



An image-based texture mapping technique for apparel products exhibition and interior design

P. Wang^{a,b,*}, M.M. Zhang^b, J.H. Xin^{a,1}, Z.G. Pan^b, H.L. Shen^a

^a*Institute of Textiles and Clothing, The Hong Kong Polytechnic University, Hong Kong, China*

^b*State Key Lab of CAD&CG, Zhejiang University, Hang Zhou 310027, China*

Received 21 November 2003; revised 21 November 2003; accepted 6 January 2004

Abstract

In this paper, we propose an image-based texture mapping technique for textile products exhibition that does not require geometric representation of 3D models. Under this technique, a texture grid is built interactively for a target area in an original image. This grid acts as the intermediate space between planar space and texture space. The texture coordinate for each pixel in the target area can be calculated based on this grid, and the 3D effect can be successfully realized by further fine adjustment of the grid. This technique can be applied in apparel products exhibition and interior design.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Texture mapping; Parameterization; Computer graphics

1. Introduction

Texture mapping is a technique commonly used in computer graphics to enhance the realistic effects of computer-synthesized images [7]. It generally focuses on building a map from a 2D image (texture space) to the surface of a 3D model so that each point on the surface is associated with a corresponding pixel in the texture space. Some previous algorithms built a parametric representation of the surface [3,4] to construct such a map. However, such representation does not exist in the commonly used polygonal mesh and the application of those previous algorithms to the polygonal mesh will fail to perform the texture mapping. A possible solution to this problem is to use an intermediate surface with simple shape that can be efficiently mapped to the texture space, and then find a map from the surface of a 3D model [2]. These algorithms often result in high distortion due to the non-linearity of the texture map [11]. Therefore, many optimization algorithms have been proposed to improve the mapping so as to obtain

natural mapping results [1,6,11–15,17–20,21]. A relatively new technique, namely projective texture mapping, has been studied by Devich and Weinhaus [5] and other researchers [8,20]. Instead of assigning fixed texture coordinates to geometry, this technique projects a texture map (for example, panoramic image) onto geometry [8,20].

Although the texture mapping has been studied intensively, almost none of the previous algorithms perform perfectly in eliminating texture distortion and texture aliasing. In addition, many of those algorithms need 3D models, and the projective texture mapping requires panoramic images. Therefore, the potential use of those texture mapping techniques has been greatly limited. For example, they cannot be easily accomplished in internet-based applications due to the huge amount of data involved when dealing with 3D models. It is also difficult for a novice to build a 3D model to accomplish his task. Therefore, an image-based technique was presented in 2000 [23], which used a concept of texture area to accomplish texture mapping. It eliminated the drawbacks caused by 3D models, but it will encounter difficulty when simulating slight texture changes.

In this paper, we propose a novel image-based texture mapping technique. It is very effective to mapping texture on soft and flexible objects such as clothes and counterpane. We build a texture grid interactively to map texture onto

* Corresponding author. Address: State Key Lab of CAD&CG, Zhejiang University, Hang Zhou 316001, China. Tel.: +86-57187951045; fax: +852-2773-1432.

E-mail addresses: wangpeng@cad.zju.edu.cn (P. Wang), tcxinjh@inet.polyu.edu.hk (J.H. Xin).

¹ Tel.: +852-2766-6474; fax: +852-2773-1432.

the target area. Since the mapping from texture grid to texture image is linear, the texture distortion can be eliminated. As the 3D model is no longer needed and the computation is very effective, the proposed technique can be easily adopted by both internet-based and stand-alone applications for visualization and exhibition of apparel products and interior design. There are some image-based proprietary products available to deal with texture mapping using grid generation and user interaction [9,16]. However, the techniques have not been published, and their accuracies are unknown.

2. Overview and the preprocessing work

As the proposed technique is image based, the texture distortion caused by traditional 2D to 3D texture mapping techniques can be avoided. The main problem in our mapping method is to find a 2D texture space that has a linear transformation relationship with the planar space of the original image. In most cases, the real surface of the object for simulation is not homeomorphic to a plane, so we cannot find a space to match the whole target area [21]. Instead, individual patches of a texture space should be generated for different portion of the target area in images. The generated patches construct a grid, namely texture grid, with which the texture coordinates of a pixel in the target area can be easily calculated. It is very complicated to generate these patches individually. We propose a new method to generate the whole texture grid with simple interactions and local adjustments. This method consists of four steps:

Step 1: Preprocessing.

