A free-viewpoint application has been developed that yields an immersive user experience. One of the simple free-viewpoint approaches called “billboard methods” is suitable for displaying a synthesized 3D view in a mobile device, but it suffers from the limitation that a billboard should be positioned in only one position in the world. This fact gives users an unacceptable impression in the case where an object being shot is situated at multiple points. To solve this problem, we propose optimal deformation of the billboard. The deformation is designed as a mapping of grid points in the input billboard silhouette to produce an optimal silhouette from an accurate voxel model of the object. We formulate and solve this procedure as a nonlinear optimization problem based on a grid-point constraint and some a priori information. Our results show that the proposed method generates a synthesized virtual image having a natural appearance and better objective score in terms of the silhouette and structural similarity.

key words: free-viewpoint system, billboard model, voxel model, image deformation, convex optimization

1. Introduction

In recent years, free-viewpoint navigation systems from a multi camera environment [1]–[3] have been among the hottest topics in computer vision, and they are seen with increasing frequency. In the navigation system, users can select their viewpoint freely (not limited to actual camera positions), and the selected scene is synthesized by using multi camera videos and several additional items of information. This makes the free-viewpoint system very useful for improving the user’s understanding of a scene, and creates an immersive and ultra-realistic user experience, especially when viewing sports.

Naturally, the demand for free-viewpoint in mobile devices (low power devices) has also increased. Therefore, as our ultimate goal, the free-viewpoint should be streamed in real-time from the video shooting to rendering. However, our first goal in this paper is only to render the viewpoint selection to rendering of free-viewpoint significantly affects the user experience, it is desirable to render free-viewpoint using local contents in devices rather than in the cloud server. This means that the contents need to be streamed to the user’s mobile devices using a wireless network comfortably. In other words, the data size of the contents should be reduced and the rendering of the streamed local contents should work in real-time in low power devices.

One of the most well-known free-viewpoint synthesis approaches is called the “model-based method”, which constructs computer graphics models of people, objects and spaces by using information from sparsely arranged cameras. Using this feature, the model-based method establishes both a good quality scene and a flexible viewpoint, in contrast to several other approaches [4] that suffer from a limited range of virtual viewpoints.

Some well-known conventional model-based methods, the “visual hull methods [5]–[9]”, generate a highly accurate 3D voxel model of the scene by using information from multiple cameras. However, with these methods, viewing the model in a mobile device incurs a high computational cost and they are not suitable for streaming because they have too much fine detail to display and the amount of data is generally huge.

As a similar approach to the visual hull, “multi-view stereo methods [10], [11]” construct 3D space as a point cloud by using the correspondence of feature points given by multiple camera images. However, these suffer from the same limitations that occur in the visual hull methods, and require a lot of cameras compared with other model-based methods including the visual hull.

On the other hand, methods based on simple 3D models like the “billboard model method[12]–[15]”, have been proposed. The method extracts the texture of a foreground object and uses it as a billboard model (a simple 3D plane), placing the plane model perpendicularly on a 3D ground at one position and rotating it to face towards virtual viewpoints. From this simple framework, the method has the advantage reducing the computational cost of display and the amount of data while maintaining the quality of the model texture. Conversely, the method has difficulty in detecting and representing where the billboard model is in 3D space. To be specific, in some cases where an object (a player) in an image touches the ground at multiple points, there will be some uncertainty in calculating the exact contact point and it is difficult to represent multiple touch points by just one extracted texture (Fig. 1). This ambiguity causes a fatal unnatural feeling in users. We call this the “touch points
problem”.

The “microfacet-billboarding method[16]–[18]” has been proposed as a hybrid of the visual hull and billboard methods. This method represents a person in the foreground as an aggregate of small billboards that is constructed from a finely subdivided image of the person and his actual position by using the visual hull technique and voxels. This can solve the unnatural touch points problem that affects the billboard model method. Nevertheless, microfacet-billboarding causes undesirable artifacts along the boundaries in the synthesized model due to the large number of small rectangles of which it is comprised.

From the above discussion, we propose a hybrid billboard method† that uses 3D voxel as support information to solve the touch points problem, while at the same time preserving the advantages of the billboard. Our method is designed to generate a standard (single plane) billboard model, but appropriately deformed for different virtual viewpoints. The deformation is represented as a mapping of grid points that are obtained by separating the elements of the initial foreground texture. We determine the optimal mapping by using the silhouette of the voxels because we assume that the voxel model can generate the optimal human silhouette from any virtual viewpoint. The optimal mapping of grid points can be obtained using a conventional constrained nonlinear optimization tool, the projected gradient method[20], and the billboard will then be generated according to the points using texture mapping. Our experimental results show that our optimal deformation can generate a billboard representation having a natural appearance while also solving the touch points problem.

2. Related Work

Almost all the conventional model-based methods are composed of five basic functions, “camera calibration”, “video time synchronization”, “silhouette mask extraction”, “model creation” and “synthesizing in viewer”. Camera calibration is a function to calculate a camera parameter that includes information of optical properties and the position of the camera in the world, and the video time synchronization function simply synchronizes the multiple cameras in absolute time. These shooting settings are predefined in advance. In addition, silhouette mask extraction extracts a silhouette of the player as a binary image and the image becomes tentative data to calculate the shape of the player. The results of these three functions greatly affect the quality of the final synthesis, but do not contribute to reducing the amount of data and to shape representation to solve the touch points problem. Taking these features into account, in the following, we summarize the differences among conventional methods from the viewpoint of how to create a model and synthesize it.

