Action History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes

Qun Shi[1] Shohei Nobuhara[2] Takashi Matsuyama[2]

Abstract: This paper presents a novel spatiotemporal action editing framework that synthesizes spatiotemporally synchronized multi-party interaction 3D video scenes from separately captured data. Our main idea is to model multi-party interaction events with synchronization among body actions and gaze actions from multiple objects, and use them as constraints to perform spatiotemporal editing on 3D video data. To achieve that, we first propose the idea of Action History Volume to enable to model multi-party interaction events into spatiotemporal constraints. Next, we propose a novel 3D gaze estimation method, taking advantage of the symmetry prior and the super-resolution technique. Finally, we propose a three-step action editing processing scheme, including data segmentation, intra-key segment editing and inter-key segment optimization. Experiment results prove the effectiveness of the proposed method.

1. Introduction

Capturing human behaviors has been a crucial issue in computer vision and graphics research fields over the past decade. Technology has evolved from sparse marker based motion capture systems to dense reconstruction of non-rigid surfaces as 3D mesh sequences, called *3D video* [1][2]. The advanced 3D video technique [3][4][5] enables to generate the full 3D dynamic shape and texture data from multiple view video, and allows replaying of the captured performances for free viewpoint browsing.

In 3D video capturing, mutual occlusions inevitably occur when multiple objects are captured simultaneously, causing phantom volumes and invisible surfaces without textures. As a result, in the acquisition and reconstruction of pre-designed multi-party interaction scenes in 3D video, multiple objects are usually required to be captured separately and then, synthesized together afterwards. In the separately captured data, objects perform motions that are designed to match with each other.

Since the separate capture scheme will inevitably result in spatial and temporal mismatches of the interaction event, the synthesis of multi-party interaction scenes from them becomes a non-trivial task and requires large amount of editing work to perform spatiotemporal alignment of the unsynchronized data sequences. Besides, while 3D video has the advantage in capturing the realistic detailed non-rigid surface shape dynamics of the body, clothing or even hairs, its acquisition results in an unstructured volumetric or mesh approximation of the surface shape at each frame without temporal correspondence. Although some research work [6][7][8][9][10] tried to estimate a smooth transition among these unstructured data, estimating accurate dense correspondence of dynamic surfaces remains an open problem. In addition, since the conventional skeleton model based motion editing methods are not directly applicable for this kind of data, the editing work [11][12][13] of 3D video becomes more challenging than that of conventional CG data or motion capture data.

This paper propose a novel method to synthesize spatiotemporally synchronized multi-party interaction scenes in 3D video from separately captured data, while preserving the original motion dynamics of each object as much as possible. Here we specify our targeting interactions as interactive events composed of spatiotemporally synchronized body motions. The technical problems we address for this research are (1) how to model multi-party interaction events into spatiotemporal constraints, and (2) how to perform spatiotemporal alignments on separately captured data given as sequences of 3D meshes. Our basic strategy is to first, divide the original captured motion sequences into key-segments and transitional segments based on whether multiple objects are contacting with each other or not. Then by introducing Action History Volume (AHV) we model the spatiotemporal correlations of multiple objects and compute the proper relative positions of them for each pair of key-segments. Finally, a global optimization will be performed to combine the whole interaction sequence together while keeping it smooth and natural looking. Note that the proposed method does not require explicit surface correspondence across time, and does not rely on any kind of skeleton models inside meshes.

The main contribution of this research consists of (1) we propose a concept of AHV, as well as an AHV-based Interaction Dictionary that can model all types of multi-party interaction events; (2) we propose a novel 3D gaze estimation method, which performs non-constraint and non-contact gaze sensing on freely moving 3D video object; (3) we have designed an executable

[1] Nara Institute of Science and Technology
[2] Kyoto University

computational scheme for performing the real multi-party interaction 3D video editing work. Possible applications include the making and editing of 3D movies, computer animations and video games. Note that earlier versions of this research have been presented in [14] [15] [16].

2. Action History Volume for Multi-party Interaction Modeling [16]

This section presents the idea of Action History Volume (AHV) and introduces the multi-party interaction modeling method using AHV. The purpose of this method is to model multi-party interaction events into different types of spatiotemporal synchronization between multiple objects' actions, creating constraints that can be used in future computation process of the action editing work. The following contents include (1) the definition of Action History Volume (AHV), (2) the description of how AHV can be used to represent both body actions and gaze actions, and (3) the AHV-based Interaction Dictionary, which could be used to model multi-party interaction events.

2.1 Action History Volume

2.1.1 Definition of AHV

The proposed Action History Volume is inspired by the idea of Motion History Volume(MHV) , which is invented by Weinland *et al.* [17] for solving free viewpoint action recognition task. The MHV based action representation has been proved to work effectively for action recognition. However, the original definition of MHV only considers the occurrence of the motion. To realize the goal of performing spatial and temporal alignments on multiple separately captured action sequences, full temporal information including both the starting and ending moments of the action is necessary for representing the spatiotemporal synchronization between multiple objects' actions. Therefore we propose the Action History Volume, which extends the original MHV by adding in more temporal information of the action as Eq. (1):

$$v_\tau(x,y,z,t) = \begin{cases} (t_{\text{start}_1}, t_{\text{end}_1})...(t_{\text{start}_n}, t_{\text{end}_n}) & \text{if } \bigcup_{s=0}^{\tau-1} D(x,y,z,s), \\ \text{empty} & \text{otherwise,} \end{cases} \tag{1}$$

where t_{start_i} and t_{end_i} $(i \in (1,...,n))$ form up one pair of the starting and ending times of actions at voxel (x,y,z) within time τ, and n denotes the total number of pair $(t_{\text{start}}, t_{\text{end}})$ at voxel (x,y,z) within time τ. $D(x,y,z,s)$ is a binary-valued occupancy function judging whether voxel (x,y,z) is occupied by the object at moment s. This function is estimated using multi-view silhouettes and thus, corresponds to the visual hull, which is easy to compute and yield robust 3D representations. We denote the centroid of AHV $v_\tau(x,y,z,t)$'s spatial volume as $C(x,y,z)$, which is considered as the root node of an AHV.

It should be noted that AHV is a voxel-wise representation. That is, within a period of time τ, if the object's actions occupy voxel (x,y,z) for at least one moment, we keep record of the starting and ending times of the action on that voxel. Then the object's AHV of this time period is a collection of all those time recorded voxels. Besides, the associate 3D Action Energy Volume(AEV) can also be easily computed by thresholding $h \neq Null$. This AEV represents the spatial volume of the corresponding AHV. We denote the centroid of an AEV $E_\tau(x,y,z,t)$'s spatial volume as $C(x,y,z)$, which is considered as the root node of an AHV.

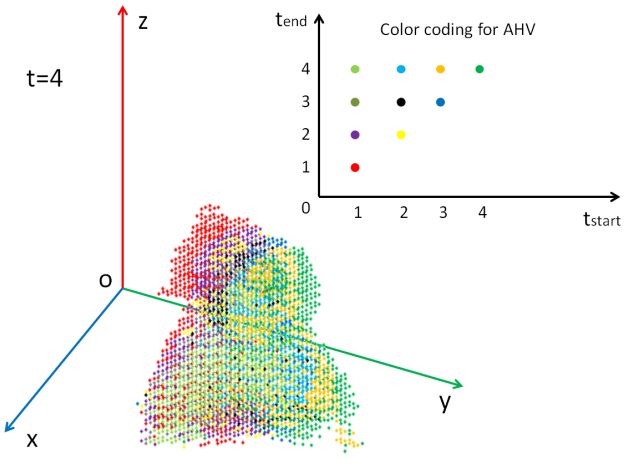
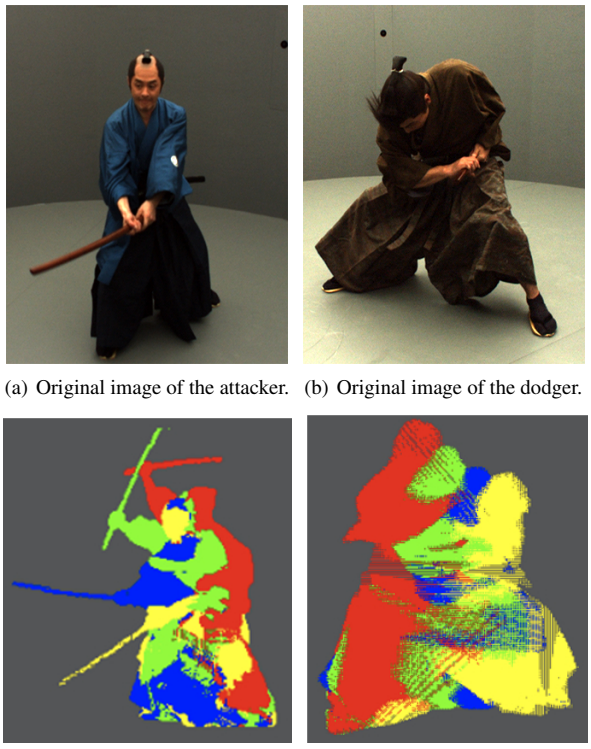


Fig. 1 Illustration of Action History Volume.



(a) Original image of the attacker. (b) Original image of the dodger.

(c) AHV of the attacker. (d) AHV of the dodger.

Fig. 2 Examples of AHV action representation.

Fig. 1 visualizes the AHV of an object swinging his body to his left side. Here each voxel of the AHV is color coded based on the starting and ending times of the action on it.

2.1.2 Body Action Modeling using AHV

The proposed Action History Volume can be used to represent both the spatial and temporal aspects of objects' body actions. Fig. 2 illustrates AHVs of the attacker and the dodger in a fighting scene, respectively. Here different colors represent different ending times of the action at the voxels. Note that AHV needs not necessarily contain the action of the entire body. Instead it can be simplified by only counting in the partial volume of interest on

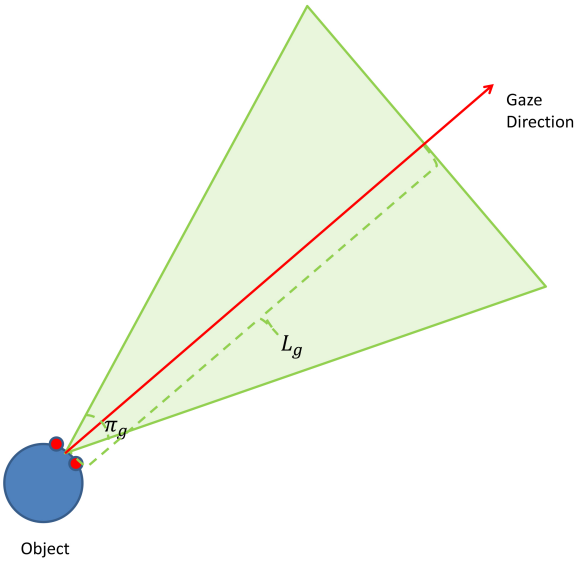


Fig. 3 Gazing cone of object.

the object's body. For example, in a sword fighting scene we may only care about the weapon of the attacker, so that an AHV of the sword would be enough for further editing work.

It should be noted that the mathematical definition of AHV allows multiple pairs of $(t_{\text{start}}, t_{\text{end}})$ to exist simultaneously on one voxel, meaning that a single AHV can model both unidirectional and multidirectional actions. However, in order to keep the simplicity of the AHV-based body action representation, we ensure the body action in each AHV to be unidirectional through the data segmentation step in the real editing process.

While in the real world human actions are usually not unidirectional, we assume that all reciprocating actions can be decomposed into unidirectional sub-actions, which can be modeled using AHVs with single pair of starting and ending times on each voxel. We ensure this requirement by taking it as one criterion in performing data segmentation, as described in 4.1.2.

