Message in a Bottle: Stylized Rendering of Sand Movies

Laurindo S. Britto Neto, Bruno M. Carvalho *
Departamento de Informática e Matemática Aplicada - UFRN
Campus Universitário, S/N, Lagoa Nova, Natal, RN, 59.072-970 - Brazil
laurindobneto@gmail.com, bruno_m_carvalho@yahoo.com

Abstract

Stylized rendering, usually referred as non-photorealistic rendering, aims to reproduce artistic techniques in renderings, trying to express feelings and moods on the rendered scenes, as opposed to realistic rendering techniques, that aim to produce realistic renderings of artificial scenes. In this paper, we present a stylized rendering technique that aims to create synthetic sand-filled bottles, simulating a typical art craft from the Northeastern region of Brazil, that usually depicts landscape images. A method for generating 2D procedural sand textures is presented, and two new techniques that mimic effects created by the artists using their tools are introduced. The technique is also used to generate stylized videos, something close to impossible in real life with this technique. The temporal coherence within these stylized videos can be enforced on individual objects with the aid of a video segmentation algorithm. The techniques were used on synthetic and real videos, and several snapshots from these videos are shown here.

1 Introduction

Non-Photorealisitc Rendering (NPR) is a class of techniques defined by what they do not aim, the realistic rendering of artificial scenes. Recently, several authors have adopted the term stylized rendering, defining this class of techniques in a non-negative manner. The main idea behind stylized rendering is the simulation, in computer generated renderings, artistic techniques. Characteristic effects of such artistic techniques can have a great impact in expressing feelings and emphasizing some information on the rendered scenes. Stylized rendering can also be defined by their input and output, as the processing of images and videos into artwork that can have the visual appeal of traditional pieces of art, expressing the visual and emotional characteristics of artistic styles (e.g., through the strength, width, and shape of brush strokes). There are several methods reported in the literature that simulate some artistic style, such as watercolor [2], mosaics [1], impressionist style painting [11, 13] and cartoon [16]. The use of stylized renderings for processing videos was first proposed in [11].

Animation techniques can show some information in a way that cannot be done by simply shooting a real scene with a video camera. The problem with such techniques is that they are usually very labor intensive. Moreover, to be properly used, these techniques require a significant amount of artistic skill. On the other hand, stylized rendering techniques can be used to generate highly abstracted animations with little user intervention. This allows non-artist users to create their own animations with little effort, when compared to traditional animations, where they had to be created from scratch. The processing of videos with stylized rendering techniques is sometimes referred to as video stylization, and can be applied to the whole video or used to mix real movies with stylized objects, as a means to emphasize them.

It is very important that the stylized video produced, from an artificial 3D scene or a real video, exhibit temporal coherence of the elements (brush strokes, curves, circles, etc.) used in the renderings. In stylized renderings of 3D modeled scenes, not moving the elements with the surfaces being drawn can make the animation appear as it is seen through a textured glass, the so called *shower door effect* [13]. The lack of temporal coherence in styled renderings of videos can produce distracting flickering, also called *swimming* [6], and it is produced when moving objects are rendered with elements that do not follow the object correctly or when static areas are rendered differently in adjacent frames, usually due to noise or changes in shadowing or illumination.

There are several approaches that try to impose temporal coherence in video stylization. In [11], the information of an optical flow [7] method was used to track movement in the video, and move, add or remove brush strokes from frame to frame. An approach for coherent rendering of

*This work was supported by CNPq Grant PDPG-TI 506555/04-6.

static areas in successive frames was proposed in [10], by keeping the brush strokes of areas not detected as changed, based on some threshold. Intra-object temporal coherence is obtained by warping the brush stroke's control points using the output of an optical flow method. In [16], the authors propose to maintain temporal coherence on a cartoon animation by the use of a segmentation algorithm to segment objects from a video shot in a end-to-end manner, followed by the user selection of constraint points on keyframes of the video shot. These points are then used for interpolating the region boundaries between frames. The video is also treated as a 3D volume in the method proposed by [6], and temporally convex segmented objects are associated in semantic regions, which have time-coherent basis vectors associated to them to enforce temporal coherence.

