Hole Concealment Algorithm Using Camera Parameters in Stereo 360 Virtual Reality System

Sangguk Cha, Da-Yoon Nam and Jong-Ki Han*

Department of Electrical Engineering, Sejong University, Seoul 05006, Korea; sjept29@sju.ac.kr (S.C.); ndy9709@sju.ac.kr (D.-Y.N.) * Correspondence: hjk@sejong.edu; Tel.: +82-02-3408-3739

Keywords: virtual reality; stereo 360 VR; hole concealment

1. Introduction

Virtual reality (VR) has been one of the most important topics in the field of multimedia signals and systems for approximately 10 years. There are many applications for 360 VR systems, such as broadcasting, movies, social media communication, remote education, and virtual tourism [1]. 360 VR images and videos are generated using a 360 VR camera rig or a spinning VR camera, which is usually expensive or consists of heavy equipment.

As the technologies related to VR have been studied, methods that use a general purpose camera or cheap smartphone instead of the technical equipment to generate a 360 VR image have been developed [2–5]. In [2], the authors explained a method to stitch images using direct alignment or feature-based alignment. When the feature-based alignment approach is employed, feature points are searched in each picture to be stitched using one of the related algorithms [6–11]. Then, the feature points are matched by RANSAC [12], which provides homography matrixes to represent the relationship between neighboring pictures with overlapping common regions. Intrinsic and extrinsic matrixes of each picture can be derived from the homography matrix. The intrinsic and extrinsic matrixes of all pictures to be used for generating a VR image are jointly optimized through a bundle adjustment procedure [13]. In [3], a real time algorithm was proposed to make a panoramic image from pictures taken by the camera of a mobile phone, where the resulting image is shown on a cylindrical surface. The pictures taken by a mobile phone have different colors and brightness values compared to other pictures because these depend on the direction and position of the camera used in taking the pictures. This difference produces various artifacts in the VR image. In order to reduce the degradation, an algorithm to find the seams and blend the overlapping regions was proposed in [4]. Automatic Panoramic Image Stitching (APIS) [5] is one of the most efficient methods to make the panorama image or VR pictures. APIS consists of a variety of modules explained in [2,7,8,11,12] which include the feature-based alignment, Scale Invariant Feature Transform (SIFT), Speed-Up Robust Features (SURF), Oriented FAST and Rotated BRIEF (ORB), Random Sample Consensus (RANSAC), bundle adjustment, straightening, and multiband blending.

In order to increase the quality of the stitched VR image, we can use high-end specialized equipment, for example GoPro VR cameras. When high-end specialized equipment is used to generate 360° VR images, it is considerably easier to align pictures without notable seam artifacts than that when using conventional cameras. Although VR images obtained from high-end specialized equipment have higher quality than those obtained from conventional cameras, conventional cameras are more affordable and simpler to use than specialized equipment. In fact, specialized equipment is bulky and expensive, thereby hindering its use in daily life.
With the increase in the viewer requirement for a sense of immersion, stereo 360 VR systems have been introduced to meet this need [14,15]. A stereo 360 VR image provides a more realistic and immersive sense to the viewer than a monocular 360 VR image, but the cost to make the contents increases further. The high cost of the equipment to generate 360 VR images may be one of the things preventing the spread of applications based on VR technology. This paper considers a system to generate a stereo 360 VR image using two cheap smartphones or two general purpose cameras attached to a rig.

In our scenario, because the general-purpose cameras are used to capture pictures, the pictures taken in arbitrary directions may produce non-overlapping regions and those become holes in the stitched image. In order to conceal the holes in stereo 360 VR images, various inpainting algorithms [16–24] can be considered as solutions, even though these were originally proposed for non-VR images. Bertalmio et al. [16] proposed an inpainting algorithm based on fluid dynamics, where a two-dimensional Navier–Stokes equation is approximated and combined with the Poisson equation for the vorticity. This approach produces an approximate solution for the holes in non-VR images. In the algorithm proposed by Telea et al. [17], each pixel in a hole is progressively concealed from the boundary to the center of the hole, where the value of a pixel to be concealed is set by the weighted average of the neighbouring pixels. Criminisi et al. [18] proposed a method to conceal a hole by filling it using a block. In this algorithm, the most similar patches are searched over a region around the hole. This scheme is a kind of template matching technique and the squared sum of the difference (SSD) is used as a criterion. On other hand, algorithms based on deep learning techniques were proposed in [19–24]. Liu et al. [19] proposed a deep learning-based method to conceal a hole in a non-VR image, where convolution weights are normalized by the mask area of the window. This effectively prevents the convolution filters from capturing too many zeros when they traverse over the incomplete region. When the deep learning-based algorithm is applied, the coefficients and variables in the convolutional network should be optimized for the huge quantity of training data. Thus, it requires a long training time and the performance depends on the type of training data. Note that certain conventional methods [16–24] use the data of the same camera stream to conceal the holes. In these techniques, the holes are filled based on the neighboring pixels and therefore their performances are limited when the size of the hole is large.

Unlike those techniques [16–24], when we fill the holes in the stereoscopic VR images, we use the data from the opposite camera stream. Using the data of the opposite camera stream is more efficient than using the data from the same camera stream, because the left and right views are correlated with each other. In order to compare the performances of two categories (using the data from same camera stream and opposite camera streams), we provide the simulation results obtained by methods in two categories in Section 4.

