Robust procedural model fitting with a new geometric similarity estimator

Zongliang Zhang[a], Jonathan Li[a,b,*], Yulan Guo[c], Xin Li[d], Yangbin Lin[e], Guobao Xiao[a], Cheng Wang[a]

[a] Fujian Key Laboratory of Sensing and Computing for Smart Cities, Xiamen Key Laboratory of Geospatial Sensing and Computing, School of Information Science and Engineering, Xiamen University, Xiamen, Fujian 361005, China
[b] Departments of Geography and Environmental Management and Systems Design Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
[c] College of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan 410073, China
[d] School of Electrical Engineering and Computer Science, Louisiana State University, Baton Rouge, LA 70808, USA
[e] College of Computer Engineering, Jimei University, Xiamen, Fujian 361000, China

ARTICLE INFO

ABSTRACT

Procedural model fitting (PMF) is a generalization of classical model fitting and has numerous applications for computer vision and computer graphics. The task of PMF is to search a geometric model set for the model that is most similar to a set of data points. We propose a strict and robust similarity estimator for PMF to handle imperfect data. The proposed estimator is based on the error from model to data, while most other estimators are based on the error from data to model. We then use the proposed estimator to guide the cuckoo search algorithm to search for the most similar model. To accelerate the search process, we also propose a coarse-to-fine model dividing strategy to early reject dissimilar models. In this paper, the proposed PMF method is applied to fit building models on laser scanning data. It is also applied to fit character models on eighteen variants of imperfect MNIST data to achieve few-shot pattern recognition. In the 5-shot recognition, our method outperforms the state-of-the-art method on thirteen variants of the imperfect data. In particular, for one of the data corrupted by grid lines, our method obtains a high accuracy of 65%, whereas the state-of-the-art method only obtains an accuracy of 30%.

1. Introduction

A procedural model set is defined by a probabilistic program consisting of several parametric rules [1,2]. Retrieving desired models from a procedural model set is a newly growing topic called inverse procedural modeling [3] or probabilistic program induction [4]. As a special case of probabilistic program induction, procedural model fitting (PMF) aims to search a procedural geometric model set for the model that is most similar to (i.e., best explains) a given set of data points. PMF has achieved remarkable progress in computer vision and computer graphics as it can extract rich structure information from the data [4,5]. For example, PMF has achieved human-level performance in the one-shot pattern recognition task [4].

However, it is difficult to perform PMF as it has three challenging issues. The first issue is the creation of the probabilistic program, which is manually addressed in some methods [6–9]. Several

methods have also been proposed to automatically create the probabilistic program [4,10–12]. Given the probabilistic program that defines the procedural model set, the second issue is the optimization problem to find the desired model from the model set. This can be addressed by Markov chain Monte Carlo [6,7,13], reinforcement learning [14,15], sequential Monte Carlo [8], active-set [4], cuckoo search [16,17], genetic algorithm [9], or neural parsing [18,19].

Different from the aforementioned two issues that come from the model side, the third issue comes from the data side. That is, PMF needs a geometric similarity estimator to estimate the similarity between model and data to guide the optimization process. Most existing PMF methods pay major attention to address the model-side issues, while tackle the data-side issue using common estimators. Such a commonly used estimator is voxel difference, which is used in PMF methods [4,6,8,9]. Another commonly used estimator is the Error from Data to Model (E**D**M), which is introduced in the well-known least squares method and is still widely used in modern computer science [20,21]. Voxel difference and E**D**M are simple to understand and easy to implement. However,

EDM is sensitive to outliers [22]. Voxel difference is also sensitive to imperfect data. In practice, data usually suffer from imperfection. That is, data are commonly contaminated by gross-outliers [23], pseudo-outliers [24,25], noise, and missing data [26].

In this paper, we propose a novel geometric similarity estimator for PMF to robustly handle imperfect data. The proposed estimator is based on the Error from Model to Data (EMD), with our key insight that EMD is more reliable than EDM if the data are imperfect. As the counterpart of EDM, EMD is as simple as EDM. Although extensive investigations have been conducted in EDM [27], only a few works can be found in EMD. Note that, similar to EDM, using EMD only is insufficient to represent the similarity between a model and data [28]. A regularization term should be used to regularize EMD to ensure that only one model is most similar to the data. The ground-truth model cannot be distinguished from some trivial (null) models by the method proposed in [29], as it uses EMD without regularization [30]. The method proposed in [31] also uses EMD but requires the model to be within a narrow crust, which limits the application of EMD. In contrast, the proposed estimator has very few requirements for the model using a novel regularization approach.

Given the similarity estimator, we use the cuckoo search algorithm [16] to perform optimization for PMF in this paper. With few parameter to tune, the cuckoo search algorithm is a recently popular random optimization algorithm. In general, the optimization algorithm needs to accurately estimate the similarity, which is time-consuming. Observing that the dissimilarity can be determined by sampling only one point from the model, we propose a novel coarse-to-fine model dividing strategy to early reject dissimilar models to accelerate the optimization process.

The contributions of this paper can be summarized as follows. (1) A novel geometric similarity estimator is proposed to strictly and robustly estimate the similarity between a complex geometric model and an imperfect data point set; (2) A novel early rejection strategy is proposed to accelerate the cuckoo search based PMF; (3) Several robust PMF applications are explored to fit cylinders, characters, and buildings.

The rest of this paper is structured as follows. In Section 2, we review the related work. In Section 3, we give the preliminary knowledge for PMF. We then present our similarity estimator, our early rejection strategy, and the experiments in Sections 4–6, respectively. We finally conclude the paper in Section 7.

2. Related work

Most existing robust model fitting methods were proposed to fit classical models. A classical model is usually represented by a single parametric rule. For example, a line, a circle, or a polynomial function can be represented by a single equation. One of the most popular robust methods is RANdom SAmple Consensus (RANSAC) [32]. Assuming that a candidate model can be determined by a subset of the data points, RANSAC finds the desired model from the candidate models using inlier number criterion [33]. A lot of methods have been proposed to improve RANSAC in terms of accuracy [34–36], efficiency [37–39], and global consistency [24,40–42]. Another popular robust method is Hough transform [43,44], which achieves model fitting by implicitly maximizing inlier number through voting in the space of model parameters [45]. Hough transform has been used to extract some classical models such as lines [46–48], circles [49,50], ellipses [51–53], curves [54–56], and planes [57,58].

However, a procedural model is usually more complex than a classical model, as it is usually represented by a number of parametric rules. For example, a procedural building model [59] can contain boundaries and holes, which are rare in a classical model. In practice, a procedural model can consist of different types of

Table 1
An example probabilistic program.

rule Building()	**rule** Facade(i, α)	**rule** Floor(i, β)
Sample $\alpha \sim p_\alpha(\alpha)$	**if** $i < \alpha$	//Generate a rectangle
Facade(0, α)	Sample $\beta \sim p_\beta(\beta)$	//at height i,
end rule	Floor(i, β)	//dig out a square hole
	Facade($i+1, \alpha$)	//with size β
	end if	//from the rectangle.
	end rule	**end rule**

sub-models, while a classical model is normally composed of sub-models with the same type. Furthermore, the relation between model parameters and model points is commonly straightforward in a classical model, while it can be complicated in a procedural model as defined by a black-box probabilistic program. Moreover, the number of model parameters is usually fixed in a classical model, while it can vary in a procedural model set [6]. Therefore, it is unclear how to extend a classical model fitting method to handle procedural models. For example, the aforementioned assumption of RANSAC-like methods are usually not applicable to procedural models. For another example, Hough transform is typically used to handle models with less than 10 parameters [60], while a procedural model can have tens of even thousands of parameters [9].

Nevertheless, the similarity estimator used by a classical model fitting method can be used for PMF. One of the most popular robust estimators is inlier number or its extension M-estimator [33,61]. However, inlier number needs a hard threshold to determine if a point is an inlier or not. Similarly, it is not straightforward to choose an appropriate robust function for M-estimator to handle different types of imperfect data. Moreover, inlier number and M-estimator are suffered from overfitting to data, as they use EDM without regularization. Actually, using EDM only is insufficient to strictly represent the similarity between a model and data [28]. To overcome overfitting, an EDM-based estimator usually has two terms, the EDM term and a regularization term. The regularization term is used to ensure that only one model is most similar to the data. There are two major types of regularization terms: EMD and model smoothness. The frequently used Hausdorff distance [62] is an EDM-based estimator regularized by EMD. Various forms of model smoothness have been proposed [27], including local smoothness [63–65], global smoothness [66,67], piecewise smoothness [68], and volumetric smoothness [69].

3. Preliminaries

A k-dimensional point set P is a subset of $\mathbb{R}^k$, i.e., $P \subset \mathbb{R}^k$. In this paper, $k \in \{1, 2, 3\}$. A geometric model is a special point set consisting of one or more continuous sub-point sets. For example, a character is composed of several strokes. As it is uncountable, a continuous point set is usually represented by a parametric rule. For instance, a 2-dimensional line segment is a continuous point set and can be represented by a rule with parameters $\theta_1 \in \mathbb{R}^2$ and $\theta_2 \in \mathbb{R}^2$: $\{\theta_1 + t\theta_2 | t \in [0, 1]\}$. It is easy to know that, if the parameters of a rule are variable, then the rule defines a set of models.

