Three-Filters-to-Normal: An Accurate and Ultrafast Surface Normal Estimator

Rui Fan, *Member, IEEE*, Hengli Wang, *Graduate Student Member, IEEE*, Bohuan Xue, *Graduate Student Member, IEEE*, Huaiyang Huang, *Graduate Student Member, IEEE*, Yuan Wang, Ming Liu, *Senior Member, IEEE*, and Ioannis Pitas, *Fellow, IEEE*

***Abstract*—This letter proposes three-filters-to-normal (3F2N), an accurate and ultrafast surface normal estimator (SNE), which is designed for structured range sensor data, *e.g.*, depth/disparity images. 3F2N SNE computes surface normals by simply performing three filtering operations (two image gradient filters in horizontal and vertical directions, respectively, and a mean/median filter) on an inverse depth image or a disparity image. Despite the simplicity of 3F2N SNE, no similar method already exists in the literature. To evaluate the performance of our proposed SNE, we created three large-scale synthetic datasets (easy, medium and hard) using 24 3D mesh models, each of which is used to generate 1800–2500 pairs of depth images (resolution: 480 $\times$ 640 pixels) and the corresponding ground-truth surface normal maps from different views. 3F2N SNE demonstrates the state-of-the-art performance, outperforming all other existing geometry-based SNEs, where the average angular errors with respect to the easy, medium and hard datasets are 1.66°, 5.69° and 15.31°, respectively. Furthermore, our C++ and CUDA implementations achieve a processing speed of over 260 Hz and 21 kHz, respectively. Our datasets and source code are publicly available at sites.google.com/view/3f2n.**

***Index Terms*—Datasets, range sensor data, surface normal.**

I. Introduction

REAL-TIME 3-dimensional (3D) object recognition is a very challenging computer vision task [1]. Surface normal is an informative and important visual feature used in 3D object recognition [2]. However, not much research has been conducted thoroughly on surface normal estimation, as it is merely considered as an auxiliary functionality for other computer vision applications. Such applications are generally required to perform in an online fashion, and therefore, surface normal estimation must be carried out extremely fast [2].

Manuscript received October 12, 2020; accepted February 21, 2021. Date of publication March 18, 2021; date of current version May 11, 2021. This letter was recommended for publication by Associate Editor T. Pham and Editor C. Cadena Lerma upon evaluation of the reviewers' comments. *(Rui Fan, Hengli Wang, and Bohuan Xue contributed equally to this work.) (Corresponding author: Rui Fan.)*

Rui Fan is with the Department of Computer Science and Engineering as well as the Department of Ophthalmology, the University of California San Diego, La Jolla, CA 92093 USA (e-mail: rui.fan@ieee.org).

Hengli Wang, Bohuan Xue, Huaiyang Huang, and Ming Liu are with the Robotics Institute, the Hong Kong University of Science and Technology, Hong Kong SAR 999077, China (e-mail: hwangdf@connect.ust.hk; bxueaa@connect.ust.hk; hhuangat@connect.ust.hk; eelium@ust.hk).

Yuan Wang is with the Industrial R&D Center, SmartMore, Shenzhen 518040, China (e-mail: ywangeq@connect.ust.hk).

Ioannis Pitas is with the School of Informatics, Aristotle University of Thessaloniki, Thessaloniki 541 24, Greece (e-mail: pitas@csd.auth.gr).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3067308, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3067308

The surface normals can be estimated from either a 3D point cloud or a depth/disparity image (see Fig. 1). The former, such as a LiDAR point cloud, is generally unstructured. Estimating surface normals from unstructured range data usually requires the generation of an undirected graph [2], *e.g.*, a k-nearest neighbor graph or a Delaunay tessellation graph. However, the generation of such graphs is very computationally intensive. Therefore, in recent years, many researchers have been focusing on surface normal estimation from structured range sensor data, *e.g.*, depth/disparity images.

Existing surface normal estimators (SNEs) can be categorized as either geometry-based [1]–[6] or machine/deep learning-based [7], [8]. The former typically computes surface normals by fitting planar or curved surfaces to locally selected 3D point sets, using statistical analysis or optimization techniques, *e.g.*, singular value decomposition (SVD) or principal component analysis (PCA) [2]. On the other hand, the latter generally utilizes data-driven classification/regression models, *e.g.*, convolutional neural networks (CNNs) to infer surface normal information from RGB or depth images [9].

In recent years, with rapid advances in machine/deep learning, many researchers have resorted to deep convolutional neural networks (DCNNs) for surface normal estimation. For instance, Xu *et al.* [10] utilized a so-called prediction-and-distillation network (PAD-Net) to 1) realize monocular depth prediction and surface normal inference, as well as 2) perform scene parsing and contour detection simultaneously. Recently, Huang *et al.* [11] formulated the problem of densely estimating local 3D canonical frames from a single RGB image as a joint estimation of surface normals, canonical tangent directions and projected tangent directions. Such problem was then addressed by a DCNN.

The existing data-driven SNEs are generally trained using supervised learning techniques. Hence, they require a large amount of hand-labeled training data to find the best CNN parameters [8]. Additionally, such CNNs were not specifically designed for surface normal estimation, because SNEs were only used as an auxiliary functionality for other computer vision applications, such as scene parsing, 3D object detection, and depth perception. Furthermore, many robotics and computer vision applications, *e.g.*, autonomous driving [12], require very fast surface normal estimation (in milliseconds). Unfortunately, the existing machine/deep learning-based SNEs are not fast enough. Moreover, the accuracy achieved by data-driven SNEs is still far from satisfactory (the average proportion of good