In the stylized rendering technique presented here, the lack of temporal coherence in some animation is not as noticeable as when drawing in other artistic styles, such as impressionism. This is due to the nature of the material used, since simulating sand implies in some randomness being present in the images. Thus, we can tolerate changes in colors or positions of sand grains, since they usually do not affect the overall appearance of the animation. However, this changes can be distracting if they are too big and too common. To avoid that, we propose a technique for restricting the amount of change in color of the grain sands, based on several thresholds that are defined by the user.

2 Sand Bottles

The creation of landscape pictures in sand filled bottles is a type of art developed in the last century by coastal craftsman families in the northeast of Brazil, mainly in the states of Rio Grande do Norte and Ceará. In the coast of these states, there are sands with several colors and shades, that can be found in beaches and dunes. An example of a typical sand filled bottle produced using this technique can be seen in Figure 1.

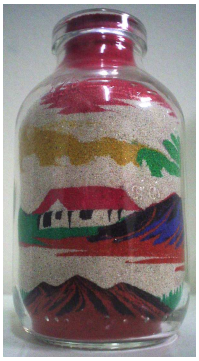
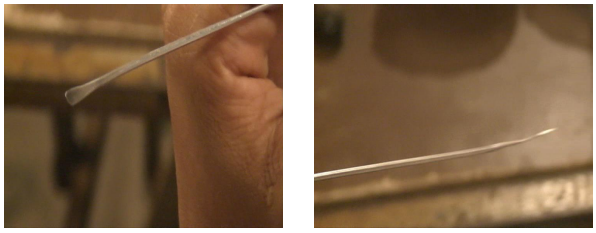


Figure 1. Sand filled bottle.

To create an image inside a bottle, such as the one shown on Figure 1, the artist pours sand of different colors inside an empty, transparent bottle, and, using an iron tool with one thin sharp end and another flat end (Figure 2), moves the sand to create shapes on the inner surface of the bottle. The drawing of an object is done first by roughly delineating the borders of the object, i.e., removing excess sand, using the flat end of the tool. Then, the shape of the object is refined by the artist using the thin end of the tool. After that, some sand from surrounding objects can be moved inside the object to create some effect. For example, when drawing a house, the artist first composes the visible walls with the sand, pours sand with a different color for the doors and windows on top of the wall's sand, and them makes way in the wall's sand with the flat end of the tool for the door and window sand to be placed in front of the wall's sand. Finally, the artist pours some sand for the roof and arranges it to finish the house. A variation of this type of craft is the creation of pictures between two flat pieces of glass, producing a "painting" that can be laid on a flat surface.



(a) (b)

Figure 2. The flat (a) and the sharp (b) ends of the tool used by the artists to compose the pictures inside the bottles.

Because of environmental reasons, the artists are not allowed to use natural sand anymore. The coastal areas where the colored sands were extracted are now tourism attractions and protected by law. Thus, artists now have to use artificial sand, that not only proved to be easier to manipulate, but also can be found in bright shades of blue, green, purple and other colors that were not available in nature.

3 Sand Style Rendering

The simulation of such images using stylized rendering also opens the possibility of producing something that is close to impossible in real life, the creation of animations and videos with the artistic technique described above. To do this, an artist would have to create a series of bottles or "paintings", as described above, with the same background, and some moving objects. It is easy to see that doing this would be extremely hard, and that the animation would have

