Harmonised Texture Mapping

Takeshi Takai
University of Surrey, UK
t.takai@surrey.ac.uk

Adrian Hilton
University of Surrey, UK
a.hilton@surrey.ac.uk

Takashi Matsuyama
Kyoto University, Japan
tm@i.kyoto-u.ac.jp

Abstract

This paper presents a novel method for rendering non-blurred and high quality free viewpoint images of 3D video with camera parameters and a reconstructed 3D shape that are not perfectly accurate. The contributions of this paper are proposing methods for: 1) view-dependent texture deformation for generating consistent textures and 2) optimisation of a 3D mesh with regard to texture consistency and shape preservation which cooperates with the deformation. By using these methods, our approach implicitly performs non linear deformation of textures to reduce such artifacts as blurring and ghosting of rendered images. In this paper, we first describe what is the problem and how to cope with it, and then demonstrate effectiveness of the proposed method with real scenes.

1 Introduction

3D video is currently a popular method for reconstructing object's 3D shape and textures with dynamic motions from multiple view video. In the last decade, several groups have developed 3D video technologies [7, 9, 8, 10, 11, 1] and it is now becoming an applicative technique for media productions, *e.g.*, film and games [5, 6]. One of the advantages of 3D video among the computer graphics and computer vision technologies such as motion capture, image-based rendering, *etc.*. is that it can record the complete dynamic 3D shape, motion and texture seen from multiple cameras. As 3D video represents the captured object using 3D meshes with textures, it can be edited in a conventional computer graphics pipeline, *e.g.*, rigging, retargeting, adding virtual objects, placing on a background model, *etc.*,

3D video reconstruction often results in a loss of visual quality compared to the original video which is caused by inaccuracy of:

Camera parameters: Errors occur due to the difficulty of accurately calibrating multiple wide-baseline views.

Shape reconstruction: Errors in shape reconstruction occur with existing volumetric and stereo-based methods due to the limited number of views and errors in correspondence.

These inaccuracies cause blurring and ghosting artifacts in rendered images.

This paper presents a novel method for 3D video rendering, which enables us to reduce such artifacts by dynamically deforming multiple textures depending on a virtual viewpoint.We also propose a mesh optimisation method with respect to texture consistency as well as shape preservation, which cooperates with the texture deformation.

For the sake of reviewing 3D video, we summarise the flow of its generation here. We first calibrate a set of cameras and capture a synchronised multiple-view image sequence of an object's performance. We reconstruct a 3D mesh from the captured images using a graph-cut based method [12] without its super-resolution process. We generate textures from the captured images, and finally, we can render free viewpoint images with the generated 3D mesh and textures. The target processes of harmonised texture mapping include mesh reconstruction, texture generation and rendering.

2 Related Work

Texture generation and visualisation of 3D video is a problem of generating a single arbitrary view image from multiple view captured images.

A simple method generates texture for a certain mesh face by choosing the 'best camera' that captures the most frontal view. However, this can cause discontinuities between contiguous faces due to inaccuracies in shape. Starck and Hilton utilise a method for generating a single continuous texture using a spherically mapped 3D shape and multiresolution texture blending, which ensures that details of the textures can be preserved [10]. However, this approach is limited to surfaces with genus-0 spherical topology.

View-dependent rendering has been used to achieve high-quality rendering of changes in appearance with viewpoint by dynamically generating textures based on the virtual viewpoint. This requires a much larger data size than a single texture but ensures that the exact appearance of a captured object is rendered from the original camera viewpoints. Several view-dependent methods have been proposed including vertex-based [8] and face-based methods [11].

Most existing methods, however, do not consider the inaccuracy of camera parameters and estimated 3D shape, and therefore textures extracted for a certain face from multiple view images are often not consistent with each other. This inconsistency causes blurring and ghosting artifacts in the rendered image.

Cobzas *et. al.*proposed an image-based rendering method for generating arbitrary poses of an object from captured images using coarse geometry (2D) and a uncalibrated camera [2]. They generate a spatial texture basis from the captured images, and synthesise a new texture for a new pose from the basis. They then warp the texture back onto the geometry, and finally render naturally synthesised images. Eisemann *et. al.*recently proposed floating textures [3], which reduces such blurring and ghosting artifacts by dynamic image warp of textures depending on a virtual viewpoint so that

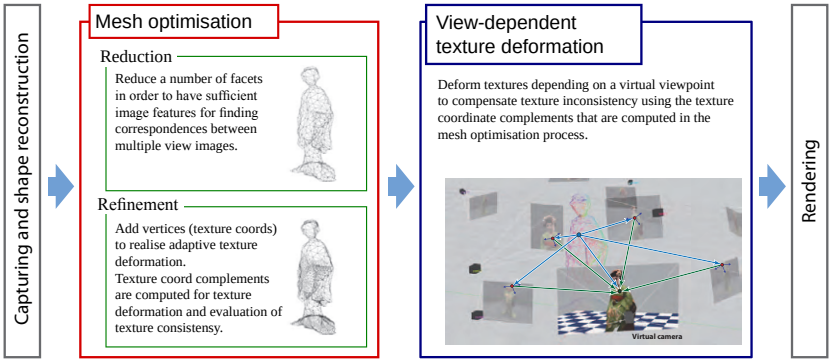


Figure 1: Process Flow of Harmonised texture mapping.

