Vis Comput (2016) 32:491–500
DOI 10.1007/s00371-015-1082-2

ORIGINAL ARTICLE

Artistic Low Poly rendering for images

Meng Gai[1,2] · **Guoping Wang**[1,2]

Published online: 9 April 2015

Abstract This paper presents an automatic approach for generating low poly rendering of images, which is particularly popular in the recent art design community. Distinguishing from the traditional image triangulation methods for the sake of compression or vectorization, we propose some critical principles of such Low Poly rendering problem, and simulate the artists creation procedures straightforwardly. To produce the visual effects with clear boundaries, we constrain the vertices along the feature edges extracted from the input image. By employing the Voronoi diagram iteration guided by a feature flow field, the vertices in the result image well reflect the feature structure of the local shape. Moreover, with the salient region detection, we can achieve different mesh densities between the front object and the background. Some special color processing techniques are employed to make our result more artistic. Our method works well on a wide variety of images, no matter raster photographs or artificial images. Experiments show that our approach is able to generate satisfying results similar to the artwork created by professional artists.

Keywords Low poly · Non-photorealistic rendering · Image stylization · Image decomposition

✉ Meng Gai
gaimeng@pku.edu.cn

Guoping Wang
wgp@pku.edu.cn

[1] Beijing Engineering Research Center of Virtual Simulation and Visualization, Peking University, Beijing, China

[2] State Key Lab of Mathematical Engineering and Advanced Computing, Wuxi, China

1 Introduction

With the popularity of flat design, low poly style takes the fancy of more and more art designers (see Fig. 1). This design style origins from the early stage of the computer modeling, when the artists use a relatively small number of polygons to represent 3D meshes. But recently it has got new vitality in 2D illustration and graphics design. Artists use adaptive polygons (mostly triangles) to represent the objects in a image to get a specific abstract visual effect. Now It has been a new design trend in the artist community [1–3]. In this paper, we focus on generating Low Poly rendering for images automatically.

Many of the previous vector editor software like Adobe Illustrator and Corel CorelDraw can help the artists to accomplish this task. But it is obviously a tedious work that the artists must draw every single triangle by themselves. A later tool Image Triangulator App [4] provides a simpler way to generate triangle mesh. The artists can add vertices by clicking. But they must arrange every point carefully to generate the desired visual effects. So it still remains time consuming.

Recently, some applications on IOS devices can generate Low Poly images automatically, such as Trimaginator [5] and Art Camera TRIGRAFF [6]. But their effects are not satisfying enough. Sometimes the output image turns into a mess that you cannot figure the proper details out.

In the traditional field of image processing, generating triangle meshes from images has been well studied. There are many methods proposed to compress an image by decomposing it into a simplified mesh. In the community of image compression, the peak signal to noise ratio (PSNR) is usually used as a measurement of compression quality.

$$\mathrm{PSNR} = 10 \log \left(\frac{255^2}{\mathrm{MSE}} \right)$$

Fig. 1 Low Poly artwork created by an artist [1]

MSE is the mean squared error:

$$\text{MSE} = \frac{\sum_{x \in X} |\hat{I}(x) - I(x)|^2}{|X|}$$

where X is the set of pixels, $I(x)$ is the luminance value of the pixel x in the original image, while $\hat{I}(x)$ is the corresponding luminance in the reconstructed image.

As these compression methods focus more about the approximation measured by PSNR, they do not care about the mesh quality much from the aesthetic aspect. In fact, good approximation does not directly leads to good visual effects. Based on the observation on the artwork created by artists (see Fig. 2), we find several critical principles used by them which may produce high PSNR:

- The artists usually use relative regular triangles rather than extremely degenerated ones.
- The arrangement of the mesh vertices implies the structure of the objects.
- Non-uniform subdivision. The background is often abstracted into larger triangles than the front salient object.
- Color disturbance. The artists often choose various colors in the flat area to make the image stereoscopic.

Based on these observations, we develop an automatic Low Poly Rendering algorithm. We straightforwardly simulate the artists approach of creation and meet the several principles mentioned above.

1.1 Contributions

As far as we know, we are the first to focus on the Low Poly rendering problem from the artistic aspect. Our method has the following features:

Fig. 2 Some principles observed from the real artists' artwork. The first row shows the non-uniform sampling. The second row shows the tendency of the vertices reflects the object's structure

- A fully automatic approach to generate Low Poly mesh for images. User interaction is not necessary unless the users need to adjust the result mesh sampling density. Algorithm with default parameters are capable of generating satisfying results as well.
- By constraining the edge features and the technique of color picking, our result image has a clear visual effect without zigzag artifacts.
- We propose a feature flow field to guide the iteration of the Voronoi diagram. The arrangement of vertices after optimization well reflects the structure of the object.
- A non-uniform sampling strategy based on salient region detection to make the front object and the background have different refinement density.
- Our method is very fast. It only costs several seconds even on images of million pixels.