2.1.3 Gaze Action Modeling using AHV

Editing interaction events among human beings requires to take objects' gaze actions into consideration. In this section we explain how AHV can be used to model the object's gaze action.

In this research work we propose to use a "gazing cone" to describe the object's gaze actions, for the following two reasons:

(1) As a matter of fact, human being's eye sight covers a wide area rather than focusing on one specific gazing point through a single ray. Thus it is more natural to use a 3D volume instead of a 3D vector to describe the object's eye sight.

(2) The main purpose of introducing the gaze actions is to create mutual visibility constraints that ensure interacting objects are inside each other's eye sight. Since evaluating the exact precise gazing point of each object is not really our concern, it is more suitable to use the object's field of vision instead of his/her specific gazing direction (gaze vector) to model his/her gaze actions in multi-party interaction event.

As shown in Fig. 3, the "gazing cone" describing the object's field of vision is a right circular cone, whose apex locates at the center of the object's two eyes, and axis equals the line on which

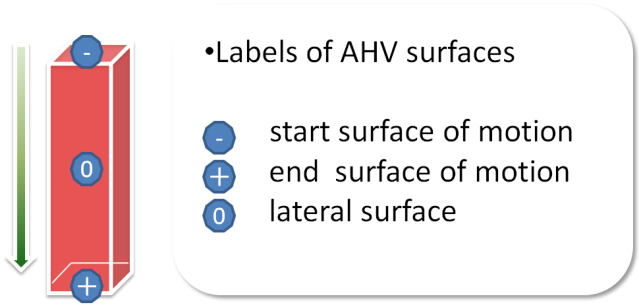


Fig. 4 Labels of AHV surfaces.

the object's gaze vector lies. Note that π_g and L_g are changeable parameters that adjust the aperture and height of the gazing cone.

For the editing work of multi-party interaction scenes, we set π_g and L_g to be 60 degree and 5 meters respectively to specify the object's communicative visual area in interaction events, that we call an "interactive zone" of the object. And normally for most of the time, objects taking part in one interaction event should be inside each other's interactive zone. Considering the gazing cone as the interest volume, then for a temporal segment the AHV of gazing cones (interactive zones) can be built to model the spatiotemporal structures of object's gaze action.

Note that in creating the object's gazing cone, his/her gaze vector and the 3D position of his/her centroid point between the two eye centers are required. We will present in the following Section how we perform 3D gaze sensing on 3D video data to acquire these information.

In addition, we will discuss later how these AHVs representing gaze actions can be used to describe the mutual visibility constraint in multi-party interaction events.

2.2 Multi-party Interaction Dictionary for Modeling Synchronization between AHVs

The AHV enables to describe the spatiotemporal structure of a single object's actions. In order to perform effective editing of multi-party interaction events, we propose an AHV-based method that models the spatiotemporal synchronization among multiple objects' actions.

2.2.1 AHV Surface Labeling and Multi-party Interaction Dictionary

In this research we propose to model multi-party interaction events by considering them as pairs of spatiotemporally synchronized actions performed by different objects. As mentioned in the former section, we assume that all reciprocating actions can be decomposed into unidirectional sub-actions, so that all kinds of interaction events can also be considered as pairs of spatiotemporally synchronized unidirectional actions. In that sense, the problem of modeling multi-party interaction events equals that of modeling all types of spatiotemporally synchronization between unidirectional actions, which can be modeled using simple AHVs with single pair of starting and ending times on each voxel.

The key idea is to descretize all possible interactions composed of unidirectional actions into a set of combinations of action directions. Up to this point, the processing unit of interaction modeling is assumed to be an interaction composed of purely unidi-

rectional simple actions that can be modeled by a pair of AHVs. Hence, we can categorize AHV surfaces into three disjoint regions where the action starts (⊕), ends (⊖), and the others (⓪), as is shown in Fig. 4. Their combinations describe all possible interactions between a pair of AHVs as listed in Table 1. We name this a multi-party interaction dictionary.

The completeness of the proposed multi-party interaction dictionary is guaranteed as follows: since multi-party interaction of reciprocating actions can be considered as a combination of sub-interactions that composed of only unidirectional actions, and the multi-party interaction dictionary enables modeling all interaction types between unidirectional actions, conclusions can be made that the proposed multi-party interaction is capable of modeling all kinds of interaction events consisting of spatiotemporally synchronized actions.

2.2.2 Mathematical Constraints for Multi-party Interaction Dictionary

In the multi-party interaction dictionary, each type of interaction contains specific spatiotemporal constraints. The mathematical descriptions of these constraints are given as follows.

First of all, for all interaction types there is a spatial constraint which states that the 3D volume of two AHVs should contact with each other:

$$V_\tau^A(x,y,z) \cap V_\tau^B(x,y,z) \neq \phi, \tag{2}$$

where $V_\tau^A(x,y,z)$ and $V_\tau^B(x,y,z)$ represent the collections of voxels in $v_\tau^A(x,y,z,t)$ and $v_\tau^B(x,y,z,t)$ respectively. Note that the two paired AHVs should have the same maximum duration τ. As well, the recorded timing should be scaled into the segment-oriented timing.

Second, each interaction type has its own temporal constraint as follows:

⊕ & ⊖:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{end}}^A = T_{\text{end}} \quad t_{\text{start}}^B = T_{\text{start}}. \end{aligned} \tag{3}$$

⓪ & ⓪: (a) If the interest volumes contact with each other at every moment within τ:

$$\bigcap_{t=0}^{\tau-1} (V_t'^A(x,y,z) \cap V_t'^B(x,y,z)) \neq \phi. \tag{4}$$

(b) If the interest volumes contact with each other at certain moments within τ:

$$\bigcup_{t=0}^{\tau-1} (V_t'^A(x,y,z) \cap V_t'^B(x,y,z)) \neq \phi. \tag{5}$$

(c) If the interest volumes have no contacts τ:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{start}}^A > t_{\text{end}}^B. \end{aligned} \tag{6}$$

⊕ & ⊕:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{end}}^A = t_{\text{end}}^B = T_{\text{end}}. \end{aligned} \tag{7}$$

⊖ & ⊖:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{start}}^A = t_{\text{start}}^B = T_{\text{start}}. \end{aligned} \tag{8}$$

⊕ & ⓪:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{end}}^A = T_{\text{end}}. \end{aligned} \tag{9}$$

⊖ & ⓪:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{start}}^A = T_{\text{start}}. \end{aligned} \tag{10}$$

Here $T_{\text{start}} = 0$ and $T_{\text{end}} = \tau$. $V_t'^A(x,y,z)$ represents a subset of $V_\tau^A(x,y,z)$ at time t. n and n' represent the set numbers of $(t_{\text{start}}, t_{\text{end}})$ from object A and B, respectively. $\forall v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B$ denotes all AHV voxels $v(x,y,z,t)$ whose 3D positions (x,y,z) have been occupied by both objects' motions for at least one moment during the time period τ.

Note that since types ⓪ & ⓪-(a), ⓪ & ⓪-(b) and ⓪ & ⓪-(c) do not contain any directional constraint, they can be easily applied onto AHVs composed of not only unidirectional actions. In that situation, the constraint equation for ⓪ & ⓪-(c) can be augmented as:

$$\begin{aligned} &\forall\ v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &\bigcup_{i=1,i'=1}^{i=n,i'=n'} ([t_{\text{start}_i}^A, t_{\text{end}_i}^A] \cap [t_{\text{start}_{i'}}^B, t_{\text{end}_{i'}}^B]) = \phi. \end{aligned} \tag{11}$$

Note that all editing-related constraints for gaze actions belong to type ⓪ & ⓪-(c), so that with the above augmented constraint equation we can directly process the changeable gaze actions without decomposing them into unidirectional sub-actions.

This multi-party interaction dictionary provides us various spatiotemporal constraints to have objective criteria for performing the editing work. For example, the objects' body actions in an attack and guard scene (Fig. 5) from sword fighting sequence can be counted as type ⊕ & ⊕. Then on the interested part of the two AHVs modeling body actions, we require $t_{\text{end}}^A = t_{\text{end}}^B = T_{\text{end}}$, meaning that the two swords contact and only contact at the end of the whole action duration.

The multi-party interaction dictionary proposed in this Section enables to define three types of spatiotemporal constraints for future editing work, that are: (1) body action interacting type constraint, (2) mutual visibility constraint and (3) fidelity constraint. The body action interacting type constraint is mainly used in modeling the interaction event with contact, while the other two constraints can be applied to support the modeling of multi-party interaction event either with or without contact. The detailed definition and application of them will be presented in Section 4.1.3.1.

Table 1 Multi-party interaction dictionary with examples.

Interaction Dictionary	⊕ & ⊖	⓪ & ⓪	⊕ & ⊕	⊖ & ⊖	⊕ & ⓪	⊖ & ⓪
Fighting	Push&Pull	Attack&Dodge	Attack&Guard	After A&G	Punch	After Punch

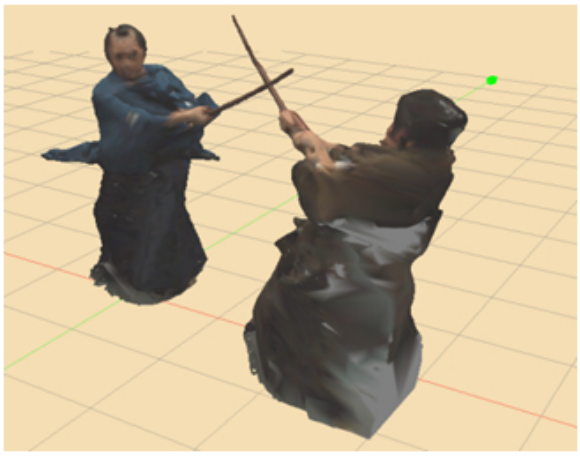


Fig. 5 Attack and guard scene from sword fighting 3D video sequence.

3. 3D Gaze Sensing for Gaze Action Modeling [14]

Gaze action is an important communicative cue in multi-party interaction events. In order to use this cue in the 3D video editing work, methods need to be developed that perform 3D gaze estimation on 3D video data, and model their spatiotemporal structures. In this section we present a novel method that estimates the three-dimensional gaze direction (gaze vector) for 3D video data.

The ideas behind this method are as follows. Generally the accuracy of the reconstructed 3D shape data is limited due to errors in the calibration and shape reconstruction processes, which could mislead the gaze estimation and/or decrease its accuracy. Fortunately, the 3D face surface is rather flat, which allows many cameras to observe it, and moreover, it has symmetric properties in both 3D shape and surface texture. Thus a super-resolution technique with symmetry prior can be applied to increase the 3D shape accuracy and the image resolution, making full use of original multi-view images.

The overall processing scheme of the proposed gaze estimation method is as follows. As is shown in Figure 6, given a sequence of 3D mesh data and corresponding multi-view video data, we first extract 2D face regions in multi-view images to estimate a rough 3D face surface area in each 3D mesh (Figure 6 II and II-I). Then estimate the symmetry plane of the 3D face surface area by: (1) first extract 3D feature points in the estimated 3D face surface area and then, (2) generate the symmetry plane by evaluating symmetric properties among the feature points (Figure 6 IV and V). Next, we reconstruct an accurate and high resolution frontal face surface by applying a super-resolution 3D shape reconstruction technique with the symmetry prior (Figure 6 VI). Then a virtual frontal face image with super-resolution can be generated (Figure 6 VII). Finally, we estimate the 3D gaze from the virtual frontal face image using a 3D eyeball model (Figure 6 VIII).

