2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341631

Vision Only 3-D Shape Estimation for Autonomous Driving

Josephine Monica[1] and Mark Campbell[2]

Abstract— **We present a probabilistic framework for detailed 3-D shape estimation and tracking using only vision measurements. Vision detections are processed via a bird's eye view representation, creating accurate detections at far ranges. A probabilistic model of the vision based point cloud measurements is learned and used in the framework. A 3-D shape model is developed by fusing a set of point cloud detections via a recursive Best Linear Unbiased Estimator (BLUE). The point cloud fusion accounts for noisy and inaccurate measurements, as well as minimizing growth of points in the 3-D shape. The use of a tracking algorithm and sensor pose enables 3-D shape estimation of dynamic objects from a moving car. Results are analyzed on experimental data, demonstrating the ability of our approach to produce more accurate and cleaner shape estimates.**

I. Introduction

To achieve safe autonomous operation of a vehicle in a dynamic environment with other cars, pedestrians, cyclists, as well as less common objects such as buses or trailers, having an accurate and detailed shape estimates of objects surrounding the car is crucial. The current state of the art tracking algorithms estimate an object's pose and shape typically with a simple bounding box approximation [1]. While a bounding box may give a rough estimate of an object, it often over-estimates the occupied space, thus significantly limiting the available motion, particularly in challenging conditions, such as heavy traffic. In the event of occlusion and rapid viewpoint changes, such as lane merging and changing, the estimated bounding box can also give a misleading estimate of the object's size and pose. Moreover, the capability to produce detailed shape estimates can be used to build a database collection of objects for other applications, such as shape completion training and auto-labeling. Thus, detailed 3-D shape estimates can yield a broad impact.

As the performance of perception algorithms is highly dependent on the supplied sensor information, most state of the art tracking and shape estimation algorithms rely on LiDAR sensors for its ability to produce accurate measurements. Common LiDAR sensors utilize multiple lasers at different longitudes rotating concurrently to give surrounding measurements. Although the state of the art LiDAR with 64 beams provides good resolution and accuracy, it is cost prohibitive for consumer products and is sparse at far ranges. Camera sensors are much more affordable than LiDAR. However, vision only based detection usually suffers from much larger depth errors compared to LiDAR [2], thus camera sensors are typically not used as the primary sensors for tracking and shape estimation.

[1],[2]School of Mechanical and Aerospace Engineering, Cornell University
jm2684@cornell.edu, mc288@cornell.edu

A recent algorithm called Pseudo-LiDAR [2] attempts to bridge the performance gap between LiDAR and vision based object detection. [2] argues that it is not the quality of the measurement data, but its representation that accounts for the performance difference. In [2], the depth map from stereo or monocular camera is converted into point cloud representation, called Pseudo-LIDAR, before being fed into deep convolutional networks. Then, detections via 3-D convolutions on point cloud or 2-D convolutions in the bird's-eye-view slices are less sensitive and more physically meaningful, leading to more accurate results. Furthermore, the identical representation of both LiDAR and Pseudo-LiDAR enables any point cloud based algorithm to be used interchangeably for the two different measurements. A more recent development, Pseudo-LiDAR++ [3], advances previous work [2] through improvement in stereo depth estimation to provide even more accurate measurements, especially for faraway objects.

In this paper we propose a probabilistic framework to produce detailed and accurate shape estimates of tracked objects around a moving car using only camera sensors. Our method leverages the novel Pseudo-LiDAR++ [3] to produce point cloud detections from stereo cameras, which bypasses the challenges inherent in 3-D object detections with convolutional networks. Given that the accuracy of each measurement varies, we learn a probabilistic model of Pseudo-LiDAR++ and incorporate it into the framework in order to improve the performance of shape estimation and tracking. Our method allows the use of any tracking algorithm. The point cloud detections are fused via a recursive Best Linear Unbiased Estimator (BLUE) to build a 3-D shape model while keeping the growth of points minimal. We show that our method can produce accurate and clean shape estimates of tracked objects despite inaccurate and noisy measurements.

II. Related Work

Estimating detailed objects' shape is vital in autonomous driving scenes, such as trajectory planning in evasive steering. A pervasive practice in shape estimation is to represent the object's shape as 2-D bird's-eye view of its extent. Besides bounding box, one common approach for the bird's-eye view outline is to represent the object as an ellipse using Random Hypersurface Model (RHM) [4], [5]. The RHM framework can be extended further to model a star-convex shape [6]. While these approaches provide richer models than bounding boxes, they still pose an underlying and limiting assumption about the shape of the object being ellipse and star-convex. [7] attempts to remove this limiting

shape assumption by using a polyline to represent the tightest 2-D bound of a tracked object. However, the performance, especially the shape growth mechanism, is very sensitive to the sensor accuracy.

While 2-D bird's eye view simplifications may be sufficient for collision avoidance, having detailed 3-D shape estimates can be beneficial for tracking, detections, building data collection, and other applications. Representations for 3-D shape estimation include point clouds, mesh, voxels, and Gaussian processes. For example, [8] initializes the tracked car model with a generic category-level car mesh. The vertices and other parameters of the mesh are then refined using sensor measurements. While promising, this method requires a prior understanding of the general object's shape, as it relies on a good mesh and parameters' initialization. [9] approximates the object's surface with a radial function and uses a Gaussian process to refine the shape function with newly received measurements. Representing the shape with radial function, however limits the object to being star convex. More importantly, the resulting shape is unnaturally smooth, causing unnecessary difficulty in representing protrusions on objects.

