An Interactive 2D-to-3D Cartoon Modeling System

Lele Feng(✉), Xubo Yang(✉), Shuangjiu Xiao, and Fan Jiang

School of Software, Shanghai Jiao Tong University, Shanghai, China
lelefengl992@gmail.com, yangxubo@sjtu.edu.cn

Abstract. In this paper, we propose an interactive system that can quickly convert a 2D cartoon painting into a 3D textured cartoon model, enabling non-professional adults and children to easily create personalized 3D contents. Our system exploits a new approach based on solving Poisson equations to generate 3D models, which is free from the limitations of spherical topology in prior works. We also propose a novel method to generate whole textures for both sides of the models to deliver colorful appearances, making it possible to obtain stylized models rendered with cartoon textures. The results have shown that our method can greatly simplify the modeling process comparing with both traditional modeling softwares and prior sketch-based systems.

Keywords: Modeling interface · Deformation · Texture · Interactive modeling

1 Introduction

The demand for personalized 3D models is rapidly growing with the increasing applications of emerging technologies like 3D printing and augmented reality. However, for novice users, it is difficult to build 3D models using professional modeling systems. Traditional 3D modeling tools, such as Maya [1] and 3ds Max [2] require users to learn a complicated interface, which are daunting challenges for novice users. In order to simplify the 3D modeling pipeline, sketch-based modeling systems, such as Teddy [3] and its follow-up works [4–8] presented approaches to create 3D models from 2D strokes. However, these systems have some common limitations. First, most of these systems require users to sketch from a large number of different views, making it difficult for novice users to complete their tasks. Second, these systems cannot generate models from 2D cartoon images directly and only use them as the guide images. In order to create desired models, users have to draw carefully to make their sketch match with the silhouettes in guide images. As a result, the silhouettes of the final shape may differ from the input sketch, which may be undesired. Third, shapes that can be handled by these systems are also limited. For example, they do not allow cycles of connection curves. Surfaces with edges or flat surfaces also cannot be generated directly. Finally, models generated by these systems lack texture information, thus the rendering of these models cannot take advantage of the original paintings. Unlike sketch-based systems, Ink-and-Ray [9] involves Poisson equations to produce bas-relief meshes, which can support global illumination effects for hand-drawn characters. However, the resulted

A. El Rhalibi et al. (Eds.): Edutainment 2016, LNCS 9654, pp. 193–204, 2016.
DOI: 10.1007/978-3-319-40259-8_17

meshes are not full 3D models, which limit their usage and artifacts may appear when rendered with a perspective camera or from different views.

The goal of our work is to design a system that allows creating full three-dimensional models from two-dimensional cartoon images directly without requiring much input. Our system makes it possible to quickly create a consistent 3D model with full textures.

Contributions. We introduce an interactive system to help non-professional users to build their own personalized 3D contents, where a full 3D textured cartoon model can be created easily from a single 2D cartoon image. First, our system employs an automatic segmentation method to reduce the difficulty of sketching silhouettes in the previous works. Second, the algorithm of mesh generation used by our system can handle a large range of models, including holes and sharp edges. We also introduce a method to generate textures for full models, which is not supported by prior works. Our results show that our system makes it an easy task for novice users to create personalized 3D models from cartoon paintings.

2 Related Work

Many efforts have been made in constructing 3D models from 2D images. Previous systems can be classified into three groups: sketch-based modeling systems, single-view modeling systems and pseudo 3D modeling systems.

2.1 Sketch-Based Modeling System

Sketch-based 3D modeling has been a popular research field. Systems such as Teddy [3] and its descendants ShapeShop and [4] FiberMesh [5] approached the problem by asking users to sketch from many views, leveraging users' 2D drawing skills. A good survey can be found in [10]. RigMesh [8] presented an approach to modeling and rigging in contrast to the traditional sequential. However, their system cannot generate a desirable shape when the input sketch does not have an obvious symmetry axis. Moreover, all of these methods cannot deliver textured models, unless users draw colorful strokes on the models directly. In addition, due to their modeling methods (limited to spherical topology), the range of models that these systems can handle is limited. Surfaces with edges or flat surfaces cannot be generated directly. And they do not allow cycles of connection curves. Another drawback of their methods is requiring users to sketch from a large number of different views. Schmidt et al. [11] found that novice users were unable to complete their tasks and became frustrated very easily with the change of views.