Using one camera parameter, we can project a corresponding camera image into 3D space, but it is impossible to know the depth (shape) of a player along the camera optical axis. To overcome this difficulty, the conventional visual hull methods[5]–[9] calculate depth by using the intersection of the silhouette of the player in each camera. This means the methods output a 3D model with actual shape in 3D space. In particular, first, the 3D virtual space would be separated along the x-y-z axis at periodic intervals and the separated minimum cube called as “voxel”. Then, projecting the silhouette of the player to each voxel, the method can make a decision as to whether the corner point of the voxel is included in the silhouette. The product set of the decision by multiple cameras makes the shape reliable and outputs the outline of the player in the 3D space. Finally, by using a conventional data conversion method, referred to as “marching cube[21]”, the set of the reliable voxels is converted into a model represented via a triangle unit, polygon, that is placed on the surface of the subject.

There is a trend for, some conventional methods to focus on refining the shape of obtained visual hulls. For example, one of the latest methods[22] can represent the shape of occluding contours that are inside the object’s silhouettes based on the assumption that the target being recorded is non-Lambertian object. Another of the latest methods[9] has proposed a new 3D primitive, Convex Bricks, to accurately express the object’s shape and reduce the number of polygons. However, even with these methods, the fine 3D model has a huge amount of data (in the results of[9], over 100,000 vertices) and incurs a high computational cost in the viewer compared with the billboard model methods.

Beingsimilar to the visual hull methods, the multi-view stereo methods[10], [11] also calculate the depth of the player as a point cloud by using correspondences of feature points extracted from input camera images. To create the point cloud from many feature points, these methods require specific equipment, some a priori information of the shooting environment, or a lot of camera images. For example, in the latest method[11], several polarizers are used for revealing surface normal information of the target object and 36 cameras are used for rendering one car. In terms of commercial use, the software[23] also uses a similar approach and it requires over 100 cameras for one human as a general case. Considering versatile use for several sports, the visual hull approach is more suitable for obtaining the exact depth of the player.

†A part of this method has been published in [19].
Billboard model methods [12]–[15] easily solve the data amount and the computational cost problems because the player is represented as only one polygon plane (billboard) and the texture of the player is just projected on the plane.

According to subjective experiments in [24], the billboard model approach is more robust to noise than the other visual hull methods, but the main problem in the conventional billboard model method is where the contact point between the billboard and the ground in 3D space is. To be specific, the contact point is calculated using simple projection from the player coordinates in the 2D image to the 3D field, and in the methods [13], [14], the bottom line in the player boundary box is used as a reference for the player coordinates. Then, in the “synthesis” function, when the virtual viewpoint is selected, the billboard would be placed on the ground at the contact point perpendicularly, and rotated in a horizontal direction toward the virtual viewpoint (Fig. 2). This means, that in the billboard method, it is not necessary to know the depth of the player surface.

However, if a player in a camera image touches the ground at multiple points (e.g. hands and feet) as shown in Fig. 1 (a), all the touch points can be a candidate for the contact point, and the contact point cannot be uniquely determined because there is no superiority among the touch points in terms of 3D space. Consequently, as shown in Fig. 1 (b), a synthesized virtual scene in the vertical direction shows an unnatural appearance as if the person were flying in the air.

One of the billboard methods [15] implicitly tackles this “touch points problem”. It separates billboard into several body parts and places them according to the position of the human skeleton. However, it assumes user manual input to estimate human skeleton frame by frame, and the manual task is generally exhaustive for the user. Therefore, none of the conventional billboard model methods have solved this problem.

The microfacet-billboard method is an approach that establishes both the shape representation of a player and the data amount reduction. Conventional microfacet-billboard methods [16]–[18] represent the player as aggregation of some units like the visual hull method, and the unit is not a voxel but a billboard. The methods calculate the position of a player’s voxel in the same manner as the visual hull method, and puts a part of a texture at the corresponding voxel position as a small unit billboard. Each unit billboard stands perpendicularly and rotates toward the virtual viewpoint in the same way as the billboard model method. The aggregation of unit billboards can represent the shape of a player.

In contrast, since the unit billboard is a rectangle texture, however, the aggregation causes a fatal artifact along the boundary of the unit billboard. In other words, it looks like a mosaic as if the unit billboard becomes a single pixel. To avoid such an undesirable artifact, the micro-facet billboard method has to make the unit billboard a smaller one at the expense of sacrificing the amount by which data are reduced.

The latest method [18] realizes how to make object model data in real-time by using a coarse-to-fine approach and avoids artifacts arising between foreground and background. However, the method cannot avoid artifacts along the boundary of the unit billboard and it takes time to visualize even when using a standard desktop PC. This fact implies that the method is not suitable for our purpose, namely, use in a mobile device.