Step 2: Generation of texture grid.

Step 3: Local adjustment of texture grid.

Step 4: Mapping texture image onto target area.

These four steps are discussed in detail in the following sections.

3. Preprocessing

In the preprocessing step, users need to provide some information interactively. First, a target area in the original image should be outlined by the users, which is indicated by the red boundary polygon in Fig. 1. Then the users should input some polygonal lines to indicate the texture directions in the area, which are termed as texture frame lines in this study. There are two kinds of frame lines, horizontal and vertical. The horizontal frame line represents the change of horizontal texture direction in the target area, while the vertical frame line represents the change of vertical texture direction. In Fig. 1, the horizontal frame lines are in green, and the vertical ones are in blue.

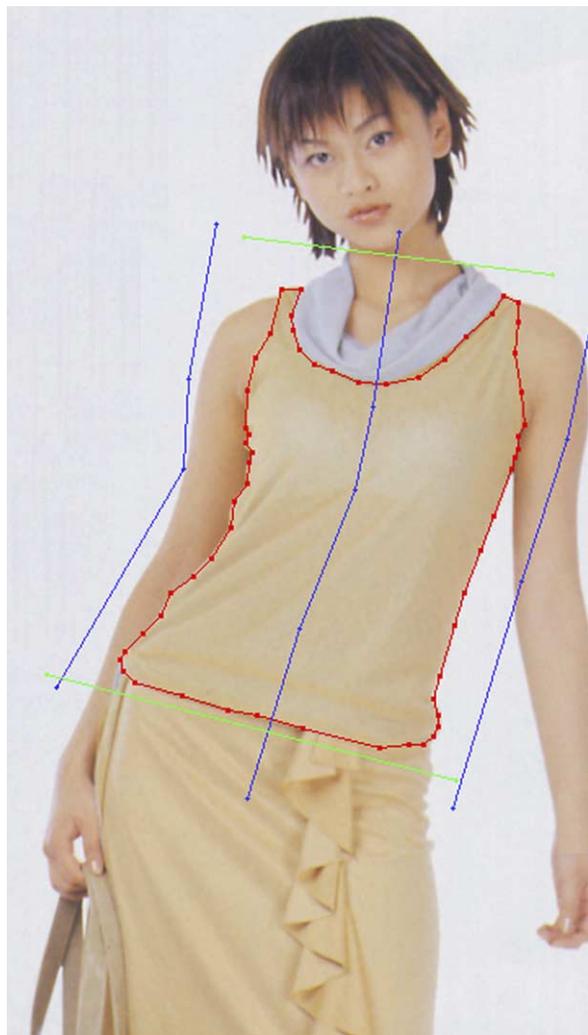


Fig. 1. Boundary and texture frame lines.

After receiving these inputs, some necessary preprocess should be performed. First, the sequence of the frame lines should be rearranged, since users may draw these lines in a random order. The new sequence is from top to bottom for the horizontal frame lines, and from left to right for the vertical frame lines. Then we use the four outmost frame lines to construct a closed polygonal of the grid. If one of frame lines is not long enough to intersect, it will be expanded up to the bounding box of all these four lines. The result is shown in Fig. 2(b). Finally, we adjust the lengths of the rest of the frame lines. If one frame line exceeds the boundary of the region, it will be clipped; if it cannot reach the boundary, it will be expanded as shown in Fig. 2(c).

4. Generation of the texture grid

The texture grid is built based on the processed frame lines. First, we calculate the gradual frame lines based on the given horizontal and vertical frame lines, respectively.

