A High-Quality VR Calibration and Real-Time Stitching Framework Using Preprocessed Features

SALEH SAEED\(^{1}\), MUHAMMAD UMER KAKLI\(^{2}\), YONGJU CHO\(^{2,3}\), JEONGIL SEO\(^{3}\), AND UNSANG PARK\(^{1}\), (Member, IEEE)

\(^{1}\)Department of Computer Science and Engineering, Sogang University, Seoul 04107, South Korea
\(^{2}\)Department of Information and Communication Technology, Korea University of Science and Technology, Daejeon 34113, South Korea
\(^{3}\)Immersive Media Section, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

Corresponding author: Yongju Cho (yongjucho@etri.re.kr)

This work was supported by the Electronics and Telecommunications Research Institute (ETRI) funded by the Korean Government through the Research of the Fundamental Media Contents Technologies for Hyper-Realistic Media Space under Grant 20ZH1200.

ABSTRACT Virtual Reality (VR) contents include 360°×180° seamless panoramic videos, stitched from multiple overlapping video streams. Many commercial VR devices use a two-camera rig to capture VR content. These devices have increased radial distortion effects along the seams of the stitching boundary. Moreover, a fixed number of cameras in the camera rig makes the VR system non-scalable. Since the VR experience is directly related to the quality of VR content, it is desirable to create a VR framework that is scalable in terms of the number of cameras attached to the camera-rig and has better geometric and photometric quality. In this paper, we propose an end-to-end VR system for stitching full spherical content. The VR system is composed of camera rig calibration and stitching modules. The calibration module performs a geometric alignment of the camera rig. The stitching module transforms texture from the camera or video stream into the VR stream using lookup tables (LUTs) and blend masks (BMs). In this work, our main contribution is the improvement of stitching quality. First, we propose a feature preprocessing method that filters out inconsistent, error-prone features. Secondly, we propose a geometric alignment method that outperforms state-of-the-art VR stitching solutions. We tested our system on diverse image sets and obtained state-of-the-art geometric alignment. Moreover, we achieved real-time stitching of camera and video streams up to 120 fps at 4K resolution. After stitching, we encode VR content for IP multicasting.

INDEX TERMS High definition video, multimedia systems, real-time systems, virtual reality.

I. INTRODUCTION

The problem of stitching and rendering a panoramic video, from multiple overlapping camera frames, is a well-known and widely sought-after research topic. Panoramic videos offer an immersive experience for users across a wide range of commercially available devices. The availability of these devices helps in real-world applications of human-computer interaction. However, these devices lack quality in terms of stitching alignment, content resolution, and rendering speed. Moreover, commercially available devices for VR content creation and consumption either have a fixed number of cameras [1] usually connected to a pre-aligned camera rig or limited display options. We refer to such devices as non-scalable or limited-scalability devices. It is commercially desirable to design a scalable VR content creation and consumption system that can be reconfigured with a variable number of cameras and different types of local or network [2], [3] based display devices [4].

A. IMMERSIVE CONTENT CREATION

The process of stitching a panoramic canvas from multiple overlapping images is highly dependent on scene texture, camera quality, and type of lenses. Typically, panorama stitching requires a camera to be rotated around its axis in a panning movement, such that there is an overlap between consecutive frames. The variations in scene texture, lighting conditions, overlap area, and rotation axis create a challenging environment for stitching frameworks. These variations become more challenging for multi-camera image
acquisition which exhibits a panning movement of 360°. Stitching immersive panorama demands higher system complexity and economical costs. System complexity can be reduced by using wide-angle lenses with additional constraints for handling non-linearity. There exist several commercially available platforms that can stitch panoramas under dynamic conditions and unconstrained environments.

**B. IMMERSIVE CONTENT CONSUMPTION**

A recent surge in demand for immersive content has resulted in the commercial availability of various kinds of immersive content consumption devices. These devices vary from single-user head-mounted VR systems [5] to wall-mounted group immersion display systems. It requires higher computational complexity and economic costs to render high-resolution content on these devices. Furthermore, different categories of immersive display devices demand a content creation system that can work across multiple categories of display devices [6], [7]. Therefore, the design of a consumer-grade and economically viable real-time 360° stitching system involves an increase in system performance along with reduced cost and complexities.

**C. CONTRIBUTIONS**

In this paper, we present a cost-effective and scalable VR system for stitching immersive content in real-time. The stitching system has a calibration phase followed by a real-time stitching phase. First, we present an algorithm for system calibration that uses a feature preprocessing stage to clean and trim feature matches. After feature preprocessing, an iterative bundle adjustment strategy is used for the alignment of multiple overlapping cameras. Secondly, we provide a real-time stitching module that can provide immersive content on multiple devices at 4K resolution. We achieve real-time stitching with 60 fps from the live camera feed and 120 fps from videos stored on a disk. Our framework can be used locally or over the network that makes it ideal for remote panorama stitching and parallel processing.

**II. RELATED WORK**

Primitive panorama stitching methods used semi-automatic algorithms that required user input for stitching mosaics. Commercially available software was able to stitch panorama with the help of the user’s input. The semi-automatic stitching process was replaced by a fully automated algorithm [8]–[10] with the use of scale-invariant features [8], [11]. Improvements in reliability of a fully automated stitching system paved way for very large-scale multi-camera, multi-display panoramic stitching [12] and rendering systems [13]–[18]. Further advancements in software and hardware resulted in the production of portable devices such as smart-phones and handheld cameras which can create very high-quality panoramic images in real-time [19]. A typical fully automated stitching algorithm [10], [13], [20]–[23] uses rectilinear lenses to capture multiple overlapping images and estimates homographic relationship between them. The homography between these images is used to warp each image into its relative position on a planar, cylindrical or spherical canvas, with seams blended in such a way that the resultant image appears to be a seamless mosaic having field-of-view (FoV) larger than each camera’s FoV.