the textures are consistent with each other, instead of trying to estimate more accurate camera parameters and 3D shape. They presume that the camera parameters and the 3D shape are almost accurate but not perfect, and thus a small deformation is enough to compensate for the texture inconsistency.

The main concept of the proposed method is similar to the floating textures. Our method is, however, based on a 3D mesh optimisation for the texture deformation that enables us to deform textures more effectively. Here we denote the advantages of utilising our 3D mesh based method over floating textures as:

Adaptive texture deformation: Our method can apply texture deformation to regions of an object adaptively by optimising a 3D mesh with respect to shape and texture consistency, that is, it can control where and how much the textures should be deformed.

Valid texture deformation: Our method can validate whether texture deformation of a certain region is appropriate or not by evaluating texture consistency and controlling the range of texture deformation through the mesh optimisation process.

Real-time rendering: Our method can render images in real time ($\geq$ 30 fps) with 15 viewpoint images, since it applies 3D mesh optimisation in an offline process.

3 Harmonised Texture Mapping

The key ideas of harmonised texture mapping are view-dependent deformation of multiple-view textures and 3D mesh optimisation which cooperates with the deformation. This enables us to produce non-blurred and high quality images using inaccurate camera parameters and errors in 3D shape estimation. That is, we do not try to estimate precise camera parameters and 3D shape, but appropriately deform the textures and shape for the novel view generation. Before describing the details of the proposed method, we summarise terms that are used in the following sections.

- Camera, $C_i, \{i = 1, 2, \cdots, n_c\}, n_c$ = number of cameras: a camera placed in a capturing studio. An image captured by C_i is denoted by I_i.
- Mesh, M: a reconstructed mesh of an object which comprises vertices, $\boldsymbol{v}_v \in \mathbb{R}^3, \{v = 1, 2, \cdots, n_v\}, n_v$ = number of vertices, and faces, f_j. We only use a mesh with triangular faces in this paper.
- Projected mesh, $\hat{M}_i, \{i = 1, 2, \cdots, n_c\}$: a mesh projected to C_i, which comprises vertices of M, vertices of M projected to C_i (*projected vertices*; $\hat{\boldsymbol{v}}_v^{\hat{M}_i} \in \mathbb{R}^2$), and visibility of the vertices from C_i.
- Image segment (face), $\mathcal{I}^f_{\hat{M}_i \leftarrow \hat{M}_j}$: an image of face, f, which is extracted from I_j and transformed onto I_i with M and camera parameters of C_i and C_j.
- Image segment (vertex), $\mathcal{I}^v_{\hat{M}_i \leftarrow \hat{M}_j}$: a set of image segments (face) of all faces that share vertex, v, which are extracted from I_j and transformed onto I_i.
- Image matching: We evaluate consistency of images with (1). We call a matched point a point which has a minimum value of (1) between two images, I_i and I_j.

$$\|I_i - I_j\| = \frac{1}{3n_p} \sum_y \sum_x \sum_{c \in \{R,G,B\}} \|I_i^{x,y,c} - I_j^{x,y,c}\|, \tag{1}$$

  where $I^{x,y,c}$ denotes intensity of colour band, c, at point (x, y) on image, I. n_p denotes the number of pixels between compared images.

As shown in Fig. 1, the flow of the harmonised texture mapping is divided into two phases, i.e., mesh optimisation (offline) and view-dependent texture deformation (online).

In the offline phase, we optimise a 3D mesh generated from multiple view images so that the texture deformation in the succeeding phase can be applied effectively. The mesh optimisation is applied in a coarse-to-fine strategy, that is, we first reduce the number of faces of a mesh so that the size of faces is large enough to have sufficient feature points for matching, and then refine the mesh by adding vertices (i.e., texture coordinates) as control points for the texture deformation. In other words, our mesh optimisation is a method to restructure the 3D mesh with respect to texture consistency as well as shape preservation. Through the refinement process, we compute *texture coordinate complements (TCC)* for all vertices of the mesh and for each viewpoint of the cameras. TCC are a set of corresponding points of image features around a projected vertex between multiple-view images. By adjusting texture coordinates with the TCC appropriately, we can deform the textures to be consistent with each other. Note that we do not try to estimate a more accurate 3D shape in the refinement process because: 1) we already have optimised the shape to a certain level by a graph-cut based stereo refinement method, and 2) it is hard to estimate it more accurately with inaccuracies in camera parameters.

