HAL
open science

# Deep Learning for Robust Normal Estimation in Unstructured Point Clouds

Alexandre Boulch, Renaud Marlet

**To cite this version:**

Alexandre Boulch, Renaud Marlet. Deep Learning for Robust Normal Estimation in Unstructured Point Clouds. Computer Graphics Forum, 2016, 35 (5), pp.281 - 290. 10.1111/cgf.12983 . hal-01631736

**HAL Id: hal-01631736**

**https://hal.science/hal-01631736v1**

Submitted on 9 Nov 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

*Volume 35 (2016), Number 5*

# Deep Learning for Robust Normal Estimation in Unstructured Point Clouds

Alexandre Boulch[1] Renaud Marlet[2]
[1]ONERA - The French Aerospace Lab, F-91761 Palaiseau, France [2]LIGM, UMR 8049, Ecole des Ponts, UPE, Champs-sur-Marne, France

**Abstract**

*Normal estimation in point clouds is a crucial first step for numerous algorithms, from surface reconstruction and scene understanding to rendering. A recurrent issue when estimating normals is to make appropriate decisions close to sharp features, not to smooth edges, or when the sampling density is not uniform, to prevent bias. Rather than resorting to manually-designed geometric priors, we propose to learn how to make these decisions, using ground-truth data made from synthetic scenes. For this, we project a discretized Hough space representing normal directions onto a structure amenable to deep learning. The resulting normal estimation method outperforms most of the time the state of the art regarding robustness to outliers, to noise and to point density variation, in the presence of sharp edges, while remaining fast, scaling up to millions of points.*

## 1. Introduction

Numerous algorithms have been developed to process point clouds, such as geometric primitive extraction [SWK07], surface reconstruction [BDLGM14, CLP10], 3D navigation [FGMP14], and point-based rendering [RL00, ZPVBG01, ABCO*03], just to name a few. For many of them, the performance significantly depends on the quality of normals estimated at each point.

Normal estimation is a well-studied topic. The problem is to infer the local orientation of the unknown surface underlying a point cloud. A good estimator should not be sensitive to outliers, to noise and to variations of point density, which are common due to the way point clouds are captured, e.g., as merged laser scans, fusion of depth images, or structure-from-motion. Non-uniform sampling, with possible anisotropic bias, occurs ordinarily due to varying incidences on scanned surfaces. Moreover, as many captured scenes include man-made objects, they generally feature sharp edges and corners, that have to be preserved and not smoothed. Last, estimation should be fast, typically to scale to millions of points.

We propose here a novel method for normal estimation in unorganized point clouds, that is robust to noise, to outliers and to density variation, in the presence of sharp edges. It is based on a robust randomized Hough transform [BM12], but rather than designing explicit criteria to select a normal from the accumulator, we learn a function for doing it using a convolutional neural network (CNN). To our knowledge, this is the first application of deep learning techniques to this kind of task, for unstructured 3D data. It outperforms most of the time the state-of-the-art of normal estimation.

## 2. Related work

Hoppe et al. [HDD*92] compute the tangent plane at a given point by regression on neighboring points. Spheres [GG07] and quadrics [CP05] have also been used to better adapt to the neighborhood and to the shape of the underlying surface. Optimal neighborhood sizes can be computed, w.r.t. curvature, sampling density and noise, to minimize the estimation error [MNG04]. However, while being robust to noise, these methods remain sensitive to outliers. Improvements have been proposed to address robustness to outliers and non-uniformity, using adaptive weights [HLZ*09]. Yet, all regression-based methods tend to smooth the normals at sharp features. Moreover, a higher robustness to outliers is usually obtained using a larger neighborhood, which makes sharp features even smoother. Minimizing the $\ell_1$ [ASGCO10] or $\ell_0$ norm [SSW15] is robust to sharp features but quite slow. Moving least squares [ABCO*03, PKKG03] and local kernel regression [ÖGG09] estimate normals as the gradient of an implicit surface, preserving sharp features, but requiring reliable normal priors as input.

Dey and Goswami [DG04] propose an original approach based on the Voronoï cells of the point cloud. The normal is chosen as the cell direction with the largest extension. It is robust to sharp features, but sensitive to noise. To address it, Alliez et al. [ACSTD07] treat cell distortion due to noise using a PCA-Voronoï approach to create elongated cell sets grouping adjacent Voronoï cells.

Li et al. [LSK*10] use sample consensus (SAC), efficiently treating noisy data and sharp features. The main drawbacks are a lack of adaptation to point density variations and a high computational time. Another line of work is based on the determination of consistent point clusters in a neighborhood to better estimate normals near edges and corners. Zhang et al. [ZCL*13] extract such clusters using low-rank subspace clustering with prior knowledge. Their method yields accurate normals, but is very slow. Using the same idea, Liu et al. [LZC*15] overcome this issue by using a different representation for subspaces, and clustering only a subset of the points before propagating the results to adjacent points. The method is much faster while being as accurate as [ZCL*13].

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

**Hough transform** is a popular tool for shape extraction [Hou62]. It is based on a change of space where the desired shape is represented by a point. Shape hypotheses populate the bins of an accumulator mapped onto this space, and the most densely populated bins identify the shapes to extract [DH72, IK88]. Originally designed for simple 2D primitive extraction in images [TM78, Dav88, SW02], it has since then been used for various purposes from 3D primitive extraction [BELN11], recognition and classification [KPW*10, PWP*11], to general model selection [Bal81]. To improve speed and scalability, Kiryati et al. [KEB91] propose a probabilistic version of the Hough transform where only a subset of the input points vote in the accumulator. A high computational efficiency is reached with the Randomized Hough Transform [XO93], where the points do not vote for all the possible shapes. Shape hypotheses are made by drawing the minimal number of points to parameterize the shape, and each such drawn hypothesis corresponds to one vote in the accumulator. It results in a sharper accumulator distribution and a faster model selection. However, it may be difficult to tune the number of hypotheses to draw before a model is estimated from the votes. The Robust Randomized Hough Transform (RRHT) addresses it [BM12].

**Convolutional neural networks,** starting with LeNet5 [LBD*89], are architectured as a sequence of convolutional and pooling layers, followed by fully-connected layers. They were mostly used in image classification, outperforming other methods by a large margin [KSH12]. Increasing layer number [SLJ*14] and size [ZF14], and using dropout to treat overfitting [HSK*12], they have been successfully applied, e.g., to object detection [GDDM14], segmentation [LSD15b] and localization [SEZ*14]. Work on normal estimation with CNNs focus on using as input RGB images [LSD*15a, WFG15], or possibly RGB-D [BRG16], but not sparse data such as unstructured 3D point clouds. CNN-based techniques have been applied to 3D data though, but with a voxel-based perspective [WSK*15], which is not accurate enough for normal estimation. Techniques to efficiently apply CNN-based methods to sparse data have been proposed too [Gra15], but they mostly focus on efficiency issues, to exploit sparsity; applications are 3D object recognition, again with voxel-based granularity, and analysis of space-time objects. An older, neuron-inspired approach [JIS03] is more relevant to normal estimation in 3D point clouds but it actually addresses the more difficult task of meshing. It uses a stochastic regularization based on neighbors, but the so-called "learning process" actually is just a local iterative optimization.