**A. SPHERICAL PANORAMA STITCHING**

Rectilinear lenses have limited (FoV), therefore for very wide area coverage, multiple cameras with overlapping FoV are required [24]. Recently, the inclusion of virtual and augmented reality in most of the commercially available multimedia devices has increased the demand for content created with wide and ultra-wide-angle lenses such as fisheye. Usage of such lenses to stitch images and videos in modern technologies is particularly helpful in various fields. For example, people use navigation services for commuting from one place to another. The use of maps, generated by a wide-angle lens mounted on a car or a drone and traversed throughout a city, is particularly helpful. In another case, the satellite imagery of different planets and constellations can be stitched together and viewed in an immersive and interactive environment. The use of panoramic videos of live sports matches [25], stitched from wide-angle cameras is also becoming mainstream. Wide-angle lenses capture curvilinear images that can be used for stitching a full spherical panorama with a FoV of 360°×180° [26]–[29]. Although, the wide FoV of these lenses require a smaller number of cameras, the radial distortion results in degraded quality of distortion-corrected images [30]. Less number of required cameras is desirable since it is economical from a commercial perspective. Therefore, the challenge of mitigating lens distortion along with stitching and transmission of a high-quality panoramic stream for VR applications is actively explored by researchers [31]–[33]. There are several commercially available, hardware-based, full spherical stitching solutions [34]–[38]. Most of these solutions employ two cameras with a fisheye lens that works in either 180° or 360° mode. These solutions are not scalable in terms of camera and display configurations. Furthermore, the quality of the seam at the overlap area is not good as they use only two cameras which creates an increased distortion. Moreover, these solutions are very expensive. Software-based state-of-the-art solutions such as [39] and [40] for stitching a full spherical panorama are also commercially available. However, a majority of these solutions are not real-time, and the stitching video exhibits poor frame rates. Lee et al. [41] proposed a real-time high-resolution 360° video foveated stitching method, however the average framerate for 4K stitching is very low.

**B. Panorama Stitching with Bundle Adjustment**

A simple way of stitching multiple images, given geometrically matched features, is by finding a homographic relationship.

A homographic relationship between a pair of matching images represents a geometric transformation from one
image to another such that one of the images is warped onto another to create a single unified view. However, the homographic relationship for a sequence of matching images tends to accumulate geometric errors. In the case of a full spherical panorama, the homography estimation method creates a gap between the first and last camera images [10]. The gap can be removed by applying constraints and using a camera parameter refinement method. Moreover, homography is a linear relationship between two matching images which cannot be used to estimate the relationship between two images with nonlinear distortions such as fisheye images. Bundle adjustment [9], [42], [31] is a method of choice for such problems. It uses Levenberg Marquardt (LM) optimization algorithm [43], to jointly estimate camera intrinsic and extrinsic parameters of all images in the form of a bundle. The parameter estimation is performed by using a cost function. A cost function is a function that minimizes the 2D or 3D distance between matched features while simultaneously estimating camera parameters. The cost function is usually sensitive to noise, therefore a data filtering method [44] is essential for an accurate solution. Bundle adjustment can be performed on all images at once, called a global bundle adjustment, or it can be performed iteratively on a pair of images, called iterative or local bundle adjustment.

In this paper, we stitch multiple video streams, captured from nonlinear sensors, into a 360° × 180° panoramic video using LM optimization. It starts with the extraction of SIFT [8] features followed by matching these features in other cameras frames using K-Nearest Neighbors [45], [46]. Geometrically consistent feature matches are obtained using the Random Sample Consensus (RANSAC) [47]. The matched features are further processed by our proposed feature preprocessing method. The processed feature matches are mapped in a sphere and the distance between them is minimized along with the joint estimation of camera parameters using LM optimization. After estimating camera parameters, we create LUTs for real-time stitching.

The organization of this paper is as follows. The proposed method is discussed in section III. Section III (A) includes a detailed discussion on the calibration module whereas section III (B) includes a detailed discussion on real-time stitching with LUTs and BMs. Section IV contains experimental results. We present the conclusion of our work in section V.

III. PROPOSED METHOD

We propose a VR system by modeling a framework in two phases i.e. calibration phase and stitching phase. In the first phase, the task of system calibration is achieved by estimating the geometric alignment of multiple cameras. We provide six video streams as input to our proposed framework. The video streams are either directly transmitted from cameras at 60fps or loaded from storage disk at over 120fps. The storage disk is an SSD of NVME type which can provide read/write speeds for handling 4K videos at over 120fps. Each video stream is captured using a fisheye lens and represented by $C_{FE} = \{C_{FEi(k)}|i = 1,..,N, k = 1,..\}$, where $N$ is the number of cameras and $k$ is the frame number. We use the geometric alignment parameters to create LUTs [13], [48] $T = \{T_{i}|i = 1,..,N\}$, and BMs $B = \{B_{i}|i = 1,..,N\}$ for each camera. In the second phase, we use these LUTs for mapping texture quads from the camera domain directly onto the panorama domain. The mapping process creates a seamless presentation of the scene on the panoramic canvas by applying linear blending with BMs. We denote $k^{th}$ panorama frame by $P_{(k)}$. The stitched panoramic stream is encoded and transmitted to multiple display devices. The flow of the proposed framework is shown in Fig. 1.

A. CALIBRATION PHASE

In the calibration phase, the camera frames $C_{FEi(k)}$ are transformed from fisheye to equirectangular images $C_{i(k)}$. The camera correspondences are evaluated by extracting scale-invariant features [8] from equirectangular images.
The features from one image are matched across all images [45]. Features extracted from equirectangular images are described by $F = \{F(x_h, d_h)_{h=1,...,H}\}$, where $h$ is the number of extracted features, $x_h \in \mathbb{R}^2$ is the feature location and $d_h$ is SIFT descriptor of $h^{th}$ feature in $k^{th}$ frame of $i^{th}$ camera $C_{(k)}$, respectively. Features from each equirectangular image are matched with other cameras to get overlapping area described by $F$. Features extracted from equirectangular images are matched across all frames from two overlapping cameras, $\{C_{(a)}, C_{(b)}\} \in C$ and $a < b$. The matching features for $CP$ are denoted by $M = \{M_{ab(k)}(F_{a(k)} \times F_{b(k)}, \delta_{l(k)})\} | a < b, l = 1, ...L \}$ where $\delta_{l(k)}$ is the ratio of distances from $l^{th}$ feature in image $a$ to first and second best matched features in image $b$ of a pair using k-nearest neighbor matching.

For each overlapping camera pair, matched features could possibly be dispersed in the form of non-uniform clusters. Since we are using LM Optimization [43] for geometric alignment, which is very sensitive to noisy data, an incorrectly matched feature or a cluster of features may tilt final solution into an incorrect local-minima. Moreover, large number of matched features makes bundle adjustment slower. We propose a feature preprocessing method for handling inconsistent, error prone features before estimating geometric alignment. The feature preprocessing module consists of a grid-based trimming method for removing redundant clusters of matched features and an outlier filtering method for removal of error prone features.

