In-Vehicle Object-level 3D Reconstruction of Traffic Scenes

Qing Rao*† and Samarjit Chakraborty‡

Abstract*—Emerging automotive applications such as in-vehicle Augmented Reality (AR) and fully automated parking require a comprehensive understanding of the vehicle's three-dimensional surrounding represented as an *object-level* environmental model. In this model, not only 3D poses (positions and orientations) and 3D sizes of detected objects are registered, but 3D shapes (geometries) need to be reconstructed precisely. A combination of 3D object detection and 3D surface reconstruction techniques, referred to as *object-level 3D reconstruction*, is fundamental to building such environmental models. However, the possibilities to incorporate object-level 3D reconstruction in a car have not been sufficiently explored either in academic research or in the industry. This primarily stems from the cost and resource constraints associated with the automotive domain. In this paper, we address these constraints by proposing implementations of *in-vehicle object-level 3D reconstruction* in two specific use cases: *(i)* augmented reality and *(ii)* automated parking. For augmented reality, we propose a cost-efficient solution called monocular *3D Shaping* that requires only a single frame from a monocular camera as input. For automated parking, we propose a resource-efficient alternative that generates more precise 3D reconstruction results by taking advantage of additional 3D sensors (such as Lidars). The crux of our proposed approaches lies in the use of a *Latent Shape Space*, where various 3D shapes are represented using only two parameters. As a result, highly complex 3D shapes can now be transmitted using a low- to medium-bandwidth in-vehicle communication infrastructure in a cost-effective manner.

***Index Terms*—3D reconstruction, deep learning, in-vehicle augmented reality.**

I. Introduction

OUTDOOR 3D surface reconstruction and 3D object detection have long been challenging research topics in computer vision and robotics. They have slightly different focuses. In general, 3D surface reconstruction aims at reconstructing the geometric surface of *static* infrastructure objects such as roads or buildings. A 3D environmental map can be thereby generated for further use, e.g., for precise localization or AR navigation. 3D object detection, on the other hand, mainly focuses on recognizing and classifying *dynamic* objects, including cars, pedestrians, and other traffic entities. The results of 3D object detection are usually interpreted as a list of 3D bounding boxes (without detailed geometric information) that indicate the 3D size, 3D position, orientation, and speed of objects in the surroundings. These are important inputs for subsequent function blocks for self-driving cars, such as prediction and trajectory planning [1], [2], [3].

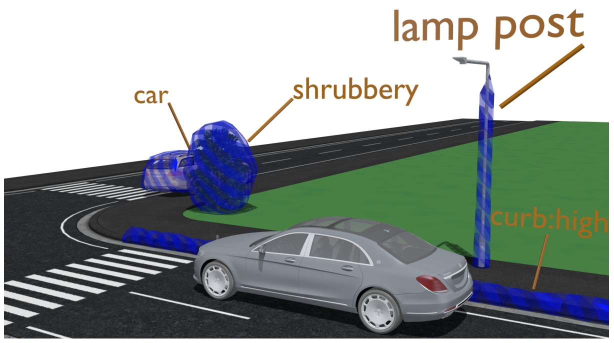


Fig. 1: Example of an object-level environmental model in urban area with precise 3D shapes.

The output of 3D object detection is considered sufficient for simple self-driving use cases, e.g., driving on the highway. However, for complicated scenarios in an urban area, knowledge about 3D object geometries might be essential to enable the ego-car to carry out more challenging tasks, such as automated parking that requires an environmental model with comprehensive geometric information of the surroundings at an object level. In this environmental model, not only 3D bounding boxes of surrounding objects are registered, but also 3D shapes need to be precisely reconstructed (Fig. 1). Combining 3D surface reconstruction and 3D object detection enables building such object-level environmental models, and the technique is referred to as *object-level 3D reconstruction*. Although its concept [4], [5] already dates back to around ten years ago, the possibilities to incorporate object-level 3D reconstruction in a car has not been sufficiently explored, either in academic research or in the industry.

Automotive E/E architectures: To integrate a function into a mass-production car, we need to take specific in-vehicle constraints into consideration. Unlike a desktop computer with a centralized architecture, automotive in-vehicle electrical/electronic (E/E) architectures are heterogeneous and distributed systems comprising multiple Electronic Control Units (ECUs) along with a communication system connecting them. Based on their main functions, ECUs in a car can be categorized into four major domains. These include the *body* domain that provides comfort-related functions, the *powertrain* domain that controls the engine, the *ADAS* (Advanced Driver Assistance System) domain that

*Real-Time Computer Systems, Technical University of Munich. Arcisstr. 21, 80290 Munich, Germany.
‡Department of Computer Science, University of North Carolina at Chapel Hill, 201 S. Columnbia St, Chapel Hill, NC 27599, USA.
†Q. Rao was with Daimler AG, Research and Development at Wilhelm-Runge-Str. 11, 89081 Ulm, Germany, when this work was done.

helps improve driver safety, and the *telematics* domain for in-vehicle telecommunication and infotainment. To meet the special requirements of these different in-vehicle domains, various in-vehicle communication networks, also known as in-vehicle buses, have been developed in the past. Today, apart from several widely-used buses such as CAN or FlexRay [6], Ethernet technology has gained importance for in-vehicle communication due to its high transmission capacity and low cost. Ethernet might be used as a *backbone* bus that carries inter-domain communication traffic in future in-vehicle E/E architectures.

Goal of this paper: Our objective in this paper is to develop a resource-efficient solution to object-level 3D reconstruction, which is implementable on an automotive E/E architecture. This leads to different design questions during the early phase of system development. On the one hand, certainly, the 3D reconstruction algorithm itself needs to be further developed to be able to run robustly in a car. In addition, questions such as *How to combine 3D object detection and 3D surface reconstruction techniques for traffic scenes?* or *Which algorithm would work best for which specific use cases?* also need to be answered. Simultaneously, the in-vehicle E/E architecture needs to be modified or redesigned to support the developed algorithms. In particular, questions such as these need to be answered: *What kind of sensor inputs are necessary? How many computational resources are required? How to effectively partition the algorithm corresponding to the different in-vehicle domains?* Finally, the cost implication needs to be seriously considered when making design decisions based on the answers to these questions.

To narrow down the scope of these open design questions, we first take the premise that computing resources in future automotive E/E architectures will be sufficient to support deep neural networks. In addition, we focus on two specific use cases, namely in-vehicle augmented reality and automated parking. For augmented reality, we propose a cost-efficient solution called monocular *3D Shaping* (Section III) that only requires a single frame from a monocular camera as input. For automated parking, we propose a resource-efficient alternative (Section IV), which generates more precise 3D reconstruction results by taking advantage of additional 3D sensors (e.g., a Lidar). The crux of our proposed approaches lies in the use of a *Latent Shape Space* [7], [8], where various 3D shapes are represented using only two parameters. Using the latent shape representation, highly complex 3D shapes can be transmitted through a low- or medium-bandwidth in-vehicle bus, which enables an efficient partition of both the proposed approaches into different in-vehicle domains. These results are fundamental to a *practical implementation* of in-vehicle object-level 3D reconstruction, which we believe is the main engineering contribution of this paper.

From a technical point of view, the contributions of this paper are three-fold. First, we exploit deep neural networks for in-vehicle object-level 3D reconstruction, which, to the best of our understanding, is the first of its kind in an automotive context. In addition, by using additional 3D sensors, we enable object-level 3D reconstruction in complicated real-world traffic scenes, even in the presence of heavy occlusion (Fig. 2). Finally, we demonstrate how the proposed approaches can be integrated into a future in-vehicle E/E architecture, which enables new applications such as in-vehicle augmented reality and fully automated parking. Our proposed techniques are good candidates for facilitating several future automotive applications, particularly those supporting autonomous features.

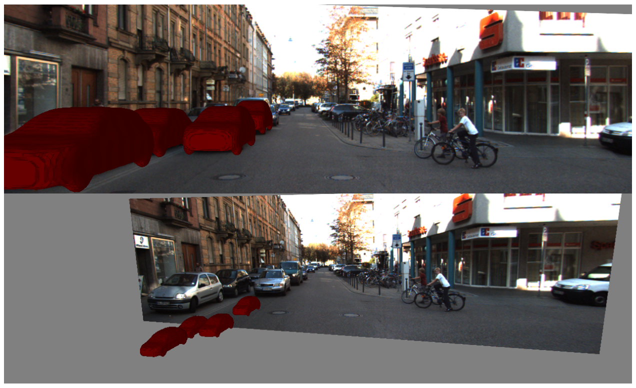


Fig. 2: Augmented Reality (AR) visualization of object-level 3D reconstruction results by heavy occlusion.

