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

Rui Fan [1], Hengli Wang [1], Bohuan Xue [1], Huaiyang Huang [1], Yuan Wang, [1], Ming Liu [1], and Ioannis Pitas [1]

[1]Affiliation not available

October 30, 2023

Abstract

Over the past decade, significant efforts have been made to improve the trade-off between speed and accuracy of surface normal estimators (SNEs). This paper introduces an accurate and ultrafast SNE for structured range data. The proposed approach computes surface normals by simply performing three filtering operations, namely, 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 the method, no similar method already exists in the literature. In our experiments, we created three large-scale synthetic datasets (easy, medium and hard) using 24 3-dimensional (3D) mesh models. Each mesh model is used to generate 1800–2500 pairs of 480x640 pixel depth images and the corresponding surface normal ground truth from different views. The average angular errors with respect to the easy, medium and hard datasets are 1.6 degrees, 5.6 degrees and 15.3 degrees, respectively. Our C++ and CUDA implementations achieve a processing speed of over 260 Hz and 21 kHz, respectively. Our proposed SNE achieves a better overall performance than all other existing computer vision-based SNEs. Our datasets and source code are publicly available at: sites.google.com/view/3f2n.

Posted on 27 May 2020 — CC-BY 4.0 — https://doi.org/10.36227/techrxiv.12362249.v1 — This is a preprint. Version of Record available at https://doi.org/10.1109/LRA.2021.3067308

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*, Ioannis Pitas, *Fellow, IEEE*

***Abstract*—Over the past decade, significant efforts have been made to improve the trade-off between speed and accuracy of surface normal estimators (SNEs). This paper introduces an accurate and ultrafast SNE for structured range data. The proposed approach computes surface normals by simply performing three filtering operations, namely, 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 the method, no similar method already exists in the literature. In our experiments, we created three large-scale synthetic datasets (easy, medium and hard) using 24 3-dimensional (3D) mesh models. Each mesh model is used to generate 1800–2500 pairs of 480×640 pixel depth images and the corresponding surface normal ground truth from different views. The average angular errors with respect to the easy, medium and hard datasets are 1.6°, 5.6° and 15.3°, respectively. Our C++ and CUDA implementations achieve a processing speed of over 260 Hz and 21 kHz, respectively. Our proposed SNE achieves a better overall performance than all other existing computer vision-based SNEs. Our datasets and source code are publicly available at: sites.google.com/view/3f2n.**

***Index Terms*—surface normal, structured range data, filters, synthetic datasets.**

I. Introduction

REAL-TIME 3-dimensional (3D) object recognition is a very challenging computer vision task [3]. Surface normal is an informative and important feature descriptor used in 3D object recognition [4]. Over the past decade, there has not been much research on surface normal estimation, as it is merely considered as an auxiliary functionality for other computer vision applications. However, such applications are generally required to perform in an online fashion, and thus, the estimation of surface normals must be carried out extremely fast [4].

The surface normals can be estimated from either a 3D point cloud or a depth/disparity image (see Figure 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, *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 researcher have been focused on surface normal estimation from structured range data, *i.e.*, depth/disparity images.

The existing surface normal estimators (SNEs) can be classified as either computer vision-based [3]–[6] or machine learning-based [7]–[13]. The former typically computes the 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) [4]. 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 [12].

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 example, Xu *et al.* [7] utilized a so-called prediction-and-distillation network (PAD-Net) to simultaneously solve two continuous regression tasks (monocular depth prediction and surface normal inference) and two discrete classification tasks (scene parsing and contour detection). Similarly, Li *et al.* [13] designed a DCNN model to learn the mapping from multi-scale image patches to surface normals and monocular depth. Such inferences were then refined using conditional random fields (CRF) [14]. Furthermore, Bansal *et al.* [10] built a skip-network model based on a pre-trained Oxford VGG-16 CNN [15] for 2.5D surface normal prediction and 3D object recognition in 2D images. Recently, Huang *et al.* [16] 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 labeled training data to find the best CNN parameters [13]. 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, *e.g.*, scene parsing [7], 3D object detection [9], depth perception [13], *etc.* Furthermore, many robotics and computer vision applications, *e.g.*, autonomous driving, require very fast surface normal estimation (in milliseconds). Unfortunately, the existing machine/deep learning-based SNEs are not that fast. Moreover, the accuracy achieved by data-driven SNEs is still far from satisfactory (the average proportion of good pixels, detailed in Section IV, is usually lower than 80%) [10], [13]. Most importantly, it can be considered more

R. Fan is with the Jacobs School of Engineering and the UCSD Health, the University of California, San Diego, La Jolla, CA 92093, United States (e-mail: rui.fan@ieee.org).

