Vis Comput (2011) 27: 187–198
DOI 10.1007/s00371-010-0522-2

ORIGINAL ARTICLE

Selective image abstraction

Lin Cong · Ruofeng Tong · Jinxiang Dong

Published online: 3 August 2010

Abstract We present a novel and convenient method for producing selective stylized simplification of images. The user uses a brush to interactively mark certain areas of the input image which are to be left unaltered. Boundaries of these areas are then automatically optimized to underlying object boundaries in the image. Our method then performs stylized simplification of the unmarked areas, while preserving the marked areas. The method ensures a smooth transition between stylized and unaltered regions to leave a *mixed reality* image which combines the real and the abstract. Stylized simplification is performed using nonlinear diffusion, which can generate sophisticated results. We modify the classic model of nonlinear diffusion to incorporate bilateral filtering; we apply diffusion speed control of each pixel based on the user's input. The level of simplification can be controlled intuitively based on the diffusion time; another parameter controls the abstraction style, giving a simple and intuitive user interface. Our contributions include a simple-to-use method to produce a novel NPR style and a modified nonlinear diffusion model suited to this selective stylized simplification task. Experimental results show that the final mixed reality results are harmonious.

Keywords Selective abstraction · Nonlinear diffusion · Image generation · Mixed reality

Nonphotorealistic rendering (NPR) is an area of computer graphics enabling a wide variety of expressive artistic styles such as painting, drawing, technical illustration, and cartoons. Image and video-based abstraction and stylization are important branches of NPR. Abstraction may be defined as a process of generalization by reducing the information content in an image, typically in order to selectively emphasise information which is relevant for a particular purpose. As noted in [13], visual communication is not always most effectively achieved using realistic images: simplification of objects or exaggeration of features can aid comprehension. In a painting, for example, the artist may use more strokes to paint some parts while simplifying others, in order to influence the viewer's perception. Selective abstraction of some regions of an image can help the user to focus on particular parts of the image, and avoid distractions in the background. In an extreme case, a realistic object in a stylized background will generate a distinctive artistic style.

L. Cong (✉) · R. Tong · J. Dong
Department of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China
e-mail: jeffersoncong@gmail.com

1 Introduction

Mixed reality video, containing a mixture of realism and non-realism, has been widely adopted for movie and television productions such as Space Jam. The principal actors are filmed against a green screen background; matting allows an alternate background video to contain realistic or abstract scenes as well as cartoon characters, for example. While this technique is widely used, it is not a particularly convenient way to construct mixed reality scenes.

We thus give here an alternative, easy-to-use, method for selective image abstraction. It provides a simple and convenient interface for users, who simply select the areas of the input images to be retained as realistic using a 'brush'. The boundaries of these areas are then automatically tidied to underlying object boundaries in the image. Users can control the abstraction level by tuning the iteration time thanks

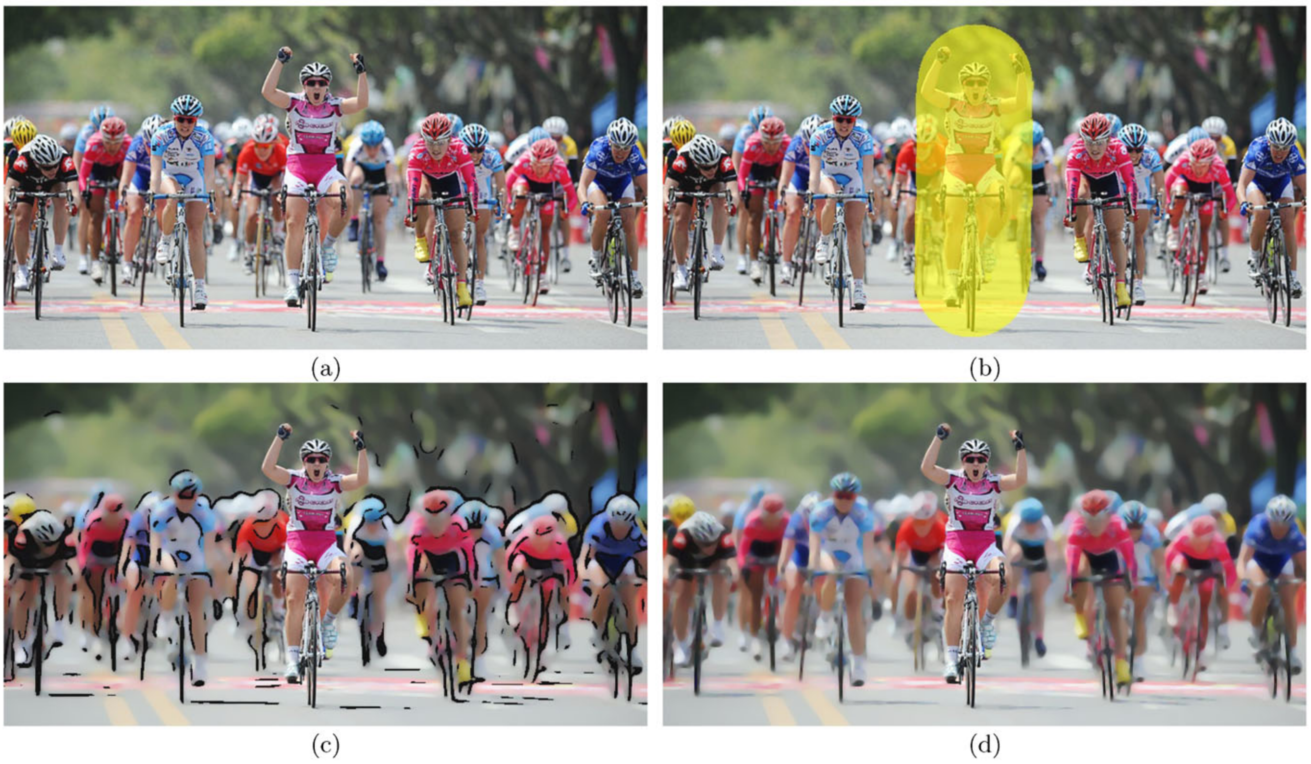


Fig. 1 (**a**) Original image, (**b**) user selection of realistic area, (**c**) result with line drawings, (**d**) result without line drawings

to a slider. Higher abstraction style value will preserve the strong edges well while lower value can produce more exaggerated output. Our tool then abstracts the background while preserving the realistic parts in such a way that these merge smoothly. Users can freely choose which parts of the image should be realistic allowing them to convey a personal understanding of the input image—different users may have different ideas for an identical image and wish to produce different abstraction results. Figure 1 shows one example produced by our method: the original image, the user input, and its corresponding selective abstraction results.

Nonlinear diffusion filtering has been proven to be able to generate sophisticated stylization results [9, 21]. Our abstraction strategy is also based on partial differential equations (PDEs), employing a combination of nonlinear diffusion and bilateral filtering. The diffusion model we use was proposed by Alvarez, Luis, and Lions [1]. We modify it to include a bilateral filtering term to reduce sensitivity to image noise. Moreover, we apply diffusion speed control of each pixel based on the user's input and add an abstraction style control parameter to the model, which makes it suit to our application.

