Virtual Reality and Augmented Reality Applications

by

TUAN P. M. HO (TUAN HO)

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

To my family, without whom this thesis would never have been finished.
My wife – Nhung Nguyen, my mother – Trang Phan, and my father – Loan Ho.

To my mentor, Dr. Madhukar Budagavi, whose valuable advice has guided me through the maze of uncertainty in Ph.D. research to eventually arrive at the finish line.

ABSTRACT

Virtual Reality and Augmented Reality Applications

TUAN P. M. HO (Tuan Ho), Ph.D.

The University of Texas at Arlington,

Supervising Professor: K. R. Rao

Committee: Madhukar Budagavi, Ioannis D. Schizas, Jonathan Bredow, William E. Dillon

Virtual Reality (VR) and augmented reality (AR) are the emerging fields of research. VR enables the immersive, being-there feeling of the viewers in a recorded or virtual environment, while AR enhances the real-world perception with the aid of an electronic device. On the VR part, this thesis introduces the novel 360-degree video stitching algorithms for dual-fisheye lens cameras, which are compact and affordable. On the AR section, this work presents a new approach in compressing the 3-dimensional point cloud models that are used in free view-point sports and tele-immersive applications.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

Fisheye-len Stitching Algorithm

1.1 Background

360-degree videos and images have become very popular with the advent of easy-to-use 360-degree viewers such as Cardboard [4] and GearVR [5]. This has led to renewed interest in convenient cameras for 360-degree capturing. A 360-degree image captures all the viewing directions simultaneously and gives users the sense of immersion when viewed. Early 360-degree imaging systems used a catadioptric camera [6], which combines lens (dioptric) and mirror (catoptric), to record 360-degree contents. Although the lens plus mirror geometry is sophisticated and usually requires proper calibration, such as one in [7] [8], to generate good visual results, a catadioptric system can produce panoramas without seams. However, due to the inherent lens+mirror arrangement, the captured field of view is typically limited to less than 360x180 degrees, and some of the catadioptric systems are not compact.

An alternate method for 360-degree recording is using a polydioptric system which incorporates multiple wide-angle cameras with overlapping field of views. The images from the multiple cameras are stitched together to generate 360-degree pictures. However, due to camera parallax, stitching artifacts are typically observed at the stitching boundaries. Example 360-degree polydioptric cameras include Ozo [9], Odyssey [10], and Surround360 [11] by some of the major companies. The number of

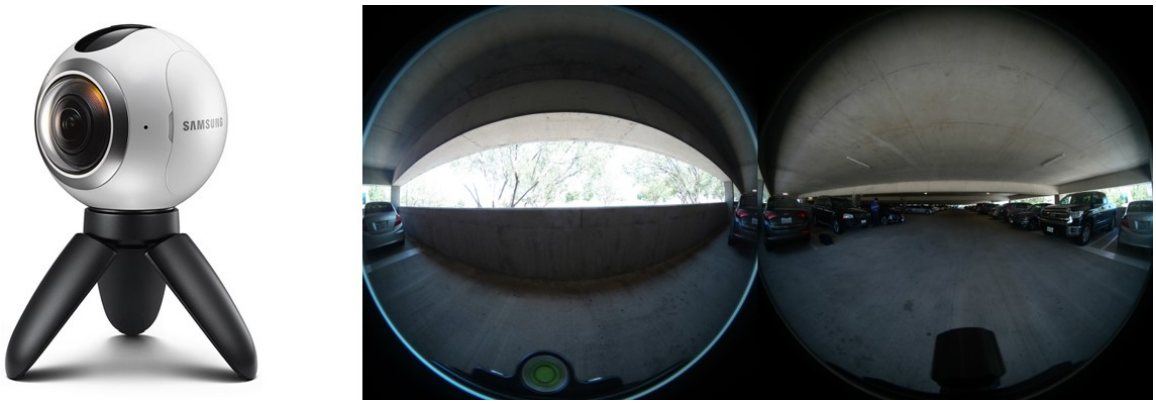


Figure 1.1: (a) Samsung Gear 360 Camera (left). (b) Gear 360 dual-fisheye output image (7776 x 3888 pixels) (right). Left half: image taken by the front lens. Right half: image taken by the rear lens.

cameras used in these systems ranges from 8 to 17. These cameras typically deliver professional quality, high-resolution 360-degree videos.

On the downside, these high-end 360-degree cameras are bulky and extremely expensive, even with the decreasing cost of image sensors, and are out of reach for most of the regular users. To bring the immersive photography experience to the masses, Samsung has presented Gear 360 camera, shown in Figure 1.1(a). To make the camera compact, Gear 360 uses only two fisheye lenses whose field of view is close to 195 degrees each. The images generated by the two fisheye lenses (Figure 1.1(b)) have very limited overlapping field of views but can, however, be stitched together to produce a full spherical 360x180 panorama.

1.2 Previous Works

For stitching of images from the multiple cameras, a feature-based stitching algorithm [12][13] is typically used to extract the features of the images being stitched. These features are then matched together. An iterative method is carried out to eliminate the incorrect matches (outliers). The reliability of this process not only

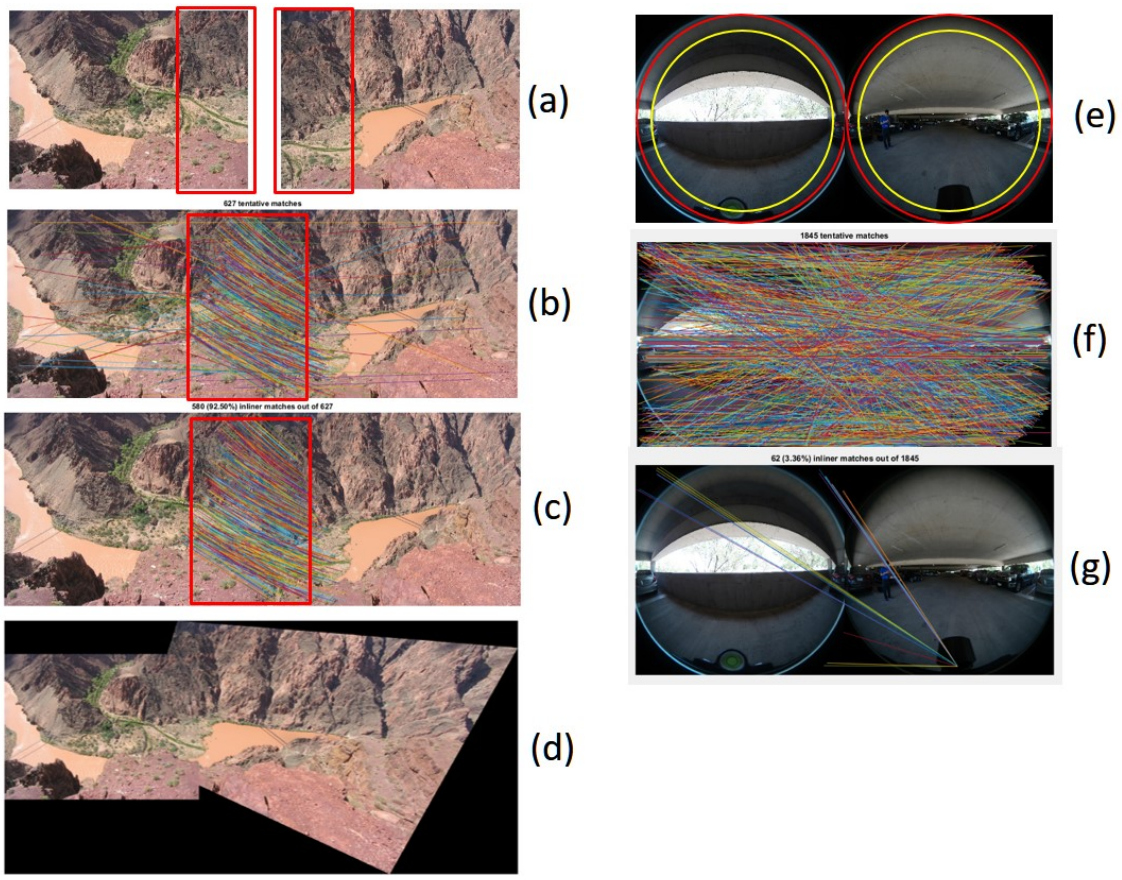


Figure 1.2: Image stitching illustration. Left column: (a) Regular pictures with good overlaps. (b)(c) Features Matching using SIFT and outlier removal using RANSAC. (d) Image warping and panorama creation. Right column: (e) Fisheye images taken by Samsung Gear 360. (f)(g) Features Matching (using SIFT) and outlier removal (using RANSAC). Courtesy: VLFeat [1] toolbox.

depends on the iterative method being used but also on the size of the overlapping regions. With sufficient overlap, more reliable matches (inliers) are retained while outliers get removed. Using these inliers, a homography matrix is computed to warp and register the pictures together (assuming the camera matrix is already estimated) before stitching them.

However, this conventional stitching method does not work well on Gear 360-produced pictures since there is very limited overlap between the two fisheye images.

Figure 1.2 shows the stitching processes for the photos taken by the regular rectilinear lens and the ones taken by Samsung Gear 360. The pictures on the left column, from [1], in Figure 1.2 have a good overlap and can be aligned and stitched well. In contrast, Gear 360 has limited overlap leading to a small number of inlier matches only on the outer ring of the fisheye images. This results in a homography matrix that is invalid for the interior of the fisheye images. Hence, a conventional stitching process cannot be directly used for stitching fisheye images from two-lens systems such as Gear 360.

1.3 The Proposed Algorithm

We have introduced a novel stitching method [2][14][15] that adaptively minimizes the discontinuities in the overlapping regions of Gear 360 images to align and stitch them together. The proposed algorithm has four steps. The first step describes how to measure and compensate for the intensity fall off of the camera's fisheye lenses. The second phase explains the geometry transformation to unwarp the fisheye images to a spherical 2-Dimensional (equirectangular projection [16]) image. The next stage introduces our proposed two-step alignment to register the fisheye unwarped images. Finally, the aligned images are blended to create a full spherical 360x180-degree panorama. Figure 1.3 shows the block diagram of our fisheye stitching framework.

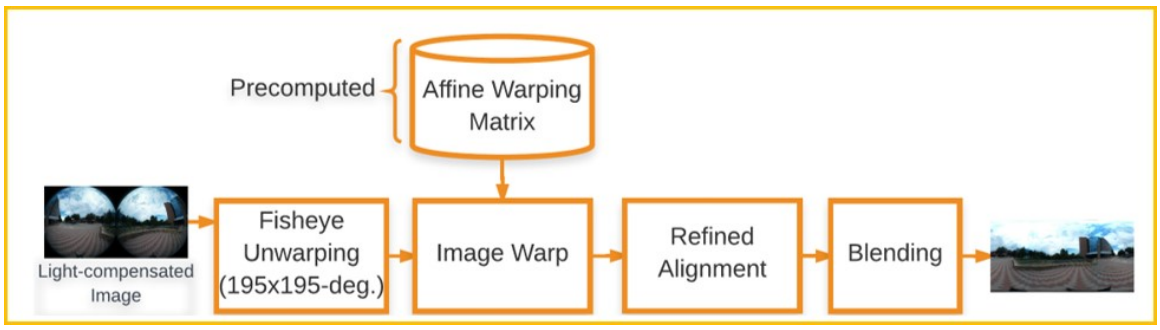


Figure 1.3: Block Diagram of Our Fisheye Images Stitching [2].

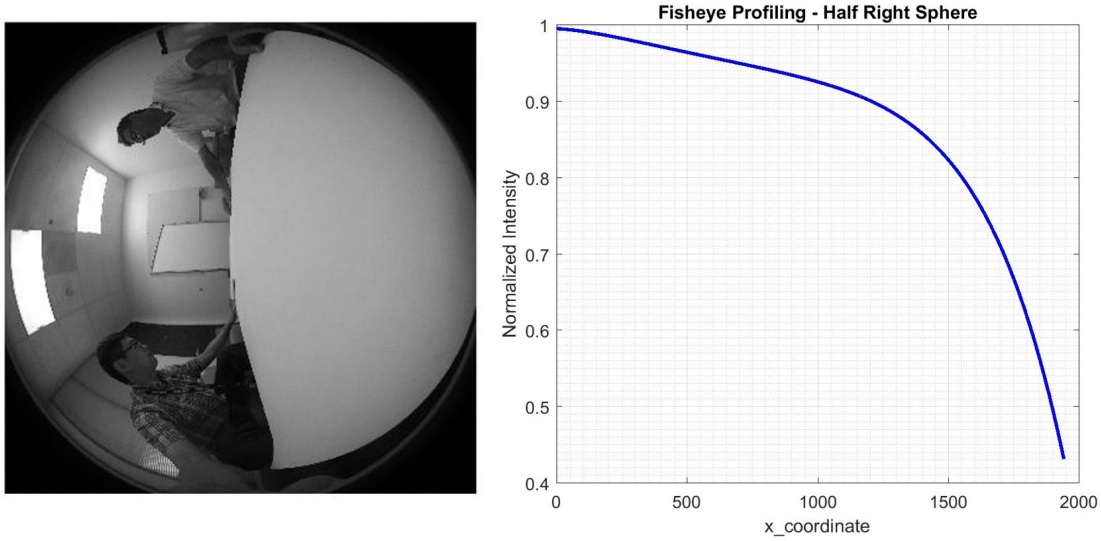


Figure 1.4: (a) Fisheye profiling experiment (left). (b) Intensity fall-off curve (right). Coordinate in x-direction is in pixel unit.