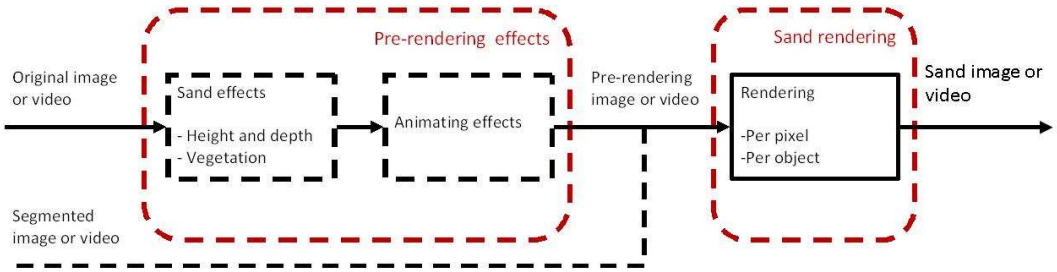


Figure 3. Rendering pipeline of our technique.

to be very limited to allow the creation of such sequence.

We now proceed to describe our technique to generate sand style rendered images and videos, from the generation of procedural sand textures and the pixel color selection, to two effects developed to mimic the artists techniques. The whole process of generating sand style rendered images is performed with the aid of graphical interfaces, in the order of the pipeline shown in Figure 3, which takes as input an image or a video. The dotted boxes in Figure 3 represent optional steps in the process (the animating effects step can only appear if some sand effect is present) while the dotted input line represents an optional input to the pipeline.

3.1 Procedural Sand Textures

Procedural texture generation is a very popular technique for simulating natural materials. For example, Perlin noise [14] has been used to procedurally generate 3D textures simulating marble, rock, and wood [8]. In our case, noise functions are used to generated 2D textures, since we are dealing with an image that will be placed on the inside of a bottle model and all the interaction used to simulate the artist's strokes will take place on the "surface" of the sand object.

Now we describe the method used to generate the sand textures, first introduced in [4]. In order to generate proper textures for representing the artificial sand used in making these bottles, we photographed samples of sand of several colors used by the artists. The images were then analyzed in the HSV space in order to determine the distribution that best describe them. The histograms of the photographed sand samples were found to be approximated with a reasonable accuracy by Gaussian distributions, by using

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \qquad (1)$$

where the mean μ and the standard deviation σ for the primary sand colors were determined from the sand samples, as well as defined by us to create new primary sand colors. We then generate sand primary color textures at least every 30° in the hue space ($[0^\circ, 360^\circ]$) with the same size as the input image, and store them in texture memory. For reasons that will become clear in a moment, we also generate white, black and gray sand textures.

3.2 Rendering Sand

When rendering an image, we compute, for every pixel location, the correspondent primary sand texture to which the pixel belongs, and draw the texture value stored on that location on the frame buffer. However, there are some special cases that have to be treated differently. For example, if the value channel (V of the HSV model) of the input image is close to 0 (smaller than a user defined threshold), we use the black sand texture. Another threshold has to be set for the saturation (S) channel, to treat the cases where the input color can be approximated by a gray value.

If a color is located between two primary sand colors in the hue (H) space, we calculate a mixture of the two closest primaries in this space that matches its hue, and draw a pseudo-random number that will determine the sand texture from which the pixel value will be retrieved. For example, if the color of the pixel under examination is formed by 30% green and 70% cyan, a pseudo-random number smaller than 0.3 (in the range $[0, 1]$) will assign to the rendered pixel a value fetched from the green texture, while a value bigger than 0.3 will assign to the rendered pixel a value fetched from the cyan texture. However, the mixing of two textures can be controlled by an upper and a lower thresholds, as explained in Section 3.2.1.

Before rendering the whole video, usually we select a frame from it and use it to set up the parameters and colors used with the aid of a graphical interface. the user can also select the rendering method, per pixel or per object rendering. All the used pseudo-random values can be generated upfront and stored in texture memory, so the whole process can be done by pixel shaders.