Our main contribution is a simple-to-use method to produce a novel NPR style which merges the realistic and the abstract in a single mixed reality image. We also propose a modified nonlinear diffusion model suited to this selective image abstraction task.

Related work is discussed in Sect. 2. Section 3 explains our method, while the user interface is discussed in Sect. 4. Experimental results and a user study are to be found in Sect. 5, while limitations and future work are considered in Sect. 6.

2 Related work

Previous work on image and video based abstraction can be divided into three classes:

The first class of methods relies on the result of mean shift image segmentation [6]. DeCarlo and Santella [7] propose a hierarchical image abstraction system based on eye-tracking data. Colored regions are the leaf nodes of their hierarchy, which selectively simplifies regions guided by eye-tracking data. They evaluated their eye movement model further in [17], while this eye-tracking based method is sophisticated, it is rather impractical. Other work of a similar kind [5, 19, 20] has similar limitations due to the use of mean shift segmentation, which produces imprecise region boundaries as a result of performing density estimation in a high dimensional space. These imprecise boundaries may generate disharmonious mixed reality scenes. Zhang et al. "trapped ball segmentation method" [22] may produce more precise segmentation results, however, it cannot control the abstraction level of each pixel. Moreover, the rough segmentation often results in complex region boundaries, which requires additional smoothing to support stylistic image abstraction. In contrast, our method does not require any additional processing while enables abstraction speed control of each pixel, which can guarantee the smooth transition between the realistic and abstract.

The second class of methods is based on filtering. Fischer, Bartz, and Straßer [8] present an approach for generating stylized images by reducing the visual realism of both real images and virtual graphical objects in video see-through systems. Bilateral filtering [18] and Canny edge detection [3] are applied to perform region smoothing and line drawing, respectively. The advantages of the bilateral filter are also used by Winnemöller, Holger, and Olsen [21] in their method of image and video abstraction. They employ a difference-of-Gaussian edge detector as a basis for line drawing, and achieve more highly stylized results with performing luminance quantization. The isotropic bilateral filter approach has been improved by Kang, Lee, and Chui, who use edge tangent flow [10] to control the bilateral filtering strategy [11]. The methods discussed above simplify the color of image, however, they do not simplify the region boundaries. To perform region boundary simplification, Kang and Lee adopt constrained mean curvature flow and shock filtering in another paper [9]. Our method simplifies the color and the region boundaries simultaneously. In addition, compared to Kang and Lee's method, which uses a shock filter for edge enhancement after several iterations of mean curvature flow, our method provides edge enhancement as an integral part of the mechanism and does not require a separate processing filter.

The third class of method is due to Orzan et al. [13], whose image abstraction system is based on a Canny edge detector and gradient reconstruction method with poisson equation used for color reconstruction. However, this method still does not simplify the region boundaries while the multiscale edge detection is computational expensive. Moreover, the difference-of-Gaussian edge detector we use is cheaper to compute and is more effective than Canny's method in terms of creating stylistic illustrations.

In a summary, there is considerable previous work on image abstraction. However, depiction using a mixture of realism and abstraction as a *mixed reality* remains underexplored, which makes our work meaningful.

To perform image abstraction, we adopt nonlinear diffusion, which is mainly used for image denoising in the previous work. Perona and Malik [15] first attempted to derive a model for image denoising that incorporates local information from an image within a PDE framework. They proposed a nonlinear diffusion model in order to avoid the blurring problems presented by linear diffusion models [12]. The Perona–Malik model has been refined further by Catté et al. [4]. An interesting variation of the Perona–Malik model was proposed by Alvarez et al. [1], who considered a class of nonlinear parabolic PDEs for image filtering. This model is powerful enough to smooth regions while preserving edge information. However, it is hard to generate stylized results with this model due to the edge enhancing item. Moreover, we wish to control the diffusion speed of each pixel based on the user input to produce the selective abstraction results. Thus, we propose a new model, which suits to our application.

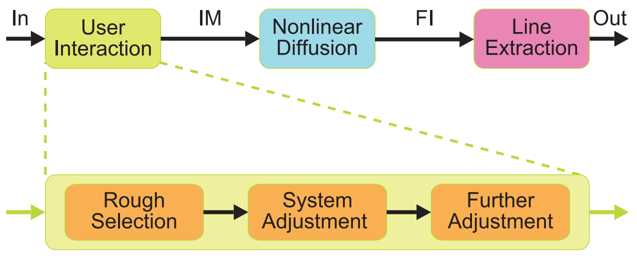


Fig. 2 Procedure to generate the realism and nonrealism mixture scene. "IM" represents the importance map. "FI" represents filtered image

To find good boundaries from user input which coarsely selects regions of interest, we use *Grab Cut* [16], an iterative image segmentation technique based upon the graph cut algorithm [2]. This reduces the work needed by the user to select the regions to be retained, making our tool easier and quicker to use. The original user interaction of Grab Cut is to drawing a rectangle around the desired foreground, which is easy to use but can generate poor results. Our system provides a brush tool, which can be used to mark the unknown area more precisely. To achieve better results, we make a further modification to grabcut based on local features rather than global ones, which both accelerates the algorithm and improves the extraction results.

3 Method

3.1 Overview

We next briefly introduce our procedure for generating a mixed reality scene (see Fig. 2).

The input to our method is an image. Users are able to use a "brush" tool to freely select the areas that must be preserved in the final image. Iteration time, abstraction style are also controlled by users. The user selection is then refined by the Grab Cut foreground extraction algorithm. The adjusted result can be further modified by the user if desired and again optimized by the tool. These regions are then used to generate a smooth importance map over the input image using series of dilations. After that, our modified nonlinear diffusion model is applied to selectively abstract the image according to the smooth importance map, which ensures a smooth transition between the unaltered and abstracted regions. Finally, line drawings are superimposed on the image to give the final mixed reality scene. We'll focus on the newly proposed nonlinear diffusion model in this section. The details of automatic boundary tidying, smooth transition and line extraction will be discussed in Sect. 6.

3.2 Image abstraction by nonlinear diffusion

We next consider the prior models used by nonlinear diffusion filters and the modifications we have made to them for our application.

Various tasks like image segmentation, multiscale image representation, and image restoration require smoothing, which can be posed as problems involving parabolic PDEs. Nonlinear diffusion filters are an interesting class of parabolic equations which are relevant to both scale-space and restoration processing.

Such filters have the general advantage that they can smooth regions while preserving or even enhancing image features such as edges. Solution of these models is usually performed iteratively and incrementally, causing gradual changes which can be intuitively controlled, which is important to our application.

