IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 21, 2019, accepted January 4, 2020, date of publication January 8, 2020, date of current version January 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964789

Spherical Lanczos Interpolation in Planar Projection or Format Conversions of Panoramic Videos

SAIPING ZHANG[1], (Student Member, IEEE), FUZHENG YANG[1], (Member, IEEE), SHUAI WAN[2], (Member, IEEE), AND PEIYUN DI[3]

[1]The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[2]The School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China
[3]HUAWEI Technologies Company Ltd., Shenzhen 518129, China

Corresponding author: Saiping Zhang (spzhang@stu.xidian.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61571337 and Grant 61601349, in part by the 111 Project under Grant B08038, in part by the China Postdoctoral Science Foundation Funded Project under Grant 2016M592757, and in part by the Fundamental Research Funds for the Central Universities under Grant JB180105.

ABSTRACT Panoramic videos provide users with an amazing experience with immersive and interactive viewing, and are now gaining global popularity. Since panoramic videos are spherical in nature while effectively processed in planar projection formats in applications, it is crucial to implement sphere-to-plane projection, where a good interpolation algorithm is of great importance regarding generating high quality planar perspective videos. Moreover, different planar projection formats adapt to different applications according to their own characteristics. It is common to convert one projection format to another, where interpolation algorithms also make a difference on performance. In this paper, an interpolation algorithm named spherical Lanczos (SLAN) is proposed to achieve advanced performance in sphere-to-plane projection or planar projection format conversions of panoramic videos. In SLAN, we fully consider panoramic videos are naturally spherical. Reference pixels are selected carefully, and the weights are calculated correctly. Experimental results demonstrate that the SLAN interpolation algorithm is beneficial for the whole processing chain of panoramic videos both with lossless compression coding and with loss compression coding in terms of the increase in various end-to-end *PSNR* and the decrease in *BD_rate*. The cubic projection (CBP) and adjusted cubemap projection (ACP) are also evaluated to illustrate that the proposed algorithm is applicable for all kinds of planar projection format due to their inevitable deformation.

INDEX TERMS Spherical interpolation, sphere-to-plane projection, planar format conversion, panoramic video.

I. INTRODUCTION

The recent years have witnessed an explosion of various video services, especially for immersive panoramic videos. The panoramic video or omnidirectional video [1], [2], which is generally captured by multiple high definition cameras with the captured frames stitched [3], [4], represents time-varying 360-degree environment in a spherical way. It can be played via immersive virtual reality (VR) players [5]–[7] overcoming the structured limits of how conventional video imagery is presented and perceived. Users can pan, zoom in/out some regions and freely change the viewing angle according to their interests while watching. Therefore, instead of being passive consumers, users of panoramic videos can interact with what they are viewing.

The panoramic video is appealing to users due to its immersive user experience, however, new challenges arise in its storage, encoding, transmission and display. Since all pixels of the scene in 360-degree are all stored for panoramic videos, huge amount of data is generally involved, resulting in a quite high resolution of video frames. Moreover, panoramic videos are spherical in nature, while existing video coding standards are originally designed for planar perspective videos. Sphere-to-plane projection [8] is the present solution for panoramic videos to effectively store, transmit [9], code [10], [11] with the off-the-shelf video coding standards and display. How to

The associate editor coordinating the review of this manuscript and approving it for publication was Xinfeng Zhang.

design reasonable projection of panoramic videos has been put in schedule of the next generation of video coding being developed by the Joint Video Exploration Team (JVET).

Many sphere-to-plane projection formats have been proposed in recent years, such as the equirectangular projection (ERP) format [12], cubic projection (CBP) format [13], adjusted cubemap projection (ACP) format [14], Craster parabolic projection (CPP) format [15], octahedron projection (OHP) format [16], icosahedral projection (ISP) format [17], segmented sphere projection (SSP) format [18], rotated sphere projection (RSP) format [19], equi-angular cubemap (EAC) format [20] and octagonal projection format (OGP) [21]. Among those formats, the ERP and CBP are regarded as the common projection formats and included in Omnidirectional MediA Format (OMAF) standard [22], which, in development by the Moving Picture Experts Group (MPEG) at present, standardizes means for representation of 360-degree videos. ERP unfolds the frame of panoramic videos using a longitude and latitude grid, which is computationally simple and has good continuity of texture in frames. Therefore, the QuickTime VR system [23] is developed to model a three-dimensional (3-D) environment in the form of a frame in the ERP format. Besides, CBP projects the frame of panoramic videos onto six square faces of a cube. Its rectilinear structure and computational simplicity make it suitable for GPUs so that the speed of processing is relatively high.

With the help of reasonable sphere-to-plane projection formats, panoramic videos can be stored, coded, transmitted and displayed effectively with existing techniques, which promotes the development of panoramic video services at present. In the process of sphere-to-plane projection of panoramic videos, integer pixels in the planar perspective frames are always projected from sub-pixels on the sphere according to mathematical calculations. An interpolation algorithm, as an extremely important step in sphere-to-plane projection, is explored to approximate the values of sub-pixels on the sphere. A good interpolation algorithm will benefit sphere-to-plane projection regarding generating high quality planar perspective videos. Moreover, considering the characteristics of planar projection formats, it is natural that different projection formats adapt to different applications, for example, panoramic videos are coded in CPP, while prefer to be rendered in ERP. Format conversions are inevitable. Interpolation algorithms, which make a difference on performance in format conversions, are fundamental.

Lanczos interpolation algorithm [24], as a popular interpolation algorithm, has been considered as a good compromise between the computational complexity and performance among several available interpolation algorithms [25]. It is generally adopted by the JVET to achieve sphere-to-plane projection or format conversions in 360Lib-4.0 software [26]. It is worth emphasizing that Lanczos interpolation algorithm here is implemented directly on the plane. Considering that panoramic videos are spherical, though effectively processed in the planar projection format, the pixels are related on the sphere originally. As mentioned in [27], there is inevitable deformation in planar perspective frames in all projection formats after sphere-to-plane projection. It is this deformation that destroys the original relation of pixels. In other words, pixels highly related to each other on the plane may be weakly related to each other on the sphere. We argue that, in such a case, interpolation algorithm should be implemented on the sphere for better performance. It appears that reference pixels in Lanczos interpolation algorithm are not properly selected, and the weights are not correctly calculated. The performance of Lanczos interpolation algorithm is therefore limited. Specially, the stronger the deformation of planar projection format is, the more severely the performance is penalized.

In this paper, a spherical Lanczos (SLAN) interpolation algorithm is proposed achieving advanced performance in sphere-to-plane projection or planar projection format conversions of panoramic videos. It is noted that the proposed SLAN interpolation algorithm is detailed in the format conversion from CPP to ERP for example in this paper, and it is applicable for all kinds of planar projection format conversions of panoramic videos. Firstly, a set of reference pixels neighboring the sub-pixel to be interpolated are selected along the deformed meridians and the parallels in frames in the original format, i.e., CPP in the example. Then the spherical distances between the reference pixels and the sub-pixel to be interpolated are measured. The weight for interpolation of each reference pixel is further calculated according to the interpolation filter's reconstruction kernel. Finally, the value of the sub-pixel to be interpolated is calculated. Similar process can be applied for all other projection format conversions, and advanced performance is always observed using the proposed algorithm.

The rest of this paper is organized as follows. Background is reviewed in section II. The details of the proposed SLAN interpolation algorithm are given in section III. Experimental results are presented in section IV. Finally, section V concludes the paper.

