Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient convex optimization-based texture mapping for large-scale 3D scene reconstruction

Xin Sheng [a], Jing Yuan [b], Wenbing Tao [a,c,*], Bo Tao [d], Liman Liu [e]

[a] *National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China*
[b] *School of Mathematics and Statistics, Xidian University, Xi'an 710126, China*
[c] *Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China*
[d] *State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China*
[e] *School of Biomedical Engineering, South-Central University for Nationalities, Wuhan 430074, China*

A R T I C L E I N F O

Article history:
Received 30 September 2019
Received in revised form 23 May 2020
Accepted 21 December 2020
Available online 29 December 2020

Keywords:
Image-based texture mapping
Multi-label problem
Convex optimization

A B S T R A C T

Texture mapping is a key step in large-scale 3D scene reconstruction, which can greatly enhance visual reality of the reconstructed scenes. However, existing techniques are unable to accomplish this task efficiently due to the high computational complexity of reconstructing large-scale real-world 3D scenes. In this work, we propose a new efficient convex optimization-based approach, i.e. the mesh-based continuous max-flow method, which can be easily implemented and accelerated upon a modern parallel computing platform, e.g. GPU. Particularly, an Markov Random Fields (MRF) based model, i.e. Potts model, is introduced to mathematically formulate the key specific view selection problem; we show that the challenging combinatorial optimization problem can be efficiently solved by resolving its convex relaxation, which recovers textures from images with the proposed duality-based continuous max-flow approach. In addition, visual effects of sharpness and deformation are utilized to define a criterion of evaluating texture quality effectively, and a large 3D triangular mesh is partitioned into structural components so as to reduce memory consumption of the proposed algorithm. The proposed mesh-based continuous max-flow approach for large-scale texture mapping demonstrates its outperformance over state-of-the-art methods, over large-scale public datasets, in both numerical efficiency and visual quality; meanwhile, our GPU-accelerated algorithm can yield 3D textured models with high quality from complex large-scale scenes in minutes.

1. Introduction

In the past few years, big progress has been made in large-scale image-based 3D reconstruction, where the geometric models of real-world sites such as 3D point clouds and 3D triangular meshes can be created over its given multiple views in hours with exiting techniques [1–6]. However, simply creating the geometric models is insufficient to high visual reality. In order to obtain photo-realistic 3D models, it is also necessary to recover textures from multiple views. Recent image-based

* Corresponding author at: National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China.

E-mail addresses: xinshenghust@hust.edu.cn (X. Sheng), jyuan@xidian.edu.cn (J. Yuan), wenbingtao@hust.edu.cn (W. Tao).

https://doi.org/10.1016/j.ins.2020.12.052
0020-0255/© 2020 Elsevier Inc. All rights reserved.

3D reconstruction techniques can recover textures by extracting corresponding texture patches from multiple views and mapping them onto the reconstructed 3D meshes. But the recovered texture patches may not be perfectly aligned on the reconstructed 3D meshes due to the unavoidable errors in both camera calibration and geometry reconstruction. Moreover varied luminance and scale between different views in large-scale datasets make it to recover photo-realistic textured models free of artifacts.Some studies proposed warping images to reduce the reconstruction errors [7–9], which can produce smooth textured models with impressive details on controlled RGB-D datasets. Unfortunately, there is no depth information for most large-scale datasets and the models generated by warping images may suffer from strong texture distortions and big geometric errors in large-scale reconstruction. Such methods are thus inappropriate for large-scale scenes. How to create textures with high visual quality for large-scale 3D scenes remains challenging.

1.1. Related works

Texture Mapping for Small-Scale Scenes: The methods of texture mapping for small-scale scenes focus on improving the visual quality of textured models. Eisemann et al. [10] estimated the optical flow between different views for warping texture patches, so as to suppress ghosting and misalignment. Zhou et al. [7] proposed to globally optimize camera poses via a non-rigid correction function to compensate for reconstruction errors, which produced superior textured results. Bi et al. [8] introduced a patch-match based method to warp all images and get collections with high photometric consistency, then a high quality textured model can be obtained with the optimized image collection. This method is suitable for texturing small objects with large texture misalignments. Fu et al. [9] proposed a global-to-local strategy to optimize camera poses, which could globally adjust texture patch positions and refine the texture boundaries, so as to produce textured models with high fidelity. Instead of correcting camera poses or optimizing photometric consistency of textures, Goldluecke et al. [11] suggested a super-resolution framework to enhance the texture mapping quality of 3D models using a primal–dual algorithm, where a displacement map was computed to minimize the projection errors. It showed the fidelity of 3D models with low resolution textures can be significantly improved by their proposed framework. Richard et al. [12] presented a novel deep-learning based super-resolution texture mapping method, which is capable of handling multi-view image super-resolution and texture redundancy jointly.

1.1.1. Texture mapping for large-scale scenes

The texture mapping methods for small-scale scenes can significantly enhance the visual quality of textured models, but most of them are either time-consuming or limited to small-scale datasets. In comparison, the methods of texture mapping for large-scale scenes are often trade off between visual quality and numerical efficiency. Texture mapping methods for large-scale scenes mainly consist of two steps: 1. *view selection* in order to select different views to create a texture patch for each face. 2. *seam optimization* which is applied to eliminate visible seams among the borders of texture patches. They can be classified into two different categories according to their view selection strategies.

Blending view-based methods choose multiple views for each mesh face and blend these views to yield texture patches. Grammatikopoulos et al. [13–15] proposed to check the visibility of the faces in different views, then create the texture patch for each face by blending all available views with weights depending on the corresponding angles and resolutions. Callieri et al. [16] created a blending function with similar weight definition, which calculated blending color values in a per-vertex manner to reduce computational cost and render the whole mesh. Due to the errors of camera calibration and dynamic image variance, textures created by blending views methods are often blurry, especially when applying to large-scale datasets.

The methods of single view texture mapping assign each face a certain view and extract a texture patch from it, which can reach a balance between reconstruction accuracy and numerical efficiency in processing large-scale datasets for the given huge solution space of view selection; for instance, there are 100^{1000} view selection options in total for a dataset with 100 images and a mesh with 1000 faces. The selection of optimal view can be modeled as a multi-label problem, which is balanced with texture quality and discontinuity. With this respect, Lempitsky and Ivanov [17] formulated view selection by minimizing an Markov Random Field (MRF) energy function with graph-cuts [18] to handle texture fidelity and texture smoothness jointly. In their work, the data term of MRF energy function prefers selecting less oblique view for each face and the submodular smooth term penalizes adjacent faces for selecting different views. Gal et al. [19] followed the same idea but chose the integration of image gradients as the data term, which hence introduced additional effects such as texture sharpness and proximity into the optimization model and achieved better performance in reconstructing complex large-scale scenes. Based on this work, Waechter et al. [20] further proposed some seam optimization strategies as post-processing to handle color discontinuities and artifacts, and a mean-shift based method was introduced to reject unreconstructed occluders, such as pedestrians as textures; large-scale texture models with impressive details were finally obtained by the proposed approach in [20]. Even such graph optimization-based approaches performed well in practice, there still remain three difficult problems that limit the application of single view texture mapping in large-scale 3D reconstruction: first, given the inherent algorithmic mechanisms of such graph or combinatorial optimization methods, their computational and memory burdens are heavy and hard to be parallelized, especially for the irregular graph settings (see Fig. 1 for illustration), which makes view selection the major computational bottleneck of the whole texture mapping pipeline; second, the data terms may not evaluate the texture quality properly, resulting in the generation of distorted textures; finally, the mesh-

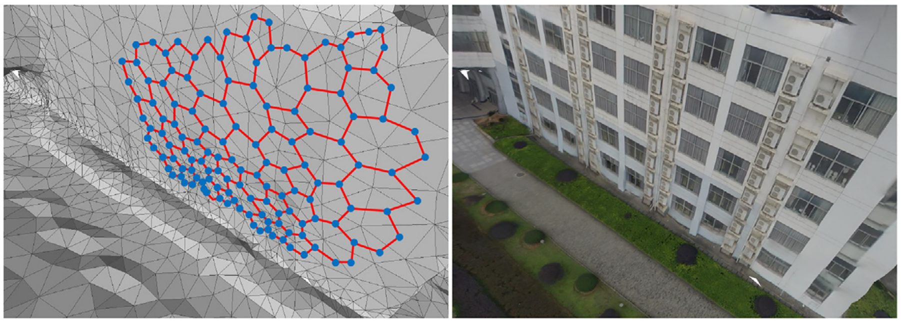

Fig. 1. *Left* shows an irregular mesh-based graph: the blue dots indicate graph nodes and the red lines indicate graph edges, the inherent algorithmic mechanism of the classical graph optimization algorithm makes it hard to be parallelized, eps. for such apparent irregular graph-based texture-mapping algorithms. *Right* demonstrates the computed textured model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

based graph of a large dataset is often too large to be stored in memory for computation, an effective mesh partition strategy should be therefore developed to divide the original large-scale optimization problem into multiple smaller ones which can be thus processed properly.