Let us denote the input image by $u(\mathbf{x})$, where $\mathbf{x}$ is a 2D vector which denotes the pixel location. For a color image, e.g., in RGB space, the 3 channels are represented by separate functions

$$u_R(\mathbf{x}), \quad u_G(\mathbf{x}) \quad \text{and} \quad u_B(\mathbf{x}).$$

The classic nonlinear parabolic differential equation proposed by Alvarez et al. [1] is of the form:

$$\begin{aligned} &\frac{\partial u}{\partial t} - g\big(|G * \nabla u|\big)|\nabla u|\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) = 0, \\ &u(\mathbf{x}, 0) = u_0(\mathbf{x}), \end{aligned} \tag{1}$$

where ∇ and $\nabla\cdot$ represent the gradient and divergence operators, respectively. u_0 means the original image. The term $|\nabla u|\nabla \cdot (\nabla u/|\nabla u|)$ diffuses u in the direction orthogonal to its gradient ∇u and prevents diffusion in the direction of ∇u. The term $g(|G * \nabla u|)$ is used for edge enhancement and it controls the speed of the diffusion. g is a nonincreasing function, satisfying

$$g(x) \to 0 \quad \text{as} \quad x \to \infty$$

and

$$g(x) \to 1 \quad \text{as} \quad x \to 0.$$

G is a smoothing kernel (for instance, a Gaussian). If ∇u has a small mean in a neighborhood of a pixel, this pixel is considered to be an interior pixel of a smooth region of the image and a high rate of diffusion is applied there. Otherwise, the pixel is considered to be an edge pixel and the rate of diffusion speed is decreased.

Our first modification to this model is to filter the input image before calculating the gradient $|\nabla u|$. This is used to reduce the sensitivity to image noise. We can then replace the argument to the function g, using $g(|\nabla(G * u)|)$. However, this approach presents at least one drawback: Gaussian smoothing not only smooths the noise, but also smoothes the edges. As a result, we employ a combination of nonlinear diffusion and bilateral filtering, which can smooth the image while preserving strong edges.

The bilateral filter is a nonlinear filter introduced by Tomasi and Manduchi [18]. The basic underlying idea is to combine domain and range filtering; in other words, to take into account how similar intensity values of pixels are, as well as how close the pixels are in space. The method can be stated as

$$\begin{aligned} &B(u(\mathbf{x})) \\ &\quad = \frac{1}{k}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G_{\sigma_s}(\mathbf{p}, \mathbf{x}) G_{\sigma_r}\big(u(\mathbf{p}), u(\mathbf{x})\big) * u(\mathbf{p})\, d\mathbf{p}, \end{aligned} \tag{2}$$

where k is a normalization constant given by

$$k = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G_{\sigma_s}(\mathbf{p}, \mathbf{x}) G_{\sigma_r}\big(u(\mathbf{p}), u(\mathbf{x})\big)\, d\mathbf{p}. \tag{3}$$

Here, G_{σ_s} is a spatial Gaussian:

$$G_{\sigma_s} = e^{-(|\mathbf{p}-\mathbf{x}|)^2/2\sigma_s^2} \tag{4}$$

while G_{σ_s} is a range (intensity) Gaussian:

$$G_{\sigma_r} = e^{-(|u(\mathbf{p})-u(\mathbf{x})|)^2/2\sigma_r^2}. \tag{5}$$

The variances σ_s and σ_r determine the spatial kernel size and range kernel size, respectively.

In our approach, we now simply apply a bilateral filter in place of the Gaussian filter in function g, so our model is

$$\begin{aligned} &\frac{\partial u}{\partial t} - g\big(\big|\nabla B(u)\big|\big)|\nabla u|\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) = 0, \\ &u(\mathbf{x}, 0) = u_0(\mathbf{x}). \end{aligned} \tag{6}$$

The bilateral filter has proven to be very useful for denoising and edge enhancement, but it is computational expensive. Fortunately, this limitation has been addressed by several authors like Paris and Durand [14], who show how to perform fast bilateral filtering. Our implementation is based on their fast kernel which works well. We set the parameters σ_s and σ_r to 16 and 0.1, respectively. We would like to emphasize that the intensity value is scaled to [0.0, 1.0] when performing bilateral filtering.

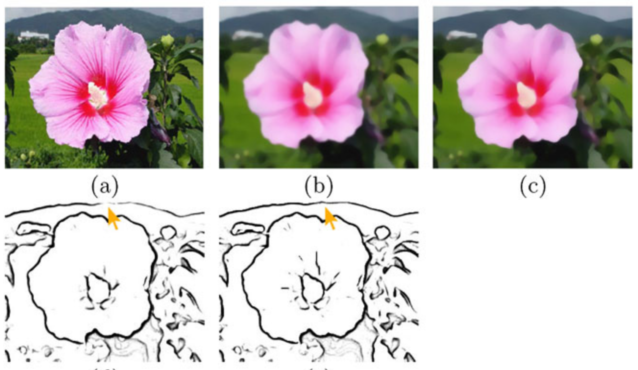


Fig. 3 Comparison of edge preserving results produced by different models: (**a**) Original image; (**b**) result of the original model in (1) after 20 iterations; (**c**) result of our modified model in (6) after 20 iterations; (**d**) DOG result of (**b**); (**e**) DOG result of (**c**). Our modified model preserves edges better

The effectiveness of our modified model is shown in Fig. 3. We apply DOG (Difference of Gaussian) to detect the images produced by the original model (see (1)) and our modified one (see (6)). Figure 3(d) shows the edge detection result of the image smoothed by the original model: the line on the top almost breaks. Based on our experimental results, this kind of situation happens more frequently in the original model than in ours. The DOG result of our model is presented in Fig. 3(e), which preserves edges better.

3.3 Speed control

The user uses a brush tool to selectively preserve areas of interest (important areas) in the input image. The output of the user interaction is an importance map $M(\mathbf{x})$, which contains the importance values in the range [0.0, 1.0] for each pixel in the image. Details of how this importance map is constructed are given later in Sect. 4.

This importance map is used to control local diffusion speed. Following Kang and Lee [9] who employ a speed function to control the evolution speed of mean curvature flow in related work, we define a speed control function $s(\mathbf{x})$ which maps importance values into corresponding diffusion speeds. If $\mathbf{x}$ has a lower importance value, this pixel is considered less important, so diffusion should be stronger. $s(\mathbf{x})$ must thus be a nonincreasing function; we require $s(\mathbf{x})$ to lie in the range [0.0, 1.0].The mapping should also be smooth. Thus, we use a linear function of the following form:

$$s(\mathbf{x}) = 1 - M(\mathbf{x}). \tag{7}$$

This functional form works well in practice. This diffusion speed control function is incorporated into the model by now setting:

$$\begin{aligned} &\frac{\partial u}{\partial t} - s(\mathbf{x})\cdot g\big(\big|\nabla B(u)\big|\big)|\nabla u|\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right) = 0, \\ &u(\mathbf{x},0) = u_0(\mathbf{x}). \end{aligned} \tag{8}$$

