Seamless NPR Video Painting

Levente Kovács

Distributed Events Analysis Research Group
Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13-17,
H-1111, Budapest, Hungary

levente.kovacs@sztaki.hu - http://web.eee.sztaki.hu

ABSTRACT

The paper presents a method for automatic transformation of ordinary videos into non-photorealistic (NPR) graphical representations, for achieving painterly effects. The primary goal is the non-interactive transformation of videos, with the intended use as a service in community video databases (as an extension to content based retrieval). There are no restrictions on the types of videos. The main steps involve scene separation, color segmentation, foreground and focus area extraction, and vector graphical transformation of selected frames.

Keywords

Non-photorealistic rendering, image/video painting, painterly rendering, visualization.

1. INTRODUCTION

Methods for creating painterly, cartoonish effects on ordinary videos are already present in the NPR literature, although their number is fairly limited. Most of these approaches are manual or at most semi-automatic in nature. A number of them is based on stroke simulation, with different optimization techniques to reduce the computational time. Also, the movie and game development industries know and use cartoon renderings, which can produce good results, although they need manual interaction.

Our goal is to create an automatic NPR video rendering method which is light in computational complexity, yet provides nice results. Since we are also working on creating content based video retrieval applications, we intend to provide additional services beside retrieval, one of which is automatic painterly effect rendering on stored videos.

Thus one of the primary requirements is that we create an automatic video renderer. Another requirement is to create the painterly effects on videos by such feature descriptors that are also used when processing images and videos for database retrieval, which are already available in the database, and have a light computational complexity.

One of the first video NPR methods was that of [Lit97a], where strokes with a given center, length, radius and orientation are generated, adding random perturbations and variation; [Hay04a] is a painting method based on generating meshes of brush strokes with adapting spatio-temporal properties. Like in [Kov02a] optic flow data is used to paint frames, in this case by propagating the stroke objects along motion trajectories. [Col05a] is also a semi-automatic NPR animation method based on stroke simulation and propagation, creating various styles, and being highly versatile. [Kim05a] presents a semi-automatic video transformation method which produces cartoonish motion cues on the images, based on segmentation and tracking. [Wan04a] provides a method based on manual object selection and temporal Mean Shift tracking.

The main steps of the approach presented here are the following. We detect the shot changes in the video; then, the renderer creates a two-layer transformation of the shot frames, a background layer which has lower complexity, and a foreground layer, with higher level of detail. The foreground is extracted by a Stauffer-Grimson based foreground detector combined with an automatic focus area extracting method. The color segmentation necessary for the two layers is obtained by using a Mean Shift color segmenting step. Between the frames, only those areas will be updated where the focus and foreground change detector signals a change in content.

Our approach differs from other NPR video creating methods in that it is not based on stroke simulation, is automatic, does not use computationally highly intensive optimizations, generates the least possible layers, and only uses features which are already available in the video database. Thus we will be able to provide an automatic effect generator service as an extension to already available content based retrieval modules in a video retrieval application.

2. NON-PHOTOREALISTIC EFFECTS ON VIDEOS

As we mentioned above, the goal is to build a non-photorealistic video renderer which uses features that are generally (and specifically, in our case) available in a video database. Such features include shot/scene, color, texture, edge, invariant feature, motion descriptors. These features are not randomly selected, but chosen based on how their combination works better for retrieving specific classes of videos [Szl08a]. We tried to reduce the amount of necessary steps and used features to a minimum, to be able to create an efficient video NPR method. The basic elements one needs for a method not based on stroke simulation is to extract moving areas (foreground), which in itself will help in identifying the background layer. Then, we need to apply color reduction which produces a painterly effect. Since among the image and video features we almost always have some sort of color segmentation, it seems natural to use the already available results in producing the NPR effects. In our case we chose a Mean Shift [Com02a] based segmenter to generate lower and higher detail layers.

In the following subsections we will present the three main steps of the method.