## 3. Motivation and overview of our approach

**Learning normal estimation.** Normal estimation can be formulated as a discrete classification problem in Hough space. Let $\hat{\mathbf{n}}_p$ be a normal estimated at point $p$ of point cloud $\mathcal{P}$, and $\mathbf{n}_p^*$ the ground truth normal. The problem is to find a set of normals $\{\hat{\mathbf{n}}_p\}_{p\in\mathcal{P}}$ s.t.:

$$\{\hat{\mathbf{n}}_p\}_{p\in\mathcal{P}} = \underset{\{\mathbf{n}_p\}_{p\in\mathcal{P}}}{\arg\min} \sum_{p\in\mathcal{P}} \phi(\mathbf{n}_p, \mathbf{n}_p^*) \tag{1}$$

where $\phi(.,.)$ is a distance function, such as the $\ell_2$ distance. The traditional Hough-based approach consists in first associating a normal $\mathbf{n}_b$ to each bin $b$ of an accumulator and then, for each point $p$ and associated filled accumulator $a_p$, selecting a bin $\hat{b}_p$, and thus a normal $\mathbf{n}_{b_p}$. The problem becomes a discrete version of (1):

$$\{\hat{b}_p\}_{p\in\mathcal{P}} = \underset{\{b_p\}_{p\in\mathcal{P}}}{\arg\min} \sum_{p\in\mathcal{P}} \phi(\mathbf{n}_{b_p}, \mathbf{n}_p^*) \tag{2}$$

In practice, $\mathbf{n}_p^*$ is unknown and $\hat{b}_p$ is estimated at each point with a classifier over the accumulator $a_p$.

[BM12] uses a very simple classifier: the most probable bin w.r.t. the empirical probability distribution of normals in Hough space. As pointed out in the paper itself, this selection process is subject to space discretization. To overcome this effect, the authors estimate a normal several times at $p$, randomly rotating the accumulator to change discretization boundaries. The final normal is a function of these few estimated normals, e.g., the average normal of the most voted cluster of normals. While this trick reduces discretization effects, it significantly increases the computation time and introduces additional parameters. In this paper, we directly address Eq. (1) as a continuous problem. We want to construct a function $\psi$ such that, given a filled accumulator $a_p$, we produce:

$$\psi(a_p) = \hat{\mathbf{n}}_p \tag{3}$$

This regression problem is more difficult than classification in that the regressor response covers a continuous space, not just a set of discrete values. We actually want to learn function $\psi$, using a CNN.

**CNNs for estimating normals in point clouds.** Deep learning is good at making decisions in complex settings, especially when a large number of unknown factors have a nonlinear influence. In particular, CNNs are very efficient on tasks such as object classification and detection, including when objects are severely occluded. CNNs can also address regression problems such as object pose estimation [PCFG12]. These same properties seem appropriate as well for the task of learning how to estimate normals, including in the presence of noise and when several normal candidates are possible near sharp features of the underlying surface.

The question, however, is how to interpret the local neighborhood of a 3D point as an image-like input that can be fed to a CNN. If the point cloud is structured, as given by a depth sensor, the depth map is a natural choice as CNN input. But if the point cloud is unstructured, it is not clear what to do. In this case, we propose to associate an image-like representation to the local neighborhood of a 3D point via a Hough transform. In this image (cf. Section 4), a pixel corresponds to a normal direction, and its intensity measures the number of votes for that direction; besides, pixel adjacency relates to closeness of directions. It is a planar map of the empirical probability of the different possible directions. Then, just as a CNN for ordinary images can exploit the local correlation of pixels to denoise the underlying information, a CNN for these Hough-based direction maps (cf. Section 5) might also be able to handle noise, identifying a flat peak around one direction. Similarly, just as a CNN for images can learn a robust recognizer, a CNN for direction maps might be able to make uncompromising decisions near sharp features, when different normals are candidate, opting for one specific direction rather than trading off for an average, smoothed normal. Moreover, outliers can be ignored in a simple way by limiting the size of the neighborhood, thus reducing or preventing the influence of points lying far from a more densely sampled surface.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

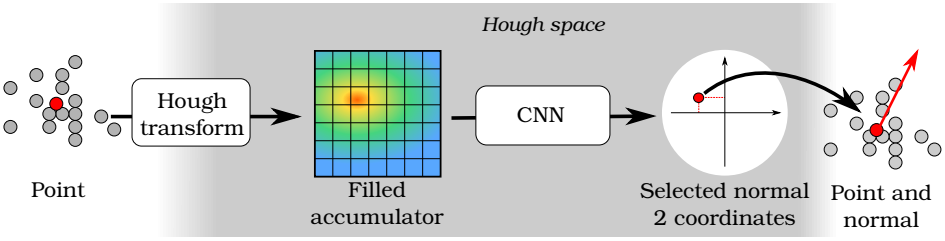



Figure 1: Our CNN-based normal estimation framework.

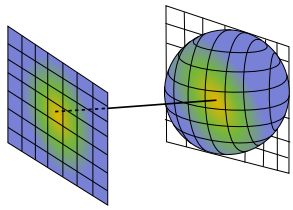


Figure 2: Distortion of bins when projected on the sphere.

**RRHT.** Applying a Hough transform to estimate the normal at a given 3D point $p$, [BM12] propose a three-step algorithm:

1. **Hypothesis generation.** Hypotheses are generated by randomly selecting three points in the neighborhood of $p$, which defines a plane and thus a possible normal direction.
2. **Vote in Hough space**. Each hypothesis votes in a 2D spherical accumulator, parameterized by spherical coordinates $\theta$ and $\phi$.
3. **Election of a normal.** Finally, the estimated normal is the average of directions in the most voted bin of the accumulator.

A contribution of [BM12] regarding running time is a robust statistical criterion to safely stop picking new hypotheses, after

$$T^* = \left\lceil \frac{1}{2\varepsilon^2} \ln\left(\frac{2M}{1-\alpha}\right) \right\rceil \tag{4}$$

are drawn, where $M$ is the number of bins of the accumulator, $\alpha \in ]0,1[$ is the confidence level, and $\varepsilon \in ]0,1[$ is the maximum distance between the empirical distribution and the theoretical distribution. The resulting method is robust to noise, outliers and sharp features but, as mentioned above, it is sensitive to bin discretization.

**Our CNN-based method,** pictured on Figure 1, keeps the same hypothesis generation scheme as [BM12], as described above (step 1), including the robust stopping criterion of Eq. (4). However, we change the accumulator and voting (step 2) to create an image structure amenable to deep learning. Besides, the estimation of a normal from a filled accumulator (step 3) is now the application of a learned function that directly yields two coordinates representing the estimated direction. It significantly reduces the discretization effect and improves normal selection while remaining fast.

**Our contributions** are as follows:

- We show how to reliably map a Hough accumulator for normal estimation into an image that can be used as input of a CNN.
- We demonstrate that a CNN can learn how to estimate a normal from such an image-accumulator.
- We define an efficient way to take point density variation into account (which could actually be used in [BM12] as well).
- We show how the sensitivity to the size of the neighborhood can be addressed in our framework.
- We provide experiments showing that our method outperforms most of the time the state-of-the-art of normal estimation.

## 4. An accumulator for CNN input

**Accumulator design.** The form of a Hough accumulator is well known to have a strong impact on the efficiency and quality of shape extraction. The more adapted to the shape (with little bias when voting), the better. To this end, [BM12] exploit a spherical accumulator, proposed earlier for plane extraction [BELN11].