3.4 Final model

We now discuss our final model. The form chosen for function g is as follows:

$$g(x) = (1-w) + w\Big/\left(1+\frac{x^2}{\lambda^2}\right) \tag{9}$$

where w is in the range of [0.0, 1.0] and $\lambda = 0.01$. If $w = 0$ and $s(\mathbf{x}) = 1$, the model becomes the mean curvature evolution model which can produce exaggerated stylized results as used in Kang and Lee's method [9]. If $w = 1$, the model preserves edges well and the results will be less stylized. Thus, w can also be used to control the desired abstraction style, in addition to the iteration time. The final proposed model is therefore:

$$\begin{aligned} \frac{\partial u}{\partial t} &= s(\mathbf{x})g\big(\big|\nabla B(u)\big|\big) \\ &\quad\times\left(\big(1-h\big(|\nabla u|\big)\big)\triangle u + h\big(|\nabla u|\big)|\nabla u|\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right)\right), \\ s(\mathbf{x}) &= 0.1\big(1-M(\mathbf{x})\big)^2 + 0.9\big(1-M(\mathbf{x})\big), \\ g\big(\big|\nabla B(u)\big|\big) &= (1-w) + \frac{w}{\left(1+\frac{|\nabla B(u)|^2}{\lambda^2}\right)}, \\ u(\mathbf{x},0) &= u_0(\mathbf{x}) \end{aligned} \tag{10}$$

where $h(x)$ is a binary function:

$$h(x) = \begin{cases} 0 & \text{if } x \le \delta \\ 1 & \text{if } x \ge \delta \end{cases} \tag{11}$$

and δ is a threshold of gradient value, which is set to 1 in our implementation. It is used so that if the local gradient of a pixel is small, then h is set to 0 and the equation becomes the isotropic diffusion function. However, if the gradient is large we get a variation of the classic anisotropic diffusion function. Thus, if $h = 0$, we simply calculate the value of a pixel as the weighted sum of its four-connected neighbors and itself; If $h = 1$, the pixel is diffused in the direction orthogonal to its gradient ∇u and does not diffuse at all in the direction of ∇u.

Compared with the original model, the model with the threshold smooth the inside region more effectively as shown in Fig. 4. The result generated by our final model contains less details than the one generated by the anisotropic model (see (8)). The main edges are protected as well as the

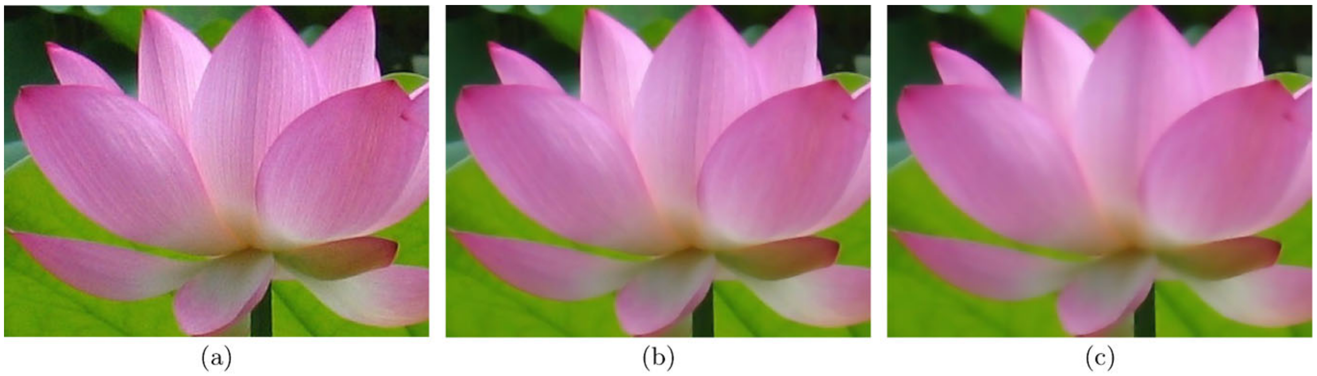


Fig. 4 Comparison of the final model and the original model; (**a**) original image; (**b**) the image smoothed by the model without the threshold (only smoothed by the anisotropic model in (8) of our paper); (**c**) image smoothed by our model with the threshold (smoothed by the mixed model in (11)); Both (**b**) and (**c**) are smoothed by 5 iterations

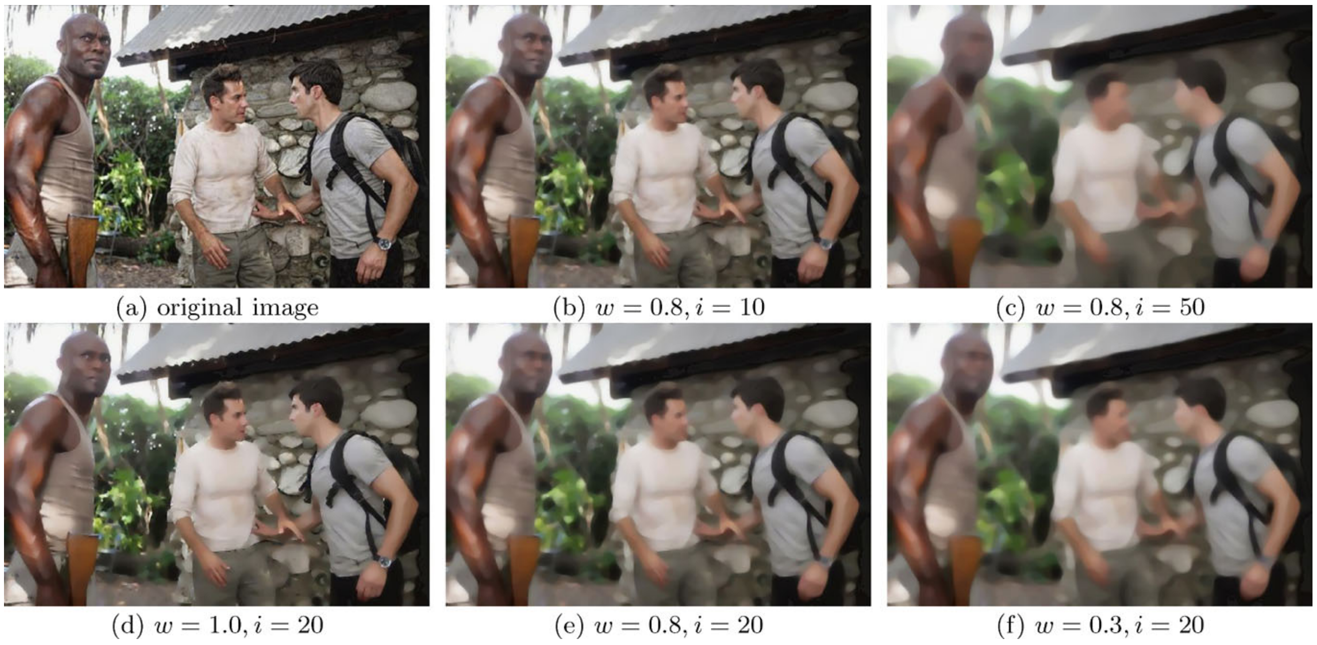


Fig. 5 Abstraction style and level controlled by w and iteration time (i)

original method. Moreover, the final mixed model is faster than the anisotropic one because some of the anisotropic blurrings are performed with isotropic blurring, which is more computational effective. We have tested these two models over 100 images with the same iterations. The time used by the anisotropic model is about 1.5 times of our final method.