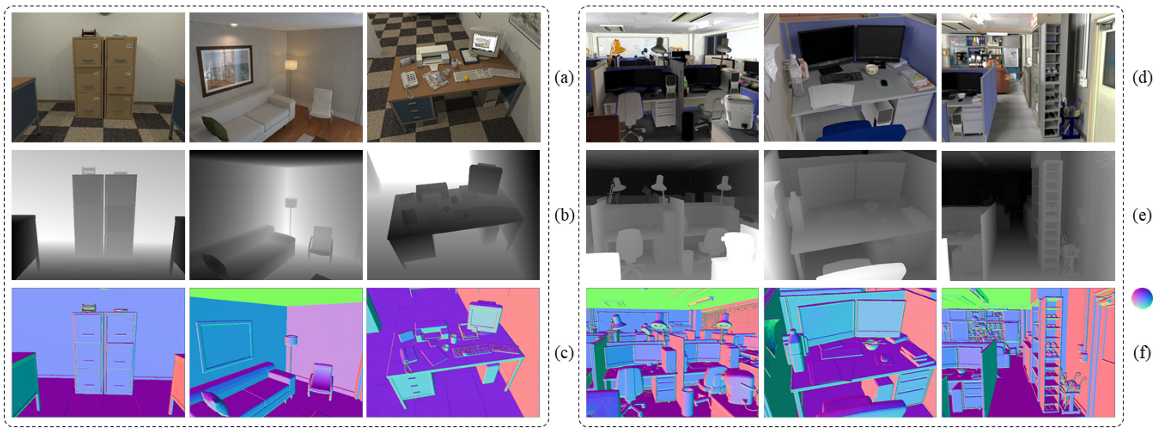

Fig. 1. Surface normal estimation from depth and disparity images: (a) and (b) show examples of RGB and depth images of the Augmented ICL-NUIM dataset [3], respectively; (d) and (e) show examples of RGB and disparity images of the Tsukuba stereo dataset, respectively; (c) and (f) show the surface normals estimated from (b) and (e), respectively, using 3F2N SNE.

pixels is usually lower than 80%) [7], [8]. Most importantly, it can be considered more reasonable to estimate surface normals from point clouds or disparity/depth images rather than from RGB images. Hence, there is a strong motivation to develop a lightweight SNE for structured range data with high accuracy and speed.

The major contributions of this work are as follows:

1) **Three-filter-to-normal (3F2N)**, an accurate and ultrafast SNE. We published its Matlab, C++ and CUDA implementations at https://github.com/ruirangerfan/three-filters-to-normal. Compared with other geometry-based SNEs, 3F2N SNE greatly improves the trade-off between speed and accuracy.
2) Three datasets (easy, medium and hard) created using 24 3D mesh models. Each mesh model is used to generate 1800–2500 depth images from different views. The corresponding surface normal ground truth is also provided, as 3D mesh object models (rather than the objects themselves) are available for surface normal ground truth generation.

II. Related Work

This section provides an overview of geometry-based SNEs.

1) PlaneSVD SNE [13]: The simplest way to estimate the surface normal of an observed 3D point $\mathbf{p}_i = [x, y, z]^\top$ in the camera coordinate system (CCS) is to fit a local plane:

$$n_x x + n_y y + n_z z + b = 0 \tag{1}$$

to the points in $\mathbf{Q}_i^+ = [\mathbf{Q}_i^\top, \mathbf{p}_i]^\top$, where $\mathbf{Q}_i = [\mathbf{q}_{i1}, \dots, \mathbf{q}_{ik}]^\top$ ($\mathbf{q}_{ij} \neq \mathbf{p}_i$) is a set of k neighboring points of $\mathbf{p}_i$. The surface normal $\mathbf{n}_i = [n_x, n_y, n_z]^\top$ can be estimated by solving:

$$\min_{\mathbf{b}_i} \left|\left| \left[\mathbf{Q}_i^+ \;\; \mathbf{1}_{k+1}\right] \mathbf{b}_i \right|\right|_2, \tag{2}$$

where $\mathbf{b}_i = [\mathbf{n}_i^\top, b]^\top$ and $\mathbf{1}_m$ is an m-entry vector of ones. (1) can be solved by factorizing $\mathbf{Q}_i^+$ into $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ using SVD. $\hat{\mathbf{b}}_i$ (the optimum $\mathbf{b}_i$) is a column vector in $\mathbf{V}$ corresponding to the smallest singular value in $\boldsymbol{\Sigma}$ [2].

2) PlanePCA SNE [14]: $\mathbf{n}_i$ can also be estimated by removing the empirical mean $\bar{\mathbf{q}}_i = \frac{1}{k+1}(\mathbf{p}_i + \Sigma_{j=1}^k \mathbf{q}_{ij})$ from $\mathbf{Q}_i^+$ and rearranging (2) as follows [15]:

$$\min_{\mathbf{n}_i} \left|\left| \left[\mathbf{Q}_i^+ - \bar{\mathbf{Q}}_i^+\right] \mathbf{n}_i \right|\right|_2, \tag{3}$$

where $\bar{\mathbf{Q}}_i^+ = \mathbf{1}_{k+1}\bar{\mathbf{q}}_i^\top$. Minimizing (3) is equivalent to performing PCA on $\mathbf{Q}_i^+$ and selecting the principal component with the smallest covariance [2].

3) VectorSVD SNE [2]: A straightforward alternative to fitting (1) to $\mathbf{Q}_i^+$ is to minimize the sum of the inner dot products between $\mathbf{r}_{ij} = \mathbf{q}_{ij} - \mathbf{p}_i$ and $\mathbf{n}_i$, namely,

$$\min_{\mathbf{n}_i} \left|\left| \left[\mathbf{Q}_i - \mathbf{1}_k \mathbf{p}_i^\top\right] \mathbf{n}_i \right|\right|_2. \tag{4}$$

This minimization is done by SVD.

4) AreaWeighted SNE [2]: A triangle can be formed by a given pair of $\mathbf{r}_{ij}$ and $\mathbf{r}_{ij+1}$, as defined above. A general expression of averaging-based SNEs is as follows [2]:

$$\mathbf{n}_i = \frac{1}{k}\sum_{j=1}^{k} w_j \frac{\mathbf{r}_{ij} \times \mathbf{r}_{ij+1}}{\|\mathbf{r}_{ij} \times \mathbf{r}_{ij+1}\|_2}, \tag{5}$$

where w_j is a weight and $\mathbf{r}_{ik+1} = \mathbf{r}_{i1}$. In AreaWeighted SNE, the surface normal of each triangle is weighted by the magnitude of its area:

$$w_j = \frac{1}{2}\|\mathbf{r}_{ij} \times \mathbf{r}_{ij+1}\|_2. \tag{6}$$

5) AngleWeighted SNE [2]: The weight w_j of each triangle relates to the angle between $\mathbf{r}_{ij}$ and $\mathbf{r}_{ij+1}$:

$$w_j = \cos^{-1}\left(\frac{\langle \mathbf{r}_{ij}, \mathbf{r}_{ij+1}\rangle}{\|\mathbf{r}_{ij}\|_2 \|\mathbf{r}_{ij+1}\|_2}\right), \tag{7}$$

where $\langle\cdot\rangle$ is a dot product operator.