H. Wang, B. Xue, H. Huang, Y. Wang and M. liu are with the Robotics Institute, the Hong Kong University of Science and Technology, Hong Kong (e-mail: {hwangdf, bxueaa, hhuangat, ywangeq, eelium}@connect.ust.hk).

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

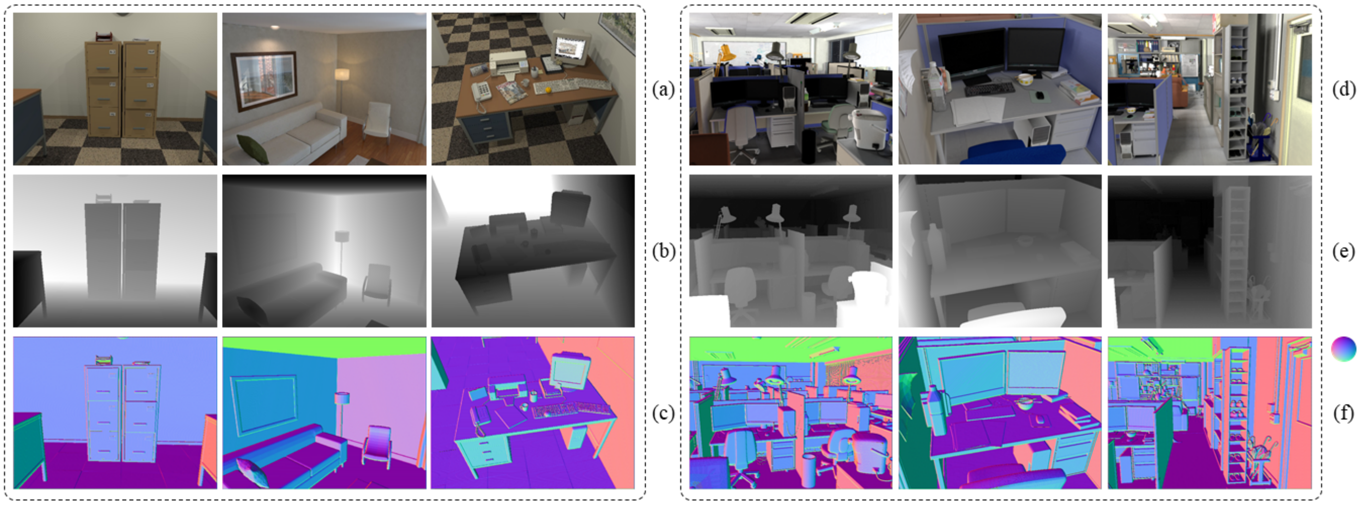


Fig. 1. Surface normal estimation from depth/disparity images: (a) and (b) show three examples of RGB and depth images of the Augmented ICL-NUIM dataset [1], respectively; (d) and (e) show three examples of RGB and disparity images of the Tsukuba stereo dataset [2], respectively; (c) and (f) show the surface normals estimated from (b) and (e), respectively, using the proposed SNE.

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 main novel contributions of this work are as follows:

a) A novel, accurate and ultrafast SNE is proposed. We implement our SNE in Matlab C, C++ and CUDA. The source code will be publicly available at IEEE Xplore for research purposes. Compared with other computer vision-based SNEs, the proposed SNE greatly improves the trade-off between speed and accuracy.

b) Three datasets (easy, medium and hard) are 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.

The rest of this paper continues in the following manner: Section II reviews the state-of-the-art computer vision-based SNEs; Section III introduces our proposed SNE; the experimental results and the performance evaluation are provided in Section IV; in Section V, we discuss the applications of our SNE; finally, Section VI summarizes the paper and provides recommendations for future work.

II. Related Work

This section provides an overview of computer vision-based SNEs.

1) PlaneSVD SNE [17]: 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:

$$\min_{\mathbf{b}_i} \left\| \begin{bmatrix} \mathbf{Q}_i^+ & \mathbf{1}_{k+1} \end{bmatrix} \mathbf{b}_i \right\|_2, \tag{2}$$

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 $\mathbf{U}\Sigma\mathbf{V}^\top$ 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 Σ [4].

2) PlanePCA SNE [18]: $\mathbf{n}_i$ 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:

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

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 [4].