In summary, the model proposed here can be used iteratively to generate stylized results while the diffusion speed of different pixels can be intuitively controlled by the user input. Unlike previous approaches, the abstraction style can also be control by parameter w. Figure 5 shows the different abstraction achieved for an input image by using different values of w and iteration time.

4 User interface

We provide a user-friendly interface for editing images. Users use a "brush" tool to choose the areas or objects in the image to be preserved. The output of the tool is a stylized scene with some realistic objects which are more-or less unaffected by the stylization. The interface is shown in Fig. 6. The thickness of the brush can be controlled by the user. To select large area, users can use thick strokes, while for precise selections, thin strokes can be used. The stylization style (corresponds to w in (10)) can be controlled easily by the user using a "style" slider while the "abstraction level" is controlled by an "iteration" slider.

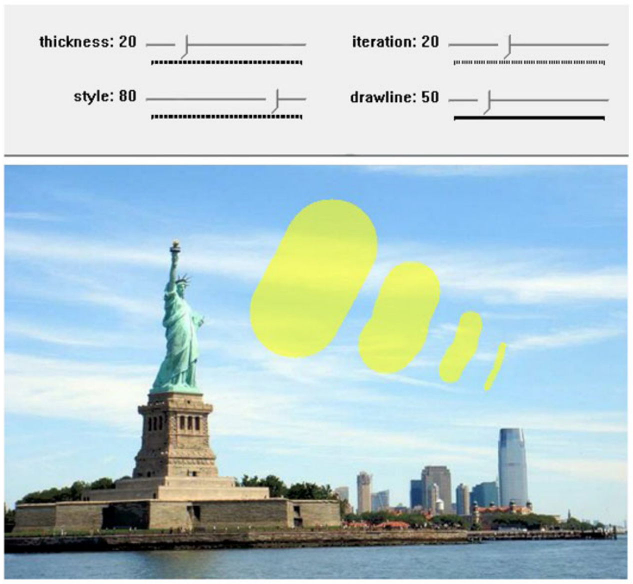

thickness: 20
iteration: 20
style: 80
drawline: 50

Fig. 6 User interface. Sliders control thickness of brush, iteration time, stylization degree (style) and line drawing

Automatic boundary tidying For the selection of areas, most users are willing to paint an object coarsely, but the precise selection of an object is a tedious and difficult task, so we use the Grab Cut algorithm [16] to provide user assistance, that is what we call the "boundary tidying assistant." Grab Cut is a well-known foreground extraction algorithm, which takes a user defined *background* and *unknown* areas as input. The original Grab Cut tool allows the user to select a rectangle to mark the unknown area, which is easy to use but can generate poor results. Our system provides a brush tool, which can be used to mark the unknown area more precisely. It is straightforward for the user to coarsely select an object with a thick stroke. We make a further modification to grabcut, which both accelerates the algorithm and improves the extraction results: the system takes the bounding box of the strokes and dilates it outward by several pixels (as shown in Fig. 7(a)). We then perform Grab Cut in this area of the whole image. This basically ensures that we are using local features instead of global ones, which works better for complex backgrounds. An adjusted user input is show in Fig. 7(b). Most of the time, the grab-cut algorithm works well. However, this approach is not always perfect. Sometimes, it misses some parts of the foreground. Thus, our tool allows user to further modify the initial mask with extra thin strokes. "Eraser" tool is also provided in case the user want to subtract stuffs.

Smooth transition After determination of the foreground, the system applies a morphological dilation operation to the foreground mask. Foreground pixels are now assigned the importance value 1. We then perform a series of dilations

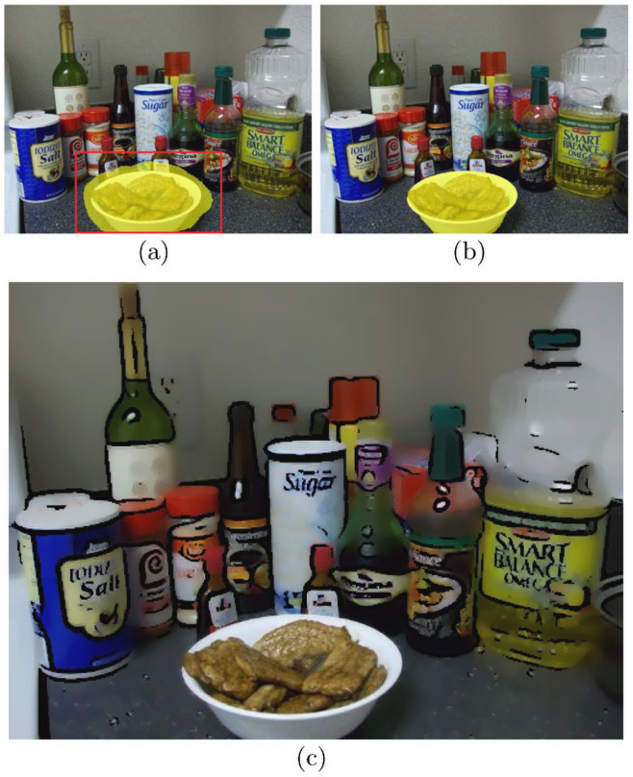


Fig. 7 User input: (**a**) user input with thick strokes, *red box* means the dilated bounding box of the strokes, (**b**) input adjusted by Grab Cut, (**c**) stylized result with parameters: $w = 0.9$, 30 iterations were used

to the foreground regions, assigning a decreasing value (according to a Gaussian function) to the newly reached pixels after each iteration. This helps to reduce the need for the user to select a precise boundary for the foreground. The nonlinear diffusion part of (10) guarantees a smooth embedding of the foreground into the background. Our experimental results show that, most of the time, this method works well.

Line extraction In order to enhance the contrast of the foreground and the background, users can selectively add line strokes to the background using "drawline" slider. Lines are extracted by a flow-based difference of Gaussian edge detector [10], which generates more stylistic lines than Canny edge detection. The importance map can be used efficiently to distinguish foreground and background.

Foreground enhancement In order to further enhance the foreground, the luminance quantization method [21] is applied to the background. Moreover, the intensity of the background is reduced, by decreasing the luminance of Lab color space. This task is also guided by the importance map:

$$L(\mathbf{x}) = L(\mathbf{x}) \cdot \big(\alpha \cdot M(\mathbf{x}) + (1 - \alpha)\big) \tag{12}$$

where $L(\mathbf{x})$ represents the luminance of pixel $\mathbf{x}$ and $M(\mathbf{x})$ means the importance map. We find that $\alpha = 0.1$ is a suitable value.