3. Proposed Method

3.1 Overview

We assume that the human target (player) is surrounded by $K$ cameras (in our experiments, $K = 8$), each camera position is fixed and the camera parameters should be known. The time synchronization between all cameras is adjusted in advance. Throughout this paper, we discuss how to synthesize a billboard from a virtual viewpoint in only one frame (the time $t$), but our method can easily be applied to all video sequences because the procedure is independent frame by frame. To simplify the problem, we assume a single target player, but our algorithm can easily be extended to deal with multiple players individually. As we discussed in Sect. 2, the conventional billboard method [12]–[15] includes the “silhouette extraction”, “model (billboard information) creation” and “synthesizing (locating the billboard on its 3D ground) in the viewer”.

Let $S$ be an input image and $M$ be the extracted binary silhouette mask of the player (camera silhouette) obtained by “silhouette mask extraction”, and the method simply constructs a billboard (plane) from the texture of the nearest camera image extracted by $M$. This fact means the viewer only needs to know the $S$ and $M$ to generate a billboard. Then the view of the billboard model is synthesized by putting the billboard on its 3D ground according to the 3D position that is estimated to be the projection of the bottom point of the billboard.

Our method is fundamentally based on the conventional billboard scheme, with the key difference being the “billboard information creation” part. A flowchart of our algorithm is shown in Fig. 3. First, we obtain camera sil-
houettes in multiple cameras using the same approach as the conventional billboard method. Then, we can generate a voxel model via the conventional visual hull method [5]. When we predefine a virtual viewpoint, we can obtain the optimal silhouette of the player $M_{opt}$ by projecting the voxels onto the camera image plane and choose the silhouette of the nearest camera as the target camera silhouette $M$. In our “billboard information creation” part, we consider $M$ as a set of small regions (patches) separated by a constant interval grid. By obtaining $M_{opt}$ and the grid, the correspondences between $M$ and $M_{opt}$ are calculated by block matching each patch on the edge of the silhouette. Using these correspondences and several a priori assumptions, we can formulate and solve the optimization problem related to the coordinate mapping of the patches.

In our viewer, to generate a deformed billboard, we only have to know the mapping, camera image, silhouette $M$ and the viewpoint selected by the user. Although the selected viewpoint might differ from the predefined one in the actual case, the mapping can be easily calculated by interpolating mappings corresponding to several near predefined viewpoints. For the sake of simplicity, hereinafter, we assume the selected viewpoint is the same as the predefined viewpoint.

According to this flow, the key task is to find the mapping of the coordinates that correspond to a selected virtual viewpoint. In this paper, we discuss a case in which the selected viewpoint differs from the camera position only on the vertical axis, because the touch points problem arises only if the viewpoint moves vertically. If the viewpoint moves horizontally, our method simply rotates the generated billboard, just as the conventional method does.

3.2 Voxel Model Construction and Optimal Silhouette Projection

To find the optimal mapping of patches, we refer to the optimal silhouette $M_{opt}$ generated by the voxel model. The voxel model provides a 3D outline of the player, and is constructed by the visual hull technique using the intersection of the mask of the player in voxel (3D) space [6]. Obtaining the 3D model, we can find the actual shape of the player from any virtual viewpoint. Therefore, by projecting the model back onto the camera image plane, we can get the optimal binary silhouette $M_{opt}$ of the player from a virtual viewpoint (Fig. 4 (b)).
of coordinates for the patch $p = \{p_1^T, p_2^T, \ldots, p_m^T\} \in \mathbb{R}^{3m}$ by separating the $M$ at periodic intervals on the grid (see Fig. 4 (a)), where $m$ is the number of patches in a player. Please note that some $p$ share the same corner point because of the structure of the grid, and we call such a set of points a “common vertex set”.

Next, we calculate the rough correspondence between $M$ and $M_{opt}$. Both are specific player regions in the binary image corresponding to each viewpoint. Since the color image is too sensitive to permit finding the correspondence of each patch by block matching, and the voxel model might generate a blurred color image even if the cameras deliver high resolution texture, we use only patches on the edge of each binary camera silhouette (edge patch).

An edge patch is defined by the proportion of silhouette pixels in the patch, as in

$$\tau_i = \sum_{p_{pix} \in M(p)} \frac{p_{pix}}{N(p)},$$

where the functions $M()$ and $N()$ return a set of binary pixel values and the number of pixels in each image patch, respectively.

Furthermore, we find matched edge patches for the optimal silhouette by block matching as follows:

$$g_i^* \in \arg\min_{g_i \in W_{M_{opt}}(q_i)} \text{MAD}(M_{opt}(g_i), M(q_i)),$$

where the function $W_{M_{opt}}()$ returns a set of patches in $M_{opt}$ included in a search window, the center of which is $q_i$’s center position (the window size is defined as $W_{size}$), and the size of returned patch $g_i$ is the same as $q_i$. Therefore, $g_i$ means a patch which should be compared with $q_i$. In addition, the function $\text{MAD}()$ calculates the Mean Absolute Difference (MAD) between $M_{opt}(g_i)$ and $M(q_i)$ by block matching (the $M_{opt}$ returns a binary pixel value in the same manner as $M()$). Finally, we obtain a set $g = \{g_1^*, g_2^*, \ldots, g_k^* : k$ is the number of edge patches $q_i\}$, as the matched edge patches in $M_{opt}$. It should be noted that the corner point in a common vertex set has different coordinates in this state, as in Fig. 4 (c).

