三维几何数据的交互式着色算法

单 欣[1)], 赵 勇[2)*], 李 玲[2)], 郭清华[2)], 董军宇[1)]

[1)] (中国海洋大学计算机科学与技术系 青岛 266100)
[2)] (中国海洋大学数学科学学院 青岛 266100)
(yongzhao.ouc@gmail.com)

摘 要: 获取三维模型的纹理信息是计算机图形学领域的一个重要研究问题. 传统的方法往往通过纹理映射来完成这个任务, 其局限是需要一幅合适的纹理图像. 为了能够更简便地得到模型的纹理, 提出一种交互式着色算法. 对于网格模型, 首先进行显著特征的提取和分类来减少用户对重复出现的特征的交互量, 进而由用户在不同区域上交互几条颜色曲线作为种子曲线. 然后结合位置、法向和曲率信息来衡量相邻顶点的相似度, 以防止相邻区域之间出现渗色, 并通过随机游走算法计算出每个顶点到每条种子曲线的跳转概率; 最后以跳转概率作为权值对各条种子曲线的颜色进行加权平均, 得到每个网格顶点的颜色. 进一步地, 还将上述算法应用到点云模型上. 实验结果表明, 该算法能够准确地区分不同的区域, 鲁棒地为三维模型着色.

关键词: 三维模型; 交互式着色; 随机游走; 显著特征
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2019.17392

An Interactive Colorization Algorithm for 3D Models

Shan Xin[1)], Zhao Yong[2)*], Li Ling[2)], Guo Qinghua[2)], and Dong Junyu[1)]

[1)] (*Department of Computer Science and Technology, Ocean University of China, Qingdao* 266100)
[2)] (*School of Mathematical Sciences, Ocean University of China, Qingdao* 266100)

Abstract: It is a fundamental problem to obtain texture information of 3D models in computer graphics. Texture mapping was often used to complete this task. However, texture mapping needed a suitable texture image. Therefore, an interactive colorization algorithm is proposed in this paper to obtain texture information more easily. For a mesh model, salient features are first extracted and classified to reduce user's interaction on repetitive features, and then user can scribble some curves with desired colors in various regions as seed curves. Secondly, position, normal, and curvature information are combined to measure vertices similarity to avoid color bleeding between neighboring regions, and random walk algorithm is adopted to compute the probability from each vertex to each seed curve. Finally, weighted by these probabilities, the color of each vertex can be expressed as the weighted average of seed curves' color. Furthermore, the above algorithm is applied to colorize point clouds. Experiment results show that our algorithm can distinguish different regions accurately, and colorize 3D models robustly.

收稿日期: 2018-06-30; 修回日期: 2018-08-23. 基金项目: 国家自然科学基金(61303145, U1706218); 山东省自然科学基金(ZR2018MF006); 浙江大学 CAD&CG 国家重点实验室开放课题(A1809). 单 欣(1993—), 女, 硕士研究生, 主要研究方向为计算机图形学、数字几何处理; 赵 勇(1982—), 男, 博士, 副教授, 硕士生导师, CCF 会员, 论文通讯作者, 主要研究方向为计算机图形学、计算机视觉、数字几何处理; 李 玲(1991—), 女, 硕士研究生, 主要研究方向为计算机图形学、数字几何处理; 郭清华(1994—), 女, 硕士研究生, 主要研究方向为计算机图形学、数字几何处理; 董军宇(1972—), 男, 博士, 教授, 博士生导师, CCF 会员, 主要研究方向为计算机视觉、机器学习、模式识别.

Key words: 3D model; interactive colorization; random walk; salient features

随着三维数据采集技术的发展, 三维模型已经被广泛地应用于计算机图形学、计算机视觉等领域. 因此, 针对模型分析和理解的需求不断增加, 而模型表面的纹理信息是分析和理解模型的重要线索. 对于三维模型而言, 通常利用扫描设备获取其纹理信息, 或者通过纹理映射用一幅图像覆盖模型表面. 但是, 这 2 种方法都有各自的局限性: 扫描得到的数据通常会带有不同程度的噪声, 并且每次扫描只能在一个角度进行, 要想获得完整的纹理信息还必须执行去噪、配准等复杂的后处理操作; 而采用纹理映射方法的前提是能够找到一幅合适的纹理图像.