1.3.1 Fisheye Lens Intensity Compensation

Before stitching the two fisheye images, one needs to compensate for the light fall-off of the two fisheye images.

Vignetting is an optical phenomenon in which the intensity of the image reduces at the periphery compared to the center. To compensate for this light fall-off, we captured an image of a large white blank paper using Gear 360 and measured the variation of pixel intensity along the radius of the fisheye image toward its periphery in Figure 1.4. The intensity is normalized to one at the center of the picture. We used a polynomial function $p(x)$ to fit the light fall-off data.

$$p(x) = p_1 x^n + p_2 x^{n-1} + ... + p_n x + p_{n+1} \tag{1.1}$$

whereas x is the radius from the center of the image. Figure 1.5 shows the result of this process.

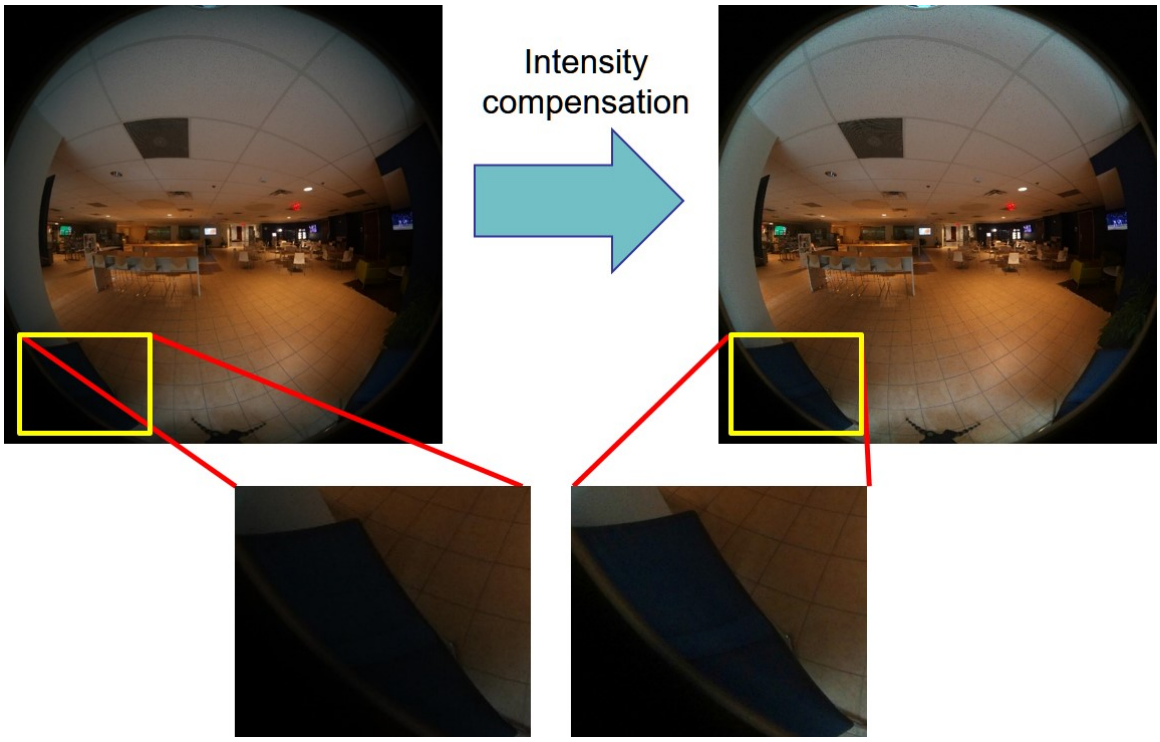


Figure 1.5: Compensation result. Without intensity compensation (left); with intensity compensation (right).

1.3.2 Fisheye Unwarping

Fisheye lenses can produce ultra-wide field of views by bending the incident lights. As a result, the image looks severely distorted, particularly in the periphery. Therefore, a fisheye unwarping–a geometric transformation is necessary to generate a natural appearance for the Gear 360 fisheye-produced picture. Instead of rectifying the fisheye-distorted image, we use a method that unwarps the image and returns a 2-D spherical projected picture for 360-degree purposes.

This method involves two steps, shown in Figure 1.6. First, each point $\mathbf{P}'(x', y')$ in the input fisheye image is projected to a 3-D point $\mathbf{P}(\cos\theta_s \sin\theta_s, \cos\varphi_s \cos\theta_s, \sin\theta_s)$ in the unit sphere. φ_s and θ_s can be derived by considering the coordinates of the fisheye image directly as pitch and yaw. Therefore, $\theta_s = (f * x)/W - 0.5$, and

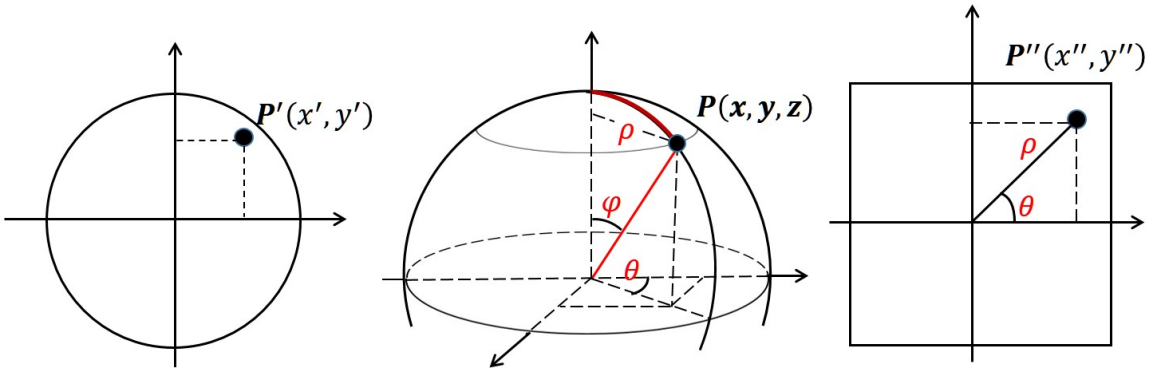


Figure 1.6: Fisheye Unwarping.

$\varphi_s = (f * y)/H - 0.5$, where f is the lens' field of view (in degree), W and H are image width and height respectively. The second step derives the distance between the projected center and the 3-D point $P(x, y, z)$: $\rho = H/f * \tan^{-1}\sqrt{(x^2 + z^2)/y}$, whereas $x = \cos\varphi_s \sin\theta_s$, $y = \cos\varphi_s \cos\theta_s$, $z = \sin\varphi_s$. Then the 2-D spherical (equirectangular) projected point $\mathbf{P}''(x'', y'')$ is constructed as $x'' = 0.5W + \rho\cos\theta$, $y'' = 0.5H + \rho\sin\theta$, and $\theta = \tan^{-1} z/x$. In this equirectangular projection, x'' and y'' are pitch and yaw respectively. The unwarped image can be viewed on a 360-degree player. Figure 1.7 shows the result of the unwarping process. Figure 1.7 (a) illustrates the natural look of the unwarped image when viewed on a 360-degree viewer compared to the distorted appearance of the original fisheye image.

Unwarping is a necessary process in fisheye image stitching. However, result in Figure 1.7 (b) shows that the unwarped images, when put together in a 360x180 plane, are not aligned with each other. Therefore, further steps have to be taken to register the pictures together.

1.3.3 Two-step Image Alignment

After unwarping, the two images are not aligned with each other. The between-lenses misalignment patterns are similar for different Gear 360 cameras of the same

model. To minimize this misalignment we have used a control-point-based approach followed by a refined alignment as follows.

1.3.3.1 Control Point-based Alignment

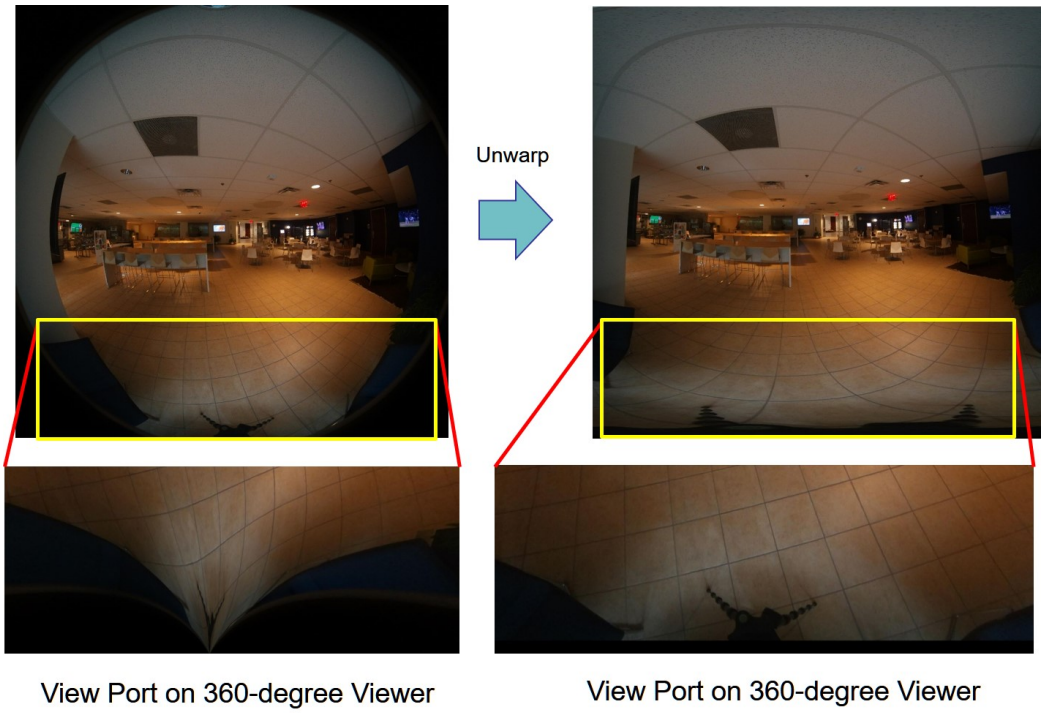
In the setup in Figure 1.8, we position the Gear 360 so that both the lenses see the checkerboards on their sides. Therefore, they have the same view of the overlapping regions. Also, the distance between the camera and the checkerboards is around 2m, which is about the maximum reach that the checkerboard corners are still clearly visible for control point selection. The images taken by the Gear 360 left and right lenses are unwarped using the method introduced in section 2.2, and arranged in 360x180-degree planes shown in Figure 1.9. About 200 pairs of control points are then manually selected from the overlapping regions between the unwarped pictures, and are used to estimate a 6-parameter affine matrix A, which warps a point $B(x_2, y_2)$ to $T(x_1, y_1)$ as follows:

$$[x_1, y_1, 1] = [x_2, y_2, 1]\ A, \quad \text{whereas } A = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix} \tag{1.2}$$

1.3.3.2 Refined Alignment

The first registration helps align the images, as shown in Figure 1.10. However, when the objects in the boundaries move closer or further away from the camera, the horizontal discontinuities become visible as shown in Figure 1.11(c).

To minimize the discontinuity in the overlapping regions, we choose to maximize the similarity in these areas. To this end, a novel adaptive alignment that involves a



(a)

(b)

Figure 1.7: Unwarping results: (a) Fisheye-unwarped image whose view port displayed on a 360-degree viewer; (b) Two unwapred images arranged in a 360x180 image.