1.2. Convex optimization and contributions

Nowadays, convex optimization was developed as an efficient technique to solve the combinatorial optimization problems, particularly the submodular graph optimization problems [21–25]. Such convex optimization-based technique properly relaxes the discrete label-selection constraint on each graph node into a convex set, hence lead to a convex relaxation version of the formulated combinatorial optimization problem. The resulted convex optimization problem can be then solved efficiently with established algorithms [26], which could be greatly speeded up by parallel implementations. In particular, the continuous max-flow methods proposed by Yuan et al. [23,24] were developed under the perspective of primal and dual. Their methods served as one of the most successful and popular convex relaxation approach to the multi-labeling problems in the context of spatially continuous settings. These methods were also suitable for implementing on graphic processing units (GPUs), and obtained big success in efficiently solving large-scale 2D/3D image segmentation, e.g. medical image segmentation [27,28] etc.

In this work, we propose a new convex optimization-based approach to large-scale texture mapping for scene reconstruction, which extends such spatially continuous max-flow approach in the context of irregular graph settings of 3D mesh, and adapts the corresponding numerical configuration into the scenario of view selection; this boils down to a new efficient optimization solver to the associate combinatorial graph optimization problem. The new method inherits the parallelism [24] of the continuous max-flow approach spontaneously, and makes it particularly proper for high-performance multi-core parallel computing. In addition, we impose the texture deformation constraint on view selection to effectively alleviate texture distortion in the results of texture mapping, where the integration of deformation information constitutes a more accurate evaluation of textures from different views. Last but not the least, for large-scale 3D scene reconstructions, we introduce a fast mesh decomposition method to quickly divide the given mesh into structural components. The main directions of a mesh are discovered leveraging the histogram distribution of mesh face normals. And each face in a structural component is labeled to a certain direction based on its normal. Then view selection is conducted over each component independently, which make it possible for GPU to handle large dataset. In this study, we evaluate the proposed approach over different large-scale 3D datasets. Experiment results showed that our approach outperformed state-of-the-art works with higher fidelity textures and smaller reprojection errors, and enjoys big numerical advantages in terms of efficient parallel implementation as well.

In summary, our primary contributions are as follows:

- Extended from regular graph to 3D mesh irregular graph, an efficient convex optimization-based optimization approach is proposed to get view selection results in a parallel manner so that the optimization of the energy function can be significantly accelerated by GPU based parallel implementation.
- A data term built on texture deformation is proposed to suppress distorted textures in large-scale textured models.
- We propose a fast mesh decomposition algorithm with face normals, which extends the scalability of our texture mapping approach.

2. Convex optimization-based method

In this work, we aim to develop a new convex optimization approach to texture mapping upon a modern primal–dual framework, namely the *continuous max-flow method*, which has great capability to handle the complex large-scale 3D scenes.

In addition, its numerical scheme can be readily implemented in parallel and accelerated on the modern parallel computing platforms. In this section, we first introduce the mathematical model, i.e. Potts model, for view selection proposed by Waechter et al. [20]; then present the basic theory of continuous max-flow methods, which essentially formulates the given convex optimization problem with its equivalent dual model, in the form of linear equality constrained convex optimization, and employ the alternating direction method of multipliers (ADMM) to establish the corresponding fast algorithmic solver.

2.1. Potts model for view selection

Our work follows a similar mathematical model as Waechter et al.'s [20]: given n calibrated images $I = \{I_1, \ldots, I_n\}$ and a 3D mesh consisted of m triangular faces $V = \{v_1, \ldots, v_m\}$, the problem of view selection can be solved as a multi-labeling problem with the label set $\mathcal{L} = \{l_1, \ldots, l_n\}$, for which each node (face) $v_i, i = 1 \ldots m$, is assigned by one label $l_{v_i} \in \mathcal{L}$ representing the image where texture chosen from. Particularly, Waetchter et al. [20] introduced a graph $G(V, E)$ as shown in Fig. 1, where each node v_j corresponds to a mesh face and each edge e_{jk}indicates adjacency between nodes $v_j$and v_k. Then Waechter et al. [20] proposed to minimize the following MRF potential function defined over the graph G:

$$E(L) = \sum_{v_i \in V} E_{\text{data}}(v_i, l_{v_i}) + \sum_{e_{jk} \in E} E_{\text{smooth}}(e_{jk}, l_j, l_k), \tag{1}$$

where the unary terms are defined over the graph nodes $v_i, i = 1 \ldots m$; and the pairwise terms are defined over graph edges $e_{jk}, j, k = 1 \ldots m$. In addition, each unary term of $E_{\text{data}}(v_i, l_{v_i})$ prefers to select a view with better texture quality, which is measured by $E_{\text{data}}(v_i, l_{v_i}) = f(v_i, l_{v_i})$; each pairwise term E_{smooth} prefers a view selection that can minimize texture difference between two nodes (faces), for which $E_{\text{smooth}} = [l_j \neq l_k]$($[\cdot]$ is the Iverson bracket) to indicate the discontinuity across texture borders. Therefore, the minimization of the energy function (1) can be reformulated exactly as the Potts model over the mesh-based graph $G(V, E)$, which aims to partition the graph G into n subgraphs $G_i, i = 1 \ldots n$, by minimizing the following energy:

$$\min_{\{G_i\}_{i=1}^n} E(L) := \sum_{i=1}^n \sum_{v \in G_i} f(v, l_v) + \alpha \sum_{i=1}^n |\partial G_i|, \tag{2}$$

subject to

$$cup_{i=1}^n G_i = G(V, E), \quad G_j \cap G_k = \emptyset, \quad \forall j \neq k,$$

where $|\partial G_i|$ denotes the total number, i.e. the 'length', of edges along the borders of G_i, and $f(v, l_v)$ measures the corresponding texture quality at the graph node v when the image $I_{l_v} \in I$ is assigned to the face v. The minimization of the energy function (2) results in an optimum labeling $l_v^* \in \mathcal{L}$ to each node $v \in G$, that partitions the mesh graph G into n subgraphs $G_i, i = 1 \ldots n$, with better overall texture qualities and compact borders.

Waechter et al. [20] proposed using the alpha-expansion algorithm of graph cuts [18] as an efficient solver to minimize the MRF energy (1). However, given the the inherent algorithmic mechanisms of the classical graph cuts algorithms, it is hard to parallelize the corresponding graph optimization algorithms fundamentally, especially for irregular graph settings. This makes solving the Potts model of view selection a major computational bottleneck of the whole texture mapping pipeline. Consequently, we introduce a convex relaxation approach to the Potts model in this study. And we propose a new fast duality-based graph optimization algorithm, upon the modern optimization theory of continuous max-flow proposed by Yuan et al. [23,24], in the context of irregular discrete graph setting $G(V, E)$.

2.2. Continuous max-flow method to Potts model

During the past decade, convex relaxation/optimization was studied as the fundamental way to develop efficient algorithms to many graph-based combinatorial optimization problems, esp. Potts model. As a special case of the multi-labeling problem in computer vision, Potts model aims to partition a region Ω into multiple subregions with high data fidelity and tight boundaries, particularly in the spatially continuous settings investigated recently [24,25,29]:

$$\min_{\{\Omega_i\}_{i=1}^n} \sum_{i=1}^n \int_{\Omega_i} f(x, l_{v_i})\, dx + \alpha \sum_{i=1}^n |\partial \Omega_i|, \tag{3}$$

$$\text{s.t.} \bigcup_{i=1}^n \Omega_i = \Omega, \quad \Omega_j \cap \Omega_k = \emptyset, \quad \forall j \neq k.$$

where α is a weight factor between data fidelity and boundary length.

In the spatially continuous setting, given the labeling function $u_i(x) = \{0, 1\}, i = 1 \ldots n$, for each pixel $x \in \Omega$, where $u_i(x) = 1$ means the pixel x is within the subregion Ω_i but not otherwise, (3) then leads to the minimization of the following energy function:

$$\min_u \sum_{i=1}^n \int_\Omega u_i(x) f(x, l_{v_i})\, dx + \alpha \sum_{i=1}^n \int_\Omega |\nabla u_i|\, dx \tag{4}$$

subject to

$$\sum_{i=1}^n u_i(x) = 1, \quad u_i(x) \in \{0,1\}, \quad i = 1 \ldots n, \quad \forall x \in \Omega, \tag{5}$$

where the total-variation function $\int_\Omega |\nabla u_i|\, dx, i = 1 \ldots n$, evaluates the total length $|\partial\Omega_i|$ of the subregion Ω_i. Potts model seeks the optimum labeling function $u_i^*(x), i = 1 \ldots n$, to the combinatorial optimization problem (4), which defines the optimum partitions Ω_i such that $u_i^*(x) = 1$ for any $x \in \Omega_i$. Clearly, the linear equality constraint $u_1(x) + \ldots + u_n(x) = 1$ states that each pixel x only belongs to a single partition.

When each binary constraint $u_i(x) \in \{0,1\}$ in (5) is relaxed to the convex set $u_i(x) \in [0,1]$, one can rewrite the following convex relaxed optimization problem of Potts model:

$$\min_{u \in S} \sum_{i=1}^n \int_\Omega u_i(x) f(x, l_{v_i})\, dx + \alpha \sum_{i=1}^n \int_\Omega |\nabla u_i|\, dx \tag{6}$$

where

$$S = \{u(x) \,|\, (u_1(x), \ldots, u_n(x)) \in \triangle_n^+, \ \forall x \in \Omega\}. \tag{7}$$

$\triangle_n^+$ is the simplex set in the space $\mathbb{R}^n$.