2 Related work

Much work in NPR has focused on a particular style, especially stroke-based painterly rendering [7]. Although we have not found any paper working on the specific Low Poly rendering problem. There are many previous work that have something in common with it.

2.1 Image compression

Among many classes of image compression based on adaptive sampling, triangle meshes have received considerable attention. This is the closest field related to our Low Poly rendering problem.

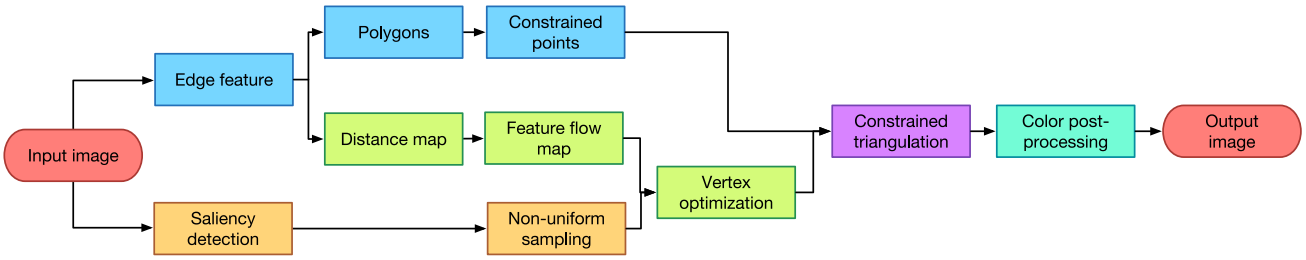


Fig. 3 Overall pipeline

As mentioned before, they only focus on the approximation of the reconstructed image to the original one, and do not care about the result mesh from the aesthetic aspects. But good approximation does not directly lead to good artistic visual effects. For example, there may exist a lot of degenerated triangles. So as an NPR (Non-photorealistic rendering) problem, we do not use the PSNR value to evaluate the result.

The adaptive thinning method [8] uses greedy point removal (GPR) scheme to produce high-quality meshes. Later the author improves this method and publishes a more powerful version called AT* [9]. AT* is the state of the art in terms of its ability to keep edge features and produce meshes with higher quality. But these adaptive thinning methods have extremely high computational and memory requirements. At the initial step, they need to compute the cost value of all pixels of the image.

Another effective mesh generation method is the error diffusion scheme (ED) [10]. It has the advantage of low computational and memory complexity. On the other hand it produces much lower quality meshes.

Adams [11] combines the two schemes above. He uses the frame work of GPR while exploiting the idea from the ED scheme in order to achieve lower computational complexity. But his method still cannot meet the rest rules of the Low Poly image.

2.2 Image vectorization

Vector graphics has been employed in a wide variety of applications due to its scalability and editability.

A subdivision-based representation is introduced in [12]. The subdivision scheme makes it support multi-resolution, and the color function is at least C^1 everywhere. However, in our Low Poly application, the continuity of the color function is not necessary.

Quad meshes based on gradient meshes form another class of representation. One of the advantages of quad meshes is its grid directions also imply the features and structures of the object. A vectorization method based on optimized gradient meshes is introduced in [13]. But it needs manual mesh initialization to assist mesh placement. This can be time consuming for an image with many features. Besides, it involves non-linear optimization, which is slow and sensitive to the initial conditions. An automatic and topology-preserving method is proposed in [14]. Its topology-preserving gradient mesh representation allows an arbitrary number of holes, and the linear operations make it much faster. But it still takes much time even for a small picture.

Ardeco [15] uses a novel unsupervised parametric segmentation method to create a vector image that approximates a bitmap image. To accelerate its convergence, they also construct a triangulation of the image. But the triangulation is only used as an intermediate tool and it does not represent the shape properly.

2.3 Image tessellation

A mosaic simulation problem is first proposed in [16]. Their idea has a strong connection with our target of vertices placement.

Some Voronoi-based methods [17,18] also give results like the Low Poly style, but they have subtle difference. Besides using polygons as primary elements rather than triangles, the polygons in the Voronoi tessellation do not adapt to the shapes and details as well as the Low Poly style, so that the objects in their result images are relatively more confusing.