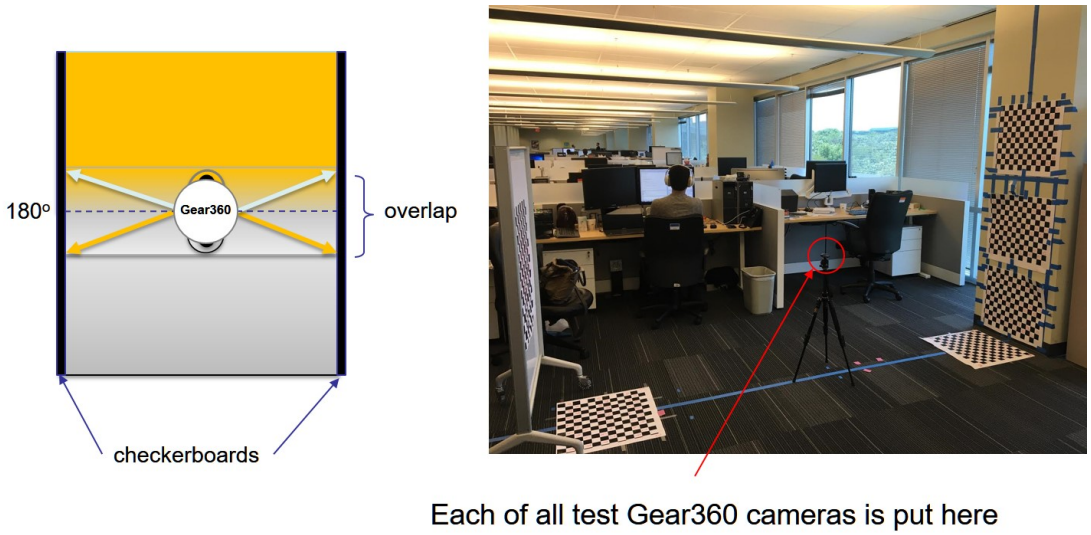


Figure 1.8: Experimental Setup for Gear 360 Dual-lens Mis-alignment.

fast template matching for objects in the overlapping region and utilizes the matching displacement to derive a refined affine matrix to align the images further is proposed.

The matching is a normalized cross-correlation operation. The cross-correlation of two signals maximizes at a point when the two signals match each other. In addition, since there are always some level of exposure differences in the overlapping regions, the template and reference images to be matched should be normalized. This proposal employs a fast normalized cross-correlation algorithm [17]:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}\,]}{\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}\,]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}\,]^2\}^{0.5}} \tag{1.3}$$

where γ is the normalized cross-correlation, f is the reference image, $\bar{t}$ is mean of the template image, $\bar{f}_(u,v)$ is the mean of $f(x,y)$ in the region under the template. The template and reference are taken from the top and bottom unwarped images respectively, as shown in Figure 1.11(a).

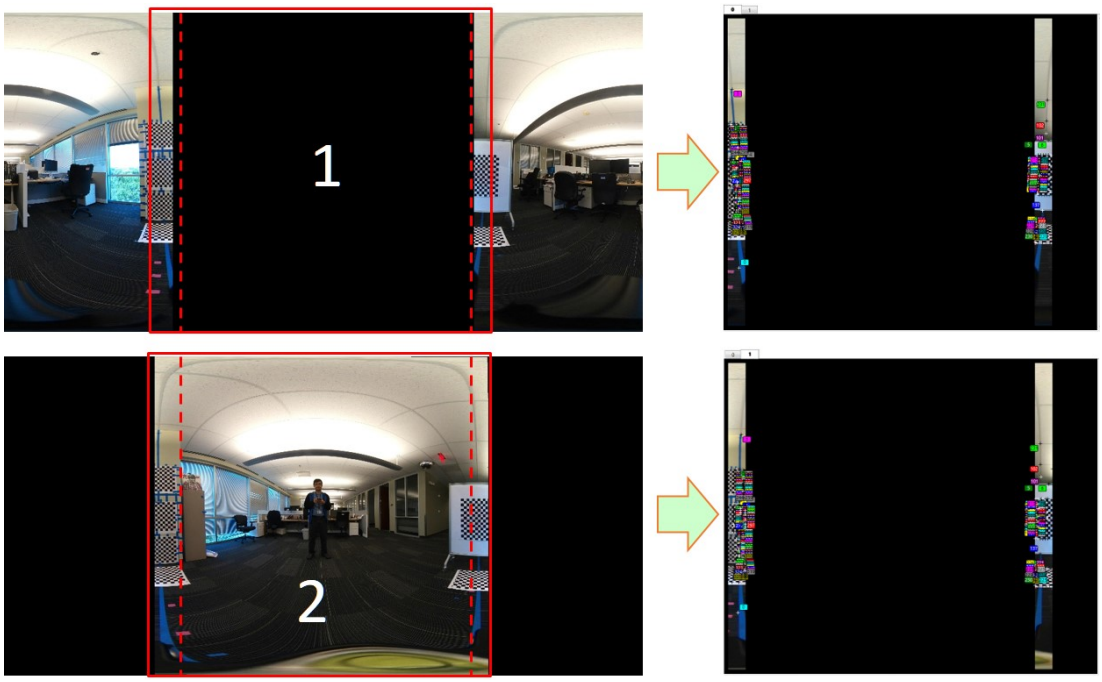


Figure 1.9: Control point selection on the overlapping area. The fisheye images are unwarped using the method presented in section 2.2.

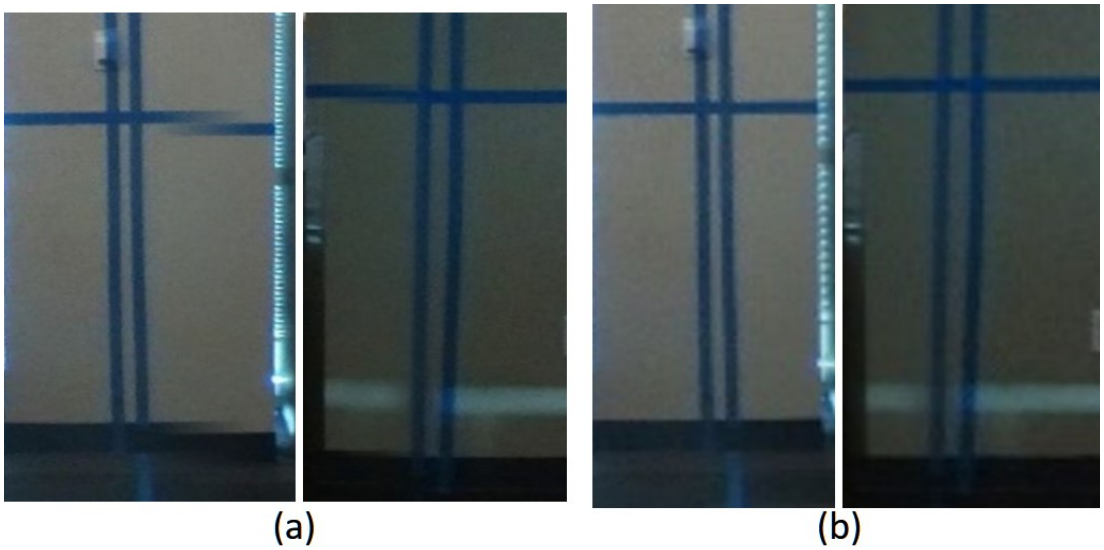


Figure 1.10: Result of the control point-based alignment. (a) Without the first alignment. (b) With the first alignment.

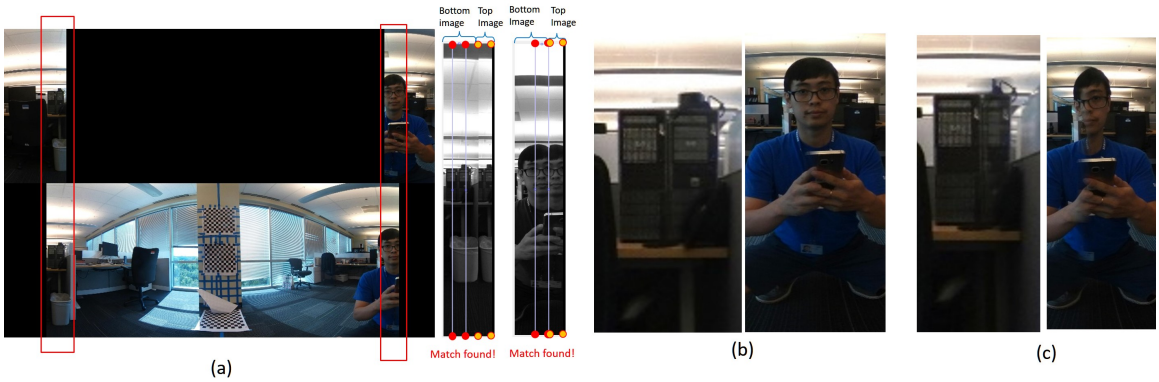


Figure 1.11: (a) A person close to the camera and between the lens boundaries. (b) The blended overlaps with the proposed refined alignment (discontinuity minimized). (c) The blended overlaps without the proposed refined alignment (very visible discontinuity). The first alignment is already applied for both (b) and (c) to align the images vertically.

The maximum value of the normalized cross-correlation returns the displacement of where the best match occurs. This shift indicates how much the template – a rectangular window has to move to match the reference. The proposed method then estimates an affine matrix from vertices of the matching windows (four in each overlapping region) and warp the bottom image to align it further with the top one.

Figure 1.11 shows that the refined alignment helps align the images by maximizing the similarity in the overlapping region. The person, close and in the lens boundary, appears as a complete one (i.e. no visible duplicate or missing any body parts) in the stitched 360x180-degree picture.

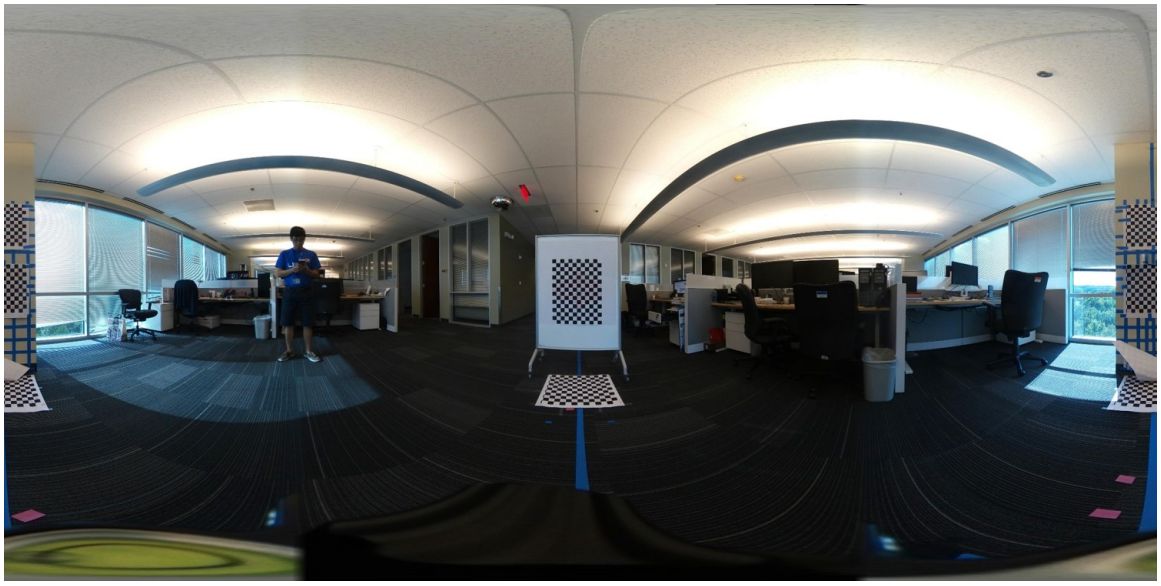
1.4 Implementation and Results

We have implemented the proposed approach in C++ with OpenCV library and Matlab. The affine matrix in the first alignment is pre-computed off-line and included as part of the fisheye unwarping process. The refined alignment, however, is computed on-line, adaptively to the scene. The polynomial coefficients in section

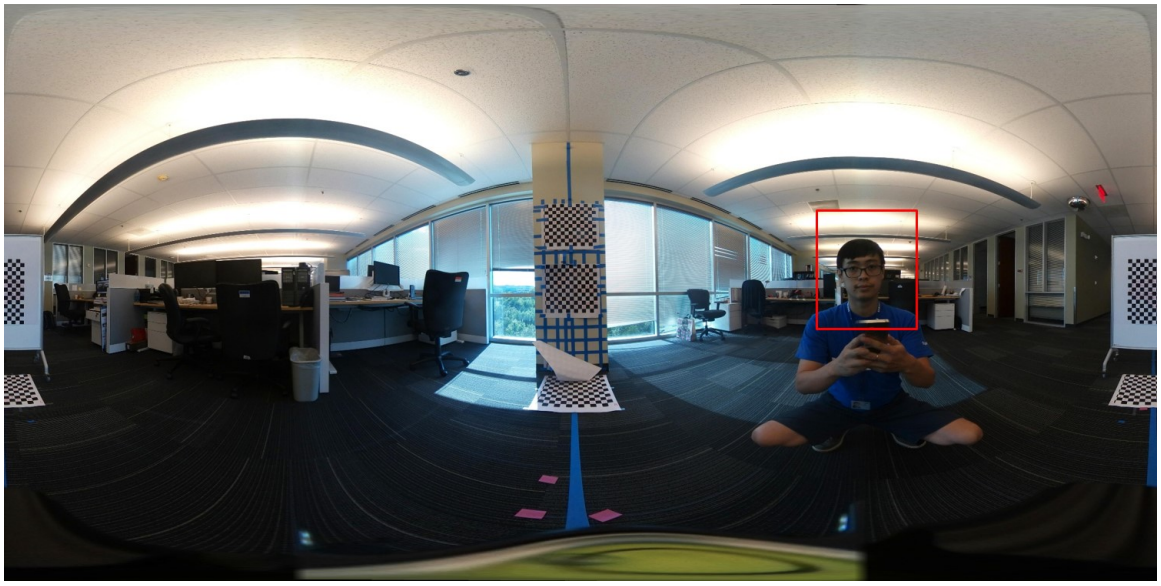
2.1 are: $p_1 = -7.5625 \times 10^{-17}$, $p_2 = 1.9589 \times 10^{-13}$, $p_3 = -1.8547 \times 10^{-10}$, $p_4 = 6.1997 \times 10^{-8}$, $p_5 = -6.9432 \times 10^{-5}$, $p_6 = 0.9976$. We found that the field of view of 193 degrees, which is very close to the documented 195-degree, gives the best results as shown in Figures 1.12 and 1.13. Our approach can also accurately stitch images taken by different Gear 360 cameras of same model thanks to the proposed refined alignment that operates adaptively and can compensate for the geometric mismatch between Gear 360 lenses.