This paper proposes an efficient algorithm to conceal the holes in stereo 360 VR images, where camera parameters are utilized to estimate the location of the most similar fraction in other view images.

This paper is organized as follows. In Section 2, we formulate the problem, where two general purpose cameras are used to capture pictures for the left and right views. These pictures are used to generate stereo 360 VR images, where some blocks related to a hole are defined. In Section 3, we propose an efficient algorithm to conceal the holes, where the geometrical property of the pixels in VR images and the relationship between them are analyzed. Simulation results that demonstrate the performance of the proposed algorithm are provided in Section 4, where the performances of various techniques are compared subjectively and objectively. In Section 5, we provide a brief conclusion for this paper.

2. Problem Formulation

The Figure 1 shows the process used to make the stereo 360 VR images, which consist of left and right equi-rectangular projections (ERPs). In this configuration, the left and right cameras are mounted on a rig and capture pictures in the same direction, where general-
purpose cameras (e.g., built-in cameras in smartphones) are used to capture pictures. The left and right ERPs are generated by applying several techniques to the sets of pictures with the left and right views. For example, the left ERP is made by consecutively applying feature extraction, mapping between features, a calculation of the homography matrix to represent the relationship, bundle adjustment to optimize the camera parameters, seam finding, and blending. The right ERP image is also made by a series of the same techniques. In this paper, APIS [5] is used to make the left and right ERPs, because it is known as one of the most efficient algorithms. APIS is applied independently to the left and right sets of pictures that have been captured by two built-in smartphone cameras. In this scenario, no mechanical equipment is used, other than a tripod. Thus, the pictures are captured in arbitrary directions and positions. Note that APIS does not include a function to conceal the holes in the stitched image. APIS is the basic style algorithm to stitch the pictures.

Figure 1. Procedure to make stereo 360 VR images consisting of left and right equi-rectangular projections (ERPs).

Figure 2 shows an example of the left and right ERPs that are generated using the procedure shown in Figure 1. As observed in Figure 2, we find some holes in these pictures. In the scenario illustrated in Figure 1, given that general-purpose cameras (instead of specialized equipment) are used, their narrow field-of-view (FOV) and the fact that pictures are often captured in arbitrary directions may produce non-overlapping regions and distortion (e.g., ghost effect or misalignment) in stitched 360° VR images. Non-overlapping regions also show up owing to the limitations of the stitching module, because the derivation of the relationship between spatially neighboring pictures is based on the RANSAC and feature points of each picture. The non-overlapping regions become holes in the stitched image, where the shape of the hole is dependent on the structure of the neighboring warped pictures. The shape of the holes may look like a square or a lozenge because those holes are surrounded by the neighboring warped pictures. Examples of the holes are shown in Section 4, where we can see the various shapes of holes.
Figure 2. Examples of left and right ERPs, where a hole is found in the left ERP: (a) $B^L_l$, $B^r_l$, $B^r_c$ in the left ERP image; (b) $B^L_r$, $B^r_k$, $B^r_k$ in the right ERP image.

In Figure 2, $B^L_l$ is a block overlapping a hole in the left ERP. $B^L_l$ and $B^r_l$ are the left and right sides neighboring blocks of $B^L_l$, respectively. $B^R_r$, $B^L_r$, and $B^r_k$ in the right ERP are blocks corresponding to $B^L_l$, $B^r_l$, and $B^L_l$, respectively. The corresponding blocks in the left and right ERPs are located in the same positions.

This paper proposes an efficient algorithm to conceal the holes, where pixels in the holes are filled with pixels in other ERPs and the location of the pixel used is derived from the camera parameters of the neighboring blocks {$B^L_l$, $B^r_l$, $B^r_r$, $B^L_r$, $B^r_l$, $B^L_l$}.

3. Algorithm Proposed to Conceal a Hole

3.1. Hole Detection

Figure 3 explains the process used to detect a hole in an ERP image. As observed in Figure 2, because a hole is a region that has not been covered by the warped pictures using the stitching algorithm, the color of the hole is black, with a luminance of zero. Thus, in step 1 of Figure 3, the colored ERP image is converted to a black-and-white picture to efficiently detect the holes. In step 2, all of the non-black pixels are replaced by white pixels, as shown in Figure 3c, where the black and white pixels have gray levels of 0 and 255 for an image represented with 8 bits/pixel, respectively. This results in the candidate regions for holes. Note that not all of the pixels in the dark objects have a zero value, even though some of the pixels in the objects may have a zero value. In order to classify the black pixels as hole and non-hole regions, a contour finding algorithm [25] is applied to image Figure 3c in step 3, which provides several contours with various sizes. In image Figure 3d, the perimeter of the contour of the hole is longer than that of a non-hole region, because the non-hole region is part of the dark object and it has the shape of an isolated particle. Therefore, if the length of the contour is shorter than a threshold, the dark region is removed, as shown in image Figure 3e. Finally, the block overlapping the dark region is set to $B^L_l$ in image Figure 3e.

