Title

Velocity and Shape from Tightly-Coupled LiDAR and Camera

Authors
Daraei, M. Hossein
Vu, Anh
Manduchi, Roberto

Publication Date
2017-06-01

Peer reviewed

Velocity and Shape from Tightly-Coupled LiDAR and Camera

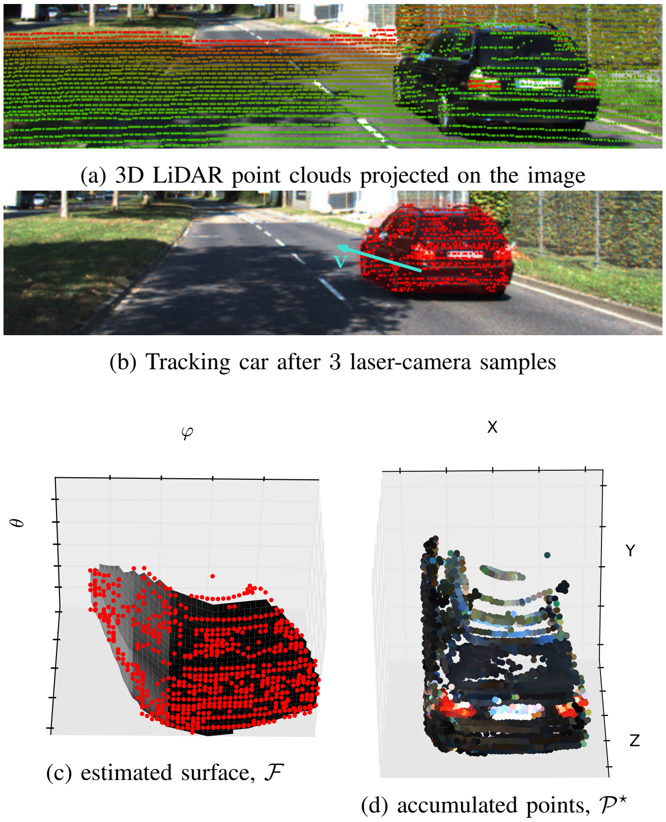
M. Hossein Daraei[1], Anh Vu[2] and Roberto Manduchi[3]

Abstract— **In this paper, we propose a multi-object tracking and reconstruction approach through measurement-level fusion of LiDAR and camera. The proposed method, regardless of object class, estimates 3D motion and structure for all rigid obstacles. Using an intermediate surface representation, measurements from both sensors are processed within a joint framework. We combine optical flow, surface reconstruction, and point-to-surface terms in a tightly-coupled non-linear energy function, which is minimized using Iterative Reweighted Least Squares (IRLS). We demonstrate the performance of our model on different datasets (KITTI with Velodyne HDL-64E and our collected data with 4-layer ScaLa Ibeo), and show an improvement in velocity error and crispness over state-of-the-art trackers.**

I. INTRODUCTION

Scene understanding is an essential component for an autonomous vehicle to safely navigate through dynamic environments. For this purpose, sensor fusion plays an important role in extracting relevant information from different sensors with different characteristics. In order to model and analyze the surroundings, the vehicle must segment the scene into moving objects, reconstruct their shape, and find their 3D motion. In the computer vision community, jointly solving 3D motion and depth is referred to as the *scene flow* problem [2]. In this paper, we lay an object-class-agnostic tracking and reconstruction algorithm for rigid obstacles with arbitrary shape and appearance. In this sense, our approach differs from tracking-by-detection [1] in which objects of one class, e.g. cars, are detected and tracked.

Cameras produce dense grids of intensity and color, rich in semantic information. With a single camera, however, the problem of computing scene flow is highly underdetermined. If the camera is not moving we cannot say much about depth, and if it is, the ambiguity lies between the distinction of camera motion and object motion [3]. Reconstruction and tracking in 3D requires depth information which would be available in a multi-camera system [4], [5]. Depth estimates in a stereo system have lower accuracy than active range-finders, cannot be computed in texture-less areas, and have limited resolution. Cameras are also sensitive to lighting conditions and susceptible to saturation. LiDARs, on the other hand, generate point clouds of a desired region in the environment. If mounted with the right pose, they can *directly and quickly estimate physical obstacles*. Also most LiDARs do not interfere with ambient light and work equally well under different lighting conditions. Radars are also employed for measuring extended object velocities, but unlike LiDARs they are not suited for reconstructing scene structure and surfaces.


(a) 3D LiDAR point clouds projected on the image

(b) Tracking car after 3 laser-camera samples

(c) estimated surface, $\mathcal{F}$

(d) accumulated points, $\mathcal{P}^\star$

Fig. 1: Our tracking and reconstruction (a) takes sequences of LiDAR point clouds and camera images, and (b) finds 3D velocity $\mathbf{v}$ for each object. It (c) estimates an intermediate surface $\mathcal{F}$ to fuse sensors, and (d) accumulates points as a 3D model.

In order to achieve robustness and accuracy in depth and motion estimation, we argue that sensors need to be fused in a *tightly-coupled* fashion. This is equivalent to fusing them at the raw measurement level, and generating joint features and models at a low level. This is different from loosely-coupled (high-level) fusion, in which object candidates or tracks are generated separately for each sensor and merged in the last step [6], [7]. There are fundamental challenges in fusing raw LiDAR and camera measurements. First, they are different in nature, e.g. image data is dense, while LiDAR point clouds are sparse angle measurements that contain range and intensity. Second, they may have different sampling rates and time-stamps. Third, since these sensors are mounted

[1,3] M.Hossein Daraei and Roberto Manduchi are with Baskin School of Engineering, University of California Santa Cruz, 1156 High St, Santa Cruz, CA 95064 USA {daraei,manduchi}@soe.ucsc.edu

[2] Anh Vu is with Electronics Research Laboratory, Volkswagen Group of America, 500 Clipper Dr, Belmont, CA 94002, USA anh.vu@vw.com

at different positions they are susceptible to parallax. Yet another challenge is to efficiently represent objects in both sensor spaces. While simple object models, such as cuboids [8], [9], cannot properly represent an extended object, non-parametric models, such as dense point clouds [10], [11], are non-differentiable and unsuitable for parametric energy minimization. To the best of our knowledge, this work is the first to address these challenges and to lay a dense, multi-modal and measurement-level LiDAR and camera fusion mechanism for object tracking and reconstruction. Our main contributions are,

- We propose an intermediate object representation $\mathcal{F}$, shown in Fig. 1c, that relates sparse 3D LiDAR data in spherical system to dense Camera data in Cartesian 2D system with different time-stamps.
- We directly use raw visual and range sensor measurements in a differential and feature-less energy minimization framework. This is achieved by directly minimizing a multi-modal cost function. We accumulate points to enrich object models and to better track them.
- Based on estimated $\mathcal{F}$ for all rigid objects in the scene, occlusion and parallax are explicitly modeled.