Using a point cloud representation for shape estimation is arguably the most natural, as measurements are already in point clouds, and the potential space of shapes is infinite. [10] estimates the velocity of an object by finding the best Markovian alignment between current and previous point cloud measurements. The Annealed Dynamic Histogram (ADH) tracker coarsely samples the velocity estimate and refines the resolution over iterations using the posterior distribution. The shape model is then obtained by simply accumulating the point cloud after transformation to the new coordinate frame using the estimated velocity. Although ADH tracker can give accurate velocity estimates, the shape estimation using simple point cloud accumulation is inferior, especially when the sensor is noisy and less accurate. Since existing shape points are never updated, and all measurements are simply appended to the shape, the quality of the shape estimate suffers from measurement error and noise. Despite being a probabilistic framework, there is no uncertainty estimate available for the shape. Furthermore, the resulting accumulation based on kinematics alignment generally contains a large number of redundant points, as the existing points are not rectified but simply expanded with new measurements.

III. 3-D Shape Estimation Framework

The work proposed here is based on the recursive Best Linear Unbiased Estimate (BLUE) for non-redundant point cloud accumulation. Each point in the measurement is either used to refine points in the current model or is accumulated into the model. This approach minimizes the growth of points, as redundant points from multiple overlapping measurements are fused [11]. More importantly, the probabilistic fusion allows the correction of errors in shape points, as well as provides an estimate of the uncertainty of the shape. A key innovation of this work is in the learning of a sensor error model and inclusion in the probabilistic framework. Finally, a shape compression algorithm and an outlier removal method are also implemented to maintain the most efficient shape.

Fig. 1 shows the overall pipeline for our vision only tracking and shape estimation framework. Measurement from stereo images are processed using Pseudo-LiDAR++, which includes PointRCNN [12] detection algorithm, resulting in 3-D point cloud detections. Our framework consists of two main components, tracking and point cloud fusion; both utilize the output from Pseudo-LIDAR++. Before fusing the point cloud, the shape model and the new measurement must be transformed into the same coordinate frame. This transformation is obtained via a tracking algorithm relative to the sensor. Note that all points are defined with respect to the ego frame, unless otherwise stated.

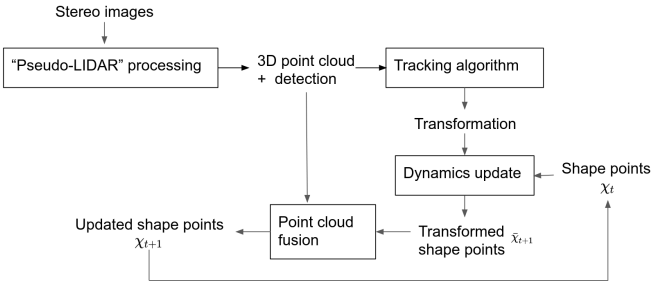

Fig. 1. Block diagram describing the overall pipeline of shape estimation and tracking.

While an object tracking algorithm is required for the proposed framework, the proposed point cloud fusion algorithm is agnostic to the specific type of object tracking algorithm. In this paper, we use a standard Kalman filter algorithm for tracking. However, more sophisticated tracking algorithms, such as the ADH tracker [10] could be used. In fact, with the benefit of non-redundant representation of the proposed shape in our algorithm, we can further improve the accuracy of ADH tracker. Instead of comparing the current measurement to only the previous measurement, the current measurement can be compared to the full shape model. This was previously infeasible, as the run time for velocity estimation scales with the total number of points.

IV. Sensor Error Analysis

The accuracy of a sensor generating point clouds is usually affected by the relative pose between the measured object to the sensor, i.e., measurements are less accurate and more noisy at farther distances. Given that the accuracy of each measurement varies, modeling the accuracy of each measurement point can be beneficial to the eventual shape estimate. We can improve the performance of both tracking and shape estimation by learning a probabilistic error model as a function of range, and incorporating this error model into tracking and shape estimation approaches.

[3] pre-trains its stereo depth network on the synthetic Scene Flow dataset [13] and fine-tunes it on the 3,172 training images of KITTI dataset. We evaluate on the KITTI tracking benchmark [14], specifically on the car category,

which is split into 25 sequences (436 cars) for training and 12 sequences (408 cars) for validation. We compare the bounding box detections from the Pseudo-LiDAR++ in our training data set to the ground truth provided by KITTI.

Fig. 2 shows the box plot of x, y, z, and yaw angle errors of Pseudo-LiDAR++ bounding box detections as a function of planar $x-y$ range measurement. An illustration of the coordinate frame notation can be seen in Fig. 4. In order to reduce bias in the statistical evaluation of the errors, if a tracked car does not move within a sequence of detections, for example when both the ego and tracked car are stationary, only the first measurement is counted. This is to avoid bias towards series of long stationary measurements. We then take the mean of every bin and fit as a quadratic function of $x-y$ measurement range. Note that bins with too few data points are removed from the fit. These quadratic functions of error as a function of measurement range are used to model the sensor error standard deviation.