II. BACKGROUND

Many sphere-to-plane projection formats have been proposed in recent years for reasonable and effective storage, encoding and transmitting panoramic videos. Commonly used projection formats include ERP, CBP, CPP, ACP and etc. [12]–[15]. ERP [12] divides the sphere into many regions with constant spacing latitude and longitude, samples at intersection of the parallel and meridian, and arranges all the pixels into a rectangle on the plane. The computational simplicity and good continuity of texture in frames of ERP make it widely adopted to store panoramic video sources. However, there is a severe oversampling problem resulting in wasted pixels especially when the sampling areas are near the poles. CBP [13] is also a common simple projection format resulting in six square faces which are suitable for planar frame packing.

Its memory-friendly rectilinear structure and computational simplicity make it directly supported by most GPUs structure so that the processing speed is appealing. However, non-uniform sampling on the sphere penalizes its representation effectiveness. ACP [14] is based on CBP targeting uniform sphere sampling while fully utilizing the hardware support of cube projection, whereas several seams are generated after planar frame packing which have a negative effect on encoding. CPP [15] achieves approximately uniform sphere sampling with no seams inside the frames. It is considered as a projection format that does not contain any redundancy pixels compared with ERP, CBP and ACP [15]. One drawback in the CPP format, however, is comparatively strong deformation after projection in frames.

Different projection formats are suitable to different applications in practice according to their own characteristics. It is inevitable to implement projection format conversions. Format conversion from CPP to ERP is illustrated as an example.

To generate frames in ERP from CPP, the value of each integer pixel in frames in ERP should be calculated. Notice that, the width and height of frames in the ERP format are equal to the width and height in the CPP format, respectively. To find out the value of the integer pixel at (m, n) in frames in ERP, the corresponding pixel at (x, y) in frames in CPP is required. The value of the pixel (x, y), which can be calculated using (1), is regarded as the value of the pixel (m, n).

$$\begin{cases} x = \frac{W}{2} + \left(m + \frac{1-W}{2}\right) \\ \quad \times \left[2\cos\left[\frac{2\pi}{3}\left(\frac{1}{2} - \frac{n+\frac{1}{2}}{H}\right)\right] - 1\right] \\ y = \frac{H}{2} - H \sin \frac{\pi \times \left(\frac{1}{2} - \frac{n+\frac{1}{2}}{H}\right)}{3} \end{cases} \tag{1}$$

where W and H are the width and height of frames, respectively.

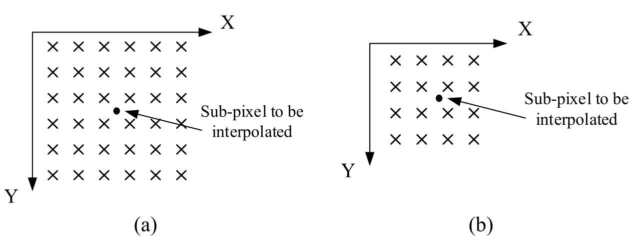


FIGURE 1. Reference pixels selected by Lanczos interpolation algorithm in frames in CPP. (a) Reference pixels selected for luminance. (b) Reference pixels selected for chrominance.

Generally, the corresponding pixel found in frames in CPP is always a sub-pixel according to (1). To approximate the value of the required sub-pixel, Lanczos interpolation algorithm is adopted in 360Lib-4.0 software [26]. It usually selects 36 (for luminance) or 16 (for chrominance) neighboring integer pixels closest to the sub-pixel to be interpolated as the reference pixels, as shown in Figure 1.

The weight of each reference pixel is calculated according to the distance between the reference pixel and the sub-pixel to be interpolated measured on the plane. In detail, the distances in the directions of the X axis and Y axis between the reference pixel and the sub-pixel to be interpolated are measured to calculate the weights for interpolation as

$$L(tx) = \begin{cases} 1 & \text{if } tx = 0 \\ \frac{a \sin(\pi tx) \sin(\pi tx/a)}{\pi^2 tx^2} & \text{if } -a \le tx \le a \text{ and } tx \ne 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and

$$L(ty) = \begin{cases} 1 & \text{if } ty = 0 \\ \frac{a \sin(\pi ty) \sin(\pi ty/a)}{\pi^2 ty^2} & \text{if } -a \le ty \le a \text{ and } ty \ne 0 \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

where tx and ty represent the distances in the directions of the X axis and Y axis between the reference pixel and the sub-pixel to be interpolated, and $L(tx)$ and $L(ty)$ represent the weights for interpolation in the horizontal and vertical directions, respectively. In (2) and (3), a is the size of Lanczos window, which equals to 3 for luminance and 2 for chrominance. The weight for interpolation of the reference pixel in two dimensions can be finally calculated by

$$L_{xy} = L(ty) \times L(tx) \tag{4}$$

where L_{xy} represents the weight of the reference pixel in two dimensions.

After calculating the weights for interpolation of all reference pixels, the value of the sub-pixel to be interpolated can be approximated by the weighted average of values of all reference pixels.

However, panoramic videos are spherical in nature. There is strong deformation in frames in the CPP format after sphere-to-plane projection. This deformation does destroy the original relation of pixels. As shown in Figure 2 (a) and (b), the reference pixels selected are in a square distribution on the plane while they are in a parallelogram distribution on the sphere. In terms of distances measured on the sphere, the reference pixels selected in frames in the CPP format by Lanczos interpolation algorithm are not actually the closest integer pixels to the sub-pixel to be interpolated. As shown in Figure 3, the pixel represented by a square is closer to the sub-pixel to be interpolated on the sphere than the pixel represented by a bold cross, which is actually selected as a reference pixel in Lanczos interpolation algorithm. Apparently, the reference pixels are not properly selected in such a case. Considering that pixels in frames of panoramic videos are originally distributed along the parallels and meridians

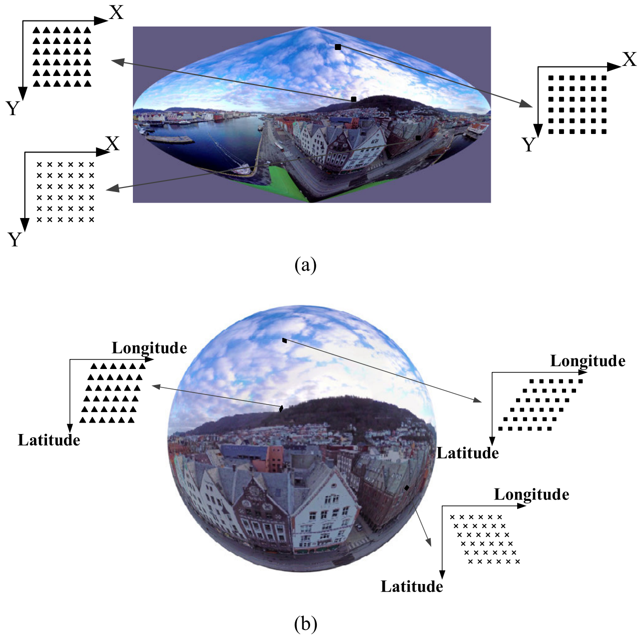


FIGURE 2. The distributions of reference pixels. (a) The distributions of reference pixels on the plane. (b) The distributions of reference pixels on the sphere.

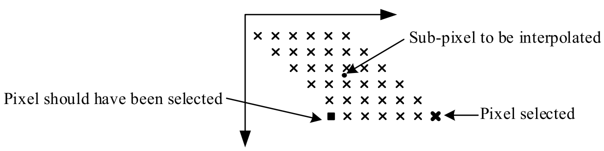


FIGURE 3. The selection of reference pixels.