近年来, 研究人员提出了利用着色理论获取纹理信息的方法. 着色理论首先被应用到二维图像上. Levin 等[1]假设灰度值相近的相邻像素应该具有相似的颜色, 提出一种简单有效的交互式图像着色算法. 为了实现网格模型的着色, Leifman 等[2]将交互式着色算法从图像扩展到网格模型上, 并提出用于解决渗色的特征线场. 随后, Leifman 等[3]又给出一种模式驱动的三维表面着色算法, 用户只需在重复模式的一个样例上着色, 就能够识别并分类所有重复的模式, 并为同一类模式着相同的颜色; 但是该算法复杂度比较高, 复杂情况下的鲁棒性不够好. 为此, 本文提出一种交互式着色算法, 能够更简便地得到模型的纹理信息, 并且同时适用于网格模型和点云模型.

1　相关工作

纹理映射. 纹理映射是计算机图形学中常用的一种技术, 可以为三维场景定义高频细节、表面纹理或颜色信息. 文献[4-6]利用参数化方法来实现该过程, 首先将流形展开到图像上, 然后通过用户定义的特征对参数进行约束, 得到模型与纹理图像之间的映射关系. 参数化方法使用的纹理图片可以是任意的, 没有考虑到拍摄效果对映射过程的影响. 因此, 文献[7-9]将拍摄图像的几何信息考虑到纹理映射过程中, 通过恢复摄像机参数即可隐式地得到图像和模型之间的映射关系.

交互式网格分割. 研究人员通过交互式方法实现了网格模型的分割, 其目标是根据用户意图将模型分割成具有不同语义的部分. Ji 等[10]首次提出交互式的网格模型分割方法, 通过用户交互在模型上产生 2 笔笔触, 区分模型的前景区域和背景区域; 然后使用改进的区域生长算法完成后续的分割任务. 随后, 交互方式也由最初的指定前景笔触和背景笔触[10], 发展到指定跨边界笔触[11]、只指定前景笔触[12]、指定沿边界笔触[13]和指定一个边界点[14]. Lai 等[15]将随机游走算法从图像扩展到三维模型上, 当用户给出随机游走的种子曲线后, 通过计算不同顶点到种子曲线的跳转概率完成模型分割. 此外, Zhang 等[16]在随机游走算法中加入 3 种不同的约束, 并给出一种有效的边界优化策略, 实现了较好的分割结果. 分割和着色在某种程度上是类似的, 分割是将不同区域严格分开, 而着色要求颜色在不同区域之间光滑过渡.

图像着色. 早期的着色工作一般是在图像上完成的. Welsh 等[17]提出一种半自动的着色方法, 可以参考一幅彩色图像实现灰度图像的着色. Levin 等[1]的算法只需用户在图像上指定几种想要的颜色, 就可以自动完成整幅图像的着色. Huang 等[18]利用自适应边缘检测解决了在图像边界处的渗色问题. Qu 等[19]给出了保持模式连续性的 Manga 图像着色算法. 之后, 朱薇等[20]实现了保色调的黑白卡通图像的着色, 通过图像分解技术获得卡通图像的色调, 利用加伯小波变换防止图像边界出现渗色问题. 最近, Zhang 等[21]结合卷积神经网络和少量用户交互, 使用深度学习解决了图像着色问题.

网格着色. 在网格模型着色方面, Leifman 等[2]将交互式着色算法从图像扩展到网格模型上, 实现了网格模型的着色; 不过在处理具有重复模式的模型时, 需要用户进行多次重复交互才能完成着色任务. Leifman 等[3]提出的模式驱动的三维表面着色算法中, 用户只需在重复模式的一个样例上简单引导, 就能够准确地将三维表面上所有重复的模式识别出来, 并为这些模式着相同的颜色; 该算法不仅实现了着色, 同时也解决了重复模式的检测问题. 除此之外, 还可以利用一些商业工具, 如 Adobe Illustrator, 3DS Max 和 3D-Brush 等实现模型着色. 但使用这些工具往往需要具备丰富的专业知识, 即使对于专业人员, 利用这些工具进行着色也是一项非常耗时的工作.