3.1. Probabilistic program

As a generalization of parametric rule, a probabilistic program is composed of several parametric rules. Actually, a probabilistic program is able to represent any model [70]. This paper focuses on building models [59] and character models [4]. Table 1 shows an example of probabilistic program, which has three rules: Building, Facade, and Floor. Building is the start rule and calls Facade with variable α sampled from the prior $p_\alpha(\cdot)$. α implicitly determines the number of Floors. The prior $p_\alpha(\cdot)$ is uniformly distributed. That
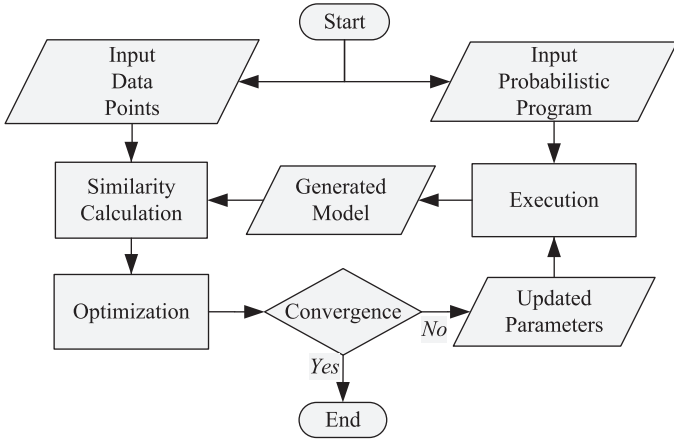
Fig. 1. Our PMF pipeline.

is, $\alpha \in [0, \alpha_{max}] \cap \mathbb{Z}$, where α_{max} is a predefined maximum number of Floors. For simplicity, this paper only investigates uniformly distributed priors, although they play important roles in a probabilistic program.

Facade is a recursive rule. Before calling itself, Facade calls Floor with variable β. Furthermore, β specifies the size of the hole generated by the Floor rule. During the execution of this probabilistic program, multiple instances of Facade may be produced. Therefore, multiple instances of β may exist. For PMF, it is needed to identify different instances of the same variable. For this example probabilistic program, β can be simply identified with the parameter i: β_i. For more complex cases, a calling trace can be used for the identification.

3.2. Procedural model fitting

Given a data point set D and a probabilistic program g, the task of PMF is to search the model set defined by g for the model that is most similar to D. As shown in Fig. 1, our PMF method proceeds as follows. Given the data D and the probabilistic program g with parameter $\boldsymbol{\theta}$, a similarity calculation procedure is used to calculate the geometric similarity between D and the model generated according to g. Based on the calculated similarity, $\boldsymbol{\theta}$ is iteratively updated by the optimization procedure.

From a Bayesian perspective, the PMF optimization problem can be formulated as follows:

$$\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}|D) \propto L(D|\boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{1}$$

where D is the data point set, $f(\cdot|\cdot)$ is the posterior of the parameters given the data, $L(\cdot|\cdot)$ is the likelihood of the data given the parameters, and $p(\cdot)$ is the parameter prior predefined in the input program g. As the prior is assumed to be uniformly distributed, the posterior is reduced to the likelihood. Let $\mathcal{M}_g$ be the model set defined by g, we define the likelihood $L(\cdot|\cdot)$ as:

$$L(D|\boldsymbol{\theta}) = s(M_g^{\boldsymbol{\theta}}, D), \tag{2}$$

where $M_g^{\boldsymbol{\theta}} \in \mathcal{M}_g$ is the model corresponding to $\boldsymbol{\theta}$, and $s(\cdot, \cdot)$ is the geometric similarity between $M_g^{\boldsymbol{\theta}}$ and D (Sections 3.3 and 4). Equation (1) defines a non-convex optimization problem, for which traditional mathematical optimization methods are inapplicable. We use the cuckoo search algorithm [16], which is a random optimization algorithm, to solve Eq. (1) (Sections 3.4 and 5).

3.3. Strict geometric similarity

A geometric similarity estimator is used to estimate the similarity between two point sets. In this paper, an estimator is consid-

ered as a strict estimator if it can ensure that a point set is most similar to itself. Formally, an estimator $s(\cdot, \cdot)$ is strict if it satisfies the following property:

$$s(Q, P) < s(P, P) \quad \forall P \in \mathcal{P}_u \ \forall Q \in \mathcal{P}_u \setminus \{P\}, \tag{3}$$

where $\mathcal{P}_u$ is the universal set of point sets. In the context of model fitting, one of the two point sets involved in similarity estimation is a geometric model. To achieve procedural model fitting, the similarity estimator should be strict. Only with a strict estimator, it can be guaranteed to find the ground-truth model if the data is perfect (i.e., the data point set is the ground-truth model itself).

A well-known strict estimator for $\mathcal{P}_u$ is Hausdorff distance [28,62], which is defined as:

$$d_H(P, Q) = \max\{d(P, Q), d(Q, P)\}, \tag{4}$$

where $P \in \mathcal{P}_u$, $Q \in \mathcal{P}_u$, and $d(P, Q)$ is the error from P to Q:

$$d(P, Q) = \max_{\mathbf{p} \in P} \min_{\mathbf{q} \in Q} \|\mathbf{p} - \mathbf{q}\|, \tag{5}$$

where $\|\cdot\|$ is Euclidean norm. In general, the error from P to Q does not equal the error from Q to P, and using only one of these two errors is insufficient to represent the similarity between P and Q [28]. That is, neither $d(P, Q)$ nor $d(P, Q)$ is a strict estimator.

3.4. Random optimization

To maximize an objective function $f(\theta)$ for $\theta \in [\theta_{min}, \theta_{max}]$, a random optimization algorithm usually works as follows [6]. Let $\theta^{(i)}$ be the value of θ in iteration i. First, θ is randomly initialized as $\theta^{(0)}$. In each iteration, a tentative $\tilde{\theta}$ is sampled from a proposal function $q(\theta|\theta^{(i)})$. If $f(\tilde{\theta}) > f(\theta^{(i)})$, then $\tilde{\theta}$ is accepted (i.e., $\theta^{(i+1)} = \tilde{\theta}$), otherwise $\tilde{\theta}$ is rejected (i.e., $\theta^{(i+1)} = \theta^{(i)}$). The proposal function q plays a critical role in a random optimization algorithm. One of the simplest proposal functions is the uniform function. That is, $\tilde{\theta} \sim [\theta_{min}, \theta_{max}]$.

The cuckoo search algorithm [16] is a random optimization algorithm inspired by the breeding behaviour of cuckoo birds. To avoid the tedious work of offspring breeding, a cuckoo lays its egg to replace the egg in the nest of a host bird with the hope that the host could help breeding the offspring. The host will also lay new egg to replace discovered cuckoo egg. The cuckoo search algorithm iteratively mimics the egg replacement. The egg corresponds to θ, the quality of egg corresponds to the objective function $f(\theta)$. The egg laying of cuckoo and host are modelled as sampling $\tilde{\theta}$ from the Levy function and the uniform function, respectively. In each iteration, the algorithm sequentially performs the egg laying of cuckoo and the host, and the egg replacement happens if the quality of new egg is better than that of old egg. So far, we have described the simplified algorithm with one cuckoo and one host bird. In practice, the algorithm mimics the behaviour of a population of cuckoos and host birds.

4. Proposed similarity estimator

Let M be the model point set and D be the data point set involved in model fitting. As discussed in Section 3.3, Hausdorff distance can be used to estimate strict similarity between M and D. However, it is time-consuming to calculate Hausdorff distance as it requires to calculate both EDM $d(D, M)$ and EMD $d(M, D)$. Consequently, a similarity estimator for model fitting is usually based on either EDM or EMD. Most estimators are based on EDM. Our insight is that EMD is more reliable than EDM if the data is imperfect. As shown in Eq. (5), all the data points are involved in the *max* operator to calculate EDM $d(D, M)$, however, only the data points survived from the *min* operator are involved in the *max* operator to calculate EMD $d(M, D)$. In other words, the outliers in
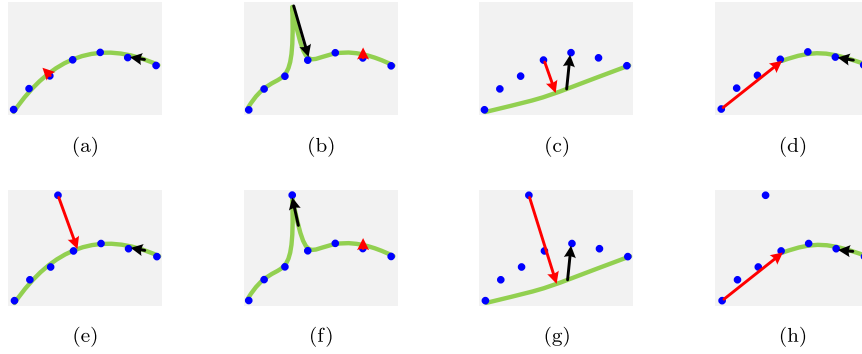
Fig. 2. An illustration of EDM and EMD. Top row: the overlaps between a nearly perfect data point set (blue dots) and different models (green curves). Bottom row: the overlaps between an imperfect data point set (blue dots) and different models (green curves). The imperfect data contains one outlier. From left to right: good-fitting model, over-fitting model, under-fitting model, and incomplete-fitting model. The red arrow denotes EDM, while the black arrow denotes EMD. The values of EDM or EMD are indicated by the lengths of the arrows. A long arrow means a large error, i.e., a small similarity. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(a) C_1 (b) C_2 (c) C_3 (d) $C_1 \cup C_2 \cup C_3$

Fig. 3. An illustration of similarity. (a) Curve C_1, (b) Curve C_2, (c) Curve C_3, and (d) the overlap between C_1, C_2 and C_3. The overlapping part (black) shows that C_2 is a part of C_1 and C_3.

data have chances to contribute to EDM but have no chance to contribute to EMD, making EMD more robust than EDM.