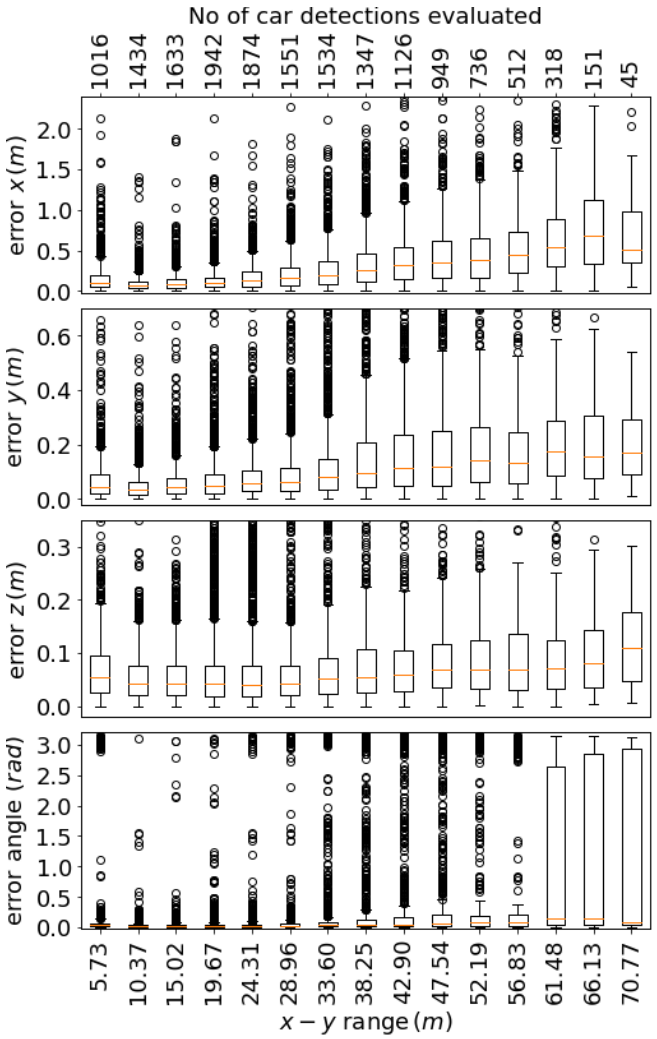

Fig. 2. Box plot for Pseudo-LiDAR++ detection errors in x, y, z and yaw angle as a function of $x-y$ planar distance.

V. Point Cloud Fusion

Each measurement point $\tilde{\boldsymbol{p}}$ of the surface of the shape is assumed to be located at $\boldsymbol{p}$ and corrupted with zero-mean Gaussian noise:

$$\tilde{\boldsymbol{p}} = \boldsymbol{p} + \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{C}}) \tag{1}$$

where $\tilde{\boldsymbol{C}}$ is the measurement covariance matrix modeled in Section IV. We assume no correlation between points in a point cloud.

Fig. 3 shows the recursive steps of point cloud fusion process in a block diagram. The new measurement points are compared with the current shape model to decide if they were previously seen or unseen. Familiar or seen points are used to refine the existing shape, while unfamiliar or unseen points are added to the shape model. Shape compression and outlier removal can also be performed to enhance the result as desired. Each component of the process is further explained in the following subsections.

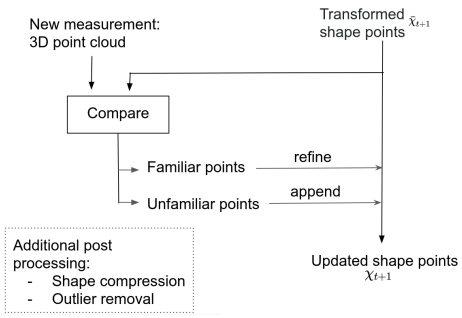

Fig. 3. Block diagram describing the process of point cloud fusion.

A. Dynamics Update

Before performing point cloud fusion, the shape model and the new measurement must be transformed into the same coordinate frame. Given a transformation comprising a rotation $\boldsymbol{R}$ about point $\boldsymbol{c}$ and a translation $\boldsymbol{\delta}$, dynamics update to a shape point estimate $\hat{\boldsymbol{p}}$ and its covariance $\hat{\boldsymbol{C}}$ is defined as:

$$\hat{\boldsymbol{p}} \leftarrow \boldsymbol{R}(\hat{\boldsymbol{p}} - \boldsymbol{c}) + \boldsymbol{c} + \boldsymbol{\delta} \tag{2}$$

$$\hat{\boldsymbol{C}} \leftarrow \boldsymbol{R}\hat{\boldsymbol{C}}\boldsymbol{R}^T \tag{3}$$

(3) assumes that all the transformation parameters are exact, i.e., ground truth transformation. For transformation with uncertainty, the covariance update can be simply derived from (2) either analytically through Jacobian matrices or numerically through sigma point filter.

B. Distinguishing Familiar and Unfamiliar Points

Before deciding whether measurement $\tilde{\boldsymbol{p}}$ and existing point $\hat{\boldsymbol{p}}$ come from the same surface point, we need to compute the merged estimate $\hat{\boldsymbol{p}}'$. That is, the updated shape point location if the two points are merged is given as:

$$\hat{\boldsymbol{p}}' = \hat{\boldsymbol{p}} + \hat{\boldsymbol{C}}'\tilde{\boldsymbol{C}}^{-1}(\tilde{\boldsymbol{p}} - \hat{\boldsymbol{p}}) \tag{4}$$

where $\hat{\boldsymbol{C}}'$ is the merged point's covariance matrix:

$$\hat{\boldsymbol{C}}' = (\hat{\boldsymbol{C}}^{-1} + \tilde{\boldsymbol{C}}^{-1})^{-1} \tag{5}$$