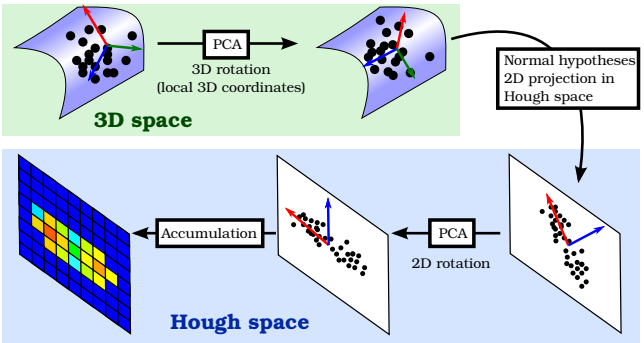



Figure 3: Rotations to ensure stability of the accumulator pattern.

In our case, we want the accumulator to be mapped to the input of a CNN. We chose a simple square image-accumulator: a 2D regular grid of size $M = m \times m$. Given a normal $\mathbf{n} = (n_x, n_y, n_z)$, the coordinates of the vote $x, y$ in the accumulator are given by:

$$(x,y) = \left(\frac{n_x+1}{2} * m, \frac{n_y+1}{2} * m\right) \tag{5}$$

Note that, when back-projected on the sphere, the sizes of the bins are not similar, as illustrated on Figure 2. This accumulator design thus leads to distortions that could affect vote count and bin selection. However, correcting the contribution of each bin is just a constant factor, that the network can easily learn. (We checked that reweighting votes explicitly does not lead to significant changes.) With this simple scheme, the image-accumulator can be filled very efficiently, not even requiring trigonometric computations.

In our experiments, the size of this image-accumulator is set $M = 33 \times 33 = 1089$ bins. However, given it is filled as if projected from an accumulator sphere, only a circular area is used, i.e., roughly $1089 \times \pi/4$ bins. This is 5 times more than [BM12], where $M = 171$. As a result, the bin discretization effect is greatly reduced with our approach. Besides, no accumulator rotation or shift is required.

**Accumulator normalization.** To reduce pattern variations and facilitate learning, we normalize the image-accumulator, as pictured on Figure 3. The 3D coordinate system is first rotated according to a Principal Component Analysis (PCA) of $\mathcal{N}_p$, the points in the neighborhood of $p$: the rotation aligns the $z$-axis on the smallest eigenvector. To further improve stability, we perform an in-plane

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

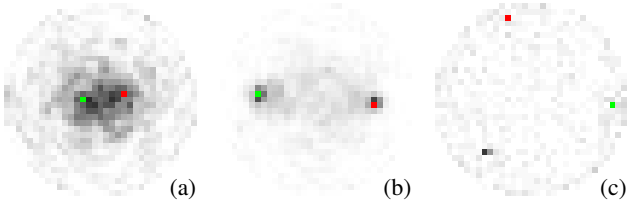



Figure 4: Examples of accumulators. Gray levels reflect the number of votes (negative images for readability: darker for more votes). The red point marks the true normal, the green point, the most voted bin. From left to right: (a) point on a rough but featureless surface, (b) point close to an edge, (c) point close to a 3-plane corner.

rotation after 3D points are projected onto the accumulator plane: using a second, 2D PCA, we align the largest eigenvector along the $x$-axis. A similar effect could be achieved by directly performing a single 3D rotation aligning the second largest 3D eigenvalue along the $x$-axis. However, it is not equivalent because of the projection from a point on the sphere to a plane, and it is less stable than doing it after projection on the accumulator plane. Examples of projected and rotated accumulators are shown in Figure 4.

These uses of PCA do not always guarantee consistent accumulator normalization. Still, when the PCA-induced rotations are not stable, their instability is irrelevant or potentially manageable by the network. Indeed, when on a smooth but possibly noisy surface, the mass of votes focuses around one main 3D direction. This results in a centered blob after 3D PCA and 2D projection (cf. Figure 4a). Its orientation after 2D PCA and rotation can be unstable, but it is little relevant because what really matters is the presence of a peak towards the center of the image, which a network can easily learn in all orientations. When the point is close to an edge, the votes focus on one arc of the sphere of directions. The main axis of the 3D PCA aligns with the arc center and the main axis of the 2D PCA aligns with the arc spanning (cf. Figure 4b). Last, when the point is close to a corner, there are as many focalization directions as main featureless surfaces supporting the corner (cf. Figure 4c). In this case, the main axis of the 3D PCA aligns more or less with the general direction of the corner, but the 2D PCA can be unstable. If there is a prominent surface, the eigenvalues of the 2D PCA might be different enough to rotate its normal near the $x$-axis. If there is no such prominent surface, the rotation has little meaning. It is the most difficult situation the CNN has to learn. However, the non-locality and nonlinearity of the network have the potential to cope with the variety of situations, learning enough elements of information to generalize well, just as a CNN for detection can cope with occluded objects. We also leverage on the ability to generate a large amount of training data for various configurations.

## 5. A CNN for normal estimation

**CNN architecture.** Our task does not require a fancy network architecture. We chose a small-sized network for a low processing time. It is based on LeNet [LBD*89], a simple network yet proven to be adaptable to various estimation problems in image processing. It is illustrated on Figure 5. It is composed of four convolutional layers, two max poolings and four fully connected layers. Most of the parameters are in the fully-connected (FC) layers.

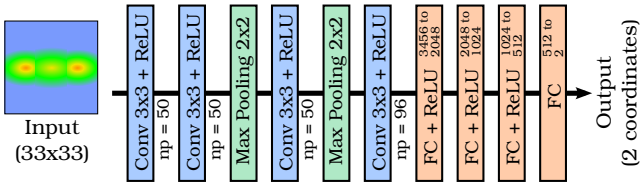



Figure 5: Our CNN architecture for normal estimation.

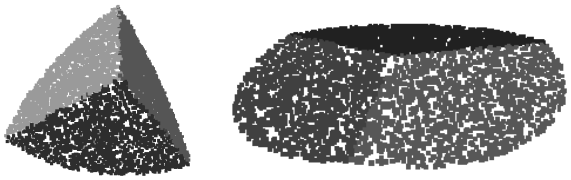


Figure 6: Illustration of training data (min and max angles).

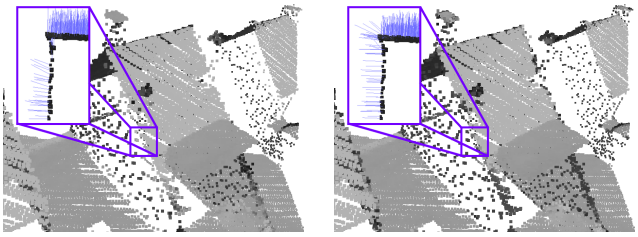
The convolutional nature of this network contributes to the capacity of handling noise by potentially smoothing out accidental peaks. Nonlinearity, provided by max poolings and ReLUs, gives rise to the ability to choose between different peaks based on their local shape. And the global choice among possible normals originates from a non purely local analysis of the direction map given both by the max poolings and the fully-connected layers. We do not pretend it is the best architecture for this task, yet that it is meaningful. (We also experimented with an architecture made from fully-connected layers; it does not perform as well, cf. Section 8.) We train this network using mean square error ($\ell_2$ penalization):

$$\{\hat{\mathbf{n}}_p\} = \arg\min_{\{\mathbf{n}_p\}} \sum_{p\in\mathcal{P}} (\mathbf{n}_p - \mathbf{n}_p^*)^2 \tag{6}$$

**Training data.** To train this network, we generate synthetic ground-truth examples. We create uniformly sampled point clouds over corners with different angles. Angles are uniformly drawn between 80° and 160°. Examples of such point clouds are shown on Figure 6. We generate point sets with 5000 points and randomly pick 1000 points in each set, for which we compute the corresponding accumulator. The training data contain 100,000 such filled accumulators. These learning samples (corners with varying angles) represent the most common situations of sharp features in real data. They are used to learn proper decisions near both edges and corners. We then rely on the ability of the network to generalize and treat partial data, as is the case for occlusion in object detection. It allows the network to treat more complex situations than just 3-side corners. Regression for noisy data, which is the general case, is learned from points far enough from the edges and the corner.