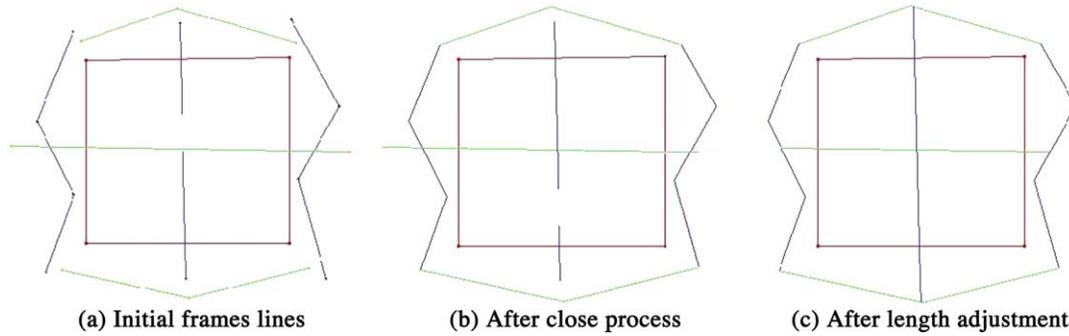


Fig. 2. Preprocess works.

These lines should change gradually from one given line to another. Then these gradual lines construct virtual horizontal and vertical texture grids. At last, these two grids are combined to generate the final texture grid. The detailed procedures are given in Sections 4.1–4.3.

4.1. Parameterization

To calculate the gradual frame lines using interpolation, the texture frame lines must be parameterized. Let L_i denote one gradual frame line, the discrete parametric representation of L_i can be defined as:

$$L_i = \begin{cases} X_i(t) \\ Y_i(t) \end{cases} \quad (1)$$

For each vertex of line L_i , there is a corresponding parameter t . The value of t is defined as the ratio of the length from the beginning vertex to the current one and the total length of the line. Then the coordinate functions $X_i(t)$ and $Y_i(t)$ can be simply represented by the coordinate of the corresponding vertex. For L_i , a set of t is calculated. As the gradual lines should reflect the characteristics of all the input lines, we calculate the union of the parameter sets of these input lines and resample these lines using this union. Finally, all the given texture frame lines are discretely parameterized based on the same parameter set.

4.2. Generation of the virtual texture grids

As the methods of generating these two grids are similar, we only discuss the horizontal one in this section. We use linear interpolation to calculate the gradual lines. During this process, a weighting should be defined for every line. Given a line L_i , we take y coordinate of the center point of its bounding box as the weighting factor w_i . A series of weightings of the gradual lines, $w_{g1}, w_{g2}, \dots, w_{gm}$, is calculated by dividing the interval $[w_1, w_n]$ equally, where m is the number of the gradual lines. Using the weighting w_{gj} , the gradual line's parameter representation is given as:

$$X_j(t) = \frac{\sum_{i=1}^n \frac{X_i(t)}{|w_i - w_{gj}|}}{\sum_{i=1}^n \frac{1}{|w_i - w_{gj}|}}, \quad Y_j(t) = \frac{\sum_{i=1}^n \frac{Y_i(t)}{|w_i - w_{gj}|}}{\sum_{i=1}^n \frac{1}{|w_i - w_{gj}|}} \quad (2)$$

where $X_i(t)$ and $Y_i(t)$ are the parameter representations of input frame line L_i ; and n is the number of the input frame lines. The calculated gradual frame lines are shown in Fig. 3(b).

As the weightings of the gradual lines are uniform, we use a uniform parameter set, $\{i/(N-1) | i = 0, 1, \dots, (N-1)\}$, where N is the number of columns, to resample each gradual line. Finally, all the vertexes of these frame lines are