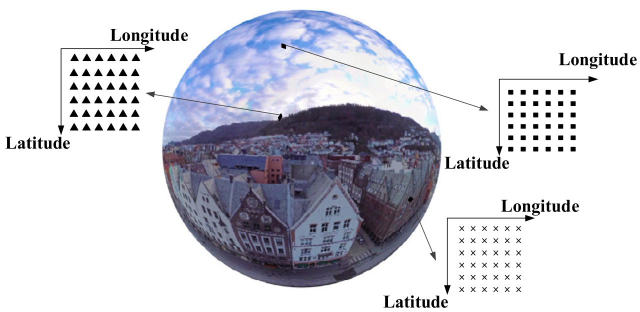


FIGURE 4. The spherical distributions of reference pixels that should be selected.

on the sphere, integer pixels that are neighboring the sub-pixel to be interpolated and distributed along the parallels and meridians should be selected as reference pixels. Compared with Figure 2 (b), the distributions of reference pixels which should be selected are shown in Figure 4. Approximately, these reference pixels are in a square distribution on the sphere in a local view.

Furthermore, the weights for interpolation are calculated according to the distances measured in frames in the CPP format in Lanczos interpolation algorithm. Due to deformation in frames in the CPP format, distances between reference pixels and the sub-pixel to be interpolated on the plane are quite different from those between them on the sphere. Actually, the distances measured on the sphere better reflect the original relation of pixels because panoramic videos are naturally spherical. Accordingly, the weights for interpolation should be calculated according to the spherical distances using the interpolation filter's reconstruction kernel.

Through the above analysis, a new interpolation algorithm named as SLAN is proposed in this paper, where the reference pixels are selected spherically and the distances between reference pixels and the sub-pixel to be interpolated are measured on the sphere.

III. PROPOSED SLAN INTERPOLATION ALGORITHM

The block diagram of the proposed SLAN interpolation algorithm is shown in Figure 5, where format conversion from CPP to ERP is used as an example. The interpolation algorithm should be implemented in frames in the CPP format in such a case. Similar spirit can also be applied to all other planar projection format conversions.

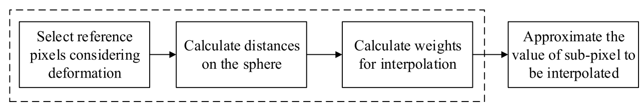


FIGURE 5. Block diagram of proposed SLAN algorithm.

As shown in Figure 5, firstly, considering the deformation due to the sphere-to-plane projection, a set of integer pixels are selected as reference pixels along the deformed parallels and meridians in frames in the CPP format. The coordinates of these reference pixels are calculated mathematically with low computational complexity. Then the distances between reference pixels and the sub-pixel to be interpolated are measured on the sphere. Unit distances for interpolation are also calculated. Then the weight for interpolation of each reference pixel is determined by the ratio of the distance and the unit distance. After the above steps, the value of the sub-pixel to be interpolated can be approximated by the weighted average of values of all reference pixels.

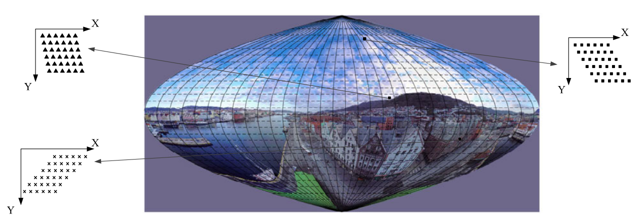


FIGURE 6. Distributions of reference pixels in frames in the CPP format.

A. SELECTION OF REFERENCE PIXELS CONSIDERING DEFORMATION

There is strong deformation in frames in the CPP format after sphere-to-plane projection. Different deformation is in different areas resulting in different distributions of reference pixels selected in frames in the CPP format. As shown in Figure 6, the distributions of these reference pixels should be along the deformed parallels and meridians which are represented by dashed lines and solid lines respectively. The specific steps to select reference pixels are as follows.

After finding the corresponding sub-pixel in frames in CPP for each integer pixel in frames in ERP using (1), the sub-pixel represented by (x, y) is to be interpolated. Assume that the spherical coordinate of this sub-pixel is (ϕ, λ) before projected in the plane (where the sphere is regarded as the unit sphere in this paper). ϕ and λ represent the latitude and the longitude, respectively. The relation between (x, y) and (ϕ, λ) is presented as

$$\begin{cases} x = W\dfrac{\lambda\left(2\cos\frac{2\phi}{3} - 1\right) + \pi}{2\pi} \\ y = H\dfrac{\pi\sin\frac{\phi}{3} + \frac{\pi}{2}}{\pi} \end{cases} \quad x \in [0, W], y \in [0, H], \tag{5}$$

Combining simultaneous equations in (5), the relation between x and y for a given λ is presented as

$$x = \frac{W}{2} + \frac{W}{2\pi} \times \lambda \times \left[1 - 4 \times \left(\frac{1}{2} - \frac{y}{H}\right)^2\right], \tag{6}$$

After differentiating for both sides of (6), the slope of the tangent at the sub-pixel (x, y) of the meridian can be further calculated by

$$\frac{dy}{dx} = \frac{1}{\frac{8}{\pi} \times \lambda \times \left(\frac{1}{2} - \frac{y}{H}\right)}, \tag{7}$$

where dx and dy represent the variation of the values of x and y, respectively. $\frac{dy}{dx}$ represents the slope of the tangent at the sub-pixel (x, y) of the meridian whose longitude is λ.

The direction of the tangent can be approximately regarded as the direction of the meridian at the sub-pixel (x, y). Besides, the directions of the parallels are all horizontal in frames in CPP. The distribution of reference pixels should be along these two directions as mentioned above.

After determining the distribution of reference pixels for the sub-pixel (x, y), the coordinates of reference pixels in frames in the CPP format can be calculated. Assume that the coordinates of reference pixels are (x_{ij}, y_i), $i \in 1, 2, \ldots, N, j \in 1, 2, \ldots, N$. 36 ($6 \times 6$) reference pixels and 16 (4×4) reference pixels are selected for luminance and chrominance respectively for YUV420 sampling format in this section, that is, $N = 6$ for luminance and $N = 4$ for chrominance. Firstly, the value of y_i calculate by

$$y_i = \lfloor y \rfloor + \Delta y_i, \tag{8}$$

where $\lfloor\rfloor$ means rounding down, $\Delta y_i = (-2, -1, 0, 1, 2, 3)$ is for luminance corresponding to $N = 6$ and $\Delta y_j = (-1, 0, 1, 2)$ is for chrominance corresponding to $N = 4$.

After calculating the value of y_i, the value of x_{ij} will be calculated. According to (7), we can further find that

$$dx = dy \times \frac{8}{\pi}\lambda(\frac{1}{2} - \frac{y}{H}) \tag{9}$$

and

$$dx_i = dy_i \times \frac{8}{\pi}\lambda(\frac{1}{2} - \frac{y}{H}) = |y - y_i| \times \frac{8}{\pi}\lambda(\frac{1}{2} - \frac{y}{H}), \tag{10}$$

where $||$ represents the absolute value. dy_i represents the absolute value of the difference between y and y_i. It can be regarded as the variation from y to y_i. dx_i is the corresponding variation of x.