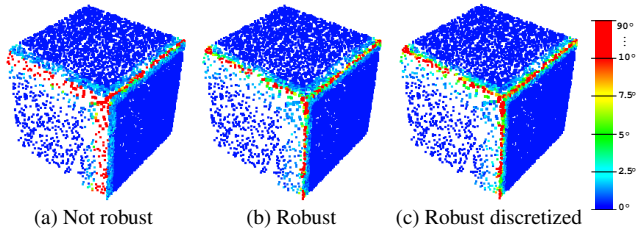
To be more realistic and provide robustness, we also add noise to the training data. In our experiments, we use a Gaussian noise for learning the network. However, it is not intrinsic to the method, as in [LSK*10]; other noise models could have been used too. We also use a Gaussian noise for testing on synthetic data, as [LSK*10, BM12, LZC*15]. However, little bias is to be expected because the noise level varies for each training sample, with a standard deviation randomly drawn in $[0\%, 200\%]$ of the mean distance between the points in the cloud, while the noise level is fixed for evaluation. For tests on real data, the noise is as present in the data.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) Not robust (b) Robust discretized

Figure 7: Detail of a DFC 2015 aerial lidar tile with normal estimation. Roofs are more densely sampled than facade walls. The cross-section illustrates the handling (or not) of density variation.




(a) Not robust (b) Robust (c) Robust discretized

Figure 8: Different sampling density on each face, and method variants with different levels of robustness to this density variation.

**Training process.** For each point $p$ in the training set, we consider a fixed neighborhood size ($K = 100$ neighbors in our experiments) in which we sample triplets of points, filling $p$'s accumulator accordingly (cf. Section 4). It creates a gray-level image, in which we scale the pixel values so that the pixel of the most voted bin is white (highest intensity). As the corresponding ground-truth normal at $p$ is known, it provides input-output examples to train the network. When learning, we randomly choose 75% of these data for actual training. The rest is used to check that learning does not overfit.

## 6. Dealing with density variation

Density variation, with possible anisotropic bias, is a common phenomenon in real-world point clouds. For example, a lidar acquires data from a single viewpoint, typically with regular angular steps for azimuth and elevation, sampling more densely surfaces with low incidence. Figure 7 illustrates this situation with a detail of an aerial acquisition of the Data Fusion Contest (DFC) 2015 [DFC15]. Another, synthetic example is shown on Figure 8. Points in the less dense regions and next to edges with denser regions are wrongly given the normal of the other side of the edges (cf. Fig. 7a and 8a).

Robustness to density variation can be efficiently obtained at plane hypothesis generation time, in the Hough transform. For this, we associate a different weight to each point depending on the local density. We then pick triplets from points having a probability proportional to these weights. A point in a sparse area will be given a higher weight; it will thus be picked more often than a point in a denser region. The weight corresponds to a kind of local (surfacic) scale. We use as local scale the square distance of the considered point to its $k^{\text{th}}_{\text{dens}}$ nearest neighbor. This square distance represents the influence area of the point on the underlying surface. (In our experiment, we use $k_{\text{dens}} = 5$.) This local scale normalization compensates for the low density, as can be seen on Figure 8b.

| Variant of CNN 3s w.r.t. density variation | Not robust | Robust | Robust discretized |
|---|---|---|---|
| Running time (s) | 51.5 | 57.3 | 52.2 |

Table 1: Running time on a DFC 2015 tile detail (185k points).

However, picking random neighboring points according to this local scale is quite slow. It requires computing an array of the cumulative sums of local scales, sorting it, and given a random number, searching the corresponding point in the array. To overcome this issue, we discretize the search space: we compute the min and max local scales and divide this range into $k_s$ equal intervals. (In our experiments, $k_s = 5$ performs well.) We then compute the score of each interval (the sum of all local scales of points in the segment), and randomly pick a segment according to this score. Last, we pick a point in this interval with a uniform probability. Robust discretized normal estimation with this optimization is illustrated on Figures 7b and 8c. The quality is almost as good as with the non-discretized version (RMS error 5.5° vs 5.6° in Figures 8b-8c), while being significantly faster, as can be seen on Table 1.

This way to deal with non-uniform densities is much more efficient than what was proposed in [BM12] where, to sample a 3D point, a ball is first uniformly sampled in which the point is then sampled. While it provides good robustness to density variation, it considerably slows down normal estimation. Note that our way of handling density variation could also be used to speed up [BM12].

## 7. Multiscale approach

Many normal estimation methods rely on a scale parameter. It usually corresponds to the size of the neighborhood to consider. It can be interpreted as the scale at which the scene should be observed, or the distance under which regularization is allowed. However it is often difficult to tune this parameter, in particular for points clouds with high density variation where different neighborhood sizes would be necessary for a robust and accurate estimation. A solution is to use non-parametric methods such as proposed by Dey and Goswami [DG04], but accuracy drops when noise is high.

To improve robustness near sharp edges, we propose a simple variant of our method using a multiscale approach. The fact is the input of the CNN can be easily modified to create a multicanal tensor input, like RGB channels for processing color images. Here our channels are the accumulators computed for different neighborhood sizes, which the PCA-based normalization can roughly align for consistency. In our experiments, we explore two multiscale approaches, with 3 and 5 scales. Given a neighborhood size of $K$ neighbors, the neighborhood sizes are $K/2$, $K$ and $2K$ for the 3-scale scheme and $K/4$, $K/2$, $K$, $2K$ and $4K$ for the 5-scale scheme.

Note that with a monoscale Hough accumulator, as with the one in [BM12], a sample of 3D points votes for one direction regardless of its location, whether it is close or far from the point considered for normal estimation. But using simultaneously different scales provides a form of distance sensitivity: a 3D point sample may contribute to a direction at a given scale, but not at another scale because the corresponding neighborhood size is smaller. As illustrated in the experiment section, this distance sensitivity seems to be enough (w.r.t. a location sensitivity) to reach a high accuracy.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