3.2. Camera Parameters of Neighboring Blocks

In Figure 1, when an ERP image is made, the stitching algorithm is applied to the set of pictures that have been taken by a single camera (the left or right camera). As explained in Section 2, each picture in a set of pictures with the left or right views is warped and translated according to the intrinsic matrix $K$ and extrinsic matrix $R$ of each picture. The following equations are the intrinsic and extrinsic matrices of picture $I_i$.

$$K(I_i) = \begin{pmatrix} f_x(I_i) & s(I_i) & c_x(I_i) \\ 0 & f_y(I_i) & c_y(I_i) \\ 0 & 0 & 1 \end{pmatrix}$$
where elements $f_x(I_i)$ and $f_y(I_i)$ are the horizontal and vertical focal lengths of the camera used, respectively. $(c_x(I_i), c_y(I_i))$ is the location of the intersection between the Z axis of the world coordinate system and the plane of the image sensor. $s$ is a skew coefficient. In (2), $R$ is a rotation matrix used to represent the direction of camera lens when picture $I_i$ is taken. $K$ and $R$ are derived from a homography matrix calculated by RANSAC. The parameters in the $K$’s and $R$’s of all the pictures are optimized by the bundle adjustment based on the Levenberg-Marquardt algorithm [26].

![Figure 3](image.png)

**Figure 3.** Process to detect the hole in an ERP image: (a) A stitched picture including a hole; (b) a black-and-white picture; (c) a binary image resulted from Step 2; (d) several contours of the candidate holes in the binary picture; (e) the map indicating the decided holes.
As explained in Section 2, the blocks \( \{B^i_R, B^i_R, B^c_R, B^c_L, B^i_L, B^i_L\} \) have their own \( K' \)'s and \( R' \)'s, which can be represented by substituting \( \{B^i_R, B^i_R, B^c_R, B^c_L, B^i_L, B^i_L\} \) into \( I_l \) in (1) and (2) as follows. As the equations have the same pattern, we represent those for \( B^i_L \) only.

\[
K(B^i_L) = \begin{pmatrix}
    f_x(B^i_L) & s(B^i_L) & c_x(B^i_L) \\
    0 & f_y(B^i_L) & c_y(B^i_L) \\
    0 & 0 & 1
\end{pmatrix}
\]  

(3)

\[
R(B^i_L) = \begin{pmatrix}
    r_{11}(B^i_L) & r_{12}(B^i_L) & r_{13}(B^i_L) \\
    r_{21}(B^i_L) & r_{22}(B^i_L) & r_{23}(B^i_L) \\
    r_{31}(B^i_L) & r_{32}(B^i_L) & r_{33}(B^i_L)
\end{pmatrix}
\]  

(4)

### 3.3. Concealment Based on Intrinsic and Extrinsic Matrixes

In this subsection, we derive the relationship between the stereo views, where the camera parameters of the warped pictures are utilized. In Figure 4, \( I_L \) and \( I_R \) are the warped images after they have been taken by the left and right cameras, respectively. Based on the geometric relation between the camera coordinate and world coordinate systems [27], the \((X, Y, Z)\) of point \( P \) in the world coordinate system is mapped to the \((x_L, y_L) \) and \((x_R, y_R) \) of points \( p_L \) and \( p_R \) in \( I_L \) and \( I_R \), respectively. The relation is represented as follows.

\[
P = R(I_R)K(I_R)^{-1}p_R
\]

(5)

\[
P = R(I_L)K(I_L)^{-1}p_L
\]

(6)

\[
\text{Combining (5) and (6) gives}
\]

\[
R(I_R)K(I_R)^{-1}p_R = R(I_L)K(I_L)^{-1}p_L
\]

(7)

\[
p_R = K(I_R)R(I_R)^{-1}R(I_L)K(I_L)^{-1}p_L
\]

(8)

Therefore, the relation between the pixels in \( I_L \) and \( I_R \) can be represented with a 3 × 3 matrix \( H(I_L \rightarrow I_R) \) as follows.

\[
p_R = H(I_L \rightarrow I_R)p_L
\]

(9)

where

\[
H(I_L \rightarrow I_R) = K(I_R)R(I_R)^{-1}R(I_L)K(I_L)^{-1}
\]

(10)
The $H(I_L \rightarrow I_R)$ can be calculated when $K(I_R), R(I_R), R(I_L)$, and $K(I_L)$ are known. If $I_L$ and $I_R$ are replaced by $B_L^r$ and $B_R^r$, respectively, then (10) becomes

$$H(B_L^r \rightarrow B_R^r) = K(B_R^r)R(B_R^r)^{-1}R(B_L^r)K(B_L^r)^{-1}$$ (11)

In the proposed algorithm based on (11), each pixel in a hole is filled with a pixel in the right ERP, where $H(B_L^r \rightarrow B_R^r)$ provides the location of the pixel in the right ERP to replace a particular pixel in a hole. This means that we can conceal the hole if $K(B_R^r), R(B_R^r), R(B_L^r)$, and $K(B_L^r)$ are known.

On the other hand, when the hole is found in the right ERP, the following equation is used instead of (11).

$$H(B_L^r \leftarrow B_R^r) = K(B_L^r)R(B_L^r)^{-1}R(B_R^r)K(B_R^r)^{-1}$$ (12)

Figure 5 shows an example of concealing each pixel in a hole when the hole is found in the left ERP.

![Concealment process based on derived homography matrix](image)

**Figure 5.** Concealment process based on derived homography matrix: (a) A part of the left ERP; (b) A part of the right ERP.