In the online phase, we dynamically deform multiple textures with the optimised mesh and the TCC depending on a virtual viewpoint. Once the multiple deformed textures are generated, we blend them with weighting factors depending on the virtual viewpoint, and finally have a rendered image with few blurring and ghosting artifacts.

In summary, by using the above methods, we can implicitly apply a non-linear texture deformation dynamically to the original 3D mesh, and this enables us to generate non-blurred and high quality free view point images. We describe the detailed algorithms in the following sections.

3.1 Mesh Optimisation

As described in the previous section, the mesh optimisation has two processes, i.e., reduction and refinement. In the mesh reduction part, we first reduce the number of faces of a mesh by edge collapse with respect to shape preservation [4]. After that, we collapse edges of faces whose textures generated from multiple-view images are *inconsistent* with each other. An ordinary method of mesh reduction generally collapses redundant edges and does not apply to those inconsistent regions, because this operation increases differences between the original and the reduced meshes. Our approach, on the other hand, eliminates such inconsistent regions in order to represent them by larger faces. Although it may seem to be contradictory, this is the point of our mesh reduction.

In the region of large texture inconsistency, the shape is not accurately recovered mainly because of concavities. This is because concave regions are not recovered by a silhouette intersection method and stereo-based methods may fail due to the wide-baseline between camera views. Furthermore, inaccurate camera parameters will result in texture inconsistencies as there may be no surface that correctly aligns all views. In order to generate consistent textures for such regions, we need to deform textures more dynamically than in the consistent regions. For more dynamic texture deformation, we need larger faces, and therefore, we eliminate such inconsistent regions and represent them by larger faces.

As a consequence, our mesh reduction method enables us to adaptively reduce a mesh with respect to both shape preservation and texture consistency.

After the mesh reduction, we refine the mesh by adding vertices, i.e., texture coordinates where textures are inconsistent. Through this refinement process, we compute texture coordinate complements (TCC). This is also a key data for the consistent texture deformation in the succeeding phase.

3.1.1 Mesh Reduction

Pseudo code of the mesh reduction is shown in Alg. 1. In the first step, we reduce the number of faces in order to have faces large enough to have sufficient image features. We then make the size of texture-inconsistent faces larger in order to deform the textures more dynamically (See the second step of Alg. 1).

3.1.2 Mesh Refinement and Texture Coordinate Complements

After the mesh reduction, we refine the mesh by adding texture coordinates and texture coordinate complements (TCC). Alg. 2 shows the procedure of the mesh refinement and TCC computation.

In the first step, we compute ‘image segment (vertex)’ matching for obtaining TCC. Fig. 2 illustrates the projection of ’image segment (vertex)’ matching and TCC for a single vertex of a mesh. The vertex is projected to image planes of cameras from the object surface using the camera parameters shown as dashed lines. Each projected vertex has its own ‘image segment (vertex)’ on the image plane, as shown by small rectangles on the image planes. We then find matched points between an image of a

Algorithm 1: Mesh reduction.

```
repeat
    Mesh reduction by edge collapse with respect to shape preservation.
until average size of faces > threshold.
repeat
    foreach face f in M do
        D_f ← Compute difference of f.
    f_⋆ ← Find a face that has the maximum D and its size < threshold.
    if f_⋆ is empty then
        continue ← false
    else
        Find one edge of f which preserves the shape most when it is removed, and collapse it.
until continue is false.
```

certain projected vertex and image segments (see Alg. 3). If the shape and the camera parameters are perfectly accurate, the matched points are the same position as the projected vertex. However, the matched points are located around the projected vertex due to the inaccuracy of the shape and camera parameters, and the differences between these points is the reason of the blurring and ghosting artifacts. Hence correcting these differences is the key object of harmonised texture mapping, and the matched points are recorded as the texture coordinate complements.

After obtaining TCC, we compute differences of textures (Alg. 4[1], Fig. 3), which are generated using corrected texture coordinates with TCC (Alg. 5). We then evaluate the TCC using the difference (Alg. 6). A small value of the difference denotes that the computed TCC is good enough to compensate the inconsistency. Most object regions have small values because the shape and camera parameters are almost accurate. We can reduce the artifacts by simply using TCC in most cases. The problem is, therefore, regions with large values. Most of the regions are concave regions of the true shape, because these regions are hard to reconstruct by silhouette and wide-baseline stereo methods. Although a stereo based refinement is applied, it is still difficult because the camera parameters are not perfectly accurate and there is a wide-baseline.

In such region with large differences, we subdivide faces by adding vertices (i.e., texture coordinates) and repeat Alg. 3. This means that our refinement method enables us to apply non-linear texture deformation to such regions adaptively.