6) FALS SNE [5]: The relationship between the Cartesian coordinate system and the spherical coordinate system (SCS) is as follows [5]:

$$\mathbf{p}_i = r_i \mathbf{v}_i = r_i \begin{bmatrix} \sin\theta_i \cos\phi_i \\ \sin\phi_i \\ \cos\theta_i \cos\phi_i \end{bmatrix}, \tag{8}$$

where $r_i \geq 0$, $\theta_i \in (-\pi, \pi]$ and $\phi_i \in (-\frac{\pi}{2}, \frac{\pi}{2}]$. Since all points in $\mathbf{Q}_i^+$ are in a small neighborhood [5], their r_i are considered to be identical in FALS SNE. (2) and (8) result in:

$$\min_{\tilde{\mathbf{n}}_i} \left|\left|\mathbf{V}_i^{+}\tilde{\mathbf{n}}_i - \mathbf{s}_i\right|\right|_2, \tag{9}$$

where $\mathbf{V}_i^{+} = [\mathbf{v}_i, \mathbf{v}_{i1}, \dots, \mathbf{v}_{ik}]^\top$, $\tilde{\mathbf{n}}_i = \mathbf{n}_i/b^2$ and $\mathbf{s}_i = [r_i^{-1}, r_{i1}^{-1}, \dots, r_{ik}^{-1}]^\top$.

7) SRI SNE [5]: Similar to FALS SNE, SRI SNE first transforms the range data from the Cartesian coordinate system to the SCS. $\mathbf{n}_i$ is then obtained by computing the partial derivative of the local tangential surface s:

$$\mathbf{n}_i = \nabla s(\theta_i, \phi_i) = [\mathbf{e}_z,\ \mathbf{e}_x,\ \mathbf{e}_y]\,\mathbf{R}_i \begin{bmatrix} 1 \\ \frac{1}{r_i\cos\phi_i}\partial r_i/\partial\theta_i \\ \frac{1}{r_i}\partial r_i/\partial\phi_i \end{bmatrix}, \tag{10}$$

where $\mathbf{R}_i$ is an SO(3) matrix with respect to θ_i and ϕ_i. $\mathbf{e}_z$, $\mathbf{e}_x$ and $\mathbf{e}_y$ are the unit vectors in the z, x and y coordinate axes, respectively. $\nabla s(\theta_i, \phi_i)$ can be obtained by applying standard image convolutional kernels.

8) LINE-MOD SNE [1]: Firstly, the optimal gradient $\nabla z = [\partial z/\partial u, \partial z/\partial v]^\top$ of a depth map is computed. Then, a 3D plane is formed by three points $\mathbf{p}_0$, $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{t}(\tilde{\mathbf{p}}_i)z, \\ \mathbf{p}_1 &= \mathbf{t}\left(\tilde{\mathbf{p}}_i + [1,0]^\top\right)\left(z + \frac{\partial z}{\partial u}\right), \\ \mathbf{p}_2 &= \mathbf{t}\left(\tilde{\mathbf{p}}_i + [0,1]^\top\right)\left(z + \frac{\partial z}{\partial v}\right), \end{aligned} \tag{11}$$

where $\mathbf{t}(\tilde{\mathbf{p}}_i)$ is the vector along the line of sight that goes through an image pixel $\tilde{\mathbf{p}}_i = [u_i, v_i]^\top$ and is computed using camera intrinsic parameters. The surface normal $\mathbf{n}_i$ can be computed using:

$$\mathbf{n}_i = \frac{(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_2)}{\|(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_2)\|_2}. \tag{12}$$

III. Three-Filters-to-Normal

In this letter, we introduce 3F2N SNE, which is simple to understand and use. Our SNE can compute surface normals from structured range sensor data using only three filters: 1) a horizontal image gradient filter, 2) a vertical image gradient filter and 3) a mean/median filter.

A 3D point $\mathbf{p}_i = [x, y, z]^\top$ in the CCS can be transformed to $\tilde{\mathbf{p}}_i = [u, v]^\top$ using [16]:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{p}_i = \begin{bmatrix} f_x & 0 & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{13}$$

where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{p}_o = [u_o, v_o]^\top$ is the image principal point, and f_x and f_y are the camera focal lengths (in pixels) in the x and y directions, respectively. Combining (1) and (13) results in:

$$\frac{1}{z} = -\frac{1}{b}\left(n_x\frac{u - u_o}{f_x} + n_y\frac{v - v_o}{f_y} + n_z\right). \tag{14}$$

Differentiating (14) with respect to u and v leads to:

$$\frac{\partial 1/z}{\partial u} = -\frac{n_x}{bf_x}, \quad \frac{\partial 1/z}{\partial v} = -\frac{n_y}{bf_y}, \tag{15}$$

which can be approximated by respectively performing horizontal and vertical image gradient filters, *e.g.*, Sobel, Scharr and Prewitt, on the inverse depth image (an image storing the values of $1/z$). Rearranging (15) results in the following expressions of n_x and n_y:

$$n_x = -bf_x\frac{\partial 1/z}{\partial u}, \quad n_y = -bf_y\frac{\partial 1/z}{\partial v}. \tag{16}$$

Given an arbitrary $\mathbf{q}_{ij} \in \mathbf{Q}_i$, we can compute the corresponding n_{zj} by plugging (16) into (1):

$$n_{zj} = b\frac{f_x\Delta x_{ij}\frac{\partial 1/z}{\partial u} + f_y\Delta y_{ij}\frac{\partial 1/z}{\partial v}}{\Delta z_{ij}}, \tag{17}$$

where $\mathbf{r}_{ij} = \mathbf{q}_{ij} - \mathbf{p}_i = [\Delta x_{ij}, \Delta y_{ij}, \Delta z_{ij}]^\top$. Since (16) and (17) have a common factor of $-b$, they can be simplified as:

$$\begin{aligned} & n_x = f_x\frac{\partial 1/z}{\partial u}, \quad n_y = f_y\frac{\partial 1/z}{\partial v}, \\ & \hat{n}_z = -\Phi\left\{\frac{\Delta x_{ij}n_x + \Delta y_{ij}n_y}{\Delta z_{ij}}\right\}, \ j = 1, \dots, k, \end{aligned} \tag{18}$$