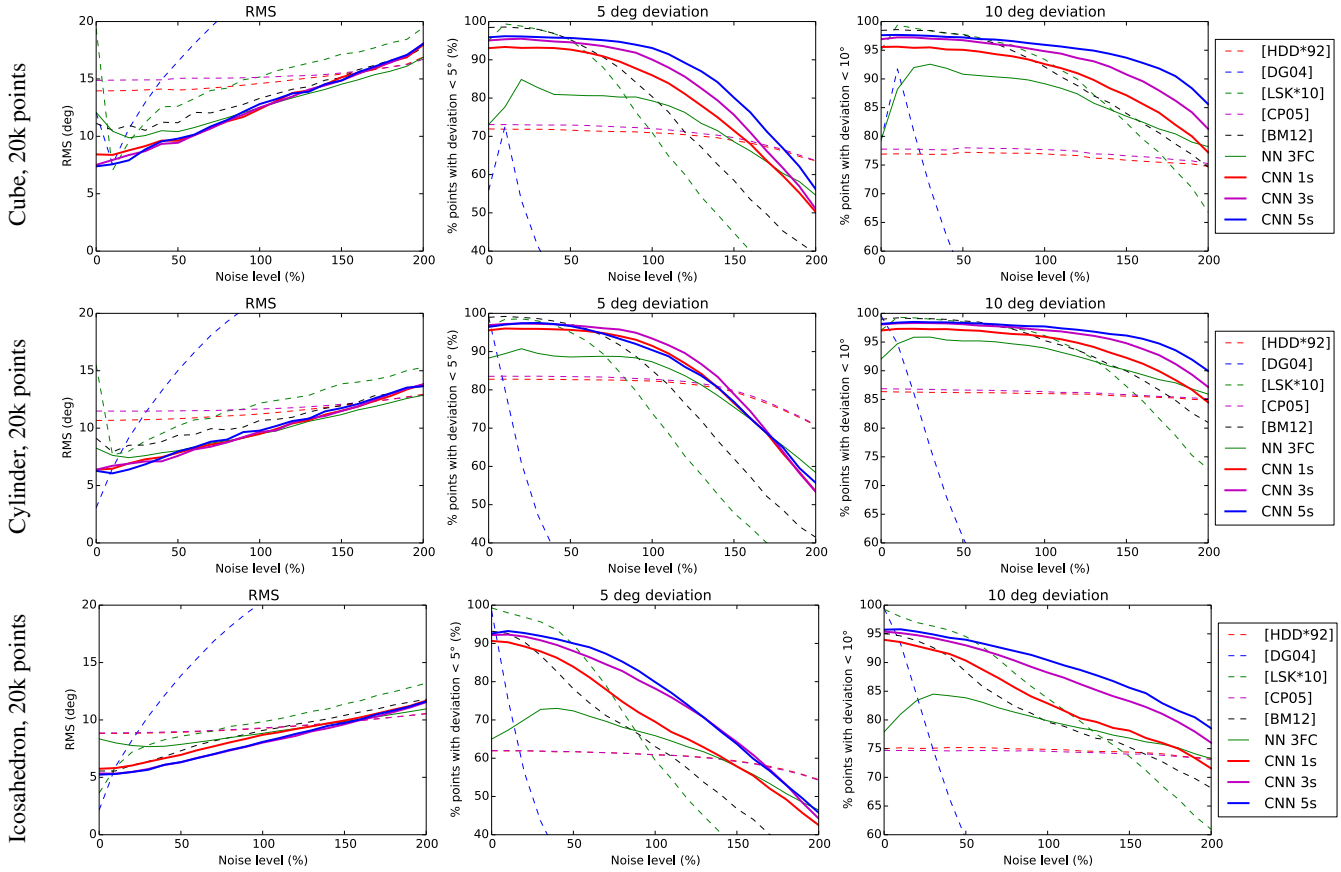



Figure 9: Comparison of various methods on simple geometric models with varying Gaussian noise (standard deviation expressed a % of the mean distance between points) and no density variation. NN 3FC: network with 3 fully-connected layers on the same Hough image-accumulator as ours; CNN $n$s: our method with $n$ scales.

## 8. Evaluation

Our method has 4 main parameters, which are set as follows:

- the accumulator size $M = 33 \times 33$,
- the number of hypothesis to pick $T = 1000$,
- the neighborhood size $K = |\mathcal{N}_p| = 100$ (for training & testing),
- the neighborhood size for estimating a local scale $k_{\text{dens}} = 5$.

$T = 1000$ corresponds to an allowed deviation from the theoretical accumulator distribution of $\varepsilon = 0.073$ for $\alpha = 0.95$. $K$ is the only practical parameter. For comparison purposes, we set $K = 100$ for all methods, except [DG04] which does not take any neighborhood size as parameter. For a fair comparison, tests with [BM12] use exactly the same parameters as ours (i.e., $T$, $K$). For multiscale CNN, we use: for 3 scales (CNN 3s), $K$=50, 100, 200, and for 5 scales (CNN 5s), $K$=32, 64, 128, 256, 512.

We consider two different scores for quantitative evaluation: the root mean square (RMS) deviation and the number of points for which the deviation is less than a given angle. The RMS is a standard error measure. It provides a good idea of the overall performance of an algorithm. It is defined by:

$$RMS = \sqrt{\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} (\widehat{\hat{\mathbf{n}}_p \, \mathbf{n}_p^*})^2} \qquad (7)$$

However, this measure does not favor sharp behaviors. Indeed, smoothing the normals of points close to an edge results in a smaller RMS than choosing the normal of the wrong side of the edge. A less compromising error measure, better suited w.r.t. "visual" applications such as rendering, is to count the proportion of good points (PGP), i.e., whose error is under a given threshold; as in [LZC*15], we study 5° and 10° maximum deviation.

**Experiments on synthetic data.** Figure 9 shows the impact of an increasing Gaussian noise on RMS and PGP for various simple geometric models. We tested our method and its variants against five methods from the literature: [DG04], [LSK*10], [BM12], [HDD*92] and [CP05]. All these methods are available on the Internet, or the code were granted to us by the authors. We also added as baseline a simpler neural network made of 3 fully connected layers with interleaved ReLUs (NN 3FC). For very low levels of noise, the best method is [DG04], but it rapidly degrades as noise increases. The regression methods [HDD*92, CP05] perform better at high noise, when the surface details are lost in the noise and very difficult to retrieve. Between those two extreme cases, our multiscale approaches perform best. Larger neighborhoods provide better robustness for high noise, while small scales give maintain good results for low noise. The comparison to the NN 3FC baseline