The faces are subdivided into four triangles by adding vertices to the centre of each edge. To preserve the topology of the mesh, we first find the faces that are required to be subdivided, then assign an ID to all the faces based on the adjacent faces as shown in Fig. 4, and finally subdivide them all at the same time (see Alg. 7).

3.2 View-dependent Texture Deformation and Rendering

With the optimised mesh and TCC, we dynamically deform textures depending on a virtual viewpoint. Fig. 5 shows a sketch of generating harmonised textures with TCC. When we specify a viewing direction to the object, we can compute weighting factors and then compute harmonised texture coordinates using TCC (Fig. 6). These har-

[1] Values of α and γ_1 are heuristically defined as 0.3 and 10, respectively.

Algorithm 2: Mesh refinement and TCC computation.

repeat
  TCC ← Computing image segment (vertex) matching (see Alg. 3).
  Computing differences of extracted textures with TCC (see Alg. 4).
  continue ← Evaluation and subdivision (see Alg. 6).
until *continue is false.*

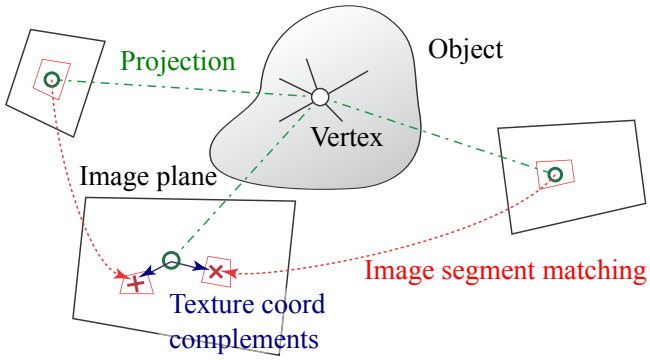


Figure 2: Sketch of projection, image segment matching and texture coordinate complements.

monised texture coordinates enable us to generate textures from each captured image by deforming them to be consistent with each other (see Alg. 8).

In the algorithm, the weighting factor, w, is computed using a dot product between the camera's viewing direction and the virtual one, i.e., a larger value means they are closer. This controls how much of a texture generated from a camera is deformed. A larger value of the weighting factor forces deformation of textures with smaller weights. That is, a texture generated from a camera with a larger value is not deformed much and preserves the captured image. This enables us to generate a texture without deformation from the camera whose viewing direction is the same as the virtual one[2].

After generating harmonised texture coordinates, we generate deformed textures with them. We then blend them with the weighting factors that are computed in Alg. 8, and finally we can render an image with few blurring and ghosting artifacts (Alg. 9).

4 Experimental Results

We evaluate our method, Harmonised texture mapping, with two 3D videos of *Maiko* and *Kung fu* data. The configuration of capturing and the details of the data are the following.

[2]We utilise a coefficient, γ_2, for controlling the weighting factor, which is heuristically defined as $\gamma_2 = 50$. With this value, the normalised weighting factor (in Eq. (6)) of the camera that has the same viewing direction as the virtual one becomes almost 1 and the others 0.

Algorithm 3: Computing image segment (vertex) matching.

$\hat{\boldsymbol{M}} = \left\{\hat{M}_i | i = 1, 2, \ldots, n\right\} \leftarrow$ Compute projected meshes with M and a set of C.
foreach *projected mesh* $\hat{M}_i$ *in* $\hat{\boldsymbol{M}}$ **do**
  foreach *vertex* $\boldsymbol{v}$ *in* M **do**
    foreach *projected mesh* $\hat{M}_j$ *in* $\hat{\boldsymbol{M}}$ **do**
      $\hat{\boldsymbol{v}}_{\boldsymbol{v}}^{\hat{M}_i \leftarrow \hat{M}_j} \leftarrow$ Search corresponding point with template matching by projecting image segment $\mathcal{I}_{\hat{M}_i \leftarrow \hat{M}_j}^{\boldsymbol{v}}$ to I_i.

Algorithm 4: Computing differences of textures of each face f.

foreach *face* f *in* M **do**
  # We define a best camera as a camera that captures face f from its front and larger.
  Find best camera $C_\star$ that has the highest value of e defined by

$$e_k = \alpha \bar{d}_{C_k} + (1 - \alpha)\bar{s}_f, \tag{2}$$

  where

$$\bar{d}_{C_k} = \frac{d_{C_k}}{\sum_i^C d_{C_i}},\ d_{C_k} = \left(\frac{\vec{V}_{C_k} \cdot \vec{N}_f + 1}{2}\right)^{\gamma_1},\ \bar{s}_f = \frac{s_f}{\sum_i^C s_{f_i}}, \tag{3}$$

  - $\vec{V}_{C_k}$: viewing vector of camera C_k,
  - $\vec{N}_f$: surface normal of face f,
  - s_f: size of face f, and
  - α, γ_1: weighting coefficient.