3.2.1 Per Pixel Rendering

In the method of per pixel rendering, for every pixel in the input image, we access the correspondent sand texture, based on its input value, and write the texture value in the correspondent position in the frame buffer. However, first we have to treat some special cases. In the HSV model, when the value of the channel V is 0, the values of the channels H and S are undefined, and the correspondent pixel is black. Similarly, when the value of the channel S is 0, H is undefined, and the color of the pixel is defined as gray, with the intensity of the gray being defined by the value of the channel V. Thus, based on specified thresholds that can be changed by the user, we test if the value of the channel V of a pixel is below a lower threshold, and if it is, we render it using the black texture. Similarly, if the value of the channel S is below a specified threshold, we use the gray texture, probably mixed with the white or black textures to render the point.

The mixing between two textures is performed using pseudo-random numbers, as described above, and lower and upper limits for mixtures can be set by the user. For example, if we are mixing the textures A and B with the lower and upper mixture values of 30% and 70%, and the value of the H channel of the pixel being processed can be represented by a mixture of 40% of A and 60% of B, then a pseudo-random number is drawn and used to select the texture to be accessed. If the amount of the primary color B needed to represent the pixel color is below 30% or above 70%, then the pixel value would be solely extracted from the textures A and B, respectively. Setting the lower and upper limits to the same value would prevent sand mixtures.

A problem of temporal incoherence arises when we mixture two or three textures. Due to the random nature of the choice described above, pixels may have their values change significantly, since they may come from different textures, and thus, generate sequences with unwanted flickering. We solve this problem by dynamically storing mixed textures in texture memory, and accessing them if a pixel value is unchanged. Another potential problem is that aliasing can occur if the camera is positioned too close to the bottle, rendering little squared sand grains. This problem can be solved by not allowing the camera to come too close to the bottle or by defining a larger maximum size for the rendered images and using mipmaps according to the distance camera to bottle.

3.2.2 Per Object Rendering

In the method of per object rendering, an object is rendered using one or more textures according to the averages of its HSV channels. As in the previous method, lower and upper limits can be set to control the mixture of the textures. In this case, setting the lower and upper limits to the same value would produce an image similar to a cartoon rendering (with the shades coming from the different segmented objects), but with the variations of the sand textures.

To perform the segmentation of objects in a video sequence, we use the fast fuzzy segmentation algorithm described in [5], based on the works presented in [9, 3]. This method computes, for each pixel in the video sequence, a grade of membership (between 0 and 1) to all objects, with the object with highest grade of membership to this pixel claiming it. The algorithm is a region growing segmentation method, and the number of objects is selected by the user, as well as seed pixels, that identify with certainty pixels which belong to the objects of interest. The regions grow based on the values of fuzzy affinity functions, that are calculated using statistics collected on the areas surrounding the selected seed pixels for the objects. The video is treated as a 3D volume, so occlusions and objects appearing and disappearing are handled without problems. In order to achieve better segmentation of moving objects, we adapted the segmentation method described in [5] to incorporate motion information from a dense optical flow map in the fuzzy affinity functions. The dense optical flow map is calculated using a multi-resolution version of the algorithm presented in [15], developed for performing the experiments of [12].

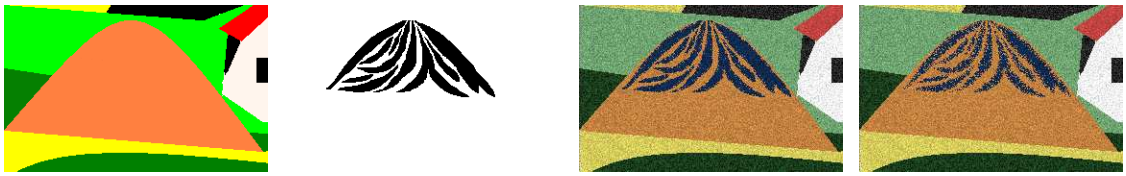
4 Simulating Sand Stroke Effects

After pouring sand inside the bottle and delineating objects, the artist can create some effects in the picture through strokes of the tool. It is important to simulate them in the stylized video if the intent of the user is to produce some image or animation that resembles the type of pictures seen on the original sand filled bottles. We now proceed to describe how these effects are made by the artist and how we simulate them.