As shown in Fig. 2, EDM is unable to distinguish the good-fitting model (Fig. 2(a)) from the over-fitting model (Fig. 2(b)) for the nearly perfect data. For the imperfect data, EDM even prefers the over-fitting model (Fig. 2(f)) than the good-fitting model (Fig. 2(e)). It is worth noting that the over-fitting problem is notorious in machine learning. For both of the data, EMD prefers the good-fitting models (Fig. 2(a) and (e)) than the over-fitting models (Fig. 2(b) and (f)) and the under-fitting models (Fig. 2(c) and (g)). However, EMD is unable to distinguish the good-fitting models from the incomplete-fitting models (Fig. 2(d) and (h)). To address this problem, a regularization term should be used to regularize EMD.

4.1. Full similarity

It is challenging to design a regularization term to regularize EMD. We observe that, in real world, two models $M \subset \mathbb{R}^k$ and $N \subset \mathbb{R}^k$ are identical if and only if every point of N is in M (i.e., $d(N, M) = 0$) and the measure of N is equal to the measure of M. That means the measure can be used as a regularization term to estimate similarity. It is worth noting that different types of models have different types of measures. For example, the measure of a curve is its length, while the measure of a surface is its area.

We hence propose a mean measure to represent the similarity between a model $M \subset \mathbb{R}^k$ and a data point set $D \subset \mathbb{R}^k$. Denoting the measure of M as $|M|$, the mean measure $r(\cdot, \cdot)$ is defined as the ratio of $|M|$ to EMD:

$$r(M, D) = \frac{|M|}{\epsilon + d^\lambda(M, D)}, \qquad (6)$$

where $\lambda > 0$ is used to tune the weight of the measure and EMD, and ϵ is a small positive number used to derive different similarities for the models with different measures but the same EMD of 0. For example, as shown in Fig. 3, both $d(C_1, C_1)$ and $d(C_2, C_1)$ are equal to 0. If ϵ is 0, then both $r(C_1, C_1)$ and $r(C_2, C_1)$ are infinite despite C_1 is more similar to C_1 than C_2. In practice, when ϵ is sufficiently small, the mean measure can ensure that a model is most similar to itself than any other models.

We now prove that the mean measure is a strict estimator in some 1-dimensional case. We consider a 1-dimensional model M that consists of only one continuous point set and has a positive finite measure, i.e., $M = [x, y]$, $-\infty < x < y < +\infty$. Let $\mathcal{M}_u \subset \mathcal{P}_u$ be the universal set of such models.

Lemma: $|N| \leq |M| + 2d(N, M)$, $\forall N \in \mathcal{M}_u$.

Proof. Let $N = [z, t]$. Note that, $|M| = y - x$ and $|N| = t - z$. There are six cases of relations of x, y, z and t: (1) $x < z < t < y$, (2) $z \leq x < t < y$, (3) $z < t \leq x < y$, (4) $z \leq x < y \leq t$, (5) $x < z < y \leq t$, and (6) $x < y \leq z < t$. Cases 2 and 5 are similar. Cases 3 and 6 are similar. We only need to prove the first four cases. For Case 1: From the case condition we have $x < z$ and $t < y$, so $t + x < z + y$, so $t - z < y - x$, i.e., $|N| < |M|$. Meanwhile, it is easy to compute that $d(N, M) = 0$ for this case. Therefore, $|N| \leq |M| + 2d(N, M)$. For Cases 2 and 3: $d(N, M) = x - z$, $|M| + 2d(N, M) - |N| = y - x + 2(x - z) - (t - z) = (y - t) + (x - z) > 0$, proved. For Case 4: if $t - y > x - z$, then $d(N, M) = t - y$, $|M| + 2d(N, M) - |N| = y - x + 2(t - y) - (t - z) = (t - y) - (x - z) > 0$. If $t - y \leq x - z$, then $d(N, M) = x - z$, $|M| + 2d(N, M) - |N| = (x - z) - (t - y) \geq 0$. proved.

Theorem: Given a $\epsilon > 0$, let $\mathcal{M}_u^{(\epsilon)} = \{M | M \in \mathcal{M}_u, |M| > 2\epsilon\}$. When $\lambda = 1$, the mean measure (Eq. (6)) is a strict similarity estimator (Eq. (3)) for the model set $\mathcal{M}_u^{(\epsilon)}$. That is, when $\lambda = 1$, for a $\epsilon > 0$, $\forall M \in \mathcal{M}_u^{(\epsilon)}$, $\forall N \in \mathcal{M}_u^{(\epsilon)} \setminus \{M\}$, $r(N, M) < r(M, M)$.

Proof. $\forall M \in \mathcal{M}_u^{(\epsilon)}$, $\forall N \in \mathcal{M}_u^{(\epsilon)} \setminus \{M\}$, noting that $|M| > 0$ and $|N| > 0$, (1) If $d(N, M) = 0$, then $N \subset M$, then $|N| < |M|$, so $r(N, M) = (|N|/\epsilon) < r(M, M) = (|M|/\epsilon)$; (2) If $d(N, M) > 0$, then $2d(N, M)\epsilon < d(N, M)|M|$ because $2\epsilon < |M|$, then $|M|\epsilon + 2d(N, M)\epsilon < |M|\epsilon + d(N, M)|M|$, so $|N|\epsilon < |M|(\epsilon + d(N, M))$ according to the lemma, so $|N|/(\epsilon + d(N, M)) < |M|/\epsilon$, i.e., $r(N, M) < r(M, M)$.

As shown in Eq. (6), to maximize the mean measure similarity over a model set for a data point set, we first minimize the EMD (denominator) (e.g., Fig. 2(d)), and then maximize the measure $|M|$ until it equals the measure of the ground-truth model (e.g., Fig. 2(a)). After that, the model M has no chance to become larger (i.e., $|M|$ becomes larger). Because the similarity will become smaller as EMD will inevitably be much larger if $|M|$ is larger than the ground-truth measure (e.g., Fig. 2(b)).

Note that, the values of mean measure are comparable on the same data point set, but are incomparable on different data point sets. That is, it does not make sense to compare the mean measure values across different data point sets. For example, as shown in Fig. 3, it is meaningless to compare $r(C_2, C_2)$ and $r(C_2, C_1)$, although $r(C_2, C_2) = r(C_2, C_1)$. It is also worth noting that, the data is unnecessary to have a geometric measure. That is, the data can be a discrete point set (i.e., a point cloud). If the data D is discrete, then ϵ is trivial because $d(M, D)$ is always larger than 0 (e.g., Fig. 2).

4.2. Partial similarity

Mean measure is defined as a full similarity estimator as it assumes that the data is complete. However, if the data is incomplete, we have to calculate partial similarity, which is challenging. Partial similarity is not straightforward and is fundamentally different from full similarity. If two models have a common part, then these two models are partially similar. As shown in Fig. 3, each pair of C_1, C_2 and C_3 are partially similar. We expect that the partial similarity between C_1 and C_2 is equal to the partial similarity between C_2 and C_2. Because the common part between C_1 and C_2 is the same as the common part between C_2 and C_2.

Therefore, we propose a Weighted Mean Measure (WMM) to represent the partial similarity between a geometric model M and a data point set D. We divide M into c non-overlapping sub-models: $M = \overset{c}{\underset{i=1}{\cup}} M_i$, and define WMM as:

$$r_w(M, D) = \frac{\sum\limits_{i=1}^{c} w_i |M_i|}{\epsilon + d_w^{\lambda}(M, D)}, \tag{7}$$

where w_i is the weight: $w_i = \exp(-d(M_i, D)h)$, h is a non-negative weighting factor. When h is 0, WMM becomes a full similarity estimator. $d_w(\cdot, \cdot)$ is the weighted mean error:

$$d_w(M, D) = \frac{\sum\limits_{i=1}^{c} w_i d(M_i, D)}{\sum\limits_{i=1}^{c} w_i}. \tag{8}$$

By weighting, the sub-models of M far away from D have less contribution to the computation of WMM than those close sub-models. In other words, the common part of M and D makes major contribution to WMM, making WMM plausible to estimate partial similarity.

4.3. Computational complexity

The computational complexity of mean measure almost depends on that of EMD, as the measure of a model can be immediately obtained from the model parameter. The computation of EMD consists of two steps. First, the model is uniformly divided into sub-models, and the center points of the sub-models are sampled (Section 5). Second, the nearest point is searched in the data for a point sampled from the model. This is time-consuming if the data contains a large number of points. A common way to perform the nearest neighbour searching is using a k-dimensional tree [71]. Let m be the number of points sampled from a model M and n be the number of points of data D, the complexity to compute the mean measure between M and D is about $O(m\log(n))$.

5. Proposed early rejection strategy

In our PMF method, the optimization algorithm (Section 3.4) accepts a proposed model with a larger similarity. However, it is time-consuming to accurately compute a similarity, as many points

---

Algorithm 1 The proposed PMF method with early rejection.