### 3.4 Optimal Grid Mapping Calculation

To find the optimal mapping of $p$, we define a constrained energy function as follows:

$$x^* = \arg\min_{x \in \mathbb{R}^m} \frac{L_1}{2} ||Ex - Ep||_2^2 + \frac{L_2}{2} ||Hx - g||_2^2 \text{ s.t. } x \in C,$$

where the function $||x||_2$ is $L_2$ norm for a vector $x \in \mathbb{R}^n$.

The first term is a “structure coherence term” that works to preserve the entire structure of $p$, and the matrix $E$ is designed to calculate the distance between the corner points of $p$. For example, if $p$ has only one patch $p_1 = \{p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}\}^T$ and these points are connected as shown Fig. 4 (c), the matrix $E$ is designed as follows:

$$E = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$ (4)

Please note this example describes only one ($x$) dimension of $p_1$ for the sake of brevity, but the matrix is simply extended in the case of two dimensions ($x, y$). The second term is an “edge patch term”, and the matrix $H$ just extracts the coordinates of $x$ corresponding to the edge patches $g_i$. In other words, the matrix $H$ shortens the length of $x$ from $8m$ to $8k$ ignoring non-edge patch coordinates in $x$. This term has the effect of making the camera silhouette similar to the optimal one. The set $C$ is a set of vectors $x$ in which the corner points in a common vertex set have the same coordinates, so this constraint guarantees connectivity between patches.

To solve Eq. (3), we use a conventional constrained nonlinear optimization tool, the projected gradient method [20]. The procedure is described in Algorithm 1.

```
Algorithm 1 Specialized projected gradient method for solving Eq. (3)
1: Set $t = 0$, and choose $\gamma > 0$, $x_0 \in \mathbb{R}^{3m}$.
2: repeat
3: \quad $x_{t+1} = \text{proj}_C(x_t - \gamma(\lambda_1 E^T E(x - p) + \lambda_2 H^T Hx - g))$
4: \quad $t \leftarrow t + 1$
5: until some stopping criterion is satisfied.
```

```
Please note this example describes only one ($x$) dimension of $p_1$, for the sake of brevity, but the matrix is simply extended in the case of two dimensions ($x, y$). The second term is an “edge patch term”, and the matrix $H$ just extracts the coordinates of $x$ corresponding to the edge patches $g_i$. In other words, the matrix $H$ shortens the length of $x$ from $8m$ to $8k$ ignoring non-edge patch coordinates in $x$. This term has the effect of making the camera silhouette similar to the optimal one. The set $C$ is a set of vectors $x$ in which the corner points in a common vertex set have the same coordinates, so this constraint guarantees connectivity between patches.

To solve Eq. (3), we use a conventional constrained nonlinear optimization tool, the projected gradient method [20].

The procedure is described in Algorithm 1.