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

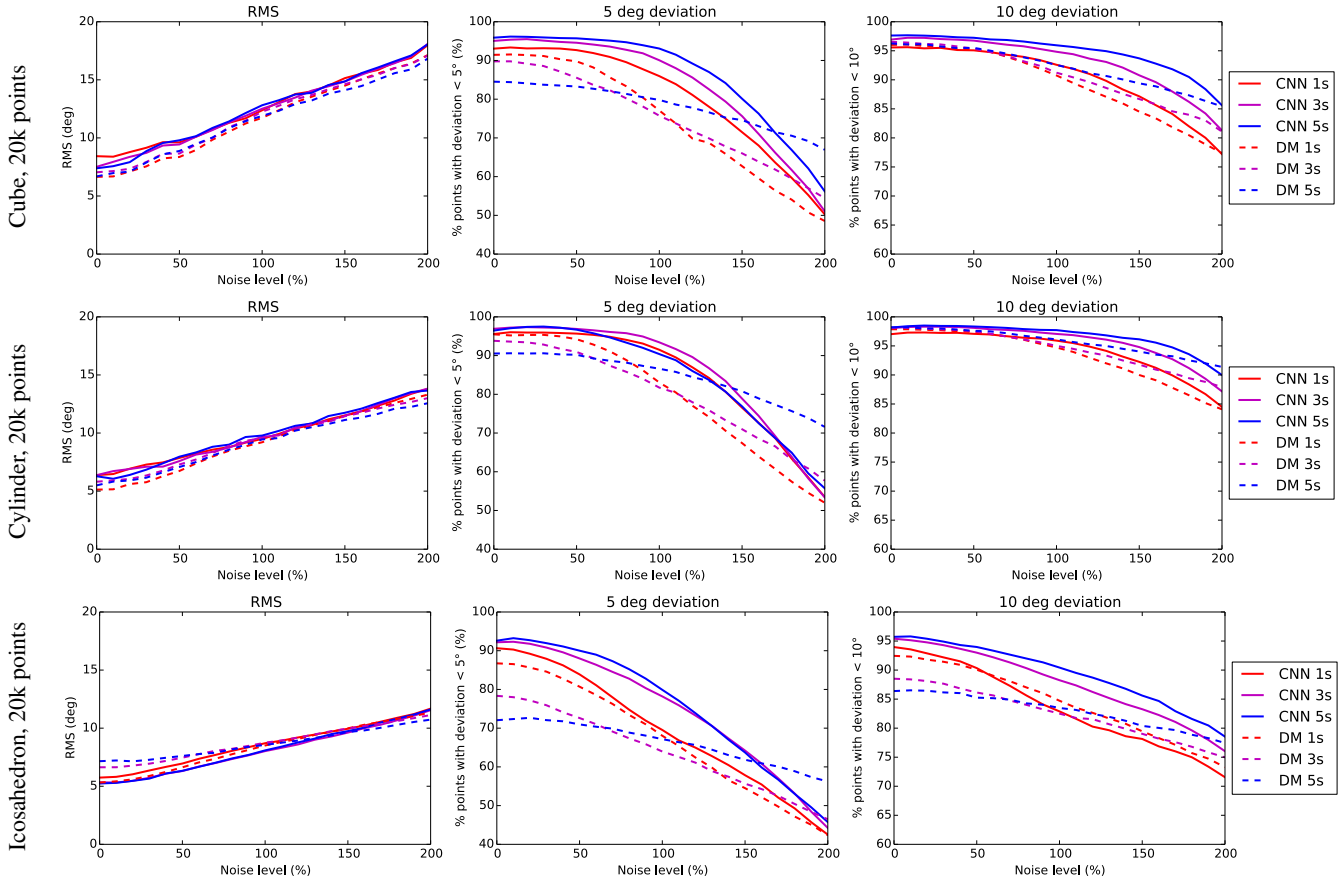



Figure 10: Comparison of our method (CNN) and a similar CNN estimation based on depth map input (DM) with multiscale (*ns*).

shows that our good results cannot be attributed to the Hough representation only. Whereas a simple neural network regressor tends to produce smoother predictions (closer to the planar regression), our CNN approach is more discriminative.

To evaluate the gains of the Hough transform for normal estimation using a CNN, we implemented another baseline method. Once the point neighborhood has been oriented via 3D PCA, we compute a depth map. The depth direction is the axis of the smallest eigenvalue. This depth map has the same dimension as the accumulator in our method. We build a corresponding learning set as for our method, and train the same network architecture. Figure 10 shows the comparison between our Hough-based method and this depth-map-based baseline. Our method performs significantly better regarding PGP, while having a slightly higher RMS. The depth map is not as regular as the Hough accumulator for learning.

We could not compare to [ZCL*13] and [LZC*15] as their code is not available. Still, we experimented on a 100k-point octahedron (see Figure 11), which is one of the synthetic model these authors used for validation. With 50% noise, they obtain slightly better results than ours (please refer to [LZC*15]), but at high computational cost; their parallel version is still more than twice slower than our implementation. Moreover, our method degrades better for high noise, not requiring to tune parameters. Compared to other baseline methods, we estimate sharp features better than using regression [HDD*92] and we are more robust to noise than sample consensus [LSK*10] and ordinary Hough transform [BM12].

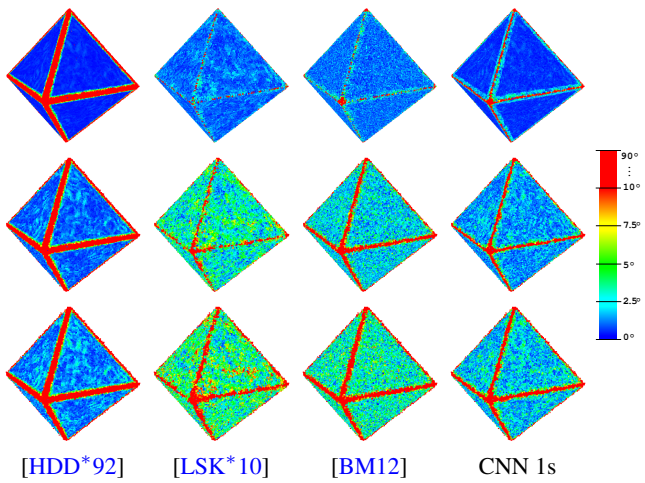



Figure 11: Visual results of four estimation algorithms on an octahedron (100k points) with 50% noise (top line), 150% noise (middle) and 200% noise (bottom). Color scale, given on the right, maps a deviation angle to a color (red is a deviation greater than 10°).

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

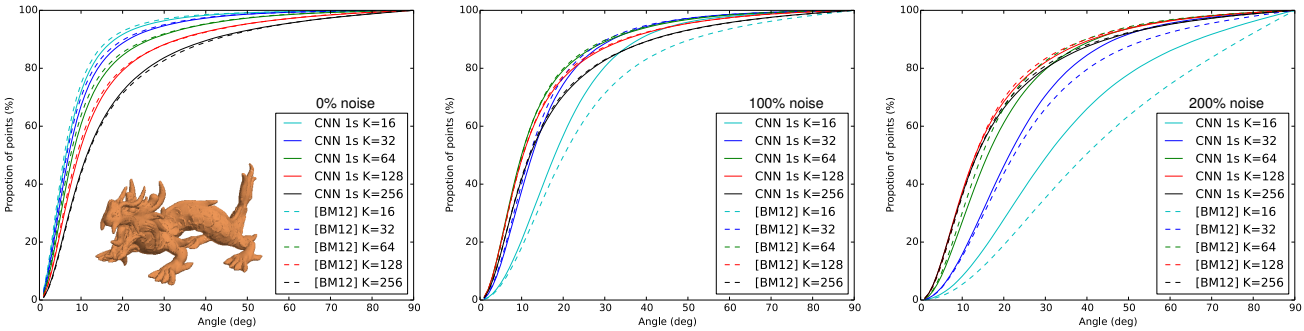



Figure 12: Proportion of normals with error less than a given angle, on 250k-point dragon with noise 0% (left), 100% (middle), 200% (right).

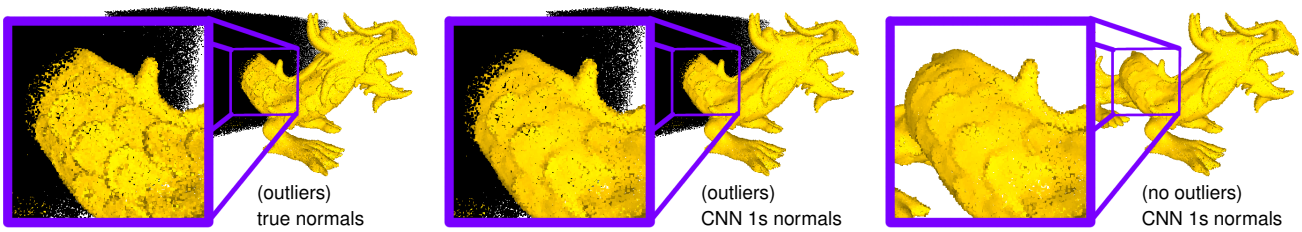



Figure 13: Robustness to outliers on 250k-point dragon with 100% noise and 1M-point outliers added in bounding box.

**Experiments on real data.** We first consider the scan data of a dragon sculpture, for which an accurate mesh is available (3.6M vertices, 7.2M triangles). The dragon features sharp attributes such as fangs, horns and scales. We randomly draw 250k points on mesh faces; these faces determine reference normals. We use this subsampled point cloud to compare with [BM12] and to study the sensitivity to the neighborhood size $K$, for different levels of noise. Unsurprisingly, when no noise is added (besides the random picking of points on the mesh), a small neighborhood provides a better sensitivity to sharp features. In this case, we only perform slightly better than [BM12]. But when noise increases, information level drops in small neighborhoods and larger ones provide a better robustness. In this more complicated setting, we perform significantly better than [BM12]. To evaluate robustness to outliers, we draw 1M random points in the dragon bounding box. With 100% noise, RMS error is 21.1°, vs 20.5° with no outliers (see Figure 13).