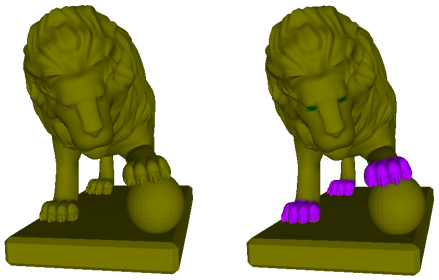
2 交互式网格模型着色框架

用二元组 $\{V,E\}$ 定义一个三角网格模型，其中，$V=\{\boldsymbol{v}_i \mid \boldsymbol{v}_i \in \mathbb{R}^3,\ i=1,\cdots,m\}$ 表示网格的顶点集，$E=\{e_{ij}=(\boldsymbol{v}_i,\boldsymbol{v}_j) \mid \boldsymbol{v}_i,\boldsymbol{v}_j \in V,\ i \neq j\}$ 表示网格的边集. $N(i)$ 表示顶点 $\boldsymbol{v}_i$ 的相邻顶点的集合.

实现着色时，用户可以在模型上交互出 n 条颜色曲线作为种子曲线，代表想为不同区域指定的颜色，其中每条种子曲线都是由若干个网格顶点构成的. 本文算法首先检测和分类模型的显著特征，从而有效地减少用户的交互量；其次计算相邻顶点的相似度，把位置、法向和曲率信息结合到相似度的计算中，避免使用单一几何特征带来的偏差，解决模型边界处的渗色问题；最后执行随机游走算法，将上述相似性度量作为顶点 $\boldsymbol{v}_i$ 沿网格边 e_{ij} 游走到顶点 $\boldsymbol{v}_j$ 的概率，构造一个线性优化问题，通过求解该问题实现整个模型的着色.

2.1 显著特征的提取与分类

通常，网格模型可能具有一些重复的显著特征，如图 1a 的 Lion 模型，在对该类模型着色时，需要用户进行多次重复操作才能为这些特征指定相应的颜色. 模型越复杂，其包含的显著特征就可能越多，完成着色需要的用户交互量也就越大. 为了减少不必要的用户交互，本文提出一种显著特征提取和分类方法，可以自动提取和分类模型的显著特征. 在着色时，本文总是假设同一类显著特征的颜色是一样的. 因此，用户只需在每种显著特征的一个样例上给出想为该类特征添加的颜色，算法就可以自动完成同类特征的着色.



a. 原始网格模型 b. 重复出现的特征

图 1 Lion 网格模型显著特征的提取和分类

给定一个网格模型，本文提取出模型所有的显著特征，并对这些显著特征进行分类. 提取显著特征时，首先利用热核描述符[22]和波核描述符[23]计算每个网格顶点的局部显著度，热核描述符和波核描述符可以很好地反映模型的内在属性，准确得出模型的显著信息. 然后根据局部显著度，以区域生长的方式对模型进行聚类，区域生长是一种自下而上的聚类方法，以种子点为中心向周围扩张，不断选取新的种子点继续扩张，直至达到阈值为止；对于未被聚类的顶点，每次都选取局部显著度最小的顶点作为新的聚类种子点，从而保证平坦的区域尽量聚为一类. 最后将显著度较大的类与其周围具有相近显著度的类进行合并，得到显著特征，此过程需要迭代地进行，直至显著度较大的类不能和周围的类合并为止，这些类就构成了模型的显著特征.

最终形成的每个显著特征都是由一组网格顶点构成的. 当模型包含多种显著特征时，还需要对这些特征进行分类. 由于同一类显著特征也可能存在着尺度、朝向等方面的差异，本文首先对提取的所有特征进行归一化处理，并通过主元分析进行配准；然后计算显著特征之间的 Hausdorf 距离，将距离较近的特征归为一类.

本文的目标是得到重复出现的特征，从而减少后续的交互量，因此没有保留只出现一次的特征. 图 1 中，每一类特征被赋予一种随机颜色. 最终得到 2 类重复出现的特征：眼睛和爪子.