### 3.4. Prediction for Camera Parameters

This subsection explains the method used to predict $K(B_R^r), R(B_R^r), R(B_L^r)$, and $K(B_L^r)$ in order to derive $H(B_L^r \rightarrow B_R^r)$ in (11). As shown in Figure 2, because $B_L^r$ includes a hole, $R(B_L^r)$ and $K(B_L^r)$ are unknown. In addition, because $B_R^r$ consists of fractions of multiple pictures in the right view, $K(B_R^r)$ and $R(B_R^r)$ should be recalculated instead of using those resulting from the process to make the right ERP. In order to predict $\{K(B_R^r), R(B_R^r), R(B_L^r), K(B_L^r)\}$, the following operations are executed.

First, we estimate homography matrices $H(B_L^l \rightarrow B_R^l)$ and $H(B_L^r \rightarrow B_R^r)$ for pairs $[B_L^l, B_R^l]$ and $[B_L^r, B_R^r]$, respectively. $H(B_L^l \rightarrow B_R^l)$ and $H(B_L^r \rightarrow B_R^r)$ are derived by applying “feature extraction,” RANSAC, and “bundle adjustment” to those pairs. Figure 6 shows the examples used to estimate $H(B_L^l \rightarrow B_R^l)$ and $H(B_L^r \rightarrow B_R^r)$, which have the following relationships.

$$H(B_L^l \rightarrow B_R^l) = K(B_R^l)R(B_R^l)^{-1}R(B_L^l)K(B_L^l)^{-1}$$ (13)

$$H(B_L^r \rightarrow B_R^r) = K(B_R^r)R(B_R^r)^{-1}R(B_L^r)K(B_L^r)^{-1}$$ (14)

From the homography matrices in (13) and (14), $K(B_R^r), R(B_R^r), K(B_L^l)$ and $K(B_R^l), R(B_R^l), K(B_L^l)$ are derived by solving the simultaneous equations related to the focal length.

Second, $R(B_R^r) + R(B_L^r)$ are estimated from $[R(B_R^l), R(B_L^l)]$ and $[R(B_R^r), R(B_L^r)]$, respectively, which were obtained in the previous step. In [28], a rotation matrix R can
be represented as a combination of rotational components $R_x, R_y, R_z$ on the $x, y, z$ axes as follows.

$$
R = R_z(\theta)R_y(\sigma)R_x(\phi) =
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \sigma & 0 & \sin \sigma \\
0 & 1 & 0 \\
-\sin \sigma & 0 & \cos \sigma
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi & \cos \phi
\end{bmatrix}
$$  
(15)

where $\phi, \sigma$, and $\theta$ are the Euler angles about the $x, y, z$ axes, respectively. Applying the property to $[R(B^l_i), R(B^c_i), R(B^r_i), R(B^L_j), R(B^R_j), R(B^c_k), R(B^r_k)]$ gives

$$
R(B^j_i) = R_z(\theta(B^j_i))R_y(\sigma(B^j_i))R_x(\phi(B^j_i))
$$  
(16)

$$
R_x(\phi(B^j_i)) =
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \phi(B^j_i) & -\sin \phi(B^j_i) \\
0 & \sin \phi(B^j_i) & \cos \phi(B^j_i)
\end{bmatrix}
$$  
(17)

$$
R_y(\sigma(B^j_i)) =
\begin{bmatrix}
\cos \sigma(B^j_i) & 0 & \sin \sigma(B^j_i) \\
0 & 1 & 0 \\
-\sin \sigma(B^j_i) & 0 & \cos \sigma(B^j_i)
\end{bmatrix}
$$  
(18)

$$
R_z(\theta(B^j_i)) =
\begin{bmatrix}
\cos \theta(B^j_i) & -\sin \theta(B^j_i) & 0 \\
\sin \theta(B^j_i) & \cos \theta(B^j_i) & 0 \\
0 & 0 & 1
\end{bmatrix}
$$  
(19)

where $i = \{l, c, r\}$ and $j = \{L, R\}$. In (16)–(19), $\phi(B^j_i)$, $\sigma(B^j_i)$, and $\theta(B^j_i)$ are the Euler angles about the $x, y, z$ axes in $R(B^j_i)$, respectively. As observed in Figure 2, because $B^L_i$ and $B^R_i$ are located in the centers of $\{B^L_j, B^c_j, B^r_j\}$ and $\{B^L_k, B^c_k, B^r_k\}$, respectively, we can assume that their Euler angles have the following relationships.

$$
\phi(B^j_i) = \frac{\phi(B^L_j) + \phi(B^R_j)}{2}
$$  
(20)

$$
\sigma(B^j_i) = \frac{\sigma(B^L_j) + \sigma(B^R_j)}{2}
$$  
(21)

$$
\theta(B^j_i) = \frac{\theta(B^L_j) + \theta(B^R_j)}{2}
$$  
(22)

where $j = \{L, R\}$. Substituting (20)–(22) into (16) provides $R(B^L_i)$ and $R(B^R_i)$.

**Figure 6.** Homography matrices $H(B^L_i \rightarrow B^L_k)$ and $H(B^R_i \rightarrow B^R_k)$ are calculated by applying the feature extraction algorithm and matching those feature points: (a) Example of feature matching when Homography matrix is $H(B^L_i \rightarrow B^L_k)$; (b) Example of feature matching when Homography matrix is $H(B^R_i \rightarrow B^R_k)$. 

Third, $K(B^c_L)$ and $K(B^c_R)$ are estimated from $\{K(B^l_R), K(B^l_L)\}$ and $\{K(B^r_R), K(B^r_L)\}$, respectively, which were derived in the first step. As we can see in (1), because the $K$ matrixes consist of the focal length, aspect ratio, and center point, $K(B^c_L)$ and $K(B^c_R)$ can be derived by averaging the $K$’s of the neighboring blocks, as follows.

$$K(B^c_j) = \frac{K(B^l_j) + K(B^r_j)}{2} \quad (23)$$

where $j = \{L, R\}$.