Scene changes

For detecting scene changes, we use a motion based approach, similar to what is used in a variety of video encoders. We obtain local motion estimation on frames, and calculate the error of the estimations. We build a statistics of these errors, and when they cross a threshold, we signal a shot change. First we calculate a block matching based motion estimation on small frame blocks. For each block b_l (8x8 pixel size) from a frame (*t*) we search around its location r_l (with a step size of 4 pixels, in the maximum range of 4 locations in each direction) on the following frame (*t+1*) to find the same size block which matches best, by calculating a local error minima:

$$D = \min \left\| b_l^t(r_l), b_i^{t+1}(r_i) \right\| \qquad i = \overline{1,n}$$

Thus we obtain a motion direction vector for each frame block b_l, i.e. (v_x, v_y). Then, we calculate the estimation error on the frame blocks, based on the motion vector estimates above:

$$E_l = \left\| b_l^t(x_l, y_l), b^{t+1}(x_l + v_x, y_l + v_y) \right\|$$

and sum them up for the whole frame $E^t = \sum_{l=1}^{m} E_l$, where *m* is the total number of blocks. If the total error crosses a threshold then we signal a shot change position. Currently we have a database with over 6000 correctly segmented scenes and shots.

Foreground

For extracting the foreground, i.e. the changing areas between consecutive frames, we use a combination of a Stauffer-Grimson [Sta00a] foreground extractor and a focus area extractor.

We consider each pixel *s* as a separate process, which generates an observed pixel value sequence over time (*t* is the time index): $\left\{x^{[1]}(s), x^{[2]}(s), \ldots x^{[t]}(s)\right\}$. To model the recent history of the pixels, [Sta00a] suggested a mixture of *K* Gaussians distributions:

$$P\left(x^{[t]}(s)\right) = \sum_{k=1}^{K} w_k^{[t]}(s) \cdot \eta\left(x^{[t]}(s), \mu_k^{[t]}(s), \sigma_k^{[t]}(s)\right)$$

where $k = 1, \ldots, K$ are unique and in time static ID-s of the mixture components, while $\eta(.)$ is a Gaussian density function, with given μ mean and σ deviation. The idea is to model the image/frame pixels as of being a mixture of the Gaussians, which are then evaluated over time to determine which of them is more likely the background and which is the foreground model. Figure 1 (top right image) shows an example.

The focus area extractor builds upon a deconvolution-based region classification approach [Kov07a]. Basically, it is a method for automatically classifying image areas relative to each other based on the local blur/focus on the image, without a priori knowledge. The estimation of local blur and reconstructed region is an iterative approach based on [Ric72a], for estimating the local blur and the original unblurred region:

$$\begin{cases} f_{k+1}(r) = f_k(r)\left[h_k(r) * \dfrac{g}{g_k}(r)\right] \\ h_{k+1}(r) = \dfrac{h_k(r)}{\gamma}\left[f_k(r) * \dfrac{g}{g_k}(r)\right] \end{cases}$$

where *k* is the current iteration, γ is a weighting factor, *f* is the unknown original undistorted image (or region), *g* is the observed image region, *h* is the unknown point spread function which caused the distortion, thus $g = h * f$, meaning g is the convolution of some original signal with a distortion function. The classification is based on a local reconstruction error measure:

$$E_r(g, g_k) = \left| \arcsin \frac{< g - g_k, g >}{|g - g_k| \cdot |g|} \right| \cdot \frac{C_r(g_r)}{\max_r \{C_r(g_r)\}}$$

where r is the location vector in the image, g is the observation, index k denotes the actual reconstruction iteration, and the second part of the function stands for the local contrast. The values obtained are fed into a classifier. A sample result is in Figure 1 (bottom left) and Figure 2.

The final foreground is a logical or combination of the two sub-steps above, and samples are presented in Figure 1 (bottom right) and transformed foregrounds are shown in Figure 4.

The colorization is produced by reduced resolution color segmentation, for which we use Mean Shift [Com02a] segmentation. Larger homogenous areas can be produced quickly, giving the sensation of a low-detail blurred background. This provides an easy way for specifying the level of detail of the background and the foreground. A resulting rendered frame is shown in Figures 4, 5.