3 Overall process

Figure 3 illustrates the overview of our pipeline. Notice that the colors in the diagram, respectively, show how we follow the principles mentioned before. The input image is first processed by an edge detector. The tracked edges are then approximated by polygon curves. The edge of the polygons will be the constrained edges in the final Delaunay triangulation step. The feature edges are also used to compute a distance map to generate the feature flow field. This field will guide the arrangement of vertices in the mesh. At the same time, a saliency detection process is alternative to guide a non-uniform sampling between the front object and the background. We then optimize the sample points using a centroidal Voronoi diagram weighted by the feature flow field. After some iteration of the Lloyd relaxation algorithm, the sample points are add to the constrained Delaunay triangulation with the constrained points in the previous step. Finally,

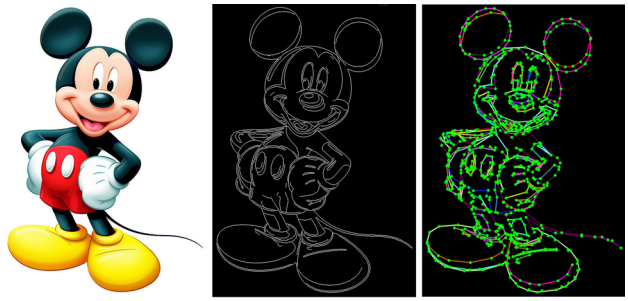


Fig. 4 **a** Original image, **b** the edge feature, **c** the simplified polygons and the constrained points

we pick the color in the triangles and make some other color post-processing to generate the final result.

4 Algorithm

4.1 Constrained edge feature

Edge feature detection is performed through the edge Ddrawing method [19], which has been proved to have better performance than the classical Canny edge detector. It can produce high-quality edge segments, which are clean, well-localized and one-pixel wide. Specifically, by connecting the successive anchor points, we can get the contiguous edge chains directly. No more extra edge tracing procedure is needed.

Similar to the idea of [20], we use these edges as constraints to generate the constrained Delaunay triangulation. But the vertices along the edges are too many. So we employ a polygon approximation algorithm to simplify the edges and leave the key points only. Although some advanced 2d shape simplification methods are proposed in recent years [21,22]. We find the classical Douglas–Peucker algorithm [23] is able to manage. But the original Douglas–Peucker algorithm will oversimplify the straight edges to few end points. So we modified it and add the edge length constrain: if an edge is longer than a minimum length, we slice it into two segments from its midpoint. We choose the minimum length same as the sampling interval L_i:

$$L_i = \eta(L_\mathrm{w} + L_\mathrm{h})$$

where L_w and L_h are the image width and height, respectively. η controls the sampling density. In our experiments, we set $\eta = 0.02$. The points in the result polygons are marked as constrained points (see Fig. 4c), while the polygonal segments between them are constrained edges. Their positions will not be moved during the latter optimization steps. Moreover, the four corner points are set to constrained points as well.

These simple but critical strategies guarantee the final image with clear feature edges and good visual effects.

4.2 Sampling based on saliency

Sometimes the artists simplify the background to emphasize the front object. To simulate this feature, we use different sample densities between the saliency region and the background. Saliency detection is a specific field in computer vision. We employ a recent method [cheng2013efficient] due to its efficiency.

The number of the sample points follows:

$$\begin{cases} N_\mathrm{s} = \lambda(N - N_\mathrm{c}) \\ N_\mathrm{b} = (1 - \lambda)(N - N_\mathrm{c}) \end{cases}$$

where N is the total number of sampling. N_c denotes the constrained point numbers in the previous step. N_s is the number of sample points in the salient region, while N_b is in the background region. λ controls the different density. When $\lambda = 1$, there will not be sampling points in the background. When $\lambda = 0.5$, the effect equals no saliency detection. In our practice, we choose $\lambda = 0.7$ as an empirical value to get desired result. The total sampling number N can be given by the user input. We also provide a default value computed by this:

$$N = \mathrm{Floor}(L_\mathrm{w}/L_i) \times \mathrm{Floor}(L_\mathrm{h}/L_\mathrm{i})$$

L_i is the sampling interval mentioned in the previous section.

In Fig. 5, we show the result of saliency detection and the effect of the parameter λ. When λ is larger, the background is more abstracted with larger triangles.

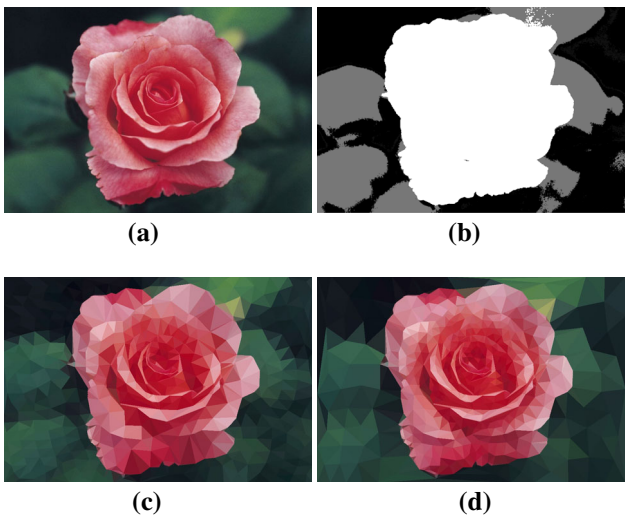


Fig. 5 Saliency detection and non-uniform sampling. **a** Input image. **b** Saliency map. **c** Result with $\lambda = 0.2$. **d** Result with $\lambda = 0.8$