Yuan et al. [24] shows that the convex optimization problem (6) can be solved efficiently under the perspective of primal and dual: first, one can formulate the equivalent dual model for (6), i.e. the *continuous max-flow model*, such that

$$\max_{p_s, p, q} \int_\Omega p_s\, dx \tag{8}$$

subject to

$$|q_i(x)| \leqslant \alpha, \quad p_i(x) \leqslant f(x, l_{v_i}), \quad i = 1 \ldots n; \tag{9}$$

$$(\mathrm{div} q_i - p_s + p_i)(x) = 0, \quad i = 1 \ldots n. \tag{10}$$

The continuous max-flow problem (8) is actually a linear equality (10) constrained optimization problem, which can be solved with high numerical performance by the augmented Lagrangian method (ALM), upon the corresponding augmented Lagrangian function:

$$L_c(u, p_s, p, q) = \int_\Omega p_s\, dx + \sum_{i=1}^n \langle u_i, p_i - p_s + \mathrm{div} q_i \rangle - \frac{c}{2} \sum_{i=1}^n ||p_i - p_s + \mathrm{div} q_i||^2 \tag{11}$$

Many recent studies showed that such ALM-based continuous max-flow algorithms outperformed state-of-the-art approaches in terms of numerical efficiency and great capabilities of incorporating prior constraints, especially for large-scale image segmentation problems. In addition, the proposed continuous max-flow algorithms can be easily accelerated with a natural parallel numerical implementation on the popular GPUs, e.g. nVidia cuda.

The introduced theory and algorithm of continuous max-flow [24] essentially paves the way to build up the efficient solver for the studied Potts model-based 3D scene reconstruction over the irregular graph $G(V, E)$, esp. for processing large-scale datasets.

3. Efficient parallel texture mapping method

In this study, we aim to solve the associate Potts model (2) of 3D texture mapping and introduce the new continuous max-flow approach in the specified settings of irregular mesh graphs. The continuous max-flow algorithm was originally proposed to solve the problem of image segmentation as the spatially continuous Potts model, thus introduced and implemented in the regular graph of image grid [24]. For the new mesh-based continuous max-flow method on the given irregular mesh graph, the combinatorial optimization problem (2) of Potts model is first reformulated to its convex relaxation version which is then described by its equivalent mathematical models in the sense of primal and dual; in addition, we apply its dual formulation, namely the mesh-based continuous max-flow model, and introduce its associate augmented Lagrangian method-based algorithm which maximizes all the flow variables while updating the label variables iteratively. The new mesh-based continuous max-flow algorithm is capable of being parallelized to reach a high numerical performance. For a large-scale scene, the size of the valid label set $\mathcal{L}_v$ for each face $v \in V$ is much smaller than the full label set $\mathcal{L}$, since a face can only be observed in few image views, which means an extremely extra computation and memory load to process a full label set. To avoid such extra computation, we propose an additional procedure to decompose the provided large mesh graph into multiple small structural meshes with proper sizes.

3.1. Texture quality term

As shown in [20], the texture quality term plays an important role in the mathematical model, which measures the quality of textures from different views. Similar as [19], the authors in [20] introduced $-\int_{\phi(v_j,l_j)} |\nabla I(p)|dp$ for the evaluation of texture quality, which is the gradient magnitude integration of the corresponding projected region of face v_j in image space. This term takes view proximity, angle and image resolution into account. However, it fails to consider the factor of texture deformation that is vital to visual texture quality. For large-scale 3D datasets, this introduces unavoidable errors into the procedure of geometric reconstruction; for example, most straight structures like windows and pipes in images can not be accurately reconstructed from existing techniques [30,6]. When projected into an available image l_j, a triangular face v_j in the mesh corresponds to a 2D triangle T_{l_j} (see Fig. 2 for demonstration). Clearly, if the two triangles differ greatly, the corresponding texture patch extracted from the image view l_{v_i} will be stretched and twisted severely when mapped on the 3D mesh. To tackle this problem, we define a novel criterion to evaluate deformation of two triangular patches explicitly:

$$D(v_j,l_j) = \frac{1}{(1+|\delta(\theta)|+|\delta(\beta)|+|\delta(\gamma)|)^2}, \tag{12}$$

where $\delta(\cdot)$ computes the corresponding angle difference between v_j and T_{l_j} (see Fig. 2 for details), and $D(v_j,l_j)$ measures the shape difference between the specified two triangles. Lempitsky et al. [17] and Callieri et al. [16] introduced the view angle to avoid selecting extremely oblique views, which can evaluate the distorted texture patches to some extent. However, the integration of such view angle information may not perform well in complex large-scale scenes, where the camera orientations and mesh face normals varied largely. As a toy example shown in Fig. 3, when a mesh face simply rotates along its normal, a series of 2D triangles on the mapping plane are obtained and vary largely in shape with the same view angle. The graph of Fig. 3b shows that the proposed deformation term can properly capture the shape difference, which punishes texture patches with strong distortion and, meanwhile, decreases the possibility of selecting them as final textures.

Now we introduce the new term of evaluating texture quality such that:

$$f_0(v_j,l_j) = G(v_j,l_j)D(v_j,l_j), f(v_j,l_j) = 1 - \text{Normalize}\{f_0(v_j,l_j)\}, \tag{13}$$

where $G(v_j,l_j)$ is the gradient magnitude integration over T_{l_j}:

$$G(v_j,l_j) = \int_{T_{l_j}} |\nabla I(p)|\,dp. \tag{14}$$

All texture quality terms are normalized to $[0,1]$ for further processing. It is obvious to see that the above proposed formulation measures texture deformation and sharpness jointly, which gives a more comprehensive evaluation of texture quality. Our experiments show that using such texture evaluation term in a complex scene with different view orientations can obtain textured models with well preserved details.

3.2. Mesh-based continuous max-flow method

Now we study the problem of partitioning the given mesh $G(V,E)$ into n labeled sub-meshes with both high texture quality and tight boundaries. Its corresponding Potts model (2) can be rewritten as follows [1]:

$$\begin{aligned} &\min_u \sum_{x\in V}\sum_{i\in\mathcal{L}(x)} f(x,i)u_i(x) + \alpha\sum_{x\in V}\sum_{i\in\mathcal{L}(x)} |\nabla u_i(x)|, \\ &s.t. \sum_{i=1}^{n} u_i(x) = 1, \quad u_i(x)\in\{0,1\},\ i=1\ldots n, \quad \forall x\in V, \end{aligned} \tag{15}$$

where $u_i(x)=1$ when $x\in G_i$, and vanishes otherwise; $\sum_{i\in\mathcal{L}(x)}|\nabla u_i(x)|$ presents the total 'length' $|\partial G_i|$ of the sub-mesh G_i. In this work, the gradient operator for each labeling function $u_i(x), i=1\ldots n$, is defined over the given discrete mesh G such that:

$$\begin{aligned} &\nabla u_i(x) = (\partial_1 u_i(x), \partial_2 u_i(x), \partial_3 u_i(x)), \\ &\partial_k u_i(x) = u_i(x) - u_i(x_k), \quad k=1,2,3. \end{aligned} \tag{16}$$

Note that x_k is the three neighbor nodes of x.

This model can be well approximated by its convex relaxation problem:

$$\min_{u\in\triangle_+} \sum_{x\in V}\sum_{i\in\mathcal{L}(x)} f(x,i)u_i(x) + \alpha\sum_{x\in V}\sum_{i\in\mathcal{L}(x)} |\nabla u_i(x)|, \tag{17}$$

[1] In this section, we use x to denote a node face for clarity

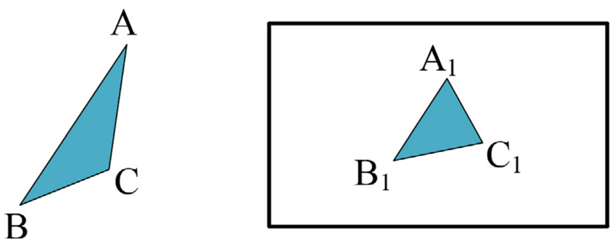

Fig. 2. $\delta(\theta) = \angle BAC - \angle B_1A_1C_1$, and $\delta(\beta)$ and $\delta(\gamma)$ are the differences of the other two pairs of angles. A triangle v_j (*Left*: $\triangle ABC$) in 3D space corresponds to a 2D texture triangle T_{I_j} (*Right*: $\triangle A_1B_1C_1$) in the view image I_j.

where the simplex constraint set $\triangle_+$ is

$$\triangle_+ = \left\{ u \in \mathbb{R}^n | \sum_{i=1}^{n} u_i(x) = 1; \quad u_i(x) \geqslant 0, \quad i = 1 \ldots n \right\}.$$