```
Algorithm 1 Specialized projected gradient method for solving Eq. (3)
1: Set $t = 0$, and choose $\gamma > 0$, $x_0 \in \mathbb{R}^{3m}$.
2: repeat
3: \quad $x_{t+1} = \text{proj}_C(x_t - \gamma(\lambda_1 E^T E(x - p) + \lambda_2 H^T Hx - g))$
4: \quad $t \leftarrow t + 1$
5: until some stopping criterion is satisfied.
```

All the optimal mappings can be calculated before transmission by the network in advance (Fig. 3). If it is difficult to calculate the mapping from all viewpoints, it can be obtained by linear interpolation from discrete sample mapping. Therefore, the computational cost to display these billboards is almost the same as in the conventional billboard method because the added overhead is the cost of texture mapping each patch. Moreover, the total amount of data in the generated result is also close to that in the conventional billboard method because the only additional information is the mapping data of each grid point from all viewpoints.

### 3.5 Iterative Matching

So far, we have described in detail the basic framework (Fig. 3) which relies on the assumption that the input silhouette is similar to the silhouette obtained by the voxel from the predefined virtual viewpoint. However, in some cases, reliable optimal mapping cannot be directly calculated using an optimal silhouette that is completely different from
the input image. In other words, this is caused by the predefined viewpoint being too far from the camera position. In this case, since the edge patch matching does not result in desirable matching, the error causes a fatal artifact. To solve this problem, we have proposed an optional procedure, called “iterative matching”.

The outline of the iterative matching is shown as Fig. 5. Please note again that the function is optional, and it would only be needed if the predefined viewpoint is distant from the camera position. The key point of this matching is that we break down a large problem into a small problem that can be solved by the basic framework. Specifically, let the input camera silhouette and the given optimal silhouette be \( M \) and \( M_{opt} \), which is the same as the basic framework, and the angle formed by two optical axes corresponding to the input camera and the virtual camera be \( \theta \). Then, we can divide the angle \( \theta \) into \( A \)-number of angles, \( \theta_n (n \in \{1, 2, 3, \ldots, A\}) \) (see Fig. 6), and the ascending order of the angle is from the input camera to the selected virtual viewpoint. This means the silhouette corresponding to \( \theta_1 \) is the nearest and the one that most closely resembles the silhouette between the input camera and the virtual viewpoint. From the virtual viewpoint with \( \theta_n \), we can obtain a certain silhouette and consider it to be the tentative optimal silhouette \( M_{opt,n} \). By using the same method described in the section above, the optimal mapping is given as \( x^*_n \). In addition, we can consider a deformed silhouette and a deformed player by \( x^*_n \) is the next camera silhouette and the next tentative input data. Continuing this procedure by generating a deformed player from the selected virtual viewpoint, we can obtain the final result of the above condition.

4. Experiments

4.1 Comparison of Output Images

We examined the performance of the proposed method by comparing it to the conventional billboard method (CB) [14] and the conventional microfacet-billboard method (CMB) [16]. The results with no iterative matching are shown in Fig. 7. The background images without the subject camera image are synthesized from the same virtual viewpoint, and the ground truth is given by using the standard visual hull method (GT) [6]. For generating the visual hull, we define an interval length between each voxel as 0.5 [cm] to express fine detail, and for the microfacet-billboard method, the size of the microfacet-billboard is set as \( 5 \times 5 \) [cm] to make use of the advantage of the method. In this experiment, we use full HD input images shot by 8 circularly positioned cameras and the size of the patch is \( 50 \times 50 \) [pixels]. The parameters of our method are heuristically defined as \( t_{h1} = 0.5, t_{h2} = 0.95, \mu = 1.0, \lambda_1 = 0.5, \lambda_2 = 0.5, W_{size} = 50 \) [pixels].

Our method can generate views that are better able to represent the ground truth than the conventional methods, confirming that our proposed method solves the “touch points problem”. For example, in scene 1-1, the conventional billboard method generates a view that makes it appear as if he is standing only on his hands (his feet are over the horizontal line), but our method can deform the shape of the man in a way that makes it similar to the outline of the ground truth and his feet in our synthesis appear to be under the horizontal line. However, the model can still be considered a billboard. On the other hand, the microfacet-billboard method represents the outline of the player well, but the boundary of the player and texture (e. g. boundary between red shirt and black shorts) has jaggy noise caused by the quadrangle shape of the microfacet-billboard.

In scene 2, the shape of the woman’s head in our method is slightly different from the ground truth. This is
because the 3D voxel is represented only by 2D information and the balancing of the weights, $\lambda_1$ and $\lambda_2$. Even in this case, compared with the conventional method, it seems our method can preserve the outline of the ground truth and solve the touch points problem and avoid the jaggy noise.

In scene 3-1, the position of the right player in CB is different from the ground truth. This is because the mask of the player is connected to the blue cone and the estimated position is wrong. In our method, although the blue cone is deformed unnaturally, we can see the whole shape structure of the player is preserved and the position is almost correct. Since the virtual viewpoint is more distant from players compared with in scene 1 and scene 2, it seems the result of CMB has fewer artifacts in this case.