4.1 Height and Depth Effect

The first effect created by the artist was named by us as the *height and depth* effect, since it can express a rough sensation of varying height and/or depth in the composed picture. An example of such effect can be seen on Figure 4. To create such effect, the artist drags sand of a different color from the center of the bottle to the inner surface of the bottle, using the flat end of the tool (see Figure 2(a)), and changing the angle it makes with the vertical axis, to drag more or less sand. We defined 0° as when the tool drags more sand, i.e., when the flat end of the tool faces the inside of the glass bottle with maximum width, and 90° when the flat end of the tool faces the inside of the glass bottle with minimum width.

To simulate this effect in the images produced by the stylized rendering method proposed here, we included con-



(a) (b) (c) (d)

Figure 5. Example of the application of the simulated height and depth effect, where we can see the original Image (a), mask (b), simulated effect with $\alpha = 20°$ (c) and $\alpha = 40°$ (d).

Figure 4. Example of the height and depth effect. The curves drawn by the artist using a different sand give a rough sensation of varying height and depth in the object.

trols in the graphical tool developed to change the approach angle of the flat end of the tool as it hits the inner glass wall. (This angle refers to the rotation around the vertical axis.) Besides choosing the tool approach angle, the user has also to select the color of the sand that will be "dragged" in front of the original sand, and indicate where he/she wants the effect to be applied. Thus, basic drawing tools were also added to the graphical tool.

After the user draws the areas (mask) where the effect will be applied, selects the sand texture that will be used in the effect, and the angle of approach of the tool, a distance map of the mask is computed and its highest value stored. The distance map is then used to determine if the selected sand will replace the sand currently in place or not, according to a Gaussian distribution with mean 0 and standard deviation given by the equation

$$\sigma = \begin{cases} \sigma_{min}, & \text{if } \; 1 - \frac{\alpha}{\alpha_{max}} < t, \\ \sigma_{max}\left(1 - \frac{\alpha}{\alpha_{max}}\right), & \text{otherwise,} \end{cases} \qquad (2)$$

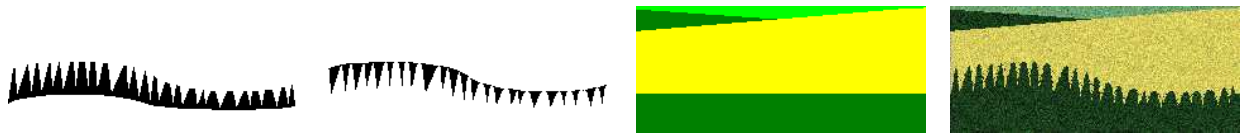
where α and α_{max} are the angle the tool is making with the inner surface of the bottle and the maximum angle allowed for the tool ($90°$), respectively, t is a threshold set to limit the smallest standard deviation value, and σ_{max} and σ_{min} are the maximum and minimum standard deviation values allowed, respectively. The value σ_{max} for a mask was set as the highest value in the distance map of the mask. Then, based on the distance map value of a pixel, a probability from the Gaussian distribution is fetched and a pseudo-random number drawn to determine if the pixel will be replaced by a pixel from the sand texture being applied in the effect or will remain with the value of the sand texture assigned before applying this effect. An example of applying this effect can be seen in Figure 5.

4.2 Vegetation Effect

The second effect created by the artist and simulated here was named the *vegetation* effect, consisting of high frequency features aimed to represent vegetation an done with the sharp end of the tool used by the artist. An example of this effect can be seen in Figure 6.

Figure 6. Example of the vegetation effect (green object).