The rest of this paper is organized as follows: In Section II, we overview the literature on multi-sensor fusion, scene flow, and object tracking. In Section III, the problem is formally stated. Section IV describes the intermediate representation and the proposed energy function. In Section V, we evaluate the results of various experiments and compare our method with available trackers. And Section VI concludes the paper.

II. RELATED WORK

Loosely-coupled tracking by detection: Several methods [6], [12], [13] first detect instances of a certain class of objects in each sensor space, then fuse and track detections in a multi-modal tracker, e.g. multi-object Kalman filter [6], Dempster-Shafer combination [14]. Another approach is to employ LiDAR for object hypothesis generation and camera for object hypothesis verification, e.g. [15], [16], [17]. In these methods, a set of 3D region candidates are found in LiDAR space based on geometric features [9]. Then the image patch corresponding to each 3D region is checked by image-based classifiers, e.g. Gaussian Mixture Models (GMMs) [15], AdaBoost [15], Histogram of Gradients (HOG) [16]. These algorithms are limited by detection performance, and work only for one object class.

Piecewise planar Scene Flow: Scene flow, first introduced by Vedula et al, [2], is a 3D motion vector defined at each pixel, that combines image optical flow with depth change. These features are computed in a sparse [18], [19] or dense [3], [20] manner and primarily using stereoscopic vision setups. Rabe and Franke first presented 6DVision [19], which computes sparse scene flow at distinctive pixels. In their Dense6D [21] system, they track each pixel in 3D using a Kalman filter and based on precomputed dense optical flow and stereo. Yamaguchi et al, [22] use a piece-wise planar model and break the scene into superpixels. They form a Markov Random Field (MRF) with a static scene assumption, then optimize segments, ego-motion, occlusion, and plane parameters. Vogel et al, [4] find plane parameters for each segment too, but also consider a pairwise term between nearby planes. In order to deal with dynamic scenes, they compute a rigid motion for each segment. Menze et al, [23] group segments into objects, and find motion parameters for each object rather than all segments. Quiroga et al, [20], [24] take two consecutive intensity and depth maps from an RGBD camera, and track pixels in both intensity and depth using Lucas-Kanade. Sun et al, [25] decompose depth map into layers, and minimize a Gaussian MRF to estimate dense scene flow. Hadfield et al, [26] use particle filters and resample from the joint depth and appearance posterior over time. Despite their promising results, most of stereo or RGBD-based scene flow computation methods take hundreds of seconds to process an image pair.

Odometry and Mapping using LiDAR and Camera: In [27] Zhang et al, propose V-LOAM which combines visual and LiDAR odometry. They employ visual and geometric feature matching to estimate ego-motion. The high-frequency visual odometry handles rapid motion, and LiDAR odometry warrants low-drift and robustness in undesirable lighting conditions. Although such feature-based methods have been employed and have produced promising results in Simultaneous Localization and Mapping (SLAM), they are not suited for multi-object tracking. Feature extraction techniques may fail to provide or match enough number of visual/geometric features for some objects, especially if the object is small, far, texture-less, or occluded.

Non-parametric object tracking: Several methods align tracked object point clouds with new segmented scans, then append the segment to and enrich the 3D model [28], [29], [10]. Wyffels and Campbell [28] lay a probabilistic method to estimate motion and shape for a single extended object using simulated LiDAR data, but do not experiment with real-world datasets and do not add camera. Held et al, [10] track 2D velocity vectors in real-time and accumulate points to form a dense model for each track. A latent surface variable is implicitly included in their velocity posterior, modeled as a collection of points sampled from the visible surface. They make use of a grid-based sampling method, annealed dynamic histograms, to maximize the velocity posterior. They start by sampling from the state space at a coarse resolution, using an approximation to the posterior distribution over velocities. As the resolution increases, they anneal this distribution so that it approaches the true posterior. Their method is capable of tracking in colored point clouds where RGB values are added to sparse points, but does not use dense image data directly and as a separate modality.

Scene flow using LiDAR and Camera: Ilg et al, [29] employ LiDAR and Camera, mounted on a stationary platform, to estimate pose and reconstruct the 3D model of a moving object. They project 3D laser points onto the image, and constrain the 3D motion based on precomputed optical flow at the projected pixel. They form an energy function, adding all motion constraints, and minimize it using IRLS to find the pose as a twist (rotation and translation). As

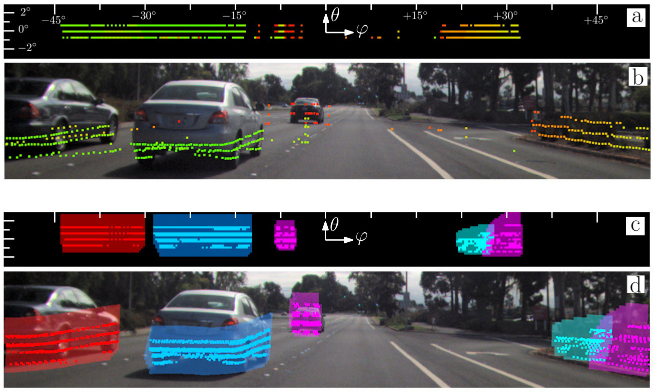


Fig. 2: (a) raw LiDAR measurements on a $\theta\varphi$ grid in spherical coordinates with color encoding distance, (b) projection of points onto image, (c) estimated object domains $\mathcal{W}$ and accumulated points $\mathcal{P}^{\star}$, (d) projection of object domains onto image, i.e. $\mathcal{X}$.

post-processing, they employ point-to-plane Iterative Closest Point (ICP) to refine motion. Our method has a similar nature. But it handles multiple moving objects, does not require a fixed platform, a nodding range finder, or an expensive pre-computation of dense optical flow. It also optimizes all variables in a single-stage joint optimization framework, and does not need ICP-based post-processing.