1.5 Conclusions

This proposal introduces a new stitching method for 360-degree cameras with dual-fisheye lens. It uses a novel alignment algorithm that adaptively maximizes the similarities in the boundary regions of the images from the two fisheye lenses for accurate registration and stitching. In summary, the proposed approach compensates for fisheye lens' intensity fall-off, unwarps the fisheye images, then registers them together using the proposed adaptive alignment, and applies blending on the registered images to create a 360x180-degree panorama that is viewable on 360-degree players. Results show that not only this method can stitch Gear 360 images that have limited overlap, but it can also produce well-stitched pictures even if there are objects that are at an arbitrary distance to the camera and stand in the lenses boundaries.



(a)



(b)

Figure 1.12: Samsung Gear360 360x180-degree panoramas stitched by the proposed method: (a) A person not close to the lenses overlapping boundaries; (b) A person close to the lenses overlapping boundaries.

(a)

(b)

Figure 1.13: Samsung Gear360 360x180-degree panoramas stitched by the proposed method: (a) Garage; (b) Building with patterned ground.

CHAPTER 2

360-degree Video Stitching Based On Rigid Moving Least Squares

In the approach presented in Chapter 1, the first step compensates for the geometric misalignment between the two fisheye lenses and depends on the camera parameters. The second step is a more refined alignment that adjusts any discontinuities caused by objects with varying depth in the stitching boundaries. In the first alignment, [2] solves an over-determined system for a warping matrix which is then used to align the images. This results in a least-squares approximated solution which globally transforms the images. Our observation is that typically the control points in the central part of the 360-degree image get aligned well resulting in improved quality compared to prior techniques. However, the control points at the top and bottom part of the image do not get aligned precisely leading to stitching artifacts in those regions. Figure 2.1 shows an example of visible discontinuities in the stitching boundary of pictures with patterns on the background.

This chapter discusses our method in [18] that improves the image alignment and stitching performance over the entire stitched 360x180-degree panoramas. It uses rigid moving least squares approach to achieve the improved alignment. We also extends the work to video stitching by incorporating a new temporal-coherent algorithm to produce jitter-free 360-degree videos.

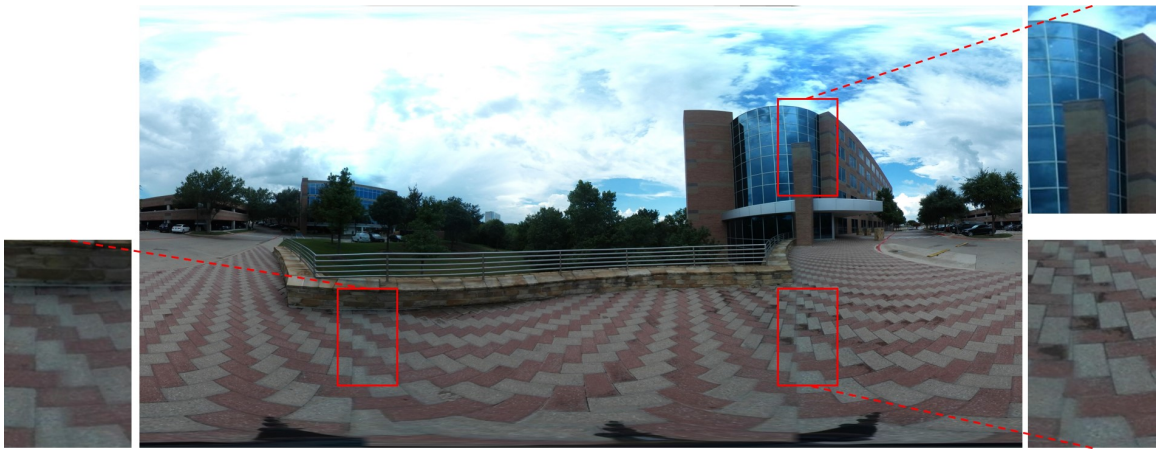


Figure 2.1: 360x180-degree panorama stitched by [2] and the discontinuities in the overlapping regions.

2.1 The proposed algorithm

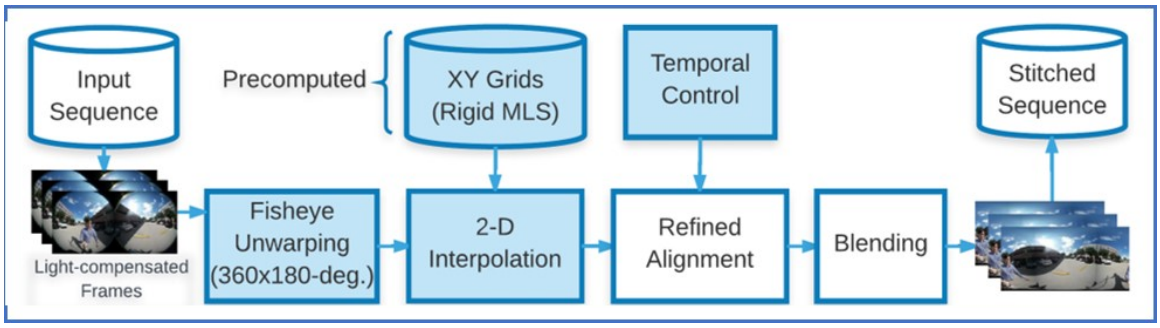


Figure 2.2: Block Diagram of the 360-degree Video Stitching.

Figure 2.2 shows the block diagram of the proposed algorithm in this paper. Similar to [2], the proposed image alignment also has two steps – the first one is dependent on camera parameters, and the second step works adaptively to the scene. However, instead of estimating a warping matrix in a least-squares sense to align the pictures in the first step, we generate interpolation grids to deform the image based on rigid moving least squares (MLS) approach.

2.1.1 Rigid Moving Least Squares

Let p and q be the control points in the overlapping regions of the original and deformed images respectively. [19] defines three properties of an image deformation function f which are: interpolation ($f(p_i) = q_i$ under deformation), smoothness (preserves smooth transition among pixels), and identity ($q_i = p_i \Rightarrow f(v) = v$).

For every point in the image, we solve for a transformation matrix M that minimizes the weighted least squares:

$$\underset{M}{\operatorname{argmin}} \sum_i w_i \left\| \hat{p}_i M - \hat{q}_i \right\|^2 \tag{2.1}$$

where the weights w_i are proportional to the distance between the image point v and the control point p_i in the sense that w_i gets smaller when v moves further away from p_i (i.e. the least squares minimization depends on the point of evaluation, thus the name moving least squares). When $v \to p_i$, f interpolates $f(v) = q_i$. [19] defines such weights as:

$$w_i = \frac{1}{|p_i - v|^{2\alpha}}$$

$\hat{p}_i = v - p^*$ and $\hat{q}_i = q^*$ are derived from each point v in the image and the weighted centroids p^* and q^* [19].

For control points selection, we adopted the checkerboard experiment from [2] with our method of picking the correspondence points. In this experiment, both fisheye lenses, each has 195-degree field of view, see the same checkerboards on their sides. The images taken by the fisheye lenses are unwarped to 360x180-degree equirectangular planes. We then arrange the unwarped images so that the right image is positioned at the center of the 360x180-degree plane, while the left image is split and put to the sides of the plane. With this arrangement, the overlapping regions are

ready for control-point selection. By choosing the same checkerboards' cross sections on the unwarped images, one can visualize the geometric misalignment between the two lenses. Figure 2.3 shows the selected control points $\{p\}_i$ and $\{q\}_i$, which indicate the differentiated positions of the same points in the stitching boundaries of the two images. Our interest is to determine the function f_r that does the transformation $f_r(p_i) = q_i$ in the overlapping regions while keeping the other areas of the image as visually intact as possible.

While the MLS is general in the matrix M in (2.1), we are only interested in the rigid transformation since it generates more realistic results than affine and similarity transformations. The similarity transformations are a subset of the affine transformations that have only translation, rotation, and uniform scaling. The similarity matrix M is defined such that $M^T M = \lambda^2 I$ (e.g. a rotation matrix). λ^2 acts as a uniform scaling factor. In rigid transformation, it is desirable that no uniform scaling is included. [19] proposed a theorem that relates the MLS solution for $M^T M = \lambda^2 I$ (similarity transformation) to its solution of $M^T M = I$ (rigid transformation), and derived the solution for the rigid MLS function f_r. We invite readers to read [19] for more details about the mathematical treatment used here.

We generate the rigid-MLS interpolation grids to deform the right unwarped image (i.e. to apply f_r over the image), thus aligning it with the left one. Figure 2.4 shows the right unwarped fisheye image gets deformed by the rigid MLS method. While the portions of the image in proximity to the stitching boundaries are transformed to match the other image, the remaining of the deformed picture have no discernible difference compared to the original.

2.1.2 Refined Alignment

The rigid MLS aligns the control points around the stitching boundary, thus registering the two unwarped fisheye images together. However, when the depth of the object in the overlapping areas changes, it introduces misalignment to the scene. Therefore, a refined alignment is necessary after the rigid MLS deformation. To this end, we adopt the same adaptive method of using the normalized cross-correlation matching in [2] to further align the images.

The refined alignment performs a fast template matching and utilizes the matching displacements on both stitching boundaries to generate eight pairs of control points. These points are then used to solve for a 3x3 affine matrix to warp the deformed image. As a least-squares solution, this refined method is not sufficient in registering images with complicated misalignment patterns, but it works very well for those with minor misalignment such as the one caused by the varied object's depth. Figure 2.5 shows that the refined alignment minimizes the discontinuity when the person is sitting close to the camera's stitching boundary.

2.2 Extension to 360-degree video

In the 360-degree video stitching, it is essential to minimize jitters–the abrupt transition between the stitched frames so that the final video appears continuous and comfortable to view. Adjacent frames in the sequence that are not stitched by the same measure can generate jitters. In the work presented here, when a bad match occurs without getting filtered out in the refined alignment, it generates a false warping matrix that abruptly distorts the stitching boundary of the picture. This attenuated scene causes jitter which is the result of the sudden transition between the previous well-stitched frame and the current bad-stitched one. Therefore, it is

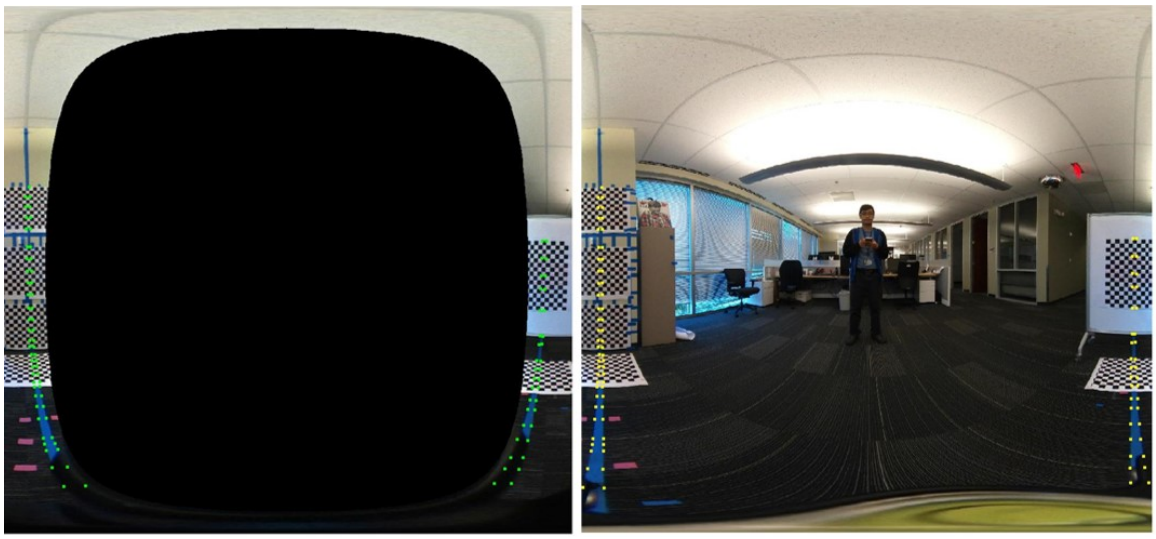


Figure 2.3: Left: the overlapping areas on the unwarped left image and $\{q\}_i$ (green dots). Right: unwarped right image and $\{p\}_i$ (yellow dots).

important to guarantee good matches throughout the entire sequence to maintain smooth frame-to-frame transition, and thus minimize jitters.