$x + dx_i$ is the abscissa of the center pixel of the pixels which are in the ith row. After rounding down $x + dx_i + \frac{1}{2}$ and adding the offset Δx_j, x_{ij} can be calculated by

$$x_{ij} = \left\lfloor x + dx_i + \frac{1}{2} \right\rfloor + \Delta x_j, \tag{11}$$

where $\Delta x_j = (-2, -1, 0, 1, 2, 3)$ is for luminance, and $\Delta x_j = (-1, 0, 1, 2)$ is for chrominance.

After the above calculations, the coordinates of reference pixels (x_{ij}, y_i) are acquired for the sub-pixel (x, y) to be interpolated.

B. CALCULATION OF DISTANCE ON THE SPHERE

After selecting the reference pixels, the distances between the reference pixel and the sub-pixel to be interpolated should be measured on the sphere to calculate weights for interpolation. Usually, the closer the distance, the stronger the correlations, which indicates the larger weight for interpolation. Pixels of panoramic videos are originally distributed on the sphere. Only the spherical distances can reflect the relation of pixels correctly.

The coordinates of reference pixels for a sub-pixel (x, y) in frames in CPP can be found out by (8) and (11), which has been discussed in section II. Given a reference pixel at (x_{ij}, y_i) in a frame in CPP, (12) is used for further calculating its spherical coordinates, as

$$\begin{cases} \phi_i = 3\arcsin\left(\dfrac{y_i}{H} - \dfrac{1}{2}\right) \\ \lambda_{ij} = \dfrac{\frac{2\pi x_{ij}}{W} - \pi}{2\cos\frac{2\phi_i}{3} - 1}, \end{cases} \tag{12}$$

where $\left(\phi_i, \lambda_{ij}\right)$ is the spherical coordinate of the reference pixel at (x_{ij}, y_i) in CPP.

Similarly, (13) is applied to calculate the spherical coordinate of the sub-pixel to be interpolated at (x, y) in a frame in CPP, as

$$\begin{cases} \phi = 3\arcsin\left(\dfrac{y}{H} - \dfrac{1}{2}\right) \\ \lambda = \dfrac{\frac{2\pi x}{W} - \pi}{2\cos\frac{2\phi}{3} - 1}, \end{cases} \tag{13}$$

where (ϕ, λ) is the spherical coordinate of the sub-pixel to be interpolated.

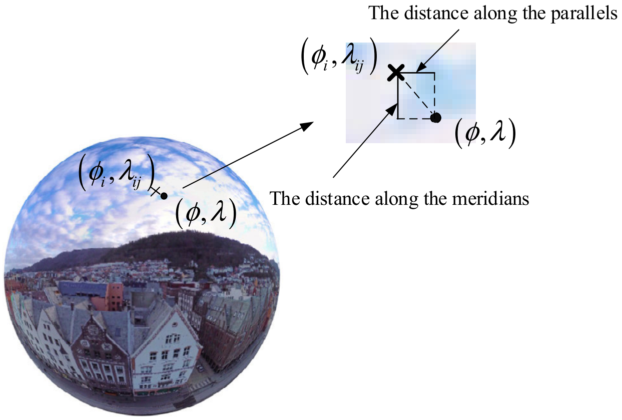


FIGURE 7. The distances along the parallels and the meridians on the sphere.

The distance measured on the sphere between (ϕ, λ) and (ϕ_i, λ_{ij}) should be considered in two directions for two-dimensional (2-D) interpolation. One is along the parallels and the other is along the meridians, as shown in Figure 7. Since the reference pixels are close enough to the sub-pixel to be interpolated on the sphere, the distances can be approximately calculated with lower computational complexity using

$$d_p = |\lambda - \lambda_{ij}| \cos \phi \tag{14}$$

and

$$d_m = |\phi - \phi_i|, \tag{15}$$

where d_p and d_m represent the distance along the parallels and the meridians between the reference pixel (ϕ_i, λ_{ij}) and the sub-pixel (ϕ, λ), respectively.

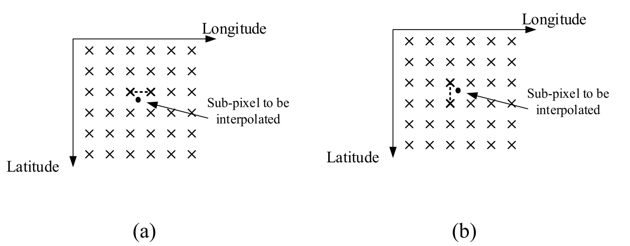


FIGURE 8. Unit distances on the sphere for luminance. (a) Unit distance along the parallels. (b) Unit distance along the meridians.

Besides, the unit distance for interpolation in the directions along the parallels and the meridians should also be calculated. The ratio of the spherical distance and the unit distance determines the weight for interpolation. Two reference pixels closest to the sub-pixel to be interpolated are selected to calculate the unit distance. For example, as for luminance, the reference pixels (ϕ_3, λ_{33}) and (ϕ_3, λ_{34}) represented by the bold crosses in Figure 8 (a) are selected to calculated the unit distance in the direction along the parallels. The reference pixels (ϕ_3, λ_{33}) and (ϕ_4, λ_{43}) represented by the bold crosses in Figure 8 (b) are selected to calculated the unit distance in the direction along the meridians. The unit distances in two directions are calculated by

TABLE 1. Simulation environment.

Reference Software	360Lib-4.0
Sequences	*AerialCity* *DrivingInCity* *DrivingInCountry* *PoleVault_le*
Resolution	3840x1920
Frames to be tested	300
Frame rate	30fps
Bit depth	8bit
Sampling Format	YUV420
Hardware Platform	Intel (R) Core(TM) i7-6700K CPU @ 4.00GHz

$$Ud_{p\_Y} = |\lambda_{33} - \lambda_{34}| \cos \phi_3 \tag{16}$$

and

$$Ud_{m\_Y} = |\phi_3 - \phi_4|, \tag{17}$$

where Ud_{p_Y} and Ud_{m_Y} represent the unit distance along the parallels and the meridians on the sphere for luminance, respectively.

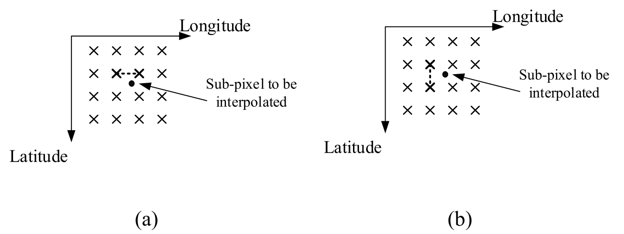


FIGURE 9. Unit distances on the sphere for chrominance. (a) Unit distance along the parallels. (b) Unit distance along the meridians.

As for chrominance, for example, the reference pixels (ϕ_2, λ_{22}) and (ϕ_2, λ_{23}) represented by the bold crosses in Figure 9 (a) are selected to calculated the unit distance in the direction along the parallels. The reference pixels (ϕ_2, λ_{22}) and (ϕ_3, λ_{32}) represented by the bold crosses in Figure 9 (b) are selected to calculated the unit distance in the direction along the meridians. Unit distances are calculated by

$$Ud_{p\_UV} = |\lambda_{22} - \lambda_{23}| \cos \phi_2 \tag{18}$$

and

$$Ud_{m\_UV} = |\phi_2 - \phi_3|, \tag{19}$$

where Ud_{p_UV} and Ud_{m_UV} represent the unit distance along the parallels and the meridians on the sphere for chrominance, respectively.