  Compute differences by

$$D_f = \frac{1}{n_C - 1} \sum_{k, k \neq \star} e_k \| \mathcal{I}_{\hat{M}_k \leftarrow \hat{M}_\star}^{f'} - I_k \|, \tag{4}$$

  where

  - n_C: number of cameras in which the face is projected, and,
  - $\| \mathcal{I}_{\hat{M}_k \leftarrow \hat{M}_\star}^{f'} - I_k \|$: difference computed by Eq. (1).

- Studio: a dodecagonal prism whose diameter and height are 4 m and 2.5 m, respectively (Fig. 7).
- Camera: 15 cameras (Sony XCD-710CR; XGA, 25 fps). The cameras have the same lenses except three cameras, i.e., frontal and ceiling cameras. An example of captured images are shown in Fig. 8.
- Maiko data: a maiko woman wearing *kimono*. Both of her sides and a part of the broad sash are concave but not perfectly reconstructed. The sash on her back is reconstructed thicker than the actual size, and thus the patterns on it extracted from multiple cameras are not consistent with each other.
- Kung fu data: a man performing kung fu. The cloths are simpler than maiko.

4.1 Mesh Optimisation

Fig. 9 shows meshes that are generated through the optimisation process. Fig. 9a shows the original mesh that has 107,971 vertices and 215,946 faces. Fig. 9b shows the

Algorithm 5: Obtaining warped image segment of face.

Suppose the vertices of f as $\{v_1, v_2, v_3\}$, adjusted texture coordinates of f is given by $\hat{\boldsymbol{v}}^{\hat{M}_i}_{v'_k} = \hat{\boldsymbol{v}}^{\hat{M}_i \leftarrow \hat{M}_j}_{v_k}, k = 1, 2, 3.$

With these texture coordinates, we obtain warped image segment of face, $\mathcal{I}^{f'}_{\hat{M}_i \leftarrow \hat{M}_j}$.

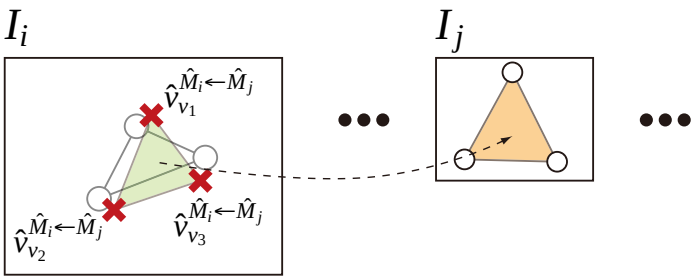


Figure 3: Projection of image segment (face) for computing difference.

reduced mesh that have large enough faces to have sufficient features for matching. Fig. 9e shows the differences computed by Alg. 4 with the colour of high value: red and low value: blue. We can find that large differences appear on the side of the body, sash, sleeves and hem of the *kimono*. The inconsistency of the side and sash is caused by concavities, and the others are mainly due to the self-occlusions seen from the ceiling cameras. The latter case is not easy to compensate by texture deformation because the textures are completely different. For example, a part of the hem (a lower part of the mesh, which shows a larger difference than the other area) has a texture of a part of the head projected from the ceiling camera. This causes an artifact of texture switching when the virtual camera moves between the ceiling cameras the others. Fig. 9c shows a reduced mesh with regard to texture consistency, and Fig. 9f shows that the inconsistent faces become larger by the reduction. Fig. 9d and g shows the final result of the mesh optimisation. Through this process, the inconsistent areas first become large to be deformed dynamically and then they shrink to limited areas. This result shows that the input mesh is adaptively optimised with regard to the texture consistency and shape preservation.

4.2 Quantitative Evaluation

We place a virtual camera that has the same parameters of camera, C_i, and render an image, I'_i, without using the image of camera, C_i. We then evaluate the quality of the harmonised texture mapping by computing PSNR between I_i and I'_i. For comparison, we also show PSNR of existing methods of view-dependent vertex based method (VDVBM) and view-dependent texture blending method (VDTBM)[3] in Fig. 10. We utilised all cameras in the studio except the frontal and ceiling cameras for this evaluation, i.e.,

[3]The textures are blended depending on the virtual viewpoint, but not deformed.

Algorithm 6: Evaluation and Subdivision.

# Evaluate differences $\boldsymbol{D} = \left\{D_f | f = 1, 2, \ldots, m\right\}$.
if `average`($\boldsymbol{D}$) > *threshold* or *Requires subdivision for self-occluded patches* **then**
    Subdivide faces of M (See Alg. 7).
    return true.
else
    return false.

Algorithm 7: Subdivision of faces.

foreach *face* f *in* M **do**
    if $E_f >$ *threshold* **then**
        $typeOfFace_f \leftarrow 4$

foreach *face* f *in* M **do**
    if $typeOfFace_f \neq 4$ **then**
        $typeOfFace_f \leftarrow 0$
        foreach *adjacent face af of* f **do**
            if $typeOfFace_{af} = 4$ **then**
                $typeOfFace_f \leftarrow typeOfFace_f + 1$

foreach *face* f *in* M **do**
    if $typeOfFace_f > 0$ **then**
        Subdivide face f as shown in Fig. 4 by $typeOfFace_f$.