3.2.1. Mesh-based continuous max-flow model

In (17), its second total-variation-like term can be rewritten as:

$$\alpha \sum_{x\in V} \sum_{i\in\mathcal{L}(x)} |\nabla u_i(x)| = \max_{q_i(x)\in C_\alpha} - \sum_{x\in V} \sum_{i\in\mathcal{L}(x)} \nabla u_i(x) \cdot q_i(x) = \max_{q_i(x)\in C_\alpha} \sum_{x\in V} \sum_{i\in\mathcal{L}(x)} u_i(x) \mathrm{div} q_i(x). \tag{18}$$

where the divergence operator $\mathrm{div}q_i(x)$ over the discrete mesh G is computed by the associate flows $(q_i^1(x), q_i^2(x), q_i^3(x))$ along the three neighbor edges of x:

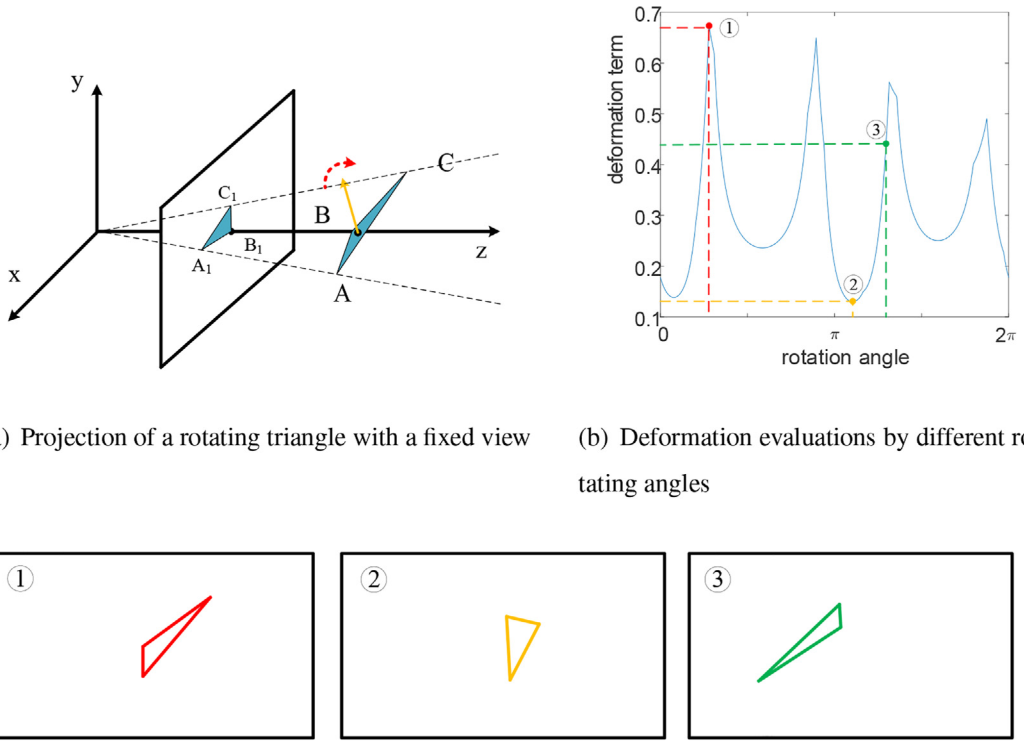

Fig. 3. (a) A triangle $\triangle ABC$ ($A(1,0,10)$, $B(0,0,11)$, $C(0,1,13)$) is projected to the plane $z = 5$ and we get a mapped triangle $\triangle A_1B_1C_1$ in this plane. (b) As the triangle rotates along its normal with the center fixed, we compute the deformation terms. We make a deformation plot to indicate the relationship between the rotation angle and deformation term. (c) We show the three triangles in the projection plane that correspond to the three point in the deformation plot.

$$\mathrm{div}q_i(x) = -(q_i^1(x) + q_i^2(x) + q_i^3(x)); \tag{19}$$

and the constraint set $q_i(x) \in C_\alpha$ means that

$$C_\alpha = \{q \,| |q_i^k(x)| \leqslant \alpha, \quad k = 1,2,3\}.$$

On the other hand, following similar analysis in [24], it is easy to see that

$$u_i(x)f(x,i) = \max_{p_i(x) \leqslant f(x,i)} u_i(x)p_i(x).$$

Therefore, the convex relaxed Potts model (17) can be equally rewritten as:

$$\min_u \max_{q_i,p_i,p_s} \sum_{x\in V}\sum_{i\in\mathcal{L}(x)} u_i(x)(p_i(x) + \mathrm{div}q_i(x)) + \sum_{x\in V} p_s(x)\left(1 - \sum_{i\in\mathcal{L}(x)} u_i(x)\right), \tag{20}$$

where, given the free variable $p_s(x)$, the second term forces $\sum_{i\in\mathcal{L}(x)} u_i(x) = 1$ for any node $x \in V$. Clearly, with helps of the new variables $p_s(x)$ and $p_i(x)$, the original constraints on the labeling function $u_i(x)$, i.e. $u(x) \in \triangle_+$, are dropped out, i.e. all $u_i(x)$ are free!

We organize the energy function of (20), then we have

$$\min_u \max_{q_i,p_i,p_s} \sum_{x\in V} p_s(x) + \sum_{x\in V}\sum_{i\in\mathcal{L}(x)} u_i(x)(\mathrm{div}q_i(x) - p_s(x) + p_i(x)), \tag{21}$$

Given the linear energy function of (21) and the min–max theorem [31], the min and max operators of (21) are interchanged. Hence, we first minimize the energy function of (21) over all the free variables $u_i(x)$, which results in the following equivalent maximization problem, namely the *mesh-based continuous max-flow problem*:

$$\max_{q_i,p_i,p_s} \sum_{x\in V} p_s(x) \tag{22}$$

subject to the linear equality constraints, i.e. the *flow conservation condition*:

$$\mathrm{div}q_i(x) - p_s(x) + p_i(x) = 0, \quad i \in \mathcal{L}(x), \tag{23}$$

and the constrained conditions for flows at each node $x \in V$:

$$q_i(x) \in C_\alpha; \quad p_i(x) \leqslant f(x,i), \;\; i \in \mathcal{L}(x); \quad q_i(x) \in C_\alpha. \tag{24}$$

Given the following equalities:

$$(17) \iff (20) \text{ and} (21) \iff (22),$$

the mesh-based continuous max-flow model (22) is thus equivalent or dual to the convex relaxed Potts model (17).

Additionally, in view of the min–max problem (21), it is easy to see that the optimum labeling functions $u_i(x), i \in \mathcal{L}$, to (17) are just working as the multipliers to the linear equality constraints, i.e. the flow conservation conditions, in the mesh-based continuous max-flow model (22). Upon this, the augmented Lagrangian method (ALM) provides an algorithmic framework to develop efficient solver, see Section 3.2.2 for details, which computes the labeling functions $u_i(x)$ and approximates the Potts model (2) jointly by maximizing (22) over all flow variables $(p_s, p_s(x), q_i(x))$!.

Particularly, the mesh-based continuous max-flow model (22) can be well explained by maximizing the free flow variables $p_s(x)$ which pass through the following mesh-networks, demonstrated in Fig. 4, under the constraints of (23) and (24):

1. n copies of the mesh-based graph $G(V,E)$ are given in parallel. In ith copy, nodes that can be observed in image I_i are classified as active nodes F_i, the other nodes are classified as inactive nodes;
2. For each active graph node $x \in V_i$, the source flow $p_s(x)$ tries to stream from the source node S to x. The source flow field is defined to be identical for the same active node at each copy;
3. For each active graph node $x \in V_i$, the sink flow $p_i(x), i \in \mathcal{L}(x)$, is directed from x at the ith copy to the sink node T;
4. The spatial flow fields q are defined within each copy. For each active node $x \in V$, we define spatial flow: $q_i(x) := \{q_1^i(x), q_2^i(x), q_3^i(x)\}, i \in \mathcal{L}(x)$. The three components are defined over edges link to x.

3.2.2. Mesh-based continuous max-flow algorithm

With the maximization problem (22) constrained by the linear equality constraints (23), the augmented Lagrangian method can be used to build up its efficient solver based on the following augmented Lagrangian function:

$$L(p_s,p,q,u) = \sum_{x\in V} p_s(x) + \sum_{x\in V}\sum_{i\in\mathcal{L}(x)} u_i(x)(\mathrm{div}q_i(x) - p_s(x) + p_i(x)) - \frac{c}{2}\sum_{x\in V}\sum_{i\in\mathcal{L}(x)} \|\mathrm{div}q_i(x) - p_s(x) + p_i(x)\|^2, \tag{25}$$

where c is the positive coefficient for the second-order quadratic penalty term.

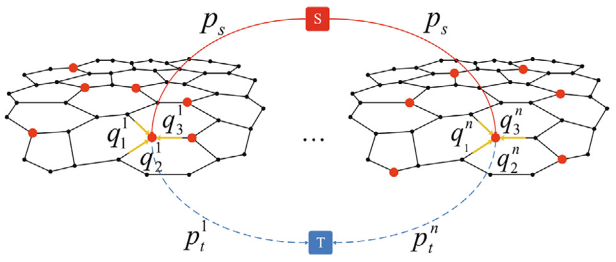

Fig. 4. Configuration of mesh-based continuous max-flow model with n labels. In each copy, the red dots indicates active nodes, and the black dots indicates inactive nodes.

