Efficient Computing for Immersive VR: Real-Time Panoramic Video Stitching on GPU Platforms

Karthik PM
Department of Computer Science and Engineering,
Amrita School of Computing,
Amrita Vishwa Vidyapeetham
Amritapuri, India
karthikpm52@gmail.com

Adarsh R Nair
Department of Computer Science and Engineering,
Amrita School of Computing,
Amrita Vishwa Vidyapeetham
Amritapuri, India
adharshrnair@gmail.com

Joshua Wilson Philip
Department of Computer Science and Engineering,
Amrita School of Computing,
Amrita Vishwa Vidyapeetham
Amritapuri, India
joshuawilsonp@gmail.com

Uma Gopalakrishnan
Center for Wireless Networks & Applications (WNA)
Amrita Vishwa Vidyapeetham
Amritapuri, India
umag@am.amrita.edu

***Abstract*—This paper delves into the advancement of live panoramic video stitching for Virtual Reality, to seamlessly merge different video feeds in real-time to enhance immersive VR experiences. The research focuses on real-world applications like deep-sea exploration and self-driving cars, where real-time first-person video from vehicles is crucial. The aim is to design an optimized video stitching application using general-purpose off-the-shelf components. The stitching pipeline makes use of SIFT for feature extraction, enabling matching across multiple images and the estimation of the Homography matrix using RANSAC. The study delves into CPU and GPU techniques, utilizing CUDA and OpenGL, to pinpoint the ideal solution for real-time, immersive VR video generation. Post-stitching, a stereo vision module enriches depth perception when using VR headgear. Video stitching with OpenGL achieved impressive computational speed and excellent frame rate, showing its potential for real-time VR video generation. OpenGL outperformed CPU by 17x (450fps), significantly boosting frame rates for real-time VR video generation.**

***Index Terms*—Virtual Reality, Computer Vision, Panaromic Video Stitching, OpenGL, CUDA**

I. Introduction

Panoramic video stitching in virtual reality (VR) is a vital technology that seamlessly blends video feeds from diverse cameras, creating an immersive experience. Leveraging classical image stitching algorithms, it aligns and connects individual frames to ensure a cohesive VR encounter. Beyond its versatility in various VR applications, this technique proves invaluable in specific contexts such as deep-sea exploration and self-driving cars. In deep-sea exploration, cameras around a submarine capture live feeds, when seamlessly stitched in real-time, allows operators to wear VR headsets for enhanced navigation and decision-making. Similarly, in self-driving cars, panoramic video stitching integrates feeds from cameras around the vehicle, providing an immersive first-person view (FPV) through VR headsets, ultimately contributing to improved situational awareness and safety in autonomous driving systems.

Off-the-shelf video stitching tools like Hugin, AutoStitch, Panorama Factory etc [6], have been instrumental in panorama generation. However, these tools are primarily designed for post-production, lack the real-time capabilities. Facebook Surround 360 enables real-time 360-degree video stitching[14]. Their fixed camera arrangement hinders its application to dynamic scenarios, such as placing cameras around vehicles.

Active research in GPU-based parallel processing for real-time video stitching has seen implementations using CUDA with Nvidia [5] and experiments with OpenGL as well [4][10]. Most studies report results with a limited number of cameras, and extending this to VR content creation remains an ongoing research area. Creating real-time 360-degree panoramas for VR applications from cameras with limited field of view (FOV) poses challenges, especially in scenarios like deep-sea exploration and autonomous vehicles, requiring more cameras than typically used for VR content.

The paper compares the computational power of CPU, NVIDIA CUDA, and OpenGL for efficient parallel processing. We implement video stitching to generate a wide panorama using three methods: a) Entirely on CPU, b) Offloading part of the stitching pipeline to NVIDIA graphics card using CUDA and c) Modified pipeline to take advantage of the inherent graphics with OpenGL. The resulting panorama is viewed in an immersive VR head-mount device with stereo vision incorporated. We compare the computational speeds from the three methods and arrive at the most optimal method for achieving real-time results for generating VR content in a scenario with cameras having limited FOV.