III. PROBLEM STATEMENT

The two primary sensors, camera and LiDAR, provide intensity (color) and depth measurements of the environment at different times. Let $\mathcal{P} = \{\mathbf{p}_1, \cdots, \mathbf{p}_N\}$ denote a cloud of points $\mathbf{p}_i = (\theta_i\, \varphi_i\, \rho_i)^T$ measured by laser scanner at t_i in spherical coordinates. The measurements are taken at discrete θ and φ intervals and in LiDAR reference system, as in Fig. 2. Also let $\mathcal{I}(\mathbf{x})$ represent an intensity (or color) image captured by camera, and $\mathbf{x} = (x\ y)$. We assume camera intrinsic and extrinsic parameters (with respect to LiDAR) are calibrated. Therefore, as shown in Fig. 2, laser points and image pixels can be geometrically related. Let us also assume that a set of object candidates are provided with the first point cloud, e.g. from dataset annotations or using cloud segmentation methods [30]. Upon arrival of a new image or laser sample at any time, our goal is to update velocity $\mathbf{v} = (v_x\, v_y\, v_z)^T$, 3D accumulated point cloud $\mathcal{P}^{\star}$, and surface model $\mathcal{F}$ for all objects. The estimation must be cross-modality, and performed in a robust Bayesian framework. Similar to [10] we assume $\mathbf{v}$ is pure translational (rotation-free) in short time periods. In order to model sensor uncertainties, we assume laser measurements have Gaussian noise with covariance,

$$\mathbf{C_p} = diag(\sigma_\theta^2, \sigma_\varphi^2, \sigma_\rho^2) \tag{1}$$

which is known from sensor characteristics.We also model images as Gaussian random variables with pixel-wise noise variance σ_I^2 across each channel.

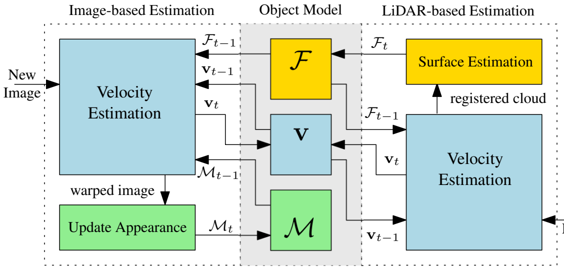


Fig. 3: Block-diagram for the proposed velocity and surface estimation method. Objects are modeled with surface $\mathcal{F}$, appearance $\mathcal{M}$, and velocity $\mathbf{v}$.

IV. PROPOSED METHOD

Fig. 3 represents the overall tracking and reconstruction algorithm for one object. The object model parameters, shown in the middle column, are updated with new LiDAR (right side) or camera (left side) samples. In this section we introduce object model and the estimation approach.

A. Object Representation

While in long-term we represent objects using accumulated set of points $\mathcal{P}^{\star}$, there is a need for an intermediate and differentiable model in short-term.

We model the visible side of objects using an intermediate depth map $\mathcal{F}(\theta, \varphi) : \mathbb{R}^2 \rightarrow \mathbb{R}$ defined on an elevation-azimuth $(\theta\varphi)$ grid. It is defined over a domain $\mathcal{W}$ (Fig. 2c) and is encouraged to be piecewise-planar. It can also be projected onto the image plane using calibration parameters, specifying a set of pixels $\mathcal{X}$ belonging to the object (Fig. 2d). This surface can be directly employed to match new laser measurements, it helps represent optical flow (on image plane) in terms of 3D velocity vector, and also makes possible explicit occlusion and parallax modeling. We also add an appearance model $\mathcal{M}$ for each object, which is simply an intensity (or color) template for the surface. As mentioned earlier, each object also has a unique 3D velocity vector, $\mathbf{v}$.

B. Joint Motion and Depth Estimation

Let $z_{1:t}$ denote the set of all sensor measurements, i.e. all images and laser scans ordered based on synchronized time-stamps. In the remainder of this section we focus on a single object. The posterior distribution of velocity $\mathbf{v}_t$ can be written as,

$$\begin{aligned} &p(\mathbf{v}_t|z_{1:t}) = \int p(\mathbf{v}_{1:t}|z_{1:t}) d\mathbf{v}_{1:t-1} \\ &\propto \int p(z_t|\mathbf{v}_t) p(\mathbf{v}_t|\mathbf{v}_{t-1}) p(\mathbf{v}_{1:t-1}|z_{1:t-1}) d\mathbf{v}_{1:t-1} \\ &= p(z_t|\mathbf{v}_t) p_v(\mathbf{v}_t|z_{1:t-1}) \end{aligned} \tag{2}$$

The first term, referred to as likelihood term, captures the fidelity of z_t given state $\mathbf{v}_t$, and the second term is the prior (prediction) distribution for $\mathbf{v}_t$. Upon receiving a new image $\mathcal{I}_t$, the likelihood can be rewritten as,

$$p_C(\mathcal{I}_t|\mathbf{v}_t) = \int p_I(\mathcal{I}_t|\mathcal{M}_{t-1}, \mathbf{v}_t)\ p_M(\mathcal{M}_{t-1}) d\mathcal{M}_{t-1} \tag{3}$$

where p_I is image likelihood term given object appearance, and p_M is the appearance model. When a new laser scan $\mathcal{P}_t$ is received the likelihood will have a form,

$$p_L(\mathcal{P}_t|\mathbf{v}_t) = \int p_P(\mathcal{P}_t|\mathcal{F}_{t-1}, \mathbf{v}_t)\ p_F(\mathcal{F}_{t-1})d\mathcal{F}_{t-1} \quad (4)$$

where p_P is the laser likelihood given surface model, and p_F is the surface distribution.

We represent the problem in its analogous energy form and address it as an energy minimization problem. Let E_F, E_C, E_L, and E_v represent energy forms for surface distribution, image likelihood, LiDAR likelihood, and velocity prior defined above. At each iteration, we first estimate the surface $\mathcal{F}$ based on accumulated object points $\mathcal{P}^\star$. As described in Sec. IV-C, surface $\mathcal{F}$ is represented with a vector of coefficients $\mathbf{d}$. Surface energy E_F is then formulated as,

$$E_F(\mathbf{d}) = E_D(\mathbf{d}) + \lambda E_R(\mathbf{d}) \quad (5)$$

where E_D is surface data term based on $\mathcal{P}^\star$, E_R is surface regularization term, and λ is a constant. Surface coefficients $\mathbf{d}$, are updated by minimizing this energy function. Surface distribution is then modeled as a Gaussian with mean $\hat{\mathbf{d}}$. We can also estimate surface error variance, σ_F^2, based on surface energy term. When a new image is received, we have in energy form,

$$E(\mathbf{v}) = E_C(\mathbf{v}) + E_v(\mathbf{v}) \quad (6)$$

and update $\mathbf{v}$ by minimizing this energy. Then we update object appearance $\mathcal{M}$ as pixels specified by $\mathcal{X}$ from the new image. If the new sample is a laser point cloud $\mathcal{P}$ we have,

$$E(\mathbf{v}) = E_L(\mathbf{v}) + E_v(\mathbf{v}) \quad (7)$$

which is minimized in a similar fashion to update $\mathbf{v}$. Finally we add new object points from $\mathcal{P}$ to accumulated cloud $\mathcal{P}^\star$. Although the probability distributions defined above are not Gaussian, they can be approximated as locally Gaussian variables. This is analogous to employing an iterative energy minimization approach, e.g. Iterative Re-weighted Least Squares (IRLS) [31], and minimizing a quadratic function at each step. In what follows we explain each term in detail.

