EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (*2000*), *Number 3*

A Vector-based Representation for Image Warping

Max Froumentin, Frédéric Labrosse, Philip Willis

University of Bath, UK

Abstract

A method for image analysis, representation and re-synthesis is introduced. Unlike other schemes it is not pixel based but rather represents a picture as vector data, from which an altered version of the original image can be rendered. Representing an image as vector data allows performing operations such as zooming, retouching or colourising, avoiding common problems associated with pixel image manipulation.

This paper brings together methods from the areas of computer vision, image compositing and image based rendering to prove that this type of image representation is a step towards accurate and efficient image manipulation.

1. Introduction

Image manipulation techniques like retouching, colourising or compositing are still very important to computer graphics both in their own right and as an adjunct to 3D synthesis. Two other advantages are that they allow using real-life pictures as input, and that they are usually faster for generating artificial pictures than rendering a 3D scene.

The main difficulties of manipulating real-life images are twofold. On the one hand, feature extraction (possibly with depth information) is necessary in order to be able to perform operations like rendering the depicted scene from a different point of view. On the other hand, occlusions between objects in the scene make such tasks as displacing or removing an element very difficult, as usually no information about what is behind that element is known from the original image.

These obstacles make 2D techniques tedious as they require a lot of human intervention in order to infer the missing information, or to clean up a resulting image, sometimes pixel by pixel.

In this paper we describe an image manipulation framework which performs analysis of a picture with reduced human intervention, extracts features from that picture and can re-synthesise a modified version of it by performing various operations such as alteration of colours, shape modification or object displacement.

1.1. Related Work

Most commercial drawing software systems offer basic image deformation, or *warping*, tools (like affine transforms, perspective mapping, ripple effects, *etc.*) and image modification, (*e.g.* cutting and pasting). In both cases, these systems are quite limited as either the warping effects do not depend on the contents of the image (like objects or people), or a lot of user intervention is needed in order to extract features, like defining contours.

Since Beier and Neely introduced feature-based image metamorphosis (or *morphing*[2]), there has been a great deal of interest in techniques for generating transitions between key images, simulating an object's movement or a change in the camera's point of view. Most of these methods either perform 2D interpolations between reference images[4, 18, 12] or infer 3D information from images. The latter can be done either from image sequences[19] or from user intervention[8].

Creating new images from a set of source images can also be done using compositing techniques[3]. The main problems associated with these are: extracting an object's contours (with techniques such as *blue screen matting*[21] or less restrictive ones[14, 13]) and correctly simulating lighting effects so that lighting interaction between the different objects is correctly rendered in the composite picture[23].

The main differences between those methods and ours are that:

- Our method works on single images rather than with image sequences.
- Images are completely represented, *i.e.* we do not extract

only one object but decompose the image into a complete covering.
- The contour extraction is region based and produces continuous, *i.e.* vectorial, descriptions at sub-pixel accuracy.
- Regions in which the image is decomposed do not have to correspond to physical objects but rather to homogeneous (although not necessarily having a constant colour) regions. It is indeed easier to render a primitive corresponding to an homogeneous region than a primitive corresponding to a complex object.
- For feature extraction, user intervention is limited to roughly selecting the parts of the image that are representative of wanted regions and these are used to produce a globally optimal segmentation of the image.
- No assumptions about image colours is made. In particular, colours do not have to be constant, even around the contours.

1.2. Overview

The following two sections discuss how our system analyses an image (Section 2) and how another image is rendered using data from the original (Section 3). Section 4 describes how the extracted data (and possibly the rendering process) can be modifed in order to perform various image manipulation operations. We conclude with some ideas of future research (Section 5).

2. Constructing Image Representations

In computer-based image manipulation applications, typically at the post-production stage of a live-action movie, a digitised version of each movie frame is used. By "digitised," we mean "turned into pixels." This kind of representation has one inherent problem: to reach the high image quality required by photography or 70mm cinema, digitisation must be performed at very high resolution, therefore producing a huge amount of image data. Moreover, due to the pixel nature of the data, methods of image alteration, like removing an object, are difficult to perform, as object contours are not well defined (they can for instance span several pixels in width because of the blur naturally present in photographs).