1) GRID TRIMMING

The process of trimming redundant feature matches in an overlap area is described in this section. Consider the 1st frame from two overlapping cameras, $C_{a(1)}$ and $C_{b(1)}$, shown in Fig. 2 (a, b). Assume that there are $L$ feature matches in the overlapping area of the two images, shown in Fig. 2(a). Create a region-of-interest (RoI) in the overlap area, shown in Fig. 2 (b, c). The dimensions of the region will be $RoI(x', y', width, height)$, such that all features should reside inside this region. Create a grid of size $S \times S$, in the RoI, where $S = 3, 6 & 12$, as shown in Fig. 3(b, c, d). For all matched features $F_{l(k)d}$ where $l = 1, 2, ...L$, locate grid box and assign a matched feature to it by scaling $x$ & $y$ coordinates to grid dimensions. If multiple matched features are assigned to a box, the trimming process is accomplished by a selection criterion (γ) such that only one matched feature is selected per grid box.

We propose two criteria for matched feature selection in a grid box. In first selection criterion (γ1), a matched feature pair $(F_{l(1)a}, F_{l(1)b}, \delta_{l(k)})$ with the lowest distance ratio between first and second-best matched features ($\delta_{l(k)}$) is selected, while rest of the matched features in the grid box are removed. In the second selection criteria (γ2), a matched feature pair $(F_{l(1)a}, F_{l(1)b}, \delta_{l(k)})$ is randomly chosen out of all matched features in the grid box. The trimming process is repeated for all camera pairs and the trimmed features are represented by $M_T = \{M_{ab(k)}(F_{a(k)} \times F_{b(k)}, \gamma(1,2)l(k))\} | l = 1, ...L_T \leq L \}$, where $l$ is the number of trimmed feature matches between pair of camera $a$ and camera $b$. Furthermore, we use a distribution score $\theta_{score}$, as in (1), for estimating the quality of distribution of trimmed feature matches along the grid.

$$\theta_{score} = \frac{L_T - L_{min}}{S \times S}$$

where $L_{min}$ is a limiting threshold for the minimum number of trimmed feature matches, $S$ is grid size and $L_T$ is the number of trimmed features. A higher value of $\theta_{score}$ indicates that trimmed feature matches are dispersed more uniformly in each grid box whereas a negative value indicates that the size of the grid or the number of trimmed feature matches is too small. The minimum number of trimmed feature matches must be greater than a limiting threshold $L_{min} = 6$ to get a positive distribution score.

2) OUTLIER FILTERING

The process of trimming redundant features is followed by outlier filtering. Outlier filtering removes loosely matched
features which creates a drift in the iterative bundle adjustment process, causing a degrade in stitching quality. The process of outlier filtering is described in this section. Consider 1st frame from a pair of overlapping cameras, \( C_{a(1)} \) and \( C_{b(1)} \). Assume that there are \( L_T \) trimmed feature matches between the camera \( a \) and camera \( b \) of a pair which are represented by \( M_T = \{ M_{ab(k)}(F_{a(k)} \times F_{b(k)}), y(1/k) \} | l = 1, ..L_T \}. We align feature matches \( M_T \) in a sphere and represents these points by \( X_f = \{ X_{l,a} \in R^3 | l = 1, ..L_T , i \in a, b \}. The spherical distance between \( X_{l,a} \) and \( X_{l,b} \) is \( \delta_l \), as in (2).

\[
\delta_l = \sin^{-1} \left( \frac{X_{l,a} \times X_{l,b}}{||X_{l,a}|| \ ||X_{l,b}||} \right)
\]

By assuming that the spherical distances between trimmed feature matches are distributed normally, we use an outlier filtering threshold \( \tau_f \) in (3), for outlier removal.

\[
\tau_f = \frac{1}{L_T} \sum_{l=1}^{L_T} \delta_l + n \times \sqrt{\frac{1}{L_T} \sum_{l=1}^{L_T} (\delta_l - \frac{1}{L_T} \sum_{l=1}^{L_T} \delta_l)^2}
\]

where \( n \) is the number of standard deviations. The process of outlier removal is repeated for each overlapping camera pair followed by the collective outlier filtering of all camera pairs to undertake the group geometry into account. From (3), the first, second and third standard deviations contain 68%, 95% and 99.7% of total number of datum points, in accordance with 3-sigma rule [49]. Therefore, it is highly likely to have an outlier for \( n = 1, 2&3 \) without much loss of inliers. After filtering outliers, matched inliers between cameras \( a \) and \( b \) are represented by \( M_F = \{ M_{ab(k)}(F_{a(k)} \times F_{b(k)}), y(1/k) \} | l = 1, ..L_F \leq L_T \}, where \( l \) is the number of filtered inliers.

3) GEOMETRIC ALIGNMENT

Geometric alignment is a process of estimating camera intrinsic and extrinsic parameters. The intrinsic parameters are the focal length and principal point. The extrinsic parameters are camera orientations in the form of rotation matrices. If the camera lenses induce radial distortion, the radial correction parameters are also estimated in the geometric alignment process. In a typical geometric alignment process, a view-graph tree of overlapping cameras is created with the help of scale-invariant feature matches. The view-graph tree contains a camera image at each vertex. If two or more cameras have an overlapping scene and matching features, we connect these camera vertices with edges. We assign a weight to each edge, equal to the number of matching features between cameras of connected vertices. We use LM optimization for the estimation of camera parameters. In LM optimization, a residual function that relates the matching features to camera parameters is minimized. We follow an approach similar to [50], where we use a local optimization between individual vertex pairs connected with edges followed by global optimization of all vertices in the view-graph tree. In order to minimize the residual function, first, we map matched scale invariant features between connected vertices into a sphere. We use vertical lines along with scale-invariant features for keeping the geometric alignment upright in the direction of z-axis. Second, we use the objective function for residual minimization, as in (4), which is used to minimize the distance of matched features between camera \( a \) and camera \( b \).

\[
e = \arg\min_{\beta} \left( \frac{1}{L_F} \sum_{l=1}^{L_F} \delta_l \right)
\]

where \( \delta_l \), as in (2), is the spherical distance between a pair of matching features \( X_{l,a(k)} \) and \( X_{l,b(k)} \), \( L_F \) is the number of preprocessed features, \( X_{l,a(k)} \) and \( X_{l,b(k)} \) are feature vectors mapped in a sphere, corresponding to matching pairs of features \( X_{l,a(k)} \) and \( X_{l,b(k)} \), from camera \( a \) and \( b \). In order to minimize the distance between pairs of matching features, we map pairs in a sphere with spherical transformation function \( F(X_{l,a(b,k)} , \beta) \), where \( \beta \) represents camera parameters. We denote the transformed feature points with \( X_{l,a(b,k)} \). Assume that the \( I^{th} \) feature from camera \( b \) is mapped on the sphere at point \( X_{l,b(k)} \) with transformation \( F(X_{l,b(k)} , \beta_b) \). The transformation function \( F(X_{l,a(k)} , \beta_a) \) transforms \( I^{th} \) feature from camera \( a \) to a point \( X_{l,a(k)} \), in spherical domain using camera parameters \( \beta_a \). We use (4) as a cost function in LM optimization and minimize the residual error \( e \). The residual error is jointly minimized for all pairs of matching features in the form of a bundle by varying the camera parameters. The camera parameters \( \beta = \{ \beta_i | i = 1, ..N & a, b \in i \} \) are classified into two categories i.e. intrinsic and extrinsic parameters. We assume that the intrinsic parameters, which
include focal length, principal point and radial distortion [51], are shared by all cameras. The extrinsic parameters are assumed to be unique to each camera, and are represented by $R = \{R_i \in SO(3)\}$. The camera parameters $\beta$ are estimated in four stages:

a. Create a view-graph tree of all cameras. Assign a camera to each vertex. If a pair of cameras have an overlapping scene and matched scale invariant features, connect their vertices with edges. Assign a weight to each edge equal to the number of matched features between the camera vertices.