Algorithm 1: Refined Alignment (with jitter control)

```
   Input: leftImage, rightImage (deformed)
 1 (scoreLeft, scoreRight) ← TemplMatch();
 2 if ( both matching scores are good ) then
 3 |   Estimate affine warping matrix affineMat;
 4 |   Store affineMat for the next frame;
 5 |   warpEn ← 1;
 6 else // bad scores on either boundary
 7 |   if matching scores of the previous frame are good then
 8 |   |   warpEn ← 1;
 9 |   |   affineMat ← previous affineMat;
10 |   else
11 |   |   warpEn ← 0; // don't warp image
12 |   end
13 end
14 if ( warpEn ) then
15 |   Warp rightImage by affineMat;
16 end
```

Figure 2.4: The original and the rigid-MLS-deformed images with their control points $\{p\}_i$ and $\{q\}_i$ overlayed.

Algorithm 1 illustrates our method to maintain the temporal coherence for the sequence. A good score is returned at one stitching boundary if all of the followings satisfied. First, the peak normalized cross-correlation is larger than 0.85/1.0. Second, the returned vertical displacement is in the margin of $[-10, +10]$ pixels. Third, the horizontal displacement of the current match must not exceed 10% margin compared to its of the previous frame. These constraints, obtained from our empirical experiments, are set to eliminate bad matching caused by poor lighting and abrupt movements of the boundaries in horizontal and vertical directions.

2.3 Implementation and Results

We have implemented the proposal algorithm in C++ with OpenCV library. The rigid MLS grids are precomputed. It takes around 500ms to stitch one 3840x1920 360-degree frame on a laptop with Intel core i7 1.8 GHz, 8GB RAM, 256GB SSD (excluding the one-time configuration setup, such as reading MLS deform 2-D inter-

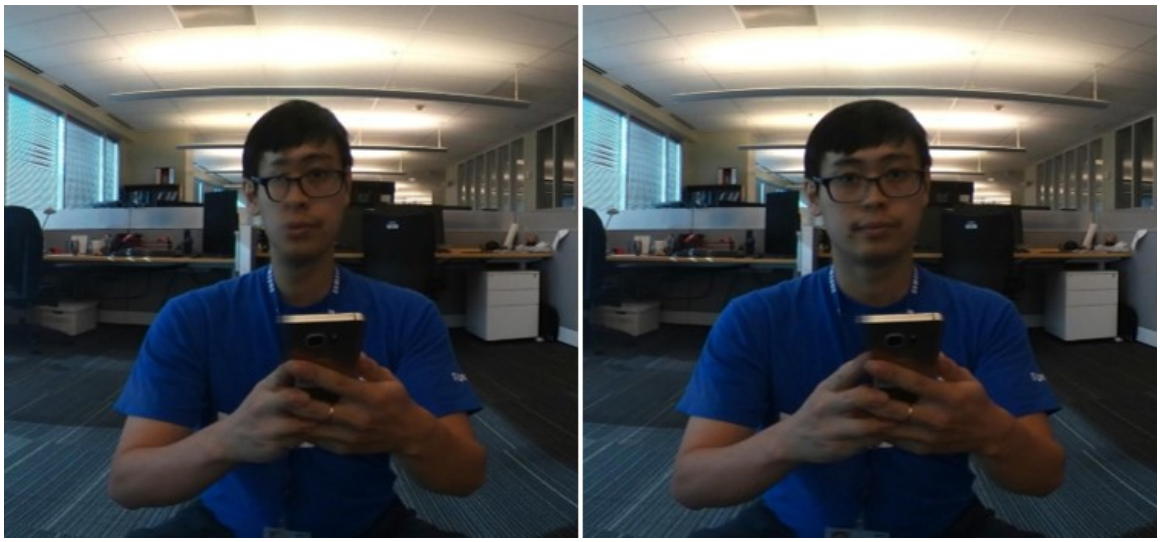


Figure 2.5: The person sitting close to the camera and in the stitching boundary. Left: after rigid MLS deformation. Right: after rigid MLS deformation and refined alignment.

polation grids, etc., in prior to the stitching process). In addition, the deformation is an interpolation process that can be accelerated by GPU.

Figure 2.6(a) illustrates an image stitched by the proposed method. In this picture, there are a fence, buildings, and patterned background on the stitching boundaries. Figure 2.6(b) shows the comparison of the stitching boundaries in the image stitched by the proposal and by [2] (also in Figure 2.1). While the discontinuities appear in the stitching boundaries in [2] as the result of the least-squares solution, the proposed method produces seamless 360-degree panorama thanks to the rigid MLS deformation.

For video stitching[1], Figure 2.7 demonstrates the adjacent stitched frames created by the proposal (top row, no jitter) and by [2] (bottom row). In the bottom row, the first jitter occurs when the refined alignment lets a bad match get through, resulting in an affine transformation that moves the image on the right side of the

Figure 2.6: (a) 360x180-degree panorama stitched by this proposal. (b)(c) The stitching boundaries in (a) (top row) compared to the same image stitched by [2] (bottom row).

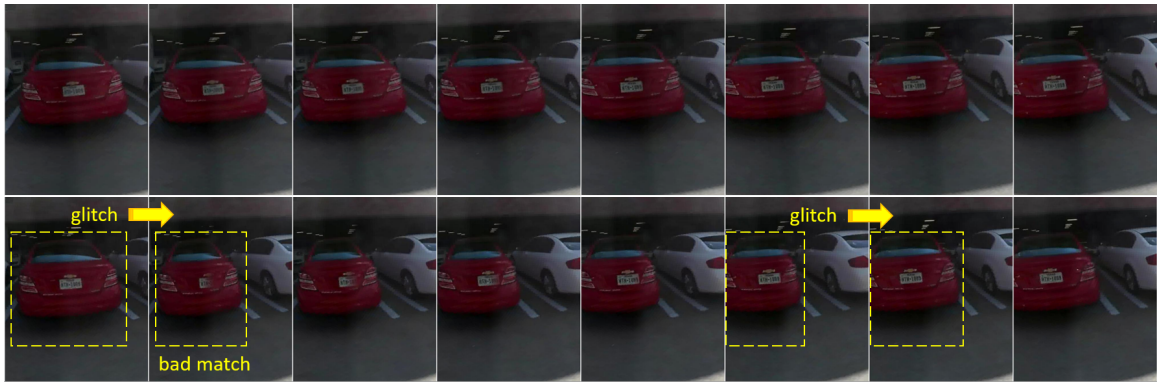


Figure 2.7: Stitching boundary in consecutive frames. Stitched by the proposal (top row), and by [2] (bottom row).

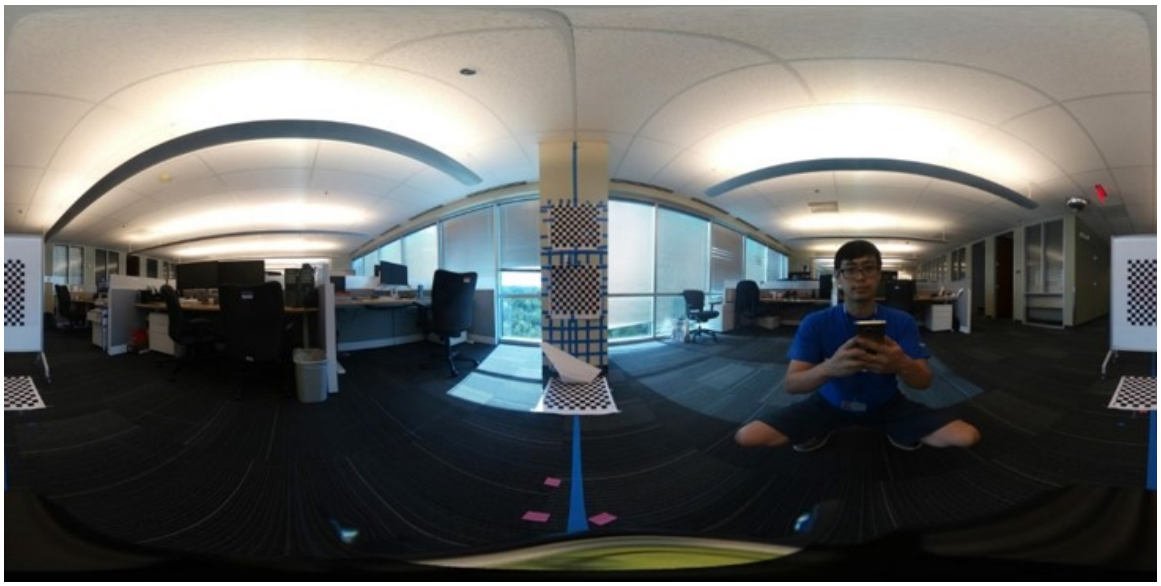
stitching boundary to the left. As a result, the car in the boundary gets distorted leading to an abrupt transition between the frames.

2.4 Conclusion

This chapter has introduced a novel method for stitching the images and video sequences generated by the dual-fisheye lens cameras. The proposed alignment has two steps. The first one, carried out offline, compensates for the sophisticated geometric misalignment between the two fisheye lenses on the camera based on rigid

Figure 2.8: Images captured by the dual-fisheye lens camera (Gear 360).

moving least squares approach. The second step, applied online and adaptively to the scene, provides a more refined adjustment for any misalignment created by the objects with varying depth on the stitching boundaries. We extend the proposed approach to 360-degree video stitching with the relevant constraints to maintain the smooth transition between frames and therefore minimize jitters. Results show that our method not only generates more accurately stitched 360x180-degree images but also jitter-free 360-degree videos.



(a)

(b)

Figure 2.9: 360x180-degree panoramas stitched by the proposed method with input images as shown in Figure 2.8.

(a)

(b)

Figure 2.10: 360x180-degree panoramas stitched by the proposed method with input images as shown in Figure 2.8.

CHAPTER 3

Point Cloud Compression

While 360-degree videos enable the being-there / immersive feeling of viewers, the novel 6-degree-of-freedom (6 DoF) or free-viewpoint video (Figure 3.1) not only brings the experience of immersion but also the possibility to adjust the viewpoint at the viewer's discretion.

To this end, a 360-degree background and the 3-D models, e.g. the basketball players, are captured. Each point in the 3-D model typically has 6 attributes, three of which define the point geometric position in the 3-D space. The other three specify the color of the point. A million point-model is not uncommon in practice. Such 3-D model, which is comprised of 3-D points, is called a point cloud. Figure 3.3 illustrates a point cloud representing a person.

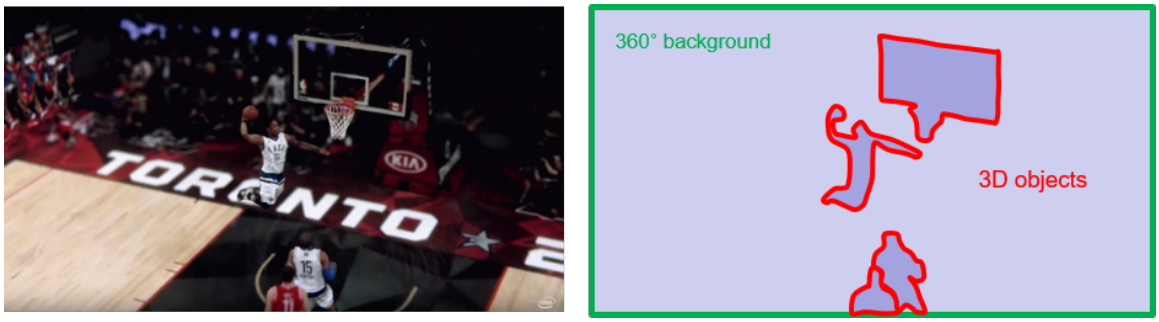


Figure 3.1: Free-viewpoint video. Courtesy: MPEG.

Figure 3.2: A 3-D point cloud model. Courtesy: MPEG.

The 3-D point cloud in its raw format is stored at 54 bits per input point for integer geometry, i.e. 3×10-bit for geometry, plus 3×8-bit for color. Floating geometry requires 3×32-bit for accurate geometric representation. Million-point models demand huge storage. Hence, a novel 3-D point cloud compression is developed to address this emerging issue. This chapter introduces our work for MPEG's call for proposal (cfp) [20] on point cloud compression [21][22][23][24].