Finally, because we obtained $\{K(B^c_R), R(B^c_R), R(B^c_L), K(B^c_L)\}$ in the previous steps, substituting them into (11) results in $H(B^c_L \rightarrow B^c_R)$, which can be used to conceal each pixel in a hole.

### 3.5. Summary of the Proposed Algorithm

Figure 7 summarizes the proposed algorithm. In the first step, the hole is found using the algorithm described in Section 3.1, where the hole $B^c_L$ and the neighboring blocks $B^l_L, B^l_R, B^r_L, B^r_R$ are defined according to the relationship between their coordinates. In the second step, we derive $K(B^c_R), R(B^c_R), R(B^c_L)$, and $K(B^c_L)$ using the algorithm proposed in Section 3.4. In the third step, each pixel in the hole $B^c_L$ is filled with the corresponding pixel in the right ERP, where the $H(B^c_L \rightarrow B^c_R)$ of (11) provides the coordinate of the corresponding pixel. When the hole is in the right VR image, $H(B^c_L \leftarrow B^c_R)$ of (12) is used to fill the hole.

![Flowchart of the Proposed Algorithm](image-url)

*Figure 7. Flowchart of the proposed algorithm.*

The main characteristics of the proposed algorithm are as follows: First, the proposed algorithm conceals the holes in stereoscopic VR images, whereas most conventional algorithms [16–24,29] for filling holes were proposed for non-VR images. In particular, certain concealment algorithms [30–32] were proposed to fill the holes in depth images, where the holes have arbitrary shapes and some of them are small particles. Second, our algorithm can derive the location of the most similar pixel for each pixel in a hole. In our scenario, we use the data of the camera stream of another view to fill the holes, where the exact location of the most similar part should be calculated to increase the objective and subjective quality of the filled region. The position is estimated by analyzing Euler angles of extrinsic matrices of the related neighbor blocks. In the analysis of Euler angles, we decompose the angles along $x, y$, and $z$ axes and then merge them to find the exact location of the reference data. The relevant explanation is provided in Section 3.4.
4. Simulation Results

In order to demonstrate the performance of the proposed algorithm, we compare it with various conventional methods, including those of Bertalmio [16], Telea [17], Criminisi [18], Liu [19], GiliSoft Stamp Remover [33], and Theirpaint [34] from the subjective and objective viewpoints. These methods were explained in Section 1.

We captured picture sets using the built-in camera of a Samsung Galaxy 9 smartphone. There were more than 100 pictures in each set. The pictures within a single set were stitched using APIS to create the stereoscopic VR images. Figure 8 shows the left or right VR images of the stereoscopic VR images having holes of various sizes and locations. The VR images including the holes were used as test sets to evaluate the performances of the proposed algorithm and various conventional methods. In Figure 8a–f, the holes have been generated through the stitching algorithm owing to the reason described in Section 2. Those pictures are used only for subjective evaluations because there is no reference VR image without any hole. When we created Figure 8g,h, the stitched VR images did not have any holes. Therefore, we have made an artificial hole in each VR image. Consequently, Figure 8g,h can be used for objective evaluation, because the difference between the reference image (without any hole) and the concealed images (the results of various algorithms) can be calculated numerically.

Figure 8. Equirectangular projection images generated from test sets to evaluate different methods. (a) Laboratory, (b) Brick Road, (c) Alleyway, (d) Tree, (e) Bench, (f) Street, (g) Hallway, and (h) Mozart Hall.
4.1. Subjective Performance

In order to check the performance of the proposed algorithm, we applied the proposed algorithm for the test sets of Brick Road, Alleyway, Tree, Bench, and Street. In Figure 9, VR images including the holes are compared with the concealed images, respectively. As shown in Figure 9, we can see that the proposed algorithm conceals the various holes effectively in the test images.

Figure 9. ERP images including holes and the concealed regions, where the holes are concealed by the proposed algorithm.
To compare the performances of various techniques subjectively, the pictures whose holes have been concealed by the methods are shown in Figure 11, where all of the pictures are a part of the left ERP picture and we made a hole artificially as Figure 10b in order to compare the concealed pictures Figure 11a–h with the reference image Figure 10a.

![Figure 10. Example of reference picture and the original picture with an artificial hole for test set Mozart Hall: (a) Reference picture with no hole; (b) Original picture with a hole.](image)

![Figure 11. Concealed images resulting from various algorithms for test set Mozart Hall: (a) Result from Bertalmio [16]; (b) Result from Telea [17]; (c) Result from Criminisi [18]; (d) Result from Liu [19]; (e) Result from Gilisoft Stamp Remover [33]; (f) Result from Theinpaint [34]; (g) Result from simple copy; (h) Result from proposed method.](image)