Furthermore, the performance of the iterative matching is shown in Fig. 8. We intentionally select a virtual viewpoint that is distant from the camera position to make the drawback of the viewpoint clear. In addition, the parameter, $W_{size}$, is 20 [pixels] different from the above experiment because the optimal silhouette in each iteration is similar to the deformed silhouette. The other parameters and settings are the same as the example above and the number of iterations is 20. From the results, we can distinguish the effect of iterative matching. Compared with the proposed method with no iterative matching in Fig. 8, the results with matching are characterized by high quality and less artifact texture. Of course, since the conventional billboard method simply extracts the texture, it has no artifact in the player texture but is potentially affected by the touch point problem (see the ball position in Fig. 8).

4.2 Objective Evaluation

To examine the performance of our method, we carried out two objective evaluations using the results shown in the Figs. 7, 8 and the results obtained by randomly selecting virtual viewpoints. In the first evaluation, we define the silhouette similarity between the ground truth and the product of
each method as follows:

\[
\kappa = 1 - \sum_{(x,y) \in |X \cup M_{\text{opt}}|} \frac{|X(x,y) - M_{\text{opt}}(x,y)|}{|X(x,y)|}, \tag{5}
\]

where the matrix \(X\) is an input binary silhouette of the comparison, namely, the camera silhouette produced by the conventional method and the silhouette output by the proposed method, and the \((x, y)\) returns a pixel value in each image region (if the pixel is not detected, the value is defined as 0).

In the case where the \(\kappa\) is 1, the input silhouette is identical to the optimal one. The score \(\kappa\) is shown in Table 1, which shows, according to an objective evaluation, that the silhouette generated by our method is similar to the ground truth compared with CB. Since CMB refers to the voxel model, of course, it generates a similar silhouette to the ground truth but our method gets higher score in most cases. It means our method represents the silhouette well.

In the second evaluation, we examine the quality of the synthesized image. The most important feature of our method is its ability to preserve both the player’s shape and structure in the synthesis process. To verify this feature, we use a well-known structure evaluation method related to human perception, Mean Structural SIMilarity (MSSIM) [25]. The input synthesized image for this evaluation is also the same as that shown in Table 1 and we calculate MSSIM for the luminance between the ground truth (the voxel synthesis) and each method. A higher value is better in MSSIM, and the range is from 0 to 1. The results of MSSIM are shown in Table 2 and the score shows our method can preserve not only the silhouette of the player but also the structure of the synthesized image. Differing from the silhouette similarity, only the score of CMB in the far shooting environment (scene 3-1, 3-2) are superior to our method because the artifacts along the boundary are not so noticeable.

### Table 1. Comparison by silhouette similarity (“Ave.” means an average score of 10 randomly selecting virtual viewpoints).

|      | CB [14] | CMB [6] | Ours with no iter. match | Ours with iter. match |
|------|---------|---------|--------------------------|----------------------|
| Scene1 1-1 | 0.84    | 0.96    | 0.95                     | 0.96                 |
|       1-2 | 0.80    | 0.94    | 0.94                     | 0.97                 |
|       Ave. | 0.83    | 0.96    | 0.94                     | 0.97                 |
| Scene2 3-1 | 0.90    | 0.93    | 0.96                     | 0.97                 |
|       Ave. | 0.87    | 0.95    | 0.94                     | 0.96                 |
|       3-2 | 0.92    | 0.98    | 0.97                     | 0.98                 |
|       Ave. | 0.90    | 0.95    | 0.96                     | 0.97                 |
| Scene3 1-1 | 0.76    | 0.83    | 0.80                     | 0.81                 |
|       3-2 | 0.92    | 0.98    | 0.97                     | 0.98                 |
|       Ave. | 0.90    | 0.95    | 0.96                     | 0.97                 |

### Table 2. Comparison by MSSIM (“Ave.” means an average score of 10 randomly selecting virtual viewpoints).

|      | CB [14] | CMB [6] | Ours with no iter. match | Ours with iter. match |
|------|---------|---------|--------------------------|----------------------|
| Scene1 1-1 | 0.69    | 0.80    | 0.85                     | 0.86                 |
|       1-2 | 0.71    | 0.82    | 0.84                     | 0.87                 |
|       Ave. | 0.70    | 0.81    | 0.84                     | 0.87                 |
| Scene2 3-1 | 0.76    | 0.87    | 0.88                     | 0.89                 |
|       Ave. | 0.75    | 0.83    | 0.86                     | 0.87                 |
|       3-2 | 0.88    | 0.93    | 0.90                     | 0.91                 |
|       Ave. | 0.85    | 0.91    | 0.88                     | 0.89                 |

### Table 3. Specification of devices in this experiment.

| model | desktop PC (device 1) | Android (device 2) |
|-------|------------------------|--------------------|
| OS    | Windows 10 64bit       | Android 6.0.1      |
| CPU   | Intel Core i7-4790     | Qualcomm Snapdragon 810 |
|       | CPU (3.6GHz)           | Octa-core          |
| GPU   | Nvidia GeForce GTX 760 | on CPU             |
| RAM   | 8GB                    | 3GB                |

### Table 4. Average processing time of generating model data [sec/frame].

|      | CB [14] | CMB [6] | Ours with no iter. match | Ours with iter. match |
|------|---------|---------|--------------------------|----------------------|
| GT   | 8.207   | 10.52   | 972.8                    | 1106                 |

### 4.3 Computational Cost in Server

As described in Introduction, we don’t strictly care about the computational cost in the server for VOD services. However, to clarify the limitation of our method, we discuss the computational cost of data creation in the server side. We use a desktop PC (device 1) with a specification shown in Table 3. The source code of each method is implemented by C++ and the procedure is executed by GPU.