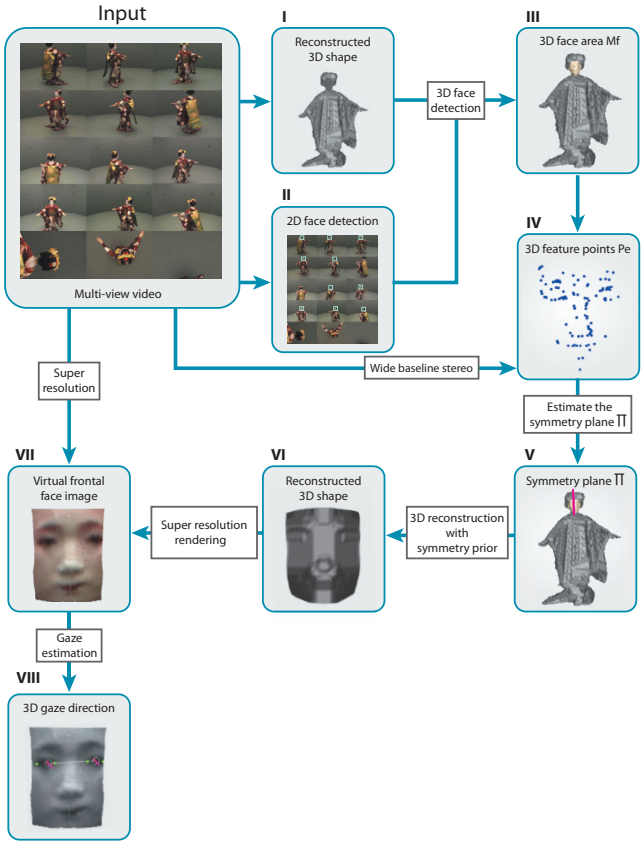


Fig. 6 Computational processes for 3D gaze estimation.

3.1 3D Face Surface Reconstruction using Symmetry Prior

This section presents the super-resolution 3D shape reconstruction algorithm using symmetry prior from the 3D mesh and corresponding multi-view images. It consists of (1) 3D face area detection, (2) symmetry plane estimation and (3) 3D face surface reconstruction in super-resolution. The algorithm processes frames one-by-one sequentially.

3.1.1 3D face area detection

First we propose an algorithm to detect the 3D positions and directions of the object's face from multiple-view videos. The basic idea is to use a 3D mesh as a *voting space* for accumulating partial evidence produced by applying an ordinary 2D face detector to each of the multi-view images. The evidence accumulation enables to (1) eliminate false-positive face detections in 2D images and (2) localize an accurate 3D face area on the 3D mesh.

Let M denote a 3D mesh of an object and $I_i (i = 1, \ldots, N)$ a set of corresponding multi-view images captured by cameras $c_i (i = 1, \ldots, N)$. The face area detection algorithm first detects a set of 2D face candidate regions F_i by applying a conventional 2D face detector to each I_i. It should be noted that F_i may include false-positive face areas due to texture patterns which accidentally look like a human face. Then all F_is are mapped onto M for evidence accumulation.

Figure 7 illustrates the detected 3D face area M_f.

Fig. 7 Detected 3D face area M_f painted in skin color.

3.1.2 Symmetry plane estimation

The assumption that human faces have symmetric properties in both 3D shape and surface texture allows us to reconstruct a more accurate 3D face surface than M_f and hence generate a higher resolution frontal face image than captured images.

The symmetry plane detection from M_f consists of two processes: (1) detect 3D feature points P_e on M_f (Figure 6 IV) and (2) apply RANSAC [18] to estimate the symmetry plane based on P_e (Figure 6 V).

3.1.3 3D feature points extraction

In order to find the symmetry plane that divides M_f into two symmetric parts, we first extract 3D feature points P_e on the local object surface specified by M_f. To avoid possible artifacts introduced by the texture generation, we apply a stereo-based edge feature detection method to the multi-view images as illustrated in Figure 8. That is, we establish sparse but reliable 2D-to-2D correspondences to obtain 3D feature points by triangulation [19]. This algorithm is based on the wide-baseline stereo by Furukawa [20] and augmented by a bi-directional uniqueness examination to improve the accuracy and robustness of the matching.

3.1.4 Symmetry plane estimation using 3D feature points

Having computed the reliable 3D feature point set $P_e = \{p_i\}$ $(i = 1, \ldots, N)$ in Section 3.1.3, we then estimate the symmetry plane π from P_e as follows (Figure 6 V). The idea is to generate a candidate symmetry plane π and compare the texture pattern around p_i with that of its symmetric position with respect to π. If π is a valid symmetry plane, then the textures should be reasonably similar.

(1) Randomly pick up two points p_i, p_j $(i \neq j) \in P_e$, and repeat the following processing for $K \leq N(N-1)/2$ times.
  (a) Compute the symmetry plane π_{ij} that makes p_i and p_j in the symmetric position.
  (b) Based on the hypothesized symmetry plane π_{ij} we can compute the symmetric position for each of the other $N-2$ points. Let $\check{p}_k$ denote the symmetric position of p_k $(k \neq i, j)$. Then we compare the textures at p_k and $\check{p}_k$. First we generate two $L \times L$ grids centered at p_k and $\check{p}_k$ in the 3D space. Note that these two grids lie on the planes that are perpendicular to the hypothesized symmetric plane, and the distance between neighboring grid points is d, which is a variable free to change according to the size of the 3D object. Since the 3D position of each grid point is computable, let p_k^{mn}, $\check{p}_k^{mn} (0 \leq m \leq L, 0 \leq n \leq L)$ denote the grid points on the grids centered at p_k and $\check{p}_k$. And let $Col(p_k^{mn})$ and $Col(\check{p}_k^{mn})$ denote the RGB color vectors of the grid points p_k^{mn} and $\check{p}_k^{mn}$ respectively, which are computed from the images by their best-observing cameras. Here we use M_f as the shape proxy for the state-based visibility evaluation[21]. Then the texture dissimilarity between p_k and $\check{p}_k$, d_{p_k}, is computed as Sum-of-Absolute-Difference:

$$d_{p_k} = \sum_{0 \leq m \leq L, 0 \leq n \leq L} \left| Col(p_k^{mn}) - Col(\check{p}_k^{mn}) \right|. \quad (12)$$

Note that if either p_k^{mn} or $\check{p}_k^{mn}$ is located outside the estimated face area, the point pair is considered as an outlier and a fixed value $diff$ is set to $\left| Col(p_k^{mn}) - Col(\check{p}_k^{mn}) \right|$. By computing d_{p_k} for all p_k $(k \neq i, j)$, we can evaluate the goodness of π_{ij} by

$$d_{ij} = \sum_{p_k \in P_e \setminus p_i, p_j} d_{p_k}. \quad (13)$$

(2) Select the symmetry plane π_{ij} having the smallest $d_{i,j}$ as the symmetry plane π.

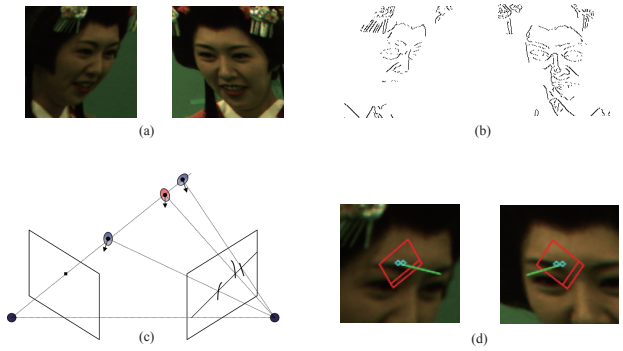


Fig. 8 Matching based on edge features. (a) rectified images, (b) edge features crossing epipolar lines, (c) texture similarity computation with normal direction optimization, (d) an example of matched pair. In (d), the red rectangles illustrate the windows used to compute the texture similarity, the green lines the surface normals, and the blue circles the endpoints of the edge features.

3.1.5 3D shape reconstruction using symmetry prior

The 3D shape reconstruction algorithm proposed in this paper utilizes the knowledge of symmetric properties of the human face to attain more accurate and higher resolution 3D shape reconstruction (Figure 6 VI). The algorithm is similar to the mesh-deformation algorithm proposed by Nobuhara *et al.* [22], but employs the symmetry constraint in deforming the mesh.

The processes so far described have generated the 3D face area $M_f = \{V_f, E_f\}$ as a sub-area of the original 3D mesh surface M and estimated its symmetry plane π (Figure 9(a)). With this symmetry plane, we first define the 3D face coordinate system

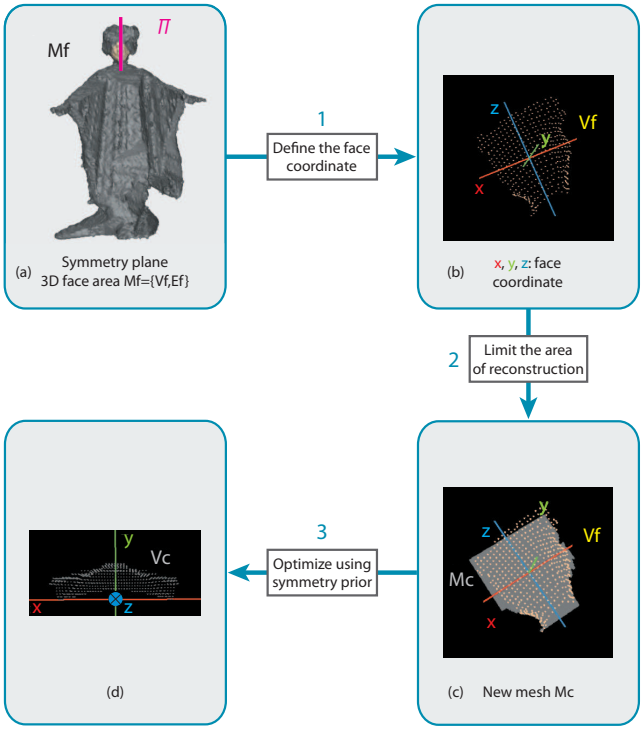


Fig. 9 3D face shape reconstruction using symmetry prior.

as illustrated in Figure 9(b): define the origin by the centroid of V_f and place the coordinate axes so that the symmetry plane π is aligned with $x = 0$ plane, the X-axis is defined by the normal vector of π, and the Z-axis by the principal axis of the point distribution of V_f on π. The Y-axis is computed by the cross-product of the other axes.

Then we generate a new mesh $M_c = \{V_c, E_c\}$ to model the higher resolution 3D face surface: project M_f onto the $y = 0$ plane and define a bounded regular mesh M_c on the 2D projected region. The gray area in Figure 9(c) illustrates M_c. That is, V_c and E_c denote the set of grid points and edges in this projected region, respectively. Note that the sampling pitch by the regular grid can be designed to increase the spatial resolution.

With this modeling, the 3D face surface reconstruction problem is transformed to that of finding the appropriate y value of each regular grid point in V_c (Figure 9(d)). Here the technical problems to be solved are (1) how we can introduce the symmetry constraint into the mesh deformation and (2) how we can find the optimal y values for V_c.

First, we represent the symmetry prior by

$$y = f(x, z) = f(-x, z), \tag{14}$$

where the function $f(x, z)$ returns the y value of the grid point at (x, z). Then, introduce the following discrete representation of y values:

$$y = \alpha i, \tag{15}$$

where i denotes an integer within a certain range, and α specifies the resolution of possible y values.