In this section we show how a continuous vectorial representation is extracted from an image. The extraction proceeds in two steps. First (Section 2.1), the image is decomposed into regions that correspond to specified colour features; we call them *structural regions.* Their contours (the *structural contours*) are continuous, smooth, and with sub-pixel accuracy. Second (Section 2.2), in order to be able to synthesise new images, we need to represent the interior of the structural contours. Therefore we extract some information inside the structural regions from the image. More details can be found in Labrosse *et al.*[11].

2.1. Structural Contours

The first part of the representation construction consists in decomposing the image into *structural regions, i.e.* regions corresponding to given colour characteristics. These are interactively specified by the user (see Section 4.2) by selecting parts of the image that are characteristic in terms of colour of the wanted regions. An interactive scheme is not too penalising since the longer-term aim is to process image sequences, and characteristics do not change dramatically between consecutive images.

Each manually specified part of the image is used to compute a colour average and variance. These statistical properties are used to compute a set of initial probabilities for each pixel to belong to each region (using the Mahalanobis distance, *i.e.* a distance to the average normalised by the variance). Probabilities are then enhanced and filtered using relaxation labelling[17, 7]. At the end of the process, we get a sharp and optimal decomposition of the image into homogeneous regions that are characteristic of the desired colour characteristics. Sharpness means that transitions between regions are spatially well determined, even if the transition is smooth in the original region. Optimality means that the transition is half-way, in terms of the Mahalanobis distance, between each colour characteristics.

Note that the regions do not correspond to physical objects of the scene but rather to homogeneous regions of the image. Indeed, the goal here is to represent the image, not the 3D scene. Moreover, disconnected regions possibly belonging to the same object, thus having the same colour characteristics, can be grouped if needed.

While segmentation can be based explicitly on textures, we found that this scheme works well even with textured images. We use it because of its more general applicability, *i.e.* for textured and non-textured images.

These regions, as well as probabilities produced by the relaxation labelling, are used to extract smooth continuous contours with sub-pixel accuracy. To achieve this, we use dynamic curves (or *snakes*)[9]. An energy function constituted of two terms is associated to each snake. The first term is the rigidity of the snake. The second one is its potential energy in a field made from a contour image. This image is the gradient modulus of the probabilities for each pixel to belong to the region whose boundary is extracted. Minimising the total energy tends to maximise the snake rigidity (except at determined discontinuity points) and make it follow the contours. A snake is initialised following the boundary of each region given by the segmentation (at pixel accuracy). Then snakes are optimised by minimising their energy as in Kass *et al.*[9] to reach sub-pixel accuracy. We use that method and not other proposed schemes[1, 6, 22] that overcome instabilities and convergence problems because they give a solution only at the pixel accuracy. Instead, we solved the often reported problems, namely that the method is instable, requires a good initial estimate and fine-tuning of several parameters.

Our initial estimate is very close to the solution since only sub-pixel displacements are required. We also developed a method to determine the needed parameters that is suited to our problem. For example, a coefficient (actually a function of the position in the snake) in the rigidity term controls the relative importance of the two terms in the energy function. It is determined to allow rigidity discontinuities in the snake (where high curvature points are detected) and to ensure an even evolution of both terms. Moreover, we construct the potential field by smoothly and continuously interpolating the contour image instead of keeping it spatially discrete as in the original method. The parameter values along with the new potential field solved the stability problems.

Tests we performed on the contour extraction of controlled shapes show that the error we obtain between the extracted contour and the original contour is less than 0.5% of the shapes' area†. Moreover, that error is symmetrical (the reconstructed contour is as often outside as inside of the original contour), which means that there is no global shift, and the maximum local error is less than a pixel at the resolution of the original image, that error only happening at very localised difficult events like very sharp angles.