The average processing time for generating the data of one frame in Figs. 7, 8 is shown in Table 4. These do not include the time by mask extraction because the procedure has to be performed among all the methods. It means there is nothing to do for the CMB in terms of data generation. For our method, we predefined 15 mappings along the vertical axis for each camera. Totally, there are 120 (15 × 8 cameras) predefined viewpoints for which we should generate corresponding mappings, for all the virtual viewpoints.
of the scene.

Since we require many somewhat predefined viewpoints and use the iterative solution to find the optimal mapping, our method takes much time compared with other conventional methods. The time without iterative matching does not differ that much from the one with iterative matching because we can refer to the mapping of the nearest predefined viewpoint in the iterative matching. In addition, Table 5 shows the influence of the patch size on the time of our method. The time is dependent on the patch size because the $E$ and $H$ in the Eq. (3) are changed according to the size and the change affects the characteristics of convergence of the equation.

Depending on the situation, this calculation cost might become a drawback of our method. However, considering the realization of VOD services only, it may be acceptable because the data generation can be done in advance. Since if we can find the optimal parameters, the processing time will be reduced dramatically, then finding the optimal size of the patch automatically becomes a matter for future consideration.

### 4.4 Computational Cost in Viewer

In this section, we show the computational cost for rendering in the viewer. The same as the Sect. 4.3, we use a device 1 shown in Table 3 and some viewers implemented by C++ with OpenGL [26] to measure the time to render. To confirm the actual processing speed for rendering, we turn off vertical synchronization of the display, because if it is on, the rendering speed is limited by the refresh rate of the display. The average processing time and frames per second (fps) for rendering one virtual viewpoint in Figs. 7, 8 are shown in Table 6. Please note that the time is described in milliseconds order to clarify the difference. From this result, we can ascertain that the rendering speed of our method is almost the same as the speed of the conventional billboard method and faster than the other methods. In addition, we developed a viewer of our method that works in an android mobile device (device 2) shown in Table 3 and confirm the viewer can render over 30 fps, the same as the CB method.

### 4.5 Discussion

From a qualitative point of view, we discuss the influence of the parameters in our method. The parameters $\lambda_1$ and $\lambda_2$ balance the structure coherence and the edge patch term described in Sect. 3.4, respectively. The results according to the change of these parameters are shown in Fig. 9. The parameters are the same as the experiment in Fig. 7 scene 1-1. If we set the $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$, since the structure coherence is stronger, our method finds the optimal mapping to make a similar shape of the player of CB while deforming the shape to ground truth. On the other hand, the $\lambda_1 = 0.1$ and $\lambda_2 = 0.9$ works to make the optimal mapping achieve the silhouette of the ground truth, as if the feet drop down. Sometimes, this setting works well, but if the edge patch matching includes many errors, the result of this setting generates many artifacts as in Fig. 8 (e). In the case that the $\lambda_1 = 0.75$ and $\lambda_2 = 0.25$, the result shows the midpoint between Fig. 9 (b) and Fig. 9 (d). Another future work is how to find the best parameters, even though in the almost all case, the lambda in Figs. 7, 8 ($\lambda_1 = 0.5$ and $\lambda_2 = 0.5$) works well.

In terms of quality, in the experiments, our method outperformed the conventional method. However, of course, our method is dependent on the accuracy of the voxel model. Therefore, we show the results of our method in the case where the generation of the voxel fails and robustness is improved as shown in Fig. 10. Different from Fig. 7, the voxel result is generated by a mask threshold value that is too high in several cameras (except the nearest one from a virtual viewpoint) and it causes part of the left hand to be missing. Even in a case like this one, our method is not affected by the accuracy of other cameras and can recover the missing part and improve synthesis. This means our method is robust against some voxel artifacts because we use a simple silhouette feature, a connected grid, and formulate a structure coherence term in our optimization.

### Table 5

| the size of patches [pixels] | 40 × 40  | 50 × 50  | 75 × 75  | 100 × 100 |
|-----------------------------|---------|---------|---------|----------|
| the processing time [sec]   | 7620    | 972.5   | 44.00   | 1616     |

**Fig. 9** Comparisons of our method according to the $\lambda_1$ and $\lambda_2$.

### Table 6

| GT      | CB [14]    | CMB [6]   | ours    |
|---------|------------|-----------|---------|
| 11 (90.9 fps) | 1.9 (526 fps) | 2.4 (417 fps) | 1.9 (526 fps) |

**Table 6** Processing time of rendering virtual viewpoint [msec/frame].
5. Conclusion

We have proposed an optimal billboard deformation method to overcome the “touch points problem” that arises in the conventional billboard method. The key point of the proposed method is that a rough silhouette correspondence between the input and the optimal modification can be calculated by block matching of the patches on the edge of the silhouette. Additionally, based on the correspondence and some a priori information, we have formulated a constrained nonlinear optimization problem and solved it to generate a suitably deformed texture. Our experiments show that the proposed method can generate a silhouette similar to that obtained from the voxel model, but is more useful as it solves the touch points problem in a way that preserves the structure with fewer artifacts while retaining the advantages of billboard.

Considering the usecase in VOD services, the requirement of processing time in the server side is not so strict but our method has the limitation of time because of the iterative solver for the optimization problem. For more acceleration and automatic quality improvement of our method, we will develop the optimal parameter definition scheme as a future work. In terms of viewer, the experiment shows our method renders players as fast as that of the conventional billboard method.