C. Surface Data Term, $E_D(\mathbf{d})$

Object surface $\mathcal{F}$, defined on elevation-azimuth grid, is updated after each iteration based on object accumulated points $\mathcal{P}^\star$. Surface probability density function is decomposed as,

$$p_F(\mathcal{F}_{t-1}|\mathcal{P}^\star) \propto p_D(\mathcal{P}^\star|\mathcal{F}_{t-1})\ p_R(\mathcal{F}_{t-1}) \quad (8)$$

with p_D being the data term, and p_R the regularization term enforcing piece-wise planar surfaces. Let us represent $\mathcal{F}$ as a linear combination of Q basis functions,

$$\mathcal{F}(\theta,\, \varphi) = \sum_{j=1}^{Q} d_j \phi_j(\theta,\, \varphi) \quad (9)$$

Let $\Omega_i = (\theta_i, \varphi_i)$ and r_i denote respectively the angular coordinates and range of adjusted laser point $\mathbf{p}_i$ in spherical coordinates. We assume p_D can be locally approximated as a Gaussian distribution at each optimization iteration. Then the log-likelihood of p_D at each iteration is defined as a quadratic function of $\mathbf{d} = (d_1, \cdots, d_Q)^T$ as,

$$E_D(\mathbf{d}) = \sum_{\mathbf{p_i} \in \mathcal{P}^\star} \frac{w_i}{\tau_i^2} \| \sum_{j=1}^{Q} d_j \phi_j(\Omega_i) - r_i)\|^2 \quad (10)$$

where w_i is Huber weight [31] computed from previous iteration error, and τ_i^2 is the range variance of $\mathbf{p}_i$. Note that $\mathcal{P}^\star$ consists of points accumulated from different times. For points in the last laser scan τ_i^2 is equal to the characteristic range variance σ_ρ^2. But older points, due to accumulation of velocity errors, have larger uncertainties and are given less weights. Huber weights also decrease over IRLS iterations for outliers.

D. Surface Regularization Term, $E_R(\mathbf{d})$

It is common [22], [4] to model urban driving environments using piecewise planar surfaces. Inspired by the regularization term in [32], [33], we minimize the second-order derivatives of $\mathcal{F}$ to encourage smooth surfaces. In addition, we allow the surface to potentially fold along sparse line patterns to prevent over-smoothing. We define a surface prior term as,

$$E_R(\mathbf{d}) = \sum_{\theta,\varphi \in \mathcal{W}} w_{\theta\varphi} \|\mathcal{H}\mathcal{F}(\theta, \varphi)\|_F \quad (11)$$

where $\|.\|_F$ is the Frobenius norm, $\mathcal{H}$ represents Hessian matrix, and $w_{\theta\varphi}$ are Huber weights at previous iteration. Most (θ, φ) cells will be assigned large Huber weights, resulting in penalization of second-order derivatives and smooth surface. But these weights are small for a sparse set of cells, which leads to less regularization and possibly surface bending (as in Fig. 4). We also define object extent $\mathcal{W}$ to be the convex hull of $\{(\theta_i, \varphi_i)\}$ for all points in $\mathcal{P}^\star$. As shown in Klowsky et al [32], the energy in (11) can be represented in terms of $\mathbf{d}$ as a quadratic function,

$$E_R(\mathbf{d}) = \mathbf{d}^T \mathbf{Q} \mathbf{d} \quad (12)$$

where elements of $\mathbf{Q}$ are computed as,

$$[\mathbf{Q}]_{\alpha,\beta} = \sum_{\theta,\varphi \in \Omega} w_{\theta\varphi} < \mathcal{H}\phi_\alpha(\theta, \varphi), \mathcal{H}\phi_\beta(\theta, \varphi) > \quad (13)$$

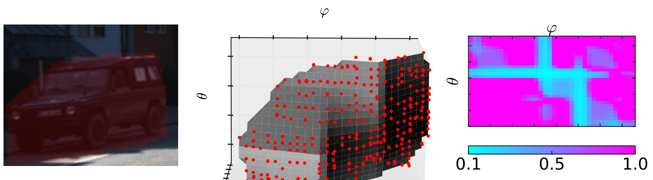
where $< .,. >$ denotes sum of the element of Hadamard multiplication, which are all pre-computed.

E. LiDAR Likelihood Term, $E_L(\mathbf{v})$

The expression in (4) represents the input data likelihood for a new laser cloud $\mathcal{P}_t$. This integral depends on surface distribution and laser data term. The laser data term $p_P(\mathcal{P}_t|\mathcal{F}_{t-1}, \mathbf{v}_t)$ encourages each new point, $\mathbf{p}_i = (\theta_i\ \varphi_i\ r_i)^T$, to fit to surface $\mathcal{F}_{t-1}$ when adjusted with $\mathbf{v}_t$. Let $\mathbf{p}_i' = (\theta_i', \varphi_i', r_i')$ denote time-adjusted points,

$$\mathbf{p}_i' = \mathbf{p}_i - \Delta t_i \mathbf{R} \mathbf{v} \quad (14)$$

where Δt_i is the difference between t_i (time-stamp for $\mathbf{p}_i$) and $\mathcal{F}_{t-1}$ time reference. Also $\mathbf{R} = (\mathbf{r}_\theta\ \mathbf{r}_\varphi\ \mathbf{r}_\rho)^T$ is the



(a) $\mathcal{F}$ on image

(b) $\mathcal{F}$ + laser points

(c) $w_{\theta\varphi}$

Fig. 4: Reconstructed surface $\mathcal{F}$ for a vehicle (a) projected on the image, (b) represented as a 2D depth map on discrete $\theta\varphi$ grid, and (c) corresponding weights for each grid cell. The surface is allowed to bend along patterns with low weights.