As observed in Figure 11, the proposed algorithm outperforms the other techniques subjectively. In the picture Figure 11a result from Bertalmio [16], the pixels in the hole are filled by copying the neighboring pixels horizontally or vertically. Thus, the concealed region has artifact lines. When the method of Telea [17] is used to conceal the hole, each pixel in a hole is concealed progressively from the boundary to the center of the hole, where the value of the concealed pixel is set to the weighted sum of the values of the neighboring pixels. Thus, the concealed region of Figure 11b has smoothed and diffused distortion. In Figure 11c, when the method of Criminisi [18] is used, the pixels in the hole are filled by blocks, where the template matching algorithm is used. In this method, the SSD of template part is considered a cost function to estimate the most similar block, and the values of the pixels in the non-template part are not considered. Thus, some pixels in the
hole are filled with wrong values. As shown in Figure 11d, the method of Liu [19] based on deep learning has filled the hole with inappropriate blocks, which are generated from the deep learning engine. Figure 11e,f were resulted from Gilisoft stamp remover [33] and Theipaint [34], respectively. As shown in Figure 11e,f, these methods do not remove the holes completely. In Figure 11g, the hole in the left ERP is concealed just by copying a block in the right ERP, where the shapes and coordinates of the blocks, including the hole in the left ERP and the copied block in the right ERP, are same. As observed in Figure 11g, the concealed region has significant mismatches around the hole, because this method does not consider the disparity between the left and right views. Whereas the conventional techniques have serious artifacts in the concealed region, the image Figure 11h produced by the proposed algorithm has a natural and continuous boundary in the concealed region.

Figures 12 and 13 show the simulation results for indoor pictures, where the size of the hole is bigger than that of Figures 10 and 11. As can be seen from Figures 12 and 13 and Figures 10 and 11, the performance tendencies of the various algorithms do not change even if the size of the hole is varied.

Figure 12. Example of reference picture and the original picture with a large artificial hole for test set Hallway: (a) Reference picture with no hole; (b) Original picture with a hole.

Figure 13. Concealed images resulting from various algorithms for a large hole for test set Hallway: (a) Result from Bertalmio [16]; (b) Result from Telea [17]; (c) Result from Criminisi [18]; (d) Result from Liu [19]; (e) Result from Gilisoft stamp remover [33]; (f) Result from Theinpaint [34]; (g) Result from simple copy; (h) Result from proposed method.
Figure 14 shows the simulation results for the naturally generated holes that occurred non-artificially during the stitching procedure. Note that no reference picture was used in this test. In this configuration, both the left and right ERP images had holes at different locations independently. In this test, the concealment algorithms were applied to the left and right ERP pictures independently. The results of the conventional and proposed techniques are shown in Figure 14b–g,h, respectively. When the proposed algorithm was applied, Equations (11) and (12) were used to conceal the holes in the left and right ERPs, respectively. As observed in Figure 14, the proposed method outperformed the other conventional techniques subjectively. In addition, the performance tendencies of the techniques were the same as those of Figures 10 and 11 and Figures 12 and 13.

Figure 14. Concealed image results from various algorithms for a hole generated during stitching procedure for test set Laboratory: (a) Original left and right ERPs with hole; (b) Left and right ERPs result by Bertalmio [16]; (c) Left and right ERPs result by Tekeia [17]; (d) Left and right ERPs result by Criminisi [18]; (e) Left and right ERPs result by Liu [19]; (f) Left and right ERPs result by Gilisoft stamp remover [33]; (g) Left and right ERPs result by Theinpaint [34]; (h) Left and right ERPs result by the proposed method.
4.2. Objective Performance

The objective performances of the concealment algorithms were evaluated with common measures like the structural similarity (SSIM) [35], peak-to-peak signal-to-noise ratio (PSNR) [36], and consumed CPU time. All experiments are run on the CPU based on AMD Ryzen 3.40 GHz and 32-GB RAM. The SSIM and PSNR are the average values of the SSIM and PSNR values for the red, green, and blue components. SSIM is in the range of 0–1.

As the values of SSIM and PSNR increase, the quality of the resulting image increases. Note that SSIM and PSNR can be evaluated when the reference pictures are given, as in Figures 10 and 11 and Figures 12 and 13.

On the other hand, the CPU times consumed by the methods were checked using a personal computer. The CPU time was measured with the resolution of a second. As the CPU time depended on the complexity of the algorithm, the technique consuming the smallest CPU time was considered the simplest method.

The outdoor pictures (Mozart Hall) shown in Figures 10 and 11 are evaluated in Tables 1 and 2, where the best and 2nd best results are represented with red and blue numbers, respectively. As seen in Table 1, the proposed algorithm produced the best quality from the viewpoints of both the SSIM and PSNR. The inpaint [34] had the 2nd best quality for both SSIM and PSNR. The “simple copy” required the smallest CPU time, because it filled the hole just by copying the same size block of the other view. The proposed algorithm is one of the simplest algorithms in the consumed time. When the method of Liu [19] was used, the consumed CPU time could not be measured because the training time was huge and depended on the quantity of the training data.

Table 1. Comparison between performances of various methods for outdoor pictures (Mozart Hall).

| Method            | SSIM     | PSNR (dB) | Time (s) |
|-------------------|----------|-----------|----------|
| Bertalmio [16]    | 0.6483   | 19.6982   | 12.9177  |
| Telea [17]        | 0.6320   | 16.5155   | 13.1796  |
| Criminisi [18]    | 0.4406   | 17.0804   | 30346    |
| Liu [19]          | 0.2381   | 17.3725   | N/A      |
| Gilisoft [33]     | 0.7427   | 24.6347   | 2.1600   |
| Theinpaint [34]   | 0.7701   | 25.0058   | 28.1900  |
| Simple copy       | 0.6196   | 21.1793   | 0.0160   |
| Proposed algorithm| 0.7954   | 25.4479   | 7.3261   |