2.2 Single-View Modeling System

In order to solve the problems in multi-view sketch-based systems, many single-view modeling systems are proposed. In Gingold et al. [7], structured annotations were introduced for 2D-to-3D modeling. In their implementation, the user can add some

primitives and annotations, which are structured and semantic. The system then generates 3D models from inconsistent drawings. However, adding annotations from a single view demands decent 3D perspective understandings, which is difficult and time-consuming for novice users. A similar work using annotations is Natura sketch [12]. The system is a sketch-based modeling system per se, but it solved the problems in sketch-rotate-sketch workflow by introducing multiple annotations. Karpenko and Hughes [13] proposed a system that can generate 3D models from single-view contours. A common drawback of these methods is that they require tedious specification of the inputs to produce the desired results.

2.3 Pseudo 3D Modeling System

Other researches focus on producing pseudo 3D models rather than full 3D models. Rivers et al. [14] proposed an approach which using 2D input to generate 2.5D models which can be rotated in 3D. Instead of generating a 3D polygonal mesh, they generate a 2.5D cartoon, which naturally supports a stylized drawing style. However, the system demands artists to draw different views (at least three), and some effects we take for granted with a 3D model, such as lighting and collision detection, are not natively supported. TexToons [15] is a system that uses depth layering to enhance 3D-like effects to texture-mapped hand-drawn cartoons, where the overall appearance feels synthetic due to the lack of full 3D details. Sýkora et al. [9] presented a system to produce bas-relief meshes. The approach preserved the look-and-feel of the two-dimensional domain and delivered a similar look compared with a full three-dimensional model. However, the meshes created by the system cannot be rotated due to their bas-relief topology. When rendered with a perspective camera or viewed from different angles, their system may cause some artifacts because the meshes are not full 3D models.

In this paper, we introduce a system for 3D modeling and deformation from 2D cartoon images. Compared with sketch-based systems, our system does not require the user to model from different views. When provided a cartoon image, our system can segment it into several regions automatically. We design a few user interface elements to help users refine the result. Inspired by the inflation method in Ink-and-Ray [9], our system employs a new approach based on solving Poisson equations to generate full 3D models, which is free from the limitations of spherical topology in prior works. We also propose a method to generate whole textures for both sides of the models to deliver colorful appearance, which is unsupported in the above systems. The user can add point handles to deform models to achieve desired poses. The results have shown that our method can take advantage of the original 2D images as much as possible, while requires a little user input.

3 User Interface

Our interface is shown in Fig. 1. The interface includes three parts: the leftmost palette, which contains all the tools supported by our system, a window displaying the user inputs, and a window displaying the generated 3D models. The user starts by selecting a cartoon image and then performs several operations to generate models. The process can be divided into five steps:

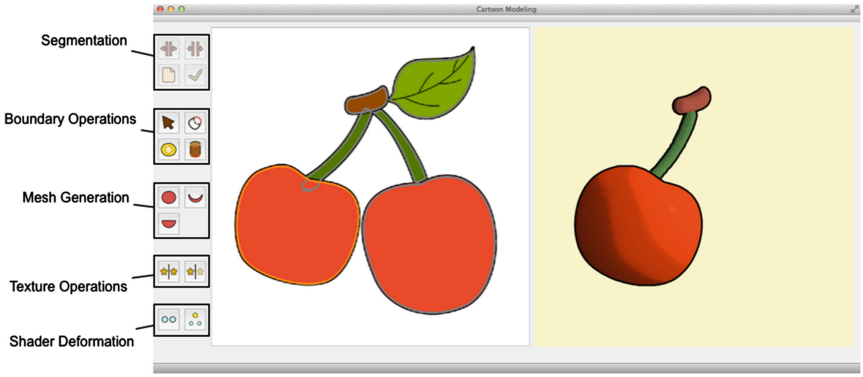

Fig. 1. The user interface of our system.