2.2 随机游走着色算法

假设用户在模型上进行了 n 次交互，产生了 n 条颜色曲线作为随机游走的种子曲线，分别记为 $S_1,S_2,\cdots,S_n$，其中，每条种子曲线都是由若干个网格顶点组成的. 随机游走算法实现着色的思路是：首先计算出每个顶点到每条种子曲线的跳转概率，然后利用跳转概率将各条种子曲线的颜色进行加权平均得到每个顶点的颜色.

用 $p^k(\boldsymbol{v}_i)$ 表示顶点 $\boldsymbol{v}_i$ 到种子曲线 S_k 的跳转概率，其中，$1 \leqslant i \leqslant m$, $1 \leqslant k \leqslant n$. 随机游走算法要求跳转概率在整个网格模型上是连续变化的，因此，顶点 $\boldsymbol{v}_i$ 到 S_k 的跳转概率 $p^k(\boldsymbol{v}_i)$ 可以表示为其邻接顶点 $\boldsymbol{v}_j\left(j \in N(i)\right)$ 到 S_k 的跳转概率的加权和

$$p^k(\boldsymbol{v}_i)=\frac{\sum\limits_{j\in N(i)} \omega_{ij} p^k(\boldsymbol{v}_j)}{\sum\limits_{j\in N(i)} \omega_{ij}} \tag{1}$$

其中，ω_{ij} 是顶点 $\boldsymbol{v}_i$ 和 $\boldsymbol{v}_j$ 的相似度，它决定了顶点 $\boldsymbol{v}_i$ 沿边 e_{ij} 随机游走一步的概率. 相邻顶点的相似度越高，沿对应边游走的概率越大. 另外，相似度的定义也会影响区域边界着色的准确程度. 如果不能准确地判断相邻顶点的相似程度，那么当颜色传播到区域边界时就可能出现渗色，影响着色效果.

本文结合位置、法向和曲率信息共同定义顶点 $\boldsymbol{v}_i$ 和 $\boldsymbol{v}_j$ 的相似度

$$\omega_{ij}=e^{\frac{-\|\boldsymbol{v}_i-\boldsymbol{v}_j\|^2}{2\sigma_v^2}}e^{\frac{-\|\boldsymbol{n}_i-\boldsymbol{n}_j\|^2}{2\sigma_{\boldsymbol{n}}^2}}e^{\frac{-\|c_i-c_j\|^2}{2\sigma_c^2}}.$$

其中，$\boldsymbol{n}_i$ 和 $\boldsymbol{n}_j$ 分别表示点 $\boldsymbol{v}_i$ 和 $\boldsymbol{v}_j$ 的单位法向；c_i 和 c_j 分别表示点 $\boldsymbol{v}_i$ 和 $\boldsymbol{v}_j$ 处的平均曲率；σ_v^2 表示 $\left\{\left\|\boldsymbol{v}_i-\boldsymbol{v}_j\right\|\right\}_{j\in N(i)}$ 的方差，$\sigma_{\boldsymbol{n}}^2$ 表示 $\left\{\left\|\boldsymbol{n}_i-\boldsymbol{n}_j\right\|\right\}_{j\in N(i)}$ 的方差，σ_c^2 表示 $\left\{\left\|c_i-c_j\right\|\right\}_{j\in N(i)}$ 的方差；$e^{(\cdot)}$ 表示高斯函数. 该顶点相似度的定义方式充分考虑了空间距离、法向差别和曲率差别，能够保证相邻顶点的距离越近、法向夹角越小、曲率越接近，那么相似度就越高.

特别地，当顶点 $\boldsymbol{v}_i$ 属于种子曲线 S_k 时，$\boldsymbol{v}_i$ 到 S_k 的跳转概率等于 1；当顶点 $\boldsymbol{v}_i$ 属于其他种子曲线时，$\boldsymbol{v}_i$ 到 S_k 的跳转概率等于 0. 即

$$p^k(\boldsymbol{v}_i)|_{\boldsymbol{v}_i\in S_k}=1 \tag{2}$$

$$p^k(\boldsymbol{v}_i)|_{\boldsymbol{v}_i\in\{S_1,S_2,\cdots,S_{k-1},S_{k+1},\cdots,S_n\}}=0 \tag{3}$$

满足式(2)(3)这 2 个约束条件，能够使得跳转概率 $p(\boldsymbol{v})\in[0,1]$.