2.2. Contour Interior

Once the structural contours are extracted, we need to represent their interior. Two basic schemes are proposed here. A third one (based on isochromatic contours) is described in a paper in preparation[10]. Others (based on texture representation) are currently under investigation.

The first scheme (*flat colours*) is particularly appropriate for "simplified" rendering for non-photorealistic applications such as cartoon-like rendering or technical illustration. A unique colour is associated with the region: the average colour of the corresponding image part that was selected by the user.

The second scheme is more suited to realistic rendering where continuous varying colours are needed: *smooth colours* are computed inside the regions from the original image. Each region is triangulated using a quality conforming Delaunay triangulation, where the area and angles of the triangles can be controlled[20]. A colour is then associated with each vertex of the triangulation, as a function of the colour of the nearest pixel in the original image. This function can be described as follows. The average and variance of the colour specified by the user define an ellipsoid in the colour space. The function is the identity for pixel colours inside that ellipsoid. For pixel colours that fall outside the ellipsoid, the colour is taken on the line from the colour towards its projection onto the ellipsoid (typically, the new colour is at half distance between the pixel's colour and its projection). By using this function, colours still reflect what was initially in the image but are closer to the region properties. This prevents colours which are very different from the ones inside the region spreading into it.

Colour projections are made in the CIE $L^*a^*b^*$ space (like all the colour manipulations). Indeed, these projections are not feasible in the RGB colour space since the results (as well as the Euclidian distance) do not correspond to any psychophysical reality, and would create colours that were visually wrong.

Note that the size of the triangles in the mesh must be of the order of the smallest detail to be represented, which means that for highly textured regions, the representation can be large in terms of number of primitives (and hence, texture based representations would be better in such cases). But this type of representation is well suited for smoothly varying regions or for smoothly varying rendered images.

2.3. Building the Representation

Each structural contour is saved as a vectorial curve, typically a NURBS curve. The curve is then associated with a texture primitive that is either a simple colour or a triangulation according to the chosen representation scheme. Each structural contour will thus produce a different object in the representation, which we shall be able to use independently of the others or with others having the same colour characteristics (*e.g.* different parts of the sky separated by objects reaching the image boundary).

3. Image Rendering

Once a description of the original image (we shall use Figure 4* as a starting point) as 2D vector information has been extracted, a new image can be re-synthesized from it. The new image will be either as close as possible to the original, or (more interestingly) a modified version of it.

The renderer we use is IRCS (Image Rendering and Compositing System[5]). Originally designed for rendering frames for cartoon animation, it reproduces a traditional *rostrum* setup, where characters are drawn on celluloid sheets and stacked together to form the final image. IRCS simulates this by organising 2D graphics primitives on layers, on which parameters such as lighting, brightness or blurring can be controlled independently of one another. The resulting image is computed by accurately simulating the interaction of light between layers.

Graphics primitives can be any type of 2D vector primitive. In this context, closed NURBS curves are used. The

† The errors are computed by rendering on the same image at very high resolution the reconstructed shape and the original shape (using their vectorial description, see Section 2.3), with two different colours and appropriate mixing rules such that pixels belonging to one shape *or* the other are of the corresponding colour. Then pixels having one or the other colour are counted and compared to the number of pixels in the original shape to produce the given percentage.

primitives can be structured in a hierarchical way, allowing 2D boolean set operations and concatenated matrix transforms.

Each node of a primitive tree can have a texture associated with it: single colours, colour gradients or bitmap images. Again, in this context only gradient textures are used, converted from the colour triangulations that the analysis produces. The colour model used by IRCS (an extension of the well-known RGBA representation[16]) allows accurate control of lighting and transparency by allowing the user to specifiy not only the colour of an object, but also that of the transparent medium on which the object is painted[15].

Three characteristics of IRCS make it particularly appropriate for our purpose.