To simulate this effect, the user draws, with the aid of the graphical interface, two lines to delineate the varying height of the simulated vegetation, and specifies the minimum (min) and maximum (max) widths of the bottom of the vegetation to be rendered. The value min is also used as the maximum width of the top of the vegetation. The algorithm then scans the lines looking for the first column in which both appear, and starts to draw quadrilaterals whose base lengths are determined by random numbers in the range $[min, max]$ and whose top lengths are determined by random numbers in the range $[1, min]$, according to uniform distributions, until the last column in common is reached. The heights are determined by the drawn upper



(a) (b) (c) (d)

Figure 7. Original (a) and complimentary masks (b) applied to the original image (c) to generate the vegetation effect seen in (d).

(a) (b) (c)

Figure 8. Three frames of the animation of an effect.

line, and the top portions of the quadrilaterals are clipped to the drawn upper line. The generated mask (Figure 7(a)) is inverted to generate is complimentary mask (Figure 7(b)).

After the masks are generated, we compute a modified distance map for each mask are calculated, in a similar manner to the distance masks used in the height and depth effect, but with one difference. For each triangle, after the standard distance map is calculated, we search in the central column of the triangle for the point where the distance map has maximum value. Connecting this point to the two triangle base vertexes we create another triangle. The values of the distance map for this triangle are then replaced by the maximum value of the distance map. This is done to limit the effect of sand mixing to the vertical borders of the triangles, leaving the area on the base of the triangles intact. The rendering of this effect is then performed for both masks, allowing the two sand textures to switch pixels, with highest probabilities when they are closer to the triangle borders than when they are farther away.(The dark green on the sand rendering of Figure 7(d) appears darker than the green in the correspondent areas of the original image shown on Figure 7(c) because the color of the sand primary used in Figure 7(d) is the closest one in hue space to the color of the original image.)

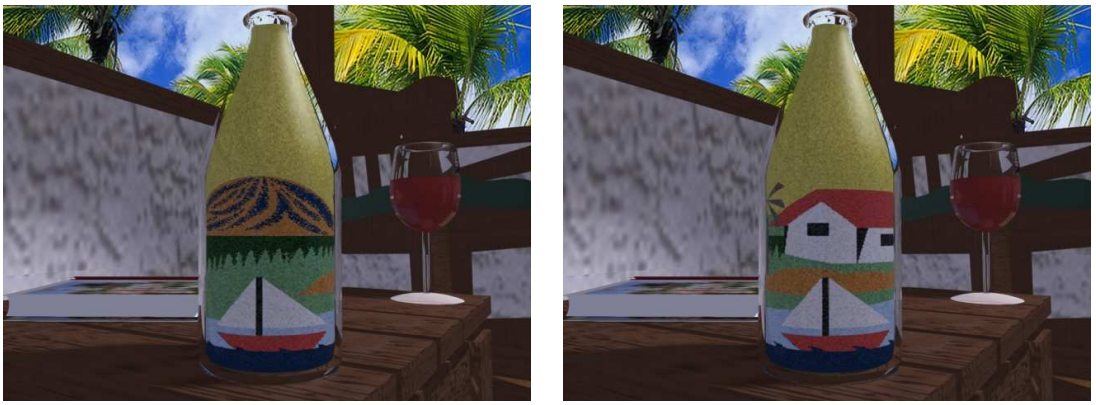
4.3 Animating the Effects

In order to allow such effects to be used in parts of videos in a non-static manner, we implemented a simple mechanism that allows the user to include animations of these effects. The user defines the initial and final frames of the video where the sequence will appear, selects the function (linear or sigmoid) that will turn the effect on and off, and a slope factor (that will tell us how fast the effect will appear/disappear). Based on the function and slope factor chosen, the values stored in the distance maps (modified distance maps, in the case of vegetation effects) will be multiplied by a factor (between 0 and 1) before the rendering of the effect is performed. Figure 8 shows three frames of an animation created using this method.