5 Results

5.1 Experimental results and comparison

We implemented our method using OpenCV and Microsoft Visual Studio 2005 on an Intel Core 2 3.0 GHz processor with 4 GB memory under Windows Vista.

Compared to the bilateral filtering based method [21], our model can generate more stylized results by simplifying colors as well as region boundaries, by setting w to a relatively small value. As shown in Fig. 8 (from [9]), Winnemöller et al. method does not simplify the shape of the fish. In contrast, our method simplifies colors as well as the boundaries.

Compared to Kang and Lee's method [9], which uses a shock filter for edge enhancement after several iterations of mean curvature flow, our method provides edge enhancement as an integral part of the mechanism (see (10)) and does not require a separate processing filter. Therefore, our method is faster than Kang and Lee's. For a 500×500 RGB image, the average computational time for 50 iterations is about 4 seconds without GPU acceleration (Kang and Lee: 30 senconds). Moreover, in the situation of no user interaction, Kang and Lee's method is a little bit too aggressive for dismantling meaningful structures. As shown in Fig. 9 (from [9]), the rectangle structures are not preserved as well as in our method.

Figures 10 and 11 present various stylized results which show the harmonious synthesis of realistic and nonrealistic objects. For all of these examples, we used our model with parameter settings $\sigma_s = 16$ and $\sigma_r = 0.1$. Parameter w and the number of iterations i were set as follows: Fig. 10(a), (b) $w = 0.8$, $i = 10$; (c), (d) $w = 0.8$, $i = 10$; (e), (f) $w = 0.8$, $i = 25$; Fig. 11(a), (b) $w = 0.8$, $i = 20$. For these results, the boat, the horse, the Statue of Liberty, and the man have been selected to be preserved, respectively.

Figure 12 shows that our method can generate the effect of changing a depth of field. If the blue can on the left is chosen, the result is shown in Fig. 12(b); If the white can is chosen, the result is shown in Fig. 12(c). This simulates the effect of changing the focal point of a camera, which can be used to emphasize the selected items in a complicated scene.

The processing time mainly depends on image size and number of iterations. The local nature of the numerical solution of our model means that it is amenable to parallelization for GPU implementation, although we have not yet done so.

Compared to earlier image abstraction methods, our mixed reality scenes show a special case of nonphotorealistic rendering.

5.2 User study

In this section, we are trying to prove the effectiveness of our tool through an evaluation.

We invited ten individuals for our user study. They include six graduate students and four undergraduate students

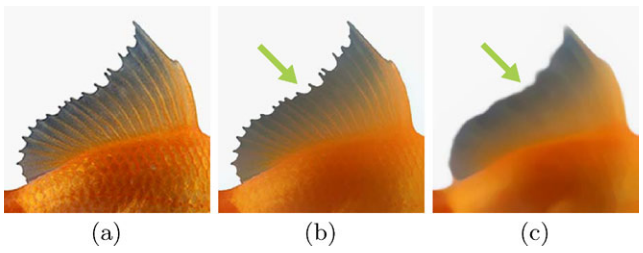


Fig. 8 Comparison of results produced by different methods: (**a**) Original image; (**b**) Winnemöller, Holger and Olsen's method; (**c**) our method with parameters: $w = 0.3$, 40 iterations were used. Our method simplifies colors as well as region boundaries

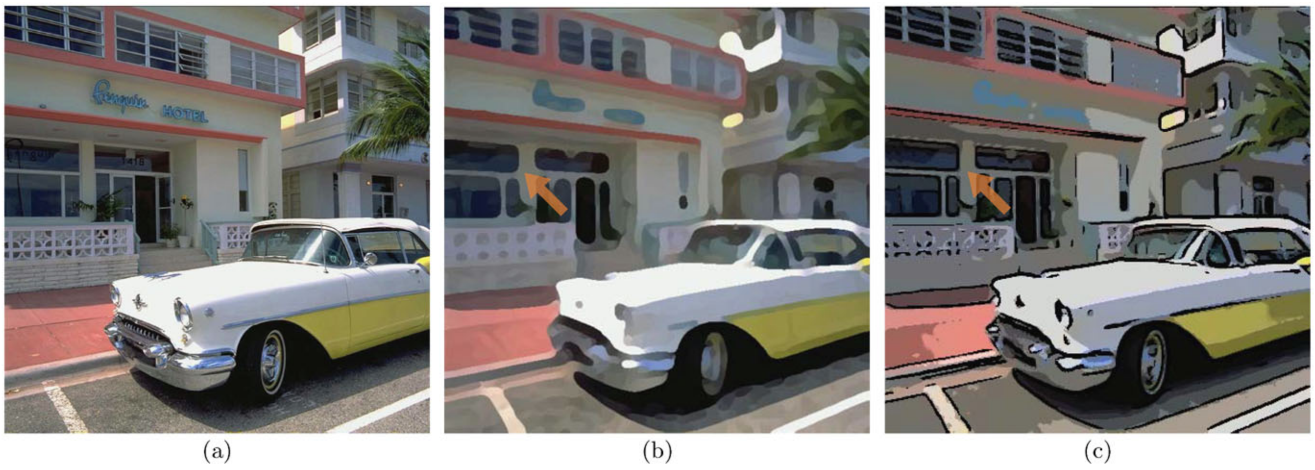
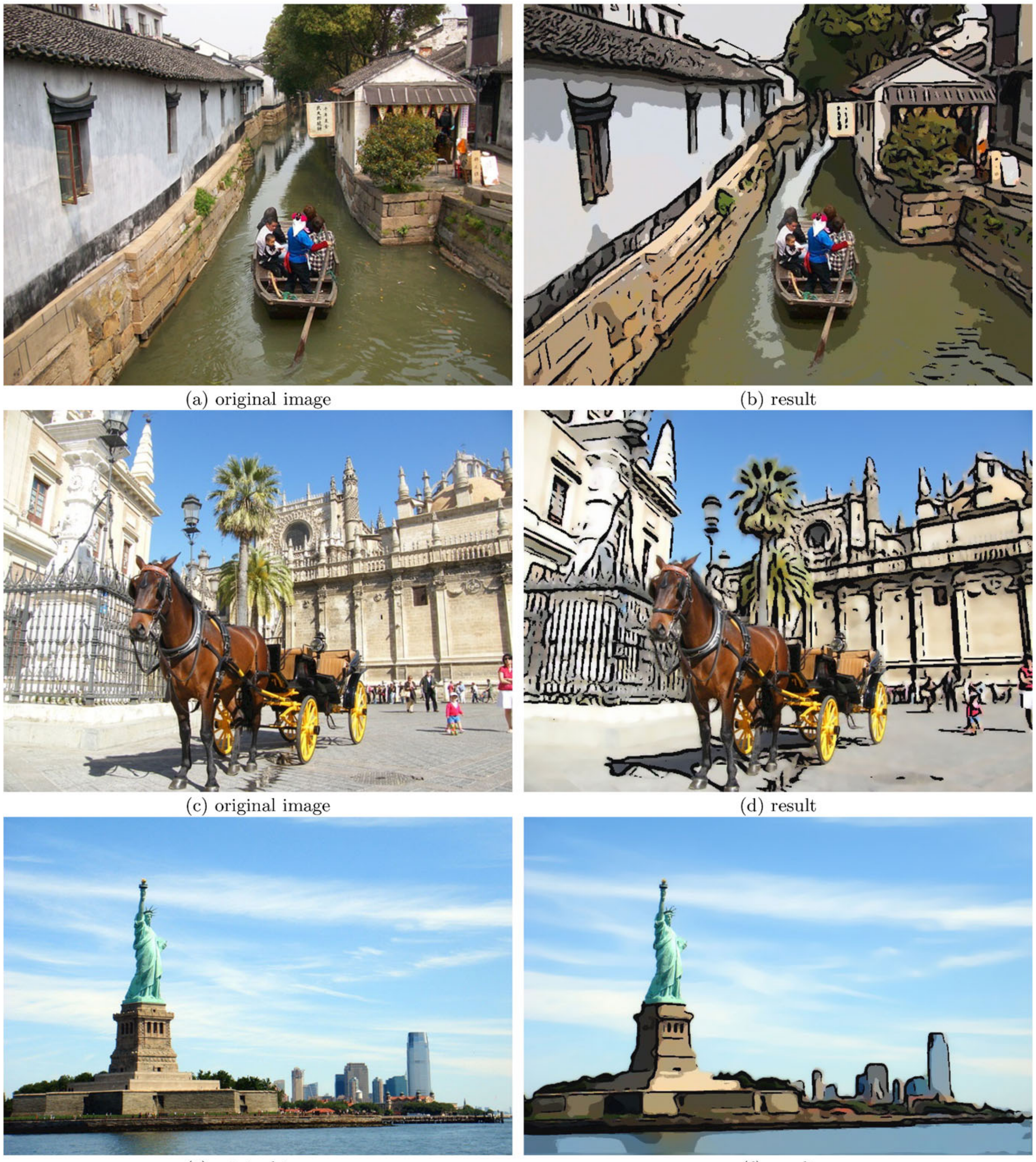