> **input**: a probabilistic program g with parameter $\boldsymbol{\theta}$, a data point set D, the posterior function $f(\boldsymbol{\theta}|D)$, a proposal function q, an iteration tolerance i_{max}, and a minimal model dividing resolution δ_{min}.
> **output**: a maximum a posteriori estimate of $\boldsymbol{\theta}$: $\boldsymbol{\theta}^*$.
> Randomly initialize $\boldsymbol{\theta}^{(0)}$, $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^{(0)}$
> **for** $i = 0$ to i_{max} **do**
>   Sample $\tilde{\boldsymbol{\theta}} \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(i)})$
>   Compute η_{max} of $M_g^{\tilde{\boldsymbol{\theta}}}$ according to δ_{min}
>   **for** $\eta = 0$ to η_{max} **do**
>   **if** $f_\eta(\tilde{\boldsymbol{\theta}}|D) > f(\boldsymbol{\theta}^{(i)}|D)$ **then** $\boldsymbol{\theta}^{(i+1)} \leftarrow \tilde{\boldsymbol{\theta}}$
>   **else** $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)}$, **break**
>   **if** $f(\boldsymbol{\theta}^{(i+1)}|D) > f(\boldsymbol{\theta}^*|D)$ **then** $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^{(i+1)}$
> **return** $\boldsymbol{\theta}^*$

---

have to be sampled from the model to compute an accurate similarity. We observe that, it is sufficient to determine the dissimilarity by sampling only one point from the model. As shown in the left part of Fig. 4, Curve C_4 consists of one horizontal line segment, and Curve C_5 consists of two vertical line segments, these two curves are dissimilar. The similarities computed by sampling one point (Fig. 4(c)) and four points (Fig. 4(a)) are the same and equal to the true similarity. However, if two points are sampled (Fig. 4(b)), the computed similarity will be incorrect as it shows that C_4 and C_5 are similar. It can be inferred that a small similarity between two point sets means that these two point sets are dissimilar. However, a large similarity between two point sets does not mean that these two point sets are really similar. In other words, a proposed model should be accepted carefully but rejected boldly.

Consequently, to reduce computational time, we propose a coarse-to-fine model dividing strategy for similarity calculation to reject dissimilar models in advance. We take a square surface for example (as shown in the right part of Fig. 4), and the conclusions can be easily adapted to other types of geometric models. Assuming that the length of the square surface is γ, given a predefined minimal dividing resolution δ_{min}, the maximum dividing level is:

$$\eta_{max} = \log_2(\gamma/\delta_{min} + 1). \tag{9}$$

At each level η, we uniformly divide the surface into $2^{2\eta}$ sub-surfaces, and sample only one point (center point) from each sub-surface to calculate EMD. The similarity is then calculated to decide whether to accept or reject the proposed surface. If it is accepted, then the surface is divided into more sub-surfaces and more points are sampled at a higher level to obtain more accurate similarity. Otherwise, a new surface is proposed.

The pseudo code of our PMF method is presented in Algorithm 1, where $f_\eta(\cdot|\cdot)$ denotes the posterior computed at dividing level η. Let δ_D be the resolution of the data D: $\delta_D = \min\limits_{\mathbf{p}\in D} \min\limits_{\mathbf{q}\in D\setminus\{\mathbf{p}\}} \|\mathbf{p} - \mathbf{q}\|$, the minimal model dividing resolution δ_{min} should be set at least two times smaller than δ_D to obtain accurate similarity.

6. Experiments

We implemented our method in MATLAB and conducted several experiments including estimator comparison (Section 6.1), cylinder fitting (Section 6.2), character fitting (Section 6.3), and building fitting (Section 6.4). In all experiments, we set $\epsilon = 10^{-8}$. Unless stated, we use WMM with $\lambda = 2$, $h = 0$, and set δ_{min} to 0.3 times the resolution of the data.
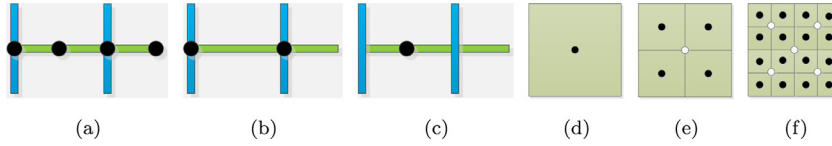
Fig. 4. An illustration of early rejection. Left: Overlap between Curves C_4 (green) and C_5 (blue). Black dots represent the points sampled from C_4. (a), (b) and (c) show that 4, 2 and 1 point(s) are sampled, respectively. Right: A square surface to illustrate the coarse-to-fine model dividing. The dividing levels in (d), (e) and (f) are 0, 1 and 2, respectively. The black dots represents the points sampled in the current level, and the white dots represents the points sampled in previous levels. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
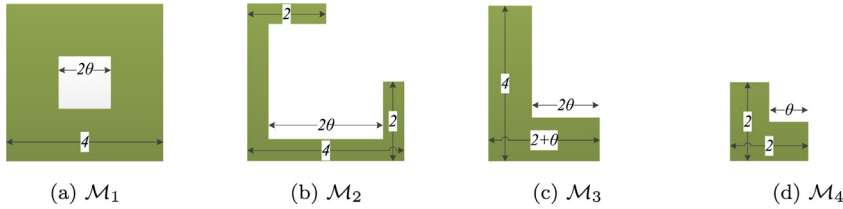
Fig. 5. Model sets. (a) Model set $\mathcal{M}_1$, (b) Model set $\mathcal{M}_2$, (c) Model set $\mathcal{M}_3$, and (d) Model set $\mathcal{M}_4$. Each of these 4 model sets has only one parameter $\theta \in [0, 2]$.
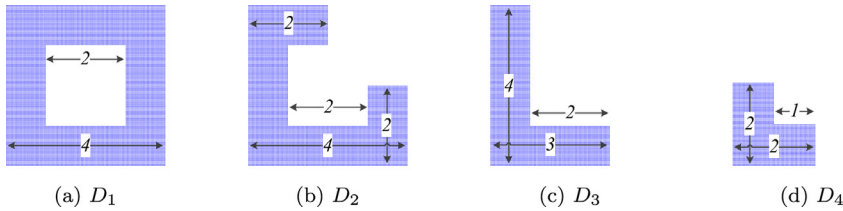
Fig. 6. Data point sets. From left to right: Point sets D_1, D_2, D_3 and D_4. For $i = 1$ to 4, D_i is a point cloud uniformly sampled from Model $M_i^{\theta=1}$ with 0.02 resolution. D_1, D_2, D_3 and D_4 consist of 12288, 9216, 6144 and 3072 points, respectively.

6.1. Estimator comparison

We compare several estimators with our WMM estimator by fitting 4 procedural models (Fig. 5) to 4 data point sets (Fig. 6). Model $M_1^\theta \in \mathcal{M}_1$ is a ring-like surface between an outer square and an inner square. The outer and inner squares share the same center. The length of the outer and inner squares are 4 and 2θ, respectively. Models $M_2^\theta \in \mathcal{M}_2$, $M_3^\theta \in \mathcal{M}_3$ and $M_4^\theta \in \mathcal{M}_4$ are 0.75, 0.5 and 0.25 part of M_1^θ, respectively. As shown in Fig. 6, for $i = 1$ to 4, the ground-truth model of D_i is $M_i^{\theta=1}$. In this paper, we refer to the target model of a data point set as the model which is partially similar to the ground-truth model. Therefore, for each data point set in Fig. 6, there is a target model in each model set (as shown in Fig. 5). That is, for $i = 1$ to 4 and $j = 1$ to 4, the target model of D_i in $\mathcal{M}_j$ is $M_j^{\theta=1}$.

The estimators used for comparison include negative Hausdorff distance (-HD), negative EDM (-EDM), negative voxel difference (-VD), and inlier number (IN). -VD is used in PMF methods [4,6,8,9], while IN is the foundation of many classical model fitting methods such as RANSAC based methods [32–34,37,40].

The comparison results of fitting the models (Fig. 5) to the data point sets (Fig. 6) are shown in Fig. 7. In these 16 experiments, we set $h = 5$ for WMM and set the resolution for -VD calculation to 0.04. Since the target models of the data are models with $\theta = 1$, it is expected that the models with $\theta = 1$ have the largest similarities. As shown in Fig. 7, our WMM is the only estimator to achieve this goal for all experiments. -HD is successful for full fitting (Fig. 7(a), (f), (k), and (p)), but failed for partial fitting except Fig. 7(g). IN fails to distinguish the target models from models with $\theta < 1$ for all experiments except Fig. 7(k). The computational time of these 16 experiments for -HD, -VD, -EDM, IN, and WMM are 94.2, 1.34, 21.8, 0.0949, and 74 s, respectively. Our WMM is faster than -HD.

It is worth noting that -HD, -EDM and WMM prefer to sample points from model with a smaller resolution to obtain more accu-

Table 2
WMM similarities between the target models and data point sets. The diagonal elements represent the similarities between the ground-truth models and the data. It is shown that, for the same data, the similarity between the ground-truth model and the data is the largest among all similarities.