II. Related Work

A. Object-Level 3D Reconstruction

The problem of object-level 3D reconstruction breaks down to two major questions: *i)* How to represent complex 3D shapes? and *ii)* How to retrieve 3D shapes from given input cues? For retrieving shapes, different methods were developed in the past including *Shape from Shading* [9], *Shape from Contour* [10], and *Shape from Silhouette* [11]. For representing shapes, there exist both *explicit* methods based on mesh models [12], [13], [14] and *implicit* methods based on implicit functions such as the Signed Distance Function (SDF) [15], [16]. The major advantage of SDF over explicit shape description or other implicit methods is that the normal vectors of the object surface can be easily calculated for further use, e.g., shading. Also, a fast approach exists to render the 3D geometry from an SDF [17], which is suitable for in-vehicle use cases. Therefore, we narrowed our focus to 3D reconstruction methods using SDFs for the literature review related to this work.

Based on the SDF representation, Prisacariu et al. [8] developed a method to reconstruct 3D object shapes using only a single frame. The authors first segment an object in an input image from the background and then optimize the viewpoint based on the projected silhouette of an offline-learned 3D shape prior. The use of the Gaussian Process Latent Variable Model (GPLVM) [7] for 3D reconstruction was first mentioned here. GPLVM is a dimensionality reduction method for representing complex 3D shapes using only two variables. We consider this property to be ideal for the partitioning of the object-level 3D reconstruction workflow in in-vehicle E/E architectures (Section III-A). However, we identified two significant drawbacks of [8] in the process. First, the algorithm used for differentiating foreground and background in an input image is only trained based on a few frames (5 to 7 frames as described in [8]). This resulted in limited performance for extracting objects from the background. In addition, [8] requires manual input for initializing the viewpoint for subsequent tracking, which is not practical in a car. According to us, both of the problems can be addressed using deep neural networks. Hence, we decided to combine the approach in [8] with deep-learning-based methods for object-level 3D reconstruction.

B. Deep-Learning-based 3D Object Detection

Today, deep-learning-based solutions exhibit excellent performance for a wide range of computer vision tasks, including *object detection* [18], [19], [20], [21], [22], *semantic segmentation* [23], [24], [25], *viewpoint estimation* [26], [27], [28], [29], and *depth estimation* [30], [31]. The advantage of a deep neural network lies in its capability to learn the "optimal" feature representation out of a training dataset. The learned features can even be shared for tasks that are intuitively irrelevant to humans, e.g., semantic segmentation in a 2D image and 3D viewpoint estimation.

Specifically for semantic segmentation, Long et al. [25] first introduced a Fully Convolutional Network (FCN) that classifies every pixel in an input image. The output (segmented) image has the same size as the input image. Several studies [23], [32], [33] attempted to improve the performance of FCN by combining it with Conditional Random Fields (CRFs). We choose the solution of Zheng et al. [23] for monocular 3D Shaping (Section III) for the following two reasons. First, unlike the other studies, the CRF in [23] is fully integrated into the deep neural network, enabling end-to-end training of both the segmentation network and the CRF. Second, back then, in 2015, they achieved the top rank on the PASCAL VOC 2012 image segmentation benchmark [34].

For viewpoint estimation, existing deep-learning-based approaches can be divided into two major categories: *multi-class classification* and *regression*. Tulsiani et al. [27] and Su et al. [28] modeled viewpoint estimation as a multi-class classification problem. A viewpoint angle was discretized into viewpoint bins, and a softmax loss function was used for training the neural network. Pepik et al. [26] showed, on the contrary, that it is more natural to model viewpoint estimation as a regression problem. In this paper, we investigate both variants of modeling viewpoint estimation for 3D Shaping (Section III-C).

III. Monocular 3D Shaping

Most production cars that provide AR applications are not necessarily equipped with 3D sensors, such as a stereo camera or a Lidar. Thus, we decided to approach the problem of object-level 3D reconstruction using a minimal sensor setup, viz., a monocular camera. Here, we further challenge ourselves to use only a single frame from the monocular camera as an input. If we could solve the problem without saving any raw image data from the past, then we would reduce the memory requirements of the ECU on which the 3D reconstruction is done. Besides, a light-weight tracking algorithm could be built on top of the single-frame solution without much additional effort.

Based on this design rationale, we propose a two-stage workflow for object-level 3D reconstruction that combines 3D object detection and 3D surface reconstruction techniques, as shown in Fig. 3. In the first stage, a *Region of Interest* (ROI)[1] of the input image is semantically segmented into a foreground and a background. In addition, an initial viewpoint angle is estimated. Among the many existing approaches for segmentation and viewpoint estimation, we decide to exploit a deep neural network, based on the premise that hardware accelerators for deep learning would be available in cars in the near future. Compared to traditional computer vision approaches, the benefit of deep-learning-based solutions is that there is no additional effort to tailor the trained neural networks for the target hardware in a car. Also, we can use the same neural network architecture for different object classes. Hence, we designed our workflow in a way that is easily extensible to multiple classes. In the second stage, the output of the first stage (foreground/background segmentation and viewpoint estimation) are used as initial input for jointly calculating the 3D pose and 3D shape of the object in the input image ROI. Here, we are inspired by [8] where 3D object shapes are efficiently represented using a Gaussian Process Latent Variable Model (GPLVM) [7]. Theoretically, complex 3D shapes can be represented by only two latent variables. This property of GPLVM is ideal for partition the entire workflow into different vehicle domains. Generally, the computation part of a workflow in a car resides in a domain near the sensors (e.g., ADAS domain) and the visualization part in a domain near the users (e.g., telematics domain). For augmented reality, visualizing the detected 3D objects by the users requires transmitting 3D shapes from the ADAS domain to the telematics domain. Using the latent shape representation allows the use of a low- or medium-bandwidth communication bus for this transmission, resulting in savings that are essential for the cost-sensitive automotive industry.

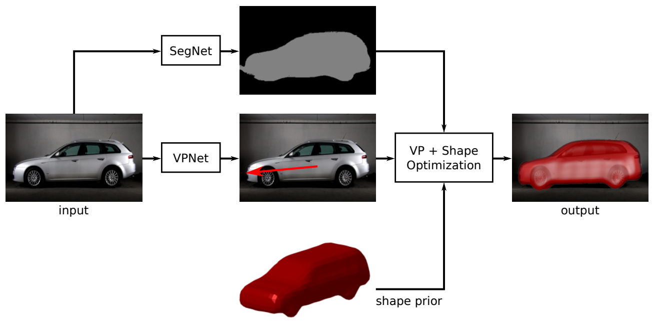


Fig. 3: Workflow of monocular 3D Shaping.

The rest of this section is structured as follows. We explain the mathematics behind the latent shape space in Section III-A and formulate the calculation of 3D shapes as an optimization problem in Section III-B. In Section III-C, we present the neural networks developed for segmentation and viewpoint

[1]Cropped out of the input image, based on the results from 3D object detection.

estimation referred to as SegNet and VPNet respectively in the rest of this paper.

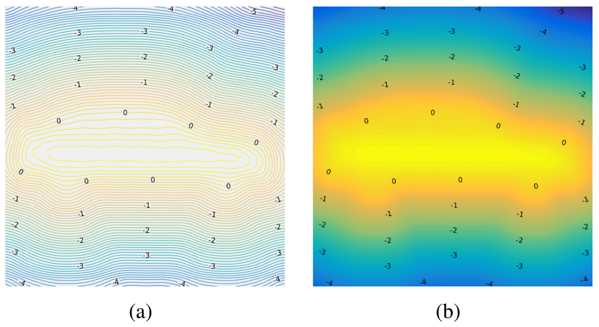
A. Latent Shape

First, 3D geometries are represented implicitly through a Signed Distance Function (SDF). The general definition of an SDF is expressed in Eq. 1, where Ω denotes a subset of a metric space with $d(\cdot, \cdot)$ being the metric function. $\partial\Omega$ and Ω^c denote the boundary and the complement set of Ω, respectively. The definition of the metric function d is expressed in Eq. 2, which returns "the nearest distance" between x and the boundary of Ω. The SDF is positive if x is "inside" Ω and negative if outside. Theoretically, an SDF is differentiable almost everywhere if Ω is in the Euclidean space.

$$\Phi(x) = \begin{cases} d(x, \partial\Omega), & \text{if} \quad x \in \Omega \\ 0, & \text{if} \quad x \in \partial\Omega \\ -d(x, \partial\Omega), & \text{if} \quad x \in \Omega^c \end{cases} \tag{1}$$

$$d(x, \partial\Omega) = \inf_{x' \in \partial\Omega} d(x, x') \tag{2}$$

Fig. 4(a) and Fig. 4(b) visualize an example of a signed distance function in $\mathbb{R}^2$. One can immediately realize that this SDF represents the silhouette of a car. The boundary of the car is implicitly embedded in the contour line that equals to zero. Hence, it is also known as the *zero level set*. A two-dimensional SDF can represent the 2D boundary of an object in an image. With an additional dimension, the surface of a 3D geometry can be represented, as shown in Fig. 4(c).