We use a measure of 'closeness' to capture familiarity: in this case, the Mahalanobis distance. The Mahalanobis distances from the merged estimate $\hat{\boldsymbol{p}}'$ to the new measurement point $\tilde{\boldsymbol{p}}$ and the existing point $\hat{\boldsymbol{p}}$ can be defined as:

$$\tilde{d} = \sqrt{(\hat{\boldsymbol{p}}' - \tilde{\boldsymbol{p}})^T \tilde{\boldsymbol{C}}^{-1} (\hat{\boldsymbol{p}}' - \tilde{\boldsymbol{p}})} \tag{6}$$

$$\hat{d} = \sqrt{(\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})^T \hat{\boldsymbol{C}}^{-1} (\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})} \tag{7}$$

Measurement $\tilde{\boldsymbol{p}}$ and existing point $\hat{\boldsymbol{p}}$ come from the same surface point if both Mahalanobis distances $\tilde{d}$ and $\hat{d}$ are smaller than a user defined threshold d_{thres}:

$$\tilde{d}, \hat{d} < d_{\text{thres}} \tag{8}$$

We use $d_{\text{thres}} = 3$ in our experiments.

C. Point Cloud Refinement

Given a set of measurement points $\{\tilde{\boldsymbol{p}}_j\}$ originating from the same surface point as $\hat{\boldsymbol{p}}$, we use best linear unbiased estimator to update our estimates of $\hat{\boldsymbol{p}}$ and its covariance $\hat{\boldsymbol{C}}$:

$$\hat{\boldsymbol{p}}' = \hat{\boldsymbol{p}} + \sum_j \hat{\boldsymbol{C}}' \tilde{\boldsymbol{C}}_j^{-1} (\tilde{\boldsymbol{p}}_j - \hat{\boldsymbol{p}}) \tag{9}$$

$$\hat{\boldsymbol{C}}' = \big(\hat{\boldsymbol{C}}^{-1} + \sum_j \tilde{\boldsymbol{C}}_j^{-1}\big)^{-1} \tag{10}$$

where $\{\tilde{\boldsymbol{C}}_j\}$ are the covariance matrices corresponding to measurement $\{\tilde{\boldsymbol{p}}_j\}$, $\hat{\boldsymbol{p}}'$ is the new estimated location, and $\hat{\boldsymbol{C}}'$ is the updated covariance estimate.

The full measurement update process is presented in Algorithm 1.

D. Shape Compression

Given a set of points χ constituting an object, shape compression aims to reduce the number of shape points n while maintaining as much information about the object's shape. To do this, first we find a pair of points that are most likely to be similar, namely:

$$(\hat{\boldsymbol{p}}_a, \hat{\boldsymbol{p}}_b) = \underset{\hat{\boldsymbol{p}}_i, \hat{\boldsymbol{p}}_j \in \chi, i \neq j}{\arg\max} \mathcal{N}(\boldsymbol{0}; \hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j, \hat{\boldsymbol{C}}_i + \hat{\boldsymbol{C}}_j) \tag{11}$$

From the pair $(\hat{\boldsymbol{p}}_a, \hat{\boldsymbol{p}}_b)$, we delete the less certain point, i.e., the point with the larger determinant of its covariance matrix $|\hat{\boldsymbol{C}}|$. This process is repeated until the desired number of points is reached or until the maximum likelihood of point pairs being similar is smaller than a user defined threshold.

Input: Measurement points $\mathcal{Z}_t = \{\tilde{\boldsymbol{P}}_j = (\tilde{\boldsymbol{p}}_j, \tilde{\boldsymbol{C}}_j)\}$ and existing shape model $\chi_{t-1} = \{\hat{\boldsymbol{P}}_i = (\hat{\boldsymbol{p}}_i, \hat{\boldsymbol{C}}_i)\}$.
Output: Updated model χ_t

```
χ_t ← χ_{t−1}, U ← Z_t;
foreach point P̂_i ∈ χ_t do
    F ← ∅;
    Compute a set of k-nearest new measurement
      points K ← {P̃_j | P̃_j ∈ kNN(P̂_i, k)};
    foreach point P̃_j ∈ K do
        Compute Ĉ'_ij from (5);
        Compute p̂'_ij from (4);
        Compute d̃_ij, d̂_ij from (6) and (7);
        if d̃_ij, d̂_ij < d_thres then
            F ← F ∪ {P̃_j};
            U ← U \ {P̃_j}
        end
    end
    if F ≠ ∅ then
        Compute P̂'_i the merged point of P̂_i and F
          with (9) and (10);
        P̂_i ← P̂'_i
    end
end
χ_t ← χ_t ∪ U;
```

Algorithm 1: Point cloud fusion algorithm.

E. Outlier Removal

Even with a statistical model of sensor error, sensor measurements often contain outliers. While our fusion algorithm accounts for noisy measurements, we prefer to not fuse outliers with existing shape points, as they may distort the shape estimate. However, since outliers are not used to refine existing points, they are counted as unfamiliar points and will be appended to the shape estimate. Thus, the purpose of outlier removal is to remove the outliers to get a cleaner shape estimate.

To detect such points, we first find the distance to the k^{th} nearest neighbor for every point. Typically, outliers do not have many nearby points. Thus, we remove points whose distance to its k^{th} nearest neighbor lie beyond $Q_3 + 1.5IQR$, where Q_3 is the 75% quartile, and IQR is the interquartile range. A value of $k = 30$ is used in our experiments.

VI. Tracking

The purpose of tracking in the context of shape estimation is to find the coordinate transformation of the tracked object across time steps. This transformation is used to bring the shape points to the same coordinate frame as the new measurement prior to fusion, as described in Section V-A. While our point cloud fusion framework can use any tracking algorithm, a standard 3-D centroid Kalman filter tracking is used here as a baseline to evaluate shape estimates using only vision detections.