This discrete modeling allows us to formalize the shape reconstruction problem as a multi-labeling problem. That is, we can formulate the shape reconstruction with the symmetry prior as the minimization of the following objective function:

$$\mathcal{E}(M_c) = \sum_{v \in V_c, v_x \geq 0} \mathcal{E}_p(i_v) + \sum_{(u,v) \in E_c, u_x, v_x \geq 0} \mathcal{E}_c(i_u, i_v), \tag{16}$$

where v_x and u_x denote the x coordinate values of v and $u \in V_c$ respectively, and i_v and i_u integer labels to specify y values at v and u respectively. $\mathcal{E}_p(i_v)$ denotes the photo-consistency evaluation function at v and its symmetric position $\check{v}$. That is,

$$\begin{aligned} \mathcal{E}_p(i_v) &= \rho(v_x, \alpha i_v, v_z) + \rho(\check{v}_x, \alpha i_{\check{v}}, \check{v}_z) \\ &= \rho(v_x, \alpha i_v, v_z) + \rho(-v_x, \alpha i_v, v_z), \end{aligned} \tag{17}$$

where $\rho()$ denotes the photo-consistency evaluation function based on the state-based visibility with M as the shape proxy[21]. $\mathcal{E}_c(i_u, i_v)$ evaluates the smoothness in the y direction between a pair of connected grid points v and u:

$$\mathcal{E}_c(i_u, i_v) = \kappa |\alpha i_u - \alpha i_v| \tag{18}$$

where κ is a weighting factor to balance the photo-consistency and smoothness terms. This formalization forces the mesh deformation to satisfy the symmetry constraint defined by Equation (14) . We solve this minimization problem by belief-propagation[23], and obtain the 3D face surface satisfying both the photo-consistency and the symmetry constraint simultaneously.

3.2 Virtual Frontal Face Image Synthesis

In this section we present the view-dependent super-resolution approach, which can generate an optimized super-resolution image of the object 3D face.

With the optimized M_c, the virtual frontal view of M_c is generated for gaze estimation (Figure 6 VII): (1) locate a virtual camera with focal length f at $(0, P_{cam}, 0)$ and align its view direction at $(0, 0, 0)$ in the 3D face coordinate system defined in Section 3.1.5, and then (2) generate the virtual frontal face image by rendering M_c from the virtual camera by the super-resolution technique proposed by Tung *et al.* [24]:

(1) Set a high-resolution pixel grid on the image plane of the virtual camera.

(2) Project each pixel of the original multi-view images, say source pixels, onto the pixel grid via M_c. That is, back project the source pixels onto M_c first, and then project the points on M_c to the pixel grid of the virtual camera. In this process we choose the nearest grid point as the final projection point of each source pixel. In addition we ignore source pixels if their projections are occluded by M.

(3) For each grid point with source pixel projections, compute its color by averaging associated source pixel colors. Otherwise, interpolate the grid point color using colors of its neighbors.

Figure 10 shows a synthesized virtual front face image, where the image resolution is increased by the super-resolution rendering process.

3.3 Gaze Estimation using 3D Eyeball Model

At the last stage, we propose to introduce a 3D eyeball model to estimate the object's gaze direction (Figure 6 VIII). Figure 11 illustrates the structure of the model. The red arrow indicates the

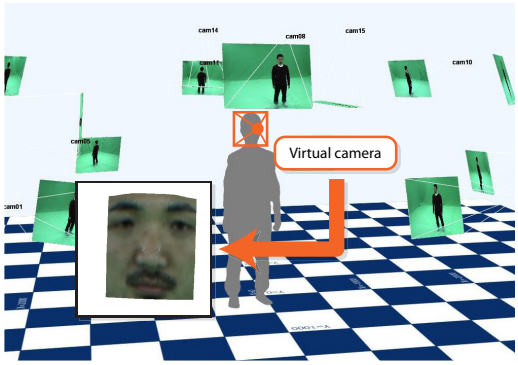


Fig. 10 Virtual frontal face image synthesis.

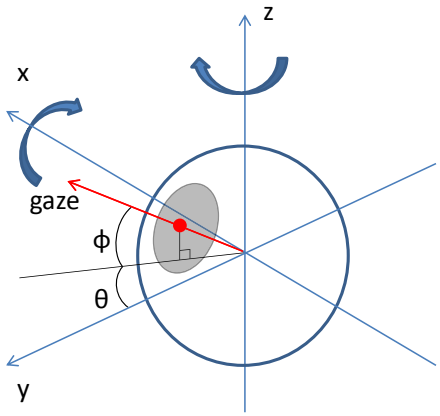


Fig. 11 3D eyeball model.

3D gaze direction, and θ and φ denote the horizontal and vertical rotation angles of the eyeball, respectively. This model is designed based on the following three assumptions:

(1) The eyeball is fixed inside the eye socket and it can rotate horizontally and vertically around the eyeball center.
(2) The gaze direction is defined by the 3D vector pointing from the eyeball center to the iris center.
(3) The radius of the eyeball is equal to the diameter of the iris. This assumption is made based on medical statics data.

To apply this model to the 3D gaze estimation, the eyeball model of the object should be estimated first by the following off-line process:

(1) Collect virtual frontal face images in which eyes look straight forward by hand.
(2) For each image, detect the following eye feature points (Figure 12) for each eye: 2D eye corners, q_a and q_e, 2D iris center, q_c, and the intersecting points between the iris border and the eye corner line connecting q_a and q_e, q_b and q_d. The eye corners are located by the AAM [25], and the iris is detected by applying Kawaguchi *et al.* [26]'s method. Note that all feature points for the right eye illustrated in Figure 12 are mirrored with respect to the symmetry plane to represent those for the left eye.
(3) For each eye, let d denote the average 3D diameter of the iris, and consequently the eyeball radius. The 3D diameter of the iris is defined by the 3D distance between p_b and p_d on the face surface M_c, which are obtained by back-projecting q_b and q_d onto M_c respectively.

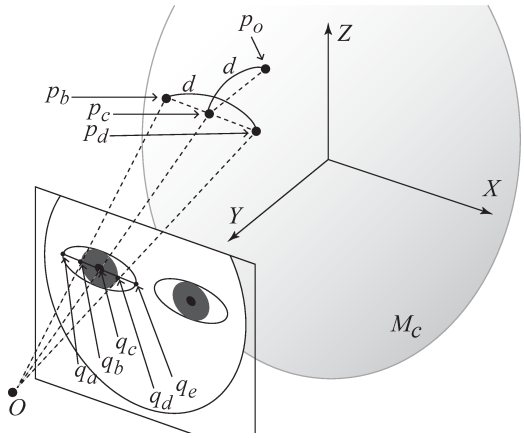


Fig. 12 Eyeball center position estimation.

(4) For each eye, compute the average 2D relative position t of the iris center q_c with respect to the eye corners q_a and q_e. That is, t denotes the weighting parameter to represent q_c by the weighted average of q_a and q_e: $q_c = (1-t)q_a + tq_e$ where $t = |q_c - q_a| \,/\, |q_e - q_a|$.

This process estimates the eye model parameters for the left and right eyes respectively: d_{left} and t_{left}, and d_{right} and t_{right}. In what follows, we eliminate the suffix for simplicity.

With the eye ball model parameters d and t, compute the 3D gaze directions of the left and right eyes from each 3D video frame by the following process (Figures 12 and 13). Note that all 3D points as well as the virtual frontal face image in the 3D gaze estimation below are represented in the face coordinated system defined in Section 3.1.5 and Figure 9(b), which is dynamically defined depending on the 3D face position and direction in each 3D video frame.

(5) Apply the following process to the left and right eyes, respectively.
(6) Detect 2D eye corners q_a and q_e, and the iris center q_c from the synthesized frontal face image.
(7) Compute the 3D iris center position p_c by back-projecting q_c onto M_c.
(8) Compute the 3D eyeball center p_o by

$$p_o = p_{\tilde{c}} + (0, -d, 0)^\top. \tag{19}$$

Here $p_{\tilde{c}}$ denotes the back-projection of $q_{\tilde{c}} = (1-t)q_a + tq_e$ onto M_c, where $q_{\tilde{c}}$ represents the 2D position of the assumed iris center if the eye were looking straight forward.
(9) Finally the 3D gaze direction is given as the line passing through p_o and p_c.

It should be noted that here the gaze vectors are estimated for the left eye and right eye, respectively. While to keep the simplicity of gaze action modeling, the average of the gaze vectors estimated from the object's two eyes, which is denoted as $\vec{G_i}$, is used to generate the gazing cone as described in Section 2.1.3. The 3D position of the center of the two eyes can also be easily computed since the 3D eyeball center p_o of each eye is already computed in Step VIII-8.

3.4 Performance Evaluation

In this section we evaluate the performance of the proposed

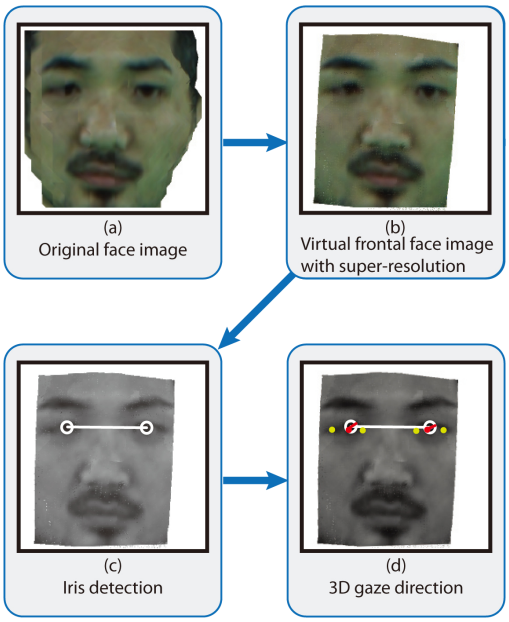


Fig. 13 Gaze estimation process. (a) original face image generated based on the original 3D mesh M (b) virtual super-resolution frontal face image generated based on the reconstructed face surface M_c (d) detected irises (circles) (e) estimated eye corners (green dots) and gaze directions (red lines).

Fig. 14 Input multi-view data A.

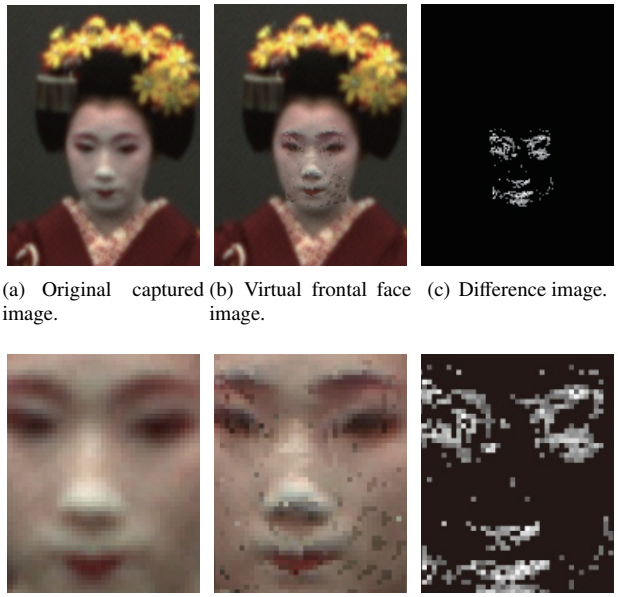
method with real data.

3.4.1 Shape reconstruction using symmetry prior

First, we present the analysis on how the accuracy of the reconstructed 3D face shape is improved by introducing the symmetry prior.

3.4.1.1 Experiment setup

For this evaluation we used two sets of data in doing the experiment. Data A (Figure 14) is captured by 15 calibrated UXGA cameras running at 25 HZ with 1 msec shutter speed, while Data B is captured by 16 calibrated UXGA cameras running at 25 Hz with 1 msec shutter speed.



(a) Original captured image. (b) Virtual frontal face image. (c) Difference image.

(d) Enlarged Original captured image. (e) Enlarged Virtual frontal face image. (f) Enlarged Difference image.

Fig. 15 Difference image with the proposed method.