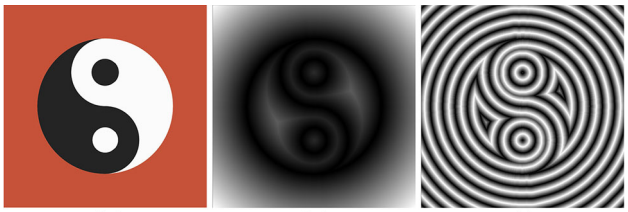


Fig. 6 The distance map and the feature flow map. **a** Input image, **b** the distance map, **c** the feature flow map

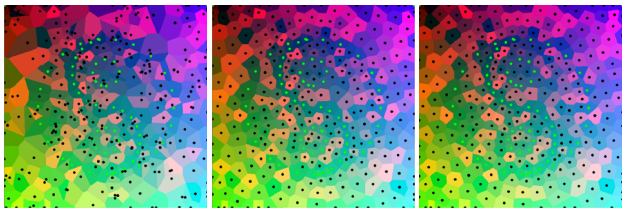


Fig. 7 The Voronoi diagram iterations. **a** Initial sampling. **b** After 5 iterations. **c** After 10 iterations

4.3 Feature flow field

In the real Low Poly artwork created by artists, the vertices placement of the mesh follow the local shape structure of the object. We propose a feature flow field to guide the optimization of the vertices positions. Similar to the idea in [16] and [24], we first compute a distance map $D(x)$. At each point x, $D(x)$ records the distance from x to the nearest edge features.

We use the Jump Flooding [25] method to solve this distance transform problem. It propagates the distance information in a jumping manner so that it can finish computing the distance map in $\log(n)$ rounds for an image of size nn, regardless of the number of starting points. Kim et al. [24] use an offset line map to guide the stippling dot placement along the feature flow, and the result shows their method has a stronger constraint on dot alignment than Hausners method [16]. Although our vertex placement target is quite similar to the image-stippling problem, we cannot use their offset line method directly. Their sampling is much denser than ours, and they take advantage of the dots density. In their iteration, they do not have to worry about the Voronoi cell crossing more than one lane. On the other hand, our Voronoi cell may be much bigger than the lane intervals. It will be difficult to deal with the cells occupied by several lanes.

Inspired by the CVT (centroidal Voronoi tessellation) steered by the image saliency map in Ardeco [15] , we propose a smoother feature flow field. And it makes the following CVT iteration problem become much easier than [24]. The value at x in the feature flow map is computed as follows:

$$F(x) = \begin{cases} \frac{255}{m} D(x) \mathrm{mod}(m), & \text{if } \frac{D(x)}{m} \mathrm{mod}(2) = 0 \\ \frac{255}{m} (1 - D(x) \mathrm{mod}(m)), & \text{if } \frac{D(x)}{m} \mathrm{mod}(2) = 0, \end{cases}$$

where m controls the width of the lanes interval in the feature flow map. We choose $m = L_\mathrm{i}/2$ in our experiments. The Distance map and feature flow map are shown in Fig. 6.

4.4 Vertex optimization

The CVT and Lloyd relaxation are often used to optimize the mesh quality. During the iteration, each seed is moved to its regions centroid. CVT approximates a Poisson-disk point distribution that the seeds can cover the space fairly. Some popular remeshing algorithms in 3D also use this method. The centroid c of a Voronoi cell is computed as follows:

$$c = \frac{\sum_\mathrm{i} w_\mathrm{i} x_\mathrm{i}}{\sum_\mathrm{i} w_\mathrm{i}},$$

where x_i denotes the pixels in the cell and w_i is the associated weight. In the basic Lloyd relaxation algorithm, $w_\mathrm{i} = 1$ for all pixel in the cell. In our approach, we set $w_\mathrm{i} = F(x_\mathrm{i})/255$. So the weights have the effect of stretching the vertices to the feature flows. Notice when x_i is near to the feature edge, $F(x_\mathrm{i})$ is very small. So the weight w_i related to x_i is really small and x_i will move away from the feature edge. This ensures that the sampling vertices will not occlude the constrained points.

Constrained points including those along feature edges and the four corner points do not change their position during the iteration. Points sampled along the boundary of the image can only move along the image boundary, or there will be a shrink effect. So in each iteration, we update the new position for all free points by the Lloyds relaxation optimization:

$$v_\mathrm{i}^{(k+1)} \leftarrow \frac{\sum_\mathrm{i} w_\mathrm{i} x_\mathrm{i}}{\sum_\mathrm{i} w_\mathrm{i}}$$

$v_\mathrm{i}^{(k+1)}$ denotes the position of vertex I in the $k+1$ iteration. In our experiment, 10 iterations are able to generate satisfying result. The process is shown in Fig. 7.