b. Select an initial pair of cameras to begin local geometric alignment. Select a pair of camera vertices that have a maximum weight assigned to their edge. Map the matched features on a sphere and use (4) to jointly minimize the residual error with the estimation of camera parameters.

c. Sort vertices connected to the vertices of the initial camera pair from step b. Select a vertex connected to the initial pair that has maximum weight. Map feature matches of the camera of the selected vertex in a sphere. Use (4) to jointly minimize the residual error with the estimation of the camera parameters of a newly selected camera. Repeat this process for all vertices one by one.

d. Use local geometric alignments from steps b and c as a starting point and perform a global geometric alignment of cameras from all vertices using (4) to jointly minimize the residual error with the refinement of camera parameters of all cameras in the view-graph tree.

The pseudo-code for calibration phase is added in Algorithm 1.

### Algorithm 1 Calibration Phase in VR Framework

**Input:** Equirectangular frames $C_{i(k)}$ obtained from camera rig in an unknown order where $i$ is camera number and $k$ is frame number such that $i = 1 to 6$ and $k = 1$. (The camera frames are transformed from fisheye to equirectangular domain).

**Output:** Camera calibration parameters $\beta_i$, LUTs $T_i$ and BMs $B_i$.

**Variables:** $F_{i(k)}$ are SIFT features in $k^{th}$ frame of $i^{th}$ camera, $M_{ab(k)}$ are matched features obtained from k-d tree based nearest neighbor matching, $\delta_{i(k)}$ is the distance ratio of first and second best matched $i^{th}$ features in cameras $a$ and $b$, $S$ is the size of grid used for trimming, $\gamma$ is the selection criterion in trimming process, $n$ is the outlier filtering parameter, $X_{i(k)}$ are matched features mapped in a sphere and $V$ is the viewpoint.

**Algorithm**

1. \[ function Preprocess(M_{ab(k)}, \delta_{i(k)}): \]
2. \[ M_{ab(k)} = GridTrim(M_{ab(k)}, \delta_{i(k)}, S, \gamma) \]
3. \[ M_{ab(k)} = FilterOutliers(M_{ab(k)}, n) \]
4. \[ return M_{ab(k)} \]
5. \[ function Align(M_{(k)}) \]
6. \[ X_{(k)} = MapFeaturesOnSphere(M_{(k)}) \]
7. \[ V = CreateViewGraph(M_{(k)}) \]
8. \[ while pair exists in V : \]
9. \[ V_{local} = SelectPair(V, X_{(k)}) \]
10. \[ \beta = LocalAlignment(V_{local}, X_{(k)}) \]
11. \[ end \]
12. \[ \beta = GlobalAlignment(V, X_{(k)}) \]
13. \[ return \beta \]
14. \[ function Calibrate(C_{(k)}): \]
15. \[ for i = 1 to 6 do \]
16. \[ F_{i(k)} = ExtractFeatures(C_{i(k)}) \]
17. \[ end \]
18. \[ \\
19. \[ for a = 1 to 5 do \]
20. \[ M_{ab(k)}(\delta_{l(k)}) = MatchFeatures(F_{a(k)}, F_{b(k)}) \]
21. \[ M_{ab(k)} = Preprocess(M_{ab(k)}, \delta_{l(k)}) \]
22. \[ end \]
23. \[ M_{ab(k)} = FilterGlobalOutliers(M_{(k)}, n) \]
24. \[ \beta = Align(M_{(k)}) \]
25. \[ for i = 1 to 6 do \]
26. \[ T_{i} = CreateLUT(C_{i(k)}, \beta_{i}) \]
27. \[ B_{i} = CreateBM(C_{i(k)}, \beta_{i}) \]
28. \[ end \]
29. \[ return \beta, T, B \]

on a panoramic canvas [13], [48]. For this purpose, we use OpenGL routines and achieve real-time rendering speeds of up to 120fps for $4K \times 2K$ pixels. LUT contains four coordinate correspondences per quadrilateral between camera and panorama canvas. The number of quadrilaterals per camera frame, along the horizontal or vertical axis, is equal to the greatest common divisor of width and height of camera frame, denoted by $Q = \text{GCD}(W, H)$ whereas the width...
and height of each quadrilateral will be \( q_w = Q/W \) and \( q_h = Q/H \), where \( W \) and \( H \) are the width and height of camera frames. We use inverse mapping from the panoramic canvas to each camera frame to avoid forward mapping holes. Assume that the top left corner of each quadrilateral in the panoramic canvas, is denoted by \( P_{1,q}^i \) where \( q \in Q^2 \). Therefore, for a quadrilateral with the top left corner at \( P_{1,q}^i \), the top right, bottom left, and bottom right corners will be \( P_{1,2}^i \), \( P_{1,1+q}^i \) and \( P_{1,2+q}^i \). We use the inverse of calibration parameters \( \beta \) for each camera to transform these corner points to each camera frame, as in (6).

\[
P_{i,q}^i = \sum_{q=1}^{N} Q \times Q T_i P \rightarrow C(P_{i,q}^i)
\]

where \( P_{i,q}^i \) is the top left corner of \( q^{th} \) quadrilateral in the panoramic canvas. We save this mapping information in the form of LUTs. We assign one LUT per camera and denote it by \( T = \{T_i|i = 1,..N\} \). Camera frame texture mapped on a panoramic canvas with LUTs, lacks spatial coherence. The spatial incoherence is caused by abrupt changes in overlap area texture. We create a smooth fading effect across overlap area texture by applying a BM on each transformed camera frame. The BMs are two-dimensional intensity layers, having the same size as the panoramic canvas. We denote the BMs \( B^i \) where \( i = 1,..N \). The stitched panorama is obtained by multiplying the BMs to geometrically transformed camera frames and summing up on a canvas, as in (7).

\[
P_{k} = \sum_{i=1}^{N} B_i T_i(C_{FEi(k)})
\]

where \( k^{th} \) camera frame from \( i^{th} \) camera is denoted by \( C_{FEi(k)} \) and transformed by LUT \( T_i \) and BM \( B_i \).