linearized transformation at p_i that maps velocity $\mathbf{v}$ from Cartesian to spherical coordinates. Note such transformation is not in general a linear operator, and $\mathbf{R}_i$ is approximated for each $\mathbf{p}_i$. Now let $\mathbf{R}_\Omega$ be $(\mathbf{r}_\theta \ \mathbf{r}_\varphi)^T$. The energy form of laser data term is,

$$\begin{aligned} E_P(\mathbf{v}) &= \sum_{\mathbf{p_i}\in\mathcal{P}} \frac{w_i}{\tau_i^2} \|\mathcal{F}(\Omega_i') - r_i'\|^2 \\ &= \sum_{\mathbf{p_i}\in\mathcal{P}} \frac{w_i}{\tau_i^2} \|\mathcal{F}(\Omega_i - \Delta t_i \mathbf{R}_\Omega \mathbf{v}) - r_i'\|^2 \\ &= \sum_{\mathbf{p_i}\in\mathcal{P}} \frac{w_i}{\tau_i^2} \left\| \sum_j d_j \phi_j(\Omega_i - \Delta t_i \mathbf{R}_\Omega \mathbf{v}) - r_i' \right\|^2 \end{aligned} \tag{15}$$

Similar to surface term, we add Huber weights w_i to eliminate outlier points. Assuming ϕ_i's are twice differentiable, and $\Delta t_i \mathbf{R}_\Omega \mathbf{v}$ is small relative to voxel size,

$$\phi_j(\theta_i', \varphi_i') \approx \phi_j(\theta_i, \varphi_i) - \Delta t_i \nabla \phi_{j,i}^T \mathbf{R}_\Omega \mathbf{v} \tag{16}$$

where $\nabla\phi_{j,i}$ is the gradient of ϕ_j evaluated at Ω_i. By substituting (16) in (15) and setting $r_i' = r_i - \Delta t_i \mathbf{r}_\rho^\intercal \mathbf{v}$ we get a quadratic laser energy term corresponding to a Gaussian distribution. Now let $\hat{\mathbf{d}}$ be the estimated basis function coefficients from surface reconstruction and σ_F^2 denote the surface error variance. The integral in (4) then results in LiDAR negative log-likelihood,

$$E_L(\mathbf{v}) = \sum_{\mathbf{p_i}\in\mathcal{P}} \frac{w_i}{\mu_i^2} \left\| \Phi_i^T \hat{\mathbf{d}} - r_i + \Delta t_i (\mathbf{r}_\rho^T - \hat{\mathbf{d}}^T \nabla \Phi_i \mathbf{R_\Omega}) \mathbf{v} \right\|^2 \tag{17}$$

where Φ_i is a vector whose elements are basis functions evaluated at Ω_i, and $\nabla\Phi_i$ is the matrix of stacked gradients. Also it can be shown $\mu_i^2 = \tau_i^2 + \sigma_F^2$. Also note $\mathcal{P}$ is the most recent laser scan, therefore τ_i^2 is equal to sensor characteristic range variance σ_ρ^2.

F. Image Likelihood Term, $E_C(\mathbf{v})$

The image-based term minimizes the photometric error between appearance model $\mathcal{M}$ and new images as a function of 3D velocity $\mathbf{v}$. We model the appearance distribution in (3) as a Gaussian with current appearance model $\hat{\mathcal{M}}$ as mean and a pixelwise error variance of σ_M^2. After processing each new image, we update $\mathcal{M}$ as the projection of $\mathcal{F}$ on the new image. Following the work of Kerl et al, [34], we assume the photometric error between $\mathcal{M}$ and a new image $\mathcal{I}$ follows t-distribution for each pixel. This assumption leads to non-quadratic log-likelihood which can be handled using iterative reweighting. The energy form corresponding to $p_I(\mathcal{I}|\mathcal{M}_{t-1}, \mathbf{v}_t)$ is,

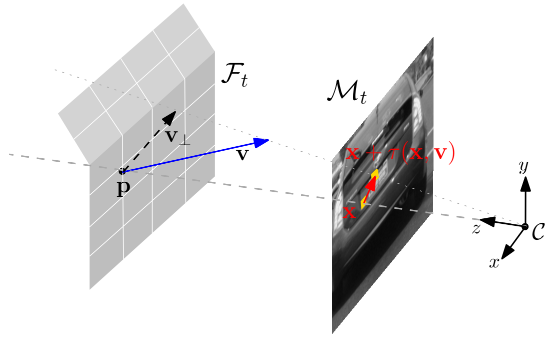


Fig. 5: Relationship between 3D velocity vector $\mathbf{v}$ at an arbitrary point $\mathbf{p}$ on reconstructed surface $\mathcal{F}$, and the induced displacement $\tau(\mathbf{x}, \mathbf{v})$ (optical flow) on the image plane.

$$E_{\mathcal{I}}(\mathbf{v}) = \sum_{\mathbf{x}\in\mathcal{X}} \frac{1}{\sigma_\mathbf{x}^2} \|\mathcal{M}(\mathbf{x} + \tau(\mathbf{x}, \mathbf{v})) - \mathcal{I}(\mathbf{x})\|^2 \tag{18}$$

where $\mathcal{X}$ is the image region corresponding to $\mathcal{W}$, and $\sigma_\mathbf{x}^2$ denotes t-distribution weights, computed based on [34]. The effect of t-distribution weighting is conceptually similar to Huber weights, where outliers get down-weighted to have less influence in the overall estimated motion. The function $\tau(\mathbf{x}, \mathbf{v})$ denotes the pixel displacement induced by three-dimensional velocity $\mathbf{v}$,

$$\tau(\mathbf{x}, \mathbf{v}) = \Delta t \mathbf{B_x} \mathbf{v} \tag{19}$$

where Δt is the time difference between appearance $\mathcal{M}$ and the new image. The 2×3 matrix $\mathbf{B_x}$ projects 3D velocity $\mathbf{v} = (v_x \ v_y \ v_z)^T$ (in the camera reference system) onto image-space as a pixel-wise optical flow vector (see Fig. 5). It can be shown [20] that this matrix can be approximated as,

$$\mathbf{B_x} = \frac{1}{p_z} \begin{pmatrix} f_x & 0 & -f_x x \\ 0 & f_y & -f_y y \end{pmatrix} \tag{20}$$

for pixel (x, y) with depth p_z. Pixel depths can be derived from reconstructed surface $\mathcal{F}$. This is where the intermediate model connects two sensor spaces. We convert 3D points on $\mathcal{F}$ to camera reference system, and then project them onto the image plane in order to determine pixel depths.