Model	$M_1^{\theta=1}$	$M_2^{\theta=1}$	$M_3^{\theta=1}$	$M_4^{\theta=1}$
Data				
D_1	**83840.8**	62861.7	41920.4	20941.3
D_2	21987.3	**62861.7**	41920.4	20941.3
D_3	8086.26	8322.63	**41920.4**	20941.3
D_4	2486.52	2494.83	2642.86	**20941.3**

rate similarity. However, -VD produces worse results with a smaller resolution for a discrete point set. Fine voxelization of a discrete data produces more empty voxels. Therefore, an empty model (e.g. $M_1^{\theta=2}$) is preferred, as shown by the example in Fig. 8(a). This indicates that voxelization is unsuitable for fine fitting of point clouds.

It is interesting to find that Fig. 7(c) is similar to Fig. 7(d). Actually, the original similarities before normalization are different. As shown in Table 2, the WMM similarities are comparable across different model sets for the same data. It can be seen from Table 2 and Fig. 7 that, a model is most similar to itself than any other models using WMM. Finally, we take the experiment of fitting M_1^θ to D_2 as an example to evaluate the effect of weighting factor h. As shown in Fig. 8(b), WMM is very stable with respect to different values of h.

6.2. Cylinder fitting

In this section, we investigate the effect of our early rejection strategy by fitting a cylinder model $\mathcal{M}_5$ to data point sets D_5, D_6, and D_7 (Fig. 9). D_5 and D_6 contain gross-outliers, while D_7 contains pseudo-outliers. The cylinder model $\mathcal{M}_5$ has 7 parameters (3 for
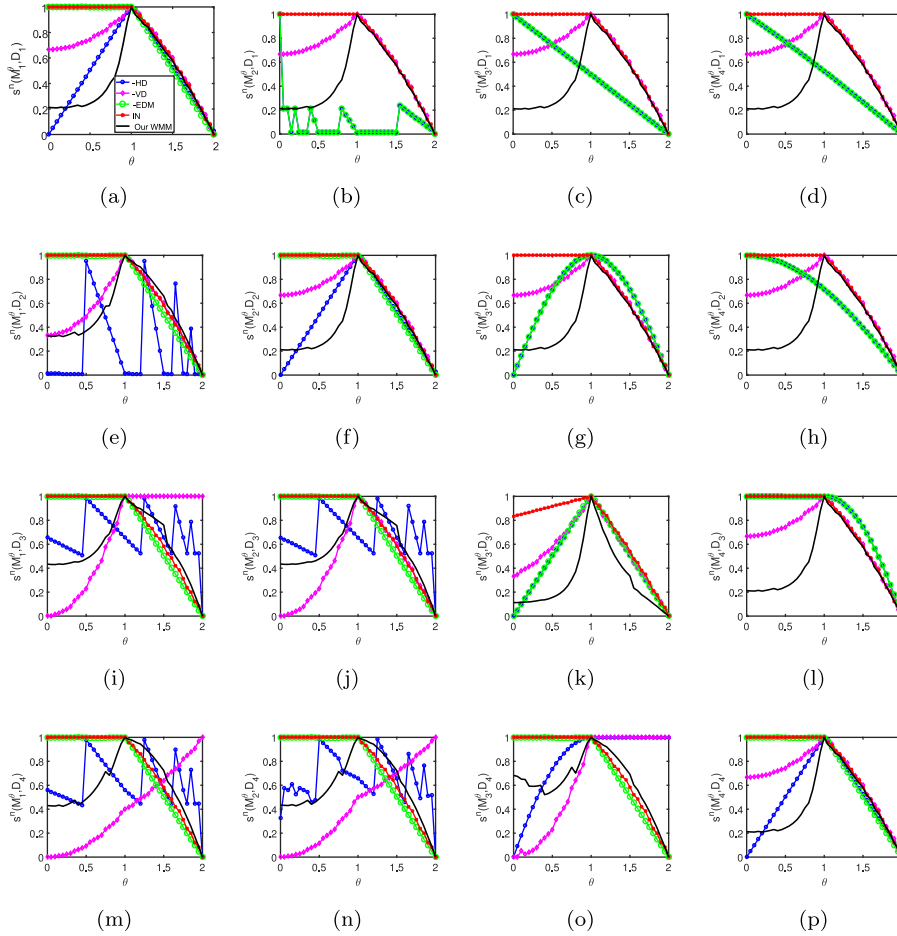
Fig. 7. Comparative results achieved by different estimators. From left to right: the results of fitting Models M_1^θ, M_2^θ, M_3^θ and M_4^θ to the data point sets. From top to bottom: the results of fitting the models to Data D_1, D_2, D_3 and D_4. The vertical axis $s^n(\cdot,\cdot)$ denotes the normalized similarity. We uniformly normalize the similarities into a range of [0, 1]. The legend for these figures is presented in (a). The diagonal figures represent the results of full fitting. The figures below diagonal, above diagonal represent the results of partial fitting on the incomplete data, partial fitting on the data with pseudo-outliers, respectively.
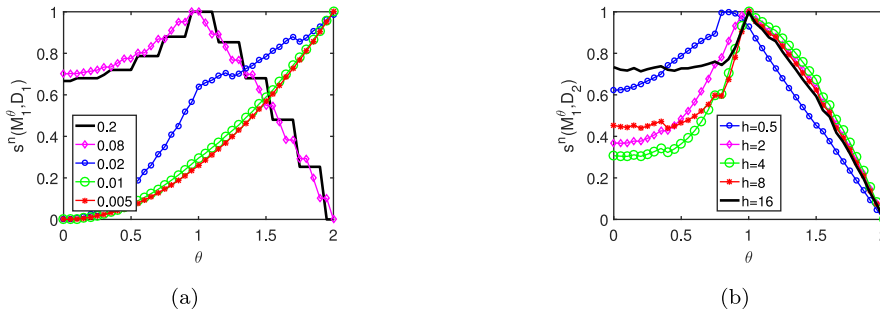

Fig. 8. Parameter sensitivity. (a) -VD similarities of fitting M_1^θ to D_1 with resolutions 0.2, 0.08, 0.02, 0.01 and 0.005. (b) WMM similarities of fitting M_1^θ to D_2 with weighting factor h=0.5, 2, 4, 8 and 16.

location, 1 for radius, 1 for height, and 2 for start and end angles). Some sample models of $\mathcal{M}_5$ are shown in the top row of Fig. 10.

For each data, we use the cuckoo search (CS) [16] and Metropolis-Hastings (MH) [6] algorithms with or without early rejection (ER) to perform cylinder fitting. The evolutions of similarity during fitting are shown in Fig. 11, where each line represents the mean similarity values and the patch around each line represents the standard deviations. The mean values and standard deviations are computed from 60 times of fitting. The fitting is performed three hours every time. Some fitted models are shown in Fig. 10. Fig. 11 shows that, with our ER strategy, both CS and MH can be accelerated by about 3 times. As shown in Fig. 11(a) and

(b), the target similarities still remain the largest similarities after a long evolution time. This indicates that our method is robust to gross outliers. Fig. 10(f) shows that our method is also robust to pseudo-outliers. It is also shown that, CS is more efficient to handle pseudo-outliers (Fig. 11(c)), whereas MH is more efficient to deal with massive gross-outliers (Fig. 11(b)).

6.3. Character fitting

We also applied our method to perform few-shot recognition on noisy variants of the MNIST dataset [73,74], which contain images of size 28×28 for digits 0–9. For convenience, we resized the