We then consider a laser scan of an office room. The point cloud naturally features edges, corners, as well as density variations with anisotropic bias. An exact ground truth is not known, but can be approximated from the implicit mesh structure of the depth map: we consider as reference at each point the mean normal of the surrounding faces. Although theses normals can be noisy in very dense areas, they are enough for algorithm comparison. Given this pseudo ground truth, Figure 14 shows the proportion of estimated normals with angular error below a given threshold. Due to the small number of points near edges, compared to points on wide planar areas, the difference between the methods is small on average. However, normals can be locally wrong, as can be seen on Figure 15, which illustrates a detail of the scene with density variations.

Finally, we show qualitative result on outdoor scenes. Figures 7 displays a detail of the DFC 2015 aerial lidar tile in Figure 16, with robustness to density variations. Figure 17 illustrates shading with normals estimated on a sparse structure-from-motion point cloud.

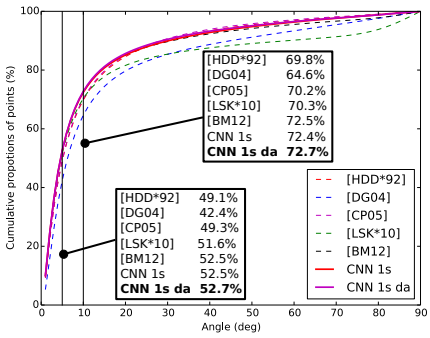



Figure 14: Office room, proportion of normals with error less than a given angle. CNN 1s da: our robust density-adaptive variant.

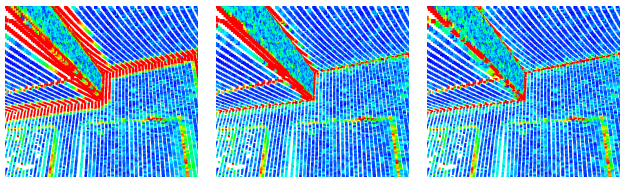
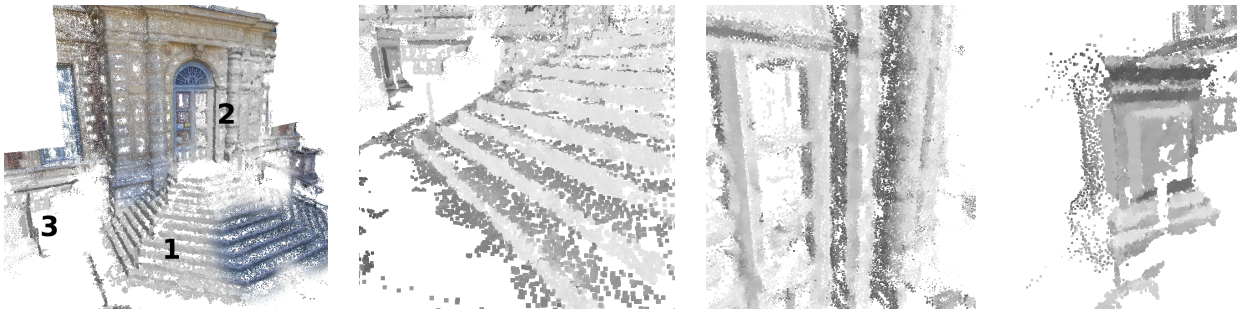


Figure 15: Office room detail. From left to right: planar regression, our plain method, and our robust density-adaptive method.

Figure 16: DFC 2015 lidar tile with normals (decimated 2.3M).

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.




Point cloud (textured) Detail 1 (untextured, shaded) Detail 2 (untextured, shaded) Detail 3 (untextured, shaded)

Figure 17: Our CNN-based normal estimation on a structure-from-motion point cloud of the Château de Sceaux (France), 400k points.

| Model | Cube | Armadillo | DFC detail | Omotondo | DFC tile |
|---|---|---|---|---|---|
| Size | 20k | 173k | 185k | 997k | 2.3M |
| [HDD*92] | 0.3 | 2.1 | 1.9 | 12 | 25 |
| [DG04] | 3.2 | 55 | 41 | 441 | 1243 |
| [CP05] | 5.8 | 50 | 54 | 304 | 711 |
| [BM12] | 1.9 | 13 | 11 | 44 | 147 |
| [LSK*10] | 8.8 | 64 | 75 | 392 | 902 |
| CNN 1s | 4.5 | 33 | 34 | 183 | 423 |
| CNN 3s | 5.9 | 48 | 52 | 273 | 639 |
| CNN 5s | 7.9 | 69 | 73 | 382 | 897 |

Table 2: Computation times (in seconds) for different models and different methods. CNN variants are without density-adaptivity.

**Computation times** are given in Table 2. We tested a cube with 50% noise and real point clouds: Armadillo, Omotondo, DFC detail of Figure 7, whole DFC tile of Figure 16. Our running times are competitive w.r.t. compared methods, except [HDD*92] which is much faster. As we share the first step of [BM12] (cf. Section 3) and as accumulator filling is fast, the difference resides mainly in the CNN computations. Using more scales increases computation time, due to the more expensive search for neighborhoods larger than $K = 100$ (up to $K = 512$), despite the use of a kd-tree.

**Limitations.** Contrary to other approaches where it is inexpensive to change the value of a parameter, we have to retrain the network if we need to adapt specifically to the input data. It mainly concerns the neighborhood size $K$, which controls the sensitivity to details. However, the multiscale approach reduces the influence of this parameter by analyzing different scales simultaneously.

## 9. Conclusion

We have proposed a novel method for normal estimation in unorganized point clouds using a convolutional neural network. Although we reused the idea of the Hough transform of [BM12] as well as its robust and efficient sampling strategy, we introduced a whole range of new features. We use a different accumulator, which is planar rather than spherical and which is less discretized. Moreover, we define a totally different, CNN-based decision procedure to select a normal from the accumulator. Besides, to deal with density variation, we introduce a fast approach to pick points according to a distribution based on a local density estimation. Finally, to improve robustness and reduce parameter tuning, we present a multiscale approach which automatically adapts to different sizes of neighborhoods, requiring basically no change in the CNN framework. As a result, we are more robust and more accurate most of the time on both synthetic and real data, although just a few times slower. We actually most often also outperform other state-of-the-art methods, even [ZCL*13, LZC*15] for high noise which anyway are slower.

Perspectives include the study of geometric transformations in Hough space to facilitate the learning and improve accuracy. The CNN architecture and training data can certainly also be improved, as the space of possibilities is quite large. Actually, future advances in research on CNNs should also benefit to this framework.

This method participates to a new trend in geometry processing where geometric decisions are learnt from ground-truth data, possibly biased towards a specific kind of scenes, rather than the result of explicit, manually-designed geometric computations.

**Implementation details.** The Hough transform is coded with Eigen (`eigen.tuxfamily.org`). Neighbor search in a point cloud uses nanoflann kd-tree (`https://github.com/jlblancoc/nanoflann`). Our CNN framework relies on Torch-nn (`https://github.com/torch/nn`). For experiments, we used a laptop with Intel i7 quad core and GPU NVidia GTX970m.