- Segmentation. After the user selects a cartoon image, our system first extracts the regions from the image. Each area enclosed by dark edges forms a region. The result is the initial segmentation and the user can perform some optional operations to correct the result, such as background selection, merging and splitting regions. After the user has done with these segmented regions, our system will grow these regions to eliminate edge pixels and then find contours for them.
- Boundary operations. Region boundaries are then extracted from the segmentation result. The user can select a boundary and modify it optionally, such as filling holes and specifying flat boundaries.
- Mesh generation. Different 3D models are generated based on these boundaries and their types. Our system supports three kinds of meshes: inflated meshes, concave meshes and half inflated meshes. Figure 2 shows an example of generating three kinds of meshes based on the same boundary.
- Texture operations. Our system can synthesize textures for the generated meshes automatically. The original paintings are used as textures of front faces directly and the back textures are synthesized based on the front textures. Our system uses a heuristic rule that the farther an area is away from the region boundary, the more likely this area should not be reused in the back texture. The user can use tools to modify the results to decide whether or not to reuse an area in the back texture.
- Shape deformation. By default, the centers of generated meshes are all located at the same depth where z = 0. The relative depth order may be undesired. The user can

add some point handles on the mesh, and deform the meshes in order to get a satisfactory depth order. The deform operations supported by our system include translation, rotation and scaling. We find that this tool gives a chance for the user to obtain more imaginative 3D models.

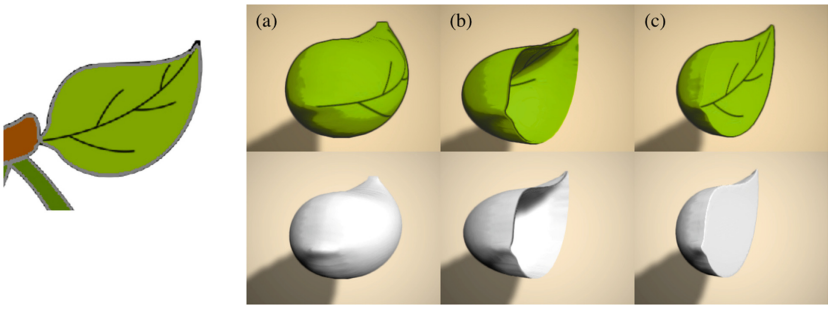

Fig. 2. Three kinds of meshes supported by our system. The left image shows the input boundary. Our system can generate three kinds of meshes based on this boundary. (a) the inflated mesh. (b) the concave mesh. (c) the half inflated mesh.

4 Implementation

In this section, we will describe more details about the above steps.

4.1 Segmentation

Given a cartoon image as input, our system first extracts regions from the image automatically. Our segmentation uses the curve structure extraction method in [16]. The method first calculates curve points by applying non-maximal suppress on secondary derivative of cartoon images, and then links them together to form structure curves. For many cartoon images, this method can extract usable outlines immediately. However, when the input images contain noises, the result is not as neat as desired. To detect final outlines from the initial result, we employ an adaptive algorithm [17] on the initial result. Our system then performs a seed fill algorithm to find enclosed regions.

4.2 Mesh Generation

Our system will find boundaries for each region. These boundaries are used to generate meshes according to their types. We first apply conforming constrained Delaunay triangulation [18] to each boundary, obtaining a discrete region ω. The triangulation uses the Triangle package [19]. We then use the inflation algorithm in [9]. The algorithm takes advantage of solving a Poisson equation:

$$-\nabla^2 f(x) = c, \forall x \in w \tag{1}$$

subject to:

$$f(x) = 0, \forall x \in B_D \tag{2}$$

$$\frac{\partial f(x)}{\partial n} = 0, \forall x \in B_N \tag{3}$$

The function $f(x)$ corresponds to the inflated heights of the region and c is a positive number specifying how much the region is inflated. The equation should subject to two types of boundary constraints, Dirichlet or Neumann boundary constraints. By default, all of the boundary vertices are subject to Dirichlet boundary constraints. The resulting $f(x)$ produces a parabolic-like cross-section. Our system then uses a function $f'(x) = \sqrt{f(x)}$ to convert it into a more smooth mesh. These heights are used by both front and back faces. For regions that need to remain flat, Neumann boundary conditions can be used. This inflation method may produce sharpness along the meshes' cross-sections. Local Laplacian smoothing methods [20, 21] are then employed to smooth the edges.