Therefore, the mesh-based continuous max-flow algorithm, see Alg. 1 for details, can be constructed by modern convex optimization theory [26], which explores an efficient numerical structure of alternating direction method of multipliers (ADMM).

Algorithm 1: Mesh-based continuous max-flow algorithm

```
Input: The mesh-based graph $G(V,E)$. The texture quality term $f$.
Output: The labeling $L$.
Initialize all the variables $(p_s, p, q, u)$, set iterator $k = 0$;
repeat
1: $k = 500$ or $e^k < 10^{-5}$1:Keep other variables fixed, update spatial flows $q_i(x)$, $i \in \mathcal{L}(x)$by:

    $q_i^{k+1} = \Pi_\alpha(q_i^k + h\nabla(\text{div}q_i^k - p_s^k + p_i^k)),$

where $h$ is a step size and $\Pi_\alpha$ is a projection operator:

    $\Pi_\alpha(g(x)) = \begin{cases} g(x), & \textit{if } |g(x)| \leqslant \alpha \\ \alpha\text{sgn}(g(x)), & \textit{if } |g(x)| > \alpha \end{cases}$

2: Keep other variables fixed, update sink flows $p_i$, $i \in \mathcal{L}(x)$ by:

    $p_i^{k+1} = p_s^k - \text{div}q_i^{k+1} + \frac{u_i^k}{c}$

3: Keep other variables fixed, update source flows $p_s$ by:

    $p_s^{k+1} = \frac{1}{c|\mathcal{L}(x)|} + \frac{1}{|\mathcal{L}(x)|}\sum_{i\in\mathcal{L}(x)}\left(\text{div}q_i^{k+1} + p_i^{k+1} - \frac{u_i^k}{c}\right)$

where $|\mathcal{L}(x)|$is the size of the label set $\mathcal{L}(x)$.
4: update multiplier $u_i, i \in \mathcal{L}(x)$ by:

    $u_i^{k+1} = u_i^k - c(\text{div}q_i^{k+1} - p_s^{k+1} + p_i^{k+1})$

5: compute average error of $u$ by:

    $e^{k+1} = \frac{\sum_{x\in V}\sum_{i\in\mathcal{L}(x)}|u_i^{k+1}(x) - u_i^k(x)|}{\sum_{x\in V}|\mathcal{L}(x)|}$

6: extract the final labeling $L$: $l(x) = \arg\max_i u_i(x), i \in \mathcal{L}(x)$
```

Obviously, the update of each flow variable of $(p_s(x), p_i(x), q_i(x))$ at each iteration of Alg. 1 is independent for each graph node and edge despite the irregular mesh graph structure, which can thus be computed edgewise and nodewise in parallel. This provides a great basis to implement the algorithmic steps on modern parallel computing platforms like GPUs or HPCs, to significantly improve its numerical performance. The implementation of the proposed mesh-based continuous max-flow can skip the optimization for all the invalid labels, which further lowers the computation cost effectively.

3.2.3. Analysis on computational complexity and scalability

Provided the mesh graph $G(V, E)$, where $n = |V|$ and $e = |E|$, and the total number of active labels $K = |\mathcal{L}|$, at each iteration of Alg. 1, the complexity of computing the flow residues $\mathrm{div} q_i^k - p_s^k + p_i^k$ at each node is $O(nK)$, the complexity of updating the flow variables q_i^{k+1} at each edge is $O(eK)$, the complexity of computing p_s^{k+1} , p_i^{k+1} and u_i^{k+1} are all $O(nK)$. Thus, the complexity of each iteration of Alg. 1 is $O(4nK + eK)$. In addition, the total iterations of the proposed ALM-based algorithm for convergence are about $O(1/\epsilon)$, where ϵ is the error between solution and global optimal solution. In conclusion, the total complexity of Alg. 1 is $O(\frac{1}{\epsilon}(4nK + eK))$.

3.3. Mesh decomposition

As shown in Alg. 1, the proposed mesh-based continuous max-flow algorithm is an iterative optimization method to process the given mesh information in the whole sense. For datasets with large meshes and thousands of image views, a single GPU may not be capable to conduct view selection over the whole mesh, with the mesh-based continuous max-flow algorithm, due to its limited embedded GPU memory; also, the proposed mesh-based continuous max-flow algorithm is not suitable for the implementation on multi-GPUs without special developments. Hence, to deal with a huge mesh with a big amount of image views, we try to partition the given mesh into multiple structural components with a reasonable size: first, we classify all the mesh face normals as a spherical coordinate system with 64×64 bins (see Fig. 5 (left)); then we count the number of face normals in each bin and create their corresponding distribution histogram (see Fig. 5 (right)); for each bin, we calculate the rate of face normals that belong to the bin, and keep bins with the rate bigger than δ_p as candidates; finally, we extract the central direction of each candidate bin, and get p main directions of the large meshes by merging the similar directions. With this respect, for the given p central directions $\left\{\vec{dir(1)}, \ldots, \vec{dir(p)}\right\}$ and the provided mesh graph $G(V, E)$, the mesh decomposition procedure can be solved as a multi-labeling problem with p labels, which assigns a central direction $\vec{dir(i)}, i = 1 \ldots p$, to each node $x \in V$ as the following data term:

$$f_p(x, i) = 1 - \left\langle \vec{n}(x), \vec{dir(i)} \right\rangle, \quad i = 1, 2, \ldots, p, \tag{26}$$

where $\vec{n}(x)$ presents the normal direction of the node face x. We decompose the given mesh by the solving the discrete Potts model with the above defined data term. For some region with big variance of normal directions, the partition tends to be trivial; thus, we merge the small mesh components to avoid over-segmentation, and the large mesh can then be approximately segmented into multiple structural mesh components. If there are still some big mesh components not suitable for processing, we further partition them into smaller pieces, so as to make sure that each component is manageable for computation.

4. Experiments

We conduct experiments over four different datasets of *Citywall* [32], *QM*, *D-9* and *OVG* (see Table 1 for detailed information), with different sizes and complexities with the proposed approach. The introduced mesh-based continuous max-flow algorithm is implemented both on CPU (with C++ and OpenMP) and on GPU (with nVidia CUDA), based on the proposed mesh partition method [2]. All the experiments were performed over a computer with Xeon E5-2630v2 2.4 GHz, 256 GB of RAM and four NVIDIA GeForce GTX 1080Ti GPUs.

4.1. Texture quality

Waechter et al.'s [20] texture quality term is used, which computes the gradient magnitude integration over the projection region for each face, to texture the scenes of *Citywall* and *D-9* with the codes provided in [20]. And we replace Waechter et al.'s [20] with the proposed term and get the textured results. In order to show the impact of different terms, we texture these models without using any seam leveling techniques. In Fig. 7, Waechter et al.'s [20] quality term selects some texture patches with large gradient magnitudes, even the textured patches are severely distorted. Further, we texture the models by replacing the deformation term with a viewing angle based term (the cosine of viewing angle). Compared with Waechter et al.'s results, our term can effectively suppress the generation of distorted textures. Imposing the viewing angle constraint on Waechter et al.'s term can avoid selecting texture patch from a highly oblique view. But in a larger site, our texture quality

[2] We plan to release the source code soon.

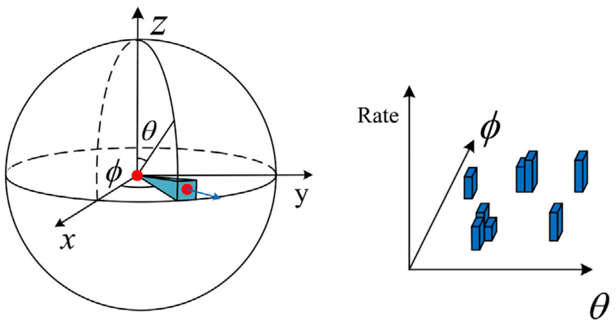

Fig. 5. *Left*: Divide all face normals into 64 × 64 bins based on angle θ and ϕ. One of the bins is marked with light blue. *Right*: The normal histogram (For clarification, we only draw part of the bins). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

term still yields a models with better visual quality since the viewing angle cannot truly indicates the texture deformation. In order to give a quantitative evaluation of the reconstruction error of our texture quality term, we use the Virtual Rephotography approach[32] to compute the 1-NCC and DSSIM error scores between the input images and model rendering photos from the same view point, where DSSIM is related to SSIM [33] via DSSIM $= \frac{1-\text{SSIM}}{2}$. We show the distribution of the scores in Fig. 6 and give the average NCC and DSSIM error in Table 2. Since the dataset *Citywall* is captured with a handheld camera, the performance of Waechter's term and our texture quality term is similar. While in more complicated scenes like *D-9*, with the proposed texture quality term, our textured models gets better visual quality with less reconstruction error.