By linearizing $\mathcal{M}$ in (18) around $\mathbf{x}$ we get,

$$E_I(\mathbf{v}) = \sum_{\mathbf{x}\in\mathcal{X}} \frac{1}{\sigma_\mathbf{x}^2} \|\Delta t \nabla \mathcal{M}_\mathbf{x}^T \mathbf{B_x} \mathbf{v} - \mathcal{M}_{t,\mathbf{x}}\|^2 \tag{21}$$

where $\mathcal{M}_{t,\mathbf{x}} = \mathcal{M}(\mathbf{x}) - \mathcal{I}(\mathbf{x})$ is the photometric error for $\mathbf{x}$, and $\nabla\mathcal{M}_\mathbf{x}$ is appearance gradient at $\mathbf{x}$. Since $\mathcal{M}$ has a Gaussian distribution, the image likelihood integral in (3) results in another Gaussian with log-likelihood,

$$E_C(\mathbf{v}) = \sum_{\mathbf{x}\in\mathcal{X}} \frac{1}{\sigma_\mathbf{x}^2 + \sigma_M^2} \|\Delta t \nabla \hat{\mathcal{M}}_\mathbf{x}^T \mathbf{B_x} \mathbf{v} - \hat{\mathcal{M}}_{t,\mathbf{x}}\|^2 \tag{22}$$

G. Velocity Prior Term, $E_v(\mathbf{v})$

This term couples velocity estimates through time, and enforces a smooth motion trajectory. Let us assume $\mathbf{v}_{t-1} \sim \mathcal{N}(\hat{\mathbf{v}}_{t-1}; \mathbf{C}_{t-1})$ as the result of last estimation step. We model the transition distribution as $p(\mathbf{v}_t|\mathbf{v}_{t-1}) = \mathcal{N}(\mathbf{v}_{t-1}; \mathbf{C}_{\Delta t})$ with $\mathbf{C}_{\Delta t} = \Delta t^2 \mathbf{C}_n$. Δt is time difference between processing time of $\mathbf{v}_t$ and $\mathbf{v}_{t-1}$, and $\mathbf{C}_n$ is a hyperparameter. It models the change in velocity over one second as a covariance matrix. The prediction distribution from (2) can be rewritten as,

$$p_v(\mathbf{v}_t|z_{1:t-1}) = \int p(\mathbf{v}_t|\mathbf{v}_{t-1})\ p(\mathbf{v}_{t-1}|z_{1:t-1}) d\mathbf{v}_{t-1} \quad (23)$$

which is another Gaussian distribution. In energy form (negative log-likelihood) it can be represented simply as,

$$E_v(\mathbf{v}_t) = (\mathbf{v}_t - \hat{\mathbf{v}}_{t-1})^T \boldsymbol{\Gamma}_v (\mathbf{v}_t - \hat{\mathbf{v}}_{t-1}) \quad (24)$$

where $\boldsymbol{\Gamma}_v = \mathbf{C}_{\Delta t}^{-1}(\mathbf{I}_3 - (\mathbf{C}_{t-1}^{-1} + \mathbf{C}_{\Delta t}^{-1})^{-1}\mathbf{C}_{\Delta t}^{-1})$

H. Occlusion and parallax modeling

Object domain $\mathcal{W}$ determines the span of angles (θ and φ) that the object surface $\mathcal{F}$ is defined for. As shown in Fig. 2c, we process all object domains at the same time. This enables us to explicitly model occlusion for objects with overlapping domains for their $\mathcal{F}$. In handling both occlusion and parallax, we are merely using scene geometry from estimated $\mathcal{F}$ for all objects. For the overlapping $\theta\varphi$ cells, we mark the domain of the farther object as *occluded*. Another problem is that sensors at different positions lead to *parallax*. This means LiDAR and camera are exposed to different regions of the scene. In order to handle parallax, we use the *projection* of object domains onto image, i.e. $\mathcal{X}$ shown in Fig. 2d. In a similar way, we mark the overlapping image pixels for the farther object as occluded.

I. Multi-resolution analysis

For first-order approximations in (16) and (21) to be valid, variations in $\mathbf{v}$ and $\mathbf{d}$ must be small compared to grid size. This is guaranteed by employing a coarse-to-fine and incremental approach. We start from a low-resolution grid for both $\mathcal{F}$ and $\mathcal{M}$, and continue to finer levels. Motion and surface parameters are updated at each iteration as $\mathbf{v} \leftarrow \mathbf{v} + \Delta\mathbf{v}$ and $\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$.

We construct a Gaussian pyramid for $\mathcal{M}$, and make sure velocity variations are small relative to grid size at each pyramid level. We also employ coarse-to-fine basis functions Φ for $\mathcal{F}$. Let $L^2(\mathbb{R})$ denote the set of all functions $f : \mathbb{R} \to \mathbb{R}$ for which $\|f\|^2$ is integrable over $\mathbb{R}$. A multi-resolution analysis of L^2 is a nested set of vector spaces $V^0 \subset V^1 \subset V^2 \subset \dots$ where $clos_{L^2}(\cup_{k\in\mathbb{Z}} V^k) = L^2(\mathbb{R})$. The basis functions for the spaces V^j are called scaling functions $\{\phi_i^j\}$. There are many possibilities for the choice of multi-resolution basis functions, but we prefer square B-spline wavelets due to their simplicity and continuous first and second order derivatives.

The pyramid implementation makes possible to compromise between accuracy and performance, analogous to histogram annealing of Held et al, [10]. If we have more processing capacity, finer levels of the pyramid are processed to attain higher accuracy.

V. EXPERIMENTS

This section covers the results of experiments with the proposed method and other tracking/velocity estimation methods. We compare our results with classic point set registration methods, e.g. ICP and Kernel Correlation (KC). To compare with multimodal trackers, we use the C++ implementation [1] of Any-time Tracker (Held et al, [10]), and implement the method of Ilg et al, [29]. In order to tune hyperparameters in the model we use 5% of sequences in each dataset for training. We need to find the optimum values for Huber function thresholds as well as step size parameters for all terms . We use Nelder-Mead to find the best set of parameters that minimize average velocity error for all objects in the training set. Also since we are experimenting with different LiDARs, each dataset requires separate parameter tuning.

A. Evaluation metrics

In order to compare the results, we use two primary metrics to evaluate estimated velocity vectors. First we define $\|\Delta\mathbf{v}\|$ in [m/s] as the magnitude of the error vector between estimated motion $\hat{\mathbf{v}}$ and ground truth $\mathbf{v}_g$.

We also build object models by aligning the point clouds using our estimated velocity. If estimates are not accurate, the resulting accumulated point cloud will be noisy. For each object, we compute the crispness score [35] as,

$$\frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T}\frac{1}{|\mathcal{P}_i|}\sum_{\mathbf{p}_k^i \in \mathcal{P}_i} G(\mathbf{p}_k^i - \mathbf{p}_k^j) \quad (25)$$

where T is number of frames that object is visible, $\mathcal{P}_i$ denotes ith scanned cloud for object, and $\mathbf{p}_k^j$ denotes the nearest point in $\mathcal{P}_j$ to $\mathbf{p}_k^i$. Also, G denotes a multi-variate Gaussian function. Crispness score computed above has a minimum of zero (poor reconstruction) and a maximum of one (perfect reconstruction).