Three kinds of meshes are supported by our system. For inflated meshes, we just use the inversed heights for back faces and glue them to front faces with a small shift. For concave meshes, the heights of front faces are scaled by a scalar smaller than one and then inversed to produce a concave effect. In the case of half inflated meshes, the heights of front faces are simply set to zero. This method is simple but delivers satisfactory results in our experiments.

4.3 Texturing

Our system will generate both front and back textures for each region. We use the [x, y] coordinates as texture coordinates for front faces and attach the original image to them directly. For back faces, the system should generate proper back textures. The essential idea is that areas that are far away from the boundary of the region are more likely not reused on the back texture. Given a region, our system first selects a main color which contributes the most along the boundary. This main color is used to initialize the back textures. Our system then applies marker-based segmentation using watershed algorithm to separate areas in the original texture. The algorithm begins with the binarization of the input image. Then we apply the distance transform on the binary image. We threshold this distance image and perform some morphology operations to extract areas from the image. For each area we create a seed for the watershed algorithm by extracting its contour. Then we apply the watershed algorithm and combine adjacent areas. In order to compare their distance from the region boundary, we apply the distance transform on the whole region and use this distance map to calculate the nearest path from each area to the boundary. If the distance is smaller than a threshold (we use 5 in our implementation), then we suppose this area should be reused in the back texture.

The user can use tools provided by our system to modify the results (whether or not to reuse an area). If an area is not reused according to a user input, the pixels of this area will be filled with the main color selected at the beginning.

4.4 Shape Deformation

Shape deformations can help the user to get correct depth orders between different regions and to deform the generated models to obtain desired poses. The user can first add some point handles on the region in the 2D cartoon image. Then our system will calculate skin weights for each point handle and assign to each vertex a set of skin weights for each handle. The weights are computed using bounded biharmonic weights [22]. The method defines the weights w_j as minimizers of the Laplacian energy subject to constrains that enforce interpolation of the handles and other desirable properties, such as smoothness, shape-awareness locality and sparsity. When the user drags the point handle, the system then uses the linear blend skinning function [23] to deform the model.

4.5 Rendering

Every mesh generated by our system is attached with a texture, which is the combination of two textures, one for front faces and one for back faces. In order to render them on the mesh correctly, we assign each vertex a texture coordinate [u, v]. For front faces, these coordinates are simply the [x, y] coordinate of this vertex. For back faces, the v component in [u, v] is still the value of y, but the u component is the sum of x and 1. So any texture coordinate larger than 1 indicates this vertex belongs to some back face. When rendering the mesh, the system will use this information to decide which part of the texture to sample.

Once the mesh is generated, our system uses a tone-based shading [24] to deliver a stylized look. More complicated lighting effects are also supported. When rendering the scenes shown in Fig. 3, we employ the method in [25] to add stylized specular effects on the models.

5 Results

We tested our system on a 2.6 GHz Intel Core i7 under OS X 10.9.5, running at interactive rates. To demonstrate the versatility of our method, we selected a set of cartoon images with different types, including characters, fruits, animals, plants and so on. These are shown in Figs. 3 and 6.

In Fig. 3, we show two sample scenes, in which all of these models (except the walls) were created from cartoon images using our system. Most of them took less than 5 min to create, and some just took a few seconds, such as the oranges and mangoes. In Fig. 6, we illustrate some examples with different poses and structural complexity. In each example, we show the original cartoon image, the result of the segmentation