对于种子曲线 S_k，任意顶点都有式(1)，则 m 个顶点就有 m 个方程

$$p^k(\boldsymbol{v}_i)-\frac{\sum_{j\in N(i)}\omega_{ij}p^k(\boldsymbol{v}_j)}{\sum_{j\in N(i)}\omega_{ij}}=0,\ 1\leqslant i\leqslant m;$$

进而与式(2)(3)联立能够得到一个线性方程组，求解后可以得到所有顶点到种子曲线 S_k 的跳转概率. 求解 n 个这样的线性方程组，就可以得到每个顶点到每条种子曲线的跳转概率，即 $\{p^1(\boldsymbol{v}_i),p^2(\boldsymbol{v}_i),\cdots,p^k(\boldsymbol{v}_i),\cdots,p^n(\boldsymbol{v}_i)\}$，$1\leqslant i\leqslant m$，并且满足 $\sum_{k=1}^{n}p^k(\boldsymbol{v}_i)=1$.

着色时，本文利用每条种子曲线的颜色加权得到各个顶点的颜色. 在加权时，将所求跳转概率作为对应种子曲线颜色的权值，可以使得颜色在整个模型上连续变化，保证良好的着色效果. 假设 $C(S_k)$表示种子曲线 S_k 的颜色，那么任意顶点 $\boldsymbol{v}_i$ 的颜色 $C(\boldsymbol{v}_i)$ 就可以表示为 $C(\boldsymbol{v}_i)=\sum_{k=1}^{n}p^k(\boldsymbol{v}_i)C(S_k)$.

3　点云模型的着色

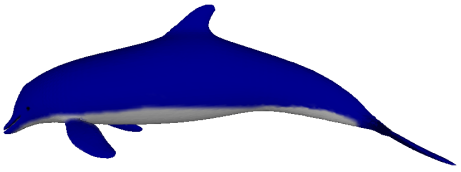
进一步地，本文还扩展了上述算法，用于解决点云模型的着色问题. 点云模型是用一系列空间无序的采样点来表示三维模型的，与网格模型不同，点云模型没有明确的拓扑连接信息. 为此，本文利用 k 最近邻域得到每个采样点的局部邻域，进而通过协方差分析[24]计算每个采样点的法向、曲率等局部几何属性. 那么，上述算法便可以被应用到点云模型，获得良好的着色效果.

4　实验结果与分析

本文算法利用 C++语言编程实现，实验是在配置为 Intel Core i5-3320M CPU, 3.45 GB 内存的 PC 机上完成的. 当用户指定颜色曲线后，系统就会根据随机游走算法完成整个模型的着色.

图 2 所示为 Dolphin 网格模型的着色结果，该模型的几何特征变化比较平缓，其背部、腹部、鳍部的边界不明显，很容易出现渗色. 可以看出，本文算法可以有效地处理这个问题，得到真实自然的效果. 图 3 中，本文分别对没有噪声和带有高斯噪声的模型进行了着色. 可以看出，即使在有噪声的情况下，本文算法依然能够完成着色，并得到良好的效果. 对比图 3b 和图 3d 可以看出，它们的结果非常相近，体现了本文算法的鲁棒性. 此外，本文算法还可以拓展到点云模型上. 图 4 给出了点云模型的着色结果，体现了本文算法的可扩展性和灵活性.

a. 原始网格模型

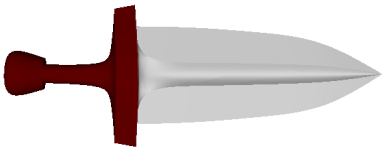


b. 着色结果

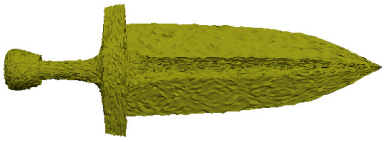
图 2　Dolphin 网格模型的着色结果

图 5 所示为对比本文算法与文献[2]算法得到的实验结果. 其中，文献[2]算法采用局部直方图和扩散距离来衡量顶点的相似程度，计算局部直方图时，网格顶点要投影到一个局部的二维平面上，难以处理复杂情形. 图5b中，螺丝刀头的一部分被判断为刀柄的一部分，导致不准确的着色边界. 相比之下，本文算法结合了位置、法向以及