B. Velocity estimation w/o Tracking

We have implemented a simple version of our method which estimates instantaneous velocity for each object without tracking. This is achieved through replacing the soft constraint of (24) with a velocity constancy assumption over a short time period. We employ this version to assess our method as a velocity estimation module. For this experiment we use 612 joint image-laser segments collected using a $24Hz$ camera and a $25Hz$ Valeo ScaLa B2 laserscanner. This LiDAR scans at a higher frequency compared to Velodyne HDL-64 but has only 4 vertical layers. Each segment includes five consecutive laser point clouds and five consecutive images (captured in a short time window of 200ms) of the same object. These segments are extracted by first removing ground plane from point clouds and then applying the point cloud clustering method of [30]. Also

[1] https://github.com/davheld/precision-tracking

	$\|\Delta \mathbf{v}\|$[m/s]	crispness
Proposed	**0.57**	**0.18**
ICP	1.08	0.1
KC	0.89	0.13

TABLE I: Magnitude of velocity error ($\|\Delta \mathbf{v}\|$) and crispness score for different methods tested on our collected data.

	car + van		pedestrian		cyclist	
	$\|\Delta \mathbf{v}\|$	crisp.	$\|\Delta \mathbf{v}\|$	crisp.	$\|\Delta \mathbf{v}\|$	crisp.
Proposed	**0.47**	**0.43**	**0.55**	**0.38**	0.56	0.38
ICP+Kalman	1.07	0.17	1.35	0.15	1.24	0.14
Ilg, [29]	0.88	0.28	1.05	0.23	1.04	0.25
Held, [10]	0.59	0.35	0.61	0.38	**0.55**	**0.41**

TABLE II: Results of different methods tested on KITTI Raw [36]

note unlike next experiment, laser and camera samples are not synchronized in this experiment. Fig. 2 demonstrates one example of our collected data, and it shows the narrow LiDAR vertical field of view (3.2°) for this experiment. We limit the dataset to scenarios in which cars and other object are all stationary with respect to scene, but the ego-vehicle is moving. Objects in the scene, then, appear to be moving in the reverse direction of ego motion. Since we are measuring ego motion using Applanix POS LV [2], we can compute the ground-truth relative velocity of a parked vehicle in this local reference frame and quantitatively evaluate the precision of our tracking velocity estimates.

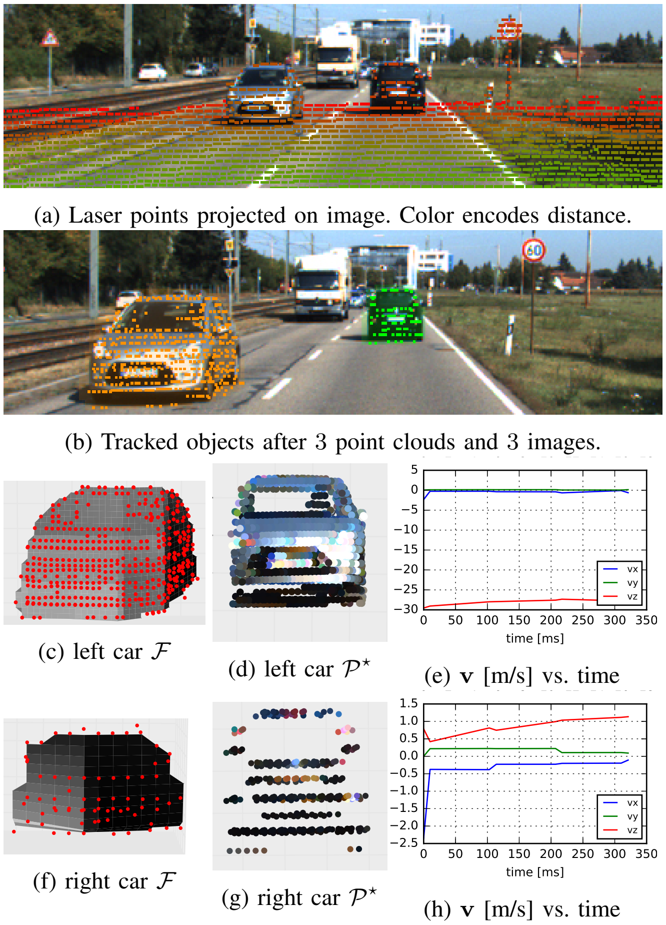
Table I summarizes the quantitative results of this experiment for various methods. ICP and KC are LiDAR only velocity estimation methods, and they work by matching each new laser point to the nearest (ICP) or all points (KC) in the 3D model $\mathcal{P}^\star$. The proposed method, which employs both sensors, outperforms ICP and KC by a large margin.

C. Velocity estimation + Tracking

In this experiment we track object velocities over longer time periods. We employ velocity prior term as in (24) to enforce a smooth motion. We benchmark our results with ICP coupled by Kalman filtering, Any-time Tracker of Held et al, [10] and the method proposed by Ilg et al, [29]. For this experiment, we use KITTI Raw [36] dataset which consists of color images and high-resolution Velodyne HDL-64 point clouds synchronized with a 10Hz sampling rate. Fig. 6 represents a typical driving scenario for this experiment involving two cars. As shown in Fig. 6c, an independent model is estimated for each object. Object segments are extracted from dataset annotations along with object ground truth velocities. We also use object crispness as another metric to compare methods.

Table II shows the results for this experiment over KITTI Raw dataset. All three methods exploit both sensor measurements in order to track objects. The proposed method achieves better results compared to state-of-the-art methods almost all object classes. Any-time Tracker (Held et al,[10]) has slightly better performance for cyclists. The primary strength of the proposed method is in tracking extended objects, e.g. cars. For objects that have negligible extent (far or small-size objects like cyclists), methods which perform point-matching achieve better results.

[2] `http://www.applanix.com/products/poslv.htm`



(a) Laser points projected on image. Color encodes distance.

(b) Tracked objects after 3 point clouds and 3 images.

(c) left car $\mathcal{F}$

(d) left car $\mathcal{P}^\star$

(e) $\mathbf{v}$ [m/s] vs. time

(f) right car $\mathcal{F}$

(g) right car $\mathcal{P}^\star$

(h) $\mathbf{v}$ [m/s] vs. time

Fig. 6: (a) shows raw LiDAR and camera measurements. (b) shows tracked cars after processing 3 point clouds and 3 images. (c-e) represent reconstructed surface, accumulated point cloud, and velocity curve for the left (near) car. (f-h) represents the results for the right (far) car.