Fig. 9 Comparison of results produced by different methods: (**a**) original image; (**b**) Kang and Lee; (**c**) Our method without user interaction. Our method preserves the rectangle structures better



(a) original image (b) result

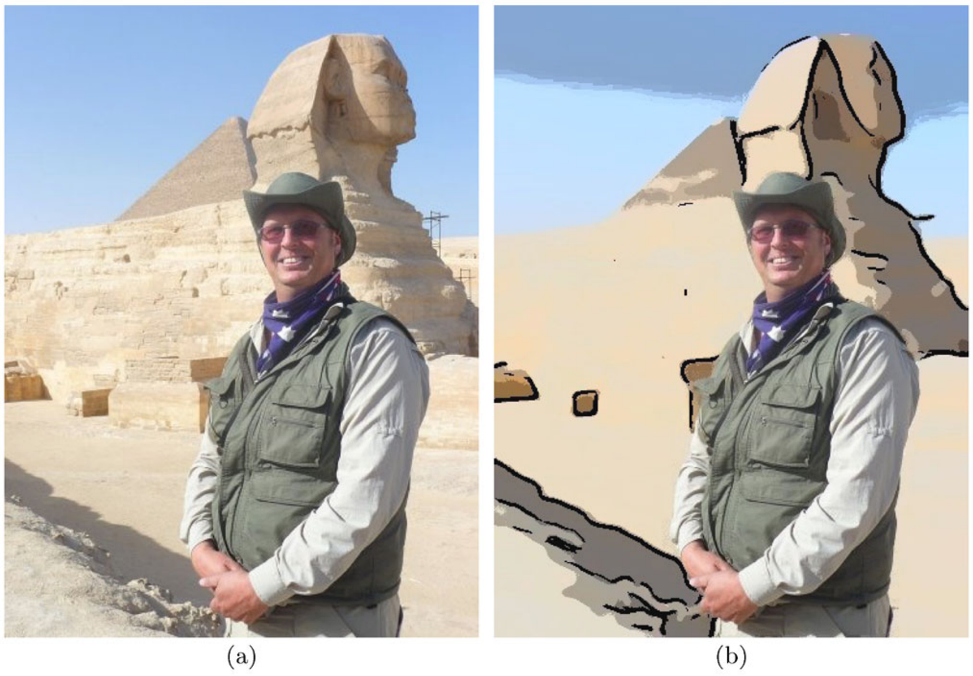
(c) original image (d) result

(e) original image (f) result

Fig. 10 Results

majoring in Computer Science and Technology, who are proficient in "Adobe Photoshop." First of all, we gave them a brief introduction about the usage of our tool including the brush, the selection of stroke thickness, the selection of stylization level, and the "boundary tidying assistant." Then we showed them two demos of editing input images to generate mixed-reality scenes. After that, each of the participants was required to edit five new images to produce desired results

Fig. 11 Results



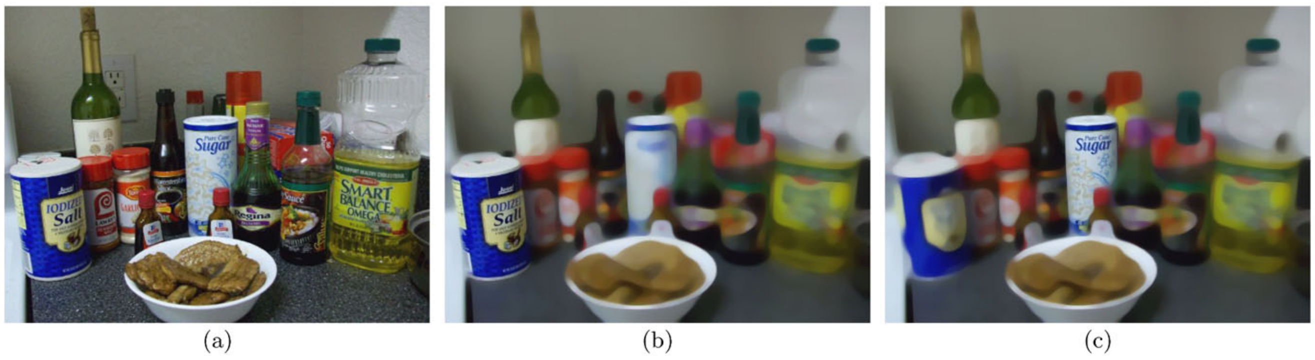


Fig. 12 Changing depth of field effect: (**a**) Original image; (**b**) The *blue can* is emphasized; (**c**) The *white can* is emphasized. All of the results are generated without line drawings

using our tool and Adobe Photoshop[1] separately. These included Figs. 5, 10(a), 10(c), 10(e), and 11. Participants were not able to see each others' work.

The average time by the participants to process each image using two different tools is given in Table 1.