where $\Phi\{\cdot\}$ represents the manner of $\hat{n}_z$ (the optimum n_z) estimation. In our previous work [12], $\mathbf{n}_i$ is written in spherical coordinates and $\Phi(\cdot)$ is formulated as an energy minimization problem [12], which is computationally intensive. Hence, in this letter, $\Phi\{\cdot\}$ represents a mean/median filtering operation used to estimate n_z. Please note: if the depth value of $\mathbf{p}_i$ is identical to those of all its neighboring points $\mathbf{q}_{ij} \in \mathbf{Q}_i$, we consider that the direction of its corresponding surface normal is perpendicular to the image plane and simply set $\mathbf{n}_i$ to $[0, 0, -1]^\top$. The performances of estimating $\mathbf{n}_i$ using the mean filter and using the median filter will be compared in Section IV.

Specifically, for a stereo camera, $f_x = f_y = f$, and the relationship between depth z and disparity d is as follows [18]:

$$z = \frac{ft_c}{d}, \tag{19}$$

where t_c is the stereo rig baseline. Therefore,

$$\begin{aligned} \frac{\partial 1/z}{\partial u} &= \frac{\partial 1/z}{\partial d}\frac{\partial d}{\partial u} = \frac{1}{ft_c}\frac{\partial d}{\partial u}, \\ \frac{\partial 1/z}{\partial v} &= \frac{\partial 1/z}{\partial d}\frac{\partial d}{\partial v} = \frac{1}{ft_c}\frac{\partial d}{\partial v}. \end{aligned} \tag{20}$$

Plugging (19) and (20) into (18) results in:

$$\begin{aligned} & n_x = \partial d/\partial u, \quad n_y = \partial d/\partial v, \\ & \hat{n}_z = -\Phi\left\{\frac{\Delta x_{ij}n_x + \Delta y_{ij}n_y}{\Delta z_{ij}}\right\}, \ j = 1, \dots, k. \end{aligned} \tag{21}$$

Therefore, our SNE can also estimate surface normals from a disparity image using three filters.

IV. Experiments

A. Datasets and Evaluation

In our experiments, we used 24 3D mesh models from Free3D[1] to create three datasets (eight models in each dataset). According to different difficulty levels, we name our datasets "easy," "medium" and "hard," respectively. Each 3D mesh model

[1]free3d.com

is first fixed at a certain position. A virtual range sensor with pre-set intrinsic parameters is then used to capture depth images at 1800–2500 different view points. At each view point, a 480×640 pixel depth image is generated by rendering the 3D mesh model using OpenGL Shading Language[2] (GLSL). However, since the OpenGL rendering process applies linear interpolation by default, rendering surface normal images is infeasible. Hence, the surface normal of each triangle, constructed by three mesh vertices, is considered to be the ground truth surface normal of any 3D points residing on this triangle. Our datasets are publicly available at sites.google.com/view/3f2n/datasets for research purposes. In addition to our datasets, we also utilize two real-world datasets: 1) the DIODE dataset[3] [17] and 2) the ScanNet[4] [19] dataset to evaluate the SNE performance on noisy depth data. Furthermore, we utilize two metrics: a) the average angular error (AAE) e_{A} and b) the proportion of good pixels (PGP) e_{P} [6]:

$$e_{\mathrm{A}} = \frac{1}{m}\sum_{k=1}^{m}\psi_k, \qquad e_{\mathrm{P}}(\varphi) = \frac{1}{m}\sum_{k=1}^{m}\delta(\psi_k,\varphi) \tag{22}$$

to quantify the SNE accuracy, where:

$$\delta(\psi_k,\varphi) = \begin{cases} 0 \ (\psi_k > \varphi) \\ 1 \ (\psi_k \le \varphi) \end{cases}, \tag{23}$$

$$\psi_k = \cos^{-1}\left(\frac{\langle \mathbf{n}_k, \hat{\mathbf{n}}_k\rangle}{\|\mathbf{n}_k\|_2\|\hat{\mathbf{n}}_k\|_2}\right), \tag{24}$$

m is the number of 3D points used for evaluation, φ is the angular error tolerance, and $\mathbf{n}_k$ and $\hat{\mathbf{n}}_k$ are the estimated and ground truth surface normals, respectively. In addition to accuracy, we also record the SNE processing time t (ms) and introduce a new metric:

$$\pi = e_{\mathrm{A}} t \text{ (degrees/kHz)} \tag{25}$$

to quantify the trade-off between the speed and accuracy of a given SNE. A fast and precise SNE achieves a low π score.

B. Filter Settings and Implementation Details

As discussed in Section III, n_x and n_y can be estimated by convolving an inverse depth image or a disparity map with image convolutional kernels, *e.g.*, Sobel, Scharr, Prewitt, *etc*. Hence, in our experiments, we first compare the accuracy of the surface normals estimated using the aforementioned convolutional kernels. The brute-force search strategy is then applied to find the best parameters for a 3×3 kernel. Our experiments illustrate that finite difference (FD) kernel, *i.e.*, $[-1, 0, 1]$, can achieve the best overall performance.

We implement the proposed SNE in Matlab C and C++ on a CPU and in CUDA on a GPU. The source code is available at github.com/ruirangerfan/three-filters-to-normal. Similar to the FALS, SRI and LINE-MOD SNE implementations provided in the opencv_contrib repository,[5] we use advanced vector extensions 2 (AVX2) and streaming SIMD (single instruction, multiple data) extensions (SSE) instruction sets to optimize our C++ implementation. Since our approach estimates surface normals from an 8-connected neighborhood, we also use memory alignment strategies to speed up our SNE. In the GPU implementation, we first create a texture object in the GPU texture memory and then bind this object with the address of the input depth/disparity image, which greatly reduces the memory requests from the GPU global memory.