3) VectorSVD SNE [4]: 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,

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

This minimization is done by SVD.

4) AreaWeighted SNE [4]: 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 [4]:

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

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 [4]: 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:

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

where $\mathbf{V}_i^+ = [\mathbf{v}_i, \mathbf{v}_{i\,1}, \dots, \mathbf{v}_{i\,k}]^\top$, $\tilde{\mathbf{n}}_i = \mathbf{n}_i / b^2$ and $\mathbf{s}_i = [r_i^{-1}, {r_i}_1^{-1}, \dots, {r_i}_k^{-1}]^\top$.

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:

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

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 [3]: 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 $\mathbf{p}_2$:

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

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. 3F2N SNE

In this paper, we propose a novel, highly accurate and ultrafast SNE, which is simple to understand and use. Our SNE can compute surface normals from structured range data using three filters, namely, a horizontal image gradient filter, a vertical image gradient filter and a mean/median filter. Hence, we call it three-filters-to-normal (3F2N) SNE.

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 [19]:

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

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:

$$\frac{\partial 1/z}{\partial u} = -\frac{n_x}{b f_x}, \qquad \frac{\partial 1/z}{\partial v} = -\frac{n_y}{b f_y}, \tag{15}$$

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:

$$n_x = -b f_x \frac{\partial 1/z}{\partial u}, \qquad n_y = -b f_y \frac{\partial 1/z}{\partial v}. \tag{16}$$

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

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

where $\mathbf{r}_{i\,j} = \mathbf{q}_{i\,j} - \mathbf{p}_i = [\Delta x_{i\,j}, \Delta y_{i\,j}, \Delta z_{i\,j}]^\top$. In this paper, $k = 8$ and $\mathbf{Q}_i$ is an 8-connected neighborhood. Since (16) and (17) have a common factor of $-b$, they can be simplified as:

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

where $\Phi\{\cdot\}$ is a mean or median operator used to estimate n_z. Please note: if the depth value of $\mathbf{p}_i$ is identical to those of all its neighboring points $\mathbf{q}_{i\,j} \in \mathbf{Q}_i$, 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 the depth z and disparity d is as follows:

$$z = \frac{f t_c}{d}, \tag{19}$$

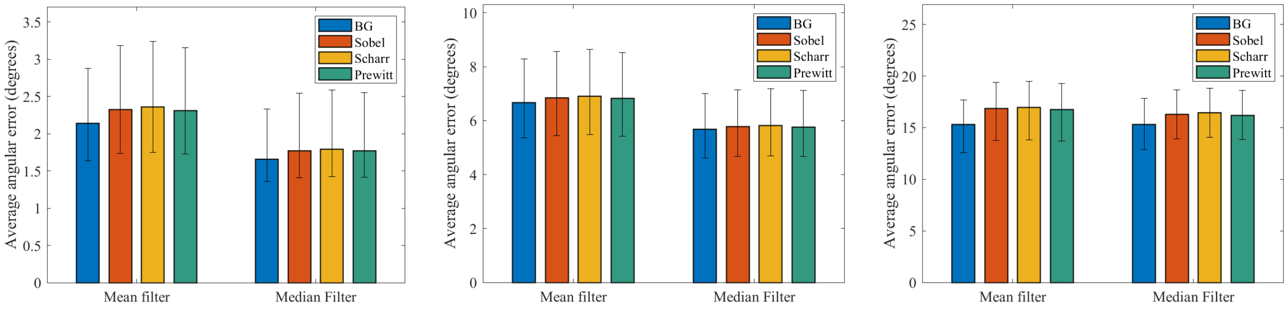
where t_c is the stereo rig baseline. Therefore,