We also evaluated the quality of the results.[2] In the user study, most of the participants use "Poster Edges" filter to mimic our effect. Thus, we compare our method with the Photoshop "Poster Edges" filter. The effect of Photoshop is different from ours. It does not simplify the structures. It enhances unnecessary details in some examples, which is not a good feature for image abstraction. In contrast, our method simplifies the unnecessary details of the background as much as possible. It increases the contrast between background and foreground, which obeys some artistic rules. Users were asked to evaluate the results generated by two

Table 1 Average operation time

Input image	Photoshop	Our tool
Fig. 5(a)	55 s	30 s
Fig. 10(a)	60 s	35 s
Fig. 10(c)	55 s	35 s
Fig. 10(e)	45 s	25 s
Fig. 11(a)	50 s	25 s

[1]Most of them used quick selection tool, feather tool, and "Poster Edges" filter to simulate our results.

[2]Some results generated by Photoshop can be found in the auxiliary files.

tools on a scale from 1 (poor) to 5 (excellent). Here, we list the average scores: Our tool (4.1) and Photoshop (3.6).

The results show that our tool is more effective than Photoshop for the selective image abstraction task.

6 Conclusions

We have introduced a convenient tool for selective image abstraction. By using a brush with a "boundary tidying assistant," and sliders to control a few simple parameters, users can efficiently produce harmonious mixed-reality scenes, in a particular artistic style. The nonlinear diffusion model we use together with the refined user input can guarantee a seamless mixture of different abstraction levels.

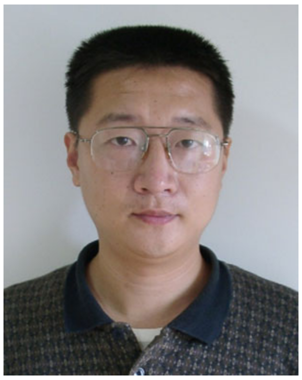
The main limitation of our method is that processing a large image (over 5 megapixels) is computationally expensive. However, our approach is amenable to GPU implementation, which will consider in future. Because the grab-cut algorithm is not always perfect, more robust foreground segmentation algorithm can be considered as a part of our future work. Another future work is to produce selective abstraction results automatically, we may adopt the concept "shrinkability map" [23] to determine the abstraction level of each pixel, since the shrinkability map implies the importance of each pixel. Our method can also be used to generate selective abstraction video based on the importance map.

Acknowledgements This work is supported by the National High-Tech Research and Development Program of China (Project Number 2009AA01Z330) and the National Basic Research Project of China (Project Number 2006CB303106).

References

1. Alvarez, L., Lions, P.L., Morel, J.M.: Image selective smoothing and edge detection by nonlinear diffusion. II. SIAM J. Numer. Anal. **29**(3), 845–866 (1992). doi:10.1137/0729052
2. Boykov, Y.Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & amp; region segmentation of objects in n–d images. In: Intl. Conf. on Computer Vision, pp. 105–112 (2001)
3. Canny, F.J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986). doi:10.1145/11274.11275
4. Catté, F., Lions, P.L., Morel, J.M., Coll, T.: Image selective smoothing and edge detection by nonlinear diffusion. SIAM J. Numer. Anal. **29**(1), 182–193 (1992). doi:10.1137/0729012
5. Collomosse, J.P., Rowntree, D., Hall, P.M.: Stroke surfaces: temporally coherent non-photorealistic animations from video. IEEE Trans. Vis. Comput. Graph. **11**(5), 540–549 (2005). doi:10.1109/TVCG.2005.85
6. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
7. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. In: SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 769–776. ACM, New York (2002). doi:10.1145/566570.566650
8. Fischer, J., Bartz, D., Straßer, W.: Stylized augmented reality for improved immersion. In: Proceedings of IEEE Virtual Reality (VR 2005), pp. 195–202 (2005)
9. Kang, H., Lee, S.: Shape-simplifying image abstraction. Comput. Graph. Forum **27**(7), 1773–1780 (2008)
10. Kang, H., Lee, S., Chui, C.K.: Coherent line drawing. In: ACM Symposium on Non-Photorealistic Animation and Rendering (NPAR), pp. 43–50 (2007)
11. Kang, H., Lee, S., Chui, C.K.: Flow-based image abstraction. IEEE Trans. Vis. Comput. Graph. **15**(1), 62–76 (2009). doi:10.1109/TVCG.2008.81
12. Koenderink, J.J.: The structure of images. Biol. Cybern. **50**(5), 363–370 (1984). doi:10.1007/BF00336961
13. Orzan, A., Bousseau, A., Barla, P., Thollot, J.: Structure-preserving manipulation of photographs. In: International Symposium on Non-Photorealistic Animation and Rendering (NPAR), pp. 103–110 (2007). http://artis.imag.fr/Publications/2007/OBBT07
14. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. Int. J. Comput. Vis. **81**(1), 24–52 (2009). doi:10.1007/s11263-007-0110-8
15. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**, 629–639 (1990)
16. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**(3), 309–314 (2004). doi:10.1145/1015706.1015720
17. Santella, A., DeCarlo, D.: Visual Interest and NPR: an Evaluation and Manifesto, pp. 71–78. ACM Press, New York (2004). doi:10.1145/987657.987669
18. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: IEEE International Conference on Computer Vision, p. 839 (1998). doi:10.1109/ICCV.1998.710815
19. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.F.: Video tooning. ACM Trans. Graph. **23**(3), 574–583 (2004). doi:10.1145/1015706.1015763
20. Wen, F., Luan, Q., Liang, L., Xu, Y.Q., Shum, H.Y.: Color sketch generation. In: NPAR '06: Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering, pp. 47–54. ACM, New York (2006). doi:10.1145/1124728.1124737
21. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, pp. 1221–1226. ACM, New York (2006). doi:10.1145/1179352.1142018
22. Zhang, S.H., Chen, T., Zhang, Y.F., Hu, S.M., Martin, R.R.: Vectorizing cartoon animations. IEEE Trans. Vis. Comput. Graph. **15**(4), 618–629 (2009). doi:10.1109/TVCG.2009.9
23. Zhang, Y.F., Hu, S.M., Martin, R.R.: Shrinkability maps for content-aware video resizing. Comput. Graph. Forum **27**(7), 1797–1804 (2008)

Lin Cong is a Ph.D. candidate in the Department of Computer Science, Zhejiang University, China. He received his B.E. from Zhejiang University, China, in 2007. His research interests include image and video processing and computer graphics.



Ruofeng Tong is a professor in the Department of Computer Science, Zhejiang University, China. He received his B.E. from Fudan University, China, in 1991, and a Ph.D. from Zhejiang University, China in 1996. He was a visiting researcher in Faculty of Engineering, Hiroshima University, Japan, in 2000. His research interests include image and video processing, computer graphics, and computer animation.

Jinxiang Dong is a professor in Department of Computer Science, Zhejiang University, China. His research interests include image and video processing, computer graphics, and computer animation.