(a) (b)

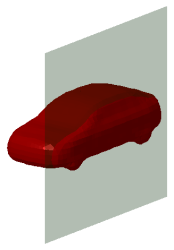

(c)

Fig. 4: Visualization of SDF. (a) and (b) 2D SDF representing the silhouette of a car. (c) 3D SDF representing a car. In fact, the 2D SDF in (a) and (b) is a slice cut out from the 3D SDF in (c).

In practice, an SDF is discretized and stored as a floating-point array. Each element of the array records the distance from its grid location to the zero-level. The higher the dimension of the array is, the finer is the discretization of the original geometry. Fig. 5 illustrates how the fineness of an SDF is affected by the dimension of the array container. As can be observed in Fig. 5, more delicate details in the concave parts of the original 3D model (mirrors, wheels) are not captured in the SDFs.

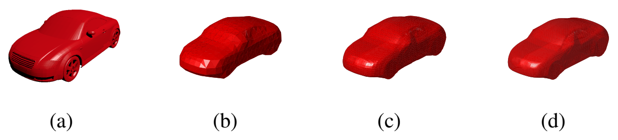

(a) (b) (c) (d)

Fig. 5: Different levels of fineness of an SDF. (a) Original 3D model. (b), (c), and (d) 3D SDFs of the original model with 20^3, 60^3, and 120^3 entries, respectively.

Although an SDF is able to embed a 3D geometry implicitly, it cannot be directly used as a feature for classification or other machine-learning-based algorithms. The reason is that an SDF is a representation of the original 3D geometry in the spatial domain. A machine learning algorithm typically needs to operate in a so-called feature domain, where sample points that belong to different classes are separated. This is the same reason why we would not directly use RGB value as a feature for object recognition in an image. Therefore, a three-dimensional Discrete Cosine Transform (DCT) is applied, which transforms an SDF into the frequency domain to make it feature-rich.

The number of DCT coefficients is the same as the number of the elements in a discretized SDF. Their sizes increase cubically with the dimension of the array. Dealing with an extremely high dimensional feature would require more computational power and slow down the entire workflow. However, if we cut down the dimension of an SDF array, the quality of 3D reconstruction needs to be compromised. Here, we take advantage of the property of DCT that centralizes low-frequency components. This allows us to discard small high-frequency coefficients without significantly affecting the reconstruction quality, as shown in Fig. 6. The difference between the reconstructions using 60^3 coefficients in Fig. 6(a) and 40^3 coefficients in Fig. 6(b) is barely visible through naked eyes.

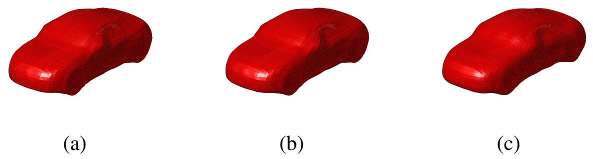

(a) (b) (c)

Fig. 6: Reconstruction with (a) 60^3, (b) 40^3, and (c) 20^3 DCT coefficients, respectively.

After collecting a small number of 3D geometries (training samples) and transforming them into DCT coefficients (features), a Gaussian Process Latent Variable Model (GPLVM) training is carried out. In general, GPLVM is a dimensionality reduction method that uses a Gaussian process to learn a low-dimensional representation (latent variables) of high-dimensional data (features). In the context of 3D shape representation, GPLVM has the following advantages. First, it only requires a small number of training samples (the nature of a Gaussian process). Second, it can significantly reduce the dimension of the feature space without any information loss. Last but not least, the low-dimensional latent space is continuous and differentiable everywhere, which is mathematically convenient for optimization purposes.

Mathematically, GPLVM training aims at finding a Gaussian process (Eq. 3) that maps a low-dimensional latent space X to a high dimensional feature space Y. K denotes a kernelized covariance matrix of the Gaussian distribution, with each element $k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

$$Y \sim \mathcal{N}(\mathbf{0}, K) \tag{3}$$

The kernel function $\kappa(\cdot, \cdot)$ is typically a radial basis function plus some white noise, as expressed in Eq. 4.

$$\kappa(\mathbf{x}, \mathbf{x}') = \theta_1 \exp\left(-\frac{\theta_2}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right) + \theta_3 + \theta_4\delta(\mathbf{x}, \mathbf{x}') \tag{4}$$

The so-called hyper-parameters $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$ and the set of latent variables X are the variables to be optimized during GPLVM training. The optimization is formulated as a Maximum Likelihood Estimation (MLE) problem in Eq. 5. During GPLVM training, we try to find the latent variables X that are most likely to generate the given training set Y through a Gaussian process controlled by the hyper-parameters Θ.

$$(X^*; \Theta^*) = \underset{X;\Theta}{\arg\max}\; p(Y|X; \Theta) \tag{5}$$

In practice, the negative log-likelihood $L = -\ln p(Y|X; \Theta)$ is minimized through a Scaled Conjugate Gradient (SCG) [35] method. Fig. 7 shows a latent space that embeds 3D geometries of cars, with an extremely low latent dimension of $q = 2$. The transition of 3D geometries inside the latent shape space is smooth everywhere, as shown in Fig. 8. Theoretically, the latent shape space is able to span a space that covers all 3D geometries in the training set. In other words, if different car types like trucks or buses are included for latent space training, they can also be reconstructed later during recall.

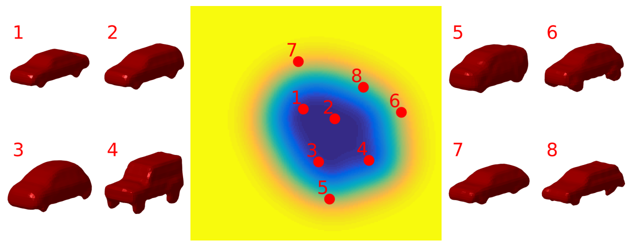


Fig. 7: A two-dimensional latent space that embeds 3D geometries of cars. Latent points in the dark area are considered to be able to generate more car-like 3D shapes.

Having a trained latent space, one can infer a feature point $\mathbf{y}^*$ of an arbitrary latent variable $\mathbf{x}^*$ by the definition of Gaussian processes (Eq. 6), where $K^* = [\kappa(\mathbf{x}^*, \mathbf{x}_1), \kappa(\mathbf{x}^*, \mathbf{x}_2), \cdots]^T$ and $K^{**} = \kappa(\mathbf{x}^*, \mathbf{x}^*)$.

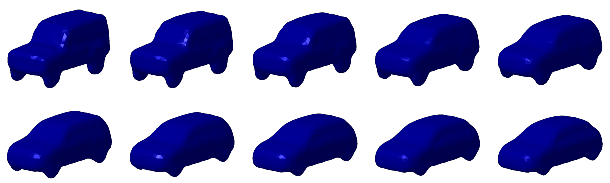


Fig. 8: A smooth transition of latent shapes.

$$\left[\frac{Y}{\mathbf{y}^*}\right] \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{c|c} K & K^* \\ \hline (K^*)^T & K^{**} \end{array}\right]\right) \tag{6}$$

Therefore, the expectation and the variance of $\mathbf{y}^*$ can be estimated using Eq. 7.

$$\begin{aligned} E(\mathbf{y}^*) &= \kappa(\mathbf{x}^*, X)K^{-1}Y \\ Var(\mathbf{y}^*) &= \kappa(\mathbf{x}^*, \mathbf{x}^*) - \kappa(\mathbf{x}^*, X)^T K^{-1}\kappa(\mathbf{x}^*, X) \end{aligned} \tag{7}$$

The inference from a low-dimensional latent variable back to the original feature space is referred to as GPLVM recall.

Now, the path between a 3D geometry and a latent variable is completed, as illustrated in Fig. 9. 3D geometries go through distance transformation, DCT, and GPLVM training to become latent variables in the latent shape space. A latent variable generates a 3D SDF through GPLVM recall and Inverse DCT (IDCT). The triangulation of a 3D SDF is realized using the Marching Cube [17] algorithm.