3.4.1.2 Evaluation method

We measure the contribution of the symmetry prior to the reconstruction accuracy by means of leave-one-out experiments. We keep one camera c_f for evaluation, and use the other 15 cameras to render the face image viewed from camera c_f by the rendering algorithm described in Section 3.2. Note that the size and resolution of the rendered image is adjusted to coincide with that of the image captured by c_f. Let I'_f denote the rendered image. Then we compute the mean-squared-error between the rendered image I'_f and the originally captured image I_f:

$$\mathrm{MSE} = \frac{1}{N} \sum_{(x,y)\in I_f} (I_f(x,y) - I'_f(x,y))^2, \qquad (20)$$

where N is the total number of effective pixels in I'_f.

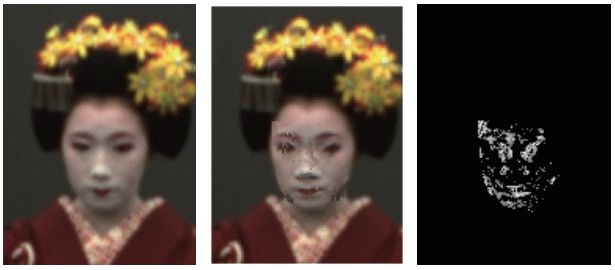
In this experiment we compute I'_f in two ways as follows: (1) the virtual view image with the original 3D shape. (2) the virtual view image with the reconstructed 3D shape using symmetry prior. By comparing these two types of I'_f with the original image I separately, we can comprehensively evaluate the effect of introducing the symmetry prior.

3.4.1.3 Experiment results

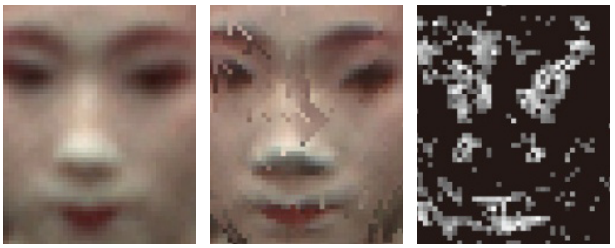
Figure 15 and 16 illustrate the difference images between the original captured image and the synthesized virtual frontal face image, generated with the original 3D shape and the reconstructed 3D shape using symmetry prior, respectively. In Figure 15, the synthesized face image in the middle is less blurred than the original captured image, and the differences mainly occur on the edges, proving that the synthesized virtual frontal face image with the proposed method has higher resolution than the original captured image. As for the one with the original 3D face shape, obvious differences appear on the cheeks as well as the edges. Table 2 illustrates that the mean-squared-error MSE of the proposed method is smaller than using the original 3D face shape.

3.4.2 Gaze estimation

To prove the effectiveness of our gaze estimation method, we



(a) Original captured image. (b) Virtual frontal face image. (c) Difference image.



(d) Enlarged Original captured image. (e) Enlarged Virtual frontal face image. (f) Enlarged Difference image.

Fig. 16 Difference image with the original 3D shape.

Table 2 MSE with the original captured image.

	MSE	effective pixels
proposed method	10.23	1900
original 3D shape	11.07	2223

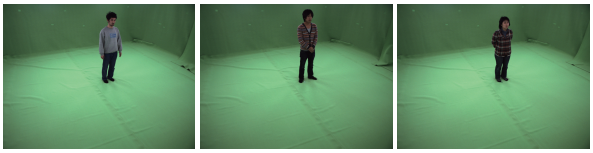


Fig. 17 Multi-view input data with three different objects.

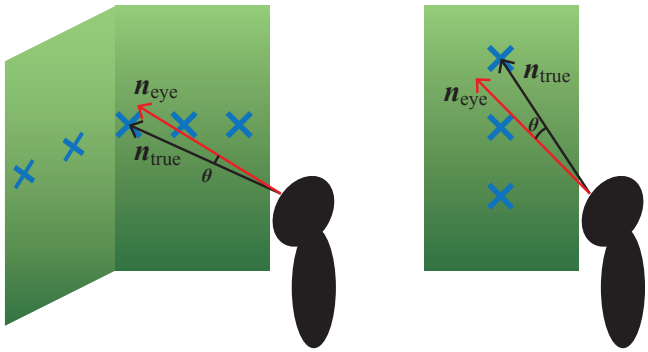


Fig. 18 Gaze estimation error evaluation.

prepared the input multi-view videos captured with three different people for evaluation (Figure 17).

3.4.2.1 Experiment setup

In order to quantitatively evaluate the accuracy of the gaze estimation processing, we designed our experiment as follows. Figure 18 illustrates the experimental environments. A human subject stands at about 2.5 m away from the wall and looks at (1) horizontally aligned markers one by one, and (2) vertically aligned markers one by one (Figure 18 left and right respectively). For each marker, its 3D position p_m in the object oriented coordinate system is measured manually.

Table 3 Average gaze estimation errors of the proposed method.

	L, horizontal	R, horizontal	L, vertical	R, vertical
original	0.4326	0.4002	0.5321	0.5063
proposed	0.3297	0.3234	0.4610	0.4472

3.4.2.2 Evaluation method

In the multi-view video data, we first selected those video frames where the subject was stably looking at each marker. Then, for each selected video frame, apply the above mentioned gaze estimation processes from Step I to Step VIII to obtain the estimated 3D gazing vector $\vec{n}_{\text{eye}}$. Note that the gaze estimation is conducted for the left and right eyes independently, meaning that we have $\vec{n}_{\text{eye}}$ for each eye. The ground-truth 3D gazing direction vector $\vec{n}_{\text{true}}$ is defined by a 3D vector from the eye ball center p_o computed by Equation (19) to each 3D marker position. Then, the angular error of the 3D gaze estimation in each selected video frame is computed for each eye by

$$\theta = \arccos\left(\frac{\vec{n}_{\text{eye}} \cdot \vec{n}_{\text{true}}}{|\vec{n}_{\text{eye}}||\vec{n}_{\text{true}}|}\right) \tag{21}$$

3.4.2.3 Experiment results

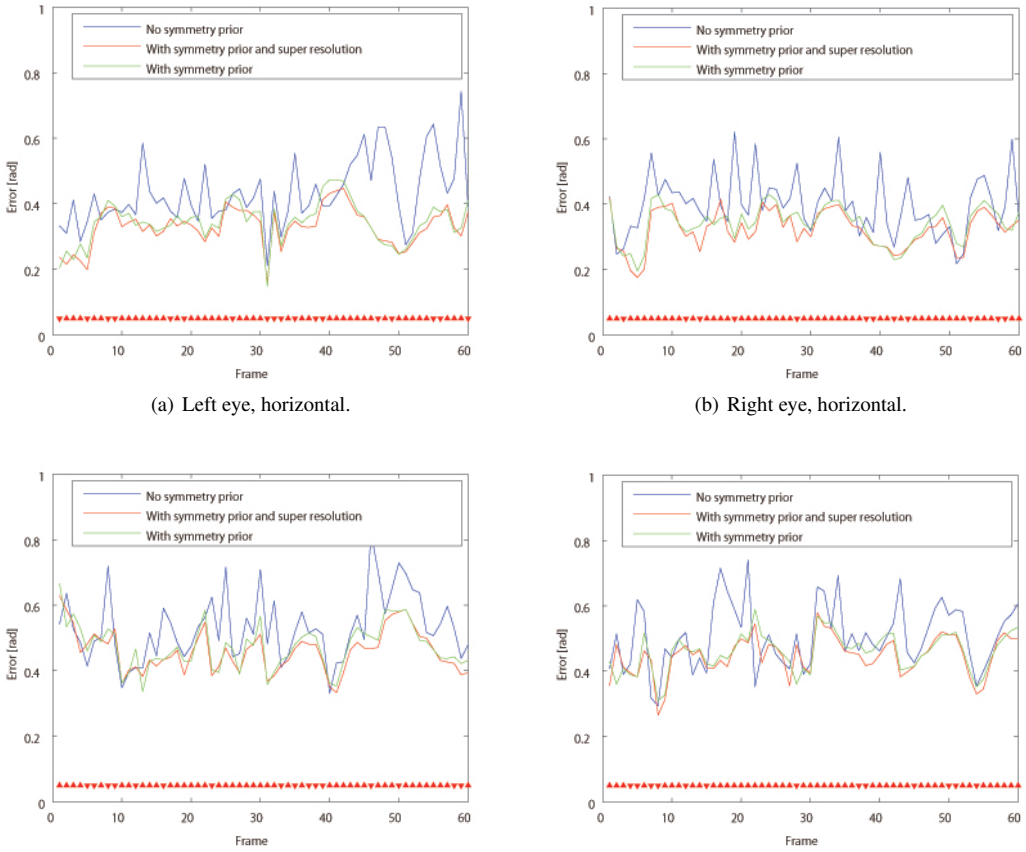
In the analysis of the proposed method, the angular gaze estimation errors are evaluated for the left and right eyes as well as for the horizontal and vertical directions, respectively, which gives four different error evaluation results as shown in Figures 19(a), 19(b), 19(c) and 19(d).

In each figure, three computational methods are compared: without the symmetry prior, with the symmetry prior alone, and with both the symmetry prior and the super-resolution image rendering technique. The horizontal axis in each figure denotes the selected frame IDs where the subject was stably looking at each marker. The upward and downward triangles at the bottom in each figure denote the signs (*i.e.* positive or negative) of the errors by the method with both the symmetry prior and the super-resolution image rendering technique. Table 3 shows the average errors for the first and the third methods.

In all results, the symmetry prior improved the stability of the iris detection and the accuracy of the gazing direction estimation, while the improvement by the super resolution is limited. This is because the performance of the iris localization is not so accurate. As is well known, errors in the horizontal direction are much smaller than those in the vertical direction, because of the shape and movable range of human eyes. These results demonstrate the effectiveness and robustness of the presented method.

4. Multi-party Interaction Scene Editing Algorithm and Evaluation [16]

In this section we present the spatiotemporal 3D editing algorithm using the AHV-based multi-party interaction modeling method. We first give a brief overview of our algorithm and the definition of related terms. Then we present each step of our action editing algorithm in details. Finally, an evaluation with real 3D video data is performed to prove the effectiveness of the proposed editing algorithm.



(a) Left eye, horizontal. (b) Right eye, horizontal.

(c) Left eye, vertical. (d) Right eye, vertical.

Fig. 19 Gaze estimation errors of the proposed method. The upward and downward triangles at the bottom in each figure denote the signs (*i.e.* positive or negative) of the errors by the method with both the symmetry prior and the super-resolution image rendering technique.

4.1 Multi-party Interaction Scene Editing Algorithm

4.1.1 Overview and definition of terms

The proposed multi-party interaction editing method consists of three steps: (1) data segmentation and temporal alignment, (2) intra-key segment editing by AHV-based constraint satisfaction and (3) inter-key segment global optimization. We first define editing-related terms as follows:

(1) Action sequence: The whole sequence of each originally captured object, in the form of 3D surface meshes. Let $M = \{V, E\}$ denote a 3D mesh consisting of a vertex set V and an edge set E, then an action sequence can be denoted as $S_i = \{M_s | s = 1, \ldots, n_s\}$.

(2) Key segment: A subset of action sequence S_i, storing frames of interactive actions with contact (*e.g.* handshaking, hitting on the body, ...). We use $K_i = \{M_k | k = 1, \ldots, n_k \quad n_k < n_s\}$ to denote key segments. Since the objects are supposed to perform actions that match with each other, key segments should appear in pairs from different action sequences.

(3) Transitional segment: Intermediate frames between key segments in the action sequence, in which interactive actions without contact are stored. We use $T_i = \{M_t | t = 1, \ldots, n_t \quad n_t < n_s\}$ to denote transitional segments, and it should be noted that there can be no transitional segment between two key segments.