5 Experiments

Now we describe some experiments we performed to show how our techniques work. In Figure 9, we can see four frames of two animations on sand filled bottles, where the images composing these animations resemble the type of scenes usually depicted by the artists. We used a bottle model similar to the ones used by the artists, but any container model could be used. However, one has to be careful in choosing the model, since details in the models can lead to significant distortions in the rendered images due to refraction calculated when rendering the glass. If we want to generate a sand-style rendered animation from a real video to produce an animation with images similar to the images produced by the artists, we have to blur the input image/video frames, creating an abstracted version of it, thus, eliminating small details that would not have been captured by the artist, in a similar way to the approach proposed in [2] for producing watercolor images from photographs.

If we want to limit the colors used to render an object, we use the per object method. An example of its usage can be seen in Figure 10. Figure 10(b) shows the segmentation map for the original frame shown in 10(a), where the hue



(a) (b)

(c) (d)

Figure 9. Four frames of two different animations showing the result of applying the bottle sand rendering techniques.

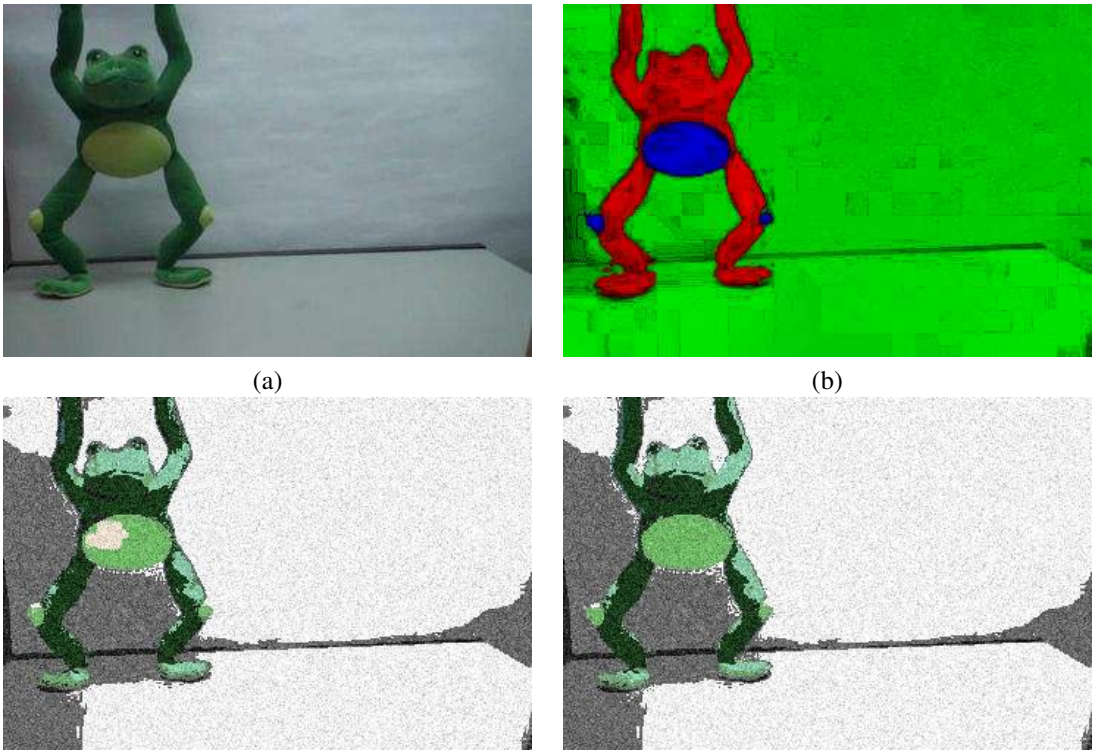
indicates to which object a pixel belongs, and the intensity (mapped from the grade of membership, in the range [0,1]) reflects the confidence that the segmentation algorithm has in this assignment. Figure 10(c) shows a per pixel rendering of the frame, while Figure 10(d) was generated using the map shown in 10(b) to restrict the textures used to render the pixels belonging to the blue object (belly and knees) to the green sand texture. This example also shows how the per pixel rendering maintains some small details inside the frog's body, such as the eyes, nose and part of the mouth. In order to do that in the per object rendering, extra objects would have to be added in the segmentation step. The per object rendering, on the other hand, can be used to maintain the appearance of whole structures, and enforce a certain level of temporal coherence.