Table 2. Performances of proposed method according to the size of the neighboring blocks for outdoor pictures (Mozart Hall).

| Width Ratio | SSIM     | PSNR (dB) | Time (s) |
|-------------|----------|-----------|----------|
| 1.0         | 0.7491   | 24.1525   | 5.4439   |
| 1.2         | 0.7585   | 24.4355   | 6.3663   |
| 1.5         | 0.7954   | 25.4479   | 7.3261   |
| 1.75        | 0.7614   | 24.6553   | 8.0263   |
| 2.0         | 0.5986   | 19.8681   | 8.8201   |

Table 2 shows the performances of the proposed algorithm for a variety of sizes of neighboring blocks \( \{B_{LR}, B_{RL}, B_{LR}', B_{RL}' \} \), where the numbers in the first column denote the ratio between the widths of neighboring block \( B_{LR}' \) and block \( B_{RL} \) that includes the hole. As observed in this table, the best performance is found when the ratio is 1.5 (i.e., the width of \( B_{LR}' \) is 50% wider than that of \( B_{RL} \)).

When the size of the neighboring blocks is small (the ratio is under 1.5), the performance of the proposed algorithm increases with the size, because the accuracy of \( H(B_{LR} \rightarrow B'_{RL}) \) increases as the number of pixels utilized to derive it increases. However, when the size is large (the ratio is over 1.5), the neighboring blocks include fractions of multiple pictures, which degrades the accuracy of \( H(B_{LR} \rightarrow B'_{RL}) \). In this table, the complex-
ity increases with the width of the neighboring blocks, because the number of pixels to be considered increases in making $H(B_L \rightarrow B'_R)$ and $H(B'_L \rightarrow B'_R)$ increases.

Tables 3 and 4 show the simulation results for the indoor pictures (Hallway) shown in Figures 12 and 13. Note that the hole in Figures 12 and 13 are larger than that of Figures 10 and 11. As observed in Tables 3 and 4, the performance tendencies of the methods are similar to those seen in Tables 1 and 2. As observed in Table 3, the proposed algorithm has the best performance. Gilisoft [33] and Theinpaint [34] had the 2nd best quality for PSNR and SSIM, respectively. The 2nd best performance is seen for the different ratios in Tables 2 and 4 because the pictures of Figures 12 and 13 include more complex regions and larger holes than those of Figures 10 and 11.

Table 3. Comparison between performances of various methods for indoor pictures (Hallway) having a large hole.

| Method         | SSIM  | PSNR (dB) | Time (s) |
|----------------|-------|-----------|----------|
| Bertalmio [16] | 0.7153| 18.6928   | 15.7136  |
| Telea [17]     | 0.7343| 18.0315   | 16.3712  |
| Criminisi [18] | 0.5691| 18.6113   | 62791    |
| Liu [19]       | 0.3511| 16.1611   | N/A      |
| Gilisoft [33]  | 0.7487| 23.5176   | 0.5500   |
| Theinpaint [34]| 0.7585| 22.5666   | 6.3600   |
| Simple copy    | 0.6544| 16.9261   | 0.2273   |
| Proposed algorithm | 0.7721| 23.8020   | 8.8516   |

Table 4. Performances of proposed method according to size of neighboring blocks for indoor picture (Hallway) having a large hole.

| Width Ratio | SSIM  | PSNR (dB) | Time (s) |
|-------------|-------|-----------|----------|
| 1.0         | 0.6031| 15.8433   | 6.7485   |
| 1.2         | 0.6476| 18.3133   | 7.6910   |
| 1.5         | 0.7721| 23.8020   | 8.8516   |
| 1.75        | 0.6505| 17.6263   | 9.3063   |
| 2.0         | 0.6656| 18.7562   | 9.9974   |

From the results in Tables 1–4, we can see that the proposed algorithm outperforms the conventional methods objectively.

5. Conclusions

We proposed an efficient algorithm to conceal the holes generated in stereo VR pictures, where the positions of the pixels to fill the hole were derived using the camera parameters. The camera parameters were predicted from their relationship with the homography matrix. Whereas conventional inpainting algorithms have constraints in the concealment of the hole because they were invented for non-VR images, the proposed method efficiently fills all of the pixels in the hole using the relationship between the left and right ERPs.

Author Contributions: Conceptualization, J.-K. H.; Data curation, S. C. and D.-Y. N.; Formal analysis, S. C., D.-Y. N. and J.-K. H.; Funding acquisition, J.-K. H.; Investigation, S. C. and J.-K. H.; Methodology, S. C. and J.-K. H.; Resources, J.-K. H.; Software, S. C.; Supervision, J.-K. H.; Visualization, J.-K. H.; Writing—original draft, S. C.; Writing—review & editing, J.-K. H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Research Foundation of Korea (NRF) under Grant NRF-2018R1A2A2A05023117 and partially by the Institute for Information & Communications Technology Promotion (IITP) under Grant 2017-0-00486 funded by the Korea government (MSIT).

Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, K.; Liu, S.J. The application of virtual reality technology in physical education teaching and training. In Proceedings of the 2016 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Beijing, China, 10–12 July 2016; pp. 245–248.

2. Szeliski, R. Image Alignment and Stitching: A Tutorial, Foundations and Trends in Computer Graphics and Computer Vision; Now Publishers: Delft, The Netherlands, 2006; Volume 2, p. 120.