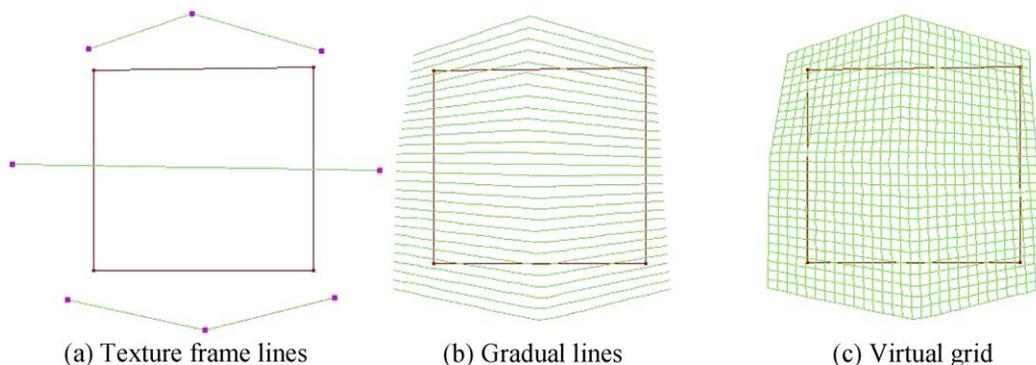


Fig. 3. Building horizontal virtual texture grid.

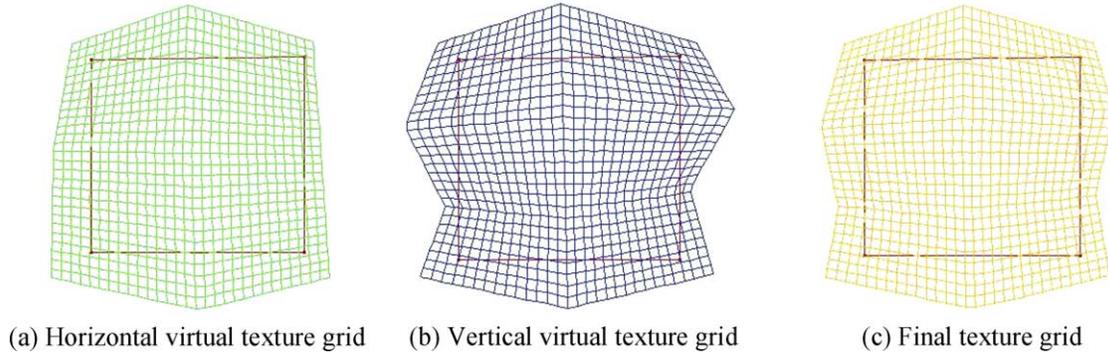


Fig. 4. Generation of texture grid.

used to construct the virtual texture grid as shown in Fig. 3(c).

4.3. Union

After the generation of the two virtual horizontal and vertical texture grids, the final texture grid can be simply built by their weighted combination. Usually, users pay similar attention to the horizontal and vertical texture directions. Therefore, to determine a vertex on the grid, we use the middle point of the line whose head and tail vertexes are the corresponding vertexes on the two virtual grids. As Fig. 4 shows, the resultant grid contains the characteristics of both the horizontal and vertical frame lines.

In some cases, however, users may pay more attention to one texture direction than the other, and the characteristics of the final grid should reflect this accordingly. For example, in the simulation of a sleeve, the vertical frame lines should affect the final grid more than the horizontal ones. Another similar situation is the simulation of garment folds. If we still use the middle point method mentioned before, the characteristics will be lost. Therefore, an adjustment step needs to be introduced to achieve better results in these cases.

The first step of the adjustment process is to decide whether the resultant grid should be adjusted. As we have built a grid using the middle point method, which is suitable in most cases, we take this grid as the reference grid. The differences between the reference grid and the two virtual horizontal and vertical grids are calculated. These differences should indicate whether the adjustment step is necessary.

For a texture frame line, the most important characteristic is indicated by the included angles at its vertexes. As these angles represent the final texture directions, the changes of these directions will generate 3D effects. Therefore, these angles can be used to calculate the differences.

We consider each row in the calculation of the difference between the reference and horizontal virtual grid. Given a row r of the reference grid, the corresponding row of

the horizontal virtual grid is denoted as r_h (the subscript v and h denote vertical and horizontal direction, respectively). We calculate the included angles at all the vertexes of r and r_h , except the first and last vertexes. Let an included angle at a vertex of r be θ . If $|\theta - \pi| \leq K_1$, where K_1 is a threshold, the two edges of angle θ are considered to belong to a straight line, and this vertex will not be checked. Otherwise, the difference between θ and its counterpart angle θ_h on r_h , is calculated. If $|\theta - \theta_h|/\theta > K_2$, where K_2 is another threshold, r_h is considered to require adjustment. In this study, the values of $\pi/10$ and 0.5 are thought to be appropriate for the thresholds K_1 and K_2 , respectively. If there are more than a half number of rows of the horizontal virtual grid need adjustment, the horizontal grid should be adjusted. In the vertical case, we check each column of the vertical grid using the similar method.