II. Related Work

Our work extensively uses some classic computer vision algorithms and extends their applications in the different CPU and GPU based video stitching methodologies. Our stitching pipeline begins with feature detection and matching across the video frames.

Feature description has been central to numerous computer vision algorithms such as object recognition, image matching etc[13]. Karami et al.[2] compare the performance of SIFT, SURF, and ORB image matching techniques under diverse image transformations, including scaling, rotation, noise, fisheye distortion, and shearing. In most scenarios SIFT outperforms the other techniques and had been extensively used in many applications

SIFT feature descriptors have been widely used in numerous applications including panoramic video generation. Gopalakrishnan et al.[7] [8] use feature generation and matching to blend images from different perspectives in innovative ways to create an immersive remote classroom experience. Anju et al.[9] propose a fusion approach for human action recognition, leveraging bag of visual words (BoVW) and neural networks. In their work they use SIFT to extract the BoVW. Vasudev et al. [12] use SIFT and SURF to extract features as a major step for a multi-factor authentication system with ID card credentials.

In real-time video stitching applications with fixed camera positions, feature detection and matching act as calibration steps followed by iterative steps of the perspective transform and panorama composition. Wang et al.[1] accelerate high-resolution dual camera image stitching using GPUs, dividing the process into CPU-optimized time-critical and GPU-accelerated time-flexible steps. They show efficient GPU utilization for real-time computation on an NVIDIA RTX 3080 Ti. Rangan et al.[6] address real-time panoramic video stitching challenges, presenting a dynamic algorithm for standard definition videos. They explore GPU parallelization and a pipe-lined approach. Baranwal et al.[3] present the Pixellayer approach for real-time image and video stitching with live wireless cameras, addressing blind spots and enhancing cinematic effects. Most of this research work utilizes the computation capabilities of NVIDIA CUDA and achieves higher speeds compared to CPU computation. They however get limited with the number and type of cameras. To achieve a real-time 360-degree panorama from input video feeds with limited FOV cameras and extend it for VR applications remains a challenge.

Further studies have experimented with the inherent graphic's parallel computing capabilities using OpenGL and OpenCL. Silva et al.[4] propose a real-time 360º video stitching and streaming methodology, with a specific emphasis on GPU processing. The solution, designed for cloud architectures, employs Genlock-synchronized cameras and GPU features, with parallel copying, OpenGL-based deformable meshes, and pixel shader operations for blending and color correction. They employ a camera rig of 4 to 6 cameras. Wei-Sheng et al.[5] developed a GPU-based spherical panorama video stitching system using OpenCL for real-time processing with four webcams. Utilizing OpenCL and OpenGL pixel buffer objects, the system achieves a 76x speedup in converting multiple images into panoramic views.

Harsha et al.[10] do a comprehensive survey of advancements in Fisheye Lens Image Stitching for Enhanced Advanced Driver Assistance Systems in Automated Vehicles. Here is use of OpenGL for blending and compositing the stitched image is noted. Jungjin Lee[11] in the paper introduces the Wand system for geometric registration of projectors using a 360 camera. Wand utilizes a single 360-degree camera to register multiple projectors into a spherical image through a 2D-grid-mesh optimization for 360-degree video projection mapping.

Generating real-time 360-degree VR panoramas from narrow FOV cameras remains challenging, notably in settings like deep-sea exploration and autonomous vehicles. This paper explores OpenGL's potential and compares various GPU-based methods to find an efficient, scalable approach for generating real-time VR content.

