaled the BBs by a factor of 1.25 since the original BBs are too tight around the cars and part of the border of cars lie outside their BBs. Such consideration account for 10% of the performances. For this reason, it is possible to see the road in Figure 5, but we believe that including the road does not negatively affect the shape representation. We also considered a fixed offset of 0.5m which proved to be less effective.

Gaussian distance vs IoU. We have tried to regress the 3D IoU between candidate and model BBs instead of regressing a Gaussian distance. However, using the IoU as a target for regression led to worse performances.

Symmetry. Most cars are visible only from one side. We attempted leveraging a prior knowledge of car symmetry in order to complete furthermore the shape of the cars. However, this method did not prove to be effective, in particular because the BBs are not well centered and introduce more noise into our model.

Gaussian Sampling. We generate candidate offsets during training by sampling from a multivariate Gaussian distribution, in contrast with sampling offsets using a fixed grid. Sampling offsets randomly improves performances, since the network is able to learn from a variety of target scores. Fixing an offset grid provides only a discrete number of target scores used for the tracking loss. Lacking variety in training induces worse performances during testing.

Timing. Our model takes on average 1.8ms to evaluate 147 candidates. We do not account for the time spent generating and preparing the candidates and model point clouds for evaluation. This allows us during deployment to increase the number of candidates as much as allowed by the GPU, while still being able to process point clouds in real-time.

6. Conclusion

In this paper, we propose, to the best of our knowledge, the first 3D Siamese tracker applied to point clouds rather than images. We leverage an efficient encoding able to embed meaningful semantic priors thanks to a shape completion regularization. We show that regularizing our network with semantic information results in better discrimination and tracking performances. Also, we provide insights on model building such as early/late fusion and shape aggregation in frames. We compare against baselines in 3D and 2D BEV, showing that our discriminator is able to outperform baselines by using exhaustive search settings. As a result, we propose a purely 3D alternative for tracking cars in urban environments, and show that geometric-oriented approaches are capable of attaining good performances.

Future works will also include improving both the similarity metric and the model update, by including a proposal loss similar to that used in region proposal networks and a smarter model point cloud selection based on the quality of point clouds. Further works will include an extension to Multiple Object Tracking and 3D Object Detection, by leveraging the similarity metric based on our 3D Siamese network. Alternatively, 3D Siamese tracking could be adapted to different classes of objects, articulated shape representation and 2D object tracking.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. 2018. 2, 3, 4

[2] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016. 2, 8

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865. Springer, 2016. 2, 3

[4] A. Bibi, T. Zhang, and B. Ghanem. 3d part-based sparse tracker with automatic synchronization and registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[5] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994. 2

[6] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde. Lidar-camera fusion for road detection using fully convolutional neural networks. *arXiv preprint arXiv:1809.07941*, 2018. 1

[7] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 4

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017. 1

[9] X. Cheng, P. Wang, and R. Yang. Learning depth with convolutional spatial propagation network. *arXiv preprint arXiv:1810.02695*, 2018. 1

[10] H. Chu, W.-C. M. K. Kundu, R. Urtasun, and S. Fidler. Surfconv: Bridging 3d and 2d convolution for rgbd images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3002–3011, 2018. 1

[11] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: science and systems*, volume 38. Philadelphia, 2006. 1

[12] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 2

[13] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *ECCV*, September 2018. 2

[14] F. Engelmann, J. Stückler, and B. Leibe. Joint object pose estimation and shape reconstruction in urban street scenes using 3d shape priors. In *German Conference on Pattern Recognition*, pages 219–230. Springer, 2016. 2

[15] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017. 4

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5

[17] S. Giancola, J. Schneider, P. Wonka, and B. S. Ghanem. Integration of absolute orientation measurements in the kinectfusion reconstruction pipeline. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 2

[18] J. Greenhalgh and M. Mirmehdi. Real-time detection and recognition of road traffic signs. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1498–1506, 2012. 1

[19] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[20] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 1

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[22] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018. 1

[23] I. Jung, J. Son, M. Baek, and B. Han. Real-time mdnet. In *ECCV*, September 2018. 2

[24] P. Karkus, D. Hsu, and W. S. Lee. Particle filter networks with application to visual localization. *arXiv preprint arXiv:1805.08975*, 2018. 3

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[26] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna. Surface networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[27] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016. 2, 5

[28] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint arXiv:1712.02294*, 2017. 1, 7

[29] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, 2018. 2

[30] P. Lei, F. Li, and S. Todorovic. Boundary flow: A siamese network that predicts boundary motion without training on motion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[31] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[32] Y. Liu, X.-Y. Jing, J. Nie, H. Gao, J. Liu, and G.-P. Jiang. Context-aware 3d mean-shift with occlusion handling for robust object tracking in rgb-d videos. *IEEE Transactions on Multimedia*, 2018. 2

[33] M. Luber, L. Spinello, and K. O. Arras. People tracking in rgb-d data with on-line boosted target models. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3844–3849. IEEE, 2011. 2

[34] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 2

[35] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. *arXiv preprint arXiv:1807.00275*, 2018. 1

[36] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 2

[37] M. Mueller, N. Smith, and B. Ghanem. Context-aware correlation filter tracking. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 6, 2017. 8

[38] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, September 2018. 2

[39] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 3

[40] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. 2

[41] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. 1

[42] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 2

[44] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003. 3

[45] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. 4

[46] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. *arXiv preprint arXiv:1802.09298*, 2018. 1

[47] S. Song and J. Xiao. Tracking revisited using rgbd camera: Unified benchmark and baselines. In *Proceedings of the IEEE international conference on computer vision*, pages 233–240, 2013. 2

[48] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart. A layered approach to people detection in 3d range data. In *AAAI*, volume 10, pages 1–1, 2010. 2

[49] D. Stutz and A. Geiger. Learning 3d shape completion under weak supervision. *CoRR*, abs/1805.07290, 2018. 2

[50] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 1

[51] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156, 2004. 2

[52] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 1

[53] R. Timofte, K. Zimmermann, and L. Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine vision and applications*, 25(3):633–647, 2014. 1

[54] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017. 1

[55] N. Umetani. Exploring generative 3d shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs on - SA 17*, 2017. 2

[56] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[57] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2

[58] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE international conference on computer vision (ICCV)*, number EPFL-CONF-230283, pages 4705–4713. IEEE, 2015. 1

[59] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2

[60] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia. Segstereo: Exploiting semantic information for disparity estimation. *arXiv preprint arXiv:1807.11699*, 2018. 1

[61] T. Zhang, S. Liu, C. Xu, S. Yan, B. Ghanem, N. Ahuja, and M.-H. Yang. Structural sparse tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 150–158, 2015. 3

[62] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *ECCV*, September 2018. 2

[63] Z. Zhang, C. Xu, J. Yang, Y. Tai, and L. Chen. Deep hierarchical guidance and regularization learning for end-to-end depth estimation. *Pattern Recognition*, 83:430–442, 2018. 1

[64] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, September 2018. 2