- Each region from the data extraction process can be assigned to one layer of the $2\frac{1}{2}D$ structure, making it easier to manipulate it independently of the others. Special effects like changing the lighting of one layer can also be performed in a simple way.
- No frame buffer is used for rendering and compositing: each rendered scan-line is output directly to an image file.
- IRCS is efficient in the way it handles pixel coherency in pictures, working on several pixels at a time instead of each individual one.

These last two features offer the possibility of rendering very large images using a limited amount of memory: resolutions of up to $10^6 \times 10^6$ pixels have been achieved on a standard PC platform. This is very relevant to the resolution independance feature of the image extraction process.

4. Image Manipulation

Using the image analyser and the renderer described above, one can either try to produce an image as close to the original as possible (Figure 4*) or alter the vector representation to produce warped versions of it. This section enumerates some of the possibilities offered by the vector representation and shows that this system can perform as well as pixel based systems, but avoids some of the common problems of discrete data.

4.1. Features

Using the extraction and synthesis programmes, one can perform a variety of image transforms, some of which we describe here.

Zooming

Since the data extracted from the original image are in vector form, the first property is that it is resolution independent. Therefore the synthesized image can be rendered at any resolution without having to use filtering techniques.

Figure 5* shows a close-up of the original pixel image and a picture produced by changing the viewport of the vector data to show the same area of the original image. As expected, the vector representation allows zooming in a picture without either producing pixel artifact or having to use costly filtering techniques.

Warping objects

Colour coherent regions often represent image features. When these features are extracted and represented in vector form, the renderer is able to apply transforms to objects independently of the rest of the scene. An example is shown on Figure 6* (left). Here also, vector data allows great freedom in object modification without pixel resampling or filtering.

Of course, this kind of operation is very limited by occlusion problems: moving an object should reveal what is behind it, which is a piece of information that cannot be extracted from the original image. However extrapolating background information is possible by interpolation of the background texture. For small displacements or when the background is uniform, good results can be obtained. In Figure 6*, we deformed the vertices of the spline curve describing the goose's head and neck and replaced the missing parts of the background with texture elements (colour vertices) from the visible parts of that texture.

Merging objects

Objects extracted from different images can also be merged together in order to build a new image. An example is shown in Figure 6* (right) where an additional object (extracted from a different image) has been added to the scene. A vector representation also has the advantage over a pixel system that it is possible to merge pictures that have different resolutions together.

Of course, the main problem with this type of image manipulation is lighting correction: unless all source images are shot using the same lighting conditions (direction or intensity), colours, shading or shadows are inconsistent in the final image. Having all objects defined on separate layers of the rendering model allows the user to apply light correction to a particular object to try and correct lighting as best as possible.

Other types of transforms

Other transforms can be applied to any rendered object, either whole images or individual components. Examples include: blurring an object in the scene, changing its colour or the background's, anti-aliasing the re-rendered image, *etc.* An example of "defocusing" an object is shown on Figure 3.

Non Photorealistic Rendering

Beyond the ability to produce modified images that remain as photorealistic as the original, the data extracted can be rendered in a variety of non-photorealistic styles: one might want to render an image as if it had been drawn by a technical illustrator, as if it had been sketched and inked (as in cartoons and comic-strips), or painted (using a variety of techniques such as watercolour, airbrushing, *etc.*). The vector representation of image features makes it very simple to produce some of these effects, as is shown in Figure 1, where the outline of each region as well as the textures have been rendered using randomly distributed line segments to simulate a pen-and-ink illustration.