III. Optimized Video Stitching and GPU Rendering for VR Environments

TABLE I: Notation Specifications

Notation	Specification
f_{ref}	Reference Frame
v_{ref}	Reference Video
f_1	Frame from Video 1
f_3	Frame from Video 3
kp	Keypoints
d	Descriptors
m	Matches
fkp	Filtered Keypoints
H_1	Homography Matrix for f_{ref} to f_1
H_2	Homography Matrix for f_{ref} to f_3
wf	Warped Frame
sf	Stitched Frames
p	Panorama

The proposed optimized video stitching methodology targets the efficient processing of three videos. f_{ref} serves as the reference frame, while f_1 and f_3 are warped with respect to it.in Algorithm 1,The video stitching algorithm, consists of two phases-

The calibration phase: which consists of sequence of operations, starting from SIFT feature detection and culminating in homography matrix which is derived from the parameters by SIFT.

Warping phase: Warps the incoming input non-reference frames (f_1 and f_3) to align with the reference frame (f_{ref}) based on the computed homography matrices (H_1 and H_2). In order to get an Optimal performance, frame rate evaluations are done to the warping process non reference frames (f_1 and f_3) done in the loop of the Algorithm 1.

A. Video Stitching with CPU

The Homography matrix is derived from the calibrated parameters in the video stitching algorithm performed in CPU

and serves as the cornerstone for geometric transformations. In this context, the OpenCV library is commonly employed, leveraging its *warpPerspective()* function. This function facilitates the warping of incoming video frames based on the calculated Homography matrix, aligning the frames to the reference frame and achieving a cohesive visual output.

B. Video Stitching with GPU

Optimizing the warping process involves calibrating two Homography matrices (H1 and H2) independently for non-reference frames, leveraging the GPU's computational power. Using NVIDIA CUDA and OpenGL, the GPU efficiently handles complex mathematical operations, enhancing overall warping performance. This strategic task division accelerates image transformations, proving advantageous for real-time video processing and computer vision applications.

*1) **Video Stitching with NVIDIA CUDA**:* In the NVIDIA-based video stitching implementation, the process starts by transferring video frames to the GPU for parallel processing. Leveraging the GPU's computational power accelerates the warping phase, aligning frames using the Homography matrix. Subsequently, the warped frames are retrieved on the CPU for further processing as shown in Fig. 1. Here, combining the frames involves adding pixel values from each warped frame to create a unified frame. The resulting stitched video demonstrates the efficiency boost from GPU acceleration, particularly evident in computationally intense tasks like warping.

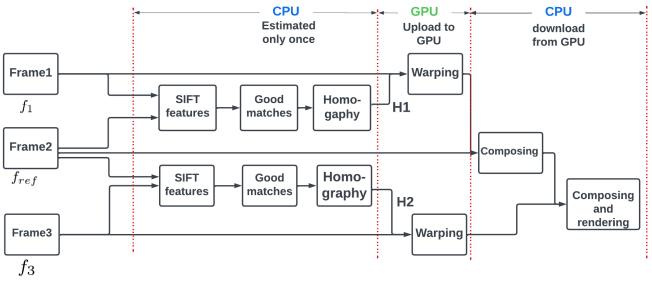


Fig. 1: Stitching Pipeline with CUDA

*2) **Video Stitching with OpenGL**:* Utilizing GPU's parallel computational capabilities, video stitching via OpenGL seamlessly blends and merges video frames into a panoramic view. Initially, video frames load as textures for rendering. Vertex and fragment shaders play a pivotal role: the vertex shader, guided by homography matrices from OpenCV, precisely adjusts texture coordinates to align frames within the panorama geometry. Simultaneously, the fragment shader orchestrates pixel-level operations, efficiently handling transformations and blending. Operating in parallel, these shaders leverage the GPU's capacity to process multiple pixels concurrently, significantly boosting performance. This synchronized interplay updates textures with incoming frames in real time, applies homography transformations, and renders adjusted textures onto the screen as illustrated in Fig. 2. OpenGL's architecture optimizes frame blending, transformation, and rendering tasks, ensuring a seamlessly stitched video output.