**Acknowledgements.** We would like to thank all the authors of the different papers for providing their code or executable. Armadillo and Omotondo come from the Aim@Shape repository. Asian Dragon comes from the Stanford 3D scanning repository.

## References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Tr. on Visualization and Computer Graphics 9*, 1 (2003), 3–15. 1

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *SGP* (2007), pp. 39–48. 1

[ASGCO10] AVRON H., SHARF A., GREIF C., COHEN-OR D.: L1-sparse reconstruction of sharp point set surfaces. *TOG) 29*, 5 (2010), 135:1–135:12. 1

[Bal81] BALLARD D. H.: Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition 13*, 2 (1981), 111–122. 2

[BDLGM14] BOULCH A., DE LA GORCE M., MARLET R.: Piecewise-planar 3D reconstruction with edge and corner regularization. *CGF 33*, 5 (2014), 55–64. 1

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

[BELN11] BORRMANN D., ELSEBERG J., LINGEMANN K., NÜCHTER A.: The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research 2*, 2 (2011). 2, 3

[BM12] BOULCH A., MARLET R.: Fast and robust normal estimation for point clouds with sharp features. *CGF 31*, 5 (2012), 1765–1774. 1, 2, 3, 4, 5, 6, 7, 8, 9

[BRG16] BANSAL A., RUSSELL B., GUPTA A.: Marr revisited: 2D-3D alignment via surface normal prediction. In *CVPR* (2016). 2

[CLP10] CHAUVE A.-L., LABATUT P., PONS J.-P.: Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR* (2010), pp. 1261–1268. 1

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (2005), 121–146. 1, 6, 9

[Dav88] DAVIES E. R.: Application of the generalised Hough transform to corner detection. *Computers and Digital Techniques, IEE Proceedings E 135*, 1 (1988), 49–54. 2

[DFC15] IEEE GRSS Data Fusion Contest, 2015. URL: http://www.grss-ieee.org/community/technical-committees/data-fusion. 5

[DG04] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *SoCG* (2004), pp. 330–339. 1, 5, 6, 9

[DH72] DUDA R. O., HART P. E.: Use of the Hough transformation to detect lines and curves in pictures. *Comm. ACM 15*, 1 (1972), 11–15. 2

[FGMP14] FERRI F., GIANNI M., MENNA M., PIRRI F.: Point cloud segmentation and 3D path planning for tracked vehicles in cluttered and dynamic environments. In *3rd IROS Workshop on Robots in Clutter: Perception and Interaction in Clutter* (2014). 1

[GDDM14] GIRSHICK R., DONAHUE J., DARRELL T., MALIK J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR* (2014), pp. 580–587. 2

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. *TOG) 26*, 3 (2007), 23. 1

[Gra15] GRAHAM B.: Sparse 3D convolutional neural networks. In *BMVC* (2015). 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *ACM SIGGRAPH Computer Graphics 26*, 2 (1992), 71–78. 1, 6, 7, 9

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *TOG) 28*, 5 (2009), 176. 1

[Hou62] HOUGH P. V. C.: Method and means for recognizing complex patterns. *U.S. Patent 3.069.654* (1962). 2

[HSK*12] HINTON G. E., SRIVASTAVA N., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R. R.: Improving neural networks by preventing co-adaptation of feature detectors. *preprint arXiv:1207.0580* (2012). 2

[IK88] ILLINGWORTH J., KITTLER J.: A survey of the Hough transform. *CVGIP 44*, 1 (1988), 87–116. 2

[JIS03] JEONG W. K., IVRISSIMTZIS I. P., SEIDEL H. P.: Neural meshes: statistical learning based on normals. In *Pacific Conference on Computer Graphics & Applications (CGA)* (2003), pp. 404–408. 2

[KEB91] KIRYATI N., ELDAR Y., BRUCKSTEIN A. M.: A probabilistic Hough transform. *Pattern Recognition 24*, 4 (1991), 303–316. 2

[KPW*10] KNOPP J., PRASAD M., WILLEMS G., TIMOFTE R., VAN GOOL L.: Hough transform and 3D SURF for robust three dimensional classification. In *ECCV* (2010), pp. 589–602. 2

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *NIPS* (2012). 2

[LBD*89] LECUN Y., BOSER B., DENKER J. S., HENDERSON D., HOWARD R. E., HUBBARD W., JACKEL L. D.: Backpropagation applied to handwritten zip code recognition. *Neural computation 1*, 4 (1989), 541–551. 2, 4

[LSD*15a] LI B., SHEN C., DAI Y., VAN DEN HENGEL A., HE M.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *CVPR* (2015), pp. 1119–1127. 2

[LSD15b] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. In *CVPR* (2015). 2

[LSK*10] LI B., SCHNABEL R., KLEIN R., CHENG Z., DANG G., JIN S.: Robust normal estimation for point clouds with sharp features. *Computers & Graphics 34*, 2 (2010), 94–106. 1, 4, 6, 7, 9

[LZC*15] LIU X., ZHANG J., CAO J., LI B., LIU L.: Quality point cloud normal estimation by guided least squares representation. *Computers & Graphics 51*, C (2015), 106–116. 1, 4, 6, 7, 9

[MNG04] MITRA N. J., NGUYEN A., GUIBAS L.: Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry & Applications 14*, 04n05 (2004), 261–276. 1

[ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M. H.: Feature preserving point set surfaces based on non-linear kernel regression. *CGF 28*, 2 (2009), 493–501. 1

[PCFG12] PENEDONES H., COLLOBERT R., FLEURET F., GRANGIER D.: *Improving Object Classification using Pose Information*. Research report Idiap-RR-30-2012, Idiap Research Institute, 2012. 2

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. *TOG) 22*, 3 (2003), 641–650. 1

[PWP*11] PHAM M.-T., WOODFORD O. J., PERBET F., MAKI A., STENGER B., CIPOLLA R.: A new distance for scale-invariant 3D shape recognition and registration. In *ICCV* (2011), pp. 145–152. 2

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH* (New York, NY, USA, 2000), pp. 343–352. 1

[SEZ*14] SERMANET P., EIGEN D., ZHANG X., MATHIEU M., FERGUS R., LECUN Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR* (2014). 2

[SLJ*14] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. In *CVPR* (2014). 2

[SSW15] SUN Y., SCHAEFER S., WANG W.: Denoising point sets via l0 minimization. *CAGD 35-36* (2015), 2–15. 1

[SW02] SHEN F., WANG H.: Corner detection based on modified Hough transform. *Pattern Recognition Letters 23*, 8 (2002), 1039 – 1049. 2

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient RANSAC for point-cloud shape detection. *CGF 26*, 2 (2007), 214–226. 1

[TM78] TSUJI S., MATSUMOTO F.: Detection of ellipses by a modified Hough transformation. *IEEE Trans. on Computers 27*, 8 (1978). 2

[WFG15] WANG X., FOUHEY D. F., GUPTA A.: Designing deep networks for surface normal estimation. In *CVPR* (2015), pp. 539–547. 2

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3D ShapeNets: A deep representation for volumetric shape modeling. In *CVPR* (2015). 2

[XO93] XU L., OJA E.: Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities. *CVGIP: Image understanding 57*, 2 (1993), 131–154. 2

[ZCL*13] ZHANG J., CAO J., LIU X., WANG J., LIU J., SHI X.: Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics 37*, 6 (2013), 697 – 706. 1, 7, 9

[ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *ECCV* (2014), pp. 818–833. 2

[ZPVBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *SIGGRAPH* (2001), ACM, pp. 371–378. 1

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.