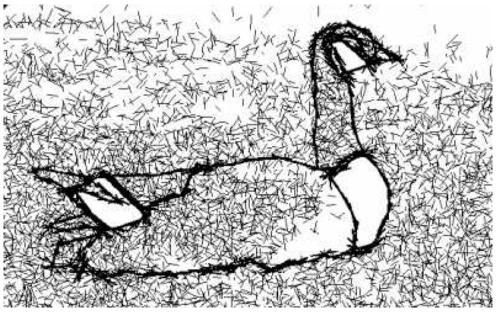


Figure 1: *Pen and Ink Rendering*

4.2. The User Interface

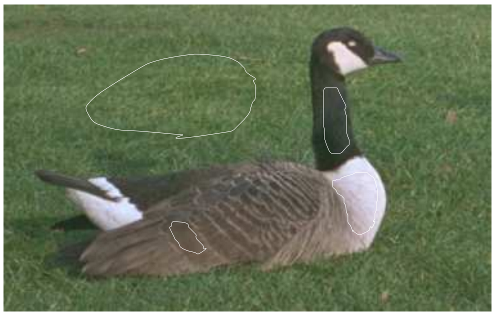
A graphical front-end has been developed for user interaction with both parts of the image analysis/synthesis package and for easy modification of the image representation.

ExRegions

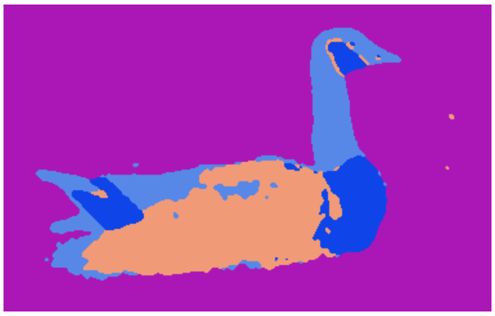
The amount of user intervention that is required for converting a picture into vector representation is typically very limited. First the user is asked to select the parts of the image that are representative of the wanted regions (see Section 2 and Figure 2(a)). Unlike traditional software where the user can spend a lot of time accurately outlining image features, our system allows very rough sketches to be drawn on the picture to be able to extract regions. Once the segmentation is done (Figure 2(b)), the user can select the regions he wants in the final representation. This can be done manually by clicking in the desired regions or automatically by providing a threshold on the area of the regions to consider. Then the snakes are created and optimised (Figure 2(c)).

CelEdit

All the data generated by ExRegions can be rendered, manipulated and saved, through the use of a wrapper interface. Each region can be modified individually (transforms, viewport, focus, *etc.*) and regions can be removed, added or moved between images using standard cut/copy/paste operations. Figure 3 shows an example of modification of the blur parameters of the goose's head using this interface.



(a)



(b)

(c)

Figure 2: *Representing an image: (a) defining image parts, (b) extracted regions, (c) final contours.*

5. Conclusion and Future Work

We have presented a system for analysing and rendering images. Homogenous regions are extracted from a given picture with minimal user intervention. These regions are represented in vector form with sub-pixel precision, forming a scale independent representation of the whole image. Modifying attributes of that representation allows one to warp the original image by altering the shape or the texture of one or several regions.

It is clear that this representation is not accurate enough as it is to correctly represent high frequencies in the original images (*e.g.* the grass background). Future research will mainly focus on a representation of textures which avoids the low-pass filter problem caused by under-sampling texture

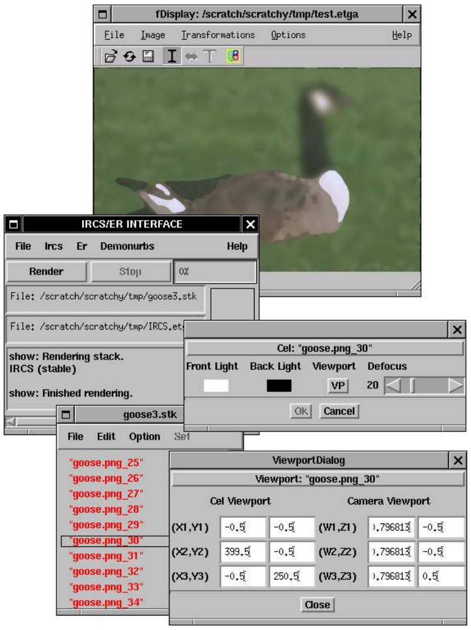


Figure 3: *The rendering interface*

colours. Common signal analysis tools like Fourier transforms should provide a better way of handling texture reproduction.