[2]opengl.org/sdk/docs/tutorials/ClockworkCoders/glsl_overview.php
[3]diode-dataset.org
[4]www.scan-net.org/
[5]github.com/opencv/opencv_contrib

TABLE I
THE RUNTIME (MS) OF THE CPU IMPLEMENTATIONS (USING A SINGLE THREAD) WITH RESPECT TO DIFFERENT IMAGE GRADIENT FILTERS AND MEAN/MEDIAN FILTERS

Gradient filter	Mean filter	Median filter
FD	**3.722**	**10.973**
Sobel	3.824	11.167
Scharr	3.848	11.355
Prewitt	3.743	11.065

TABLE II
THE RUNTIME (MS) OF THE GPU IMPLEMENTATIONS WITH RESPECT TO DIFFERENT IMAGE GRADIENT FILTERS AND MEAN/MEDIAN FILTERS

Method	Jetson TX2	GTX 1080 Ti	RTX 2080 Ti
FD-Mean	**0.823521**	**0.049504**	**0.046944**
Sobel-Mean	0.855843	0.052288	0.051232
Scharr-Mean	0.860319	0.052320	0.051280
Prewitt-Mean	0.857762	0.052256	0.050816
FD-Median	**1.206337**	**0.102368**	**0.065536**
Sobel-Median	1.217023	0.104608	0.067840
Scharr-Median	1.239041	0.105376	0.071008
Prewitt-Median	1.240479	0.105152	0.069024

C. Performance Evaluation

We first compare the performances of the proposed SNE with respect to different image gradient filters (FD, Sobel, Scharr and Prewitt) and mean/median filter. e_{A} scores achieved on our and the DIODE [17] datasets are given in Fig. 2. The runtime of our implementations on an Intel Core i7-8700 K CPU (using a single thread) and three state-of-the-art GPUs (Jetson TX2, GTX 1080 Ti and RTX 2080 Ti) is also given in Tables I and II, respectively. We can observe that FD outperforms Sobel, Scharr and Prewitt in terms of e_{A} on all datasets. Also, using the median filter can achieve better surface normal accuracy than using the mean filter, because an n_z candidate in (17) can differ significantly from the ground truth value, introducing significant noise to the mean filter. The e_{A} scores achieved using FD-Median SNE are lower than those achieved by FD-Mean SNE by 0.5°, 1.0°, 0.6°, 0.7° and 0.6° with respect to 3F2N-easy, 3F2N-medium, DIODE-indoor [17], DIODE-outdoor [17], and ScanNet [19] datasets, respectively. However, median filter is much more computationally intensive and time-consuming than the mean filter, because it needs to sort eight n_z candidates and find the median value. From Tables I and II, we can observe that both FD-Mean SNE and FD-Median SNE perform much faster than real-time across different computing platforms. The processing speed of FD-Mean SNE is over 1 kHz and 21 kHz on the Jetson TX2 GPU and RTX 2080 Ti GPU, respectively. Furthermore, FD-Mean SNE performs around 1.4 to 2.1 times faster than the FD-Median

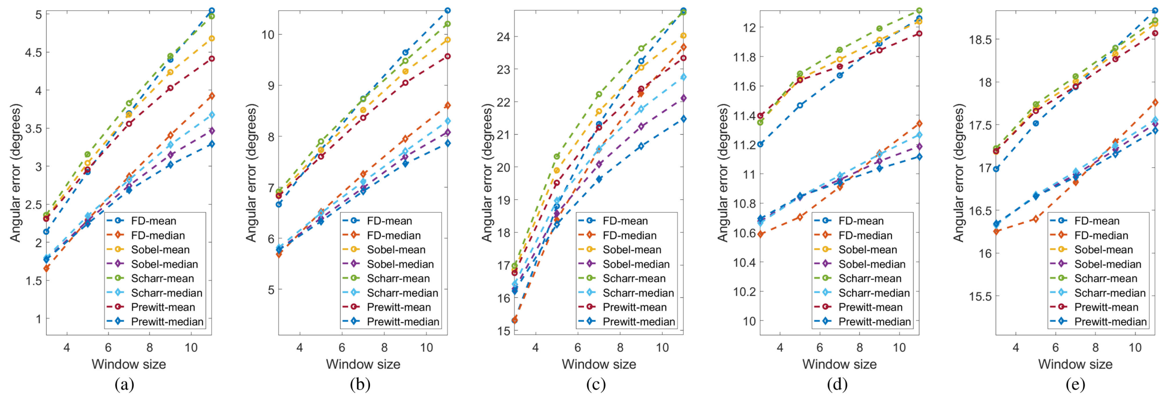

Fig. 2. e_A comparisons with respect to different filter types and sizes: (a) 3F2N-easy dataset; (b) 3F2N-medium dataset; (c) 3F2N-hard dataset; (d) DIODE indoor dataset [17]; (e) DIODE outdoor dataset [17]. Please note: (a)-(e) use different scales.

TABLE III
COMPARISONS AMONG GEOMETRY-BASED SNES ON OUR CREATED SYNTHETIC DATASETS

Method	t (ms) ↓	e_A (degrees) ↓			π (degrees/kHz) ↓		
		Easy	Medium	Hard	Easy	Medium	Hard
PlaneSVD [14]	393.69	2.07	6.07	17.59	813.87	2389.73	6923.18
PlanePCA [13]	631.88	2.07	6.07	17.59	1306.29	3835.59	11111.92
VectorSVD [4]	563.21	2.13	6.27	18.01	1199.63	3529.11	10142.34
AreaWeighted [4]	1092.24	2.20	6.27	17.03	2407.74	6843.56	18600.68
AngleWeighted [4]	1032.88	1.79	**5.67**	**13.26**	1850.00	5855.62	13693.24
FALS [5]	4.11	2.26	6.14	17.34	9.26	25.20	71.17
SRI [5]	12.18	2.64	6.71	19.61	32.18	81.66	238.78
LINE-MOD [3]	6.43	6.53	9.94	31.45	41.93	63.84	202.08
SNE-RoadSeg [12]	7.92	2.04	6.28	16.37	16.16	49.74	129.65
FD-Mean (ours)	**3.72**	2.14	6.66	15.30	**7.96**	**24.80**	**56.96**
FD-Median (ours)	10.97	**1.66**	5.69	15.31	18.18	62.38	168.03