2) CONTENT RENDERING

The LUTs and BMs are created only once using 1st frames, at the initialization of the VR calibration and stitching system. The rest of the frames are rendered directly on the canvas. We get a frame rate of 60 or over 120 panorama frames per second depending on the input, camera feed, or storage disk. The frames are shared via the network to multiple display devices. For network sharing, we encode panoramic frames with H.264. The transmitted stream can be consumed across different media such as YouTube, VR headset, or custom designed displays. In the next section, we demonstrate the superiority of our calibration and stitching system.

IV. EXPERIMENTS

We test our system with six video streams. The video streams are either acquired directly from a camera rig, shown in Fig. 4(a, b), at 60fps or loaded from a storage disk at 120fps. Each video stream is captured with a camera having a fisheye lens with a vertical FoV of 180° and horizontal FoV ~75°. The resolution of each camera frame is full HD and frames are captured in portrait mode. We capture frames with an overlap of approximately 10%-25% with neighboring cameras. These frames are stitched into a full spherical panorama with horizontal and vertical FoV of 360 and 180 degrees respectively. The calibration system is equipped with a quad-core processor at 3.3GHz-3.9GHz and 64GB DDR4 RAM. The calibration code is written in C++. For our VR calibration and stitching system, we face these research questions;
How to improve stitching quality with similar or better calibration time?
How to design an efficient end-to-end framework for stitching VR content?

To look for the solution of these questions, we use 14 sets of camera frames captured at distinct places. These image sets have diverse illumination, texture, and scene variations to provide a challenging environment for the calibration and stitching process. We demonstrate improvements in the calibration phase by first trimming unwanted clusters of feature matches. Secondly, we filter the outlying error-prone feature matches. We create 1680 trials per camera pair out of which 672 trials are used to assess grid trimming whereas 1008 trials are used to evaluate the outlier filtering method.
TABLE 1. Comparison of different feature preprocessing configurations.

| Img. Set | Features Extracted (Per Image) | GRID | Dist. Score | SSIM | Feature Inliers (Per Pair) | γ₁⁺ | Projection Error (px) | Calibration Time (s) |
|---------|---------------------------------|------|-------------|------|---------------------------|-----|----------------------|---------------------|
| 1       | 4538                            | 3x3  | 0.11        | 0.80 | 0.80                      | 0.81| 0.80                 | 0.81                |
|         |                                 | 6x6  | 0.47        | 0.81 | 0.80                      | 0.81| 0.80                 | 0.80                |
|         |                                 | 12x12| 0.41        | 0.80 | 0.81                      | 0.81| 0.80                 | 0.80                |
| 2       | 4333                            | 3x3  | 0.59        | 0.82 | 0.81                      | 0.82| 0.81                 | 0.81                |
|         |                                 | 6x6  | 0.59        | 0.82 | 0.81                      | 0.82| 0.81                 | 0.81                |
|         |                                 | 12x12| 0.29        | 0.81 | 0.82                      | 0.82| 0.81                 | 0.81                |
| 3       | 3152                            | 3x3  | 0.00        | 0.78 | 0.78                      | 0.78| 0.78                 | 0.78                |
|         |                                 | 6x6  | 0.28        | 0.78 | 0.79                      | 0.78| 0.79                 | 0.78                |
|         |                                 | 12x12| 0.22        | 0.79 | 0.80                      | 0.78| 0.80                 | 0.78                |
| 4       | 5155                            | 3x3  | 0.33        | 0.81 | 0.80                      | 0.81| 0.81                 | 0.81                |
|         |                                 | 6x6  | 0.33        | 0.81 | 0.80                      | 0.81| 0.81                 | 0.81                |
|         |                                 | 12x12| 0.29        | 0.80 | 0.81                      | 0.81| 0.81                 | 0.81                |
| 5       | 2461                            | 3x3  | -0.11       | 0.84 | 0.83                      | 0.84| 0.83                 | 0.84                |
|         |                                 | 6x6  | 0.11        | 0.85 | 0.84                      | 0.84| 0.84                 | 0.84                |
|         |                                 | 12x12| 0.11        | 0.82 | 0.84                      | 0.84| 0.84                 | 0.84                |
| 6       | 1392                            | 3x3  | -0.22       | N/A  | 0.84                      | 0.85| 0.84                 | 0.85                |
|         |                                 | 6x6  | 0.14        | 0.86 | 0.85                      | 0.85| 0.85                 | 0.85                |
|         |                                 | 12x12| 0.10        | 0.84 | 0.85                      | 0.85| 0.85                 | 0.85                |
| 7       | 4243                            | 3x3  | 0.00        | 0.76 | 0.78                      | 0.79| 0.78                 | 0.79                |
|         |                                 | 6x6  | 0.31        | 0.80 | 0.79                      | 0.79| 0.79                 | 0.79                |
|         |                                 | 12x12| 0.24        | 0.78 | 0.79                      | 0.79| 0.79                 | 0.79                |
| 8       | 2898                            | 3x3  | 0.11        | 0.75 | 0.77                      | 0.77| 0.77                 | 0.77                |
|         |                                 | 6x6  | 0.36        | 0.79 | 0.78                      | 0.78| 0.78                 | 0.78                |
|         |                                 | 12x12| 0.26        | 0.76 | 0.78                      | 0.77| 0.77                 | 0.77                |
| 9       | 3129                            | 3x3  | 0.11        | 0.77 | 0.78                      | 0.79| 0.78                 | 0.79                |
|         |                                 | 6x6  | 0.37        | 0.81 | 0.79                      | 0.80| 0.79                 | 0.80                |
|         |                                 | 12x12| 0.28        | 0.77 | 0.80                      | 0.79| 0.80                 | 0.79                |
| 10      | 2370                            | 3x3  | 0.00        | 0.82 | 0.85                      | 0.85| 0.85                 | 0.85                |
|         |                                 | 6x6  | 0.22        | 0.87 | 0.85                      | 0.86| 0.85                 | 0.86                |
|         |                                 | 12x12| 0.18        | 0.84 | 0.85                      | 0.85| 0.85                 | 0.85                |
| 11      | 2216                            | 3x3  | 0.00        | 0.87 | 0.89                      | 0.90| 0.89                 | 0.90                |
|         |                                 | 6x6  | 0.25        | 0.90 | 0.89                      | 0.90| 0.89                 | 0.90                |
|         |                                 | 12x12| 0.13        | 0.89 | 0.90                      | 0.90| 0.90                 | 0.90                |
| 12      | 3458                            | 3x3  | 0.00        | 0.74 | 0.76                      | 0.77| 0.76                 | 0.77                |
|         |                                 | 6x6  | 0.33        | 0.78 | 0.78                      | 0.78| 0.78                 | 0.78                |
|         |                                 | 12x12| 0.22        | 0.74 | 0.76                      | 0.75| 0.76                 | 0.75                |
| 13      | 1670                            | 3x3  | -0.11       | 0.87 | 0.89                      | 0.89| 0.89                 | 0.89                |
|         |                                 | 6x6  | 0.17        | 0.91 | 0.90                      | 0.90| 0.90                 | 0.90                |
|         |                                 | 12x12| 0.15        | 0.88 | 0.90                      | 0.90| 0.90                 | 0.90                |
| Avg.    | 3004                            | 3x3  | 0.00        | 0.81 | 0.82                      | 0.82| 0.82                 | 0.82                |
|         |                                 | 6x6  | 0.28        | 0.84 | 0.83                      | 0.83| 0.83                 | 0.83                |
|         |                                 | 12x12| 0.22        | 0.81 | 0.83                      | 0.83| 0.83                 | 0.83                |