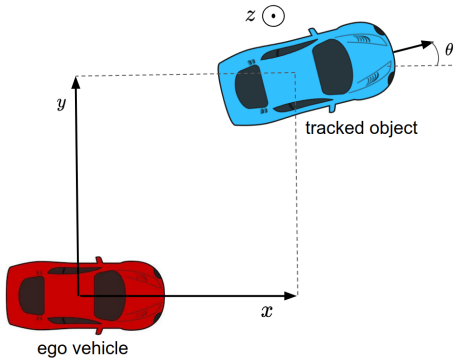

Fig. 4. Diagram picturing a tracked object's pose relative to the ego vehicle.

A. State

The state $\boldsymbol{s}$ of our tracker is defined as the pose of the center of the object's bounding box and its first time derivatives:

$$\boldsymbol{s} = \begin{bmatrix} x & y & z & \theta & \dot{x} & \dot{y} & \dot{z} & \dot{\theta} \end{bmatrix}^T$$

where x, y, z, θ are the pose of the tracked object relative to the ego vehicle as depicted in Fig. 4, and $\dot{x}$, $\dot{y}$, $\dot{z}$, $\dot{\theta}$ are their first time derivatives.

B. Measurements

Pseudo-LiDAR++ [3] in conjunction with PointRCNN [12] generates bounding box detection for objects. We use the pose of the bounding box detection as our measurement vector:

$$\boldsymbol{h} = \begin{bmatrix} x_{bb} & y_{bb} & z_{bb} & \theta_{bb} \end{bmatrix}^T \tag{12}$$

where (x_{bb}, y_{bb}, z_{bb}) is the center location of the detected bounding box, and θ_{bb} is its detected orientation.

VII. Experiments

We present experiments on car objects in the KITTI tracking dataset [14] that is split into 25 sequences (436 cars) for training and 12 sequences (408 cars) for validation. In all experiments, every point cloud measurement from the Pseudo-LiDAR++ is stochastically down-sampled to a maximum of 2000 points.

The following subsections are organized as follows. In Section VII-A, we present a method and metrics to evaluate shape estimates. To first develop an insight of the importance and merit of our probabilistic fusion approach for shape estimation, we present a study case of a challenging scenario of approaching car in Section VII-B. Next, we evaluate our algorithm on the whole KITTI validation set to show that our algorithm can work for varying cases. Finally, Section VII-D presents a qualitative study to demonstrate the novelty of our shape estimation algorithm pictorially. In all the studies, we show a comparison of our method to the simple accumulation baseline.

A. Evaluation Metric for Shape Estimation

Determining an appropriate evaluation metric to evaluate a point cloud is challenging. Ideally, we want a reference of the true shape to compare. Unfortunately, generally there is no true shape reference except for controlled experiments with known cars and well defined CAD models. Consequently, this does not lend to a holistic statistical analysis. Instead, we compare to a shape reference defined by the accurate measurements from LiDAR which are also known to be metrically very precise. In order to compare to the best full shape as possible, we accumulate all available LiDAR point clouds across all frames for a given car using the ground truth transformation, which is provided in the KITTI dataset. Even after the LiDAR point clouds are accumulated, the shape reference developed from LiDAR can still be incomplete, especially for cars at far ranges. Therefore, a shape completion method called Point Completion Network (PCN) [15] is applied to complete the shape for evaluation.

Before evaluation, the shape reference must be transformed to the same coordinate frame as the shape estimate. For every point in a shape estimate, its nearest neighbor distance to point cloud reference d_i is computed. We define evaluation metrics d_{nn} and σ_{nn}, the average and standard deviation of nearest neighbor distances among all n shape points:

$$d_{nn} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{13}$$

$$\sigma_{nn} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - d_{nn})^2} \tag{14}$$

d_{nn} measures the accuracy of the shape, while σ_{nn} indicates its precision or uncertainty.

B. Study Case: Car Approaching from Far

In order to gain insight into the performance of the 3-D shape estimation algorithm with Pseudo-LiDAR++, we first study a particularly challenging case from the KITTI dataset: a car approaching from far distance. Intuitively, initial detections are sparse and uncertain, but get better as the car approaches. The varying number and quality of points will stress the fusion algorithm. Fig. 5 shows images seen by the ego vehicle at two different time instances. Initially, the point cloud detections are inaccurate, as depth prediction is challenging for small objects at far ranges. This is mostly due to the poor quality of associated pixels when the car is faraway; the car is also in shadow. As the car comes closer, the depth errors and noises reduce over time; close detections are quite accurate and detailed. For this analysis, we compare our algorithm with a baseline of simple accumulation, with both transformations from ground truth and tracking result.

We first isolate the point cloud fusion from the tracking component to independently analyze the performance of only the fusion algorithm. Thus, ground truth transformation is used here. Fig. 7 (left) shows the shape result of the simple accumulation baseline. The baseline result shows multiple

Fig. 5. Images from the left camera. The car of interest here is the one in front. Top: Initial detection. Bottom: Sixteenth detection (one and a half seconds later).

car front layers corresponding to inaccurate point cloud measurements from different time instances. These inaccuracies are mostly due to the detections when the car is very distant. If the measurements were perfectly accurate, the layers would perfectly align. It shows that simple accumulation baseline suffers in the event of inaccurate measurements. Fig. 7 (right) shows the result of our fusion algorithm. Unlike the simple accumulation model, our algorithm nicely fuses the multiple car layers. Quantitative metrics comparing the two approaches as a function of number of detections received are plotted in Fig. 9. The accuracy of our approach is clear. In addition, while we do not apply shape compression in these results, the number of points n reduces significantly using our approach (Fig. 10).