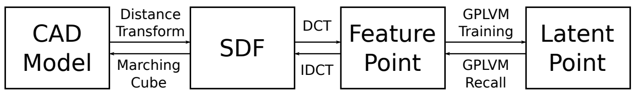


Fig. 9: From latent variable to 3D geometry.

B. Optimization Problem

After the latent shape space is trained, the problem of 3D reconstruction can be reformulated as an optimization problem to find the latent variables that generate the original 3D shape. Fig. 10 illustrates an example of how this optimization works. The screen at the left side shows a semantically segmented input image, with blue pixels illustrating a car. On the right side, a 3D geometry of a car is projected onto the screen. The objective of the optimization is to find the best position, orientation, and 3D shape of the car so that the silhouette of the car on the screen matches the segmented output as close as possible.

Mathematically, the energy function in Eq. 8 is maximized during 3D shape optimization. Ω denotes a region of interest in an image and $\mathbf{x}$ an image pixel. $P_f(\cdot)$ and $P_b(\cdot)$ record how likely a pixel belongs to the foreground (the target object) or the background. They are estimated using a segmentation neural network (SegNet). $\pi(\cdot; \cdot)$ denotes a projection function that projects a 3D SDF denoted as Φ onto the image plane. The projection is controlled by the parameter set ρ. More specifically, ρ comprises three parameters for the translation, four parameters[2] for the rotation, one parameter for the scale,

[2]Quaternion representation.

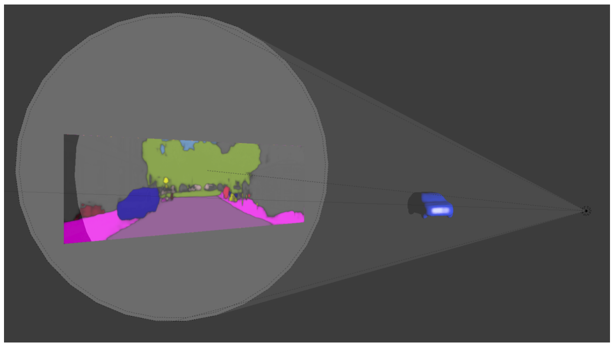


Fig. 10: Visualization of 3D shape optimization.

and two additional latent variables for the 3D geometry. Relying on the latent shape representation, we only need to solve two additional variables in the latent space instead of, e.g., $40^3 = 64,000$ variables in the original feature space.

$$E(\Phi;\rho) = \sum_{\mathbf{x}\in\Omega} P_f(\mathbf{x})\pi(\Phi;\rho) + P_b(\mathbf{x})[1-\pi(\Phi;\rho)] \quad (8)$$

The energy function is differentiable with respect to the latent variable and the viewpoint parameters. Thus, it can be optimized through standard non-linear optimization methods, such as gradient descent or Levenberg-Marquardt [36]. For more details of the derivatives on the energy function, we refer the reader to [37].

We conducted different experiments under controlled conditions to examine the influence of latent variables and viewpoint parameters on the energy function. First, we render a car's dummy image from a fixed viewpoint and calculate the energy value at each point in the latent shape space. The result is visualized in Fig. 11(a), which clearly shows an area of global maxima that can be reached through gradient-based optimization methods. Second, we fix the latent shape and unlock the orientation angle to change the viewpoint. In other words, we rotate the camera around the car and calculate the energy at each angle. Fig. 11(b) illustrates the result. We observe two local maxima on opposite sides of the circle, i.e., there is a 50 percent chance that the optimization converged at the "wrong" local maximum if we randomly initialize the viewpoint at the beginning of the optimization. Last but not least, to show the ambiguity between depth and scale, we unlock the longitudinal distance and the scale parameter while fixing the others. A ridge-like structure can be observed in the result shown in Fig. 11(c). This is quite straightforward to understand since a big car that is far away might look similar in an image to a small car right in front of us. The ambiguity between depth and scale will not disappear if only one image is used during the optimization. For pure visualization purposes such as video overlay for augmented reality, this ambiguity is acceptable.

C. SegNet and VPNet

For monocular 3D Shaping, we assume that the object bounding boxes are given in an earlier processing step, i.e., each input image contains only one object. For the segmentation network (SegNet), we use a Fully Convolutional Network (FCN) [25] combined with a Conditional Random Field (CRF), as suggested in [23]. This produces a sharp-edged foreground-background segmentation, which is essential to the performance of the subsequent 3D shape optimization. The architecture of the FCN is shown in Fig. 12. The output of the FCN is fused from the last three pooling layers and sampled up to the same size as the input image. A coarse level segmentation output from the FCN is then refined through the conditional random field, which yields fine and sharp-edged segmentation. The finer the segmentation results are, the better is the quality of the reconstructed 3D shape. According to [23], the performance of this approach on the object class *Car* achieved 82.4% mean IoU (Intersection over Union) evaluated using the Pascal VOC 2012 dataset, which proved to outperform the state-of-the-art back then.

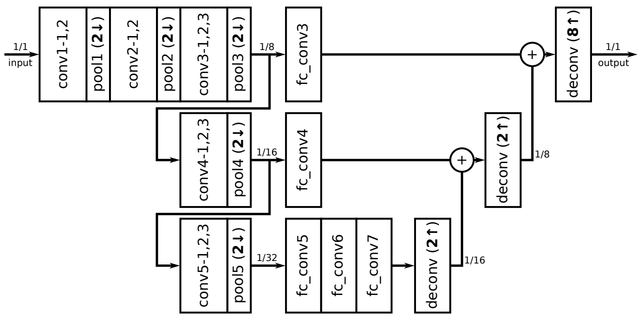


Fig. 12: Network architecture of the FCN. Only convolution layers and pooling layers are illustrated. The architectural parameters of the network remain the same as those of a VGG16 network [38].

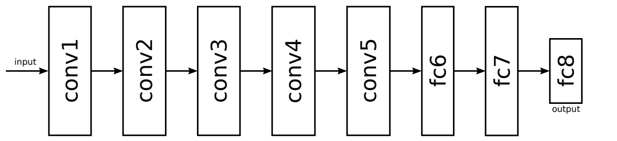
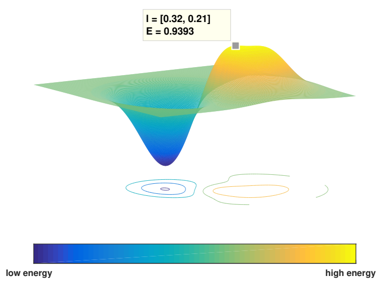


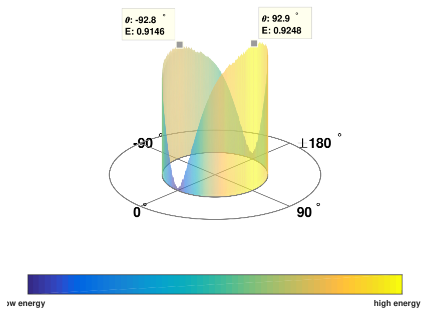
Fig. 13: Network architecture of the viewpoint classification network, which resembles an AlexNet except for the output layer `fc8`. Pooling, ReLu, and dropout layers are omitted in the illustration.

In addition, we train a separate neural network specifically for viewpoint initialization. The network architecture is adapted from AlexNet [18], as shown in Fig. 13. By modifying the last fully connected layer of the AlexNet, we can model the problem of viewpoint estimation differently from a mathematical point of view. Here, we use two types of loss functions: the softmax loss expressed in Eq. 9 and the Euclidean loss in Eq. 10. Using the softmax loss function, we formulate viewpoint estimation as a classification problem, while we define it as a regression problem using Euclidean loss.

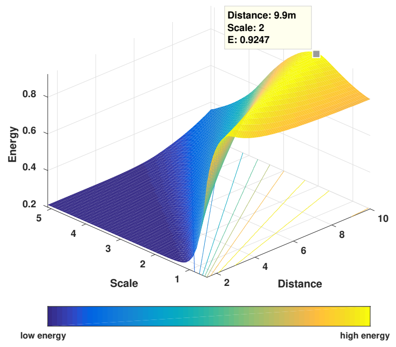
$$L = -\frac{1}{N}\sum_{n=1}^{N} \ln\left(\frac{e^{x_{l_n}}}{\sum_{k=1}^{K} e^{x_k}}\right) \quad (9)$$



(a) Latent variable.



(b) Orientation angle.



(c) Depth and scale.

Fig. 11: Energy space exploration for monocular 3D Shaping.