(4) Action Segment: A generic term for both key segment and transitional segment, denoted by $A_i = \{M_a | a = 1, \ldots, n_a \quad n_a < n_s\}$.

(5) Interaction scene: Spatiotemporally synchronized short multi-party interaction scene synthesized from a single pair of key segments.

(6) Interaction sequence: Spatiotemporally synchronized long multi-party interaction sequence synthesized by integrating all the key segment pairs and transitional segments in the original action sequences.

4.1.2 Action sequence segmentation and temporal alignment

Before performing action editing process, the original action sequences are required to be segmented into pairs of key segments and transitional segments. As described earlier, this action sequence segmentation is performed manually by the editor, through the following two steps.

(1) Select the action frames containing interactive body actions with contact to be key segments, then those remaining action frames of interactive body actions without contact belong to the transitional segments.

(2) For each pair of key segments, if body actions from each object inside this pair are unidirectional, keep it as is. Otherwise, further decompose it into multiple key segment pairs until all the body actions inside each key segment pair are unidirectional. This will ensure the simplicity of the AHV

of body actions for future editing work.

With the segmentation process above we can ensure that the body actions inside each key segment pair are unidirectional so that the multi-party dictionary defined in Section 2.3.1 can be easily applied to perform the action editing process.

On the other hand, since the potential interactively corresponding actions in the original captured data would inevitably have temporal mismatches, action segments of each pair may also have different lengths of time duration. Therefore, we apply the following linear stretching to normalize the durations.

For a pair of action segment A_i^A and A_i^B,
if $\tau_i^A > \tau_i^B$,

$$M'^A_{i_m} = M^A_{i_n} \quad (0 < m < \tau_i^B, n = \left\lfloor m \times \frac{\tau_i^A}{\tau_i^B} \right\rfloor), \tag{22}$$

otherwise,

$$M'^B_{i_m} = M^B_{i_n} \quad (0 < m < \tau_i^A, n = \left\lfloor m \times \frac{\tau_i^A}{\tau_i^B} \right\rfloor), \tag{23}$$

where τ_i^A and τ_i^B represent the duration of A_i^A and A_i^B, $M_{i_n}^A$ represents the nth frame in A_i^A, and $M'^A_{i_m}$ represents the mth frame of the normalized A_i^A. $\lfloor x \rfloor$ is the floor function that returns the largest integer not greater than x.

4.1.3 Intra-key segment editing

4.1.3.1 Editor defined constraints

By performing the data segmentation and temporal alignment, multiple objects' interactive body actions with contact are organized into key segment pairs, and inside each pair objects' body actions are scaled into the same temporal length. Then the AHVs of each object's body action in each segment pair can be computed as $v^A_{\tau_i}(x, y, z, t)$ and $v^B_{\tau_i}(x, y, z, t)$.

For each interaction event inside a key segment pair, we model that the two interacting objects should follow certain spatiotemporal constraints based on (1) body action interacting type, (2) mutual visibility and (3) fidelity requirement.

(1) Body action interacting type constraint:

First for each key segment pair the interactive contact type of objects' body actions should be decided by the editor based on his/her editing intension, producing one type of spatiotemporal constraint from the interaction dictionary.

(2) Mutual visibility constraint:

Second, for multi-party interaction scenes, usually the objects should be visible for each other. We first compute the object's gazing direction $\overrightarrow{G_i}$ for each frame using the method introduced in Section 3. Then as discussed in Section 3.1.3, a gazing cone can be generated for each object at each frame. With these gazing cones we can create the AHV of each object's interactive zone, denoted by $g^A_{\tau_i}(x, y, z, t)$ and $g^B_{\tau_i}(x, y, z, t)$. Then the mutual visibility constraint can be defined as: one object's AHV of the gazing cone $g^A_{\tau_i}(x, y, z, t)$ and the other object's AHV of the body action $v^B_{\tau_i}(x, y, z, t)$ fulfill the ⓪ & ⓪-(a) type from the multi-party interaction dictionary. The detailed mathematical constraint is described as follows:

$$\bigcap_{t_i=0}^{\tau_i-1}((G'^A_{t_i}(x,y,z) \cap V'^B_{t_i}(x,y,z)) \cap (G'^B_{t_i}(x,y,z) \cap V'^A_{t_i}(x,y,z))) \neq \phi. \tag{24}$$

Here $G'^A_{t_i}(x, y, z)$ represents a subset of $g^A_{\tau_i}(x, y, z, t)$'s spatial volume $G^A_i(x, y, z)$ at time t, and $V'^B_{t_i}(x, y, z)$ represents a subset of $v^B_{\tau_i}(x, y, z, t)$'s spatial volume $V^B_i(x, y, z)$ at time t

(3) Fidelity constraint:

Third, the fidelity requirement avoids unnatural fakeness, such as two objects merge into each other's body. Let $v^{\bar{A}}_{\tau_i}(x, y, z, t)$ and $v^{\bar{B}}_{\tau_i}(x, y, z, t)$ denote the AHVs of the objects that should not contact in the interaction event, then the fidelity constraint can be defined as: $v^{\bar{A}}_{\tau_i}(x, y, z, t)$ and $v^{\bar{B}}_{\tau_i}(x, y, z, t)$ fulfill the ⓪ & ⓪-(c) type from the multi-party interaction dictionary. The detailed mathematical constraint is described as follows:

$$\forall\, v^{\bar{A}}_{\tau_i} \bigcap_{(x,y,z)} v^{\bar{B}}_{\tau_i}, \quad \bigcup_{i=1,i'=1}^{i=n,i'=n'} ([t^{\bar{A}}_{\text{start}_i}, t^{\bar{A}}_{\text{end}_i}] \cap [t^{\bar{B}}_{\text{start}_{i'}}, t^{\bar{B}}_{\text{end}_{i'}}]) = \phi. \tag{25}$$

4.1.3.2 Action scene editing using constraint satisfaction

With the editor defined constraints, we compute acceptable relative positions of multiple objects by constraint satisfaction.

For each action scene consisting of a pair of key segments K_i^A and K_i^B, first multiple objects are put into the same world coordinate system. Let C_i^A, C_i^B denote the 2D positions of the objects's AHV root nodes on the ground plane, $L_i^{AB} = |\overrightarrow{C_i^A C_i^B}|$ denote the distance between multiple objects, and let θ_i^A, θ_i^B represent the included angles between the average viewing direction of each AHV and line $C_i^A C_i^B$, then the relative positions between multiple objects can be represented by $P_i^{AB} = (L^{AB}_{i_n}, \theta^A_{i_n}, \theta^B_{i_n})$.

The editing work for a single action scene is to find a sub-space $R_i^{AB} = (L^{AB}_{i_j}, \theta^A_{i_j}, \theta^B_{i_j}) \subset P_i^{AB}$, in which parameters ensure that the objects completely fulfill the editor defined constraints.

We solve this problem by a generation-and-test approach. We sample parameters in P_i^{AB} space, and test if the synthesized segment satisfies the editor-defined constraints or not. Hence, the acceptable parameter space R_i^{AB} is descretized as a set of sampled points, each of which represents an acceptable editing result for this segment pair.

In order to increase the computational efficiency of the constraint satisfaction process, we introduce a coarse-to-fine strategy that tries to reduce the amount of samples to be tested by using a multi-level processing.

Step 1 Firstly the P_i^{AB} space is decomposed into N levels, and the sampling resolution of level n ($1 < n < N$) is 2^{n-1} times lower than the finest resolution ($Res(L), Res(\theta), Res(\theta)$).

Step 2 Next, the editor-defined constraints are tested for all the samples in *level* $n = N$, discarding the parameters that do not fulfill the constraints and getting a solution group $R^{AB}_{i^N}$.

Step 3 In turn, the same test will be repeated for *level* $(n - 1) = (1 < n <= N)$ sampling space, and only the samples whose distances to their nearest points in $R^{AB}_{i^n}$ are no larger than the sampling resolution of the current level will be tested.

By repeating such process down to *level* $n = 1$, the samples of

possible solutions required to be tested would be drastically reduced.

4.1.4 Inter-key segment optimization

The intra-key segment editing process computes the solution groups for each key segment pair. Then we will continue to combine them together with the transitional segments in order to synthesize a complete multi-party interaction sequence, by (1) finding the optimal solution in each key segment pair and (2) adding in the transitional segments and performing path optimization.

4.1.4.1 Optimal solution searching for key segments

In the original data before editing, we denote the 2D positions of the AHVs' root node on the ground plane in the world coordinate system with $C_1^A, C_2^A, \ldots, C_I^A, C_1^B, C_2^B, \ldots, C_I^B$, and we consider the object's average gazing directions from each frame as the facing directions of the AHVs in the world coordinate system, denoted by $\overrightarrow{F_1^A}, \overrightarrow{F_2^A}, \ldots, \overrightarrow{F_I^A}, \overrightarrow{F_1^B}, \overrightarrow{F_2^B}, \ldots, \overrightarrow{F_I^B}$. In order to make natural and smooth editing preserving the original action, we should maintain the vectors $\overrightarrow{C_i^A C_{i+1}^A}$ and $\overrightarrow{C_i^B C_{i+1}^B}$ (i=1,...,I-1) as much as possible to minimize the artificial offsets. As well, the facing directions $\overrightarrow{F_i^A}$, $\overrightarrow{F_i^B}$ should also be preserved. If we denote the edited ideal positions of the AHVs' root node as $\bar{C}_1^A, \bar{C}_2^A, \ldots, \bar{C}_I^A, \bar{C}_1^B, \bar{C}_2^B, \ldots, \bar{C}_I^B$, the edited facing directions of the AHVs as $\overrightarrow{\bar{F}_1^A}, \overrightarrow{\bar{F}_2^A}, \ldots, \overrightarrow{\bar{F}_I^A}, \overrightarrow{\bar{F}_1^B}, \overrightarrow{\bar{F}_2^B}, \ldots, \overrightarrow{\bar{F}_I^B}$, and the angles between $\overrightarrow{\bar{F}_i^A}$ and $\overrightarrow{\bar{C}_i^A \bar{C}_i^B}$ as $\bar{\theta}_i^A$, then we should search for the optimal solution for each key segment pair by fulfilling:

$$\min \sum_{i=1}^{I-1} [|\overrightarrow{C_i^A C_{i+1}^A} - \overrightarrow{\bar{C}_i^A \bar{C}_{i+1}^A}|^2 + |\overrightarrow{C_i^B C_{i+1}^B} - \overrightarrow{\bar{C}_i^B \bar{C}_{i+1}^B}|^2 + \lambda(|\overrightarrow{F_i^A} - \overrightarrow{\bar{F}_i^A}|^2 + |\overrightarrow{F}_i^B - \overrightarrow{\bar{F}_i^B}|^2)], \tag{26}$$

and

$$|\overrightarrow{\bar{C}_i^A \bar{C}_i^B}| = L_i^{\bar{A}\bar{B}} = L_{i_j}^A B = |\overrightarrow{C_{i_j}^A C_{i_j}'^B}|, \tag{27}$$

$$\bar{\theta}_i^A = \theta_i^A, \bar{\theta}_i^B = \theta_i^B, \tag{28}$$

where λ is a weighting factor to balance the translational and rotational artifacts. The optimal solution for all the key segments can be computed by solving function (26), (27) and (28) using dynamic programming approach.

4.1.4.2 Path optimization for transitional segments

Having found the optimal solutions for all the key segments, we then need to add in the transitional segments and perform a path optimization to smooth out the offsets caused by the intra-key segment editing process.