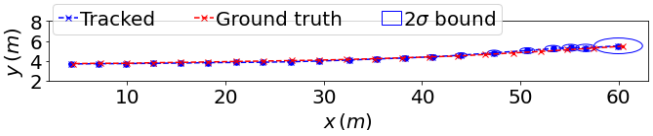

Fig. 6. Top down view of the tracked and true trajectories of the car in ego frame.

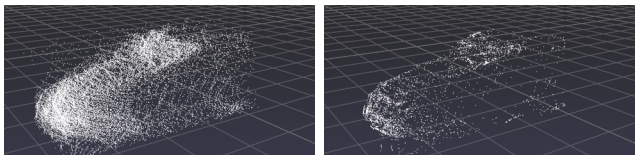

Fig. 7. Point cloud model after running shape estimation with Pseudo-LIDAR++ measurements for one and a half seconds (sixteen detections). Left: Baseline method. Right: Our fusion method.

Next, we perform the same analysis using the transformation obtained from the tracking algorithm. Fig. 6 shows a bird eye view comparison between the true trajectory and the trajectory from our tracker. We can see that there is a larger error at ranges from $60m$ (initial detection) to $40m$. These range dependent errors align with our intuition of the sensor processing, and with the statistical error analysis in Fig. 2. Fig. 8 compares our full fusion and tracking algorithm with the simple accumulation baseline using the same tracking

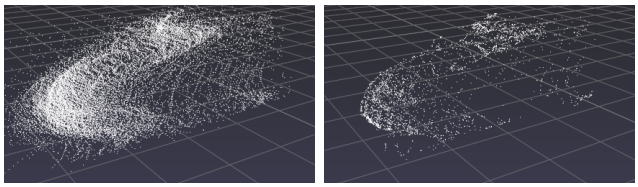

Fig. 8. Point cloud model after running shape estimation and tracker with Pseudo-LIDAR++ measurements for one and a half seconds (sixteen detections). Left: Baseline method. Right: Our fusion method.

result. Again, the shape estimate from our fusion algorithm looks cleaner and better aligned than the simple accumulation baseline. The quantitative results are also reported in Fig. 9 and Fig. 10, where the performance of our point cloud fusion algorithm surpasses the simple accumulation method in all aspects.

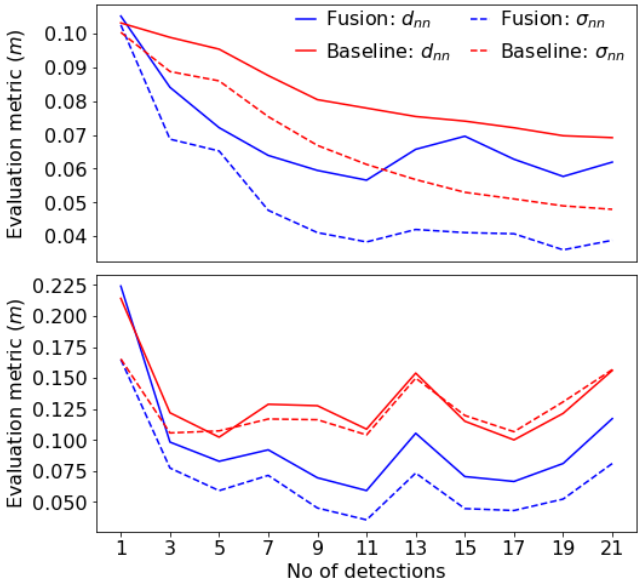

Fig. 9. The evolution of performance metrics of the shape in the study case. Top: Run with ground truth transformation. Bottom: Run with tracker.

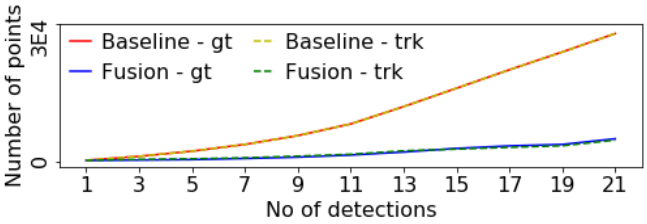

Fig. 10. The evolution of the number of shape points in the study case, evaluated for transformations from both tracker (trk) and ground truth (gt).

In this study case, the quality of shape estimate generally improves over time (Fig. 9), as better point cloud detections are received when the car comes closer. One exception is in the last few detections of the tracking case (Fig. 9 (bottom)), where the error increases. This is because the car becomes partially occluded, as it comes too close to

the ego vehicle, causing difficulty in tracking accurately. While we quantitatively and qualitatively observe (Fig. 7-10) that shape estimation performs better given its ground truth transformations due to errors from tracker, the performance gap of shape estimation between the two is narrower using our fusion algorithm, showing the robustness of our method.

C. Quantitative Evaluation on KITTI Dataset

We evaluate the performance of our vision only 3-D shape estimation algorithm on the whole KITTI validation set, and compare it to the simple accumulation baseline. The validation set consists of a mix of trajectory scenarios. The statistics of the measurement ranges in Fig. 12 shows that most cars tend to be at farther distances at their first detections and approach the ego over time. For each car, we run the two shape estimation algorithms from its initial to its final detection and evaluate the shape estimate at every time step, resulting in shape estimates built using 1 detection, 2 detections, and so on. In this study, we show the results only up to 30 detections, where the number of data points (cars) in the category is still sufficiently large.