6 Conclusion

In this paper we presented a stylized rendering technique that simulates a typical art craft from the Northeastern region of Brazil, that uses colored sand to compose landscape images on the inner surface of glass bottles. A method for generating 2D procedural sand textures, initially presented in [4] is described, and two new techniques that mimic effects created by the artists inside these sand filled bottles using their tools are introduced.

The techniques described here are also used to generate stylized videos, something close to impossible in real life with this technique. We employed a fast fuzzy segmentation algorithm to segment videos as 3D volumes, thus, allowing the segmentation algorithm to deal with occlusion and non-temporally convex objects. The temporal coherence within these stylized videos is enforced on individual objects with the segmentation result, by constraining the sand textures used in some objects.

The techniques described here were used on synthetic and real videos, and several snapshots from these videos are shown here, as flat images, as they would appear in a sandbox, or inside a sand bottle, as they would appear if made by an artist skilled in this craft. Future work will include gen-



(a) (b)

(c) (d)

Figure 10. A frame of the Frog sequence (a), its segmentation map (b), a per pixel rendering (c), and the frame with the blue object of (b) rendered using only the green sand texture (d).

erating $2\frac{1}{2}$D animations, where object movements could be represented as vertical displacements of sand grains on the sandbox plane.

References

[1] G. D. Blasi and G. Gallo. Artificial mosaics. *The Vis. Comp.*, 21:373–383, 2005.

[2] A. Bousseau, M. Kaplan, J. Thollot, and F. Sillion. Interactive color rendering with temporal coherence and abstraction. In *Proc. of NPAR'06*, volume 1, pages 141–149, 2006.

[3] B. Carvaho, G. Herman, and T. Kong. Simultaneous fuzzy segmentation of multiple objects. *Discr. Appl. Math.*, 151:55–77, 2005.

[4] B. Carvalho, L. B. Neto, and L. Oliveira. Bottled sand movies. In *Proc. of CGIV'06*, pages 402–407, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[5] B. Carvalho, L. Oliveira, and G. Silva. Fuzzy segmentation of color video shots. In *Proc. of DGCI'06*, volume 4245, pages 402–407, London, 2006. Springer-Verlag.

[6] J. Collomosse, D. Rowntree, and P. Hall. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Trans. Vis. and Comp. Graph.*, 11:540–549, 2005.

[7] E. Davies. *Machine Vision: Theory, Algorithms, Practicalities, 2nd Ed.* Academic Press, London, 1996.

[8] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texture and Modeling: A Procedural Approach. 2nd Ed.* Morgan Kaufmann, San Francisco, 2003.

[9] G. Herman and B. Carvalho. Multiseeded segmentation using fuzzy connectedness. *IEEE Trans. Patt. Anal. Mach. Intell.*, 23:460–474, 2001.

[10] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *Proc. of NPAR00*, pages 7–12, 2000.

[11] P. Litwinowicz. Processing images and video for an impressionist effect. In *Proc. of ACM SIGGRAPH'97*, pages 407–414, 1997.

[12] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *Comp. Vis. and Image Underst.*, 84:126–143, 2001.

[13] B. Meier. Painterly rendering for animation. In *Proc. of ACM SIGGRAPH'96*, pages 477–484, 1996.

[14] K. Perlin. Improving noise. *ACM Trans. on Graph.*, 21:681–682, 2002.

[15] M. Proesmans, L. Gool, E. Pauwels, and A. Oosterlinck. Determination of optical flow and its discontinuities using nonlinear diffusion. In *Proc. of the 3rd ECCV*, volume 2, pages 295–304, 1994.

[16] J. Wang, Y. Xu, H.-Y. Shum, and M. Cohen. Video tooning. *ACM Trans. on Graph.*, 23:574–583, 2004.