4.5 Constrained triangulation

Delaunay triangulation is often used to produce high-quality mesh. And it is the dual-mesh of the Voronoi diagram so that it can be computed easily. We use constrained Delaunay triangulation to produce the result mesh.

4.6 Color post-processing

Although we constrained the feature edge in the result mesh, there still may be zigzag effects. That is caused when a trian-

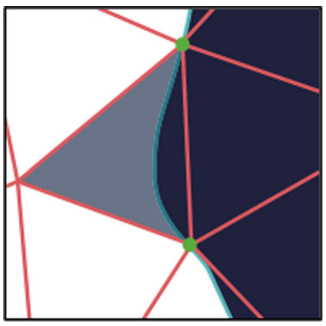


Fig. 8 The zigzag artifact of the *triangle*. The edge feature is shown in *cyan color* and the constrained points are shown in *green*

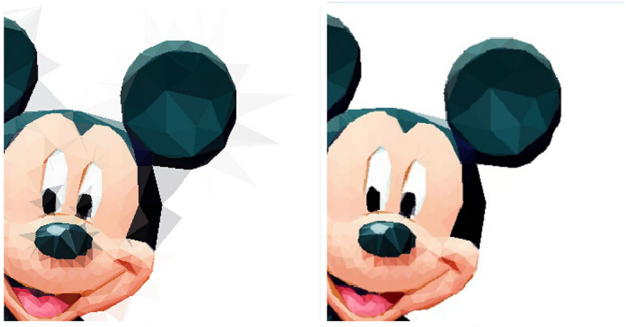


Fig. 9 Color picking. **a** The zigzag artifacts, **b** the result with our special color picking strategy

gle covers both a dark region and a light region as shown in Fig. 8. The triangle will look grey even when only a small set of dark pixels in it, while it is supposed to be in light color.

To eliminate this artifact, we sort the colors in a triangle by the L component of its Lab color space. Then we pick the median part of the colors to avoid those too dark or too light. In our experiment, we select the 40–60% part. The result with our special color picking strategy is shown in Figure 9. The zigzag artifacts are no longer exits with our technique.

To make the colors in the result image look more vivid. We apply some additional post-color processing. We add some noise less than 10 to the L component of each triangle in the Lab color space to make the flat color area full of variety. Furthermore, we raise the saturation of the result image to make its color more artistic.

5 Result

5.1 Comparison

We did some comparison test with other image triangulation methods and applications. A wide variety of images are used in our experiment, both raster photographs and artificial cartoon images.

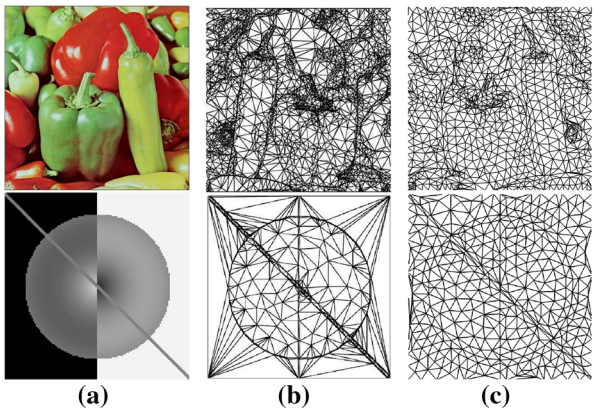


Fig. 10 Comparisons with AT method. **a** Original image, **b** mesh using AT, **c** our result

Figure 10 shows our result mesh compared with the mesh generated by AT* algorithm [9]. Our method gets more regular triangles. Notice how the vertices in our mesh follow the local feature and imply the structure of the peppers shape.

Figure 11 shows our result on a cartoon image compared with the result by an artist and the IOS application TRIGRAFF [6]. Our result constrains the feature edges of the original image and gets a close visual effect to the artists.

In Figure 12, we compare our result on the classical Lena example with the triangle mesh generated in ARDECO [15]. They also employ a saliency map-guided Lloyd relaxation and get fine mesh quality. But our result has more clear feature and better visual effect.

Some more post-processing can make the result looks more artistic. Figure 13 shows some simple post-processing in different color spaces. More complex filters can be applied to the result as well, but thats not the main problem focused in this paper.

More results are shown in Figure 15. We recommend the readers to zoom in to get better visual effects if you are reading the electronic version of this paper. Most of them are portraits as the Low Poly style is mostly used on portrait in real practice. Although we have not employed any face reorganization algorithms, the results are pretty satisfying.