In Eq. 9, N denotes the number of training samples, K the total number of viewpoint bins, l the groundtruth viewpoint label, and x the predicted score. We tried different numbers of viewpoint bins. Generally, using a larger K increases the precision of the estimated viewpoint angle.

$$L = \frac{1}{2N}\sum_{n=1}^{N} \|\hat{x}_n - x_n\|^2 \tag{10}$$

The Euclidean loss in Eq. 10 turns the neural network into a regressor. $\hat{x}_n$ denotes the groundtruth and x_n the predicted viewpoint, respectively. The Euclidean distance between the groundtruth and the neural network output is minimized during the training process.

We used a subset of ImageNet with viewpoint annotations [39] for training the VPNet. A layer-separation for different classes was not considered since for monocular 3D Shaping, we specifically focused on *car* objects. All training processes were carried out using the Caffe deep learning framework [40].

IV. 3D Shaping+Lidar

Future cars with autonomous driving functions will be equipped with more powerful 3D sensors such as Lidars or stereo cameras. These can be used to make the results of object-level 3D reconstruction more precise, and enable applications like automated parking (already available in high-end cars today). In particular, the ambiguity problem (Fig. 11(c)) of monocular 3D Shaping ceases to exist if 3D measurements of the environment are directly used. Also, by taking advantage of deep learning-based 3D object detection techniques such as [41], the scene can be clustered in 3D before being fed into the 3D Shaping pipeline. In other words, the self-occlusion issue of monocular 3D Shaping (Fig. 18) can be resolved. Therefore, in this section, we propose another variant of 3D Shaping, referred to as *3D Shaping+Lidar* in the rest of this paper. This is a resource-efficient alternative to monocular 3D Shaping, which requires less computing resources on the ECU in exchange of using more expensive sensors.

The workflow of 3D Shaping+Lidar is similar to that of monocular 3D Shaping. Here, we assume that the 3D scene is already clustered into object clusters. For each object, an image ROI and a point cloud cluster are taken as the input to the proposed 3D Shaping+Lidar workflow. We use the point cloud as an additional constraint for the 3D shape optimization, which yields more precise reconstruction results in 3D than monocular 3D Shaping. The updated energy function is explained in Section IV-A.

A. Energy and Optimization

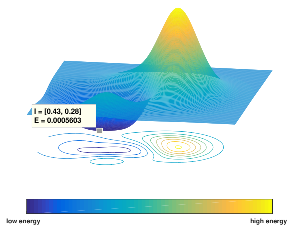
The new energy function for 3D Shaping+Lidar (Eq. 11) is a weighted combination of the image-based energy E_{img} (Eq. 8) and a point cloud energy (Eq. 12). Φ denotes an SDF that encodes 3D geometries, and ρ denotes a set of pose parameters. This energy function was first used in [42] for similar purposes.

$$E(\Phi;\rho) = E_{img}(\Phi;\rho) + \lambda E_{cloud}(\Phi;\rho) \tag{11}$$

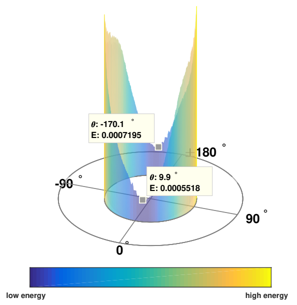
The additional cloud energy is formulated in Eq. 12, where $\mathbf{X}^L$ denotes a 3D point in a Lidar cluster $\mathcal{L}$, and $g \in SE(3)$ the Lie algebra that transforms a point from Lidar coordinates to the object geometry coordinates. The object geometry coordinates system is attached to the geometrical centroid of an object. The exponential term of the Geman-McClure function [43] reaches its minimum if $\mathbf{X}^O$ lies exactly on the zero-level of the SDF Φ. It increases monotonically with the distance between $\mathbf{X}^O$ and the object surface, and the increasing rate is controlled by σ. This energy function was also used in [42].

$$\begin{aligned} E_{cloud}(\Phi;\rho) &= \sum_{\mathbf{X}^L\in\mathcal{L}} \exp\left\{\frac{\Phi^2(g(\mathbf{X}^L;\rho))}{\Phi^2(g(\mathbf{X}^L;\rho))+\sigma}\right\} \\ &= \sum_{\mathbf{X}^O} \exp\left\{\frac{\Phi^2(\mathbf{X}^O)}{\Phi^2(\mathbf{X}^O)+\sigma}\right\} \end{aligned} \tag{12}$$

In our case, to examine the properties of the point cloud energy, we conducted similar experiments under controlled conditions as in Section III-B. We generated a dummy point cloud cluster by random sampling the surface of an SDF of a car and used the dummy cluster to simulate Lidar measurements. First, we fixed the viewpoint and varied the latent variables within a trained latent shape space. Second, we fixed the latent shape and altered the orientation angle. The results are shown in Fig. 14(a) and Fig. 14(b), respectively. We observed similarities between the point cloud energy and image-based energy. There is a unique global minimum when varying latent variables, while a 180°-flipped local minimum occurs when changing the orientation angle.

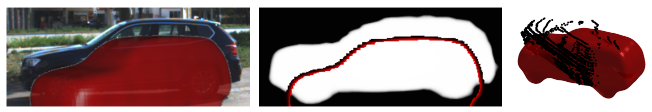


(a) Latent variable.



(b) Orientation angle.

Fig. 14: Energy space exploration for 3D Shaping+Lidar.


(a) Initial state.

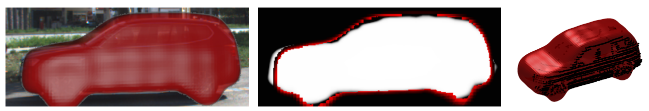

(b) Final state.

Fig. 15: 3D Shaping optimization using image and Lidar measurements. Image statistics are illustrated in the middle column, with white pixels being more likely to be foreground pixels. The right column shows SDFs and Lidar measurements.

In practice, the image energy and the point cloud energy can be jointly optimized using gradient-based optimization methods. Fig. 15 shows an example to help understand such optimization processes. Lidar points would "stick" to the surface if the optimization converges ideally.

V. Experimental Results

A. Results of Monocular 3D Shaping

We chose viewpoint accuracy as the metric to evaluate monocular 3D Shaping. The reason is that for in-vehicle augmented reality, viewpoint accuracy is considered one of the most critical factors that affect user experience.

The viewpoint error is measured by the geodesic distance over the rotation sphere, as expressed in Eq. 13, where R_{est} denotes the estimated rotation matrix and R_{gt} the groundtruth. We calculated the median viewpoint error $Med(\bullet)$ over the entire evaluation dataset. In addition, we show the percentage of accurate estimates $Acc(< \theta)$, with θ being the angular threshold. The results are presented in Tab. I.

$$\delta_{vp}(R_{est}, R_{gt}) = \frac{1}{\sqrt{2}} \| \ln(R_{est}^T R_{gt}) \|_F \tag{13}$$

The first half of Tab. I shows the results after the first stage, and the second half after the second stage of the 3D Shaping workflow. The network using softmax loss for viewpoint estimation is denoted by VPNet-Kbin, with K indicating the number of viewpoint bins. The viewpoint regression network using the Euclidean loss function is denoted by VPNet-Reg. VDPM-24bin stands for Viewpoint Deformable Part Model [39] with 24 viewpoint bins, which was considered the state-of-the-art before deep-learning-based algorithms were introduced.

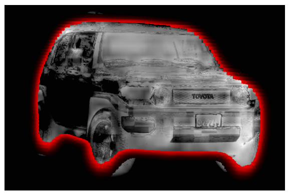
TABLE I: Evaluation of viewpoint estimation by monocular 3D Shaping.

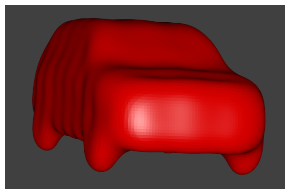
Approach	$Med(\delta_{vp})$	$Acc(30°)$	$Acc(15°)$	$Acc(5°)$
VDPM-24bin	19.60°	67.51%	35.57%	8.16%
VPNet-Reg	23.11°	58.74%	34.75%	12.34%
VPNet-24bin	11.82°	86.77%	64.29%	19.26%
VPNet-72bin	7.48°	**90.47%**	73.61%	35.23%
VPNet-24bin+Shape	5.47°	88.04%	**78.35%**	47.08%
VPNet-72bin+Shape	**4.29°**	89.13%	77.30%	**54.13%**

The following three points can be summarized from the results in Tab. I. First, the viewpoint network with the most viewpoint bins outperforms the other approaches. Second, the estimated viewpoint angles are more close to the groundtruth after the second stage optimization. There is a nearly 20 percent performance gain in $Acc(< 5°)$ by VPNet-72bin+Shape. Last but not least, the optimization process is proven to be robust against initialization to some extent, as VPNet-24bin+Shape and VPNet-72bin+Shape yield similar results.