TABLE IV
e_P COMPARISON AMONG GEOMETRY-BASED SNES WITH RESPECT TO DIFFERENT φ ON OUR CREATED SYNTHETIC DATASETS

Method	e_P ↑								
	Easy			Medium			Hard		
	φ=10°	φ=20°	φ=30°	φ=10°	φ=20°	φ=30°	φ=10°	φ=20°	φ=30°
PlaneSVD [14]	0.9648	0.9792	0.9855	0.8621	0.9531	0.9718	0.6202	0.7394	0.7914
PlanePCA [13]	0.9648	0.9792	0.9855	0.8621	0.9531	0.9718	0.6202	0.7394	0.7914
VectorSVD [4]	0.9643	0.9777	0.9846	0.8601	0.9495	0.9683	0.6187	0.7346	0.7848
AreaWeighted [4]	0.9636	0.9753	0.9819	0.8634	0.9504	0.9665	0.6248	0.7448	0.7977
AngleWeighted [4]	**0.9762**	**0.9862**	**0.9893**	**0.8814**	**0.9711**	**0.9809**	0.6625	**0.8075**	**0.8651**
FALS [5]	0.9654	0.9794	0.9857	0.8621	0.9547	0.9731	0.6209	0.7433	0.7961
SRI [5]	0.9499	0.9713	0.9798	0.8431	0.9403	0.9633	0.5594	0.6932	0.7605
LINE-MOD [3]	0.8542	0.9085	0.9343	0.7277	0.8803	0.9282	0.3375	0.4757	0.5636
SNE-RoadSeg [12]	0.9693	0.9810	0.9871	0.8618	0.9512	0.9725	0.6226	0.7589	0.8113
FD-Mean (ours)	0.9563	0.9767	0.9864	0.8349	0.9423	0.9674	0.6191	0.7671	0.8368
FD-Median (ours)	0.9723	0.9829	0.9889	0.8722	0.9600	0.9766	**0.6631**	0.7821	0.8289

SNE. Therefore, the latter achieves the best surface normal accuracy, while the former achieves the best processing speed.

Moreover, we compare 3F2N SNE with all other state-of-the-art geometry-based SNEs, as mentioned in Section II. Some examples of the experimental results are shown in Fig. 3, where it can be seen that the bad estimates mainly reside on the object edges. Additionally, Table III shows comparisons of e_A on the easy, medium and hard datasets, where we can find that FD-Median SNE achieves the best e_A score on the easy dataset, while AngleWeighted [2] SNE achieves the best e_A scores on the medium and hard datasets. Meanwhile, the e_A scores achieved by FD-Median SNE and AngleWeighted [2] SNE are very similar. The runtime (C++ implementations using a single thread) and π scores achieved by the aforementioned SNEs are given in Table III, where we can observe that the averaging-based SNEs are the most time-consuming ones, while FD-Mean SNE achieves the fastest processing speed. Furthermore, FD-Mean, FALS [5] and FD-Median SNEs occupy the first three places, respectively, in terms of π score. Moreover, Table IV compares their PGP scores with respect to different φ on the easy, medium and hard datasets, where we can see that AngleWeighted [2] SNE achieves the best e_P scores, except for $\varphi = 10°$ (hard dataset). However, according to Table III, AngleWeighted [2] SNE is extremely time-consuming and achieves a very bad π score. On the other hand, FD-Median SNE and AngleWeighted [2] SNE achieve similar e_P scores, but the former performs about