A. IMPROVEMENTS WITH FEATURE PREPROCESSING

In an image set with 6 images, a single image has an average of 3004 SIFT features and 194 matched features per pair. The process of finding alignment parameters for a camera pair takes 174 seconds with an average error of 3.24 pixels. We improve the alignment quality of stitched panorama by using a grid-based trimming and outlier filtering method. For this purpose, we first perform grid trimming at 3 different sizes i.e. 3, 6 and 12, and assign matched features to corresponding grid boxes.

We select one pair of matched features per grid box with matched feature selection methods i.e. Distance Ratio Criteria (γ₁) and Random Selection Criteria (γ₂). In Fig. 5(a), a comparison of distance ratio γ₁ and random selection criteria γ₂ shows that distance ratio criteria γ₁ performs better than random selection criteria γ₂ in terms of average error. Therefore, we select γ₁ criteria in grid trimming.

In Fig. 5(b), we compare the distribution score θ with grid size 3, 6, and 12. The θ score, for a grid of size 3 occasionally drifts to negative value whereas, for the grid of size 6, we get the highest score in comparison to the score for a grid of size 12. The average distribution scores for grid 3, 6, and 12 are 0.00, 0.28, and 0.22. Therefore, we select grid size 6 for our framework. To find outliers among trimmed feature

VOLUME 8, 2020

190308
pairs, we geometrically align feature matches in a sphere. We assume that for all matched features in overlapping camera pairs, the distances between corresponding features in a spherical domain are distributed normally. The outliers are detected and removed by calculating the mean and standard deviation of the distance between matched features. We have used a maximum filtering threshold $\tau_f$ as in (3), for valid inlier selection. We have tested the calibration system with $n = 1, 2, 3$, where $n$ is the number of standard deviations. The filtering of outliers with $n = 1$ results in the removal of $\sim 32\%$ of matched features which causes a significant decrease in projection error. In this case, the decrease of projection error might not signal an improvement in alignment quality since projection error does not guarantee that the matched features are dispersed uniformly. In order to select the value of $n$, we use a structural similarity measure, SSIM [52]. The value of the SSIM index ranges from 0 to 1 where a higher value indicates better stitching quality. In Table 1, we compare the projection error, SSIM [52], calibration time, and the number of inliers, by calibrating 14 image sets, for $n = 1, 2, 3$ and grid sizes 3, 6, and 12 with distance ratio selection criteria $\gamma_1$.

We get the highest SSIM score for $n = 1$ and grid size 6. The projection error corresponding to the highest score of SSIM is 2.26 pixels. Although 2.26 pixels is second-best average projection error, we will select this configuration for our framework. With the selected configuration, we obtain 30.25% improvement in stitching quality as compared to the panoramas obtained without any trimming and filtering. In Fig. 6, we compare the calibration performance of our framework with and without grid trimming and outlier filtering. The number of inliers after outlier filtering with $n = 1, 2, 3$, and grid trimming with size 6 is shown in Fig. 6(a). In Fig. 6(b) and 6(c), the comparison of projection error and calibration time is shown.

In Table 2, we compare the performance of the proposed feature preprocessing method on our VR system. First, we use all the matched features to estimate the geometric alignment of 14 image sets. The average projection error for geometric alignment with all matched features is 3.24 pixels. Secondly, we use our proposed feature preprocessing method to process feature matches. The geometric alignment obtained from preprocessed feature matches is significantly improved with an average projection error of 2.26 pixels.

### TABLE 2. Performance comparison of preprocessed features.

| Inlg. Set | Features Extracted (Per Image) | Preprocess Feature | SSIM Feature | Error (px) | Calibration Time (s) |
|-----------|-------------------------------|-------------------|--------------|------------|---------------------|
| 1         | Yes                           | 0.81              | 18           | 2.19       | 79.53               |
|           | No                            | 0.78              | 323          | 3.01       | 342.5              |
| 2         | Yes                           | 0.82              | 15           | 2.78       | 75.52               |
|           | No                            | 0.80              | 183          | 3.65       | 196.8              |
| 3         | Yes                           | 0.78              | 12           | 2.57       | 49.44               |
|           | No                            | 0.78              | 222          | 3.01       | 195.7              |
| 4         | Yes                           | 0.81              | 13           | 2.90       | 54.33               |
|           | No                            | 0.80              | 359          | 2.96       | 194.4              |
| 5         | Yes                           | 0.85              | 165          | 3.43       | 49.06               |
|           | No                            | 0.83              | 165          | 3.88       | 110.8              |
| 6         | Yes                           | 0.86              | 8            | 2.58       | 56.55               |
|           | No                            | 0.82              | 94           | 3.62       | 170.8              |
| 7         | Yes                           | 0.80              | 13           | 2.69       | 52.79               |
|           | No                            | 0.77              | 239          | 3.66       | 162.8              |
| 8         | Yes                           | 0.79              | 13           | 3.17       | 55.56               |
|           | No                            | 0.75              | 259          | 4.29       | 158.3              |
| 9         | Yes                           | 0.81              | 14           | 2.32       | 57.50               |
|           | No                            | 0.79              | 236          | 3.17       | 240.5              |
| 10        | Yes                            | 0.87              | 10           | 1.97       | 54.03               |
|           | No                            | 0.83              | 165          | 2.53       | 127.5              |
| 11        | Yes                            | 0.90              | 12           | 2.62       | 63.00               |
|           | No                            | 0.88              | 79           | 4.20       | 154.2              |
| 12        | Yes                            | 0.78              | 12           | 2.61       | 56.55               |
|           | No                            | 0.75              | 145          | 4.30       | 110.0              |
| 13        | Yes                            | 0.91              | 9            | 0.84       | 41.18               |
|           | No                            | 0.89              | 145          | 1.64       | 127.4              |
| 14        | Yes                            | 0.90              | 8            | 0.98       | 48.05               |
|           | No                            | 0.90              | 106          | 2.42       | 143.9              |
| Avg.      | Yes                            | 0.84              | 12           | 2.26       | 56.65               |
|           | No                            | 0.81              | 194          | 3.24       | 174.0              |