3.1 Previous Works

A widely-used approach to compress the 3-D point clouds is based on the octree data structure [25][26]. [3], which was developed on top of the Point Cloud Library [27] and adopted as MPEG anchor for point cloud compression, composes the input point cloud model into an octree structure [28] and applies compression for its geometry

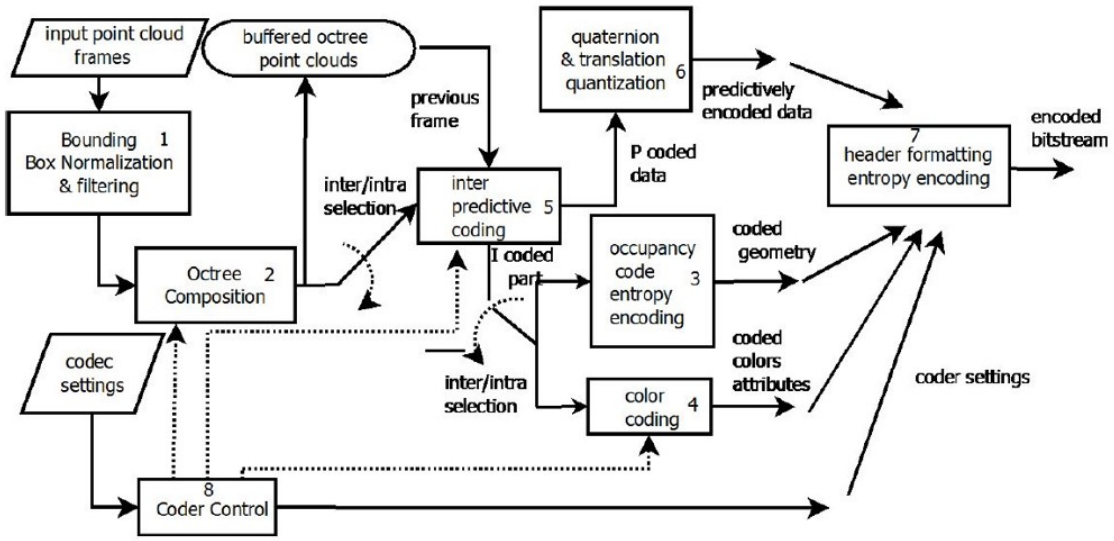


Figure 3.3: MPEG Point Cloud Compression Anchor.

and color components separately. In this method, the geometry atrributes are coded based on the voxel position in the octree, while the color attributes are coded using a JPEG coder. Figure 3.3 shows the block diagram of the point cloud compression introduced in [3].

While the octree-based approaches are tailored to the geometry attributes, it does not take full advantage of the state-of-the-art video coding standards to maximize the compression efficiency.

3.2 Our Approach

Instead of composing the 3-D point cloud into the octree data structure, we propose to map the former into 2-D video which are, then, compressed by the High Efficiency Video Coding Standard (HEVC) [29]. Figure 3.4 shows the block diagram of the proposed method.

For encoding, a 3-D point cloud model is mapped into 2-D frames by our sorting method based on the geometric distribution of the points in the model. A sub-

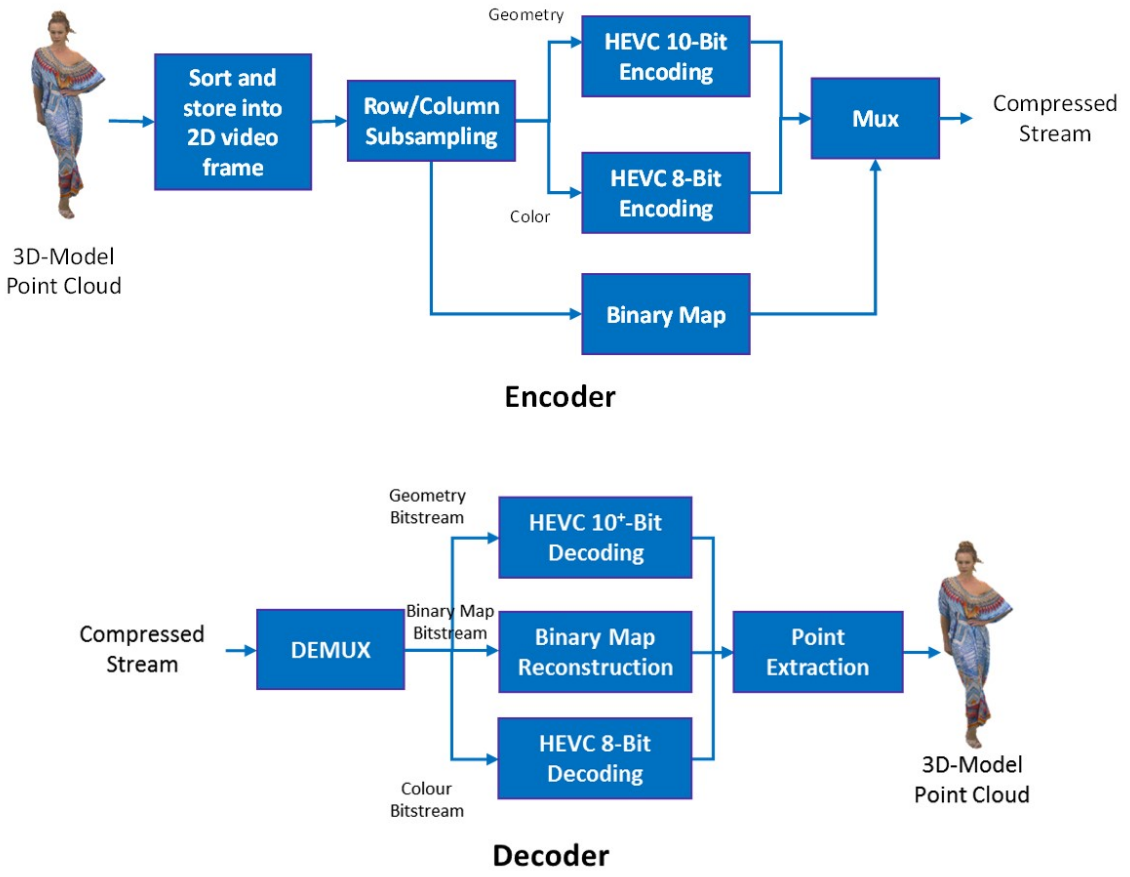


Figure 3.4: Block Diagram of the proposed point cloud compression.

sampling process is then applied to reduce the number of samples in the 2-D frames before encoding. In addition, to reconstruct the 3-D model, a binary map which indicates the position of the write data is compressed and stored into the bitstream.

The benefits of this approach are two-fold. First, it allows the fast industry adoption to meet quickly emerging applications such as AR due to:

- Use of existing high-performance hardware implementations (if codecs such as HEVC Main10 are used).
- Use of existing ISOBMFF (ISO base media file format) based delivery methods.

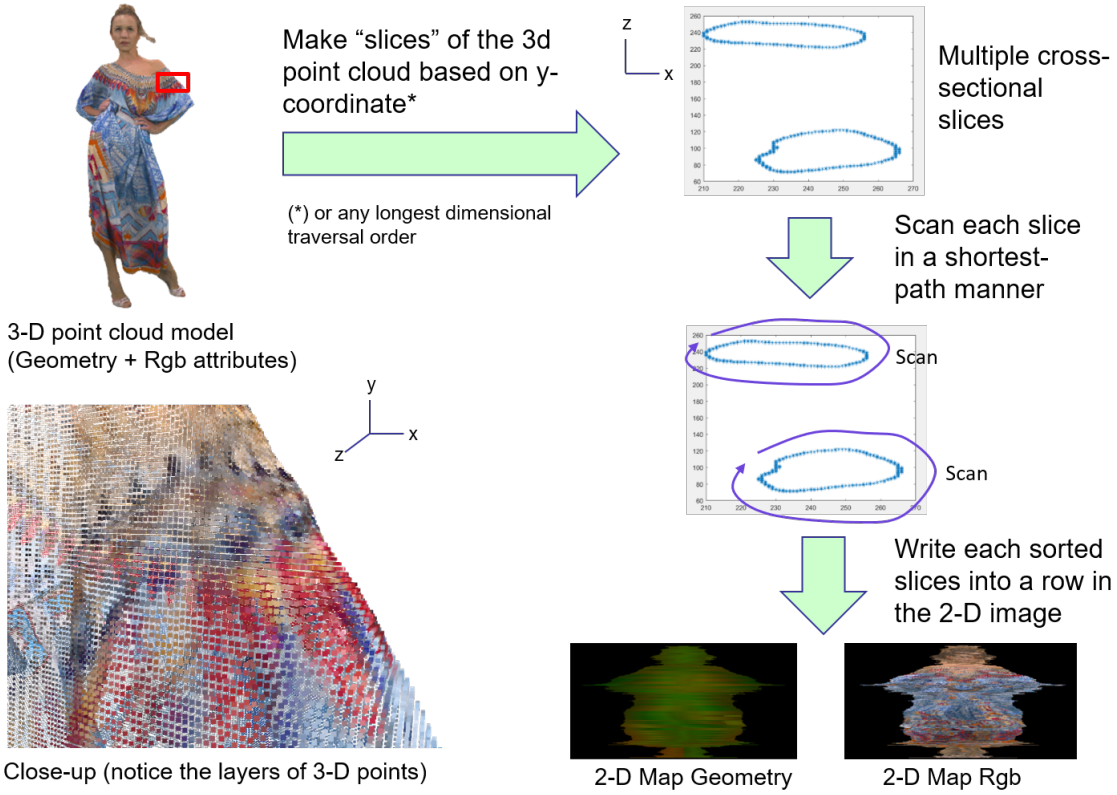


Figure 3.5: The proposed 3-D to 2-D mapping.

Second, the performance will be automatically improved with the on-going future video codec development.

3.3 The Proposed Mapping

Figure 3.5 depicts our proposed method to map a 3-D point cloud model into a 2-D frame. This approach is comprised of two main steps. First, the 3-D model is scanned by its longest coordinate axis to produce the 2-D cross-section slices. For example, the model illustrated in Figure 3.5 is a human-shaped one, so scanning the model by the y-axis, the longest dimension in this case, would preserve the shape better. Second, each slices produced from the first step are scanned in a shortest path

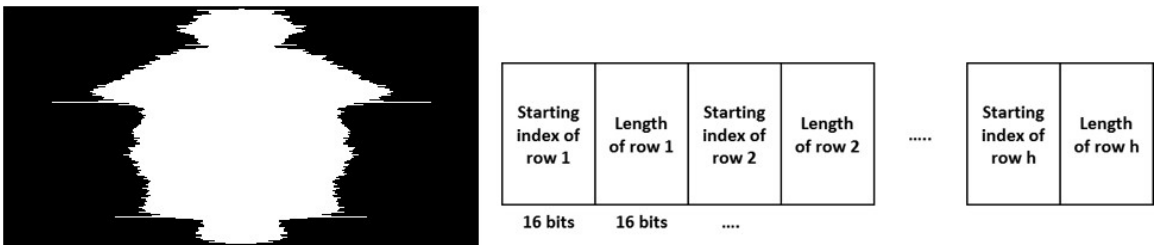


Figure 3.6: A sample binary map image and its corresponding binary map bitstream.

manner and written to the 2-D frame line by line. The result are two 2-D frames, one of which is for the geometry attribute, and the other is for color.

In addition, to reconstruct the point cloud correctly, a binary map that indicates the position of points that retain after the subsampling process is stored. Figure 3.6 shows a sample binary map image.

3.4 Evaluation

3.4.1 Encoding Profile

We use HEVC 4:4:4 Screen content coding [30] profile with an intra period of either 1 or 32 and a GOP size of 1 (no B frame). HEVC 10-bit and 8-bit are used to encode the geometry and color attributes respectively.

3.4.2 Distortion Metric

To evaluate the distortion of the reconstruction point cloud, [31] proposes a metric that calculates the geometric and color distortions of the decoded point cloud. In short, this method finds a nearest neighbor in the original point cloud for each point in the reconstructed one (the vice versa search is also supported). Once a pair of correspondence is established, the direct geometric (point-to-point) distortion and the YUV color distortion are calculated. In addition, a point-to-plane distortion is produced based on the projected vector of the distance along the normal vector

direction. The latter distortion is supposed to provide better metric for the perceptual of the reconstructed point cloud since it estimates the error along the surface of the 3-D point cloud. A normal vector set is required to perform the point-to-plane distortion.

3.5 Results

We have written the 3-D mapping in Matlab and tested our proposed compression method for the complete Category-2 dataset of the MPEG PCC CfP which includes five dynamic point clouds at five bitrates within the limits specified in the MPEG CfP (Call for Proposal) [21]. We use HM codec using the following configuration:

- HEVC 4:4:4 Screen content coding profile.
- Geometry (XYZ) and color (RGB) is subsampled and coded either losslessly or in a lossy fashion.
- All-Intra and random-access coding with an intra period of 32.

Figure 3.7 and Figure 3.8 show the BD-rate comparison between our proposed scheme and MPEG PCC software for all-intra and random-access compression, respectively. Numbers with a green/red background indicate a higher/lower performance of our PCC compared to MPEG PCC. Numbers with a white background show an almost equal performance.