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

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

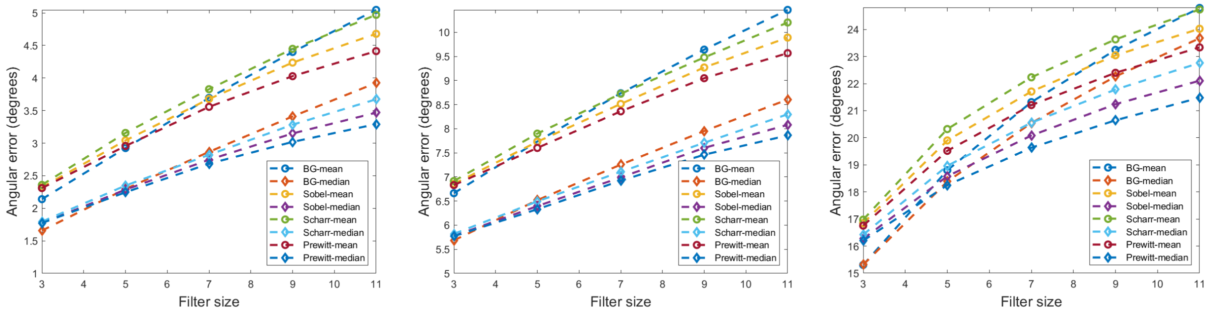
$$\begin{aligned} & n_x = \partial d / \partial u, \qquad n_y = \partial d / \partial v, \\ & \hat{n}_z = -\Phi \left\{ \frac{\Delta x_{i\,j} n_x + \Delta y_{i\,j} n_y}{\Delta z_{i\,j}} \right\}, \quad j = 1, \dots, k. \end{aligned} \tag{21}$$

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



(a) (b) (c)

Fig. 2. e_{A} comparisons with respect to different image gradient filters and mean/median filter: (a) easy dataset; (b) medium dataset; (c) hard dataset. Please note: (a), (b) and (c) use different scales.



(a) (b) (c)

Fig. 3. e_{A} comparisons with respect to different filter sizes: (a) easy dataset; (b) medium dataset; (c) hard dataset. Please note: (a), (b) and (c) use different scales.

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
BG	**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
BG-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
BG-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

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 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. In addition to our datasets, we also utilize the DIODE dataset[3] [20] to evaluate the SNE performance.

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:

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

[1] free3d.com

[2] www.opengl.org/sdk/docs/tutorials/ClockworkCoders/glsl_overview.php

[3] diode-dataset.org

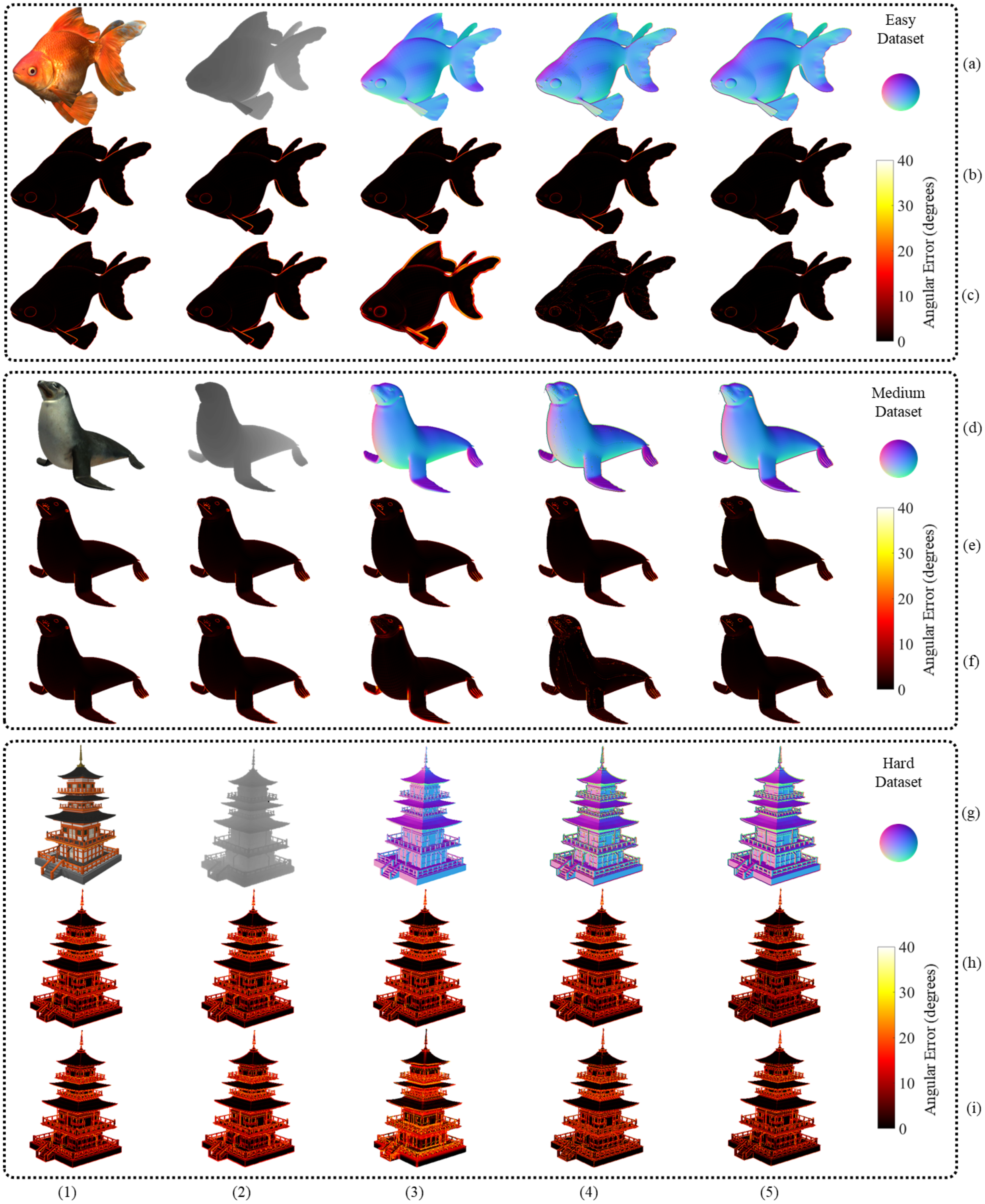


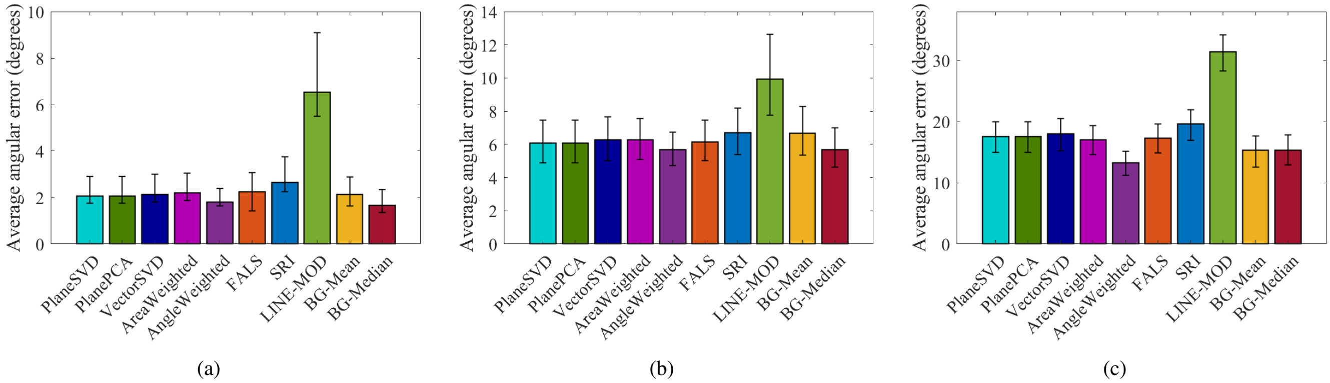
Fig. 4. Examples of the experimental results: (1)–(5) columns on (a), (d) and (g) rows show the 3D mesh models, depth images, surface normal ground truth and the experimental results obtained using BG-Mean and BG-Median SNEs, respectively; (1)–(5) columns on (b), (e) and (h) rows show the angular error maps obtained by PlaneSVD, PlanePCA, VectorSVD, AreaWeighted and AngleWeighted SNEs, respectively; (1)–(5) columns on (c), (f) and (i) rows show the angular error maps obtained by FALS, SRI, LINE-MOD, BG-Mean and BG-Median SNEs, respectively.

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

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

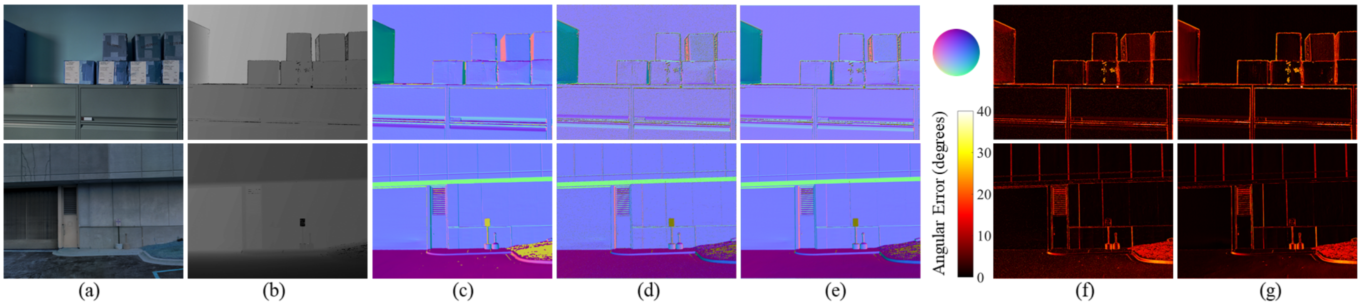


(a)

(b)

(c)

Fig. 5. e_{A} comparisons among different computer vision-based SNEs: (a) easy dataset; (b) medium dataset; (c) hard dataset. Please note: (a), (b) and (c) use different scales.