*B. COMPARISON WITH COMMERCIAL SOLUTIONS*

In Table 3, we compare our framework with state-of-the-art VR stitching solutions i.e. Pro 11.28 version of [39] (Trail Version) and [40] version 2019.2.0. We use the average projection error as a measure of quality for this comparison, where a lower value indicates better alignment. We use our framework with $n = 1$ and grid size 6, which is the configuration corresponding to the highest SSIM score, to obtain average projection error for 14 image sets. We stitch the same image sets with [39] and [40] and reported the projection error in Table 3. The highest average projection error is 5.94 pixels obtained from [40]. On the contrary, [39] produced relatively good alignment for all image sets with an average projection error of 2.34 pixels.

### TABLE 3. Performance comparison with commercial solutions.

| Image Set | Projection Error (px) | Calibration Time (s) |
|-----------|-----------------------|----------------------|
| A*        | B*                    | Our                  |
| Avg.      | 5.94                  | 2.26                 |

*B. COMPARISON WITH COMMERCIAL SOLUTIONS*

In Table 3, we compare our framework with state-of-the-art VR stitching solutions i.e. Pro 11.28 version of [39] (Trail Version) and [40] version 2019.2.0. We use the average projection error as a measure of quality for this comparison, where a lower value indicates better alignment. We use our framework with $n = 1$ and grid size 6, which is the configuration corresponding to the highest SSIM score, to obtain average projection error for 14 image sets. We stitch the same image sets with [39] and [40] and reported the projection error in Table 3. The highest average projection error is 5.94 pixels obtained from [40]. On the contrary, [39] produced relatively good alignment for all image sets with an average projection error of 2.34 pixels.

### TABLE 3. Performance comparison with commercial solutions.

| Image Set | Projection Error (px) | Calibration Time (s) |
|-----------|-----------------------|----------------------|
| A*        | B*                    | Our                  |
| Avg.      | 5.94                  | 2.26                 |

*Commercial solutions are referred as A and B, where A is [40] and B is [39].

This software uses integrated/external GPU for image stitching.

Our method is calibrated with $n = 1$ & grid size 6. Using a grid size 6, we get average calibration time comparable to [40] as shown in Table 1.
We obtained a 3.42% improvement with our framework as compared to [39] with an average projection error of 2.26 pixels. A visual comparison of the stitched panoramas using our framework, [39] and [40] is shown in Fig. 7. The individual camera frames of the image set 8 are shown in Fig. 7(a). The panoramas stitched with our framework, [39] and [40] are shown in Fig. 7(b), 7(c), and 7(d). A visual comparison of 14 image sets for our framework, [39] and [40] is added in the supplementary material.

V. CONCLUSION
In this paper, we have proposed a framework for stitching VR content. The proposed system performs system calibration and stitching of 4K contents followed by rendering in real-time across a multitude of devices. We have tested the proposed system on a comprehensive set of images with challenging texture, illumination, and background variations. The proposed system achieves state-of-the-art stitching quality, as compared to commercial solutions, with an average projection error of 2.26 pixels. The stitched panoramic stream is rendered via multicasting, at up to 120 fps. The calibration phase of the proposed framework is relatively slower to obtain initial camera rig alignment. This limitation can be solved by porting it to GPU in the future.

