Texture Generation over the Marker Area

Sanni Siltanen
VTT Technical Research Center of Finland
P.O. Box 1000, FI-02044 VTT, Finland

Figure 1: Photorealistic hiding of the marker

ABSTRACT

In this paper, we present a method for generating a texture for hiding a marker in augmented reality applications. The texture is generated using the neighbourhood of the detected marker area in the image, which enables photorealistic results. The method presented here shows clear potential for real time use.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems - Artificial, augmented, and virtual realities; I.3.3 [Computer Graphics]: Picture/Image Generation

Keywords: augmented reality, mixed reality, image inpainting, texture generation, photorealistic rendering, hiding marker

1 INTRODUCTION

Markers are widely used in augmented reality applications for calculating 3D position and pose of the camera. For example the ARToolkit [1] that is widely used for AR application uses black and white square markers as in figure 1. In applications that aim for a realistic view, there is a need for hiding the marker from the final image, or at least to make it less eye-catching.

One approach is to use markers that are invisible to the human eye, but detectable to a computer as in the wearable AR system using an additional IR-camera with IR-LEDs and a retro-reflective marker described in [8]. Another approach is to use image processing to hide the visible marker from the view. In most cases markers are placed on a uniform background such as on the floor, and using a bilinearly interpolated square works well as stated in [2]. But if the marker lies on a textured background this approach results in a clearly visible square. Here we describe a method for generating texture using the area around the detected marker. This method gives realistic results on textured backgrounds (see figures 1 and 3).

2 RELATED WORK

Image inpainting means modifying an image in such a way that modifications are as unnoticeable as possible. Examples of image inpainting are correcting damaged images, removing objects from images and filling in missing blocks of transmitted images. The processing time is often counted in minutes as in [5] or in tens of seconds as in [6 and 9], or they require user interaction as in [9]. Hence, image inpainting methods are not directly suitable for real time applications.

Filling in missing blocks in transmitted video is also called error concealment. It is customary to utilize the information from the previous and following frame, as proposed for inter frames for H264/AVC in [7]. In a case of a marker based AR application, the same information is missing in all frames; hence no additional data is available in the time dimension. This makes the initial conditions for solving these problems different.

Mesh and grid generation aim for interpolating a surface using fixed data. In [3] a method is presented for extrapolating values to the inside of square using values along the edges. The method was originally presented in [4]. The basic idea is the same as in error concealment for intra frames in H264/AVC, where the weighted average of the edge pixels of correctly transmitted neighbouring blocks is used [7]. In our method we use values from outside of the area instead of border values.

3 TEXTURE GENERATION

We assume that in the near neighbourhood of the marker area the background is similar to the one covered with the marker. The texture outside each border is mirrored onto the inside so that it fades towards the other border. The source texture is limited to the inside of a certain area in the neighbourhood to assure its relevance. The impact of the surrounding pixels is inversely proportional to their distance to the border of the marker. The value at point (x,y) is

$$V(x,y) = \frac{-r\cdot s}{l^2}\cdot V(x_0,y_0) + \frac{-r\cdot(l-s)}{l^2}\cdot V(x_1,y_1) + \frac{-(l-r)\cdot(l-s)}{l^2}\cdot V(x_2,y_2)$$
$$+\frac{-(l-r)\cdot s}{l^2}\cdot V(x_3,y_3) + \frac{s}{l}\cdot V(x_4,y_4) + \frac{(l-s)}{l}\cdot V(x_5,y_5)$$
$$+\frac{r}{l}\cdot V(x_6,y_6) + \frac{(l-r)}{l}\cdot V(x_7,y_7),$$

where points (x_0,y_0)-(x_3,y_3) are the four corner points of the area and points (x_4,y_4)-(x_7,y_7) are points outside of the area. Here l is the width and height of the area, r and s are the distances from the border (see figure 2). The sum of the coefficients is one. The

Sanni.Siltanen@vtt.fi

yellow square in the figure 2 indicates the area that will be covered with the generated texture. The area between the blue and the yellow squares is the area that is used for texture generation. The outside points are mirrored against the border but then oscillated inside the desired band around the marker.