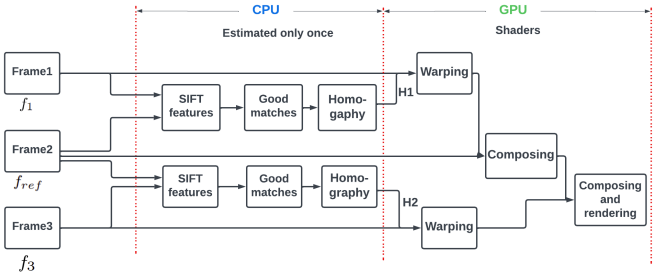


Fig. 2: Stitching Pipeline with OpenGL

C. Stereo Vision

Stereo vision in VR presents distinct images to each eye, creating a 3D effect. Using a fixed image shift, it simulates natural variations in human vision, enhancing immersion. When applied to a smartphone VR player in a Google Cardboard-like headset, it provides a convincing perception of depth and spatial relationships.

D. Video Stitching Algorithm

Video stitching combines multiple video frames into a seamless panorama using computer vision algorithms like KNN, SIFT, and Lowe's Ratio Test. These algorithms match and align features across frames, creating a smooth panoramic view, following the steps shown in Algorithm 1.

1) Scale-Invariant Feature Transform (SIFT): SIFT[15], depicted in Fig. 3, detects distinctive key points and computes invariant descriptors, accommodating scale, rotation, and illumination variations.

Fig. 3: SIFT Keypoints

2) K-Nearest Neighbors (KNN): KNN is utilized for feature matching as shown in Fig.4. Given the feature descriptors extracted by SIFT from different frames, KNN searches for the nearest neighbours of each feature in the descriptor space. This process establishes correspondences between key points in different frames, providing the necessary information for aligning frames during stitching.

3) Lowe's Ratio Test: Lowe's Ratio Test is employed to filter and refine the feature matches obtained through KNN. It assesses the quality of matches by comparing the distance ratios between the best and second-best matches. This ratio test helps to discard ambiguous matches and retain more reliable correspondences, enhancing the accuracy of the

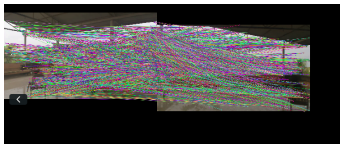


Fig. 4: Matching

subsequent holography estimation.

4) Random Sample Consensus (RANSAC): Random Sample Consensus (RANSAC)[16] is a robust iterative method widely used in computer vision, including video stitching. It estimates model parameters by iteratively selecting random data subsets, commonly applied to find the homography matrix in video stitching. RANSAC's robustness in handling outliers, such as noise or occlusions, enhances accuracy and reliability in tasks like homography estimation.

5) Warping: Warping the estimated homography matrix involves transforming the non-reference video frame to align with the reference frame as shown in Fig.5. This process adjusts the pixel coordinates based on the homography matrix, ensuring geometric consistency between frames.

Fig. 5: Warping

6) Compositing and Rendering: Aligning all the frames and stitching them together forms a panoramic video as shown in Fig.6 This process ensures a smooth transition between frames, creating an immersive viewing experience.

Fig. 6: Stitched Video

7) Stereo Vision: Utilizing a fixed shift, depth perception is simulated by presenting distinct images to each eye as shown in Fig.7. This view is generated and played back in the VR headset

Fig. 7: Stereo Vision

Algorithm 1 Video Stitching Algorithm

$kp_1, d_1 \leftarrow$ `sift.detectAndCompute`(first_frame_video1)
$kp_{ref}, d_{ref} \leftarrow$ `sift.detectAndCompute`(first_frame_video2)
$kp_3, d_3 \leftarrow$ `sift.detectAndCompute`(first_frame_video3)
$m_1 \leftarrow$ `knn_match`(d_1, d_{ref})
$m_2 \leftarrow$ `knn_match`(d_3, d_{ref})
$fkp_1 \leftarrow$ `lowe_ratio_test`(m_1)
$fkp_2 \leftarrow$ `lowe_ratio_test`(m_2)
$H_1 \leftarrow$ `calc_homog_RANSAC`(fkp_1)
$H_2 \leftarrow$ `calc_homog_RANSAC`(fkp_2)
for $f1$, $f2$ in video1, video3: **do**
    $wf_1 \leftarrow$ `warp_frame`($f1$, H_1)
    $wf_2 \leftarrow$ `warp_frame`($f2$, H_2)
    sf.append(wf_1, wf_2)
end for
$p \leftarrow$ `compose_panorama`(sf)
`render_panorama`(p) =0