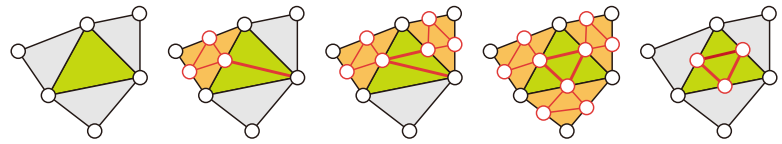
12 cameras from the top-left in Fig. 8. Our method shows higher values than the other methods as shown in the figure.

4.3 Qualitative Evaluation

We show captured images and images that are rendered with VDVBM, VDTBM and our method (HTM) in Fig. 11 for the qualitative evaluation. We can easily find difference of quality between VDVBM and HTM from the figures. The images generated by VDVBM contain blurring artifacts due to both of interpolation of vertex colors and the inaccuracy of the camera parameters and the 3D shape. While VDTBM generates textures with almost the same quality to the captured images, blurring and ghosting artifacts appear on the broad sash and the sleeves (Fig. 11e). The artifacts also appear on the designs of the sash (Fig. 11f). On the other hand, our method can obviously reduces such artifacts and render high quality images as shown in Fig. 11g and Fig. 11h.

4.4 Real-time Rendering

Lastly, we show a result of computational time for rendering one frame in Tbl. 1. The number of faces of optimised Maiko and Kung fu data are approximately 5,000. The specification of the PC that we utilised for the evaluation is CPU: Core2Duo 2.4 GHz, Memory: 4 GB, GPU: GeForce 8800 GTX and VRAM: 750 MB, and the software



(a) Type 0 (b) Type 1 (c) Type 2 (d) Type 3 (e) Type 4

Figure 4: Types of face subdivisions.

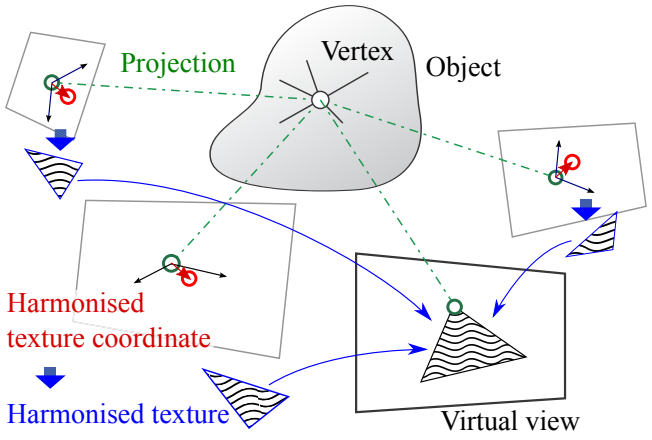


Figure 5: Sketch of generating harmonised textures with texture coordinate complements.

is built with C# and Managed DirectX with Pixel shader 2.0. Fig. 12 shows rendered images of the kung fu sequence. Our method can render these high quality images in real time ($\geq$ 30 fps) using 15 multiple viewpoint images.

5 Conclusion

We have presented a novel rendering method for 3D video, Harmonised texture mapping, which can render with few blurring and ghosting artifacts that are caused by the inaccuracy of the camera parameters and the reconstructed 3D shape. Our method optimises the 3D mesh of an object with regard to texture consistency and shape preservation, and compensates for the inconsistency of the textures by dynamically deforming multiple-viewpoint textures. The experimental results have shown that our method can reduce the artifacts effectively, and render the optimised 3D video in real time.

Our method, however, failed to compute harmonised texture coordinates for regions which are grossly different to the true surface and whose textures are completely inconsistent. To cope with this issue, we need a framework to reject such completely different textures.

Algorithm 8: Computing harmonized texture coordinates with TCC.

Set viewing direction, $\vec{V}_{eye}$.

foreach *projected mesh* $\hat{M}_i$ *in* $\hat{\boldsymbol{M}}$ **do**

Compute weighting factor by

$$w_{\hat{M}_i} = \left(\frac{\vec{V}_{eye} \cdot \vec{V}_{C_i} + 1}{2} \right)^{\gamma_2}, \tag{5}$$

where γ_2 denotes a weighting coefficient.

foreach *vertex* v *in* M **do**

Compute point for extracting texture, i.e., harmonized texture coordinates, $\bar{\boldsymbol{v}}_v^{\hat{M}_i}$, with the weighting factors and the TCC by

$$\bar{\boldsymbol{v}}_v^{\hat{M}_i} = \sum_j \frac{w_{\hat{M}_j}}{\sum_k w_{\hat{M}_k}} \hat{\boldsymbol{v}}_v^{\hat{M}_i \leftarrow \hat{M}_j}. \tag{6}$$