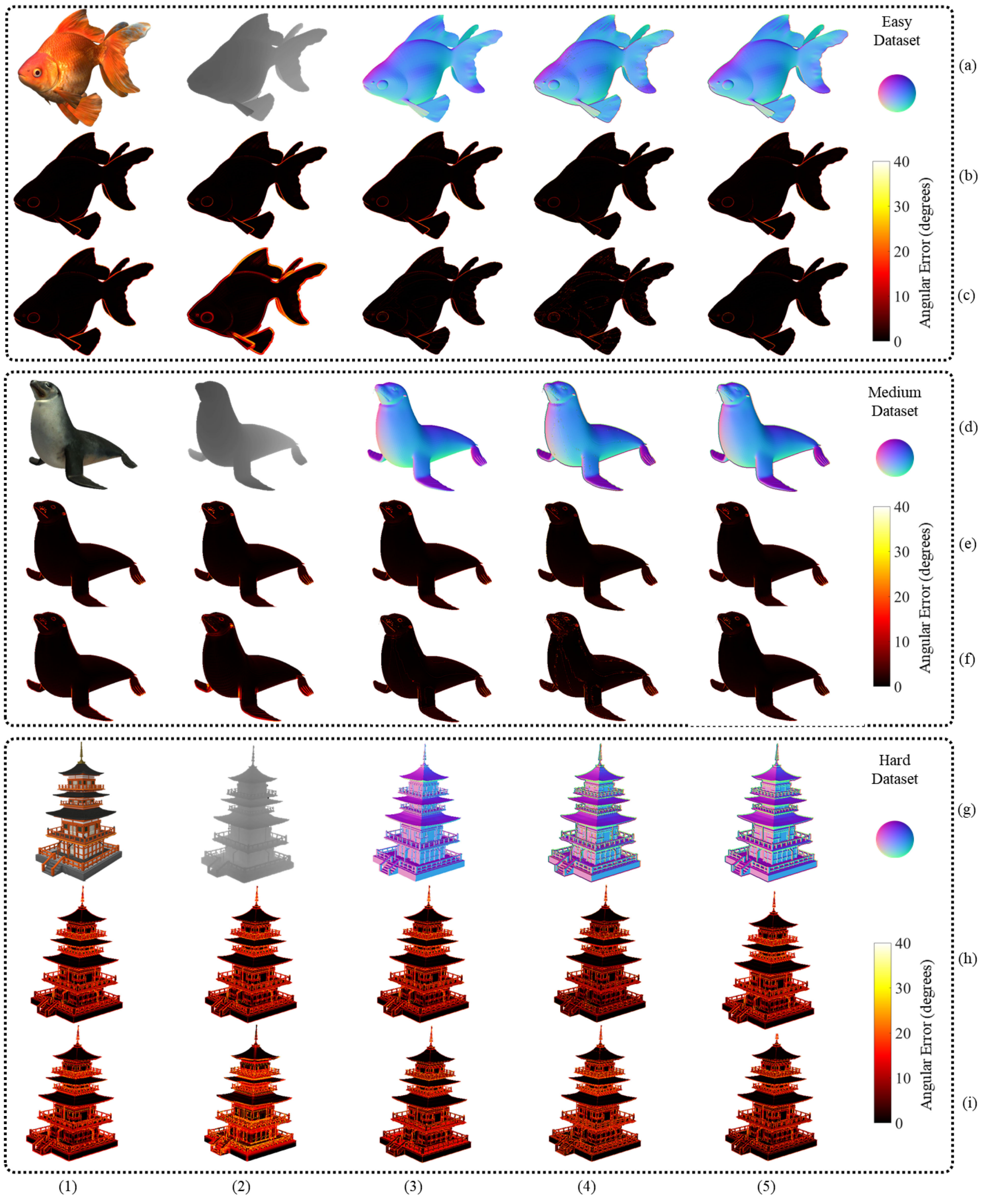

Fig. 3. Examples of the experimental results: (1)–(5) columns on (a), (d) and (g) rows show the 3D mesh models, depth images, surface normal ground truth and the experimental results obtained using FD-Mean and FD-Median SNEs, respectively; (1)–(5) columns on (b), (e) and (h) rows show the angular error maps obtained by PlaneSVD/PlanePCA [13], VectorSVD [2], AreaWeighted [2], AngleWeighted [2] and FALS [5] SNEs, respectively; (1)–(5) columns on (c), (f) and (i) rows show the angular error maps obtained by SRI [5], LINE-MOD [1], SNE-RoadSeg [12], FD-Mean and FD-Median SNEs, respectively.

100 times faster than the latter. Furthermore, we add random Gaussian noise to our created depth images and provide comprehensive comparisons of these SNEs in the supplement at sites.google.com/view/3f2n/supplement.

In addition to our created datasets, we also use the DIODE [17] and ScanNet [19] datasets, respectively, to compare the performances of the above-mentioned SNEs on noisy depth data. Examples of our experimental results are shown in Figs. 4 and 5, respectively. The runtime and average angular errors obtained by different SNEs are given in Tables V and VI, respectively, where it can be seen that FD-Mean SNE is the fastest among all SNEs, while FD-Median SNE achieves the lowest $e_{\rm p}$ when

TABLE V
COMPARISONS AMONG DIFFERENT GEOMETRY-BASED SNEs ON THE DIODE DATASET [17]

Method	t (ms) ↓	e_{A} (degrees) ↓		π (degrees/kHz) ↓		e_{p} ↑					
		Indoor	Outdoor	Indoor	Outdoor	Indoor			Outdoor		
						φ=10°	φ=20°	φ=30°	φ=10°	φ=20°	φ=30°
PlaneSVD [14]	883.458	10.888	16.579	9619.002	14646.762	0.693	0.924	0.942	0.574	0.763	0.811
PlanePCA [13]	1501.707	10.888	16.579	16350.436	24896.650	0.693	0.924	0.942	0.574	0.763	0.811
VectorSVD [4]	1327.847	10.868	16.514	14431.572	21928.464	0.696	0.925	0.942	0.577	**0.764**	0.812
AreaWeighted [4]	2522.729	10.887	16.560	27465.203	41775.635	0.691	0.924	0.942	0.572	0.763	0.812
AngleWeighted [4]	2661.607	10.759	16.545	28636.496	44037.086	0.689	**0.925**	**0.943**	0.568	0.763	**0.815**
FALS [5]	10.706	11.072	16.671	118.531	178.474	0.682	0.923	0.941	0.571	0.759	0.813
SRI [5]	39.075	11.154	16.903	435.854	660.481	0.685	0.918	0.936	0.571	0.757	0.807
LINE-MOD [3]	17.026	12.839	17.272	218.593	294.071	0.663	0.886	0.907	0.577	0.749	0.796
SNE-RoadSeg [12]	20.310	**10.316**	**15.431**	209.599	313.383	0.692	0.921	0.941	0.555	0.760	0.810
FD-Mean (ours)	**9.511**	11.202	16.981	**106.540**	**161.507**	0.613	0.854	0.903	0.477	0.713	0.779
FD-Median (ours)	30.193	10.589	16.254	319.705	490.769	**0.706**	0.922	0.940	**0.578**	0.761	0.809

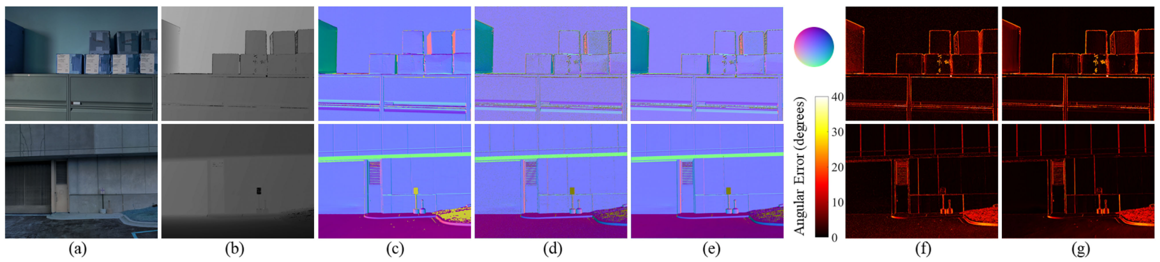

Fig. 4. Examples of the DIODE dataset [17]: (a) RGB images; (b) depth images; (c) surface normal ground truth; (d) FD-Mean SNE results; (e) FD-Median SNE results; (f) FD-Mean SNE error maps; (g) FD-Median SNE error maps.

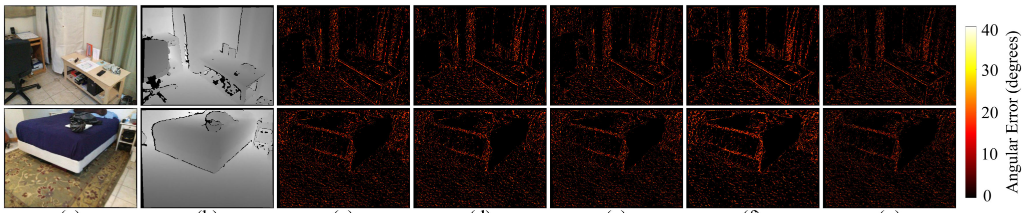

Fig. 5. Examples of the ScanNet dataset [19]: (a) RGB images; (b) depth images; (c) PlaneSVD/PlanePCA SNE error maps; (d) AngleWeighted error maps; (e) SNE-RoadSeg SNE error maps; (f) FD-Mean SNE error maps; (g) FD-Median SNE error maps.