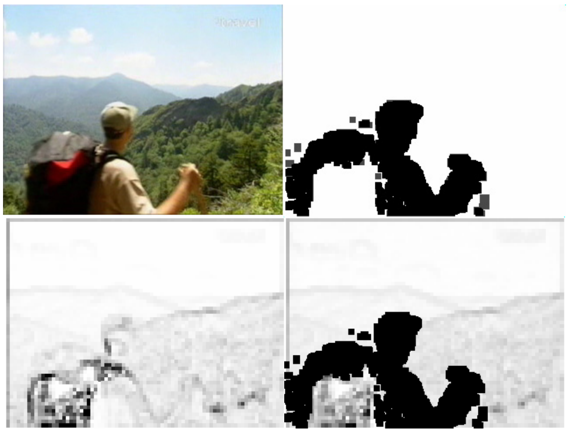

Figure 1. Input frame (top left), foreground mask (top right). Focus region mask (bottom left), final combined mask (bottom right).

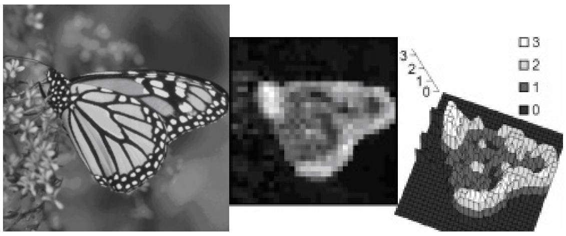

Figure 2. Focus area extraction sample.

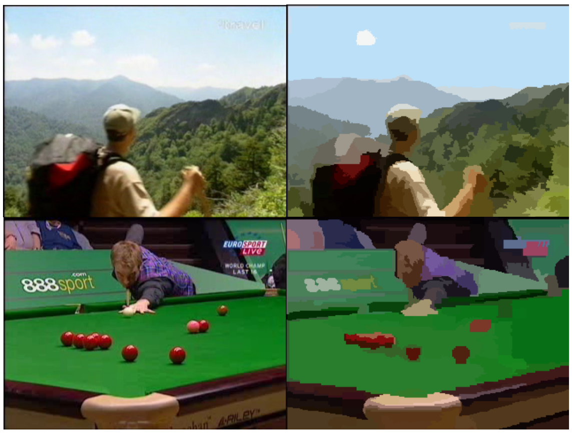

Figure 3. Input frames (left column) and backgrounds (right column).

Background

The background is generated as a low detail version of the current frame. For this purpose we use a Mean Shift color segmentation of the frame, with a high area setting and a low color count. The idea is to create a coarse representation of the frames, with larger consolidated areas. Figure 3 shows some examples. These generated segmentations will serve as the background of each frame.

Results and Application

As we do not use tracking, the question of inter-frame coherence arises. Since we transform every frame of the videos (or single frames, as requested) separately, we need to take care that color vibration does not cause disturbing artifacts. As a prevention step, we provide the possibility of adding a temporal blur to the resulting frame sequences, with a varying radius setting. Thus eventual color vibrations become unnoticeable and the computation complexity does not increase noticeably.

Using the above presented feature extractions, the final videos will be the result of foreground extraction and color segmentation steps. The specific steps have been chosen so that they should be already available features in a content based video retrieval application, mostly for speed and reaction time considerations.

As an additional possibility, we provide a step to create scalable vector graphical (SVG) versions of the rendered frames (one frame at a time). To do this, first, we extract the perimeter points of the background areas with a color area extracting algorithm (based on contour tracking with color similarity checking) and describe them as *path* SVG elements. Thus each region becomes a closed path, with specified stroke and fill color settings.

3. FUTURE WORK

We intend to include the described method into a web service which combines content based video retrieval with NPR video creation. Videos or segments/shots and frames will be available for transformation. The service will provide visually interesting versions of the uploaded video content, and ways to create content for enthusiasts and videographers. We intend to provide a style transfer method, i.e. transport a style of a rendering onto another input material.

4. CONCLUSIONS

We presented a method for automatic video NPR, which uses fairly simple features already available in most video databases intended for content based retrieval. The computational complexity of the approach is fairly low, yet the produced effect is visually appealing. Reproduction of the method is also quite easy, yet it provides a plethora of new possibilities for video database applications besides "simple" content based retrieval features.

5. ACKNOWLEDGMENTS

This work is partially supported by the JUMAS EU FP7 project.

6. REFERENCES

[Sta00a] Stauffer, C., Grimson, E. Learning patterns of activity using real-time tracking. IEEE Tr. on Pattern Analysis and Machine Intelligence, 22(8), pp. 747-757, 2000.