C. CALCULATION OF WEIGHTS FOR INTERPOLATION

The ratio of the distance and the unit distance determines the weight for interpolation and indicates the relation

TABLE 2. The comparison results of *PSNR* and *SPSNR_NN* for "CPP".

Sequences	Channel	PSNR (dB) Lanczos algorithm	PSNR (dB) SLAN algorithm	ΔPSNR (dB)	SPSNR_NN (dB) Lanczos algorithm	SPSNR_NN (dB) SLAN algorithm	ΔSPSNR_NN (dB)
AerialCity	Y	43.1783	45.2325	2.0542	44.0668	45.8622	1.7954
	U	50.6252	52.1514	1.5262	51.6210	52.9811	1.3601
	V	50.0315	51.5221	1.4906	51.0772	52.3945	1.3173
DrivingInCity	Y	47.3095	49.2384	1.9289	47.6208	49.3350	1.7142
	U	53.6696	55.3097	1.6401	54.7871	56.3292	1.5421
	V	53.0259	54.8124	1.7865	54.2095	55.8167	1.6072
DrivingInCountry	Y	40.3567	46.1189	5.7622	41.1829	46.1294	4.9465
	U	54.6676	56.3982	1.7306	55.3523	56.9503	1.5980
	V	54.8965	56.2955	1.3990	55.6971	56.9497	1.2526
PoleVault_le	Y	42.4857	47.2233	4.7376	44.1316	47.8694	3.7378
	U	45.2575	48.2285	2.9710	46.3680	48.7850	2.4170
	V	46.8319	49.8129	2.9810	47.9327	50.3744	2.4417
Average	Y	-	-	3.6207	-	-	3.0485
	U	-	-	1.9670	-	-	1.7293
	V	-	-	1.9143	-	-	1.6547

TABLE 3. The comparison results of *WS_PSNR* and *SPSNR_I* for "CPP".

Sequences	Channel	WSPSNR (dB) Lanczos algorithm	WSPSNR (dB) SLAN algorithm	ΔWSPSNR (dB)	SPSNR_I (dB) Lanczos algorithm	SPSNR_I (dB) SLAN algorithm	ΔSPSNR_I (dB)
AerialCity	Y	44.0640	45.8611	1.7971	45.0248	47.0458	2.0210
	U	51.6262	52.9774	1.3512	52.1924	53.4839	1.2915
	V	51.0790	52.3911	1.3121	51.7464	52.9920	1.2456
DrivingInCity	Y	47.6125	49.3341	1.7216	48.2524	50.8108	2.5584
	U	54.8095	56.3344	1.5249	54.8706	56.2758	1.4052
	V	54.2344	55.8317	1.5973	54.3350	55.8219	1.4869
DrivingInCountry	Y	41.1212	46.1097	4.9885	41.7190	47.4040	5.6850
	U	55.3409	56.9340	1.5931	55.3931	56.8285	1.4354
	V	55.6954	56.9432	1.2478	55.6792	56.8075	1.1283
PoleVault_le	Y	44.0974	47.9161	3.8187	44.7165	48.9999	4.2834
	U	46.3310	48.7516	2.4206	47.3012	49.8736	2.5724
	V	47.9090	50.3577	2.4487	48.7101	51.2326	2.5225
Average	Y	-	-	3.0815	-	-	3.6370
	U	-	-	1.7225	-	-	1.6761
	V	-	-	1.6515	-	-	1.5958

between the reference pixel and the sub-pixel. Two ratios are calculated by

$$tx = \frac{d_p}{Ud_p} \tag{20}$$

and

$$ty = \frac{d_m}{Ud_m}, \tag{21}$$

where d_p and d_m represent the spherical distance along the parallels and the meridians between the reference pixel and the sub-pixel to be interpolated. Ud_p and Ud_m represent the unit distance along the parallels and the meridians. tx and ty are two ratios, respectively.

Weights for interpolation for reference pixels in two directions are calculated respectively by (2) and (3). After that, the weights for interpolation of reference pixels in two dimensions can be calculated by (4). Finally, the value of the sub-pixel to be interpolated can be calculated by

$$P_A = \sum_{j=1}^{N}\sum_{i=1}^{N} L(x_{ij}, y_i) \times P_{ij.i}, \tag{22}$$

where P_A is the value of sub-pixel to be interpolated, $P_{ij.i}$ is the value of the reference pixel (x_{ij}, y_i), and N is 6 for luminance, and N is 4 for chrominance.

IV. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed SLAN interpolation algorithm, we compared with Lanczos interpolation algorithm which is generally adopted in 360Lib-4.0 software. We present experimental results both with lossless compression coding and with loss compression coding.

A. LOSSLESS COMPRESSION CODING

The simulation environment is shown in Table 1. It is worth emphasizing that all test sequences were downloaded in the

TABLE 4. The comparison results of *PSNR* and *SPSNR_NN* for "CBP".

Sequences	Channel	PSNR (dB) Lanczos algorithm	PSNR (dB) SLAN algorithm	ΔPSNR (dB)	SPSNR_NN (dB) Lanczos algorithm	SPSNR_NN (dB) SLAN algorithm	ΔSPSNR_NN (dB)
AerialCity	Y	45.0833	45.1845	0.1012	45.2817	45.4010	0.1193
	U	52.0600	52.1074	0.0474	52.8173	52.8728	0.0555
	V	51.3427	51.3880	0.0453	52.2742	52.3328	0.0586
DrivingInCity	Y	47.9707	48.0364	0.0657	47.6039	47.6701	0.0662
	U	54.9613	55.0007	0.0394	56.0649	56.1235	0.0586
	V	54.4160	54.4630	0.0470	55.5136	55.5712	0.0576
DrivingInCountry	Y	44.6954	44.7797	0.0843	44.4681	44.5568	0.0887
	U	56.3576	56.4078	0.0502	56.6445	56.7027	0.0582
	V	55.8704	55.8905	0.0201	56.5801	56.6346	0.0545
PoleVault_le	Y	44.8296	44.8528	0.0232	46.0656	46.1371	0.0715
	U	46.8792	46.9301	0.0509	47.9336	48.0021	0.0685
	V	48.4632	48.5150	0.0518	49.5491	49.6169	0.0678
Average	Y	-	-	0.0686	-	-	0.0864
	U	-	-	0.0470	-	-	0.0602
	V	-	-	0.0411	-	-	0.0596

TABLE 5. The comparison results of *WS_PSNR* and *SPSNR_I* for "CBP".

Sequences	Channel	WSPSNR (dB) Lanczos algorithm	WSPSNR (dB) SLAN algorithm	ΔWSPSNR (dB)	SPSNR_I (dB) Lanczos algorithm	SPSNR_I (dB) SLAN algorithm	ΔSPSNR_I (dB)
AerialCity	Y	45.2916	45.4121	0.1205	46.3913	46.5022	0.1109
	U	52.8348	52.8926	0.0578	53.3201	53.3746	0.0545
	V	52.2919	52.3535	0.0616	52.8398	52.8954	0.0556
DrivingInCity	Y	47.5881	47.6546	0.0665	48.4458	48.5106	0.0648
	U	56.0968	56.1554	0.0586	56.0462	56.1061	0.0599
	V	55.5596	55.6169	0.0573	55.5708	55.6286	0.0578
DrivingInCountry	Y	44.5020	44.5924	0.0904	45.6124	45.6972	0.0848
	U	56.6563	56.7168	0.0605	56.5829	56.6434	0.0605
	V	56.5932	56.6496	0.0564	56.5374	56.5955	0.0581
PoleVault_le	Y	46.0832	46.1560	0.0728	47.0497	47.2101	0.1604
	U	47.9245	47.9961	0.0716	49.1021	49.1705	0.0684
	V	49.5431	49.6125	0.0694	50.5059	50.5712	0.0653
Average	Y	-	-	0.0876	-	-	0.1052
	U	-	-	0.0621	-	-	0.0608
	V	-	-	0.0612	-	-	0.0592

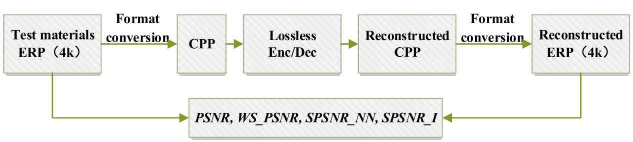


FIGURE 10. The test procedure with lossless compression coding.