IV. Experimental Setup

Our experiment utilized three 1280 × 720, 30 FPS cameras with an AMD Ryzen 7 3750h, NVIDIA GTX 1650 and Intel Iris Xe. OpenCV played a pivotal role in our workflow, extensively utilized for calibration and homography generation. The warping phase, a critical aspect of our methodology, was executed using three distinct methods: standard warping in OpenCV with CPU, GPU-accelerated warping with NVIDIA CUDA (executed on a GTX 1060 GPU), and GPU-accelerated warping with OpenGL (executed on an Intel Iris Xe GPU).

Our methodology utilized a Logitech C922 Pro camera with 30 FPS and a 78-degree field of view, balancing frame rate and perspective. Cameras were systematically placed with controlled rotation for a 27-degree overlap, ensuring comprehensive coverage. This approach successfully yielded high-quality results in panoramic stitching.

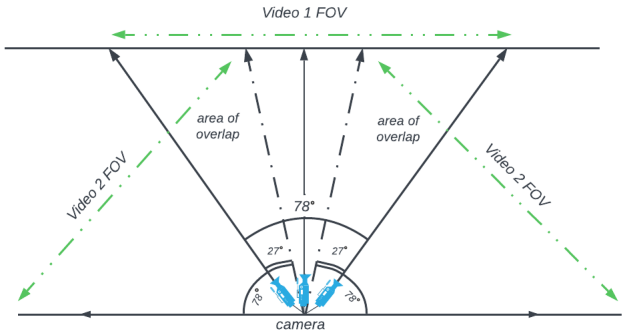


Fig. 8: camera-placement

Assessing panoramic video stitching performance across CPU, CUDA, and OpenGL involves evaluating execution time and frames per second (FPS). Differing warping methods done by these methods make this assessment crucial.

Execution Time: The execution time refers to the time taken by the algorithm to perform the warping process for a single output frame. It is computed by capturing the time at the start (T_a) and end (T_b) of the warping process, excluding the homography computation, denoted as:

$$Execution\ Time\ per\ Frame = T_b - T_a$$

Here, T_b denotes the time in milliseconds before the start of the warping operation, while T_a represents the time in milliseconds at the end of the warp operation.

Simple Moving Average (SMA): The Simple Moving Average is a commonly used technique to smooth out fluctuations in time series data. In the context of video stitching, SMA is employed to calculate the average frame stitching time over a specified window size. The formula for SMA is given by:

$$SMA_t = \frac{\sum_{i=1}^{N} FrameTime_i}{N}$$

Here, SMA_t represents the Simple Moving Average at time t, $FrameTime_i$ is the duration of the i-th frame, and N is the window size.

Weighted Moving Average (WMA): Unlike SMA, the Weighted Moving Average assigns different weights to each frame within the window, giving more importance to recent frames. This can be particularly useful in scenarios where recent data holds more relevance. The WMA formula is expressed as:

$$WMA_t = \frac{\sum_{i=1}^{N} (\frac{i}{N} \cdot FrameTime_i)}{\sum_{i=1}^{N} \frac{i}{N}}$$

In this equation, WMA_t represents the Weighted Moving Average at time t, and i is the position of the frame in the window. The weight increases linearly with the frame's position.

Both SMA and WMA are integrated into the project to evaluate and monitor the performance of the video stitching process. By calculating these averages over a window of 1000 frames, the project gains insights into the average frame time dynamics.

Frames Per Second (FPS): FPS quantifies the rate of frame processing by the algorithm, excluding homography computation time. It is calculated as the reciprocal of the execution time per frame, denoted by either SMA or WMA:

$$FPS = \frac{1}{Execution\ Time\ per\ Frame}$$

A higher FPS value signifies that the algorithm can process and stitch frames faster, resulting in smoother and more real-time video output.