(a) (b) (c) (d) (e) (f) (g)

Fig. 6. Examples of the DIODE dataset: (a) RGB images; (b) depth images; (c) surface normal ground truth; (d) BG-Mean SNE results; (e) BG-Median SNE results; (f) BG-Mean SNE error maps; (g) BG-Median SNE error maps.

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. Then, the brute-force search method is utilized to find the best parameters for a 3×3 kernel. Our experiments illustrate that the basic gradient (BG) 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 are publicly available at: sites.google.com/view/3f2n. Similar to the FALS, SRI and LINE-MOD SNE implementations provided in the opencv_contrib repository,[4] 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.

TABLE III
THE COMPARISONS OF RUNTIME (MS) AND π SCORES AMONG DIFFERENT COMPUTER VISION-BASED SNES.

Method	t (ms)	π (degrees/kHz)		
		Easy	Medium	Hard
PlaneSVD [18]	393.69	813.87	2389.73	6923.18
PlanePCA [17]	631.88	1306.29	3835.59	11111.92
VectorSVD [4]	563.21	1199.63	3529.11	10142.34
AreaWeighted [4]	1092.24	2407.74	6843.56	18600.68
AngleWeighted [4]	1032.88	1850.00	5855.62	13693.24
FALS [5]	4.11	9.26	25.20	71.17
SRI [5]	12.18	32.18	81.66	238.78
LINE-MOD [3]	6.43	41.93	63.84	202.08
BG-Mean	**3.72**	**7.96**	**24.80**	**56.96**
BG-Median	10.97	18.18	62.38	168.03

C. Performance Evaluation

We first compare the performances of the proposed SNE with respect to different image gradient filters (BG, Sobel, Scharr and Prewitt) and mean/median filter. e_{A} scores with respect to the easy, medium and hard datasets are illustrated in Figure 2. The runtime of our implementations on an Intel Core i7-8700K 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 Table I and II, respectively. We can see that BG 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

[4] github.com/opencv/opencv_contrib

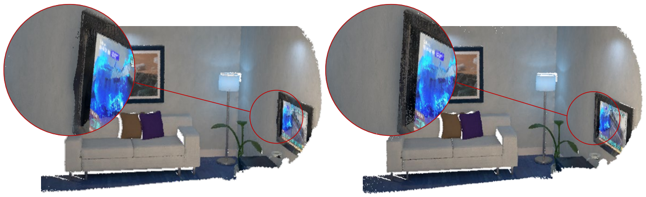
TABLE IV

e_{P} COMPARISON AMONG DIFFERENT COMPUTER VISION-BASED SNES WITH RESPECT TO DIFFERENT φ ON EASY, MEDIUM AND HARD 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 [18]	0.9648	0.9792	0.9855	0.8621	0.9531	0.9718	0.6202	0.7394	0.7914
PlanePCA [17]	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
BG-Mean	0.9563	0.9767	0.9864	0.8349	0.9423	0.9674	0.6191	0.7671	0.8368
BG-Median	0.9723	0.9829	0.9889	0.8722	0.9600	0.9766	**0.6631**	0.7821	0.8289

truth value, introducing significant noise to the mean filter. The e_{A} scores achieved using BG-Median SNE are approximately 1.0°, 0.8° and 0.1° (with respect to the easy, medium and hard datasets, respectively) higher than those obtained using BG-Mean SNE. Furthermore, Figure 3 illustrates the values of e_{A} with respect to different filter sizes, where readers can see that e_{A} decreases gradually with the increase of the filter size. 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 Table I and II, we can observe that both BG-Mean SNE and BG-Median SNE perform much faster than real-time across different computing platforms. The processing speed of BG-Mean SNE is over 1 kHz and 21 kHz on the Jetson TX2 GPU and RTX 2080 Ti GPU, respectively. Furthermore, BG-Mean SNE performs around 1.4 to 2.1 times faster than the BG-Median SNE. Therefore, the latter achieves the best surface normal accuracy, while the former achieves the best processing speed.