Given a vertex V of the resultant grid, its counterparts of horizontal and vertical virtual grids are denoted as V_h and V_v , respectively. The coordinate of V is calculated as follows:

$$V = \frac{K_h \times V_h + K_v \times V_v}{K_h + K_v} \quad (3)$$

where K_h and K_v are weighting factors. In the calculation of the reference grid, their values are both 0.5. In the grid adjustment, their values should be calculated based on the characteristics of frame lines. If L_i is a horizontal frame line, we take the horizontal middle line of its bounding box as the base line. We calculate the distances between those vertexes and the corresponding base lines, and add these distances up to C_i . The value of C_i may reflect the characteristic of frame line L_i . For all the initial horizontal and vertical frame lines, their averages, \bar{C}_h and \bar{C}_v , and variances, $D(C_h)$ and $D(C_v)$, are further calculated. If $|\bar{C}_h - \bar{C}_v| > \min(\bar{C}_h, \bar{C}_v)/2$, the two weightings K_h and K_v are calculated as:

$$K_h = \frac{\bar{C}_h}{\bar{C}_h + \bar{C}_v}, \quad K_v = \frac{\bar{C}_v}{\bar{C}_h + \bar{C}_v} \quad (4)$$

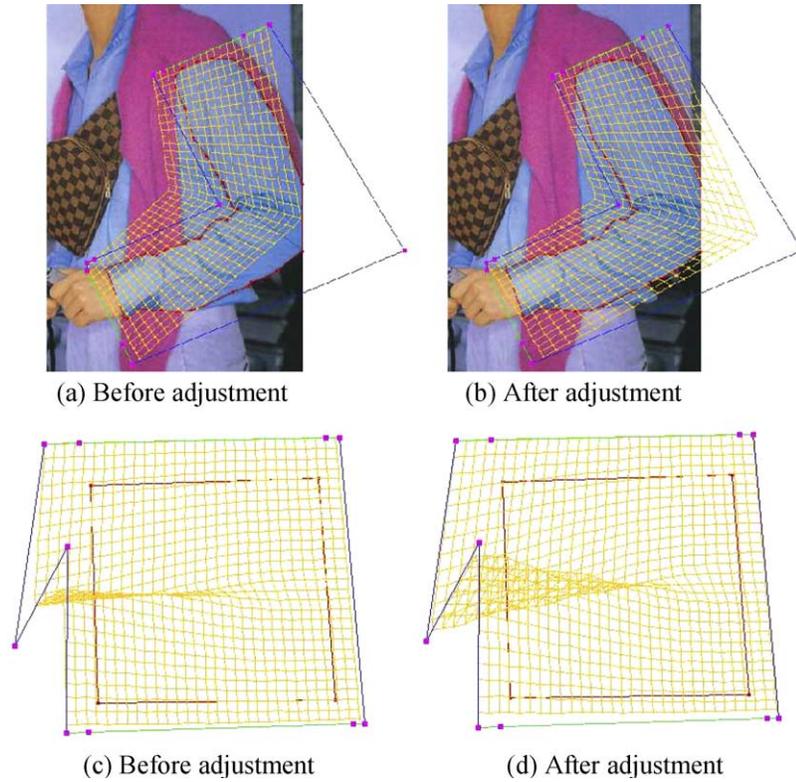


Fig. 5. Result of adjustment.

Otherwise, they become

$$K_h = \frac{D(C_h)}{D(C_h) + D(C_v)}, \quad K_v = \frac{D(C_v)}{D(C_h) + D(C_v)} \quad (5)$$

Some examples for the grid adjustment are shown in Fig. 5.