We test our algorithm on an Intel Core i5 2300 PC with 6GB memory. Detailed performance about some examples is shown in Table 1. The main time cost is the CVT iteration, because we have only implemented the CPU computation version of jump flooding method. The speed can be further improved if we employed the GPU version.

5.2 User study

To evaluate the artistic verisimilitude of our method, we choose 40 images for a user study. 20 of them are created by artists manually, while the other 20 are produced by our

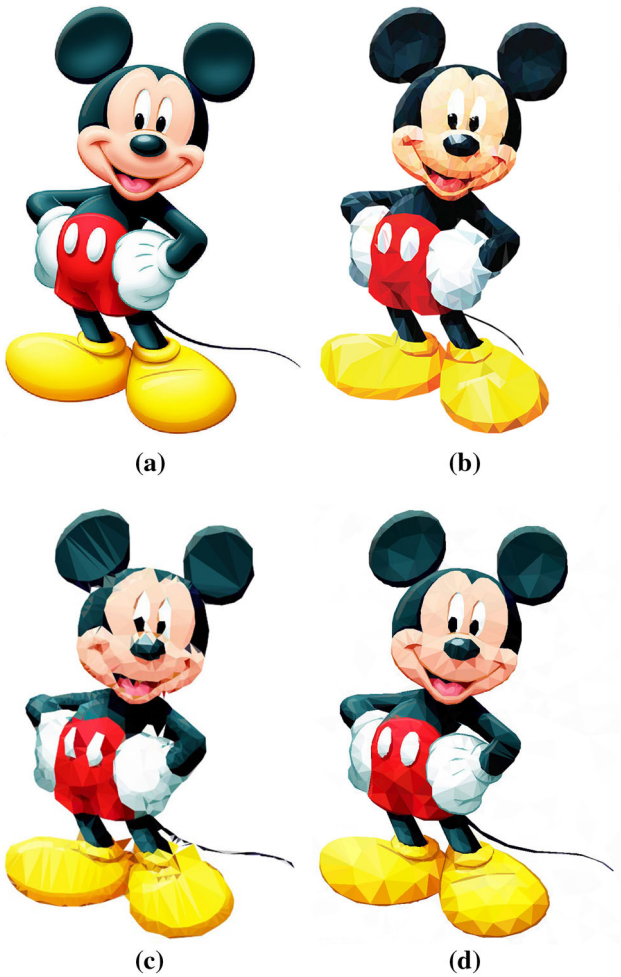


Fig. 11 Comparison with artists artwork and TRIGRAFF. **a** Original image, **b** Low Poly artwork by an artist [2], **c** result by TRIGRAFF [6], **d** our result

Fig. 12 Comparisons with the triangulation in ARDECO. **a** ARDECO. **b** Our result

algorithm automatically. We ask the users to evaluate the images by scores from 0 to 5.0 means it is obviously generated by computer while 5 means it is definitely created by a human artist. We put the test online and received 46 valid feedbacks in total. There are 6 users who are skilled designers with art design experience for more than two years. We divide them as the professional group. And the rest users are in the non-professional group.

Fig. 13 Additional color post-processing to achieve more artistic effects. **a** Original image, **b** histogram equalization in the RGB color space, **c** histogram equalization in the YCrCb color space

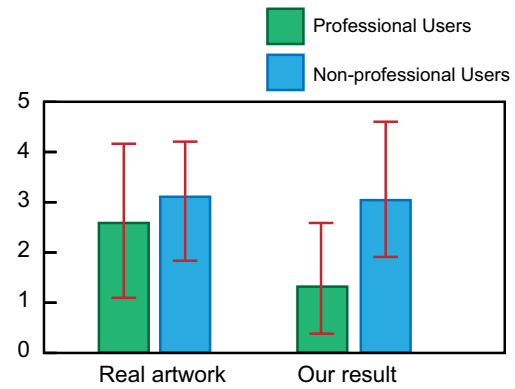


Fig. 14 User study result

The average scores and standard deviations are shown in Figure 14. From the figure we can see that our results get the similar scores to the real artwork in the non-professional group. Although we only get 6 professional users, the scores given by them show some significant difference. Generally speaking, the non-professional users cannot tell much difference from our results with the real artwork. But our results are still obvious to those experienced designers.