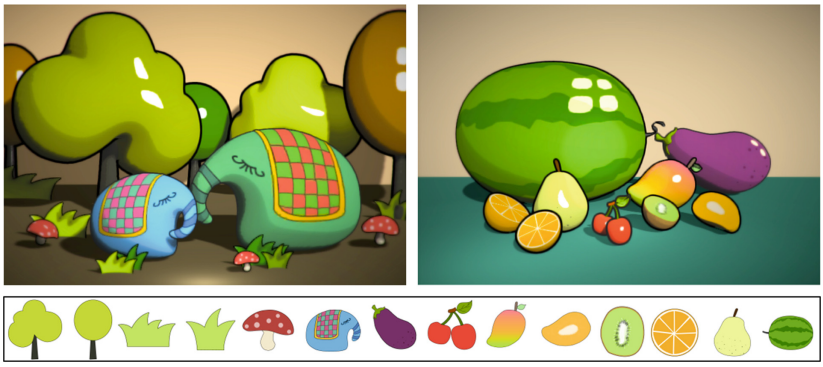

Fig. 3. Sample scenes created using our system. Most of the models took less than 5 min to create, and some just took a few seconds. Top: the scenes constructed by these models. Below: the original cartoon images used in the scene.

phase, the annotations including region operations and point handles made by the user, the front view and side view of the generated and deformed 3D models.

6 Evaluation

The results have shown that our system can handle a variety of cartoon images. We compared our system with the prior works. Sketch-based modeling systems [3–8] can only handle tubular organic shape (limited to spherical topology). Surfaces with edges or concave silhouettes (see the grasses and trees in Fig. 3) and cycles of connection curves (such as the letter "B") cannot be created by these systems. Our system is free from these limitations and can create models with almost the same silhouettes in the original images. Moreover, flat surfaces (see the mushrooms in Fig. 3, the sleeves of the Boy and Bear in Fig. 6) and concave meshes (see the grasses in Fig. 3, the ears of the Boy and Bear in Fig. 6) can also be created easily.

One of the main advantages of our system is that we don't require the user to rotate from different view during modeling. Because our system can segment the input images by their silhouettes accurately and automatically, the user doesn't need to sketch a lot like other sketch-based systems do. We created two models similar to those in [7]. With their system, one required 20 min and the other required 13 min to create, which only took 8 min and 5 min with our system.

7 User Study

To test the usability of our system, we performed an informal user study consisting of fifteen children whose average age was twelve. These children were divided into three groups, each using a different system among our system, RigMesh [8] and the system with structured annotations [7]. After a 20 min training of each system, the children

were allowed to create their own models. We also encouraged them to draw their own cartoon images with color pens and to use these paintings as the inputs of each system. Figure 4 shows a young girl was drawing a yellow star and the resulted model she created with our system which took about 3 min. We then collected their feedback of each system. Many children noted that the system with structured annotations [7] was not easy to produce desired models. They had to deform from certain basic primitives and the annotations provided by the system were somewhat complicated. The screen was easily messed up with these primitives and annotations. Some children using RigMesh struggled with creating models with holes and complained the system cannot create models with the exactly same silhouettes they sketched. In contrast, children using our system gave positive feedback of the simple and convenient annotations provided by our system. They remarked that the automatic segmentation and inflation was very helpful. Many children, including those using the other two systems, were excited with the textured models created by our system and would like to have this in the other two systems.

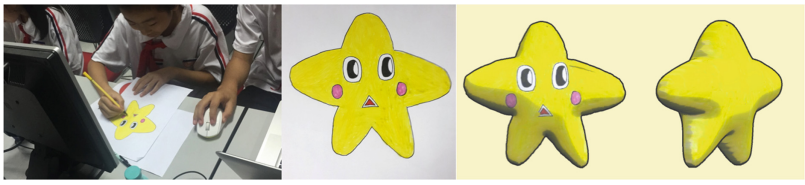

Fig. 4. A young girl was drawing a yellow star and the model she created with our system (Color figure online).

8 Conclusions and Future Work

We have implemented an interface for 3D modeling from cartoon images to simplify the modeling process, making it convenient for novice users to obtain their own 3D models. Compared with the prior sketch-based systems, our system can handle a large range of models, such as cycles of connection curves, surfaces with edges and flat surfaces. The models generated by our system also contain texture information, which can deliver a similar look to the original cartoon images as much as possible. The deformation employed by our system gives a chance for the user to get a better pose for their models. Our results show that our system makes it possible to create personalized 3D contents from cartoon images effectively.