We plan to investigate techniques for adaptive deformation that preserve specific features of the player, for instance by preserving faces, the goal being to improve the subjective quality of the deformed texture and to accelerate solving the nonlinear problem so that the method can find wide commercial use.

References

[1] M. Tanimoto, M.P. Tehrani, T. Fujii, and T. Yendo, “Free-viewpoint tv,” IEEE Signal Process. Mag., vol.28, no.1, pp.67–76, 2011.
[2] T. Kanade, P. Rander, and P.J. Narayanan, “Virtualized reality: Constructing virtual worlds from real scenes,” IEEE Multimedia Mag., vol.4, no.1, pp.34–47, 1997.
[3] R. Sitnik, W. Puech, R. Suenaga, K. Suzuki, T. Tezuka, M.P. Tehrani, K. Takahashi, and T. Fujii, “A practical implementation of free viewpoint video system for soccer games,” Proceedings of SPIE Electronic Imaging, 3D Image Processing, Measurement and Applications, 93930G, 2015.
[4] C. Lipski, C. Linz, K. Berger, A. Sellent, and M.A. Magnor, “Virtual video camera: Image-based viewpoint navigation through space and time,” Computer Graphics Forum, vol.29, no.8, pp.2555–2568, 2010.
[5] C. Buehler, W. Matusik, L. McMillan, and S. Gortler, “Creating and rendering image-based visual hulls,” Technical Report, Massachusetts Institute of Technology, 1999.
[6] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, and L. McMillan, “Image-based visual hulls,” Proceedings of Annual Conference on Computer Graphics and Interactive Techniques, pp.369–374, 2000.
[7] A. Ishikawa, M.P. Tehrani, S. Naito, S. Sakazawa, and A. Koike, “Free viewpoint video generation for walk-through experience using image-based rendering,” Proceedings of ACM International Conference on Multimedia, pp.1007–1008, 2008.
[8] Y. Furukawa and J. Ponce, “Carved visual hulls for image-based modeling,” International Journal of Computer Vision, vol.81, no.1, pp.53–67, 2009.
[9] V. Chari, A. Agrawal, Y. Taguchi, and S. Ramalingam, “Convex bricks: A new primitive for visual hull modeling and reconstruction,” Proceedings of IEEE International Conference on Robotics and Automation, pp.770–777, 2012.
[10] Y. Furukawa and C. Hernández, “Multi-view stereo: A tutorial,” Found. Trends. Comput. Graph. Vis., vol.9, no.1-2, pp.1–148, 2015.
[11] Z. Cui, J. Gu, B. Shi, P. Tao, and J. Kautz, “Polarmetric multi-view stereo,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.369–378, 2017.
[12] T. Koyama, I. Kitahara, and Y. Ohtu, “Live mixed-reality 3d video in soccer stadium,” Proceedings of IEEE/ACM International Symposium on Mixed and Augmented Reality, pp.178–186, 2003.
[13] K. Hayashi and H. Saito, “Synthesizing free-viewpoint images from multiple view videos in soccer stadium,” Proceedings of International Conference on Computer Graphics, Imaging and Visualisation, pp.220–225, 2006.
[14] Y. Yoshida and T. Kawamoto, “[demo] displaying free-viewpoint video with user controllable head mounted display demo,” Proceedings of IEEE International Symposium on Mixed and Augmented Reality, pp.389–390, 2014.
[15] M. Germann, A. Hornung, R. Keiser, R. Ziegler, S. Würmlin, and M. Gross, “Articulated billboards for video-based rendering,” Proceedings of Eurographics, Computer Graphics Forum, vol.29, no.2, pp.585–594, 2010.
[16] S. Yamazaki, R. Sagawa, H. Kasawaki, K. Ikeuchi, and M. Sakauchi, “Microfacet billboarding,” Proceedings of the 13th Eurographics Workshop on Rendering, pp.169–180, 2002.
[17] B. Goldlack and M. Magnor, “Real-time microfacet billboardung for free-viewpoint video rendering,” Proceedings of International Conference on Image Processing, pp.713–16, 2003.
[18] T. Maeda, R. Suenaga, K. Suzuki, M.P. Tehrani, K. Takahashi, and T. Fujii, “Free viewpoint video for sports events using multi-resolution visual hull and micro-facet billboarding,” Proceedings of International Workshop on Smart Info-media System in Asia, SIS–5, 2016.
[19] K. Nonaka, Q. Yao, H. Sabirin, J. Chen, H. Sankoh, and S. Naito, “Billboard deformation via 3d voxel by using optimization for free-viewpoint system,” Proceedings of European Signal Processing Conference, pp.1500–1504, 2017.
[20] P.H. Calamai and J.J. Moré, “Projected gradient methods for linearly constrained problems,” Math. Program., vol.39, no.1, pp.93–116, 1987.
[21] T.S. Newman and H. Yi, “A survey of the marching cubes algorithm,” Computers & Graphics, vol.30, no.5, pp.854–879, 2006.
[22] X. Zuo, C. Du, S. Wang, J. Zheng, and R. Yang, “Interactive visual hull refinement for specular and transparent object surface reconstruction,” Proceedings of IEEE International Conference on Com-
Keisuke Nonaka received his Ph.D. degree in Information Processing from Tokyo Institute of Technology, Japan, in 2014. He joined KDDI Corporation in 2014, and is currently a research engineer in the area of image processing and computer vision. His research interests include free-viewpoint video, virtual reality, 3D model rendering and convex optimization.

Houari Sabirin is a principal research engineer at KDDI Research, Inc., where he is mostly working on object recognition and analysis with some applications in free-viewpoint video. Dr. Sabirin received his Ph.D. degree in Information and Communications Engineering from KAIST, Korea in 2012.

Jun Chen received his Ph.D. degree in Engineering from Hiroshima University, Japan, in 2015. He joined KDDI Research Inc. in 2016 and is currently a research engineer. His research interests are real-time image processing and parallel computation.

Sei Naito received his B.E., M.E., and Ph.D. degrees from Waseda University in 1994, 1996, and 2006, respectively. He joined Kokusai Denshin Denwa Corporation in 1996 and is currently a senior manager of Ultra-Realistic Communications Laboratory at KDDI Research, Inc., Japan. His research interests include free-viewpoint video applications and virtual reality systems.

Hiroshi Sankoh received B.Eng., M.Info., and Ph.D. degrees in Information Science from Kyoto University, in 2006, 2008, and 2015 respectively. He joined KDDI Corporation in 2008, and since then, he has been engaged in the study of free-viewpoint video technology. He is currently a research engineer of Ultra-Realistic Communications Laboratory in KDDI Research, Inc., Japan.