Fig. 16 shows a reconstructed 3D car overlaid on the image, as an example use case of 3D Shaping for in-vehicle augmented reality. The foreground possibilities estimated through the segmentation network are illustrated in Fig. 16(b), ranging from zero (black) to one (white). The red frame in Fig. 16(b) shows the outer contour of the projected 3D geometry.

(a)


(b)


(c)

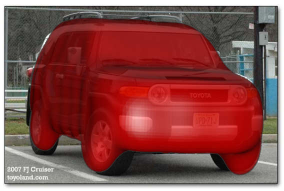
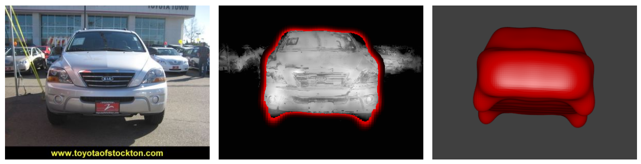
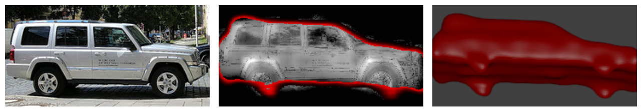

(d)

Fig. 16: Example of satisfying 3D reconstruction. (a) Input image. (b) Segmentation score. (c) 3D shape recovered. (d) 3D shape overlay.

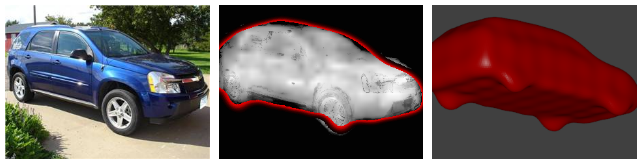
Three failed reconstructions due to ambiguous silhouette (Fig. 17(a)), divergence in the latent shape space (Fig. 17(b)), and poor viewpoint initialization (Fig. 17(c)) are presented in Fig. 17. More reconstruction results of 3D Shaping are shown in Fig. 22.



(a) Ambiguous silhouette.

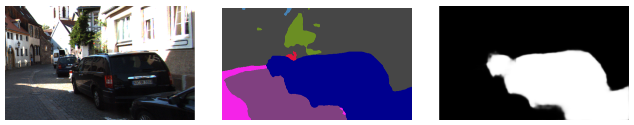


(b) Divergence in the latent shape space.



(c) Bad viewpoint initialization.

Fig. 17: Failed 3D reconstructions.



(a) (b) (c)

Fig. 18: Occlusion problem by monocular 3D Shaping. (a) Original image patch. (b) Segmented image. (c) Probability of car pixels. The optimizer would try to cover the entire white area in (c) with only one instance of the object class car.

Certainly, the second stage 3D shape optimization is subject to viewpoint initialization and image segmentation. Additional constraints such as discriminant keypoints could be introduced to solve the problem of ambiguity shown in Fig. 17(a). For example, wheels or side mirrors could be used to identify the viewpoint. Another idea would be to use the shared latent space that associates latent shape variables and keypoints. However, compared to deep neural networks, it is not easy to parallelize these algorithms and efficiently implement them on an in-vehicle platform. Also, as already mentioned at the beginning of Section III, a light-weighted tracking algorithm could be built upon the single frame solution to solve the problem of viewpoint ambiguity.

Nevertheless, the workflow presented in this section is designed for augmented reality use cases, e.g., highlight a single object standing in front of a clean background. Multiple instances of the same object class would cause a self-occlusion problem, as illustrated in Fig. 18. Without knowing the exact number of cars in the scene and the boundaries between them, the optimizer in the second stage is unable to deliver reasonable estimations of the viewpoint and 3D shapes. Using additional inputs from 3D sensors (e.g., a Lidar) could solve this issue.

B. Results of 3D Shaping+Lidar

For 3D Shaping+Lidar, we use the accuracy of orientation angle and occupancy bounding box in 3D to evaluate the performance. These two metrics are most relevant to the target use case – automated parking. We use a more challenging dataset for the evaluation, namely the KITTI vision benchmark [44], which is widely used in the context of autonomous driving. We selected representative sequences from the raw recordings, which cover a wide range of scenarios, such as city, road, and residential. More details of the evaluation dataset are given in Tab. II.

TABLE II: Details of the evaluation dataset. From left to right: Sequence name of KITTI raw recordings, data category, number of *Car* tracklets, number of image patches of *non-occluded* cars at short range (up to 20 meters away from the ego-car) and full range, respectively. The sequences were recorded on Sep. 26, 2011.

Sequence	Category	Tracklet	Short Range	Full Range
0001	city	12	30	129
0002	city	1	0	11
0005	city	9	42	48
0009	city	89	246	592
0011	city	15	30	276
0013	city	8	100	195
0014	city	26	38	348
0015	road	33	44	746
0017	city	4	13	13
0018	city	11	137	336
0019	residential	13	135	283
0020	residential	5	0	0
0022	residential	53	452	960
0023	residential	150	288	668
0027	road	3	9	32
0028	road	9	29	139
0029	road	3	21	75
0032	road	21	67	793
0035	residential	23	128	382
0036	residential	80	200	363
0039	residential	35	276	689
0046	residential	8	35	35
0048	city	7	32	61
0051	city	26	114	557
0052	road	4	27	43
0056	city	13	35	444
0057	city	22	1	877
0059	city	52	439	1082
0060	city	2	0	128
0061	residential	39	217	639
0064	residential	38	100	226
0070	road	2	8	50
0079	residential	2	0	0
0084	city	49	261	988
0086	residential	3	39	74
0087	residential	6	50	235
0091	city	2	0	0
0093	city	54	322	634
	sum	932	3965	13151

Note that the metrics to be calculated for the evaluation strongly depends on the viewpoint initialization for the 3D Shape optimization, as already discussed in Section V-A. In order to better evaluate the added value of the proposed 3D Shaping+Lidar to existing approaches, we decided to decouple the influence of viewpoint initialization by manually adding a normal-distributed bias $\mathcal{N}(15°, 15°)$ to the groundtruth orientation. This bias reflects the upper bound performance of the top-ranking approaches in the KITTI vision benchmark.

Table III and Fig. 20 present the evaluation results. In

Tab. III, $Med(\delta)$ shows the median of the absolute estimation error δ in degrees, and $Acc(< \theta)$ indicates the percentage of accurate estimates that are smaller than the threshold angle θ. According to the results, both 3D Shaping approaches – using and without using a Lidar – can correct viewpoint angle within a certain initialization error. At short range, the approach using a Lidar is able to correct an error more than 30° and at full range up to 20°. This clearly shows the improvement when using a Lidar.

TABLE III: Evaluation of 3D Shaping in viewpoint estimation.

	Approach	$Med(\delta)$	$Acc(< 20°)$	$Acc(< 10°)$	$Acc(< 5°)$
short	Initialization	16.07°	61.23%	31.80%	15.15%
	Orientation Net	12.77°	63.00%	42.12%	23.61%
	Shaping Cam	13.96°	65.67%	37.95%	21.64%
	Shaping Lidar	**7.02°**	85.45%	67.53%	37.99%
full	Initialization	16.01°	62.08%	31.87%	15.49%
	Orientation Net	20.44°	49.42%	30.61%	16.57%
	Shaping Cam	16.08°	60.24%	32.59%	17.91%
	Shaping Lidar	14.14°	61.50%	38.44%	20.22%

The occupancy bounding box is defined as the rectangular bounding box around an object from a bird's-eye view. The bounding box overlap ratio is given by Eq. 14, which calculates the Intersection over Union (IoU) of two bounding boxes in 3D. We present the evaluation results of bounding box accuracy in Tab. IV and Fig. 20. In Tab. IV, $Mean(Ovl)$ denotes the average of the overlap ratios. $Acc(> \bullet)$ indicates the percentage of accurate estimates larger than a certain threshold of overlap ratio. An estimated occupancy bounding box is considered correct if the overlap ratio is larger than 0.5. According to the results, the approach using a Lidar significantly improves the bounding box estimation performance compared to the monocular approach, due to the fact that the ambiguity between distance and scale is eliminated by directly using 3D measurements.

$$Ovl(A, B) = \frac{A \cap B}{A \cup B} \tag{14}$$

TABLE IV: Evaluation of 3D Shaping in bounding box estimation.