Although our lossy results are not objectively dominating those of MPEG PCC in all cases, subjective evaluation shows a significant gain toward our PCC. Screenshots of the reconstructed point clouds (still image of the 'Longdress' test point cloud) by [3] and the proposed approach, at different target birates are shown in Figure 3.9 to Figure 3.12 (the 0.6 bit per point is equal to the compression ratio of 90:1 for

the 10-bit geometry point cloud models). In addition, the screenshots of the reconstructed point clouds at different bitrates are shown in Figure 3.13 to Figure 3.17. In addition, the mapping algorithm has a very-high capability for parallel processing since each row of the point cloud model is independently mapped into a line of the 2-D video frame.

3.6 Conclusion

This section presents our approach for encoding 3-D point cloud videos. The proposed algorithm maps the 3-D models into 2-D frames and uses the state-of-the-art HEVC to encode the video. The presented approach takes advantage of the advanced video codec and is low complex and parallel processing-capable. The results show that the proposed method produces better visual quality than the anchor software from MPEG at the time of evaluation.

					Bottom 4 Points BD-rate (piecewise cubic)					Top 4 Points BD-rate (piecewise cubic)				
		Target Rate (Mbit/s)	# Frames	fps	Geo. PSNR D1 (dB)	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)	Geo. PSNR D1 (dB)	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)
Class A	Queen	55.00	250	50	-22.7%	-71.4%	54.7%	-78.0%	-64.3%	-26.2%	-62.6%	32.1%	-88.0%	-79.9%
		30.00	250	50										
		15.00	250	50										
		5.00	250	50										
		3.00	250	50										
	8i VFB – Loot	27.00	300	30	-20.9%	-77.6%	-7.0%	0.0%	0.0%	-19.5%	-63.5%	-20.9%	0.0%	0.0%
		16.00	300	30										
		8.00	300	30										
		5.00	300	30										
		3.50	300	30										
	8i VFB – Red_and_Black	30.00	300	30	-15.9%	-77.2%	21.0%	0.0%	0.0%	-12.8%	-57.4%	7.9%	0.0%	0.0%
		18.00	300	30										
		9.00	300	30										
		6.00	300	30										
		3.50	300	30										
	8i VFB – Soldier	37.10	300	30	-13.8%	0.0%	44.7%	-76.0%	-81.3%	-14.7%	-65.0%	26.4%	-79.4%	-80.4%
		20.00	300	30										
		11.00	300	30										
		6.00	300	30										
		3.50	300	30										
Class B	8i VFB – Long_dress	42.70	300	30	-18.9%	0.0%	41.6%	-84.1%	-81.3%	-20.0%	-85.7%	17.9%	0.0%	-85.7%
		27.00	300	30										
		13.00	300	30										
		6.00	300	30										
		3.90	300	30										
	Avg. for Class A				-18.3%	-56.6%	28.4%	-38.5%	-36.4%	-18.3%	-62.1%	11.4%	-41.8%	-40.1%
	Avg. for Class B				-18.9%	0.0%	41.6%	-84.1%	-81.3%	-20.0%	-85.7%	17.9%	0.0%	-85.7%
	Overall Average				-18.4%	-45.3%	31.0%	-47.6%	-45.4%	-18.7%	-66.8%	12.7%	-33.5%	-49.2%

Figure 3.7: The BD-rate comparison of our proposed PCC compared to MPEG PCC for all-intra coding.

					Bottom 4 Points BD-rate (piecewise cubic)					Top 4 Points BD-rate (piecewise cubic)				
		Target Rate (Mbit/s)	# Frames	fps	Geo. PSNR D1 (dB)	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)	Geo. PSNR D1 (dB)	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)
Class A	Queen	55.00	250	50	-28.2%	-82.7%	20.3%	-85.9%	-75.3%	-27.6%	-81.3%	-6.8%	0.0%	-87.7%
		30.00	250	50										
		15.00	250	50										
		5.00	250	50										
		3.00	250	50										
	8i VFB – Loot	27.00	300	30	-39.5%	-75.3%	-21.2%	0.0%	0.0%	-19.5%	-63.5%	-20.9%	0.0%	0.0%
		16.00	300	30										
		8.00	300	30										
		5.00	300	30										
		3.50	300	30										
	8i VFB – Red_and_Black	30.00	300	30	-12.4%	-96.2%	58.5%	-96.3%	-95.1%	-7.2%	0.0%	11.9%	0.0%	0.0%
		18.00	300	30										
		9.00	300	30										
		6.00	300	30										
		3.50	300	30										
	8i VFB – Soldier	37.10	300	30	2.9%	-97.0%	88.7%	-80.8%	-71.9%	5.5%	-84.7%	34.8%	-64.4%	22.2%
		20.00	300	30										
		11.00	300	30										
		6.00	300	30										
		3.50	300	30										
Class B	8i VFB – Long_dress	42.70	300	30	-33.4%	-96.5%	43.5%	-97.0%	-96.2%	-21.4%	-86.1%	16.3%	0.0%	-86.1%
		27.00	300	30										
		13.00	300	30										
		6.00	300	30										
		3.90	300	30										
	Avg. for Class A				-19.3%	-87.8%	36.6%	-65.7%	-60.6%	-12.2%	-57.4%	4.8%	-16.1%	-16.4%
	Avg. for Class B				-33.4%	-96.5%	43.5%	-97.0%	-96.2%	-21.4%	-86.1%	16.3%	0.0%	-86.1%
	Overall Average				-22.1%	-89.5%	38.0%	-72.0%	-67.7%	-14.0%	-63.1%	7.1%	-12.9%	-30.3%

Figure 3.8: The BD-rate comparison of our proposed PCC compared to MPEG PCC for random access.

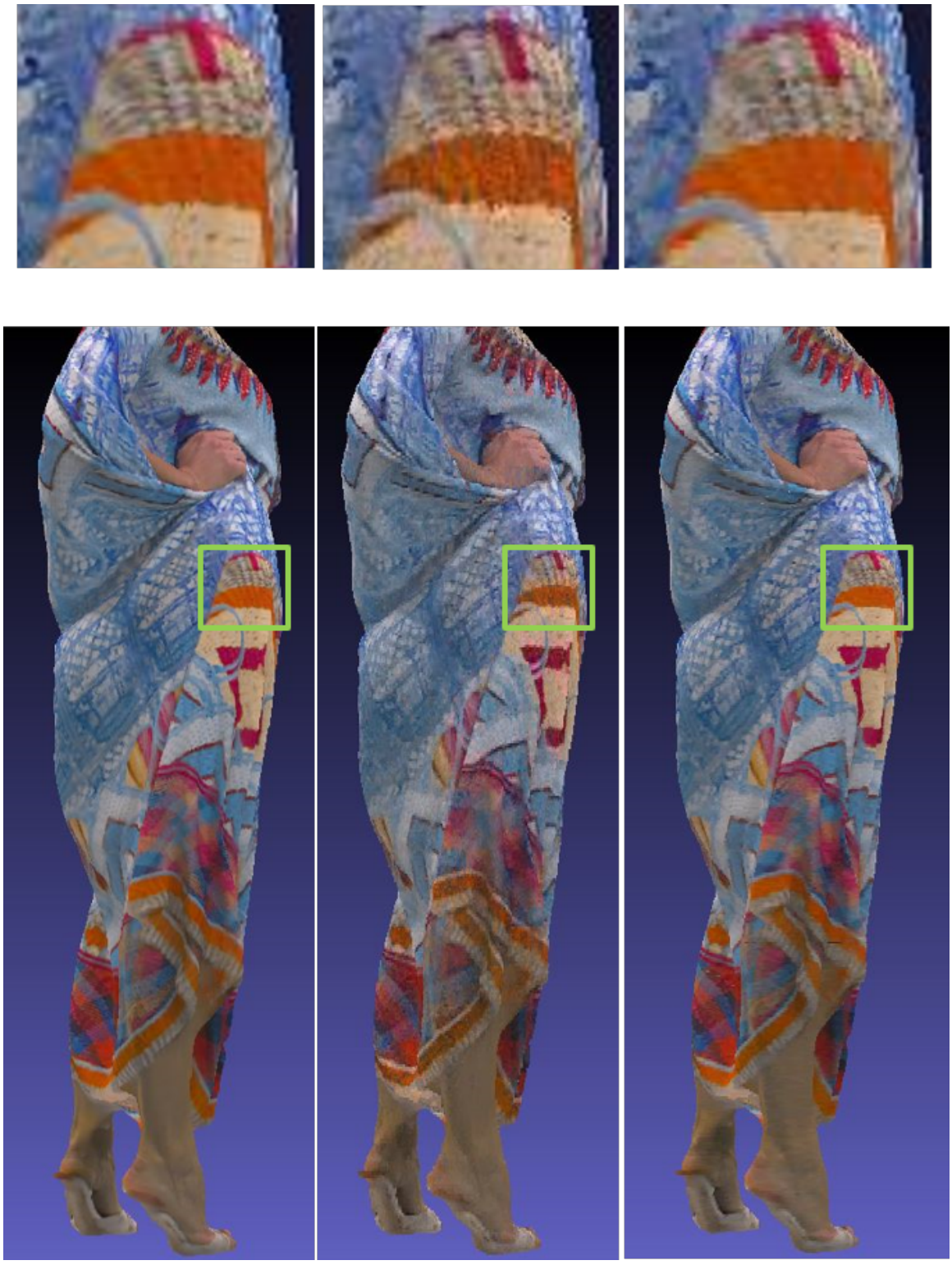


Figure 3.9: Single point cloud frame distortion at 8 bit per point. Left: original, middle and right: compressed/reconstructed by [3] and the proposed method respectively.

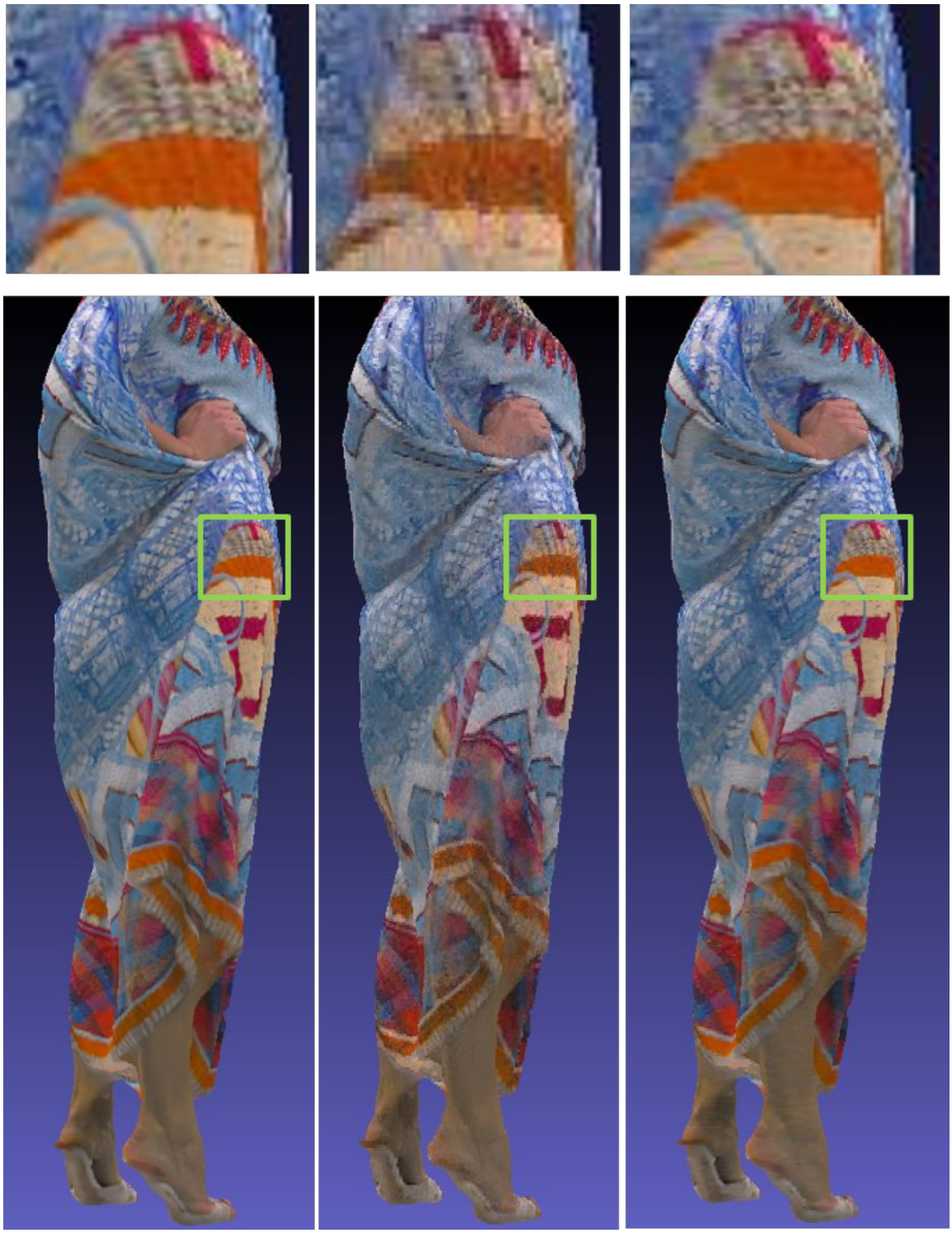


Figure 3.10: Single point cloud frame distortion at 4.5 bit per point. Left: original, middle and right: compressed/reconstructed by [3] and the proposed method respectively.