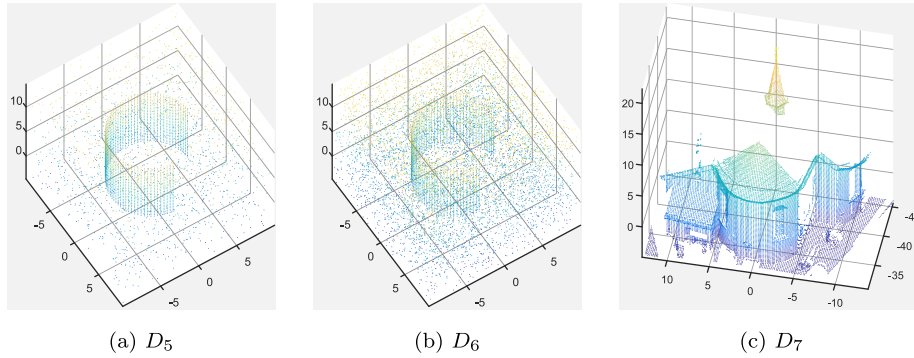(a) D_5 (b) D_6 (c) D_7

Fig. 9. Data point sets. From left to right: Data D_5, D_6, and D_7. D_5 and D_6 are generated by adding low-level and high-level gross outliers to a noise-free data, respectively. The noise-free data contains 2048 points sampled from a cylinder surface with resolution 0.2. D_5 consists of 4096 points, while D_6 consists of 10,240 points. D_7 has 14,778 points and is generated by downsampling a laser scanning point cloud [72] with a resolution of 0.2.
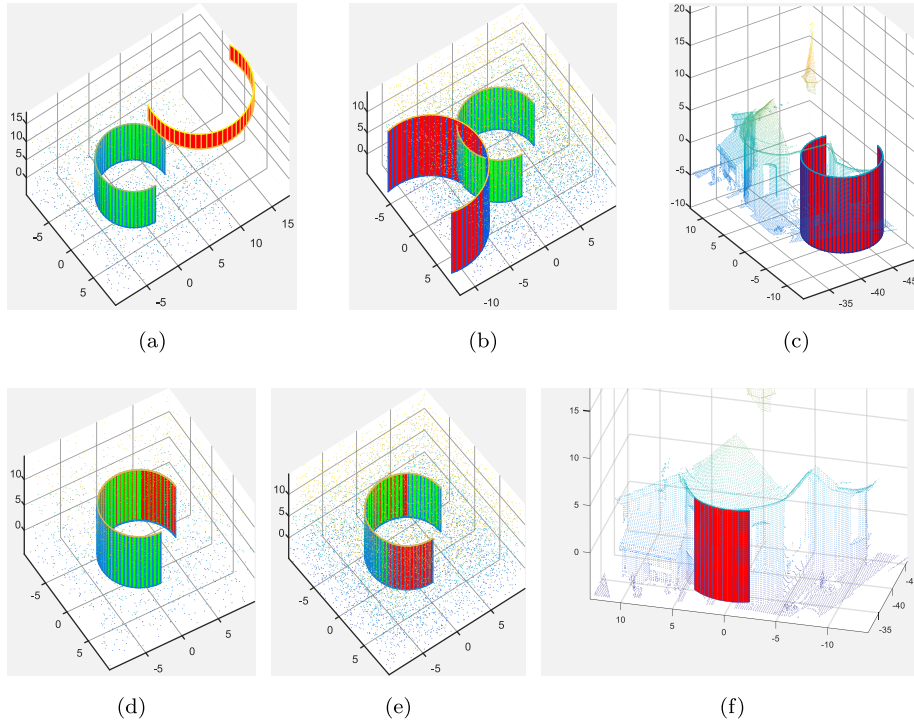

(a) (b) (c)

(d) (e) (f)

Fig. 10. Fitted cylinder models (red), the target models (green), and the data point sets (other colors, Fig. 9). Top row: randomly initialized models. Bottom row: final fitted models (Fig. 11). From left to right: fitting the cylinder to D_5, D_6, and D_7. Note that, we do not show the target model of the real data D_7 as it is unknown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

images to 65×65 and then zero-paded them to 105×105. Fig. 12 illustrates the process of 2-shot recognition. We extract a character model (the second left column in Fig. 12) from a given training image (the leftmost column in Fig. 12) based on the bottom-up method proposed in [4]. The bottom-up method is originally used to handle images with size 105×105. The extracted model consists of several strokes. Each stroke has 14 parameters (2 for location, 10 for shape, 1 for scale, and 1 for rotation). The model also has 6 global parameters (1 for rotation, 2 for affine, 1 for width, and 2 for location). Let the parameters of the model be $\boldsymbol{\theta}_t$, we define a range around $\boldsymbol{\theta}_t$ to define a probabilistic program corresponding to the training image (Table 3). Some models generated by the probabilistic program are shown in the right columns of Fig. 12.

We then perform PMF for each probabilistic program to find the ground-truth model of a test image. That is, the test image is classified to the class of the training image with the highest fitting similarity. Note that, the similarity is computed with the binarization of images. Table 4 shows the classification results achieved

Table 3
Parameter ranges of the probabilistic character programs. The ranges of the shapes for 1-shot and 5-shot recognition are defined as $[\theta_t - 5, \theta_t + 5]$ and $[\theta_t - 2.5, \theta_t + 2.5]$, respectively.

θ	rotation (global)	affine	width	location (global)	location	scale	rotation
$\theta_{\min}$	$\theta_t - 90$	1/1.5	0	$\theta_t - 60$	$\theta_t - 2.5$	$\theta_t/1.1$	$\theta_t - 20$
$\theta_{\max}$	$\theta_t + 90$	1.5	6	$\theta_t + 60$	$\theta_t + 2.5$	$1.1\theta_t$	$\theta_t + 20$

on the noisy MNIST data [74]. The noisy data are corrupted by six types of noise with three levels of intensity. We compare our PMF method with the recursive cortical network (RCN) [74] for the tasks of 1-shot and 5-shot recognition. RCN is the state-of-the-art few-shot recognition method. For both PMF and RCN, we sequentially select the training and testing images from the data to perform several runs of recognition. Specifically, in each run, we select n training images and 1 test image from each class to perform n-shot
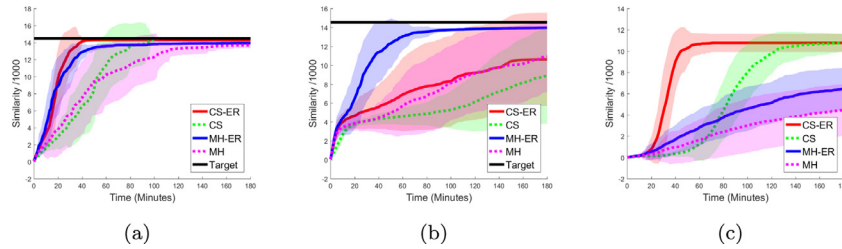
Fig. 11. Cylinder fitting results. (a), (b) and (c) are the results of fitting the cylinder model $\mathcal{M}_5$ to D_5, D_6 and D_7 (Fig. 9), respectively. Each line represents average values of the similarities, while the patch around each line represents the standard deviation of the similarities. CS-ER, CS, MH-ER, MH, and Target denote CS with ER, CS without ER, MH with ER, MH without ER, and the similarity of the target model, respectively. These experiments were conducted on a machine with an Intel Xeon E5-2650 v4 2.20 GHz CPU.
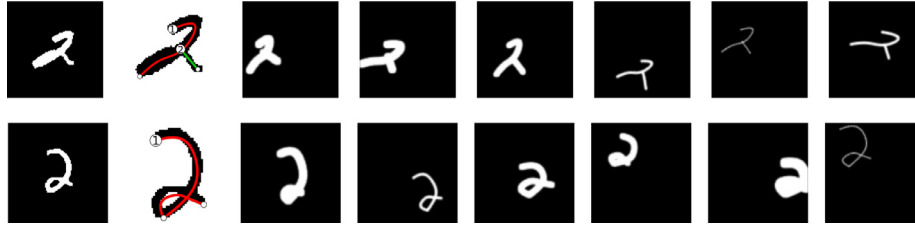

Fig. 12. Generation of character models for Class '2'. The leftmost column: training images. The second column: character models extracted from the training images. The other columns: models randomly generated by the probabilistic program corresponding to the training images. Top row: the first training. Bottom row: the second training.

Table 4

Classification accuracies achieved on the noisy MNIST data. bg-noise denotes background noise. RCN-1 denotes RCN with 1 training image. The pool size, perturbation factor for RCN-1 and RCN-5 are set to 57, 1.0 and 45, 1.0, respectively. These settings are reported to achieve the best performance for RCN-1 and RCN-5 on the noise-free MNIST data [74]. In this paper, for the noise-free MNIST data, RCN-1, PMF-1, RCN-5, and PMF-5 achieve accuracies of 0.722, 0.604, 0.9, and 0.83, respectively.

Noise Type	bg-noise	border	patches	grid	clutter	deletion
Noise Level 1						
RCN-1	0.628	0.524	0.63	0.298	0.338	0.566
PMF-1	**0.63**	0.23	0.608	**0.426**	**0.428**	**0.588**
RCN-5	0.735	0.82	0.815	0.315	0.495	0.73
PMF-5	**0.825**	0.27	**0.835**	**0.57**	**0.55**	**0.735**
Noise Level 2						
RCN-1	0.548	0.426	0.64	0.22	0.302	0.498
PMF-1	**0.616**	0.142	0.608	**0.47**	**0.408**	**0.556**
RCN-5	0.695	0.735	0.81	0.3	0.385	0.65
PMF-5	**0.875**	0.135	0.805	**0.65**	**0.52**	**0.78**
Noise Level 3						
RCN-1	0.466	0.344	0.606	0.202	0.282	0.478
PMF-1	0.43	0.146	0.6	**0.312**	**0.374**	**0.556**
RCN-5	0.575	0.645	0.795	0.23	0.34	0.575
PMF-5	**0.605**	0.135	0.795	**0.385**	**0.535**	**0.745**

recognition. The numbers of runs for 1-shot and 5-shot recognition are 50 and 20, respectively.

In these character fitting experiments, we set $\delta_{\min}$ to 0.5 times the resolution of the data. Specially, for the data corrupted by grid and clutter, we set $\lambda = 5$. Let n be the number of parameters of a probabilistic program, we also set the iteration tolerance to $1000n$ and $500n$ for 1-shot and 5-shot recognition, respectively. As shown in Table 4, our method outperforms RCN on the data corrupted by grid, clutter, and deletion in all cases. In particular, for the data corrupted by the level-2 grid, our PMF-5 method outperforms RCN-5 by 35% . Our method also outperforms RCN on the data with back-

ground noise in most cases. For the data with patches, our method is only slightly worse than RCN.