ERP format, i.e., spherical panoramic video sources were all stored in the ERP format online. In terms of CPP format, the test procedure with lossless compression coding is shown in Figure 10. In this case, we ignore the loss caused by coding and only consider the loss brought by interpolation. We test the end-to-end performance, which is evaluated by *PSNR*[28], *WS_PSNR*[29], *SPSNR_NN*[30] and *SPSNR_I*[30], to indicate the performance of the proposed SLAN algorithm compared with the original Lanczos algorithm in the format conversion. The higher value of various end-to-end *PSNR* means the less loss in the interpolation, and thus further indicates a better algorithm.

The comparison results of the various end-to-end *PSNR* of Lanczos interpolation algorithm and the proposed SLAN interpolation algorithm for CPP format are shown in TABLE 2∼TABLE 3, where ΔX was calculated by

$$\Delta X = X_{SLAN} - X_{Lanczos}, \tag{23}$$

where X is *PSNR*, *SPSNR_NN*, *WS_PSNR* or *SPSNR_I*.

It can be seen that, compared with Lanczos interpolation algorithm, the proposed SLAN interpolation algorithm could achieve a *PSNR* rise up to 5.76dB and by 3.62dB on average for Y, up to 2.97dB and by 1.97dB on average for U as well as up to 2.98dB and by 1.91dB on average for V. The similar increase could be achieved in terms of *SPSNR_NN*, *WS_PSNR* and *SPSNR_I*. As expected, the proposed SLAN interpolation algorithm performed better in format conversion for CPP format.

Besides, for CBP format and ACP format, the end-to-end performances are also evaluated to illustrate that our proposed SLAN algorithm is applicable for all kinds of planar

TABLE 6. The comparison results of PSNR and SPSNR_NN for "ACP".

Sequences	Channel	PSNR (dB) Lanczos algorithm	PSNR (dB) SLAN algorithm	$\Delta PSNR$ (dB)	SPSNR_NN (dB) Lanczos algorithm	SPSNR_NN (dB) SLAN algorithm	$\Delta SPSNR\_NN$ (dB)
AerialCity	Y	46.2281	46.3969	0.1688	46.7831	47.0033	0.2202
	U	52.7816	52.8454	0.0638	53.4516	53.5338	0.0822
	V	52.0507	52.1131	0.0624	52.8401	52.9272	0.0871
DrivingInCity	Y	49.997	50.1312	0.1342	50.3826	50.5335	0.1509
	U	55.7276	55.788	0.0604	56.7789	56.8711	0.0922
	V	55.2589	55.3272	0.0683	56.2783	56.3682	0.0899
DrivingInCountry	Y	46.7698	46.9547	0.1849	46.9878	47.2187	0.2309
	U	57.0168	57.0965	0.0797	57.2878	57.3867	0.0989
	V	56.5771	56.6375	0.0604	57.2376	57.3321	0.0945
PoleVault_le	Y	47.6758	47.7695	0.0937	48.9211	49.0500	0.1289
	U	48.2539	48.3651	0.1112	48.9410	49.0607	0.1197
	V	49.811	49.8925	0.0815	50.5263	50.6243	0.0980
Average	Y	-	-	0.1454	-	-	0.1827
	U	-	-	0.0788	-	-	0.0983
	V	-	-	0.0681	-	-	0.0924

TABLE 7. The comparison results of *WS_PSNR* and *SPSNR_I* for "ACP".

Sequences	Channel	WSPSNR (dB) Lanczos algorithm	WSPSNR (dB) SLAN algorithm	$\Delta WSPSNR$ (dB)	SPSNR_I (dB) Lanczos algorithm	SPSNR_I (dB) SLAN algorithm	$\Delta SPSNR\_I$ (dB)
AerialCity	Y	46.7660	46.9906	0.2246	47.9188	48.1217	0.2029
	U	53.4385	53.5228	0.0843	53.8718	53.9414	0.0696
	V	52.8284	52.9176	0.0892	53.3596	53.4305	0.0709
DrivingInCity	Y	50.3494	50.5030	0.1536	51.0012	51.1356	0.1344
	U	56.7804	56.8732	0.0928	56.6737	56.7554	0.0817
	V	56.2848	56.3871	0.1023	56.2255	56.3053	0.0798
DrivingInCountry	Y	46.9651	47.2048	0.2397	48.1658	48.3870	0.2212
	U	57.2679	57.3698	0.1019	57.1429	57.2306	0.0877
	V	57.2194	57.3152	0.0958	57.0964	57.1804	0.0840
PoleVault_le	Y	48.8420	48.9739	0.1319	49.8522	49.9775	0.1253
	U	48.9092	49.0330	0.1238	50.0438	50.1539	0.1101
	V	50.5046	50.6059	0.1013	51.3921	51.4797	0.0876
Average	Y	-	-	0.1875	-	-	0.1710
	U	-	-	0.1007	-	-	0.0873
	V	-	-	0.0972	-	-	0.0806

projection formats. The comparison results of various end-to-end *PSNR* are shown in TABLE 4∼TABLE 7. Compared with the original Lanczos interpolation algorithm, the proposed SLAN interpolation algorithm could achieve a PSNR rise by 0.07dB, 0.05dB and 0.04dB for Y, U and V on average respectively for CBP format. As for ACP format, proposed SLAN interpolation algorithm could achieve a PSNR rise by 0.14dB, 0.08dB and 0.07dB for Y, U and V on average, respectively. The similar increase could be achieved in terms of *SPSNR_NN*, *WS_PSNR* and *SPSNR_I*.

From the above results, we can easily find that the larger the deformation in the planar perspective frames, the better the performance can SLAN achieve compared with Lanczos interpolation algorithm. The deformation in frames in CPP is the largest, so that proposed SLAN algorithm performs the best. As for CBP and ACP format, the improvement of the performance is limited because the deformation is limited. However, no matter the deformation is severe or not, as long as there is deformation in the planar perspective frames, the proposed SLAN interpolation algorithm is effective and will perform better compared with Lanczos interpolation algorithm. In fact, it has been proved that there is inevitable deformation in planar perspective frames, no matter in which projection formats [27], the proposed SLAN interpolation algorithm is always significant.