Suppose inside a transitional segment, the original positions of the object's center of masses for all the frames are $C_1, C_2, \ldots, C_{m-1}$ and the edited positions are denoted as $\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_{m-1}$. Original and edited facing directions are denoted respectively as $\overrightarrow{F_1}, \overrightarrow{F_2}, \ldots, \overrightarrow{F_{m-1}}$ and $\overrightarrow{\bar{F}_1}, \overrightarrow{\bar{F}_2}, \ldots, \overrightarrow{\bar{F}_{m-1}}$. Then we define our objective function for performing path optimization as follows:

$$f = \sum_{i=1}^{m-1} [\delta f_a(\bar{C}_i) + (1-\delta)\omega_i^C f_v(\bar{C}_i) + \lambda(\delta f_a(\overrightarrow{\bar{F}_i}) + (1-\delta)\omega_i^F f_v(\overrightarrow{\bar{F}_i})]. \tag{29}$$

Here δ and λ are weighting factors to balance the acceleration and speed, the translation and rotation, respectively. Specifically, $f_a(\bar{C}_i)$ and $f_a(\overrightarrow{\bar{F}_i})$ is defined as follows to preserve the original accelerations for each frame:

$$f_a(\bar{C}_i) = |(C_{i-1} - 2C_i + C_{i+1}) - (\bar{C}_{i-1} - 2\bar{C}_i + \bar{C}_{i+1})|^2, \tag{30}$$

$$f_a(\overrightarrow{\bar{F}_i}) = |(\overrightarrow{F_{i-1}} - 2\overrightarrow{F_i} + \overrightarrow{F_{i+1}}) - (\overrightarrow{\bar{F}_{i-1}} - 2\overrightarrow{\bar{F}_i} + \overrightarrow{\bar{F}_{i+1}})|^2, \tag{31}$$

and $f_v(\bar{C}_i)$, $f_v(\overrightarrow{\bar{F}_i})$ is defined as follows to preserve the original speed for each frame:

$$f_v(\bar{C}_i) = |\frac{C_{i-1} - C_{i+1}}{2} - \frac{\bar{C}_{i-1} - \bar{C}_{i+1}}{2}|^2, \tag{32}$$

$$f_v(\overrightarrow{\bar{F}_i}) = |\frac{\overrightarrow{F_{i-1}} - \overrightarrow{F_{i+1}}}{2} - \frac{\overrightarrow{\bar{F}_{i-1}} - \overrightarrow{\bar{F}_{i+1}}}{2}|^2. \tag{33}$$

Besides,

$$\omega_i^C = \frac{1}{1 + \exp\{\alpha(v_i - v_k)\}}, \tag{34}$$

$$\omega_i^F = \frac{1}{1 + \exp\{\alpha(r_i - r_k)\}}, \tag{35}$$

$$v_i = \frac{|C_{i-1} - C_i| + |C_i - C_{i+1}|}{2}, \tag{36}$$

$$r_i = \frac{|\overrightarrow{F_{i-1}} - \overrightarrow{F_i}| + |\overrightarrow{F_i} - \overrightarrow{F_{i+1}}|}{2}, \tag{37}$$

where v_k and r_k are constant parameters to be set by editors.

Minimizing this objective function is a non-linear optimization of $2m$ parameters for positions and m parameters for rotation. By applying this process for each object in each transitional segment, we can perform path optimization onto all the transitional segments and finally, acquire a natural looking spatiotemporally synchronized multi-party interaction 3D video sequence for free-viewpoint browsing.

4.2 Experiment and Evaluation

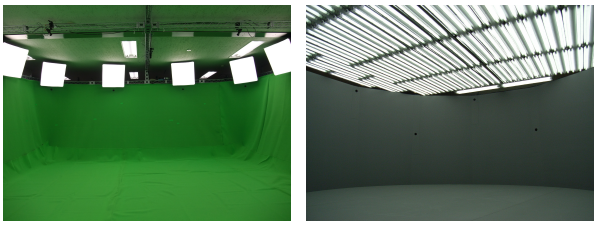
In this section we present an evaluation of the proposed multi-party interaction editing method with real 3D video data.

4.2.1 Experiment setup and pre-processing

In order to prove the effectiveness of the proposed method, we prepared four sets of real 3D video data (Handshaking, Highfive, Chambara, Kungfu) captured from two studios with different setups. Each of them includes a multi-party interaction sequence performed by two actors, separately captured by 16/15 calibrated UXGA/XGA cameras running at 25 Hz with 1 msec shutter speed. Fig. 20 shows an overview of the 3D video studios being used for this evaluation. Fig. 21 shows the original separately captured image data of multiple objects. Note that the Chambara and Kunfu data are captured using studio A (Fig. 20(a)), while the Handshaking and Highfive data are captured under another studio setup B (Fig. 20(b)).

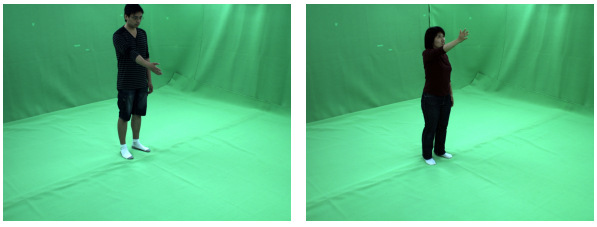
In the Handshaking and Highfive datasets, objects perform simple daily communicative actions of shaking hands and clapping hands. While in the Chambara dataset, the two separately captured actors are performing a complex pre-designed sword fighting action sequence, and in the Kungfu dataset, a pre-designed Kungfu fighting sequence is separately performed by two actors as the multi-party interaction events.

As for pre-processing, each dataset is manually segmented into

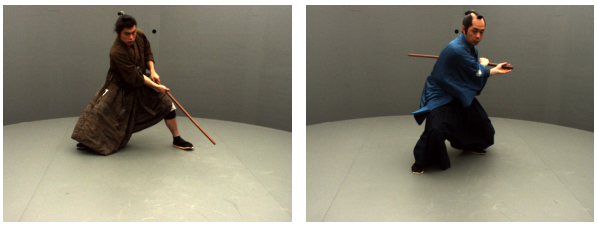


(a) Studio A. (b) Studio B.

Fig. 20 Two different 3D video studios used in experiments.



(a) Handshaking data, object A. (b) Handshaking data, object B.



(c) Chambara data, object A. (d) Chambara data, object B.

Fig. 21 Separately captured image data of multiple objects from two different studios.

key segment and transitional segment pairs, following the criteria described in earlier section. Inside each action segment pair, actions of multiple objects are aligned into the same temporal length.

4.2.2 Qualitative evaluation

To demonstrate the effectiveness of the proposed method, we first perform a qualitative evaluation by visually observing the editing results. Figs. 22(a)-(h) show the editing results of the Handshaking and Highfive dataset. In the editing results multiple objects actions properly match with each other.

For the two more complex dataset, Chambara and Kunfu, we conduct the evaluation by comparing the editing results by our method (Figs. 23(e)-(h), Figs. 24(e)-(h)) with the results by a baseline method (Figs. 23(a)-(d), Figs. 24(a)-(d)) in which only the time-duration is aligned and an initial relative position of multiple objects is given in original data. Note that the baseline method only puts the original data into the same coordinate system and manually gives an initial relative position of the separately captured data. It does not contain any more spatial editing, so the action dynamics of its results is nearly the same as the original data.

As illustrated in Figs. 23(a)-(d) and Figs. 24(a)-(d), lots of spatial mismatches exist in the synthesized 3D scenes. On the other hand, the editing results using the proposed method are illustrated in Figs. 23(e)-(h) and Figs. 24(e)-(h). It can be clarified that in each synthesized interaction scene, the relative location of the two objects looks natural and reasonable, as well they well qualify the editor defined spatiotemporal constraints.

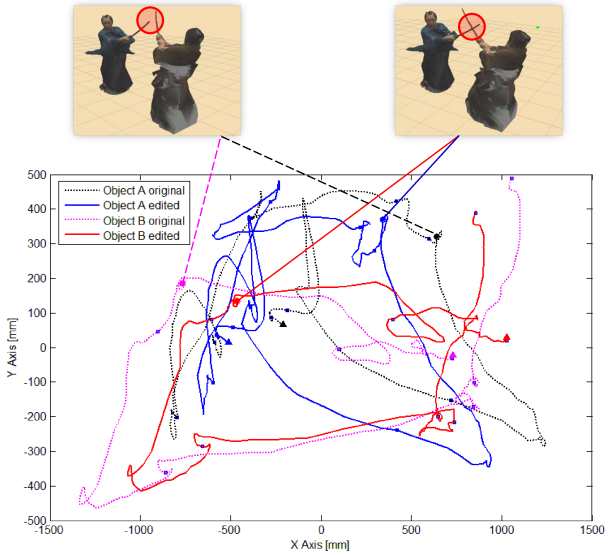


Fig. 25 Spatiotemporal editing result for Chambara dataset.

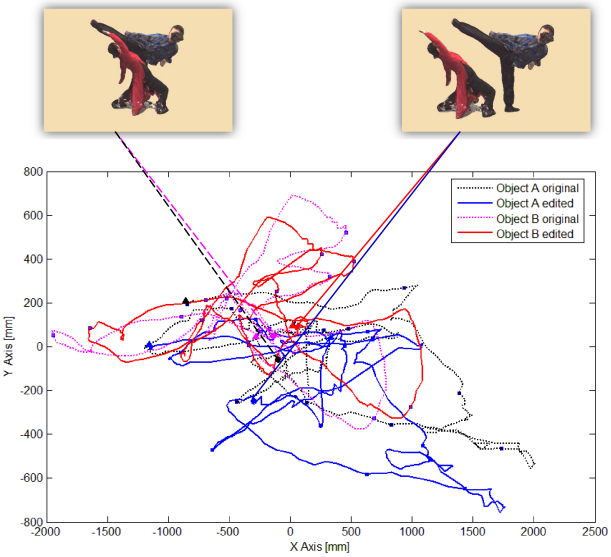


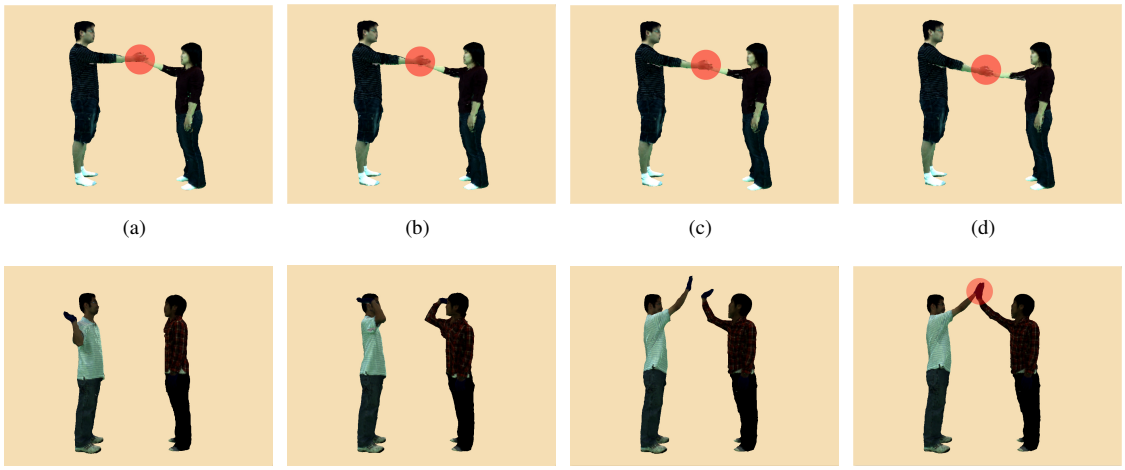
Fig. 26 Spatiotemporal editing result for Kungfu dataset.

4.2.3 Quantitative evaluation

Quantitative evaluations are performed for the Chambara and Kunfu data, respectively.