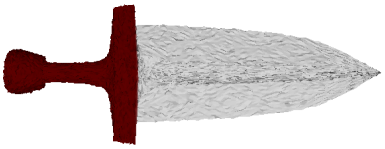
a. 原始网格模型



b. a 的着色结果



c. 带噪声的网格模型

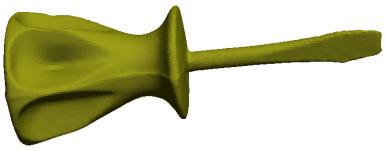


d. c 的着色结果

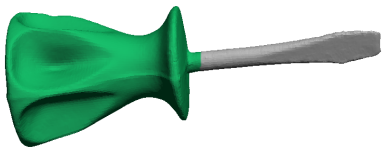
图 3 Sword 网格模型的着色结果

a. 原始点云模型 b. 着色结果

图 4 Chair 点云模型的着色结果



a. 原始网格模型



b. 文献[2]算法

c. 本文算法

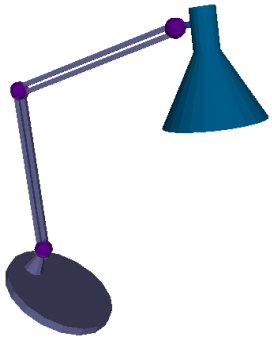
图 5 Screwdriver 网格模型着色结果的比较

曲率共同描述顶点相似度，可以更加准确地区分不同的区域，如图 5c 所示.

图 6 所示为对比本文算法与文献[3]算法得到的实验结果，对于 Lamp 网格模型，不同区域之间的关节部分是重复出现的显著特征. 文献[3]中采用特征直方图和局部拟合方法，未能正确地提取出这些特征，从而造成关节部分和灯架部分之间出现了渗色，如图 6b 所示. 如图 6c 所示，本文算法的结果更加真实自然.

a. 原始网格模型

b. 文献[3]算法



c. 本文算法

图 6 Lamp 网格模型着色结果的比较

表 1 中统计了采用本文算法对文中一些例子模型的几何信息和时间效率，其中，算法时间取决于三维模型的特征情况、顶点个数、交互的颜色曲线数量等因素. 总体而言，本文算法能够比较快速地完成模型的着色.

表 1　本文算法对不同模型的处理时间

模型	顶点数	算法时间/s
Dagger	8325	3.619
Dolphin	11432	4.387
Sword	17985	5.012
Screwdriver	27152	6.423
Lamp	18132	6.589

5　结　语

本文利用随机游走算法实现了三维模型的着色, 用户只需在模型的不同区域上交互几笔颜色, 系统就可以自动完成模型的着色. 由于通过随机游走求出的跳转概率可以构成一个调和场, 使得跳转概率在整个模型上均匀变化, 因此在着色时, 本文用跳转概率作为权值, 将每个顶点的颜色表达为所有种子曲线颜色的加权平均, 可以保证颜色在整个模型上连续变化, 得到令人满意的效果. 针对具有重复模式的模型, 为了降低用户的交互量, 本文提出一种显著特征的提取和分类算法. 实验结果表明, 本文算法具有良好的鲁棒性, 能够有效地为各种模型实现着色.

本文算法有两方面的不足. (1) 如果模型的采样比较差, 会导致不准确的特征提取结果. 因此, 将来会提出更鲁棒的特征提取方法. (2) 对于点云模型, 本文使用的是 k 最近邻域, 有些情况下会产生错误的邻域关系. 此时, 需要使用更多的局部属性来确定邻域, 如法向、曲率等.