REFERENCES
[1] T. Bertel, N. D. F. Campbell, and C. Richardt, “MegaParallax: Casual 360° panoramas with motion parallax,” IEEE Trans. Vis. Comput. Graphics, vol. 25, no. 5, pp. 1828–1835, May 2019, doi: 10.1109/TVCG.2019.2988799.
[2] J. Hwang, J. Lee, N. Choi, and C. Yoo, “HdAVS: Hybrid adaptive video streaming for mobile devices,” IEEE Trans. Consum. Electron., vol. 60, no. 2, pp. 210–216, May 2014, doi: 10.1109/TCE.2014.6851996.
[3] C. Yao, J. Xiao, Y. Zhao, and G. Li, “Video streaming adaptation strategy for multiview navigation over DASH,” IEEE Trans. Broadcast., vol. 65, no. 3, pp. 521–533, Sep. 2019, doi: 10.1109/TBC.2018.2871370.
[4] J. Cubelos, P. Carballa, J. Gutierrez, and N. Garcia, “QoE analysis of dense multiview video with head-mounted devices,” IEEE Trans. Multimedia, vol. 22, no. 1, pp. 69–81, Jan. 2020, doi: 10.1109/TMM.2019.2924575.
[5] R. Arkin, J. Fooko, M. Sperling, and D. K. Pai, “Perception of looming motion in virtual reality egocentric interception tasks,” IEEE Trans. Vis. Comput. Graphics, vol. 25, no. 10, pp. 3042–3048, Oct. 2019, doi: 10.1109/TVCG.2018.2859987.
[6] T. Rhee, S. Thompson, D. Medina, R. Denis, and A. Chalmers, “Augmented virtual teleportation for high-fidelity telecollaboration,” IEEE Trans. Vis. Comput. Graphics, vol. 26, no. 5, pp. 1923–1933, May 2020, doi: 10.1109/TVCG.2020.2973065.
[7] J. Ratcliﬀ, A. Suitnik, S. Alfaro, and R. Azuma, “ThinVR: Heterogeneous microarrays for compact, 180 degree FOV VR near-eye displays,” IEEE Trans. Vis. Comput. Graphics, vol. 26, no. 5, pp. 1981–1990, May 2020, doi: 10.1109/TVCG.2020.2973064.
[8] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, Nov. 2004.
[9] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” Int. J. Comput. Vis., vol. 74, no. 1, pp. 59–73, Apr. 2007.
[10] R. Szeliski, “Image alignment and stitching: A tutorial,” Found. Trends Comput. Graph. Vis., vol. 2, no. 1, pp. 1–104, 2007, doi: 10.1561/0600000009.
[11] Y. Cho, D. Kim, S. Saeed, M. U. Kakli, S.-H. Jung, J. Seo, and U. Park, “Keypoint detection using higher order Laplacian of Gaussian,” IEEE Access, vol. 8, pp. 2169–3536, 2020, doi: 10.1109/ACCESS.2020.2965169.
[12] K. Krishnakumar and S. I. Gandhi, “Video stitching using interacting multiple model based feature tracking,” Multimedia Tools Appl., vol. 78, no. 2, pp. 1375–1397, Jan. 2019.
[13] S. Saeed, R. Hafiz, A. Rasul, M. M. Khan, Y. Cho, U. Park, and J. Cha, “A unified panoramic stitching and multi-projector rendering scheme for immersive panoramic displays,” Displays, vol. 40, pp. 78–87, Dec. 2015, doi: 10.1016/j.displa.2015.06.002.
[14] M. U. Kakli, Y. Cho, and J. Seo, “Minimization of parallax artifacts in video stitching for moving foregrounds,” IEEE Access, vol. 6, pp. 57763–57777, 2018, doi: 10.1109/ACCESS.2018.2871685.
[15] Y.-S. Kang and Y.-S. Ho, “An efficient image rectification method for parallel multi-camera arrangement,” IEEE Trans. Consum. Electron., vol. 57, no. 3, pp. 1041–1048, Aug. 2011, doi: 10.1109/TCE.2011.6018853.
[16] E. Quevedo, J. de la Cruz, G. M. Callico, F. Tobajas, and R. Sarmiento, “Video enhancement using spatial and temporal super-resolution from a multi-camera system,” IEEE Trans. Consum. Electron., vol. 60, no. 3, pp. 420–428, Aug. 2014, doi: 10.1109/TCE.2014.6937326.
[17] B.-S. Kim, K.-A. Choi, W.-J. Park, S.-W. Kim, and S.-J. Ko, “Content-preserving video stitching method for multi-camera systems,” IEEE Trans. Consum. Electron., vol. 63, no. 2, pp. 109–116, May 2017, doi: 10.1109/TCE.2017.014841.
[18] H.-K. Kim, K.-W. Lee, J.-Y. Jung, S.-J. Jung, and S.-J. Ko, “A content-aware image stitching algorithm for mobile multimedia devices,” IEEE Trans. Consum. Electron., vol. 57, no. 4, pp. 1875–1882, Nov. 2011, doi: 10.1109/TCE.2011.6131166.
[19] Y. Xiong and K. Pulii, “Fast panorama stitching for high-quality panoramic images on mobile phones,” IEEE Trans. Consum. Electron., vol. 56, no. 2, pp. 298–306, May 2010, doi: 10.1109/TCE.2010.5505931.
[20] W. Jiang and J. Gu, “Video stitching with spatial-temporal content-preserving warping,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Boston, MA, USA, Jun. 2015, pp. 42–48.
[21] Y. Nie, T. Su, Z. Zhang, H. Sun, and G. Li, “Dynamic video stitching via shakiness removing,” IEEE Trans. Image Process., vol. 27, no. 1, pp. 164–178, Jan. 2018, doi: 10.1109/TIP.2017.2736603.
[22] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, “High-level multiple-UVU cinematography tools for covering outdoor events,” IEEE Trans. Broadcast., vol. 65, no. 3, pp. 627–635, Sep. 2019, doi: 10.1109/TBC.2019.2892585.
[23] H. Afshari, V. Popovic, T. Tasci, A. Schmid, and Y. Leblebici, “A spherical multi-camera system with real-time omnidirectional video acquisition capability,” IEEE Trans. Consum. Electron., vol. 58, no. 4, pp. 1110–1118, Nov. 2012, doi: 10.1109/TCE.2012.6414975.
[24] J. Seo, Y. Cho, S. Saeed, and U. Park, “Keypoint detection using higher order Laplacian for compact, 180 degree FOV VR near-eye displays,” IEEE Trans. Consum. Electron., vol. 63, no. 2, pp. 109–116, May 2017, doi: 10.1109/TCE.2017.014841.
[25] Y. Nie, T. Su, Z. Zhang, H. Sun, and G. Li, “Dynamic video stitching via shakiness removing,” IEEE Trans. Image Process., vol. 27, no. 1, pp. 164–178, Jan. 2018, doi: 10.1109/TIP.2017.2736603.
SALEH SAED received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Lahore, Pakistan, in 2011, and the M.Sc. degree in electrical engineering from the National University of Sciences and Technology, Pakistan, in 2015. He is currently pursuing the Ph.D. degree with Soyang University, South Korea. From 2014 to 2015, he was a Research Assistant with the Vision, Imaging, and Signal Processing Laboratory, NUST. His research interests include image processing, computer vision, and machine learning.

MUHAMMAD UMER KAKLI received the B.S. degree in electronic engineering from International Islamic University, Pakistan, in 2009, and the M.S. degree in electrical engineering from the National University of Sciences and Technology (NUST), Pakistan, in 2013. He is currently pursuing the Ph.D. degree with the Korea University of Science and Technology, Daejeon, South Korea. From 2013 to 2015, he was a Research Associate with the Vision, Imaging, and Signal Processing Laboratory, NUST. His research interests include image and video processing, multiview imaging, and computational photography.

YONGJU CHO received the B.S. and M.S. degrees in electrical and computer engineering from Iowa State University, in 1997 and 1999, respectively, and the Ph.D. degree in electrical and computer engineering from Michigan State University, in 2003. He joined the Electronics and Telecommunications Research Institute, Daejeon, South Korea, in 2001. He was engaged in development of a data broadcasting systems, multimedia networking, and immersive media. His research interests include digital signal processing, adaptive wireless video communications, and computer vision.

JEONGIL SEO was born in Goryeong, South Korea, in 1971. He received the M.S. and Ph.D. degrees in electronics from Kyungpook National University (KNU), Daegu, South Korea, in 1996 and 2005, respectively. He was a Senior Member of engineering staff with the Laboratory of Semiconductor, LG-Semicon, Cheongju, South Korea, from 1998 to 2000. He has been a Managing Director with the Tera-Media Research Group, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, since 2000. His recent research interests include immersive video (panorama, 360 video, light field, and so on) processing, immersive audio processing, and realistic broadcasting systems.

UNSANG PARK (Member, IEEE) received the B.S. and M.S. degrees in materials science from Hanyang University, Seoul, South Korea, in 1998 and 2000, respectively, and the M.S. and Ph.D. degrees in computer science from Michigan State University, East Lansing, MI, USA, in 2004 and 2009, respectively. He is currently an Associate Professor with the Department of Computer Science and Engineering, Soyang University. His research interests include pattern recognition, image processing, computer vision, and machine learning.