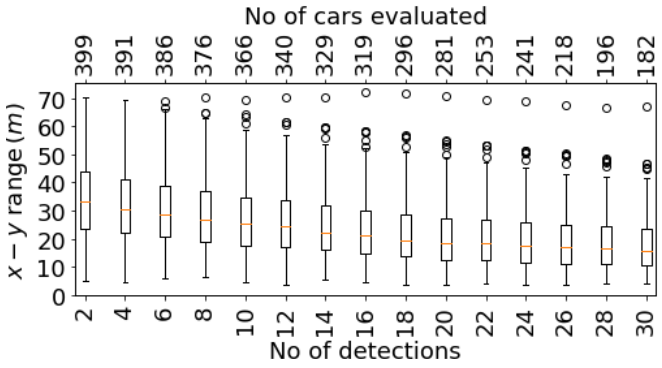

Fig. 11. Statistics of measurement range Vs the number of measurements received from the initial detection up to the current measurement for cars on the KITTI validation set.

1) Shape estimation with ground truth transformation: First, we perform shape estimation using the ground truth transformation to analyze the performance of shape estimation apart from the influence of tracking error. The distribution of the evaluation metrics (Section VII-A) of shape estimates with fusion and simple accumulation are compared in Fig. 12. Here, it is clear that our fusion algorithm outperforms the baseline method in all aspects. Additionally, it shows a general trend of improvement in the quality of shape estimates with more detections. This aligns with the statistics of the measurement ranges (Fig. 11) that most cars in the validation set are initially at farther distances and approach the ego vehicle, hence giving more accurate measurements, over time.

2) Shape estimation with tracking result: We perform the same analysis as in VII-C.1, but now using the estimated transformation from tracking. The distribution of performance metrics for both algorithms are shown in Fig. 13. Comparing Fig. 12 to Fig. 13, both shape estimation algorithms generally perform worse when using tracking

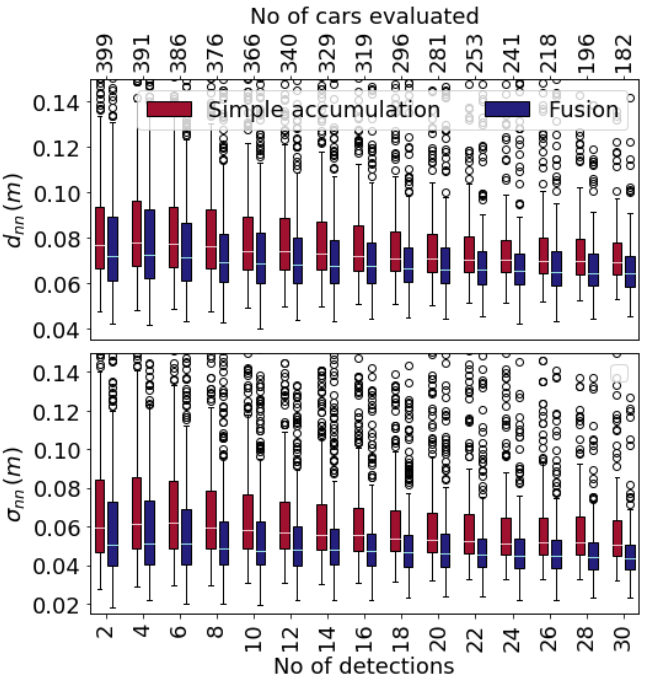

Fig. 12. Statistical analysis on the metrics of our fusion method Vs the simple accumulation baseline on the KITTI validation set.

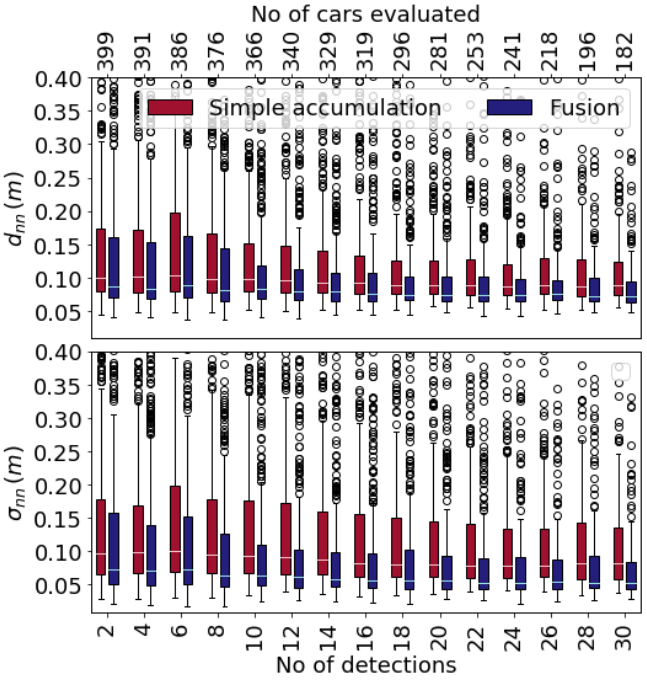

Fig. 13. Statistical analysis on the metrics of our fusion method Vs the simple accumulation baseline with tracking on the KITTI validation set.

result, due to the additional error induced from the estimated transformation. Nevertheless, the performance of our fusion approach outperforms the simple accumulation method.