	Approach	$Mean(Ovl)$	$Acc(> 0.5)$	$Acc(> 0.7)$
short	Shaping Cam	0.17	8.34%	1.97%
	Shaping Lidar	0.69	**90.57%**	62.77%
full	Shaping Cam	0.11	5.67%	1.21%
	Shaping Lidar	0.53	58.36%	28.87%

We also evaluated the absolute translation error, as presented in Tab. V and Fig. 20. In Tab. V, Med and Std denote the median and the standard deviation of the absolute translation errors. We observe a significant improvement in estimating longitudinal distance for 3D Shaping using a Lidar compared to monocular 3D Shaping.

TABLE V: Evaluation of 3D Shaping in translation estimation.

	Approach	Med (m)		Std (m)	
		longitudinal	lateral	longitudinal	lateral
short	Shaping Cam	2.724	0.522	1.751	0.805
	Shaping Lidar	**0.157**	**0.085**	0.376	0.525
full	Shaping Cam	3.856	0.494	2.540	0.982
	Shaping Lidar	0.359	0.106	0.702	0.655

Fig. 19 gives an example of how occupancy estimation is improved by using a Lidar. It is quite common in a traffic scene that only a part of the object surface is captured by the Lidar, yielding an L-shape or I-shape measurements in the resulting point clouds when looking from a bird's-eye view. The reasons for this include reflective surface material, occlusion, and clustering errors. Relying on our prior knowledge about 3D geometries of a specific object class, e.g., a car, the estimated occupancy bounding box is more close to the groundtruth even if the point cloud cluster is sparse and incomplete.

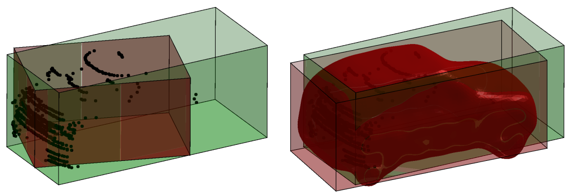


Fig. 19: Improvement in occupancy estimation for 3D Shaping. The left image shows the bounding box (dark red) that is directly obtained from a Lidar cluster. The right image shows the estimated bounding box (red) using 3D Shaping+Lidar. The green bounding box illustrates the groundtruth.

Figure 2 shows a rendered result when heavy self-occlusion occurred. Using a Lidar allows us to cluster the scene directly in 3D and, thus, to reconstruct multiple instances within the same class that are occluded by each other. This enables us to build a 3D environmental model at the object level, which is essential for near-future in-vehicle features such as augmented reality and autonomous driving.

VI. System Integration

Fig. 21 demonstrates our design proposal to integrate the proposed 3D Shaping workflow into a future in-vehicle E/E system. The computation and visualization of the workflow are distributed into the ADAS domain and the telematics domain. For each new object, the optimizer would first minimize a combined energy function (Eq. 8 and Eq. 12). Then, it would send out the optimized latent shape variables to the sensor fusion interface, where they would be tracked according to the shape consistency. The sensor fusion interface would pack the two additional latent variables into an object list and send the list to the telematics gateway through the vehicle backbone bus. The payload data to be transmitted on the backbone would only increase by the size of *two floating-point numbers* multiplied by the number of tracked objects.

We analyzed the payload requirements for transmitting 3D geometries through in-vehicle communication buses. Assuming that the 3D shaping optimizer operates at 25 frames

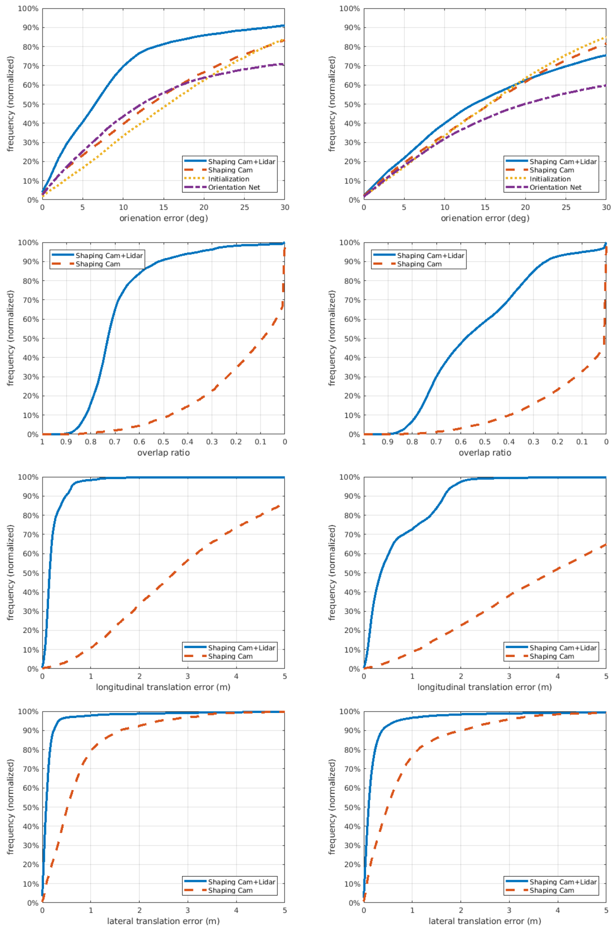


Fig. 20: Evaluation of 3D Shaping+Lidar. Left: short range objects. Right: all objects. From top to bottom: absolute orientation error, overlap ratio of occupancy bounding box, absolute longitudinal error, and absolute lateral error. The vertical axis indicates the percentage of estimates that are smaller or larger than the corresponding threshold on the horizontal axis.

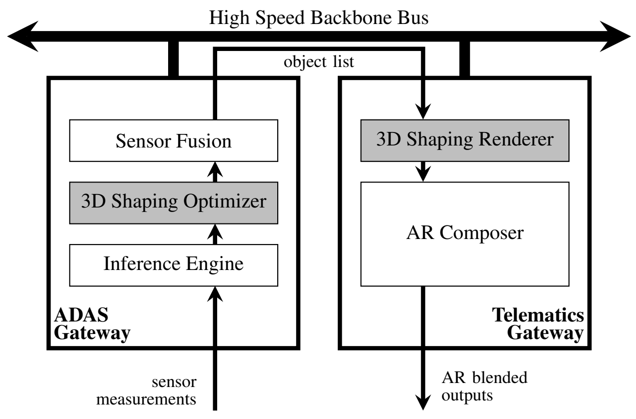


Fig. 21: Integration of 3D Shaping into a future in-vehicle E/E architecture.

per second, and at most 32 objects can be simultaneously tracked, we calculated the required payload using different representations of object geometry. The size of the SDF is set to be $40\times40\times40$, and a single-precision floating-point is used. The result is presented in Tab. VI, where FE indicates Fast Ethernet with a transfer rate of 100 Mbps, and N-GE indicates Gigabit Ethernet with N Gbps transfer rate. The result clearly shows the advantage of using latent shape representation. In fact, a vehicle backbone will be filled with all different signals for inter-domain communication. Every free byte on the backbone is considered a critical resource. 3D Shaping is a practical solution to realize 3D object-level reconstruction in a car, with a minimum sensor requirement of only one monocular camera. Therefore, it is clear that 3D Shaping is a potential candidate for new applications such as augmented reality and autonomous driving.

TABLE VI: Payload analysis for transmitting 3D geometries.

	SDF	Face + Vertex	Latent Shape
Single Object	250 KB	$\sim$ 73 KB	8 B
Required Payload	$\sim$ 195 MB/s	$\sim$ 57 MB/s	6.25 KB/s
Required Backbone	10-GE	1-GE	FE
Cost	High	Medium	Low

Regarding the run-time efficiency of the entire workflow, we consider it inconclusive in the current state of development to measure the end-to-end run-time performance for the following reasons. First, the proposed 3D Shaping software runs on a desktop workstation installed in a prototype vehicle, while the system inputs (cameras and Lidars) are already highly optimized close-to-production sensors. It is not meant to measure the run-time in such a *mixed* system setup. Second, for production-level maturity, the proposed 3D Shaping software will be optimized for the target hardware, which will significantly increase its run-time performance. Finally, the proposed partitioning of the entire workflow (Fig. 21) will also influence run-time efficiency.

VII. Concluding Remarks

In this paper, we presented a novel workflow named 3D Shaping that enables *in-vehicle object-level 3D reconstruction*. We proposed two variants that focused on two specific uses cases, viz., augmented reality and automated parking. For augmented reality, we proposed a cost-efficient solution that only requires a single frame from a monocular camera as input. For automated parking, we proposed a resource-efficient alternative that generates more precise reconstruction results by additionally using 3D sensors. We took advantage of the latent shape representation of geometries, which only requires two additional parameters for reconstructing various 3D shapes. This enables partitioning the proposed 3D Shaping workflow into different vehicle domains, making its implementation in mass-production cars practical and cost-effective.