Another direction of further development is work on image sequences. Even if our image analysis system requires much less human intervention than other methods, it remains demanding if one has to work on sequences of several hundred frames. Global colour analysis of the original image may be useful to detect features with little user intervention. A different way would be to use inter-frame coherency, which would allow automatic tracking of a region across the sequence. Work has also started on keyframing: computing a sequence of images as the interpolation between two vector representations of an object, extracted from two different pictures.

The process of modifying and then rendering extracted data also offers interesting lines of development: a wider choice of object transforms (and corresponding user interfaces), as well as versatile rendering methods (non-photorealistic rendering for artistic drawing or technical illustration) should make our system suitable for a large range of digital imaging applications.

Acknowledgements

This work was supported by EU-TMR project *Platform for Animation and Virtual Reality* and EPSRC grant *Quasi-3D Compositing and Rendering*. The authors would like to thank Maurizio Scotti for designing and implementing the wrapper interface.

References

1. Amir A. Amini, Terry E. Weymouth, and Ramesh C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.
2. Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 35–42, 1992.
3. Ron Brinkmann. *The Art and Science of Digital Compositing*. Morgan-Kaufmann, 1999.
4. Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Computer Graphics (Proceedings of SIGGRAPH 93)*, pages 279–288, 1993.
5. M. Froumentin and P. J. Willis. An efficient $2\frac{1}{2}D$ rendering and compositing system. *Computer Graphics Forum (Proceedings of Eurographics '99)*, 18(3):385–394 and 428, 1999.
6. Davi Geiger, Alok Gupta, Luiz A. Costa, and John Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, 1995.
7. Michael W. Hansen and William E. Higgins. Relaxation methods for supervised image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):949–962, 1997.
8. Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Computer Graphics (Proceedings of SIGGRAPH 97)*, pages 225–232, 1997.
9. Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
10. Frédéric Labrosse, Sylvain Brugnot, Philip Willis, and John Patterson. Towards a continuous image representation: Sub-pixel contours. In preparation, 2000.
11. Frédéric Labrosse and Philip Willis. Towards a Continuous Image Representation: Sub-pixel Contours. Technical report, Department of Mathematical Sciences, Computing Group, University of Bath, UK, 2000. `http://www.maths.bath.ac.uk/~masfl`.
12. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 39–46, 1995.

13. Tomoo Mitsunaga, Taku Yokoyama, and Takashi Totsuka. AutoKey: Human assisted key extraction. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 265–272, Los Angeles, CA, USA, 1995.

14. Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *Proceedings of SIGGRAPH 95*, pages 191–198, Los Angeles, CA, USA, 1995.

15. Robert J. Oddy and Philip J. Willis. A physically based colour model. *Computer Graphics Forum (Proceedings of Eurographics '91)*, 10(2):121–127, 1991.

16. Thomas Porter and Tom Duff. Compositing digital images. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, pages 253–259, 1984.

17. Azriel Rosenfeld, Robert A. Hummel, and Steven W. Zucker. Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):420–433, 1976.

18. Steven M. Seitz and Charles R. Dyer. View morphing: Synthesizing 3d metamorphoses using image transforms. In *Computer Graphics (Proceedings of SIGGRAPH 96)*, pages 21–30, 1996.

19. Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *Computer Graphics (Proceedings of SIGGRAPH 98)*, pages 231–242, 1998.

20. Jonathan R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996. From the First ACM Workshop on Applied Computational Geometry.

21. Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Computer Graphics (Proceedings of SIGGRAPH 96)*, pages 259–268, 1996.

22. Donna J. Williams and Mubarak Shash. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.

23. Douglas E. Zongker, Dawn M. Werner, Brian Curless, and David H. Salesin. Environment matting and compositing. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 205–214, August 1999.

Figure 4: *Sample image (left) and image generated from vector representation (right)*

Figure 5: *Close-up of the original image (left) and from vector representation (right)*

Figure 6: *Special effects: warping (left) and merging (right)*