These performance metrics serve as essential benchmarks for evaluating the real-time processing capabilities of the video stitching algorithm.

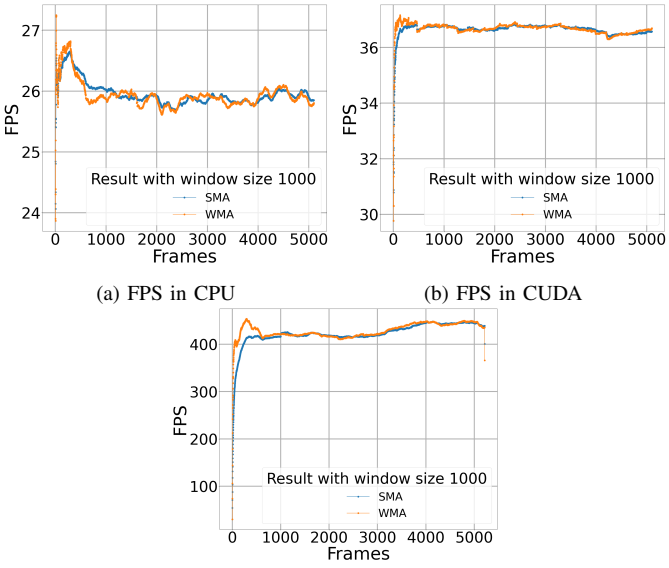
Speedup: Speedup is a crucial performance metric that quantifies the improvement in execution time achieved by parallelizing or optimizing a computational task. It is defined as the ratio of execution time between a sequential and a parallel algorithm. Mathematically, the speedup (S) can be expressed using the following equation:

$$S = \frac{T_{\text{sequential}}}{T_{\text{parallel}}}$$

where $T_{\text{sequential}}$ is the execution time on a sequential (CPU) platform, and T_{parallel} is the execution time on a parallel or optimized (GPU) platform. The greater the speedup value, the more efficient the parallelized or optimized implementation is compared to the sequential one.

V. Results and Discussion

OpenGL attains a frame rate of 440 (SMA) and 450 (WMA), yielding 17x and 18x acceleration compared to CPU processing as seen in Ttable II. This exceptional performance, driven by OpenGL's strengths in graphics rendering and parallel processing on modern GPUs, is crucial for accommodating future cameras.



(a) FPS in CPU (b) FPS in CUDA

(c) FPS in OpenGL

Fig. 9: FPS Trends in CPU, CUDA, and OpenGL

TABLE II: Results with SMA and WMA

Method	CPU	CUDA	OpenGL
Exec.time (ms) - SMA	38.6	27.3	2.27
FPS - SMA	25.9	36.6	440
Speed up - SMA		1.4	17.3
Exec.time (ms) - WMA	39.4	27.2	2.22
FPS - WMA	25.8	36.7	450
Speed up - WMA		1.4	17.7

Table II shows CUDA was 1.4 times faster than the CPU-based approach, and OpenGL was even faster, with a speedup of over 17x, making it significantly better for real-time video stitching. In comparison, WMA offers a faster response to changes due to its emphasis on recent results. In our results,in fig.9 we observed nearly identical graphs for both SMA and WMA, but with occasional peaks and depressions in the WMA graph. These fluctuations underscore the responsiveness of WMA to dynamic changes, highlighting its ability to adapt

swiftly to variations in the data. The emphasis on real-time performance and low latency in our application aligned well with OpenGL capabilities, enabling direct access to the GPU texture buffer. One significant limitation we observed with CUDA was the delay caused by the back-and-forth movement of image data between the CPU and GPU. This inherent latency, resulting from the necessity to transfer video frames between the two processing units, impacted the real-time performance of our video stitching application. On the contrary, the direct access to GPU memory and the broader hardware and software compatibility provided by OpenGL, contributed to its overall superiority in our research.