For Chambara and Kungfu data, in both Figs. 25 and 26, four trajectories on the floor plane are compared: object A before editing, object A after editing, object B before editing and object B after editing. They are drawn in different colors, and the dotted lines and full lines represent the trajectories before and after editing, respectively. Besides, the small colored squares on each curve denote every 50 frames of the data. We also give an example of the objects' positions and the corresponding rendered images of the synthesized results.

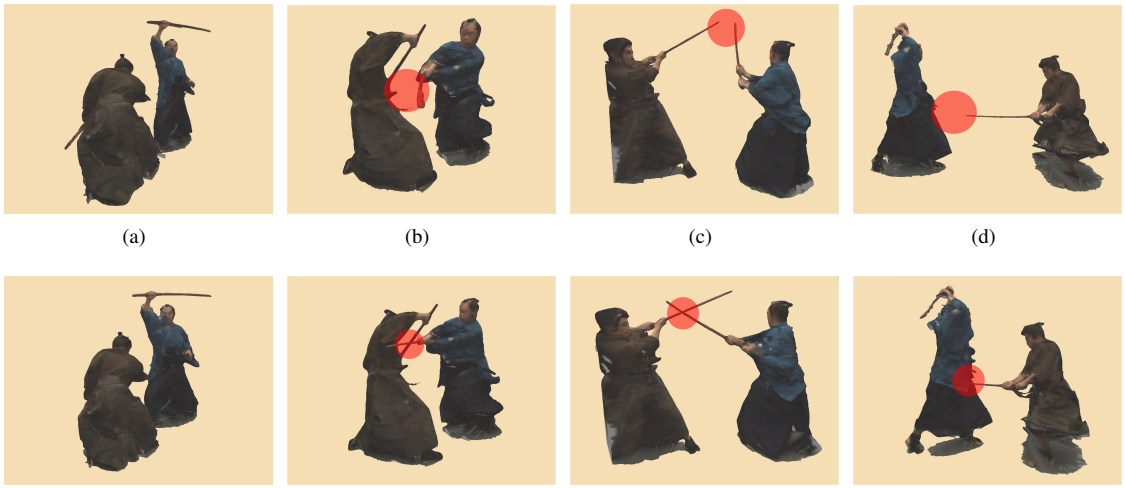
It can be seen in these figures that the global locations of the trajectories have changed distinctively, while the local shapes of the trajectories before and after editing still look similar to each other, meaning that the action dynamics in the original data have been kept well in the editing process.



(a) (b) (c) (d)

(e) (f) (g) (h)

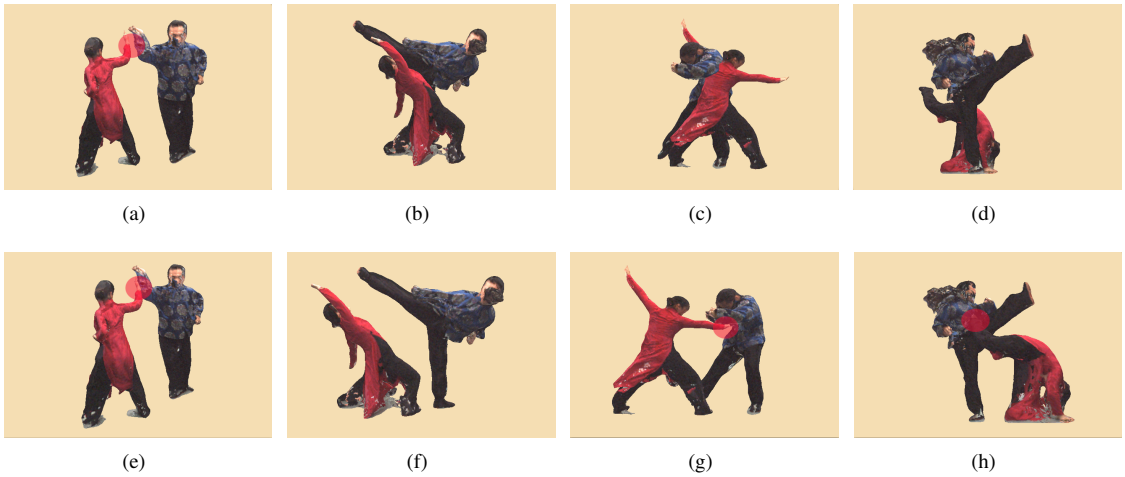
Fig. 22 Spatiotemporal editing results for Handshaking and Highfive data. The red circles are superposed to highlight the editing results.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 23 Spatiotemporal editing results for Chambara data. Top: Editing results of the baseline method. Bottom: Editing results of the proposed method. The red circles are superposed to highlight the editing results.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 24 Spatiotemporal editing results for Kungfu data. Top: Editing results of the baseline method. Bottom: Editing results of the proposed method. The red circles are superposed to highlight the editing results.

Table 4 shows a comparison between the baseline data and the editing result, considering seven components: the average difference of position, the mean-squared-difference (MSD) of position, the average translation velocity (baseline/proposed method),

Table 4 Quantitative evaluation of the editing results.

Data	Avg. DOP	MSD(DOP)	Avg. TV	Avg. DOTV
Cham. A	257.01	69457	15.86 / 17.15	1.29
Cham. B	263.10	73582	17.09 / 16.22	-0.96
Kung. A	241.75	62408	28.53 / 30.46	1.93
Kung. B	237.50	58317	21.10 / 22.98	1.88
Data	MSD(DOTV)	Avg. RV	Avg. DORV	MSD(DORV)
Cham. A	15.76	2.63	0.20	0.09
Cham. B	8.58	3.22	0.24	0.12
Kung. A	7.47	4.09	0.18	0.12
Kung. B	6.48	5.53	0.21	0.16

the average difference of translation velocity, the mean-squared-difference of translation velocity and angular velocity, the average rotation speed (baseline/proposed method), the average difference of rotation speed and the mean-squared-difference of rotation speed. Here the mean-squared-difference is computed by:

$$\mathrm{MSD} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2, \tag{38}$$

where N is the total frame number of the data, $\hat{Y}$ is a vector of the editing results by the proposed method, and Y is a vector of the results by the baseline method. The units for positions, translation velocities and rotation speeds are mm, mm/frame and degree/frame.

Table 4 shows that after editing, the spatial positions of the data have been changed drastically, while the changes on translation velocity and rotation speed are relatively small, meaning that the action dynamics of the original data have been preserved well.

5. Conclusions

In this paper we presented a novel action editing framework that synthesizes spatiotemporally synchronized multi-party interaction 3D video scenes from separately captured data. To realize that we propose the idea of Action History Volume and model the synchronization among multiple objects' body actions and gaze actions into spatiotemporal constraints that describe the multi-party interaction event. A three-step processing scheme, which includes (1) data segmentation, (2) intra-key segment editing and (3) inter-key segment optimization, has been developed to perform effective multi-party interaction synthesis work. Experiment results have proved the effectiveness of the proposed method in this paper.

Future work will seek to introduce the skeletal model based editing and combine it with the proposed framework, which will promisingly raise the flexibility of the intra-key segment editing process, and make the entire system become more robust against the original data with large degree of unsynchronization.

Acknowledgments

This study was supported by JSPS Ayame project "Visual Gesture Perception".

References

[1] J. Starck, A. Hilton. Surface capture for performancebased animation. IEEE Computer Graphics and Applications, 27(3):21C31, 2007.

[2] D. Vlasic, I. Baran, W. Matusik, and J. Popovic. Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph., 27(3):1C9, 2008

[3] T. Matsuyama, X.J. Wu, T. Takai, and T. Wada. Real-Time Dynamic 3D Object Shape Reconstruction and High-FidelityTexture Mapping for 3D Video, IEEE Trans. on Circuits and Systems for Video Technology, Vol.CSVT-14, No.3, pp.357-369, 2004.3.

[4] T. Matsuyama, S. Nobuhara, T. Takai, and T. Tung. 3D Video and Its Applications, Springer, 2012.6.

[5] T. Matsuyama, X.J. Wu, T. Takai, and S. Nobuhara. Real-Time 3D Shape Reconstruction, Dynamic 3D Mesh Deformation, and High Fidelity Visualization for 3D Video. International Journal on Computer Vision and Image Understanding,Vol.96, No.3,pp.393-434, 2004.12.

[6] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video.

[7] J. Starck and A. Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. IEEE International Conference on Computer Vision (ICCV), pages 1387C1394, 2005.

[8] K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In Proceedings of the Tenth European Conference on Computer Vision, volume Part II of LNCS, pages 30C43, Marseille, France, October 2008. Springer-Verlag.

[9] J. Starck and A. Hilton. Correspondence labelling for widetimeframe free-form surface matching. Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1C8, Oct. 2007.

[10] C. Cagniart, E. Boyer, and S. Ilic. Free-Form Mesh Tracking: a Patch-Based Approach. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10), Jun 2010, San Francisco, United States.

[11] P. Huang, A. Hilton, and J. Starck. Human motion synthesis from 3d video. In CVPR, 2009.

[12] J. Starck, G. Miller, and A. Hilton, Video-Based Character Animation, Proc. ACM Symp. Computer Animation, 2005.

[13] D. Casas, M. Tejera, J. Guillemaut, A. Hilton. Interactive Animation of 4D Performance Capture, Visualization and Computer Graphics, IEEE Transactions on Circuits and System for Video Technology, vol.19, no.5, pp.762,773, May 2013

[14] Q. Shi, S. Nobuhara, and T. Matsuyama. 3D Face Reconstruction and Gaze Estimation from Multi-view Video using Symmetry Prior. IPSJ Transactions on Computer Vision and Applications, Vol.4, pp.149-160, 2012.10.1.

[15] Q. Shi, S. Nobuhara, and T. Matsuyama. Augmented Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes, 3DV 2013, Washington, Seattle, USA, 2013.06.30.

[16] Q. Shi, S. Nobuhara, and T. Matsuyama. Augmented Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes. IEEE Transactions on Circuits and Systems for Video Technology, vol.25, no.1, pp.63-76, 2015.1.

[17] D. Weinland, R. Ronfard, and E. Boyer. Free view-point action recognition using motion history volumes. COMPUTER VISION AND IMAGE UNDERSTANDING (2006).

[18] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM
24, 381C395 (1981)

[19] Shohei Nobuhara, Yuta Kimura and Takashi Matsuyama: Object-Oriented Color Calibration of Multi-viewpoint Cameras in Sparse and Convergent Arrangement. IPSJ Transactions on Computer Vision and Applications(2010), Vol. 2, pp.132-144.

[20] Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1C8 (2007)

[21] Vogiatzis, G., Torr, P.H.S., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 391C398 (2005)

[22] Shohei Nobuhara, Takashi Matsuyama: Deformable Mesh Model for Complex Multi-Object 3D Motion Estimation from Multi-Viewpoint Video, 3rd International Symposium on 3D Data Processing. Visualiztion and Transmission (2006).pp.264-271.

[23] Felzenszwalb, P., Huttenlocher, D.: Efficient belief propagation for early vision. International Journal of Computer Vision 70, 41C54 (2006)

[24] Tony Tung, Shohei Nobuhara and Takashi Matsuyama: Simultaneous super-resolution and 3D video using graph-cuts. 26th IEEE Conference on Computer Vision and Pattern Recognition (2008), pp.1-8.

[25] Cootes, T.F. and Edwards, G.J. and Taylor, C.J: Active appearance models. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6), 681-685(2001).

[26] Tsuyoshi Kawaguchi, Mohamed Rizon and Daisuke Hidaka: Detection of eyes from human faces by Hough transform and separability filter. Electronics and Communications in Japan(2005), Vol.88(5),pp.29-39.