However, for the data with border, the performance of our method decreases drastically. In these cases, our method recognizes most images as '7'. That is, our method finds a big '7' located on the border of the image. To some extent, this is reasonable as the border can be seen to contain a character '7'. Nevertheless, a potential solution to address this problem is to impose some more specific prior on the objective function (Eq. (1)), such as [4].
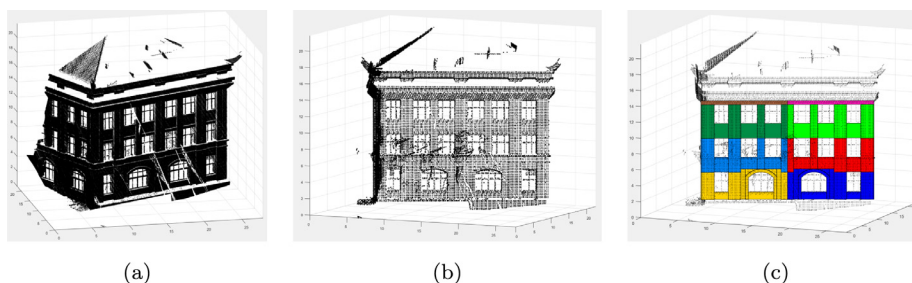
Fig. 13. Fitted model. (a) An original point cloud consisting of 385,793 points. (b) A point cloud D_{11} consisting of 23,266 points. D_{11} is generated by downsampling the original point cloud (a) with a resolution of 0.2. (c) Final fitted model (after 55,080 iterations) for fitting $\mathcal{M}_8$ to D_{11}.
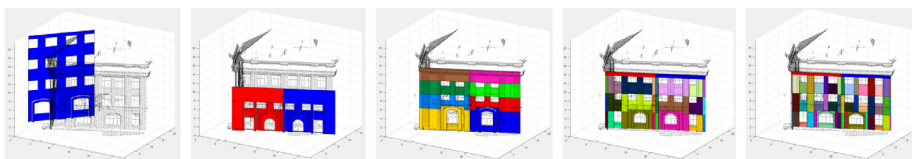

Fig. 14. Model evolution of fitting $\mathcal{M}_8$ to D_{11}. From left to right: fitted models (color) at different iterations: 0, 360, 1240, 9200, and 39360. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.4. Building fitting

We also conducted experiments to fit building facades on laser scanning 3-dimensional point clouds. The results of fitting a facade model $\mathcal{M}_8$ to a 3-dimensional point cloud D_{11} (Fig. 13(b)) are shown in Figs. 13 and 14. $\mathcal{M}_8$ has 18 parameters (1 for rotation, 3 for location, 2 for extrusion, and 12 for size). We use the CGA (Computer Generated Architecture) shape grammar [59] to manually create the probabilistic program to define $\mathcal{M}_8$. The depth of the derivation tree [6] of the probabilistic program is 5. That is, each model in $\mathcal{M}_8$ has a 5-level hierarchical structure. Fig. 13(c) shows the terminal rules of the final model. The colors of the terminal rules represent the third level structure of the final model. That is, the terminal rules in the same color are derived from the same non-terminal rule at the third level of the derivation tree. The first, second, third, fourth, and fifth levels of structures are shown in Fig. 14 from left to right.

7. Conclusions

In this paper, we investigated the robust PMF problem. We proposed a novel estimator for PMF to handle imperfect data. The optimization problem in PMF was formulated as a Bayesian inference problem and was addressed by the cuckoo search algorithm. We also proposed a novel technique to accelerate the inference process. Our PMF method has been tested on complex geometric models and imperfect data. Experimental results show that, our estimator is robust to gross-outliers and a wide variety of pseudo-outliers. It is also shown that, our method can be accelerated by several times.

Our estimator is highly robust but extremely easy for understanding and implementation. It consists of only two natural concepts: the length (or area) of the model, and the error from model to data. It has only one parameter (λ in Eq. (6)) for complete data, and has only two parameters (λ and h in Eq. (7)) for incomplete data. Note that, the dividing parameter δ_{mim} is trivial in terms of effectiveness as it should be set as small as possible.

We believe that our work takes a step towards making PMF more useful. However, several issues still remain open. First, all the point sets involved in the experimental part of this paper are either 2-dimensional or 3-dimensional. Although our estimator is theoretically not limited to low-dimensional point sets, it is time-consuming to apply our estimator on high-dimensional point sets.

Second, it is time-consuming to perform PMF if the procedural model has a large number of parameters. Advanced techniques are expected to address these issues.

References

[1] R.M. Smelik, T. Tutenel, R. Bidarra, B. Benes, A survey on procedural modelling for virtual worlds, Comput. Graphics Forum 33 (6) (2014) 31–50.

[2] T.D. Kulkarni, P. Kohli, J.B. Tenenbaum, V. Mansinghka, Picture: A probabilistic programming language for scene perception, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4390–4399.

[3] P. Musialski, P. Wonka, D.G. Aliaga, M. Wimmer, L. Gool, W. Purgathofer, A survey of urban reconstruction, Comput. Graphics Forum 32 (6) (2013) 146–177.

[4] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1332–1338.

[5] J. Weissenberg, H. Riemenschneider, M. Prasad, L.V. Gool, Is there a procedural logic to architecture? in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 185–192.

[6] J.O. Talton, Y. Lou, S. Lesser, J. Duke, R. Mch, V. Koltun, Metropolis procedural modeling, ACM Trans. Graphics 30 (2) (2011) 11.

[7] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Měch, O. Deussen, B. Benes, Inverse procedural modelling of trees, Comput. Graphics Forum 33 (6) (2014) 118–131.

[8] D. Ritchie, B. Mildenhall, N.D. Goodman, P. Hanrahan, Controlling procedural modeling programs with stochastically-ordered sequential monte carlo, ACM Trans. Graphics 34 (4) (2015) 105.

[9] K. Haubenwallner, H.-P. Seidel, M. Steinberger, Shapegenetics: using genetic algorithms for procedural modeling, Comput. Graphics Forum 36 (2) (2017) 213–223.

[10] M. Bokeloh, M. Wand, H. Seidel, A connection between partial symmetry and inverse procedural modeling, ACM Trans. Graphics 29 (4) (2010) 104.

[11] A. Martinovic, L.V. Gool, Bayesian grammar learning for inverse procedural modeling, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 201–208.

[12] R. Gadde, R. Marlet, N. Paragios, Learning grammars for architecture-specific facade parsing, Int. J. Comput. Vision 117 (3) (2016) 290–316.

[13] C.A. Vanegas, I. Garcia-Dorado, D.G. Aliaga, B. Benes, P. Waddell, Inverse design of urban procedural models, ACM Trans. Graphics 31 (6) (2012) 168.

[14] G. Wan, A. Sharf, Grammar-based 3D facade segmentation and reconstruction, Comput. Graphics 36 (4) (2012) 216–223.

[15] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, N. Paragios, Parsing facades with shape grammars and reinforcement learning, IEEE Trans. Pattern Anal. Mach. Intell. 35 (7) (2013) 1744–1756.

[16] X.S. Yang, S. Deb, Engineering optimisation by cuckoo search, Int. J. Math. Model. Numer. Optim. 1 (4) (2010) 330–343.

[17] A. Iglesias, A. Glvez, Cuckoo search with Levy flights for reconstruction of out-line curves of computer fonts with rational Bezier curves, in: IEEE Congress on Evolutionary Computation, 2016, pp. 2247–2254.

[18] G. Nishida, I. Garcia-Dorado, D.G. Aliaga, B. Benes, A. Bousseau, Interactive sketching of urban procedural models, ACM Trans. Graphics 35 (4) (2016) 130:1–130:11.

[19] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, S. Maji, CSGNet: Neural shape parser for constructive solid geometry, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Roy. Stat. Soc. B (Met.) 58 (1) (1996) 267–288.

[21] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[22] J. Wang, K. Xu, Shape detection from raw LiDAR data with subspace modeling, IEEE Trans. Visual Comput. Graphics 23 (9) (2017) 2137–2150.

[23] R. Tennakoon, A. Babhadiashar, Z. Cao, R. Hoseinnezhad, D. Suter, Robust model fitting using higher than minimal subset sampling, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2016) 350–362.

[24] C. Papalazarou, P.H. de With, P. Rongen, Sparse-plus-dense-RANSAC for estimation of multiple complex curvilinear models in 2D and 3D, Pattern Recognit. 46 (3) (2013) 925–935.

[25] A. Nurunnabi, G. West, D. Belton, Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data, Pattern Recognit. 48 (4) (2015) 1404–1419.

[26] A. Nurunnabi, Y. Sadahiro, D.F. Laefer, Robust statistical approaches for circle fitting in laser scanning three-dimensional point cloud data, Pattern Recognit. 81 (2018) 417–431.

[27] M. Berger, A. Tagliasacchi, L.M. Seversky, P. Alliez, G. Guennebaud, J.A. Levine, A. Sharf, C.T. Silva, A survey of surface reconstruction from point clouds, Comput. Graphics Forum 36 (1) (2017) 301–329.

[28] N. Aspert, D. Santa-Cruz, T. Ebrahimi, MESH: measuring errors between surfaces using the Hausdorff distance, in: IEEE International Conference on Multimedia and Expo, 1, 2002, pp. 705–708.

[29] H.-K. Zhao, S. Osher, R. Fedkiw, Fast surface reconstruction using the level set method, in: IEEE Workshop on Variational and Level Set Methods in Computer Vision, 2001, pp. 194–201.

[30] V. Lempitsky, Y. Boykov, Global optimization for shape fitting, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, doi:10.1109/CVPR.2007.383293.

[31] A. Hornung, L. Kobbelt, Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information, in: Eurographics Symposium on Geometry Processing, 2006, pp. 41–50.

[32] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.

[33] H. Le, T.J. Chin, D. Suter, An exact penalty method for locally convergent maximum consensus, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 379–387.

[34] P.H. Torr, A. Zisserman, MLESAC: A new robust estimator with application to estimating image geometry, Comput. Vis. Image Und. 78 (1) (2000) 138–156.