Despite capturing a wide field of view, stretching issues forced us to crop parts of the output. To eliminate this, we'll explore creating 180-degree panoramas with optimized camera placement in future work.

VI. Conclusions

OpenGL achieves a remarkable 17x and 18x speedup over the CPU for WMA and SMA. This positions OpenGL as a strong contender for VR applications. Currently on platforms like Google Cardboard, there's potential for expansion to advanced headsets like Oculus, promising enriched VR experiences.

References

[1] Wang, Jingsi, Dong Liu, Ran Zheng, Yongcai Hu, Wu Gao, and Chen Zhao. "Accelerating High-resolution Image Stitching for the Dual Camera System based on GPU." In 2022 China Automation Congress (CAC), pp. 3009-3013. IEEE, 2022.

[2] Karami, Ebrahim, Siva Prasad, and Mohamed Shehata. "Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images." arXiv preprint arXiv:1710.02726 (2017).

[3] Baranwal, Amritanshu, Ajeet Rohilla, Asha Rani Mishra, and Sansar Singh Chauhan. "Pixellayer-A novel approach for stitching digital Images and Videos." In 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), pp. 112-117. IEEE, 2023.

[4] Silva, Rodrigo MA, Bruno Feijó, Pablo B. Gomes, Thiago Frensh, and Daniel Monteiro. "Real-time 360 video stitching and streaming." In ACM SIGGRAPH 2016 Posters, pp. 1-2. 2016.

[5] Liao, Wei-Sheng, Tung-Ju Hsieh, and Yang-Lang Chang. "Gpu parallel computing of spherical panorama video stitching." In 2012 IEEE 18th International Conference on Parallel and Distributed Systems, pp. 890-895. IEEE, 2012.

[6] Rangan, P. Venkat, Uma G. Balaji Hariharan, N. Ramkumar, and Rahul Krishnan. "Real-time Spatial Video Panorama using Iterative Compositing."

[7] Hariharan, Balaji, S. Padmini, and Uma Gopalakrishnan. "Gesture recognition using Kinect in a virtual classroom environment." In 2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), pp. 118-124. IEEE, 2014.

[8] Gopalakrishnan, Uma, N. Ramkumar, P. Venkat Rangan, and Balaji Hariharan. "Multilayered Presentation Architecture in Intelligent eLearning Systems." In Intelligent Systems Technologies and Applications 2016, pp. 532-540. Springer International Publishing, 2016.

[9] Nair, S. Anju Latha, and Rajesh Kannan Megalingam. "Fusion of Bag of Visual Words with Neural Network for Human Action Recognition." In 2022 12th International Conference on Cloud Computing, Data Science Engineering (Confluence), pp. 14-19. IEEE, 2022.

[10] Vardhan, Harsha, and Vinay Hegde. "Advancements in Fisheye Lens Image Stitching for Enhanced ADAS in Automated Vehicles." In 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), pp. 1-6. IEEE, 2023.

[11] Lee, Jungjin. "Wand: 360 video projection mapping using a 360 camera." Virtual Reality (2023): 1-13.

[12] Vasudev, R., N. Harini, and M. R. Neethu. "Multi-Factor Authentication System With ID Card Credentials For Secure Transactions." In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-8. IEEE, 2023.

[13] Kavin Kumar, D., Latha Parameswaran, and Senthil Kumar Thangavel. "A computer vision-based approach for object recognition in smart buildings." New Trends in Computational Vision and Bio-inspired Computing: Selected works presented at the ICCVBIC 2018, Coimbatore, India (2020): 13-22.

[14] Cabral, Brian K. "Introducing Facebook Surround 360: an open, high quality 3D-360 video capture system." Retrieved March 25 (2016): 2017.

[15] Lowe, D.G. (2004) 'Distinctive image features from scale-invariant keypoints', International Journal of Computer Vision, 60(2), pp. 91–110.

[16] Fischler, Martin A., and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." Readings in Computer Vision, 1987, 726–40.