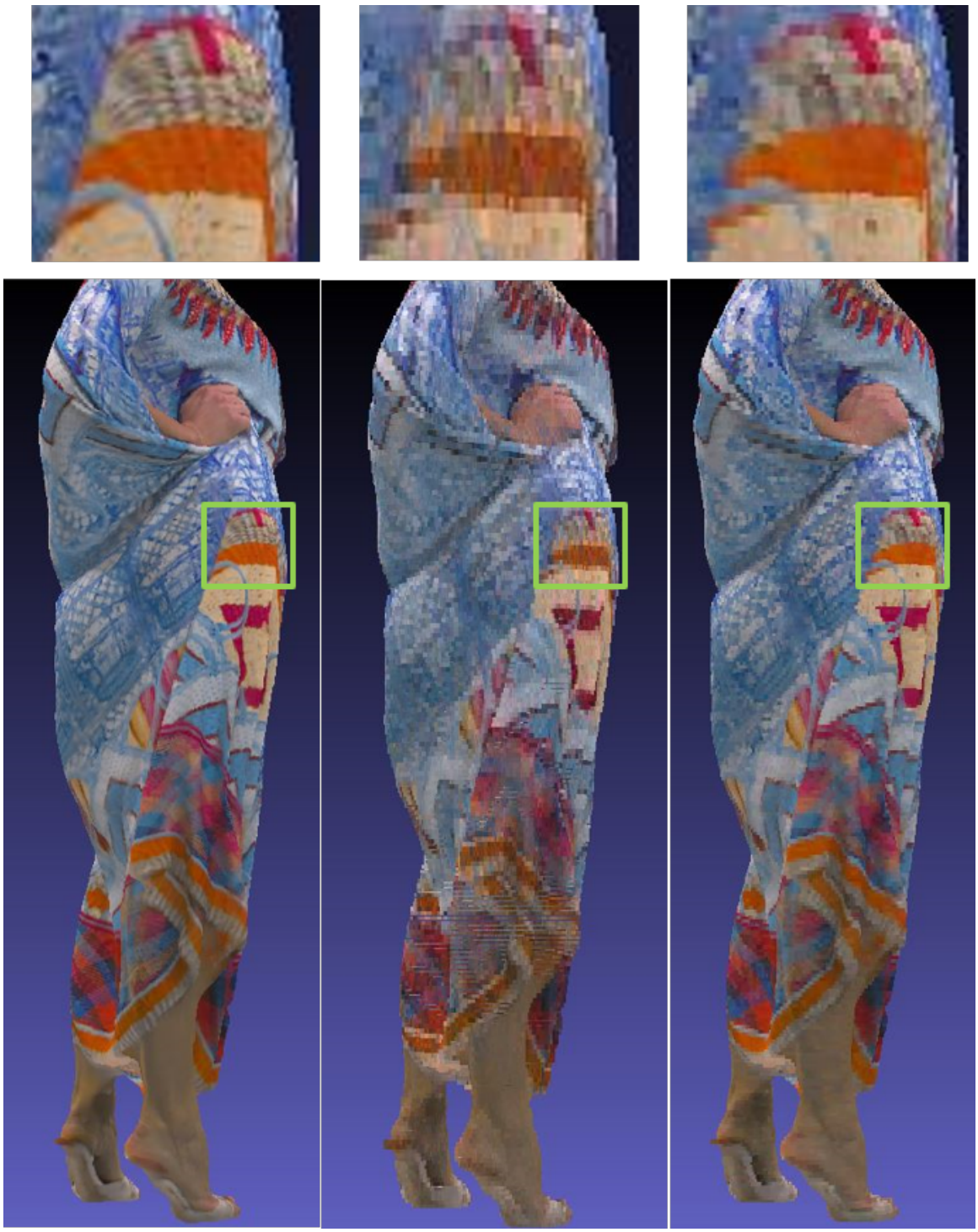


Figure 3.11: Single point cloud frame distortion at 2 bit per point. Left: original, middle and right: compressed/reconstructed by [3] and the proposed method respectively.

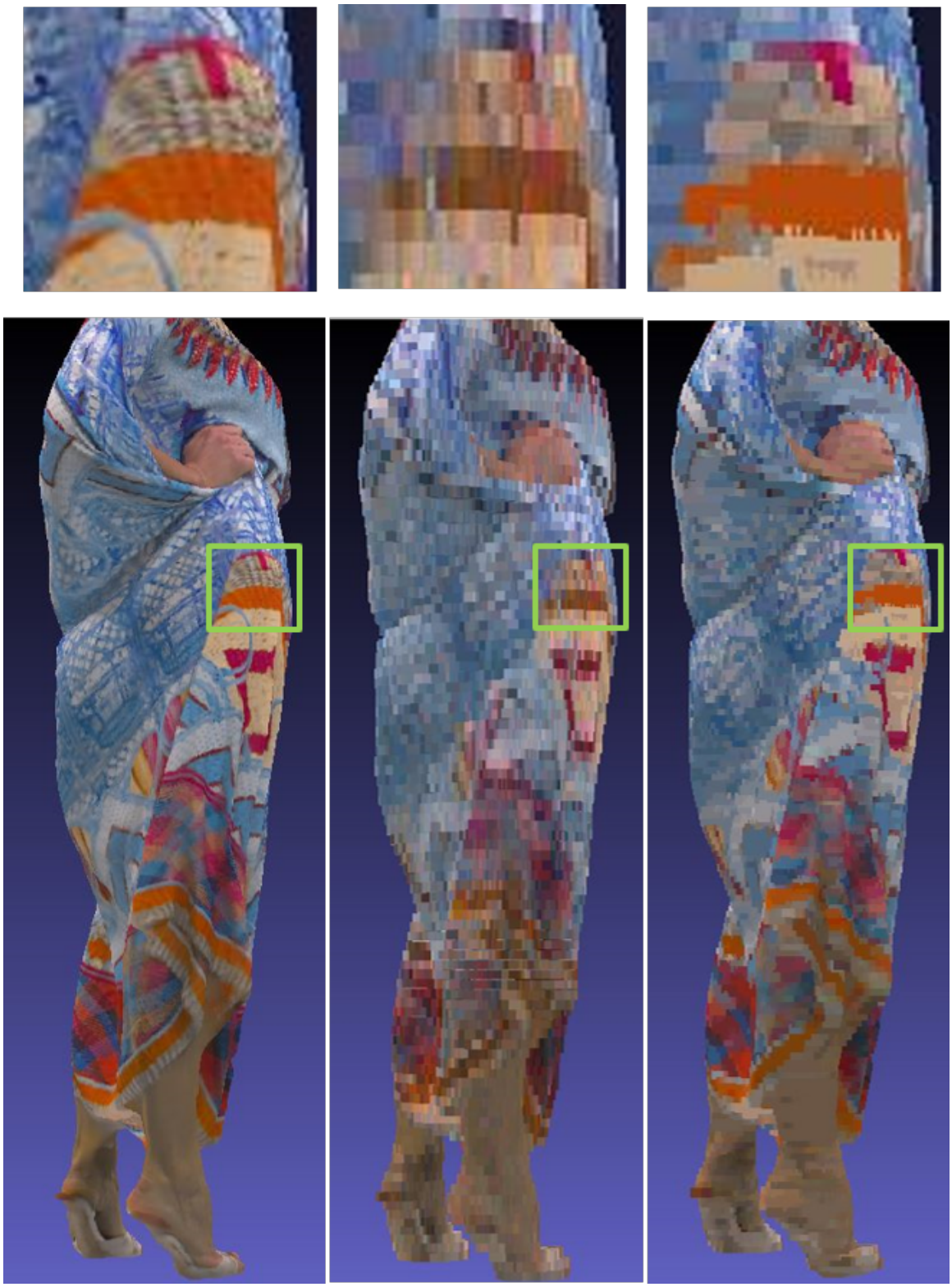


Figure 3.12: Single point cloud frame distortion at 0.6 bit per point. Left: original, middle and right: compressed/reconstructed by [3] and the proposed method respectively.

Figure 3.13: Screenshot of the reconstructed point cloud 'Longdress' at different bitrates.

Figure 3.14: Screenshot of the reconstructed point cloud 'Loot' at different bitrates.

Figure 3.15: Screenshot of the reconstructed point cloud 'Queen' at different bitrates.

Figure 3.16: Screenshot of the reconstructed point cloud 'RedandBlack' at different bitrates.

Figure 3.17: Screenshot of the reconstructed point cloud 'Soldier' at different bitrates.

REFERENCES

[1] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library computer vision algorithms," in *Proc. of the 18th ACM international conference on Multimedia*, 2010.

[2] **T. Ho** and M. Budagavi, "Dual-fisheye lens stitching for 360-degree imaging," in *Proc. of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'17)*, Mar. 2017.

[3] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, pp. 828–842, April 2017.

[4] "Google Cardboard," https://vr.google.com/cardboard, [Accessed August, 2016].

[5] "Samsung GearVR," www.samsung.com/us/gearvr, [Accessed August, 2016].

[6] S. K. Nayar, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, 1997.

[7] D. Scaramuzza, A. Martinelli, and R. Siegwart, "Catadioptric omnidirectional camera," in *Proc. of IEEE International Conference on Vision Systems (ICVS'06)*, 2006.

[8] ——, "A toolbox for easy calibrating omnidirectional cameras," in *Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS'06)*, 2006.

[9] "Nokia Ozo," https://ozo.nokia.com/, [Accessed August, 2016].

[10] "GoPro Odyssey," https://gopro.com/odyssey, [Accessed August, 2016].

[11] "Facebook Surround360," https://facebook360.fb.com/facebook-surround-360, [Accessed August, 2016].

[12] M. Brown and D. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, pp. 59–73, August 2007.

[13] R. Szeliski, Ed., *Computer Vision: Algorithms and Applications*, 1st ed. London: Springer, 2011.

[14] **T. Ho** and M. Budagavi, "Lens shading parameters metadata for omnidirectional video," ISO/IEC JTC1/SC29/WG11 MPEG2047/m39469 (adopted proposal), Oct. 2016.

[15] ——, "360-degree video stitching," U.S. Patent Pending, May, 2017.

[16] "Equirectangular projection," en.wikipedia.org/wiki/Equirectangularprojection, [Accessed August, 2016].

[17] J. Lewis, "Fast template matching," in *Vision Interface 95*. Canadian Image Processing and Pattern Recognition Society, 1995.

[18] **T. Ho**, I. D. Schizas, K. R. Rao, and M. Budagavi, "360-degree video stitching for dual-fisheye lens cameras based on rigid moving least squares," in *Proc. of the 24th IEEE International Conference on Image Processing (ICIP'17)*, Sep. 2017.

[19] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares," in *Proc. of ACM SIGGRAPH '06*, 2006.

[20] "Call for proposals for point cloud compression v2," ISO/IEC JTC1/SC29/WG11 N16763, Apr. 2017.

[21] M. Budagavi, E. Faramarzi, **T. Ho**, H. Najaf-Zadeh, and I. Sinharoy, "Samsung's response to cfp for point cloud compression (category 2)," ISO/IEC JTC1/SC29/WG11 MPEG2017/M41808 (proposal), Oct. 2017.

[22] M. Budagavi, E. Faramarzi, and **T. Ho**, "Point cloud compression," U.S. Provisional Patent, 2017.

[23] M. Hosseini, **T. Ho**, M. Budagavi, and I. Bouazizi, "Point cloud streaming," U.S. Provisional Patent, 2017.

[24] **T. Ho**, M. Budagavi, E. Faramarzi, H. Najaf-Zadeh, and I. Sinharoy, "Point cloud compression," in *Paper draft*.

[25] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 440 – 453, Jan. 2008.

[26] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proceeding SPBG'06 Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*, July 2006.

[27] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[28] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129 – 147, 1982.

[29] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[30] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging hevc screen content coding extension," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 50–62, Jan 2016.

[31] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Updates and integration of evaluation metric software for pcc," ISO/IEC JTC1/SC29/WG11 MPEG2016/M40522, Apr. 2017.

ACRONYM

SIFT	Scale-invariant feature transform
RANSAC	Random sample consensus
MPEG	Moving Picture Experts Group
JCT-VC	Joint Collaborative Team on Video Coding - ITU