Moreover, we compare our SNE with all other computer vision-based SNEs, as mentioned in Section II. Some examples of the experimental results are shown in Figure 4, where it can be seen that the bad estimates mainly reside on the object edges. Additionally, Figure 5 shows comparisons of e_{A} on the easy, medium and hard datasets, where we can find that BG-Median SNE achieves the best e_{A} score on the easy dataset, while AngleWeighted SNE achieves the best e_{A} scores on the medium and hard datasets. Meanwhile, the e_{A} scores achieved by BG-Median SNE and AngleWeighted 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 BG-Mean SNE achieves the fastest processing speed. Furthermore, BG-Mean, FALS and BG-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 SNE achieves the best e_{P} scores, except for $\varphi = 10°$ (hard dataset). However, according to Table III, AngleWeighted SNE is extremely time-consuming and achieves a very bad π score. On the other hand, BG-Median SNE and AngleWeighted SNE achieve similar e_{P} scores, but the former performs about 100 times faster than the latter.



(a) (b)

Fig. 7. 3D scene reconstruction comparison: (a) conventional 3D scene reconstruction; (b) 3D scene reconstruction aided by our proposed SNE.

In addition to our created datasets, we also use the DIODE dataset [20] to compare the performances of the above-mentioned SNEs. Examples of our experimental results are shown in Figure 6. The runtime and average angular errors obtained by different SNEs are given in Table V, where it can be seen that BG-Mean SNE is the fastest among all SNEs, while BG-Median SNE achieves the lowest average angular errors. Therefore, 3F2N SNE outperforms all other state-of-the-art computer vision-based SNEs in terms of both accuracy and speed. Researchers can use either BG-Mean SNE or BG-Median SNE in their work, according to their demand for speed or accuracy.

V. Discussion

A SNE can be applied in a variety of computer vision and robotics tasks. In this section, we first use the ICL-NUIM RGB-D dataset [21] to show an example of 3D geometry reconstruction benefiting from 3F2N SNE. Then, we discuss the possibilities of using 3F2N SNE to improve the performance of the state-of-the-art CNNs.

In our experiments, we first utilize an off-the-shelf registration algorithm provided by the point cloud library[5] (PCL) to match the 3D point cloud generated from each depth image with a global 3D geometry model. The sensor poses and motion trajectory can then be obtained. Meanwhile, we integrate the surface normal information into the point cloud registration process and acquire another collection of sensor poses and motion trajectory. Then, we utilize ElasticFusion [22], a real-time dense visual simultaneous localization and

[5] http://pointclouds.org/

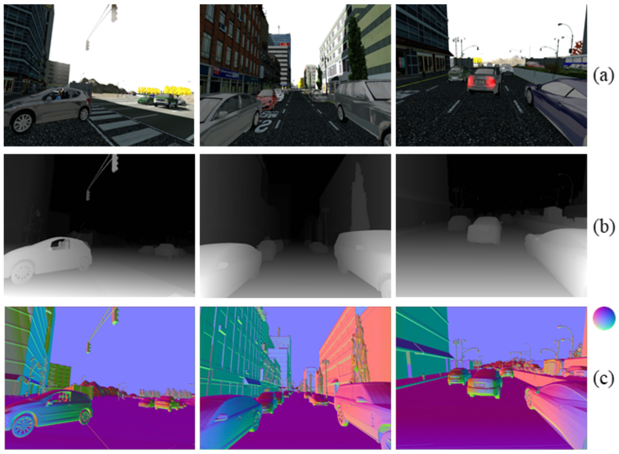


Fig. 8. Examples of the Synthia-SF dataset: (a) RGB images; (b) disparity images; (c) 3F2N SNE results.

TABLE V
THE RUNTIME (MS) AND e_A COMPARISONS AMONG DIFFERENT COMPUTER VISION-BASED SNES ON THE DIODE DATASET.