TABLE VI
COMPARISONS AMONG DIFFERENT GEOMETRY-BASED SNEs ON THE SCANNET DATASET [19]

Method	t (ms) ↓	e_{A} (degrees) ↓	π (degrees/kHz) ↓	e_{p} ↑		
				φ=10°	φ=20°	φ=30°
PlaneSVD [14]	462.349	13.164	6086.362	0.645	0.861	0.890
PlanePCA [13]	782.475	13.164	10300.501	0.645	0.861	0.890
VectorSVD [4]	687.917	13.239	9107.333	0.646	0.856	0.887
AreaWeighted [4]	1391.188	13.213	18381.767	0.641	0.858	0.889
AngleWeighted [4]	1475.558	12.958	19120.281	0.642	0.863	**0.894**
FALS [5]	5.308	13.256	70.363	0.639	0.860	0.891
SRI [5]	15.704	13.626	213.983	0.637	0.849	0.881
LINE-MOD [3]	7.679	14.479	111.184	0.631	0.834	0.866
SNE-RoadSeg [12]	10.634	12.669	134.722	0.630	0.847	0.881
FD-Mean (ours)	**4.630**	13.225	**61.232**	0.565	0.805	0.865
FD-Median (ours)	13.924	**12.628**	175.832	**0.651**	**0.864**	0.893

$\varphi = 10°$. FD-Mean greatly minimizes the trade-off between speed and accuracy. Therefore, 3F2N SNE outperforms all other state-of-the-art geometry-based SNEs in terms of both accuracy and speed. Researchers can use either FD-Mean SNE or FD-Median SNE in their work, according to their demand for speed or accuracy.

V. DISCUSSION

An SNE can be applied in a variety of computer vision and robotics tasks. In this letter, we perform ElasticFusion [20], a real-time dense visual simultaneous localization and mapping (SLAM) algorithm, on the ICL-NUIM RGB-D dataset [21] with and without surface normal information incorporated, respectively. According to the quantitative analysis of our experimental results, the 3D geometry reconstruction accuracy can be improved by approximately 19%, when using the surface normal information obtained by 3F2N SNE. Examples of the experimental results are given in the supplement.

Moreover, it has been proven in [12] and [22] that surface normal information can be employed for various planar surface segmentation applications. Therefore, we believe that 3F2N SNE can be utilized to extract informative features for CNNs in various autonomous driving perception tasks, without affecting their training/prediction speed.

Finally, it is emphasized that the proposed SNE is essentially different from the approaches developed for dominant surface normal estimation (or Manhattan frame model inference [23]). The latter aims at estimating the surface normal of each planar surface instead of the one of every single pixel.

VI. Conclusion

In this letter, we presented a precise and ultrafast SNE named 3F2N for structured range data. Our proposed SNE can compute surface normals from an inverse depth image or a disparity image using three filters, namely, a horizontal image gradient filter, a vertical image gradient filter and a mean/median filter. To evaluate the performance of our proposed SNE, we created three datasets (containing about 60 k pairs of depth images and the corresponding surface normal ground truth) using 24 3D mesh models. Our datasets are also publicly available for research purposes. According to our experimental results, FD outperforms other image gradient filters, *e.g.*, Sobel, Scharr and Prewitt, in terms of both precision and speed. FD-Median SNE achieves the best surface normal precision (1.66°, 5.69° and 15.31° on easy, medium and hard datasets, respectively), while FD-Mean SNE is most effective for minimizing the trade-off between speed and accuracy. Furthermore, our proposed 3F2N SNE achieves better overall performance than all other geometry-based SNEs.

References

[1] S. Hinterstoisser *et al.*, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 876–888, May 2011.
[2] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 3206–3211.
[3] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5556–5565.
[4] S. Martull, M. Peris, and K. Fukui, "Realistic CG stereo image dataset with ground truth disparity maps," in *Proc. Int. Conf. Pattern Recognit. workshop TrakMark*, vol. 111, no. 430, 2012, pp. 117–118.
[5] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3084–3091.
[6] F. Lu, X. Chen, I. Sato, and Y. Sato, "SymPS: Brdf symmetry guided photometric stereo for shape and light source estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 221–234, Jan. 2018.
[7] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2D-3 D alignment via surface normal prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5965–5974.
[8] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFS," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1119–1127.
[9] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "GeoNet: Geometric neural network for joint depth and surface normal estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 283–291.
[10] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-Net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 675–684.
[11] J. Huang, Y. Zhou, T. Funkhouser, and L. J. Guibas, "FrameNet: Learning local canonical frames of 3 d surfaces from a single RGB image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 8638–8647.
[12] R. Fan, H. Wang, P. Cai, and M. Liu, "SNE-RoadSeg: Incorporating surface normal information into semantic segmentation for accurate freespace detection," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 340–356.
[13] K. Jordan and P. Mordohai, "A quantitative evaluation of surface normal estimation in point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 4220–4226.
[14] K. Klasing, D. Wollherr, and M. Buss, "Realtime Segmentation of Range Data Using Continuous Nearest Neighbors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 2431–2436.
[15] R. Fan, U. Ozgunalp, B. Hosking, M. Liu, and I. Pitas, "Pothole detection based on disparity transformation and road surface modeling," *IEEE Trans. Image Process.*, vol. 29, pp. 897–908, 2019, doi: 10.1109/TIP.2019.2933750.
[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge University Press, 2003.
[17] I. Vasiljevic *et al.*, "DIODE: A dense indoor and outdoor DEpth dataset," *CoRR*, 2019.
[18] R. Fan, L. Wang, M. J. Bocus, and I. Pitas, "Computer stereo vision for autonomous driving," CoRR, 2020.
[19] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3 d reconstructions of indoor scenes," in *Proc. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
[20] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," *Robot.: Sci. Syst.*, 2015.
[21] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, Hong Kong, China, May 2014, pp. 1524–1531.
[22] H. Wang, R. Fan, Y. Sun, and L. Ming, "Dynamic fusion module evolves drivable area and road anomaly detection," *IEEE Trans. Cybern.*, 2021, doi: 10.1109/TCYB.2021.3064089.
[23] J. Straub, O. Freifeld, G. Rosman, J. J. Leonard, and J. W. Fisher, "The manhattan frame model-manhattan world inference in the space of surface normals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 235–249, Jan. 2018.