To evaluate the result more quantitatively, we employ the independent two-sample t test of the statistical hypothesis test theory. We compute the average score of each image. The average scores of our results are treated as sample X_1, while those of the real artwork are in sample X_2. We assume the two distributions have the same variance. (We can also assume the two distributions have unequal variances, which is known as Welchs t test. But the result shows no big difference. Since the assumption of equal variance is reasonable here, we choose the simpler model to explain.) We choose X_1 and X_2 share the same distribution as the null hypothesis H_0. The t statistic to test whether the means are different can be calculated as follows:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_w} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}},$$

Table 1 Time cost in each step

Test case	Image size	Feature flow	Saliency detection	CVT	Total
Micky (Fig. 11)	558 × 800	0.295	0.531	2.491	4.791
Yin-Yang (Fig. 6)	600 × 600	0.221	0.146	2.038	3.624
Flower (Fig. 5)	954 × 637	0.503	0.307	3.713	5.573
Peppers (Fig. 10)	512 × 512	0.175	0.175	1.557	3.437
Lena (Fig. 12)	344 × 344	0.07	0.051	0.652	1.844

The times is in seconds

Fig. 15 More results

where

$$S_{\mathrm{w}} = \sqrt{\frac{(n_1 - 1)S_{X_1}^2 + (n_2 - 1)S_{X_2}^2}{n_1 + n_2 - 2}}$$

$\bar{X_1}$ and $\bar{X_2}$ are the mean values of the two samples X_1 and X_2, respectively. S is the sample standard deviation. N is the sample size. In our case, $n_1 = n_2 = 20$. The calculated P value is 0.383142786, larger than the usual statistical significance $\alpha = 0.05$. So the hypothesis H_0 is supported and two set of data share the same distribution. This proves that our result have the same effects with the real artwork for common users from the statistical theory.

We also do the *t* test for the professional group only. The *P* value is 7.69492E-06 < 0.05. This means the images created by our algorithm and artists have significant differences for the experienced users.

6 Limitation and future work

Since our constrained points are gained from the first step of feature extraction, the final result gets a lot of influence by the quality of extracted feature edges. If the input image is full of delicate details, there will be a large amount of constrained points. As a result, the output image does not get much abstraction and makes no big difference with the original input image. But in our test, only a few images have

this problem. In most cases, our method is able to generate a satisfying result. Generally, our methods works better on artificial cartoon images than raster photos, for their clear feature edges.

Some advanced image segmentation or edge detection methods [26] may help to solve this problem. We have not used any region information explicitly by now. Moreover, some preprocessing techniques such as Line emphasis and color adjustment will also contribute [27,28].

Besides, inspired by parametric meta-filter transfer [29], learning the scheme of an existing Low Poly Image created by the artist and apply it to a new image is another way to generate convincing artwork.

7 Conclusion

In this paper, we have presented an automatic method to generate Low Poly style images. To our knowledge, we are the first to study this problem from the artistic aspect. We conclude several critical rules used by artist and straightforwardly simulate their procedure of creating such artwork.

By constraining the feature edges, our result image have a clear visual effect. With the non-uniform sampling based on saliency detection, the result may have different refinement between the front object and the background. Moreover, our feature flow-guided iteration technique confers our method the ability of arranging the points by the shape feature. These features cannot be achieved by traditional triangulations methods from the aspect of image compression or vectorization.

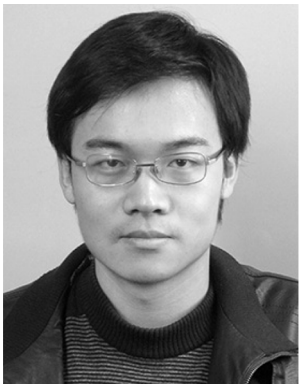
The experiment shows our method is efficient and effective. A wide variety of images can be processed into satisfying results. According to the user study data, we use the t test of the statistical hypothesis test theory to prove that our result has almost the same visual effects for most common users.

Acknowledgments This research was supported by Grant Nos. 61421062, 61170205, 61232014, 61472010 from National Natural Science Foundation of China. Also was supported by Grant No. 2012AA011503 from The National Key Technology Research and Development Program of China.