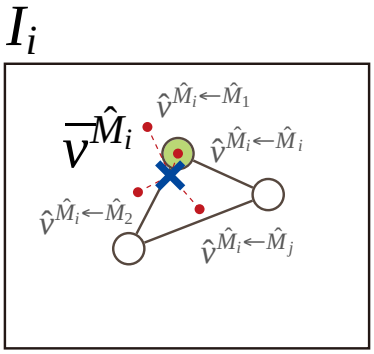


Figure 6: Computing harmonised texture coordinates.

References

[1] J. Allard, J.-S. Franco, C. Ménier, and E. Boyer. The GrImage Platform: A Mixed Reality Environment for Interactions. In *4th IEEE International Conference on Computer Vision Systems*, 2006.

[2] D. Cobzas, K. Yerex, and M. Jägersand. Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. In *Eurographics*, volume 21, pages 1067–7055, 2002.

[3] M. Eisemann, B. D. Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Eurographics '08*, 27(2), 2008.

[4] H. Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996.

[5] i3DPost. Intelligent 3D content extraction and manipulation for film and games. http://www.i3dpost.eu/.

[6] R. Ichikari, R. Tenmoku, F. Shibata, T. Ohshima, and H. Tamura. Mixed reality pre-visualization for filmmaking: On-set camera-work authoring and action rehearsal. *International Journal of Virtual Reality*, 7(4):25–32, December 2008.

[7] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia, Immersive Telepresence*, 4(1):34–47, 1997.

[8] T. Matsuyama, X. jun Wu, T. Takai, and S. Nobuhara. Real-Time 3D Shape Reconstruction, Dynamic 3D Mesh Deformation, and High Fidelity Visualization for 3D Video. *Computer Vision and Image Understanding*, 96(3):393–434, 2004.

Algorithm 9: Output image rendering.

foreach *pixel p in a rendered image* **do**
  foreach *projected mesh* $\hat{M}_i$ *in* $\hat{\boldsymbol{M}}$ **do**
    if $T^p_{\hat{M}_i}.Alpha = 0$ **then**
      $\bar{w}_{\hat{M}_i} \leftarrow 0$
    else
      $\bar{w}_{\hat{M}_i} \leftarrow w_{\hat{M}_i}$
  Compute colour by

$$\mathrm{Col}^p = \sum_i \frac{\bar{w}_{\hat{M}_i}}{\sum_j \bar{w}_{\hat{M}_j}} T^p_{\hat{M}_i}, \tag{7}$$

  where $T^p_{\hat{M}_i}$ denotes a colour value for pixel p extracted from I_i with $\bar{\boldsymbol{v}}_v^{\hat{M}_i}$.

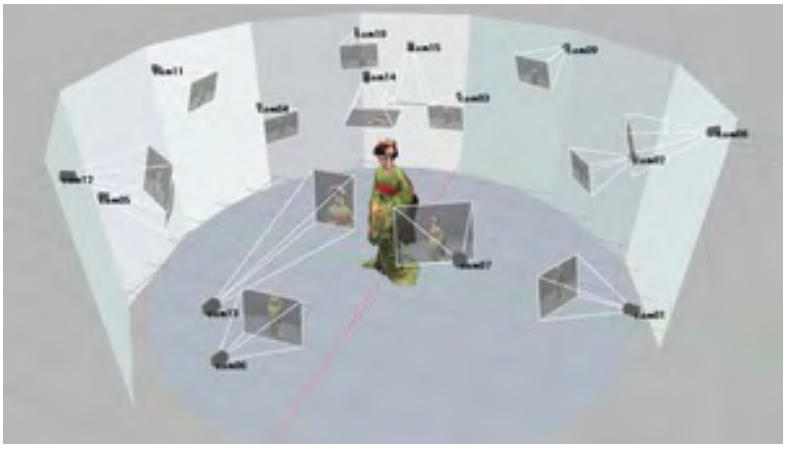


Figure 7: Studio and cameras (CG model).

[9] S. Moezzi, L.-C. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia, Immersive Telepresence*, 4(1):18–26, 1997.

[10] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, pages 21–31, 2007.