参考文献(References):

[1] Levin A, Lischinski D, Weiss Y. Colorization using optimization[J]. ACM Transactions on Graphics, 2004, 23(3): 689-694

[2] Leifman G, Tal A. Mesh colorization[J]. Computer Graphics Forum, 2012, 31(2pt2): 421-430

[3] Leifman G, Tal A. Pattern-driven colorization of 3D surfaces[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2013: 241-248

[4] Lévy B. Constrained texture mapping for polygonal meshes[C] //Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 2001: 417-424

[5] Tai Y W, Brown M S, Tang C K, *et al*. Texture amendment: reducing texture distortion in constrained parameterization[J]. ACM Transactions on Graphics, 2008, 27(5): Article No.136

[6] Ma Y W, Zheng J M, Xie J. Foldover-free mesh warping for constrained texture mapping[J]. IEEE Transactions on Visualization and Computer Graphics, 2015, 21(3): 375-388

[7] Sinha S N, Steedly D, Szeliski R, *et al*. Interactive 3D architectural modeling from unordered photo collections[J]. ACM Transaction on Graphics, 2008, 27(5): Article No.159

[8] Xiao J X, Fang T, Tan P, *et al*. Image-based facade modeling[J]. ACM Transaction on Graphics, 2008, 27(5): Article No.161

[9] Tzur Y, Tal A. Flexistickers: photogrammetric texture mapping using casual images[J]. ACM Transactions on Graphics, 2009, 28(3): Article No.45

[10] Ji Z P, Liu L G, Chen Z G, *et al*. Easy mesh cutting[J]. Computer Graphics Forum, 2006, 25(3): 283-291

[11] Zheng Y Y, Tai C L. Mesh decomposition with cross-boundary brushes[J]. Computer Graphics Forum, 2010, 29(2): 527-535

[12] Fan L B, Liu L G, Liu K. Paint mesh cutting[J]. Computer Graphics Forum, 2011, 30(2): 603-612

[13] Meng M, Fan L B, Liu L G. ICutter: a direct cut-out tool for 3D shapes[J]. Computer Animation and Virtual Worlds, 2011, 22(4): 335-342

[14] Zheng Y Y, Tai C L, Au O K C. Dot scissor: a single-click interface for mesh segmentation[J]. IEEE Transactions on Visualization and Computer Graphics, 2012, 18(8): 1304-1312

[15] Lai Y K, Hu S M, Martin R R, *et al*. Rapid and effective segmentation of 3D models using random walks[J]. Computer Aided Geometric Design, 2009, 26(6): 665-679

[16] Zhang J Y, Zheng J M, Cai J F. Interactive mesh cutting using constrained random walks[J]. IEEE Transactions on Visualization and Computer Graphics, 2011, 17(3): 357-367

[17] Welsh T, Ashikhmin M, Mueller K. Transferring color to greyscale images[J]. ACM Transactions on Graphics, 2002, 21(3): 277-280

[18] Huang Y C, Tung Y S, Chen J C, *et al*. An adaptive edge detection based colorization algorithm and its applications[C] //Proceedings of the 13th Annual ACM International Conference on Multimedia. New York: ACM Press, 2005: 351-354

[19] Qu Y G, Wong T T, Heng P A. Manga colorization[J]. ACM Transactions on Graphics, 2006, 25(3): 1214-1220

[20] Zhu Wei, Liu Ligang. Tone-preserving colorization of black-and-white cartoon image[J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(3): 392-398(in Chinese)
(朱　薇, 刘利刚. 保色调的黑白卡通图像着色方法[J]. 计算机辅助设计与图形学学报, 2011, 23(3): 392-398)

[21] Zhang R C, Zhu J Y, Isola P, *et al*. Real-time user-guided image colorization with learned deep priors[J]. ACM Transactions on Graphics, 2017, 36(4): Article No.119

[22] Sun J, Ovsjanikov M, Guibas L J. A concise and provably informative multi-scale signature based on heat diffusion[J]. Computer Graphics Forum, 2009, 28(5): 1383-1392

[23] Aubry M, Schlickewei U, Cremers D. The wave kernel signature: a quantum mechanical approach to shape analysis[C] //Proceedings of the IEEE International Conference on Computer Vision Workshops. Los Alamitos: IEEE Computer Society Press, 2011: 1626-1633

[24] Pauly M, Gross M, Kobbelt L P. Efficient simplification of point-sampled surfaces[C] //Proceedings of the IEEE Conference on Visualization. Los Alamitos: IEEE Computer Society Press, 2002: 163-170