References

1. Bitencourt, B.: https://dribbble.com/Bitencourt (2014)
2. Bhishek Aggarwal a.k.a AbhiKreationz. https://www.behance.net/abhikreationz (2014)
3. Mordi, L.: https://www.behance.net/mdlv (2014)
4. Image triangulator app: http://www.conceptfarm.ca/2013/portfolio/image-triangulator/ (2014)
5. Trimaginator: https://itunes.apple.com/cn/app/trimaginator/id874969053 (2014)
6. Art camera trigraff: https://itunes.apple.com/cn/app/art-camera-trigraff/id646603902 (2014)
7. Kyprianidis, J., Collomosse, J., Wang, T., Isenberg, T.: A taxonomy of artistic stylization techniques for images and video, State of the'art' (2012)
8. Demaret, L., Dyn, N., Floater, M.S., Iske, A.: Adaptive thinning for terrain modelling and image compression. In Advances in multiresolution for geometric modelling, pp. 319–338. Springer (2005)
9. Demaret, L., Dyn, N., Iske, A.: Image compression by linear splines over adaptive triangulations. Signal Process. **86**(7), 1604–1616 (2006)
10. Yang, Y., Wernick, M.N., Brankov, J.G.: A fast approach for accurate content-adaptive mesh generation. Image Process. IEEE Trans. **12**(8), 866–881 (2003)
11. Adams, M.D.: A flexible content-adaptive mesh-generation strategy for image representation. Image Process. IEEE Trans. **20**(9), 2414–2427 (2011)
12. Liao, Z., Hoppe, H., Forsyth, D., Yu, Y.: A subdivision-based representation for vector image editing. Vis. Comp. Graph. IEEE Trans. **18**(11), 1858–1867 (2012)
13. Sun, J., Liang, L., Wen, F., Shum, H.-Y.: Image vectorization using optimized gradient meshes. In: ACM Transactions on Graphics (TOG), vol. 26, p. 11. ACM (2007)
14. Lai, Y.-K., Hu, S.-M., Martin, R.R.: Automatic and topology-preserving gradient mesh generation for image vectorization. In: ACM Transactions on Graphics (TOG), vol. 28, p. 85. ACM, (2009)
15. Lecot, G., Levy, B.: Ardeco: Automatic region detection and conversion. In Proceedings of the 17th Eurographics conference on Rendering Techniques, pp. 349–360. Eurographics Association (2006)
16. Hausner, A.: Simulating decorative mosaics. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 573–580. ACM (2001)
17. Chen, Z., Xiao, Y., Cao, J.: Approximation by piecewise polynomials on voronoi tessellation. Graphical Models (2014)
18. Faustino, G.M., De Figueiredo, L.H.: Simple adaptive mosaic effects. In Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on, pp. 315–322. IEEE (2005)
19. Topal, C., Akinlar, C.: Edge drawing: a combined real-time edge and segment detector. J. Vis. Commun. Image Represent. **23**(6), 862–872 (2012)
20. Swaminarayan, S., Prasad, L.: Rapid automated polygonal image decomposition. In Applied Imagery and Pattern Recognition Workshop, 2006. AIPR 2006. 35th IEEE, pp. 28–28. IEEE (2006)
21. Garland, M., Zhou, Y.: Quadric-based simplification in any dimension. ACM Trans. Graph. (TOG) **24**(2), 209–239 (2005)
22. De Goes, F., Cohen-Steiner, D., Alliez, P., Desbrun, M.: An optimal transport approach to robust reconstruction and simplification of 2d shapes. In Computer Graphics Forum, vol. 30, pp. 1593–1602. Wiley Online Library (2011)
23. Hershberger, J., Snoeyink, J.: An o (n log n) implementation of the douglas-peucker algorithm for line simplification. In Proceedings of the tenth annual symposium on Computational geometry, pp. 383–384. ACM (1994)
24. Kim, D., Son, M., Lee, Y., Kang, H., Lee, S.: Feature-guided image stippling. In: Computer Graphics Forum, vol. 27, pp. 1209–1216. Wiley Online Library (2008)
25. Rong, G., Tan, T.-S.: Jump flooding in gpu with applications to voronoi diagram and distance transform. In: Proceedings of the 2006 symposium on Interactive 3D graphics and games, pp.109–116. ACM (2006)
26. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Pattern Anal. Mach. Intell. IEEE Trans. **33**(5), 898–916 (2011)

27. Zang, Y., Huang, H., Li, C.-F.: Artistic preprocessing for painterly rendering and image stylization. Vis. Comp. pp. 1–11 (2013)
28. Li, X.-Y., Gu, Y., Hu, S.-M., Martin, R.R.: Mixed-domain edge-aware image manipulation. IEEE Trans. Image Processi. Publ. IEEE Signal Process. Soc. **22**(5), 1915–1925 (2013)
29. Huang, S.-S., Zhang, G.-X., Lai, Y.-K., Kopf, J., Cohen-Or, D., Shi-Min, H.: Parametric meta-filter modeling from a single example pair. Vis. Comp. **30**(6–8), 673–684 (2014)



Meng Gai is currently a Ph.D. student in Electronics Engineering and Computer Science, Peking University. He received his B.S degree from Peking University in 2011. His research interests include image and geometry processing.

Guoping Wang received his B.S. and M.S. degree from Harbin Institute of Technology, Harbin, China, and Ph.D. degree from Fudan University, Shanghai in 1987, 1990 and 1997, respectively, all in Mathematics. He was an Associate Professor of the Department of Computer Science and Technology in Peking University, Beijing, from 1999 to 2002, and has served as the Director of HCI and Multimedia Laboratory and a Professor of the School of Electronics Engineering and Computer Science, Peking University, Beijing, since 2002. Now he serves as the Deputy Director of the Institute of Software in Peking University. He is also the director of BERCVSV (Beijing Engineering Research Center of Virtual Simulation and Visualization). His research interests include computer graphics, virtual reality, geometrical modeling, CAD, and human–computer interaction.