Although the presented 3D Shaping solutions have already achieved satisfying reconstruction results, there are still possible improvements that could be taken into consideration in the future. Instance-aware segmentation could be applied further to increase the quality of 3D reconstruction for cluttered scenes. An end-to-end neural network that directly reconstructs a 3D geometry from an image would also be of interest in future research.

References

[1] Waymo LLC, "Waymo Safety Report," 2018. [Online]. Available: https://waymo.com/safety

[2] J. Dickmann, N. Appenrodt, J. Klappstein, H.-L. Blöcher, M. Muntzinger, A. Sailer, M. Hahn, and C. Brenk, "Making Bertha See Even More: Radar Contribution," *IEEE Access*, vol. 3, pp. 1233–1247, 2015.

[3] U. Franke, D. Pfeiffer, C. Rabe, C. Knöppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making Bertha See," in *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2013, pp. 214–221.

[4] R. B. Rusu, N. Blodow, Z. C. Marton, A. Soos, and M. Beetz, "Towards 3D Object Maps for Autonomous Household Robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3191–3198.

[5] R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 3601–3608.

[6] W. Zimmermann and R. Schmidgall, *Bussysteme in der Fahrzeugtechnik – Protokolle und Standards [Bus Systems in Vehicle Technology – Protocols and Standards]*. Vieweg, 2011.

[7] L. Lawrence, "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models," *The Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[8] V. Prisacariu, A. Segal, and I. Reid, "Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction," in *Proceedings of Asian Conference on Computer Vision (ACCV)*, 2012, pp. 593–606.

[9] B. Horn, *Shape from Shading*. Cambridge, MA: MIT Press, 1989, ch. 4.

[10] R. Horaud and M. Brady, "On the Geometric Interpretation of Image Contours," *Artificial Intelligence*, vol. 37, no. 1-3, pp. 333–353, 1988.

[11] A. Laurentini, "The Visual Hull Concept for Silhouette-based Image Understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 150–162, 1994.

[12] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models – Their Training and Application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[13] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape Completion and Animation of People," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 408–416, 2005.

[14] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler, "Revisiting 3D Geometric Models for Accurate Object Shape and Pose," in *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2011, pp. 569–576.

[15] R. Sandhu, S. Dambreville, A. Yezzi, and A. Tannenbaum, "Non-rigid 2D-3D Pose Estimation and 2D Image Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 786–793.

[16] C. Y. Ren and I. Reid, "A Unified Energy Minimization Framework for Model Fitting in Depth," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2012, pp. 72–82.

[17] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[18] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[19] R. B. Girshick, "Fast R-CNN," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[20] S.-Q. Ren, K.-M. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[21] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[22] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[23] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z.-Z. Su, D.-L. Du, C. Huang, and P. H. S. Torr, "Conditional Random Fields as Recurrent Neural Networks," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1529–1537.

[24] B. Hariharan, P. Arbelàez, R. B. Girshick, and J. Malik, "Simultaneous Detection and Segmentation," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014, pp. 297–312.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[26] B. Pepik, M. Stark, P. Gehler, T. Ritschel, and B. Schiele, "3D Object Class Detection in the Wild," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 1–10.

[27] S. Tulsiani and J. Malik, "Viewpoints and Keypoints," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1510–1519.

[28] H. Su, C. R. Qi, Y.-Y. Li, and L. Guibas, "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2686–2694.

[29] L. Beyer, A. Hermans, and B. Leibe, "Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels," in *Proceedings of German Conference on Pattern Recognition (GCPR)*, 2015, pp. 157–168.

[30] F. Liu, C. Shen, and G. Lin, "Deep Convolutional Neural Fields for Depth Estimation from a Single Image," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5162–5170.

[31] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2650–2658.

[32] B. Hariharan, P. Arbeláez, R. B. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-grained Localization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 447–456.

[33] G.-S. Lin, C.-H. Shen, A. van dan Hengel, and I. Reid, "Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3194–3203.

[34] M. Everingham, S. M. A. Eslami, L. van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[35] M. F. Møller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

[36] J. J. Moré, "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Numerical Analysis*, 1978, pp. 105–116.

[37] V. Prisacariu and I. Reid, "PWP3D: Real-Time Segmentation and Tracking of 3D Objects," *International Journal of Computer Vision*, vol. 98, no. 3, pp. 333–354, 2012.

[38] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[39] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014, pp. 75–82.

[40] Y.-Q. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*, 2014, pp. 675–678.

[41] M. Braun, Q. Rao, Y.-K. Wang, and F. Flohr, "Pose-RCNN: Joint Object Detection and Pose Estimation Using 3D Object Proposals," in *Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1546–1551.

[42] A. Dame, V. Prisacariu, C. Y. Ren, and I. Reid, "Dense Reconstruction Using 3D Object Shape Priors," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1288–1295.

[43] S. Geman and D. McClure, "Bayesian Image Analysis: An Application to Single Photon Emission Tomography," in *Proceedings of Statistical*

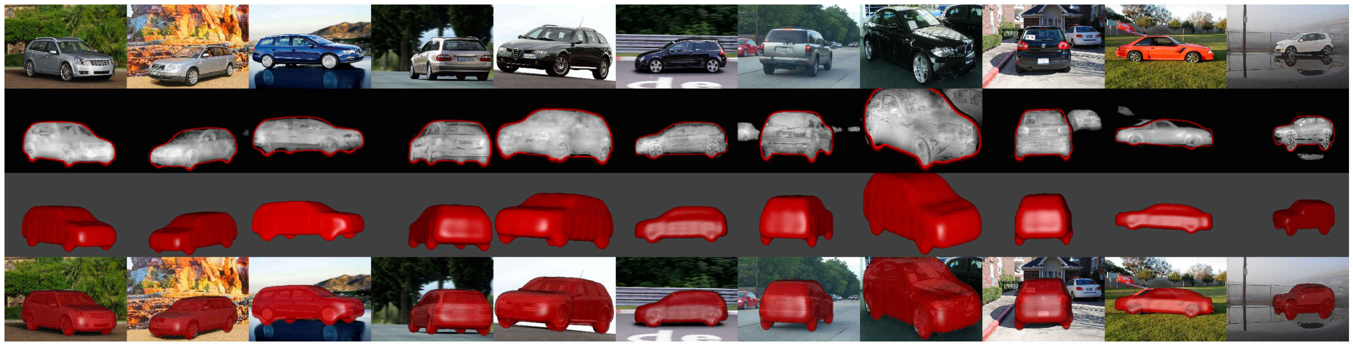
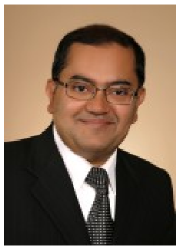


Fig. 22: More results of monocular 3D Shaping. The first nine columns show successful reconstructions from different viewpoints. The next-to-last column shows a failed reconstruction with a 180°-flipped viewpoint. The last column exhibits another failure due to strong reflection from the wheel.

Computing Section of the American Statistical Association, 1985, pp. 12–18.

[44] A. Geiger, P. Lenz, and R. Urtasun, “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

Qing Rao received his Bachelor’s degree in 2010 from Shanghai Jiao Tong University and his Master’s degree in 2012 from TU Munich, with a major focus on computer vision and robotics. In 2019, he received his Ph.D. degree from TU Munich with a dissertation entitled “*Merging the Virtual and Real in a Car: In-Vehicle Augmented Reality*”. Since mid-2017, Qing works at BMW AG in Munich, Germany, as a machine learning expert in the area of autonomous driving. His current research interests include deep learning and environment perception.



Samarjit Chakraborty is a William R. Kenan, Jr. Distinguished Professor in the Department of Computer Science at UNC Chapel Hill, USA. Prior to taking up this position at UNC, from 2008 to 2019, he was a Professor of Electrical Engineering at TU Munich in Germany, where he held the Chair for Real-Time Computer Systems. From 2011 to 2016 he also led a research program on embedded systems for electric vehicles at the TUM CREATE Center for Electromobility in Singapore, where he also served as a Scientific Advisor. He was an Assistant Professor of Computer Science at the National University of Singapore from 2003 to 2008, and obtained his Ph.D. in Electrical Engineering from ETH Zurich in 2003. His research interests include all aspects of system-level design of real-time, embedded, and cyber-physical systems.