[Kov07a] Kovács, L., Szirányi, T. Focus area extraction by blind deconvolution for defining regions of interest. IEEE Tr. on Pattern Analysis and Machine Intelligence, 29(6), pp. 1080-1085, 2007.

[Lit97a] Litwinowicz, P. Processing Images and Video for An Impressionist Effect. Proc. of ACM SIGGRAPH 97, pp. 407-414, 1997.

[Kov02a] Kovács, L., Szirányi, T. Creating Animations Combining Stochastic Paintbrush Transformation and Motion Detection, In Proc. of 16th ICPR, pp. 1090-1093, 2002.

[Com02a] Comaniciu, D., Meer, P. Mean shift: A robust approach toward feature space analysis. IEEE Tr. on Pattern Analysis and Machine Intelligence 24(5), pp. 603–619, 2002.

[Hay04a] Hays, J., Essa, I. Image and video based painterly animation. In Proc. of NPAR, pp. 113–120, 2004.

[Col05a] Collomosse, J.P., Hall, P.M. Video Paintbox: The fine art of video painting, Computers & Graphics vol. 29, pp .862–870, 2005.

[Kim05a] Kim, B., Essa, I. Video-based nonphotorealistic and expressive illustration of motion. Computer Graphics International, pp. 32-35, 2005.

[Wan04a] Wang, J., Xu, Y.Q.., Shum, H.Y., Cohen, M.F. Video tooning. In Proc. of SIGGRAPH 2004, pp. 574-683, 2004.

[Ric72a] Richardson, W. Bayesian-based iterative method of image restoration. Journal of the Opt. Soc. of America vol. 62, pp. 55–59, 1972.

[Szl08a] Szlávik, Z., Kovács, L., Havasi, L., Benedek, Cs., Petrás, I., Utasi, Á., Licsár, A., Czúni, L., Szirányi, T. Behavior and event detection for annotation and surveillance. In Proc. of CBMI 2008, pp. 117-124, 2008.

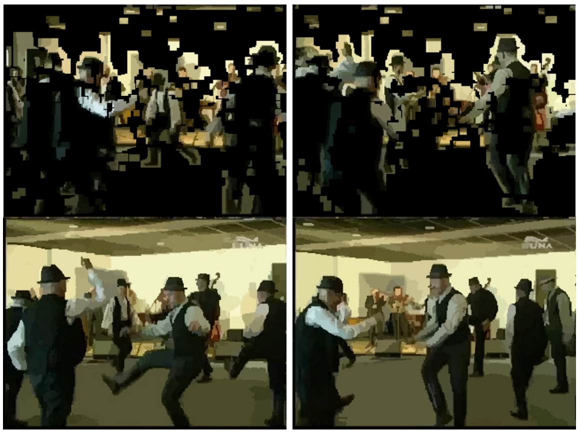

Figure 4. Rendered foreground layers (top). Transformed frames (bottom).

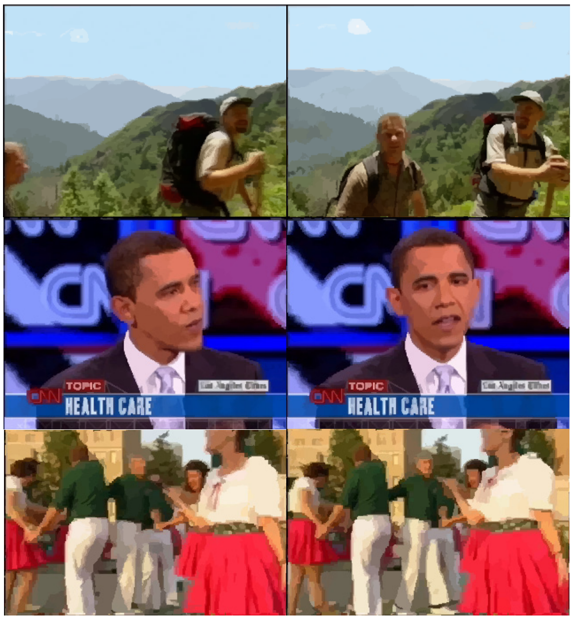

Figure 5. Transformed frames.