Our system can be applied to many related areas, such as education and augmented reality. In applications such as Magicbook [26], children can see three-dimensional virtual models appearing out of the book pages through a handled augmented reality display. However, those applications demand developers to prepare models in advance, which is time-consuming. Instead, our system focuses on modeling from 2D cartoon images rapidly, and enables children to author their own models, giving full play to their creativity. We build our demo based on Vuforia SDK [27]. Figure 5 demonstrates such an ideal Augmented Reality example using our system. The child can first draw a

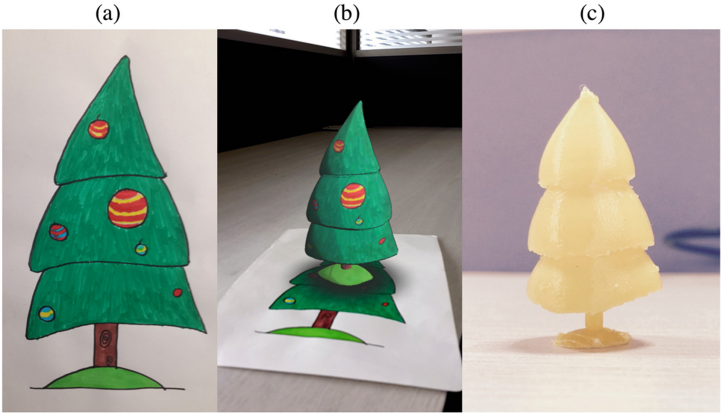

Fig. 5. Applications. (a) the picture of a hand-drawn cartoon image. (b) the scenario in augmented reality. (c) the substance printed by a 3D printer.

cartoon image, and then take a photo of it (Fig. 5a). The picture is provided as the input of our system. Next, our system generates the desired model based on a few user inputs. The model is then added to the database. Finally, children can see their own three-dimensional virtual models standing on the cartoon images through a display (Fig. 5b). The time that the entire process takes is just a few minutes. The resulted 3D models can also be printed by 3D printers (Fig. 5c).

Fig. 6. More complex models created using our system. (a) the original cartoon images. (b) the results of the segmentation phase. (c) the annotations including region operations and point handles made by the user (gray: region boundaries, red: Neumann boundary conditions, blue: point handles). (d) the front views. (e) the side views (Color figure online).

Our system still has its limitations. First, the user needs to modify region boundaries to model occluded parts. In the future, we plan to employ more intelligent methods [28, 29] to complete these regions automatically. Second, the textures of occluded areas are simply filled with the main color selected by our system. However, in some cases the method does cause some artifacts. This drawback can be fixed by using more advanced texture analysis methods such as [30].

Acknowledgments. We are grateful to all the volunteers who participated in our user studies. This work is supported in part by the National Natural Science Foundation of China (nos. 61173105 and 61373085) and the National High Technology Research and Development Program of China (no. 2015AA016404).