5. Interactive adjustment

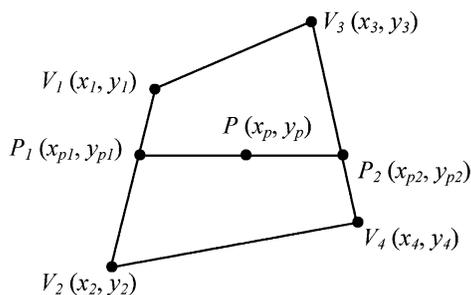
The texture grid represents the global texture directions in the target area selected by users. However, it may not be sufficient in reflecting the real 3D effect. For instance, in the simulation of soft and flexible textile products, there are always many wrinkles that cannot be successfully

represented by the generated texture grid. Therefore, some operations are provided so that users can adjust the texture grid when necessary. In the technique developed in this study, users can adjust a single vertex, a line or a rectangle. Furthermore, the grid density is also adjustable so that users can perform texture mapping in different detail level.

6. Texture mapping

Using the texture grid, we can calculate the texture coordinate of a pixel in the target area. First, the texture coordinates of all the vertices of the grid are calculated. For a vertex at i th row and j th column, its texture coordinate is

Planar Space



Texture Space

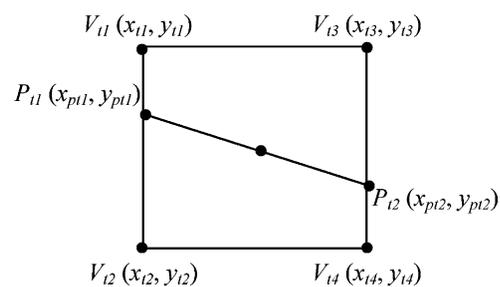


Fig. 6. Calculating texture coordinates.



Fig. 7. Example 1.

$y_t = i/(r - 1)$, $x_t = j/(l - 1)$, where r and l are the numbers of rows and columns, respectively. For a pixel P in a quadrangle consisted by vertexes V_1 , V_2 , V_3 , and V_4 , we can calculate its texture coordinates using the coordinates of

these four vertexes. Fig. 6 shows these vertexes in the planar space and texture space. P_1P_2 is a horizontal line passing through P . The x direction texture coordinate of pixel P_1 is the same as that of V_1 and V_3 , and the y direction texture coordinate is given by Eq. (6):

$$y_{pt1} = \begin{cases} y_{t1} + \frac{(y_{p1} - y_1)(y_{t1} - y_{t2})}{y_1 - y_2} & (y_1 \neq y_2) \\ \frac{y_{t1} + y_{t2}}{2} & (y_1 = y_2) \end{cases} \quad (6)$$

Then the texture coordinates of P_2 can be calculated in a similar way. Therefore, the texture coordinates of P is given by:

$$\begin{cases} x_{pt} = \begin{cases} x_{pt1} + \frac{(x_p - x_{p1})(x_{pt2} - x_{pt1})}{x_{p2} - x_{p1}} & (x_{p2} \neq x_{p1}) \\ \frac{x_{pt1} + x_{pt2}}{2} & (x_{p2} = x_{p1}) \end{cases} \\ y_{pt} = \begin{cases} y_{pt1} + \frac{(x_p - x_{p1})(y_{pt2} - y_{pt1})}{x_{p2} - x_{p1}} & (x_{p2} \neq x_{p1}) \\ \frac{y_{pt1} + y_{pt2}}{2} & (x_{p2} = x_{p1}) \end{cases} \end{cases} \quad (7)$$

After the calculation of texture coordinates, we simply take the nearest neighbor pixel on the texture image as the texture of P . Finally, we make a color-blending operation in CIELAB color space [19] to reflect the lightness distribution of the original image. It is known that the color information is mainly recorded in the chromaticity coordinates and