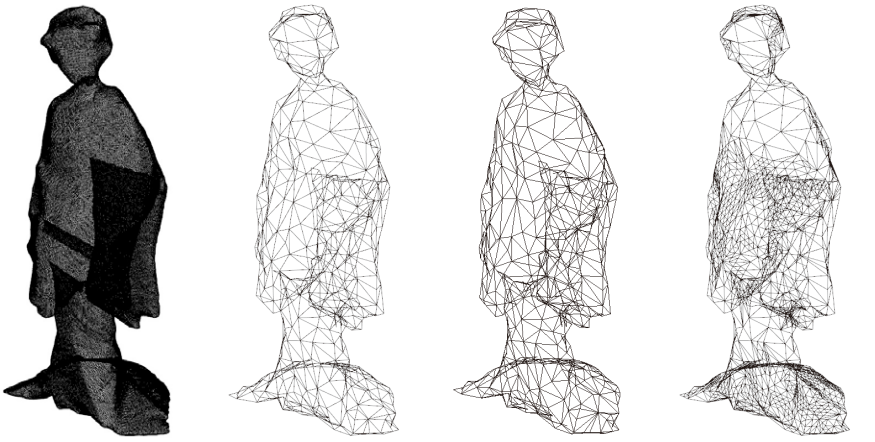
[11] T. Tomiyama, M. Katayama, Y. Orihara, and Y. Iwadate. Arbitrary viewpoint images for performances of japanese traditional art. In *2nd Conference on Visual Media Production*, pages 70–77, 2005.

[12] T. Tung, S. Nobuhara, and T. Matsuyama. Simultaneous super-resolution and 3d video using graph-cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2008.

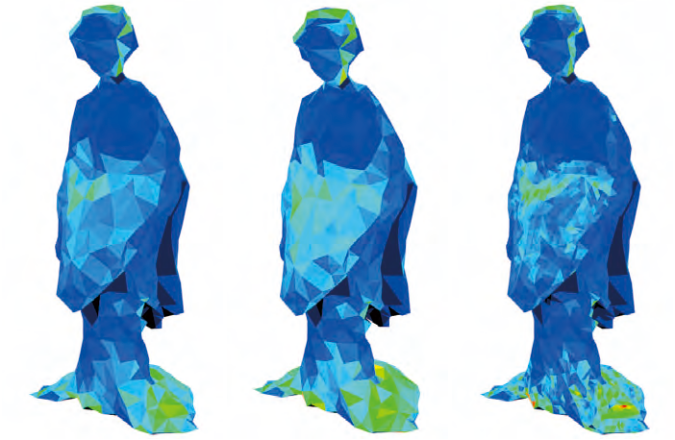
Figure 8: Example of captured images.

Table 1: Computational time for rendering one frame (ms).

	Maiko	Kung fu
Weighting factors (Eq. (5))	< 1	< 1
Harmonised texture coordinates (Eq. (6))	27.2	15.8
Texture generation and rasterisation (Alg. 9)	< 1	< 1



(a) Original 215,946 faces (b) Reduced-S 1,500 faces (c) Reduced-T 1,486 faces (d) Refined 4,406 faces



(e) Reduced-S (f) Reduced-T (g) Refined

Figure 9: Mesh optimisation results.

Images a–d show meshes generated by the optimisation process. Images e–g show the texture inconsistency of each mesh, where the colour of high value (inconsistent): red and low value (consistent): blue. 'Reduced-S' and 'Reduced-T' denote reduced meshes with respect to shape preservation and texture consistency, respectively.

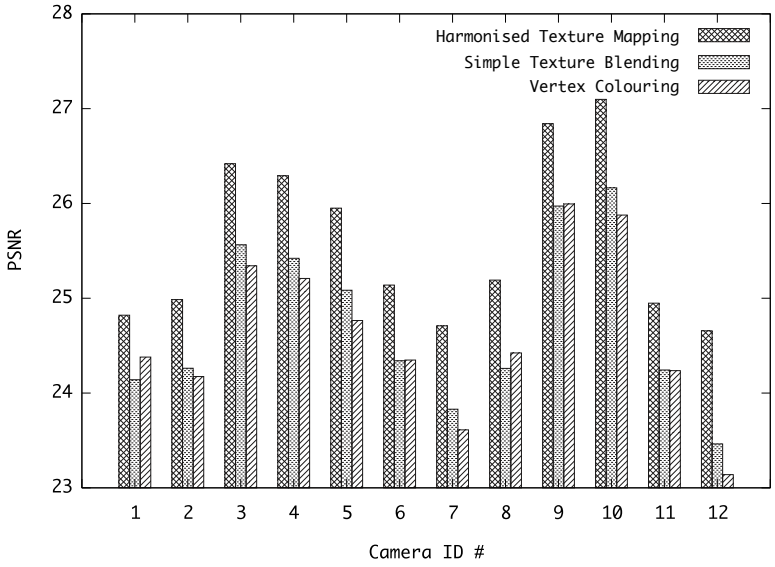


Figure 10: Quantitative evaluation of various methods.

(a) Captured image (front). (b) Captured image (back). (c) VDVBM (front) (d) VDVBM (back)

(e) VDTBM (front) (f) VDTBM (back) (g) HTM (front) (h) HTM (back)

Figure 11: Images for qualitative evaluation.

Figure 12: Examples of 3D video sequence.