[35] C.-M. Cheng, S.-H. Lai, A consensus sampling technique for fast and robust model fitting, Pattern Recognit. 42 (7) (2009) 1318–1329.

[36] G. Wang, Y. Chen, X. Zheng, Gaussian field consensus: a robust nonparametric matching method for outlier rejection, Pattern Recognit. 74 (2018) 305–316.

[37] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for pointcloud shape detection, Comput. Graphics Forum 26 (2) (2007) 214–226.

[38] H.S. Wong, T.-J. Chin, J. Yu, D. Suter, Mode seeking over permutations for rapid geometric model fitting, Pattern Recognit. 46 (1) (2013) 257–271.

[39] V.-H. Le, H. Vu, T.T. Nguyen, T.-L. Le, T.-H. Tran, Acquiring qualified samples for RANSAC using geometrical constraints, Pattern Recognit. Lett. 102 (2018) 58–66.

[40] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohenor, N.J. Mitra, GlobFit: consistently fitting primitives by discovering global relations, ACM Trans. Graphics 30 (4) (2011) 52.

[41] G. Xiao, H. Wang, T. Lai, D. Suter, Hypergraph modelling for geometric model fitting, Pattern Recognit. 60 (2016) 748–760.

[42] H. Wang, G. Xiao, Y. Yan, D. Suter, Searching for representative modes on hypergraphs for robust geometric model fitting, IEEE Trans. Pattern Anal. Mach. Intell. (2018), doi:10.1109/TPAMI.2018.2803173.

[43] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.

[44] P. Mukhopadhyay, B.B. Chaudhuri, A survey of Hough transform, Pattern Recognit. 48 (3) (2015) 993–1010.

[45] H. Isack, Y. Boykov, Energy-based geometric multi-model fitting, Int. J. Comput. Vision 97 (2) (2011) 123–147.

[46] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic Hough transform, Pattern Recognit. 24 (4) (1991) 303–316.

[47] L.A. Fernandes, M.M. Oliveira, Real-time line detection through an improved Hough transform voting scheme, Pattern Recognit. 41 (1) (2008) 299–314.

[48] Z. Xu, B.-S. Shin, R. Klette, Closed form line-segment extraction using the Hough transform, Pattern Recognit. 48 (12) (2015) 4012–4023.

[49] T.-C. Chen, K.-L. Chung, An efficient randomized algorithm for detecting circles, Comput. Vis. Image Und. 83 (2) (2001) 172–191.

[50] K.-L. Chung, Y.-H. Huang, S.-M. Shen, A.S. Krylov, D.V. Yurin, E.V. Semeikina, Efficient sampling strategy and refinement strategy for randomized circle detection, Pattern Recognit. 45 (1) (2012) 252–263.

[51] S.-C. Zhang, Z.-Q. Liu, A robust, real-time ellipse detector, Pattern Recognit. 38 (2) (2005) 273–287.

[52] W. Lu, J. Tan, Detection of incomplete ellipse in images with strong noise by iterative randomized Hough transform (IRHT), Pattern Recognit. 41 (4) (2008) 1268–1279.

[53] S. Chen, R. Xia, J. Zhao, Y. Chen, M. Hu, A hybrid method for ellipse detection in industrial images, Pattern Recognit. 68 (2017) 82–98.

[54] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1981) 111–122.

[55] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized hough transform (RHT), Pattern Recognit. Lett. 11 (5) (1990) 331–338.

[56] D. Walsh, A.E. Raftery, Accurate and efficient curve detection in images: the importance sampling Hough transform, Pattern Recognit. 35 (7) (2002) 1421–1431.

[57] F.A. Limberger, M.M. Oliveira, Real-time detection of planar regions in unorganized point clouds, Pattern Recognit. 48 (6) (2015) 2043–2053.

[58] E. Vera, D. Lucio, L.A.F. Fernandes, L. Velho, Hough transform for real-time plane detection in depth images, Pattern Recognit. Lett. 103 (2018) 8–15.

[59] P. Müller, P. Wonka, S. Haegler, A. Ulmer, L. Van Gool, Procedural modeling of buildings, ACM Trans. Graphics 25 (3) (2006) 614–623.

[60] O.J. Woodford, M.-T. Pham, A. Maki, F. Perbet, B. Stenger, Demisting the Hough transform for 3D shape recognition and registration, Int. J. Comput. Vision 106 (3) (2014) 332–341.

[61] P.J. Huber, Robust estimation of a location parameter, Ann. Math. Stat. 35 (1) (1964) 73–101.

[62] M.P. Dubuisson, A.K. Jain, A modified Hausdorff distance for object matching, Int. Conf. Pattern Recogn. 1 (1994) 566–568.

[63] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, SIGGRAPH Comput. Graphics 26 (2) (1992) 71–78.

[64] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level partition of unity implicits, ACM SIGGRAPH Courses (2005) 173–180.

[65] M. Alexa, J. Behr, D. Cohenor, S. Fleishman, D. Levin, C.T. Silva, Computing and rendering point set surfaces, IEEE Trans. Visual Comput. Graphics 9 (1) (2003) 3–15.

[66] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, T.R. Evans, Reconstruction and representation of 3D objects with radial basis functions, in: Annual Conference on Computer Graphics and Interactive Techniques, 2001, pp. 67–76.

[67] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Trans. Graphics 32 (3) (2013) 29.

[68] S. Fleishman, D. Cohenor, C.T. Silva, Robust moving least-squares fitting with sharp features, in: International Conference on Computer Graphics and Interactive Techniques, 24, 2005, pp. 544–552.

[69] A. Tagliasacchi, H. Zhang, D. Cohen-Or, Curve skeleton extraction from incomplete point cloud, ACM Trans. Graphics 28 (3) (2009) 71:1–71:9.

[70] K. Ellis, A. Solar-Lezama, J. Tenenbaum, Sampling for Bayesian program learning, in: Advances in Neural Information Processing Systems (NIPS), 2016, pp. 1289–1297.

[71] M. Muja, D.G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Trans. Pattern Anal. Mach. Intell. 36 (11) (2014) 2227–2240.

[72] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, M. Pollefeys, Semantic3D.net: A new large-scale point cloud classification benchmark, in: ISPRS Annals, 4, 2017, pp. 91–98.

[73] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[74] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, et al., A generative vision model that trains with high data efficiency and breaks text-based captchas, Science 358 (6368) (2017) eaag2612.

Zongliang Zhang received his B.E. and M.E. degrees in computer science and technology from Fuzhou University, China in 2010 and Northwest A & F University, China in 2013, respectively. He is currently a Ph.D. student in computer science at Xiamen University, China. His current research interests are 3D computer vision, 3D pattern recognition, and robust model fitting.

Jonathan Li received the Ph.D. degree in geomatics engineering from the University of Cape Town, South Africa. He is currently professor with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, China. He is also professor and head of the Mobile Sensing and Geodata Science Lab at the Faculty of Environment, University of Waterloo, Canada. His current research interests include information extraction from LiDAR point clouds and from earth observation images. He has co-authored more than 300 publications, over 150 of which were published in refereed journals including IEEE-TGRS, IEEE-TITS, IEEE-GRSL, IEEE-JSTARS, ISPRS-JPRS, IJRS, PE & RS and RSE. He is Chair of the ISPRS Working Group I/6 on LiDAR for Airborne and Spaceborne Sensing (2016–2020), Chair of the ICA Commission on Sensor-driven Mapping (2015–2019), and Associate Editor of IEEE-TITS and IEEE-JSTARS.

Yulan Guo received his B.E. and Ph.D. degrees from the National University of Defense Technology in 2008 and 2015, respectively. He is currently an assistant professor at NUDT. He was a visiting PhD student at the University of Western Australia from November 2011 to November 2014. He authored more than 20 peer reviewed journal and conference publications, including IEEE TPAMI, IJCV, PR and IEEE TMM and one book chapter. He served as a reviewer for more than 10 international journals and several conferences. His research interests include 3D feature extraction, 3D modeling, 3D object recognition, and 3D face recognition.

Xin Li is an associate professor at the School of Electrical Engineering & Computer Science (EECS), Louisiana State University (LSU), USA. He received the B.Sc. degree in computer science from the University of Science and Technology of China in 2003 and M.Sc. and Ph.D. degrees in computer science from the State University of New York at Stony Brook in 2005 and 2008, respectively. He joined the faculty of LSU in 2008. He is also an adjunct associate professor at Pennington Biomedical Research Center. He is currently leading the Geometric and Visual Computing (GVC) Group at LSU.

Yangbin Lin received the Ph.D. degree in computer science and technology from Xiamen University, China in 2016. He is currently an assistant professor at Jimei University, China. His current research interests are 3D computer vision and computer graphics.

Guobao Xiao received the Ph.D. degree in computer science and technology from Xiamen University, China, in 2016. He is currently a postdoc fellow at Xiamen University, China. His current research interests are robust statistics and model fitting.

Cheng Wang received the Ph.D. degree in information and communication engineering from the National University of Defense Technology in 2004. He is currently professor and associate dean of the School of Information Science and Engineering, and Executive Director of the Fujian Key Laboratory of Sensing and Computing for Smart Cities, Xiamen University, China. His research interests include point cloud processing, 3D modeling, and satellite image understanding. He has published more than 100 peer reviewed journal and conference papers.