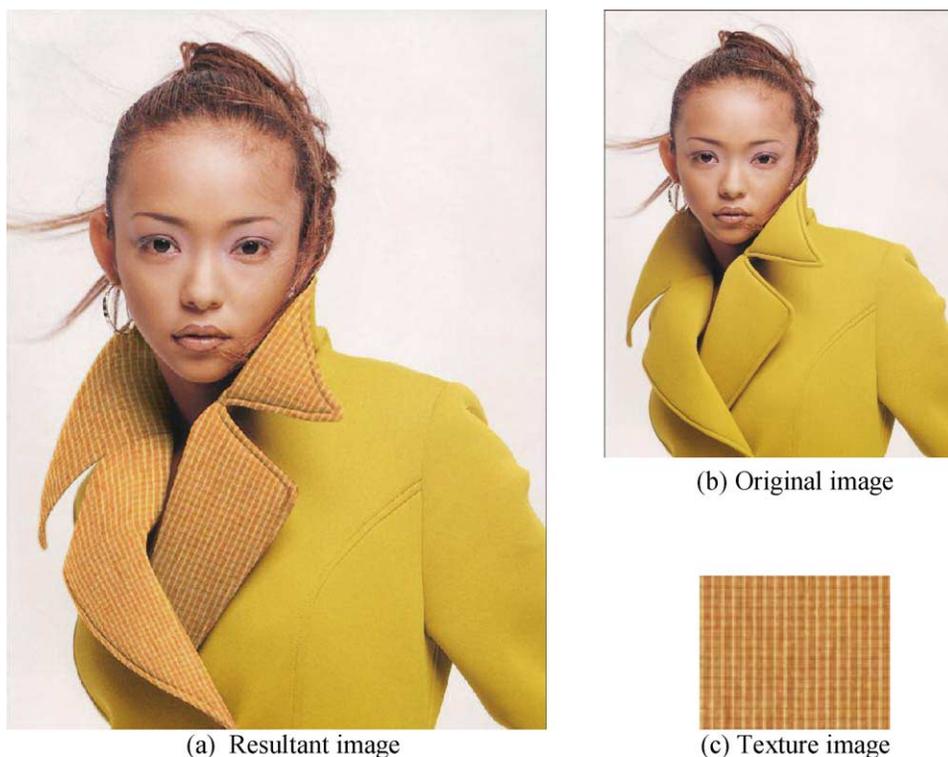


Fig. 8. Example 2.

the intensity information is mainly recorded in the lightness channel [18]. Therefore, in the color-blending process, we only consider the change of lightness and assume constant chromaticity. For a pixel in the target area, let the lightness of its original color be L_o , the average lightness of the whole area be L_m , and its texture color be (L_t, a_t, b_t) . The lightness after color blending is calculated as

$$L = \begin{cases} L_t + \frac{(L_{\max} - L_t)(L_o - L_m)}{L_{\max} - L_m} & \text{if } (L_o - L_m \geq 0) \\ L_t + \frac{(L_t - L_{\min})(L_o - L_m)}{L_m - L_{\min}} & \text{otherwise} \end{cases} \quad (8)$$

where L_{\max} and L_{\min} represent the maximum and minimum lightness in CIELAB color space, respectively.

Accordingly, the new color is (L, a_t, b_t) after the color-blending process.

7. Simulation examples

Some simulation examples of texture mapping using the proposed technique are shown in Figs. 7–9. For each example, the target areas are selected interactively in the original images, and the texture images are used to simulate new images according to the four steps discussed above. In Fig. 7, the complex texture directions and the stretch effect of the cloth are well simulated. The 3D effect of smooth curved face of the coat collar is also realistically represented in Fig. 8. The last example further shows the good performance of the color-blending technique.



(a) Resultant image



(b) Original image



(c) Texture images

Fig. 9. Example 3.

8. Conclusion

We studied a texture mapping technique for the simulation of soft and flexible objects such as apparel products. A texture grid was introduced to act as the intermediate space between the planar and texture spaces. Since users can interactively build and modify the texture grid, the target object can be simulated flexibly. Compared with the traditional texture mapping techniques for polygonal models, our technique dramatically reduces the computation and storage in simulation process, as it is image based. Another advantage of our technique is the simpleness of using especially for amateur users in comparison with the transitional techniques using polygonal models. The simulation examples demonstrate that our technique is very useful in potential applications in apparel and interior design exhibitions. In our implementation, user may need to adjust the texture grid locally to simulate the special effect such as wrinkles on the clothes. The color-blending process used in this work was considered to be adequate in simulating realistic and pleasing effect for the majority of garment. However, it was not intended to provide high color fidelity. Further work need to be conducted in accurate color rendering based on the investigation of the interaction between light and object surface [10,22,24]. The automatic generation of the texture grid according to the scenes characteristics based on image analysis is another challenging task.