4.2. Mesh-based continuous max-flow algorithm

In Section 3, it is demonstrated that we can get a approximately optimal view selection results to by solving the discrete Potts model with our mesh-based continuous max-flow algorithm. In the studied Potts model of this work, α controls the bias between texture quality and the total perimeter (smoothness) of all the mesh partitions. The parameter c is introduced in the augmented Lagrangian function 25 as the coefficient of its second-order quadratic function of the linear equality term, which actually conducts as the enforcement of the given linear equality constraint. The parameter h is introduced to the step of updating spatial flows in Alg. 1, which works as the step-size of the gradient-descent optimization. In order to determine the optimal parameters, we conduct experiments over the datasets *Citywall* and *QM*. We set $h = 0.2,\ c = 0.2$ and take different α ranging from 0.01 to 3. Then we set $\alpha = 0.5,\ h = 0.2$. And we take different c ranging from 0.001 to 1. At last, we set $\alpha = 0.5, c = 0.5$ and take different h ranging from 0.01 to 0.5. The experiment results are given in Fig. 8. As shown in Fig. 8, the algorithm can converge quickly with low energy if we set $\alpha = 0.5$. Meanwhile the algorithm converges faster with a smaller c. However the algorithm will get poor results if c is too small. Moreover, the algorithm will be hard to converge with an overlarge gradient step h. And if h is too small, the algorithm will converge slowly. To reach the balance between convergence speed and optimization effect, we set $\alpha = 0.5,\ c = 0.3$ and $h = 0.2$ in our implementation.

In order to validate the effectiveness of the proposed optimization algorithm, we texture all the four datasets with GCO solver (alpha expansion) [18], the parallel MRF MAP solver MAPMAP proposed by Thuerck et al. [34] and our mesh-based CMF using the proposed texture quality term. The codes of GCO and MAPMAP are downloaded from the authors' homepage. The final energies after the optimization is given in Table 3. The energies of our approach is slightly higher than the alpha-expansion graph cuts since in CMF the original non-convex problem is relaxed to be a convex problem. The performance of energy optimization is very close between MAPMAP and our mesh-based CMF in the medium size datasets *Citywall* and *QM*. When applied to the larger datasets *D-9* and *OVG*, our methods get solutions with lower energies than MAPMAP.

4.3. Mesh decomposition

For large large-scale datasets, we first find the main directions of the meshes as illustrated in Fig. 5. Using these main directions, we partition the large meshes into structural components with our mesh-based continuous max-flow algorithm.

Table 1
Properties of the four large-scale datasets.

Datasets	Views	Faces	Image resolution	Device
Citywall	560	1.3 million	2000 × 1500	Camera
QM	682	1.7million	1500 × 1125	UAV
D-9	596	7.7 million	3971 × 2978	UAV
OVG	858	11.2 million	3969 × 2976	UAV

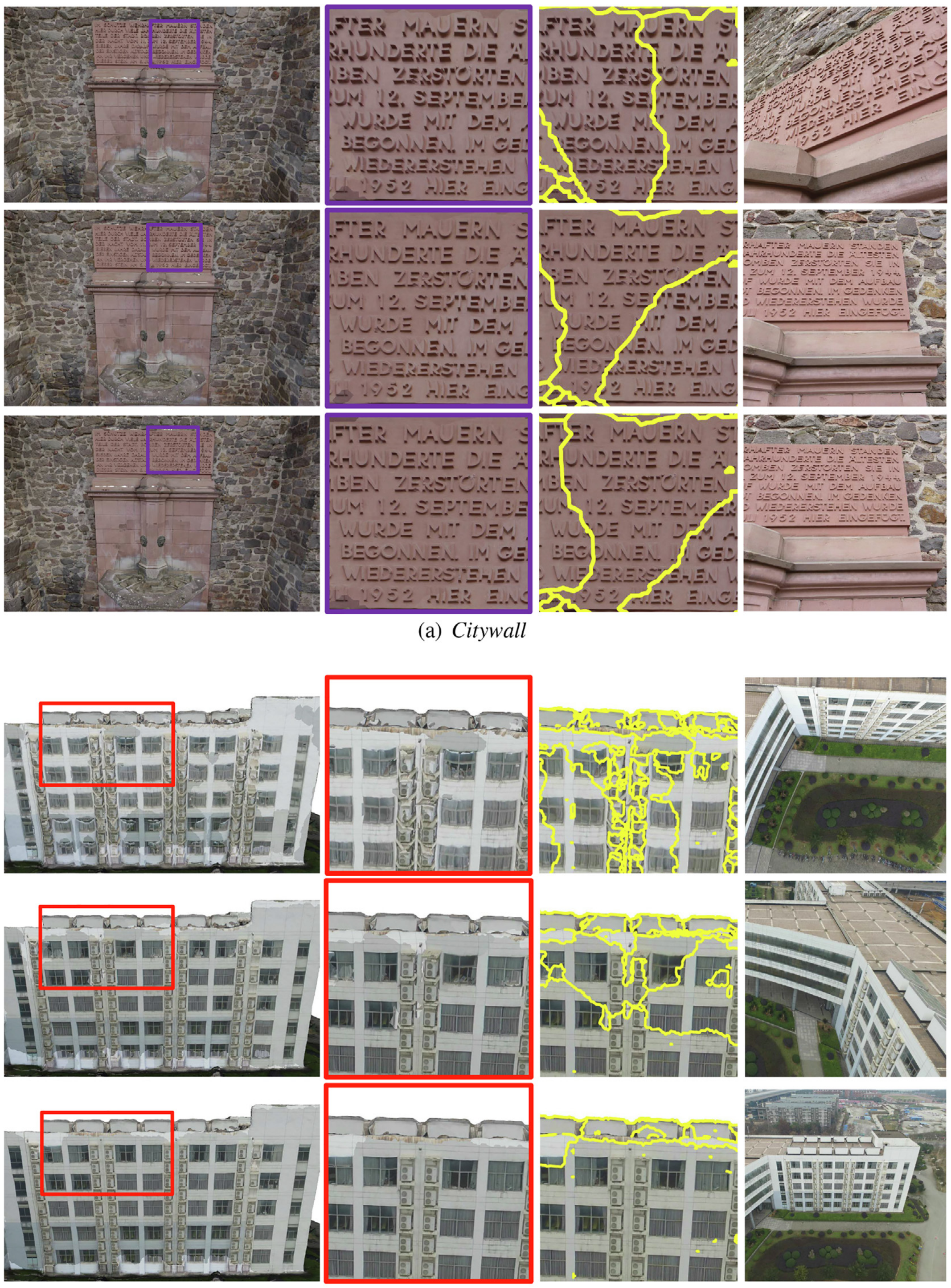

Fig. 7. Comparison with [20] on texture quality effect. From *Top* to *Bottom*: Models textured with Waechter's texture quality term, models textured with Waechter's term plus viewing angle and models textured with our texture quality term, and repeated this order. From *Left* to *Right*: Textured models, texture details, texture patch borders (marked with the yellow lines) and the source images. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The algorithm parameters are the same as mentioned in Section 4.2. The number of main directions are limited, making our mesh-based CMF algorithm fast to converge (see Table 4 for mesh partition time). As a consequence, the partition approach

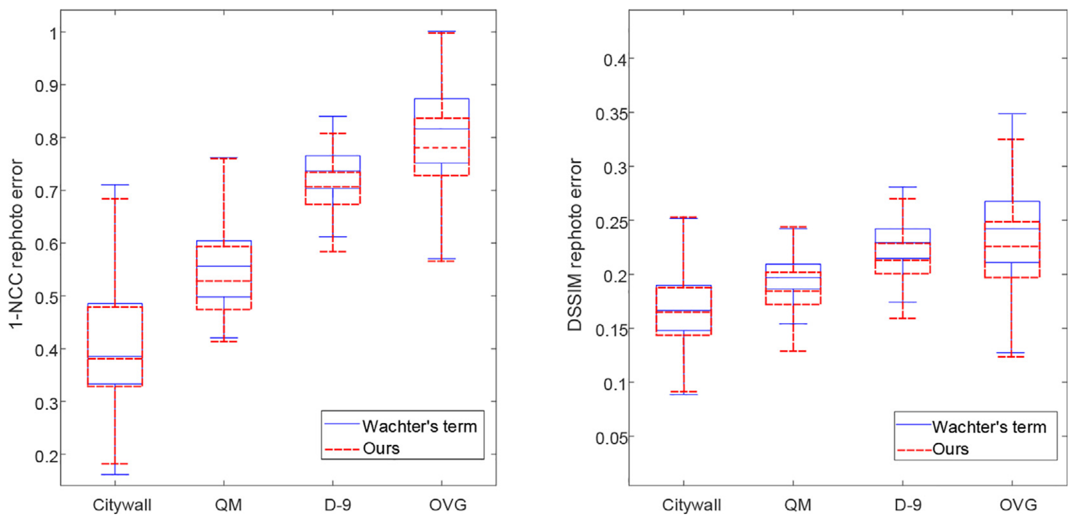

Fig. 6. Quantitative comparison of the textured models obtained with different texture quality terms. Boxplots show minimum, lower quartile, median, upper quartile, and maximum of 1-NCC rephoto errors and DSSIM errors.