3. Kim, B.S.; Lee, S.H.; Cho, N.I. Real-time panorama canvas of natural images. IEEE Trans. Consum. Electron. 2011, 57, 1961–1968.

4. Xiong, Y.; Pulli, K. Fast panorama stitching for high-quality panoramic images on mobile phones. IEEE Trans. Consum. Electron. 2010, 56, 298–306. [CrossRef]

5. Brown, M.; Lowe, D.G. Automatic panoramic image stitching using invariant features. Int. J. Comput. Vis. 2007, 74, 59–73. [CrossRef]

6. Viswanathan, D.G. Features from accelerated segment test (fast). In Proceedings of the 10th Workshop on Image Analysis for Multimedia Interactive Services, London, UK, 6–8 May 2009; pp. 6–8.

7. Lowe, D.G. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 2004, 60, 91–110. [CrossRef]

8. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In European Conference on Computer Vision; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.

9. Calonder, M. Brief: Binary robust independent elementary features. In European Conference on Computer Vision; Springer: Berlin/Heidelberg, Germany, 2010; pp. 778–792.

10. Leutenegger, S.; Chili, M.; Siegwart, R.Y. BRISK: Binary robust invariant scalable keypoints. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2548–2555.

11. Rublee, E. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.

12. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 1981, 24, 381–395. [CrossRef]

13. Triggs, B.; McLauchlan, P.F.; Hartley, R.I.; Fitzgibbon, A.W. Bundle adjustment—A modern synthesis. In International Workshop on Vision Algorithms; Springer: Berlin/Heidelberg, Germany, 1999; pp. 298–372.

14. Livatino, S. Stereoscopic visualization and 3-D technologies in medical endoscopic teleoperation. IEEE Trans. Ind. Electron. 2014, 62, 525–535. [CrossRef]

15. Kramida, G. Resolving the vergence-accommodation conflict in head-mounted displays. IEEE Trans. Vis. Comput. Graph. 2015, 22, 1912–1931. [CrossRef]

16. Bertalbio, M.; Bertozzi, A.L.; Sapiro, G. Navier-stokes, fluid dynamics, and image and video inpainting. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 8–14 December 2001; p. I-I.

17. Telea, A. An image inpainting technique based on the fast marching method. J. Graph. Tools 2004, 9, 23–34. [CrossRef]

18. Criminisi, A.; Pérez, P.; Toyama, K. Region filling and object removal by exemplar-based image inpainting. IEEE Trans. Image Process. 2004, 13, 1200–1212. [CrossRef]

19. Liu, G. Image inpainting for irregular holes using partial convolutions. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 85–100.

20. Elango, P. Digital image inpainting using cellular neural network. Int. J. Open Probl. Compt. Math. 2009, 2, 439–450.

21. Yan, Z. Shift-net: Image inpainting via deep feature rearrangement. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 1–17.

22. Wu, X. A light cnn for deep face representation with noisy labels. IEEE Trans. Inform. Forensics Secur. 2018, 13, 2884–2896. [CrossRef]

23. Zhang, S.; He, R.; Tan, T. Demeshnet: Blind face inpainting for deep meshface verification. IEEE Trans. Inform. Forensics Secur. 2017, 13, 637–647. [CrossRef]

24. Zhang, K. Learning deep CNN denoiser prior for image restoration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3929–3938.

25. Arbeláez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2010, 33, 898–916. [CrossRef] [PubMed]

26. Lourakis, M.I.A. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. Found. Res. Technol. 2005, 4, 1–6.
27. Ye, Y. Algorithm descriptions of projection format conversion and video quality metrics in 360Lib Version 9. In Proceedings of the Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting, Marrakech, MA, USA, 9–18 January 2019.

28. Slabaugh, G.G. Computing Euler angles from a rotation matrix. *Retriev. August* 1999, 6, 39–63.

29. Guillemot, C.; le Meur, O. Image Inpainting: Overview and Recent Advances. *IEEE Signal Process. Mag.* 2014, 31, 127–144. [CrossRef]

30. Lu, H.; Zhang, Y.; Li, Y.; Zhou, Q.; Tadoh, R.; Uemura, T.; Kim, H.; Serika, S. Depth Map Reconstruction for Underwater Kinect Camera Using Inpainting and Local Image Mode Filtering. *IEEE Access* 2017, 5, 7115–7122. [CrossRef]

31. Buyssens, P.; le Meur, O.; Daisy, M.; Tschumperlé, D.; Lézoray, O. Depth-Guided Disocclusion Inpainting of Synthesized RGB-D Images. *IEEE Trans. Image Process.* 2019, 26, 525–538. [CrossRef]

32. Serrano, A.; Kim, I.; Chen, Z.; DiVerdi, S.; Gutierrez, D.; Hertzmann, A.; Masia, B. Motion parallax for 360° RGBD video. *IEEE Trans. Vis. Comput. Graph.* 2019, 25, 1817–1827. [CrossRef] [PubMed]

33. GiliSoft Video Watermark Removal Tool. Available online: http://www.gilisoft.com/product-video-watermark-removal-tool.htm (accessed on 8 February 2021).

34. Theinpaint. Available online: https://theinpaint.com/download (accessed on 8 February 2021).

35. Wang, Z. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* 2004, 13, 600–612. [CrossRef] [PubMed]

36. Wang, Z.; Bovik, A.C. A universal image quality index. *IEEE Signal Process. Lett.* 2002, 9, 81–84. [CrossRef]