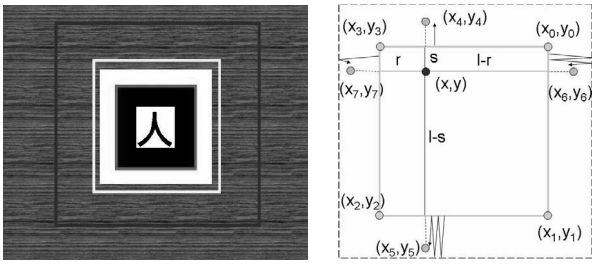

Figure 2: Area used for texture generation.

In general, when we fit a surface through control points, the calculated values may lie outside the range of the values at the control points. The same phenomenon may occur with this method. However in the implementation all values are truncated onto the range of the colour space.

We use the same coefficients as in [3], but in contrast to using edge values, we move the source values in the neighbourhood of the marker. We start by mirroring the texture on the border, from where we move the source points outwards until they reach the predefined boundary. Then we change the direction and start moving the source points towards the marker. If we reach the marker border, the direction is changed again and so forth.

We create the texture in 3D coordinates. Calculated positions are first projected on to the window coordinates, and then to the pixel coordinates with the nearest neighbourhood mapping. The value for the points in question is calculated and rounded to the nearest value in the range of colour space. Each colour component is treated separately.

4 Real time use and results

One implementation of this method that works in real time is to generate a texture once during the calibration process and use the same texture for every frame after that. Mapping a texture over a marker requires only a few OpenGL method calls and takes almost no processing time. A way to make the texture merge better into the background in varying lighting conditions is to make it partly transparent near the edges. This way the change in the darkness is not dramatic on the border of texture, but smooth over the transparent part. The rightmost image in figure 1 shows an example of this. The partly transparent texture was generated using the first frame, that is, the centremost image in the figure 1, and then applied for the following frames.

Without any optimizations the generation of a texture of the size 128*128 pixels took a bit less than 0.8 second and size 64*64 took 0.2 second (Pentium4/2.4GHz). The quality achieved by using a texture of size 64*64 would be acceptable in real time applications where the camera is moving.

This code was written to demonstrate the outcome of this texture generation algorithm, and no significant work has been put into performance issues as yet. We have a strong belief that with optimization it is possible to make this method fast enough to update the texture for every frame.

5 Conclusion and Future Work

It is clear that hiding a marker with a texture generated using the neighbourhood of the area gives a natural output. The results of the method presented here are promising for real time application and the future development is straightforward; we will optimize the performance. The performance of the algorithm could be further improved by using GPU shaders.

A variety of AR applications are based on using still images or offline processing: virtual tours, AR advertising videos, etc. In these cases also more sophisticated image inpainting algorithms could be used.

References

[1] ARToolKit: http://www.hitl.washington.edu/artoolkit/

[2] Siltanen, S. and Woodward, C. (2006). In Proc. Seventh Australasian User Interface Conference (AUIC2006), Hobart, Australia. CRPIT, 50. Piekarski, W., Ed. ACS. 33-36.

[3] J. F. Thompson, B. K. Soni, and N. P. Weatherill, editors. *Handbook of Grid Generations* CRC Press, 1999.

[4] W. Gordon and C. Hall, "Construction of curvilinear co-ordinate systems and applications to mesh generation", Int. J. Numer. Methods Eng. 7, 461-477, 1973.

[5] M. Bertalmio, G. Sapiro, C. Ballester and V. Caselles, "Image inpainting," Computer Graphics, SIGGRAPH 2000, July 2000.

[6] Kokaram, A., "Parametric texture synthesis for filling holes in pictures," *Image Processing. 2002. Proceedings. 2002 International Conference on* , vol.1, no.pp. I-325- I-328 vol.1, 2002

[7] Ye-Kui Wang; Hannuksela, M.M.; Varsa, V.; Hourunranta, A.; Gabbouj, M., "The error concealment feature in the H.26L test model," *Image Processing. 2002. Proceedings. 2002 International Conference on* , vol.2, no.pp. II-729- II-732 vol.2, 2002

[8] Y. Nakazato, M. Kanbara, and N. Yokoya: "Wearable augmented reality system using invisible visual markers and an IR camera", *Pro. IEEE Int. Sympo. on Wearable Computers*, pp. 198-199, 2005.

[9] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum: "Image Completion with Structure Propagation", *ACM Transactions on Graphics* (SIGGRAPH 2005).

Figure 3: Our method on different kinds of backgrounds