To further evaluate the performance of the proposed SLAN interpolation algorithm, block-*PSNR* was calculated by

$$PSNR_{m,n} = 10\log_{10}\left(\frac{MAX_I^2}{\frac{1}{W_BH_B}\sum_{i=(m-1)W_B}^{mW_B}\sum_{j=(n-1)H_B}^{nH_B}[I(i,j)-K(i,j)]^2}\right) = 10\log_{10}\left(\frac{MAX_I^2}{MSE_{m,n}}\right), \quad (24)$$

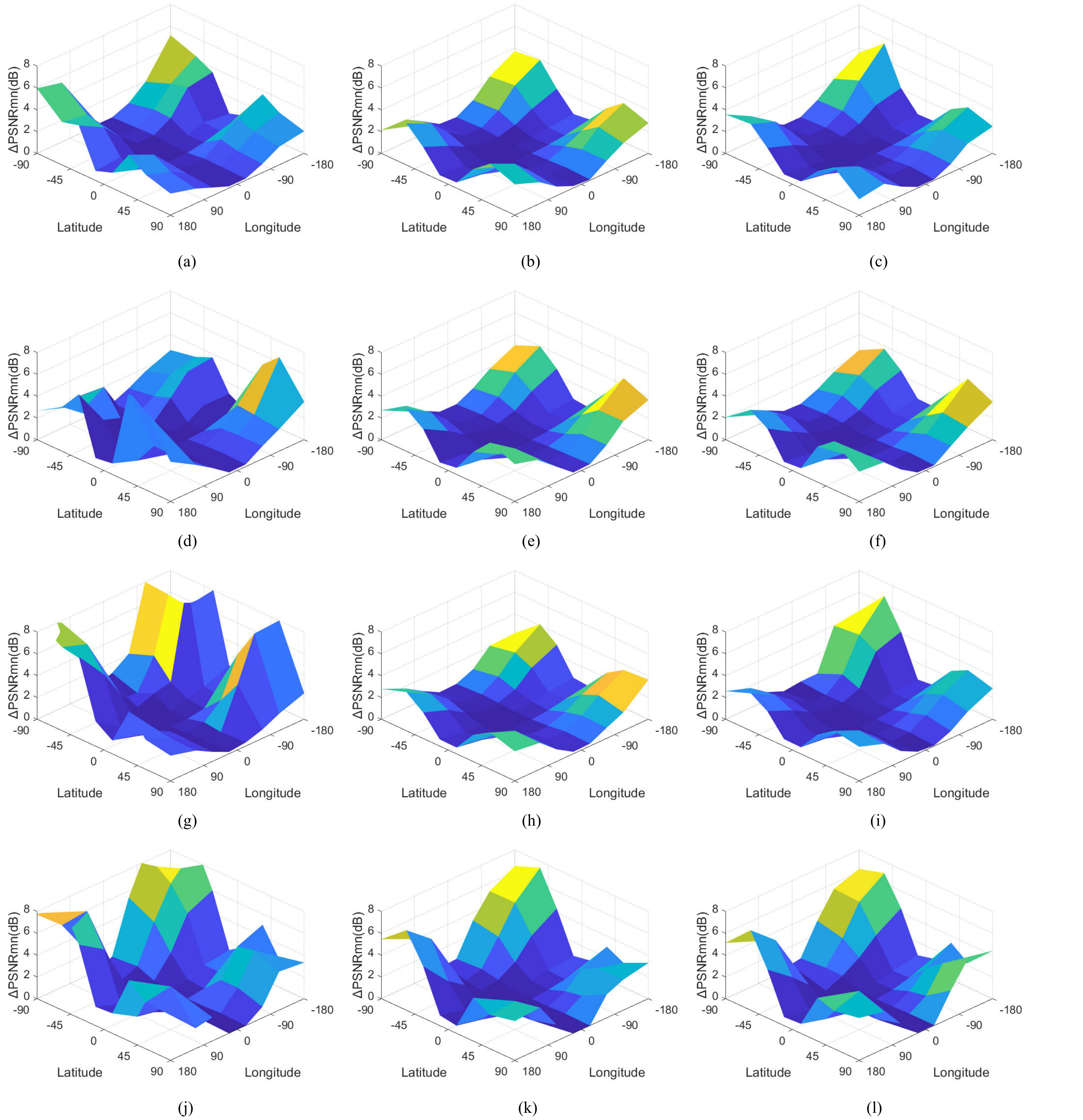


FIGURE 11. $\Delta PSNR_{m,n}$ **between SLAN and Lanczos. (a)** $\Delta PSNR_{m,n}$ **in** ***AerialCity*** **for Y. (b)** $\Delta PSNR_{m,n}$ **in** ***AerialCity*** **for U. (c)** $\Delta PSNR_{m,n}$ **in** ***AerialCity*** **for V. (d)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCity*** **for Y. (e)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCity*** **for U. (f)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCity*** **for V. (g)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCountry*** **for Y. (h)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCountry*** **for U. (i)** $\Delta PSNR_{m,n}$ **in** ***DrivingInCountry*** **for V. (j)** $\Delta PSNR_{m,n}$ **in** ***PoleVault_le*** **for Y. (k)** $\Delta PSNR_{m,n}$ **in** ***PoleVault_le*** **for U. (l)** $\Delta PSNR_{m,n}$ **in** ***PoleVault_le*** **for V.**

where $PSNR_{m,n}$ represents the $PSNR$ in the block which is located at mth row and nth column. MAX_I represents the max value of pixels (i.e., 255 for pixels with 8 bits in this paper). W_B and H_B represent the width and the height of the block. $I(i,j)$ and $K(i,j)$ represent the value of the pixel located at (i,j) in the reconstruct frame and the source frame respectively. The test procedure is shown in Figure 11. The differences of $PSNR_{m,n}$ between the proposed SLAN interpolation algorithm and Lanczos interpolation algorithm was calculated by

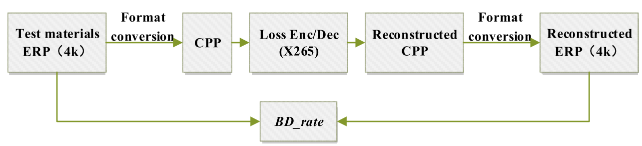
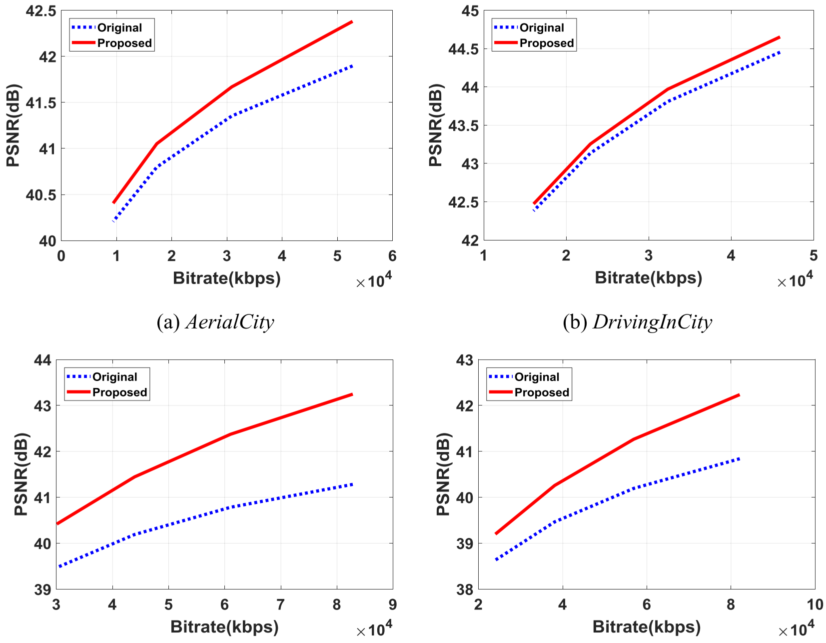


FIGURE 12. **The test procedure with loss compression coding.**

$$\Delta PSNR_{m,n} = PSNR_{m,n\_pro} - PSNR_{m,n\_ori}, \qquad (25)$$



(a) *AerialCity* (b) *DrivingInCity* (c) *DrivingInCountry* (d) *PoleVault_le*

FIGURE 13. RD curves of the original Lanczos algorithm and the proposed SLAN algorithm of four test sequences.