Acknowledgements

The authors wish to acknowledge the funding of this project from the Hong Kong Polytechnic University.

References

- [1] C. Bennis, J.M. Vezien, G. Iglesias, Piecewise flattening for non-distorted texture mapping, *Compute Graphics* 25 (4) (1991) 237–246.
- [2] E. Bier, K. Sloan, Two-part texture mapping, *IEEE Comput. Graphics Appl.* 6 (9) (1986) 40–53.
- [3] J.F. Blinn, M.E. Newell, Texture and reflection in computer generated images, *Comm. ACM* 19 (10) (1976) 542–547.
- [4] E. Catmull, A Subdivision Algorithm for Computer Display of Curved Surfaces, PhD Thesis. Department of Computer Science of Utah University, 1974.
- [5] F.M. Weinhaus, R.N. Devich, Photogrammetric texture mapping onto planar polygons, *Graphical Models Image Processing* 61 (2) (1999) 63–83.
- [6] M.S. Floater, Parametrization and smooth interpolation in geometric modeling, *ACM Trans. Comput. Graphics* 8 (2) (1997) 121–144.
- [7] P.S. Heckbert, Survey of texture mapping, *IEEE Comput. Graphics Appl.* 6 (11) (1986) 56–67.
- [8] D. Kim, J.K. Hahn, Projective texture mapping with full panorama, *Comput. Graphics Forum* 21 (3) (2002) 421–430.
- [9] Lectra, see <http://www.lectra.com/en/index.php>.
- [10] H.C. Lee, E.J. Breneman, C.P. Schulte, Modeling light reflection for computer vision, *IEEE Trans. Pattern Anal. Mach. Intell.* 12 (4) (1990) 402–409.
- [11] B. Levy, J.L. Mallet, Non-distortion texture mapping for sheared triangulated meshes, *Proceedings ACM SIGGRAPH* 98, pp. 343–352.
- [12] Bruno Lévy, Constrained texture mapping for polygonal meshes, *Proceedings ACM SIGGRAPH'2001*, pp. 417–424.
- [13] P. Litwinowicz, G. Miller, Efficient techniques for interactive texture placement, *Proceedings ACM SIGGRAPH'94*, pp. 119–122.
- [14] S.D. Ma, H. Lin, Optimal texture mapping, *Proceedings EUROGRAPHICS'88*, pp. 421–428.
- [15] J. Maillot, H. Yahia, A. Verroust, Interactive texture mapping, *Comput. Graphics* 27 (3) (1993) 27–34.
- [16] Nedgraphics, see <http://www.nedgraphics.com>.
- [17] H.K. Pedersen, Decorating implicit surfaces, *Proceedings ACM SIGGRAPH'95*, pp. 291–300.
- [18] K.N. Plataniotis, A.N. Venetsanopoulos, *Colour Image Processing and Application*, Springer, Berlin, 2000.
- [19] Publication CIE, *Colorimetry, Second ed.*, Central bureau of the CIE, 1986.
- [20] A. Sheffer, E.D. Sturler, Smoothing an overlay grid to minimize linear distortion in texture mapping, *ACM Trans. Graphics* 21 (4) (2002) 874–890.
- [21] H.Q. Sun, H.J. Bao, Interactive texture mapping for polygonal models, *Comput. Geometry* 15 (1) (2000) 41–49.
- [22] S. Tominaga, Dichromatic reflection models for rendering object surfaces, *Imaging Sci. Technol.* 40 (6) (1996) 549–555.
- [23] Peng Wang, A new texture morphing method for visual presentation of textile product, *Proceedings ICIG'2002*, pp. 1011–1016.
- [24] J.H. Xin, H.L. Shen, A computational model for colour mapping on texture image, *Electron. Imaging* (2003) 12.