Also, [10] achieves velocity error magnitude of 0.56[m/s] and 0.58[m/s] for objects with distance less than and more than 45m. These numbers are, respectively, 0.48[m/s] and 0.57[m/s] for the proposed method. As shown in Fig 6, the reconstructed surface and 3D model of near objects is richer due to more laser samples falling in each discrete $\theta\varphi$ grid cell. The proposed method is currently implemented in Python, and takes 3.55 seconds per frame (KITTI Raw) to run on an Intel Core i7 CPU (single core). A real-time implementation which runs in parallel on multiple processors is to be done as future work.

VI. Conclusion

A multi-object tracking and reconstruction method was prototyped in this paper which combines raw measurements from LiDAR and camera. We introduced a piece-wise planar surface which serves as an intermediate representation between two sensor spaces. The quantitative results prove the robustness of the proposed method when used with different laserscanners. Currently, our method requires object clusters to be given as annotations or by a separate clustering algorithm. One direction for future work could be joint image and LiDAR object segmentation and coupling it with the estimation framework of this paper. This will also help refine object contours over time, i.e. object angular domain $\mathcal{W}$ or its image domain $\mathcal{X}$, and will contribute to the estimation robustness as well.

Acknowledgments

The authors would like to thank Joerg Schlinkheider, Nikhil George, Volkswagen Group of America, Inc., and its Electronics Research Laboratory (ERL) for their support.

References

[1] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3d traffic scene understanding from movable platforms,” *Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[2] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, “Three-dimensional scene flow,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 722–729.

[3] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers, *Efficient dense scene flow from sparse or dense stereo data*. Springer, 2008.

[4] C. Vogel, K. Schindler, and S. Roth, “Piecewise rigid scene flow,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1377–1384.

[5] K. Yamaguchi, D. McAllester, and R. Urtasun, “Robust monocular epipolar flow estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1862–1869.

[6] L. Spinello, R. Triebel, and R. Siegwart, “Multiclass multimodal detection and tracking in urban environments,” *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1498–1515, 2010.

[7] F. Nashashibi, A. Khammari, and C. Laurgeau, “Vehicle recognition and tracking using a generic multisensor and multialgorithm fusion approach,” *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1, pp. 134–154, 2008.

[8] A. Petrovskaya and S. Thrun, “Model based vehicle tracking for autonomous driving in urban environments,” *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland*, vol. 34, 2008.

[9] N. Kaempchen, M. Buehler, and K. Dietmayer, “Feature-level fusion for free-form object tracking using laserscanner and video,” in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 453–458.

[10] D. Held, J. Levinson, S. Thrun, and S. Savarese, “Combining 3d shape, color, and motion for robust anytime tracking,” in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[11] R. Dominguez, E. Onieva, J. Alonso, J. Villagra, and C. Gonzalez, “Lidar based perception solution for autonomous vehicles,” in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*. IEEE, 2011, pp. 790–795.

[12] T.-D. Vu, O. Aycard, and F. Tango, “Object perception for intelligent vehicle applications: A multi-sensor fusion approach,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, June 2014, pp. 774–780.

[13] H. Weigel, P. Lindner, and G. Wanielik, “Vehicle tracking with lane assignment by camera and lidar sensor fusion,” in *Intelligent Vehicles Symposium, 2009 IEEE*, June 2009, pp. 513–520.

[14] D. Nuss, M. Thom, A. Danzer, and K. Dietmayer, “Fusion of laser and monocular camera data in object grid maps for vehicle environment perception,” in *Information Fusion (FUSION), 2014 17th International Conference on*. IEEE, 2014, pp. 1–8.

[15] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, “A lidar and vision-based approach for pedestrian and vehicle detection and tracking,” in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, Sept 2007, pp. 1044–1049.

[16] F. Garcia, B. Musleh, A. de la Escalera, and J. Armingol, “Fusion procedure for pedestrian detection based on laser scanner and computer vision,” in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, Oct 2011, pp. 1325–1330.

[17] J. P. Hwang, S. E. Cho, K. J. Ryu, S. Park, and E. Kim, “Multi-classifier based lidar and camera fusion,” in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, Sept 2007, pp. 467–472.

[18] P. Lenz, J. Ziegler, A. Geiger, and M. Roser, “Sparse scene flow segmentation for moving object detection in urban environments,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 926–932.

[19] U. Franke, C. Rabe, H. Badino, and S. Gehrig, “6d-vision: Fusion of stereo and motion for robust environment perception,” in *Pattern Recognition*. Springer, 2005, pp. 216–223.

[20] J. Quiroga, F. Devernay, and J. Crowley, “Local scene flow by tracking in intensity and depth,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 98–107, 2014.

[21] C. Rabe, T. Müller, A. Wedel, and U. Franke, “Dense, robust, and accurate motion field estimation from stereo image sequences in real-time,” in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 582–595.

[22] K. Yamaguchi, D. McAllester, and R. Urtasun, “Efficient joint segmentation, occlusion labeling, stereo and flow estimation,” in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 756–771.

[23] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[24] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, “Dense semi-rigid scene flow estimation from rgbd images,” in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 567–582.

[25] D. Sun, E. B. Sudderth, and H. Pfister, “Layered rgbd scene flow estimation,” in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 548–556.

[26] S. Hadfield and R. Bowden, “Kinecting the dots: Particle based scene flow from depth sensors,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2290–2295.

[27] J. Zhang and S. Singh, “Visual-lidar odometry and mapping: Low-drift, robust, and fast,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[28] K. Wyffels and M. Campbell, “Joint tracking and non-parametric shape estimation of arbitrary extended objects,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3360–3367.

[29] E. Ilg, R. Kuummerle, W. Burgard, and T. Brox, “Reconstruction of rigid body models from motion distorted laser range data using optical flow,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4627–4632.

[30] A. Golovinskiy and T. Funkhouser, “Min-cut based segmentation of point clouds,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 39–46.

[31] P. J. Huber, *Robust statistics*. Springer, 2011.

[32] R. Klowsky and M. Goesele, “Wavelet-based surface reconstruction from multi-scale sample points,” 2014.

[33] F. Calakli and G. Taubin, “Ssd: Smooth signed distance surface reconstruction,” in *Computer Graphics Forum*, vol. 30, no. 7. Wiley Online Library, 2011, pp. 1993–2002.

[34] C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation for rgb-d cameras,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3748–3754.

[35] M. Sheehan, A. Harrison, and P. Newman, “Self-calibration for a 3d laser,” *The International Journal of Robotics Research*, p. 0278364911429475, 2011.

[36] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.