Table 2
The comparison of average NCC error between the two texture quality terms. (Lower is better.)

datasets	Average NCC		Average DSSIM	
	Waechter et al.'s	Ours	Waechter et al.'s	Ours
Citywall	0.4093	**0.4039**	0.1751	**0.1727**
QM	0.5571	**0.5385**	0.1992	**0.1889**
D-9	0.7290	**0.7015**	0.2276	**0.2140**
OVG	0.8050	**0.7789**	0.2362	**0.2213**

has no relevant negative impact on performance. In Fig. 9 we show the partition results of the four large datasets. Our method can yield a high quality partition result where structures like wall and doors can be segmented into individual components. For component with over 2 million faces, we further partition it into smaller components based on its bounding box. For example, in Fig. 9(c) and (d), the ground of meshes are segmented into manageable parts. We conduct the mesh-based CMF over the structural components of the *Citywall* and *QM*, and make comparison with the energies optimized by CMF in the whole meshes. As shown in Table 3, using CMF with mesh partition, the final energy differences against mesh-based CMF algorithm without partition are very small.

4.4. Running times

We run Waechter et al.'s [20] Texrecon code with GCO, MAPMAP and our mesh-based CMF on 32 CPU threads. Then we test our GPU code with one GPU and one CPU thread (The parameters of our algorithm are given in 4.2 and 4.3). As shown in Table 4, compared with [20], our approach can achieve four to six times speedup ratio over the four datasets with one GPU, despite that the code of [20] is run with 32 threads. In the large dataset *OVG*, the MAPMAP achieves 1.4$\times$ sppedup while the mesh-based GPU CMF keeps a speedup ratio higher than 6.

We also test our approach with four GPUs and four CPU threads. Since the meshes have been segmented into components with our algorithm, energy function over different components can be optimized in multiple GPUs. With the obtained results, we adopt Waechter et al.'s [20] seam leveling techniques to eliminate visible seams and get the textured models. In Fig. 10 all the models are textured with our textured quality term and optimized with our GPU version mesh-based CMF. Utilizing the high parallel computing ability of GPU, our texture mapping approach can generate a high quality large-scale textured model in a short time.

5. Conclusion

Texture mapping is the key in large-scale image-based scene reconstruction to enhance visual reality, which is, however, the computational bottleneck due to its huge computational complexity. Most existing techniques are inefficient in numerics and hard to be paralleled, hence limited for their applications to the task of large-scale texture mapping. In this work, we propose a new convex optimization approach, namely the mesh-based continuous max-flow method, which provides an effi-

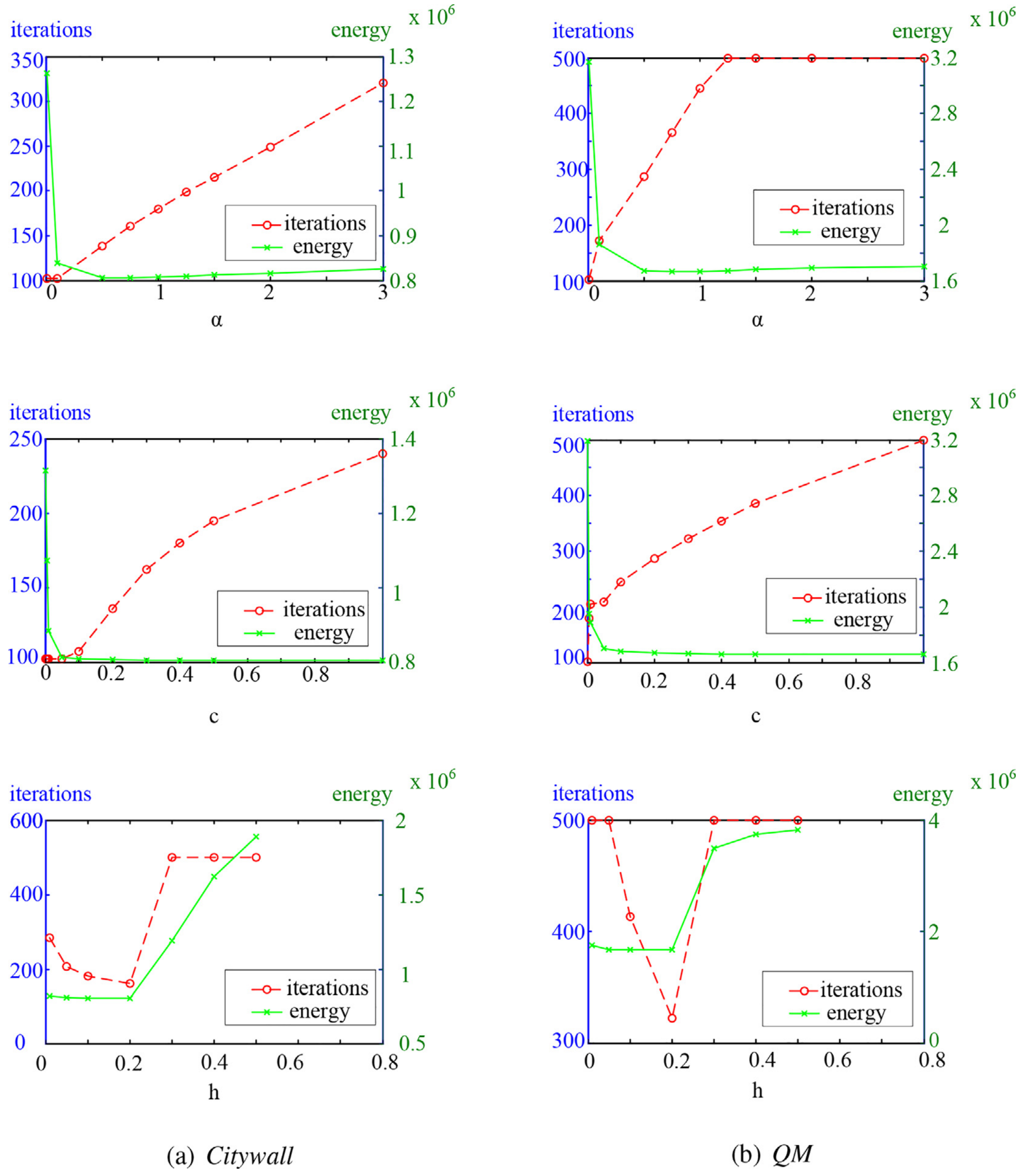

Fig. 8. Experiments for parameter adjustment. The iteration numbers and the energies of converged results are plotted. (The result is better with a smaller energy value.)

Table 3
The final energies of different optimization methods.

Datasets	GCO	MAPMAP	CMF(CPU)
Citywall	825331	831351	827142
QM	1623250	1657980	1659128
D-9	8217290	8324840	8307329
OVG	6141220	6251670	6208813

cient approximation solver to the introduced MRF-based model, i.e. Potts model, of the specific view selection problem. The proposed mesh-based continuous max-flow algorithm can be further implemented in parallel to significantly speedup its numerical performance. Moreover, visual effects of sharpness and deformation are utilized to define a criterion of evaluating

Table 4
The running time statistic of our mesh-based CMF algorithm against GCO and MAPMAP. The numbers in bracket are the accelerative ratios of optimization speed against GCO.

Time(s)		*Citywall*	*QM*	*D-9*	*OVG*
GCO		134.1	553.5	1322.7	2441.9
MAPMAP		30.5(**4.40**)	109.2(**5.07**)	708.1(**1.87**)	1749.1(**1.40**)
Mesh partition		2.4	8.7	29.1	30.5
CMF	cpu	**35.2(3.81)**	**246.1(2.24)**	**645.2(2.05)**	**1021.7(2.39)**
	One GPU	**19.1(7.02)**	**70.4(7.86)**	**208.3(6.35)**	**376.8(6.48)**
	Four GPUs	**6.74(19.89)**	**21.5(25.71)**	**61.4(21.54)**	**106.1(23.01)**

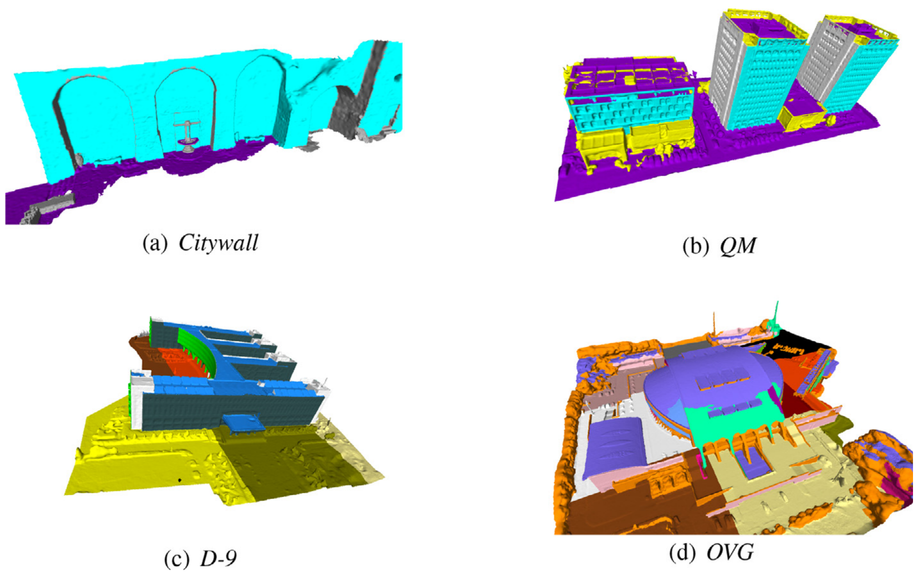

Fig. 9. Mesh partition results over large-scale datasets.