D. Qualitative evaluation on KITTI dataset

To complement the quantitative evaluation and demonstrate the merit of our fusion algorithm, several shape es-

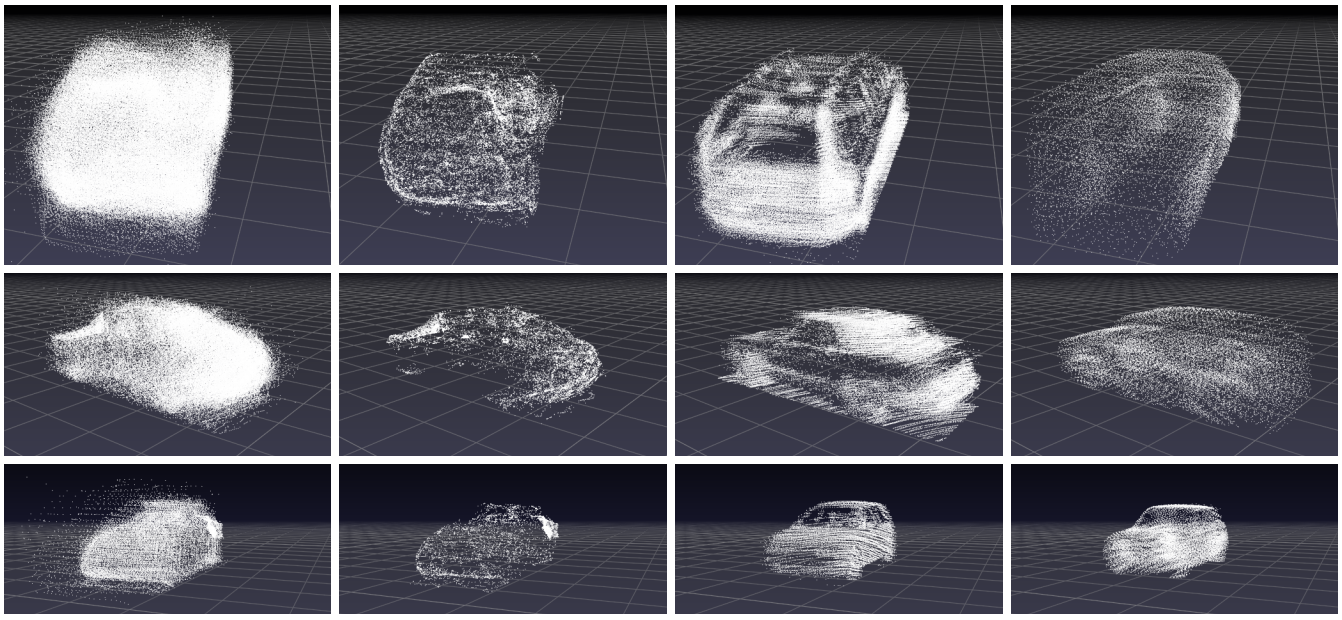

Fig. 14. Leftmost: Shape estimates of the baseline method with a tracker. Left: Shape estimates of our method with a tracker. Right: Accumulated LiDAR point clouds transformed with ground truth information. Rightmost: Completed point clouds output by PCN [15]

timation with tracking results using the full measurement sequences are shown in Fig. 14, along with the developed shape references. It can be seen that our fusion algorithm accounts for bias due to measurement and tracking error, and reduces noisy measurement, resulting in more accurate and precise shape estimates.

VIII. Conclusion

We present a probabilistic framework for precise 3-D shape estimation and tracking using vision only measurement. For novel Pseudo-LiDAR++ measurements, we learn a probabilistic error model of the point clouds as a function of range, and incorporate this error model into tracking and shape estimation approaches. Our shape estimation algorithm minimizes the growth of shape points, and can be used with any tracking algorithm. Quantitative and qualitative experiments demonstrate that our fusion algorithm is able to give more accurate and efficient shape estimates compared to the simple accumulation baseline in spite of inaccurate measurements and tracking errors.

Acknowledgment

The authors would like to acknowledge the support from NSF grant S&AS: INT: Inference, Reasoning and Learning for Robust Autonomous Driving Grant, IIS-1724282.

References

[1] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *ICRA*. IEEE, 2014.

[2] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[3] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *ICLR*, 2020.

[4] Marcus Baum and Uwe D Hanebeck. Random hypersurface models for extended object tracking. In *International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2009.

[5] Marcus Baum, Benjamin Noack, and Uwe D Hanebeck. Extended object and group tracking with elliptic random hypersurface models. In *13th International Conference on Information Fusion*. IEEE, 2010.

[6] Marcus Baum and Uwe D Hanebeck. Shape tracking of extended objects and group targets with star-convex rhms. In *14th International Conference on Information Fusion*, pages 1–8. IEEE, 2011.

[7] Kevin Wyffels and Mark Campbell. Precision tracking via joint detailed shape estimation of arbitrary extended objects. *IEEE Transactions on Robotics*, 33(2), 2016.

[8] Q. Feng, Y. Meng, M. Shan, and N. Atanasov. Localization and mapping using instance-specific mesh models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[9] Murat Kumru and Emre Özkan. Three-dimensional extended object tracking and shape learning using gaussian processes. *arXiv preprint arXiv:1909.11358*, 2019.

[10] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3d shape, color, and motion for robust anytime tracking. In *Robotics: science and systems*, 2014.

[11] Tomi Kyöstilä, Daniel Herrera, Juho Kannala, and Janne Heikkilä. Merging overlapping depth maps into a nonredundant point cloud. In *Scandinavian Conference on Image Analysis*. Springer, 2013.

[12] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[13] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[15] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *International Conference on 3D Vision*. IEEE, 2018.