Method	Runtime (ms)	e_A (degrees)	
		indoor	outdoor
PlaneSVD [18]	883.458	10.8879	16.5789
PlanePCA [17]	1501.707	10.8879	16.5789
VectorSVD [4]	1327.847	10.8684	16.5143
AreaWeighted [4]	2522.729	10.8871	16.5597
AngleWeighted [4]	2661.607	10.7591	16.5453
FALS [5]	10.706	11.0715	16.6705
SRI [5]	39.075	11.1543	16.9029
LINE-MOD [3]	17.026	12.8388	17.2719
BG-Mean	**9.511**	11.2018	16.9811
BG-Median	30.193	**10.5887**	**16.2544**

mapping (SLAM) system, to reconstruct the 3D scenery using the input RGB-D data and two collections of sensor poses and motion trajectories. Two reconstructed 3D scenes are illustrated in Figure 7, where it is obvious that the proposed SNE can improve the 3D geometry reconstruction accuracy. According to the quantitative analysis of our experimental results, the 3D reconstruction accuracy can be improved by approximately 19%, when using the surface normal information obtained by 3F2N SNE.

Furthermore, we perform 3F2N SNE on the disparity images provided in the Synthia-SF dataset [23]. Examples of the experimental results are shown in Figure 8. It can be seen that the 3D points on each planar (or near planar) surface, such as a road or building side, possess similar surface normals. Therefore, we believe that our proposed SNE can be utilized to extract informative features for CNNs in various autonomous driving perception tasks, such as semantic image segmentation and freespace detection, without affecting their training/prediction speed.

VI. CONCLUSION AND FUTURE WORK

In this paper, 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 60k pairs of depth images and the corresponding surface normal ground truth) using 24 3D mesh models. Our datasets are publicly available at https://sites.google.com/view/3f2n for research purposes. According to our experimental results, BG outperforms other image gradient filters, *e.g.*, Sobel, Scharr and Prewitt, in terms of both precision and speed. BG-Median SNE achieves the best surface normal precision (1.6°, 5.6° and 15.3° on easy, medium and hard datasets, respectively), while BG-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 computer vision-based SNEs. We believe that our SNE can be easily applied in various computer vision and robotics tasks, *e.g.*, autonomous driving, *etc*.

As a future work, we plan to use the proposed method to learn depth prediction from monocular images, as many methods have already applied the constraints between depth and normal in monocular depth prediction.

REFERENCES

[1] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[2] S. Martull, M. Peris, and K. Fukui, "Realistic cg stereo image dataset with ground truth disparity maps," in *ICPR workshop TrakMark2012*, vol. 111, no. 430, 2012, pp. 117–118.

[3] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2011.

[4] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3206–3211.

[5] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 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 transactions on pattern analysis and machine intelligence*, vol. 40, no. 1, pp. 221–234, 2017.

[7] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 675–684.

[8] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille, "Surge: Surface regularized geometry estimation from a single image," in *Advances in Neural Information Processing Systems*, 2016, pp. 172–180.

[9] T. Hashimoto and M. Saito, "Normal estimation for accurate 3d mesh reconstruction with point cloud model incorporating spatial structure," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 54–63.

[10] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2d-3d alignment via surface normal prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5965–5974.

[11] S. Tozza, W. A. Smith, D. Zhu, R. Ramamoorthi, and E. R. Hancock, "Linear differential constraints for photo-polarimetric height estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2279–2287.

[12] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "Geonet: Geometric neural network for joint depth and surface normal estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 283–291.

[13] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1119–1127.

[14] H. M. Wallach, "Conditional random fields: An introduction," *Technical Reports (CIS)*, p. 22, 2004.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[16] J. Huang, Y. Zhou, T. Funkhouser, and L. J. Guibas, "Framenet: Learning local canonical frames of 3d surfaces from a single rgb image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8638–8647.

[17] K. Jordan and P. Mordohai, "A quantitative evaluation of surface normal estimation in point clouds," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4220–4226.

[18] K. Klasing, D. Wollherr, and M. Buss, "Realtime segmentation of range data using continuous nearest neighbors," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2431–2436.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] I. Vasiljevic, N. Kolkin, S. Zhang, R. Luo, H. Wang, F. Z. Dai, A. F. Daniele, M. Mostajabi, S. Basart, M. R. Walter, and G. Shakhnarovich, "DIODE: A Dense Indoor and Outdoor DEpth Dataset," *CoRR*, vol. abs/1908.00463, 2019. [Online]. Available: http://arxiv.org/abs/1908.00463

[21] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[22] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph." Robotics: Science and Systems, 2015.

[23] D. Hernandez-Juarez, L. Schneider, A. Espinosa, D. Vazquez, A. M. Lopez, U. Franke, M. Pollefeys, and J. C. Moure, "Slanted stixels: Representing san franciscos steepest streets," in *British Machine Vision Conference (BMVC), 2017*, 2017.