References

1. AUTODESK Maya (2015). http://www.autodesk.com/products/maya/overview
2. AUTODESK 3ds Max (2015). http://www.autodesk.com.cn/products/3ds-max/overview
3. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: ACM SIGGRAPH 2007 Courses, p. 21. ACM (2007)
4. Schmidt, R., Wyvill, B., Sousa, M.C., et al.: Shapeshop: sketch-based solid modeling with blobtrees. In: ACM SIGGRAPH 2007 Courses, p. 43. ACM (2007)
5. Nealen, A., Igarashi, T., Sorkine, O., et al.: FiberMesh: designing freeform surfaces with 3D curves. ACM Trans. Graph. (TOG) **26**(3), 41 (2007). ACM
6. Cordier, F., Seo, H., Park, J., et al.: Sketching of mirror-symmetric shapes. IEEE Trans. Vis. Comput. Graph. **17**(11), 1650–1662 (2011)
7. Gingold, Y., Igarashi, T., Zorin, D.: Structured annotations for 2D-to-3D modeling. ACM Trans. Graph. (TOG) **28**(5), 148 (2009). ACM
8. Borosn, P., Jin, M., DeCarlo, D., et al.: Rigmesh: automatic rigging for part-based shape modeling and deformation. ACM Trans. Graph. (TOG) **31**(6), 198 (2012)
9. Sýkora, D., Kavan, L., Čadík, M., et al.: Ink-and-ray: bas-relief meshes for adding global illumination effects to hand-drawn characters. ACM Trans. Graph. (TOG) **33**(2), 16 (2014)
10. Olsen, L., Samavati, F., Sousa, M.C, et al.: A taxonomy of modeling techniques using sketch-based interfaces. In: Eurographics State of the Art Report (2008)
11. Schmidt, R., Isenberg, T., Jepp, P., et al.: Sketching, scaffolding, and inking: a visual history for interactive 3D modeling. In: Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering. ACM, pp. 23–32 (2007)
12. Olsen, L., Samavati, F.F., Jorge, J.A.: NaturaSketch: Modeling from images and natural sketches. IEEE Comput. Graph. Appl. **31**(6), 24–34 (2011)
13. Karpenko, O.A., Hughes, J.F.: SmoothSketch: 3D free-form shapes from complex sketches. ACM Trans. Graph. (TOG) **25**(3), 589–598 (2006). ACM
14. Rivers, A., Igarashi, T., Durand, F.: 2.5 D cartoon models. ACM Trans. Graph. (TOG) **29**(4), 59 (2010). ACM
15. Sýkora, D., Ben-Chen, M., Čadík, M., et al.: TexToons: practical texture mapping for hand-drawn cartoon animations. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-photorealistic Animation and Rendering, pp. 75–84. ACM (2011)
16. Cheng, M.M.: Curve structure extraction for cartoon images. In: Proceedings of the 5th Joint Conference on Harmonious Human Machine Environment, pp. 13–25 (2009)

17. Sýkora, D., Buriánek, J., Žára, J.: Colorization of black-and-white cartoons. Image Vis. Comput. **23**(9), 767–782 (2005)
18. Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Comput. Geom. **22**(1), 21–74 (2002)
19. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) Applied Computational Geometry Towards Geometric Engineering. LNCS, vol. 1148, pp. 203–222. Springer, Heidelberg (1996)
20. Field, D.A.: Laplacian smoothing and Delaunay triangulations. Commun. Appl. Numer. Methods **4**(6), 709–712 (1988)
21. Vollmer, J., Mencl, R., Mueller, H.: Improved laplacian smoothing of noisy surface meshes. Comput. Graph. Forum **18**(3), 131–138 (1999). Blackwell Publishers Ltd.
22. Jacobson, A., Baran, I., Popovic, J., et al.: Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. **30**(4), 78 (2011)
23. Magnenat-Thalmann, N., Laperrire, R., Thalmann, D.: Joint-dependent local deformations for hand animation and object grasping. In: Proceedings on Graphics Interface 1988 (1988)
24. Gooch, A., Gooch, B., Shirley, P., et al.: A non-photorealistic lighting model for automatic technical illustration. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 447–452. ACM (1998)
25. Anjyo, K., Hiramitsu, K.: Stylized highlights for cartoon rendering and animation. IEEE Comput. Graph. Appl. **23**(4), 54–61 (2003)
26. Billinghurst, M., Kato, H., Poupyrev, I.: MagicBook: transitioning between reality and virtuality. In: CHI 2001, Extended Abstracts on Human Factors in Computing Systems, pp. 25–26. ACM (2001)
27. Developer, V.: SDK, Unity extension Vuforia–2.8 (2014)
28. Geiger, D., Pao, H., Rubin, N.: Salient and multiple illusory surfaces. In: Proceedings of the 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 118–124. IEEE (1998)
29. Orzan, A., Bousseau, A., Barla, P., et al.: Diffusion curves: a vector representation for smooth-shaded images. Commun. ACM **56**(7), 101–108 (2013)
30. Elad, M., Starck, J.L., Querre, P., et al.: Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Appl. Comput. Harmonic Anal. **19**(3), 340–358 (2005)