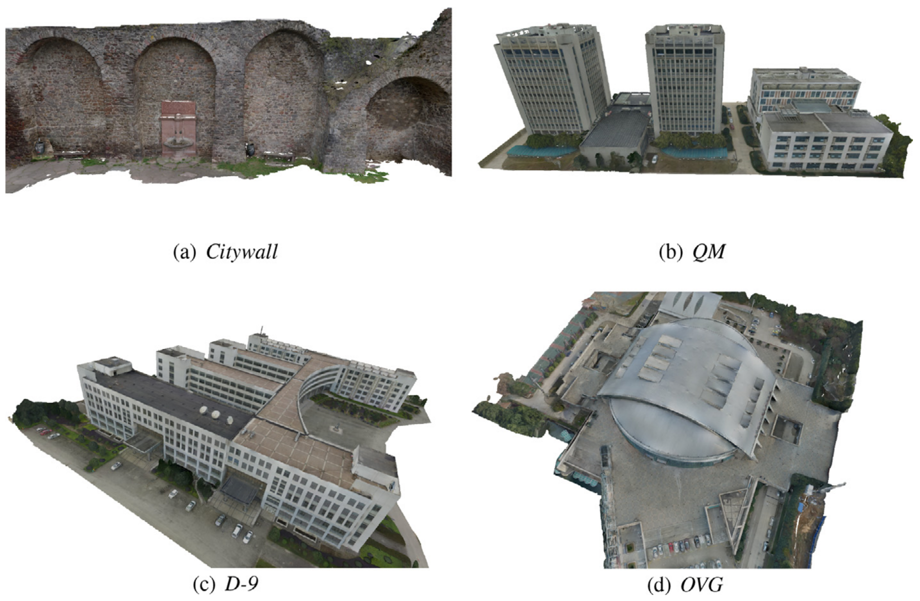

Fig. 10. Textured models with our view selection results. These models have been post processed via seam leveling techniques.

texture quality effectively, and a large 3D mesh graph is partitioned into structural components so as to reduce memory consumption of the proposed algorithm. Experimental results over multiple large-scale datasets demonstrated the proposed algorithm's outperformance over state-of-the-art methods in both numerical efficiency and visual quality; meanwhile, our GPU-accelerated algorithm can yield textured models with high quality from complex large-scale scenes in minutes.

CRediT authorship contribution statement

Xin Sheng: Methodology, Software, Validation, Writing - original draft. **Jing Yuan:** Conceptualization, Investigation, Supervision, Writing - original draft. **Wenbing Tao:** Methodology, Software, Validation. **Bo Tao:** Methodology, Software, Validation. **Liman Liu:** Supervision, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 61772213, 61976227, 91748204 and 61877047 and in part by the Wuhan Science and Technology Plan under Grant 2017010201010121 and Shenzhen Science and Technology Plan under Grant JCYJ20170818165917438.

References

[1] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, R. Szeliski, Building rome in a day, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2009, pp. 72–79.
[2] K. Sun, W. Tao, A center-driven image set partition algorithm for efficient structure from motion, Inf. Sci. 479 (2019) 101–115.
[3] B. Zhao, X. Le, J. Xi, A novel sdass descriptor for fully encoding the information of a 3d local surface, Inf. Sci. 483 (2019) 363–382.
[4] J.L. Schönberger, J.-M. Frahm, Structure-from-motion revisited, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104–4113.
[5] Q. Xu, W. Tao, Multi-scale geometric consistency guided multi-view stereo, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5483–5492.
[6] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Trans. Graph. (ToG) 32 (2013) 29.
[7] Q.-Y. Zhou, V. Koltun, Color map optimization for 3d reconstruction with consumer depth cameras, ACM Trans. Graph. (TOG) 33 (2014) 155–166.
[8] S. Bi, N.K. Kalantari, R. Ramamoorthi, Patch-based optimization for image-based texture mapping, ACM Trans. Graph. (TOG) 36 (2017) 106–117.
[9] Y. Fu, Q. Yan, L. Yang, J. Liao, C. Xiao, Texture mapping for 3d reconstruction with rgb-d sensor, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4645–4653.
[10] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, A. Sellent, Floating textures, in: Computer Graphics Forum, vol. 27, 2008, pp. 409–418..
[11] B. Goldlücke, M. Aubry, K. Kolev, D. Cremers, A super-resolution framework for high-accuracy multiview reconstruction, Int. J. Comput. Vision 106 (2014) 172–191.
[12] A. Richard, I. Cherabier, M. R. Oswald, V. Tsiminaki, M. Pollefeys, K. Schindler, Learned multi-view texture super-resolution, CoRR abs/2001.04775. https://arxiv.org/abs/2001.04775..
[13] L. Grammatikopoulos, I. Kalisperakis, G. Karras, T. Kokkinos, E. Petsa, On automatic orthoprojection and texture-mapping of 3d surface models, Int. Arch. Photogr. Remote Sens. Spatial Inf. Sci. 35 (2004) 360–365.
[14] L. Grammatikopoulos, I. Kalisperakis, G. Karras, E. Petsa, Data fusion from multiple sources for the production of orthographic and perspective views with automatic visibility checking, in: CIPA 2005 XX International Symposium, vol. 26, 2005..
[15] L. Grammatikopoulos, I. Kalisperakis, G. Karras, E. Petsa, Automatic multi-view texture mapping of 3d surface projections, in: Proceedings of the 2nd ISPRS International Workshop 3D-ARCH, 2007, pp. 1–6.
[16] M. Callieri, P. Cignoni, M. Corsini, R. Scopigno, Masked photo blending: mapping dense photographic data set on high-resolution sampled 3d models, Comput. Graph. 32 (2008) 464–473.
[17] V. Lempitsky, D. Ivanov, Seamless mosaicing of image-based texture maps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–6.
[18] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 1222–1239.
[19] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, D. Cohen-Or, Seamless montage for texturing models, in: Computer Graphics Forum, vol. 29, 2010, pp. 479–486..
[20] M. Waechter, N. Moehrle, M. Goesele, Let there be color! large-scale texturing of 3d reconstructions, in: Proceedings of the European Conference on Computer Vision (ECCV), 2014, pp. 836–850.
[21] T.F. Chan, S. Esedoglu, M. Nikolova, Algorithms for finding global minimizers of image segmentation and denoising models, SIAM J. Appl. Math. 66 (5) (2006) 1632–1648 (electronic).
[22] T. Pock, T. Schoenemann, G. Graber, H. Bischof, D. Cremers, A convex formulation of continuous multi-label problems, in: Proceedings of the European Conference on Computer Vision (ECCV), 2008, pp. 792–805.
[23] J. Yuan, E. Bae, X.-C. Tai, A study on continuous max-flow and min-cut approaches, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2217–2224.
[24] J. Yuan, E. Bae, X.-C. Tai, Y. Boykov, A continuous max-flow approach to potts model, in: Proceedings of the European Conference on Computer Vision (ECCV), 2010, pp. 379–392.
[25] E. Bae, J. Yuan, X.-C. Tai, Global minimization for continuous multiphase partitioning problems using a dual approach, Int. J. Comput. Vision 92 (2011) 112–129.
[26] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, 1999.
[27] M. Rajchl, J. Yuan, J. White, E. Ukwatta, J. Stirrat, C. Nambakhsh, F. Li, T. Peters, Interactive hierarchical max-flow segmentation of scar tissue from late-enhancement cardiac mr images, IEEE Trans. Med. Imaging 33 (1) (2014) 159–172.
[28] W. Qiu, J. Yuan, E. Ukwatta, Y. Sun, M. Rajchl, A. Fenster, Prostate segmentation: an efficient convex optimization approach with axial symmetry using 3d trus and mr images, IEEE Trans. Med. Imaging 33 (4) (2014) 1–14.

[29] J. Yuan, K. Yin, Y.-G. Bai, X.-C. Feng, X.-C. Tai, Bregman-proximal augmented lagrangian approach to multiphase image segmentation, in: International Conference on Scale Space and Variational Methods in Computer Vision (SSVM), 2017, pp. 524–534.
[30] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, 2006, pp. 61–70.
[31] I. Ekeland, R. Temam, Convex Analysis and Variational Problems, vol. 28, 1999..
[32] M. Waechter, M. Beljan, S. Fuhrmann, N. Moehrle, J. Kopf, M. Goesele, Virtual rephotography: novel view prediction error for 3d reconstruction, ACM Trans. Graph. (TOG) 36 (2017) 8.
[33] W. Zhou, B. Alan Conrad, S. Hamid Rahim, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (2004) 600–612.
[34] D. Thuerck, M. Waechter, S. Widmer, M. von Buelow, P. Seemann, M. Pfetsch, M. Goesele, A fast, massively parallel solver for large, irregular pairwise markov random fields, in: Proceedings of High Performance Graphics, 2016, pp. 173–183..