TABLE 8. Encoding environment.

Reference Software	X265_2.8
Sequences	*AerialCity* *DrivingInCity* *DrivingInCountry* *PoleVault_le*
Resolution	3840x1920
Bframes	0
Frames to be tested	300
Frame rate	30fps
Bit depth	8bit
Sampling Format	YUV420
Multithread	not allowed
QP	20,22,24,26
Hardware Platform	Intel (R) Core(TM) i7-6700K CPU @ 4.00GHz

TABLE 9. The ***BD_rate*** of four test sequences.

Sequences	*BD_rate*	Average *BD_rate*
AerialCity	-23.83%	-27.18%
DrivingInCity	-6.63%	
DrivingInCountry	-44.55%	
PoleVault_le	-33.70%	

where $\Delta PSNR_{m,n}$ represents the difference of $PSNR_{m,n}$, $PSNR_{m,n\_pro}$ represents $PSNR_{m,n}$ of the proposed SLAN interpolation algorithm, and $PSNR_{m,n\_ori}$ represents $PSNR_{m,n}$ using the Lanczos interpolation algorithm. Values of $\Delta PSNR_{m,n}$ in the first frame in sequences *AerialCity*, *DrivingInCity*, *DrivingInCountry* and *PoleVault_le* are shown in Figure 11.

It is true that proposed SLAN interpolation algorithm performed better especially in high latitude and high longitude areas that are deformed more severely in the whole frame. Again, it indicates that the larger the deformation is, the better the performance can be achieved. Our proposed SLAN algorithm consistently outperforms its competitor Lanczos by considering the spherical characteristics of panoramic videos.

B. LOSS COMPRESSION CODING

The encoding environment is shown in TABLE 8. Because strong deformation and edge effect are responsible for low efficiency in encoding videos in CPP format, we chose small QPs to ensure the quality of reconstructed videos. In terms of CPP format, the test procedure with loss compression coding is shown in Figure 12.

We test the *BD_rate* [31] which is shown in TABLE 9 to indicate the significance of the proposed SLAN interpolation algorithm for the compression in the processing chain of panoramic videos [32]. Compared with the original Lanczos interpolation algorithm, the proposed

SLAN interpolation algorithm could achieve a *BD_rate* decrease of 27.18% on average, with the highest value being 44.55%. RD curves of the original Lanczos algorithm and the proposed SLAN algorithm of four test sequences is shown in Figure 13.

TABLE 10. The comparison results of computational complexity.

Sequences	"CPP TO ERP" (s)		"CBP TO ERP" (s)		"ACP TO ERP" (s)	
	Lanczos algorithm	SLAN algorithm	Lanczos algorithm	SLAN algorithm	Lanczos algorithm	SLAN algorithm
AerialCity	336.638	471.957	442.883	629.123	253.963	402.530
DrivingInCity	295.902	416.817	447.800	631.025	293.007	483.340
DrivingInCountry	299.506	434.951	434.984	632.762	248.304	390.403
PoleVault_le	349.264	500.790	416.227	591.154	255.263	419.315
Average	320.328	456.129	435.474	621.106	262.634	423.897

C. COMPUTATIONAL COMPLEXITY

The computational complexity in this section is evaluated by the time consumed by a format conversion. The hardware platform is shown in TABLE 8. The comparison results of computational complexity of the original Lanczos interpolation algorithm and proposed SLAN interpolation algorithm in terms of "CPP to ERP", "CBP to ERP" and "ACP to ERP" are shown in TABLE 10.

The consuming time of the proposed SLAN algorithm is slightly higher than that of the original algorithm because of the calculation of mass trigonometric functions though we have adopted SSE acceleration command. The sphere-to-plane projection or projection format conversions in current 3DOF multimedia signal processing pipeline are usually performed in an offline manner where computational complexity is not the major concern. However, we will still consider its complexity in future work.

V. CONCLUSION

Panoramic videos are spherical in nature which brings great challenges to their storage, encoding, transmission and display. Sphere-to-plane projection is an effective solution at present, and many sphere-to-plane projection formats have been proposed, such as ERP, CPP, CBP and ACP. Moreover, considering that different planar projection formats may adapt to different applications, we often need to convert one projection format to another. In both cases, the interpolation algorithm is crucial. By fully considering the spherical characteristics of panoramic videos, the SLAN interpolation algorithm is proposed in this paper. Experimental results have demonstrated the advanced performance of the proposed algorithm both with lossless compression coding and with loss compression coding in terms of the increase in various end-to-end *PSNR* and the decrease in *BD_rate*. The proposed SLAN algorithm is meaningful for all planar projection formats because of their inevitable deformation. Besides, the computational complexity will be considered as future work.

ACKNOWLEDGMENT

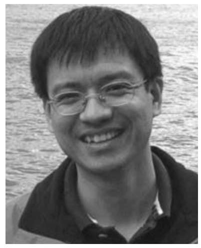
The authors would like to thank the experts of HUAWEI Technologies Company, Ltd. for their contributions and fruitful discussions. In particular, the authors would also like to thank P. Li, Y. Song, and H. Fang.

REFERENCES

[1] J. Meehan, *Panoramic Photography*. New York, NY, USA: Amphoto, 1990.
[2] H. Yong, J. Huang, W. Xiang, X. Hua, and L. Zhang, "Panoramic background image generation for PTZ cameras," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3162–3176, Jul. 2019.
[3] J. Li, Z. Wang, S. Lai, Y. Zhai, and M. Zhang, "Parallax-tolerant image stitching based on robust elastic warping," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1672–1687, Jul. 2018.
[4] N. Li, Y. Xu, and C. Wang, "Quasi-homography warps in image stitching," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1365–1375, Jun. 2018.
[5] T. Pakkanen, J. Hakulinen, T. Jokela, I. Rakkolainen, J. Kangas, P. Piippo, R. Raisamo, and M. Salmimaa, "Interaction with WebVR 360° video player: Comparing three interaction paradigms," in *Proc. IEEE Virtual Reality (VR)*, Mar. 2017, pp. 279–280.
[6] U. Neumann, T. Pintaric, and A. Rizzo, "Immersive panoramic video," in *Proc. 8th ACM Int. Conf. Multimedia*, 2000.2000, pp. 493–494.
[7] W.-K. Tang, T.-T. Wong, and P.-A. Heng, "A system for real-time panorama generation and display in Tele-Immersive applications," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 280–292, Apr. 2005.
[8] J. P. Snyder, *Flattening the Earth : Two Thousand Years of Map Projections*. Chicago, IL, USA: Univ. Chicago Press, 1993.
[9] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen, "Tiling in interactive panoramic video: Approaches and evaluation," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1819–1831, Sep. 2016.
[10] C.-W. Fu, L. Wan, T.-T. Wong, and C.-S. Leung, "The rhombic dodecahedron map: An efficient scheme for encoding panoramic video," *IEEE Trans. Multimedia*, vol. 11, no. 4, pp. 634–644, Jun. 2009.
[11] C. Zhu, K. Huang, and G. Li, "An innovative saliency guided ROI selection model for panoramic images compression," in *Proc. Data Compress. Conf.*, Mar. 2018, p. 436.
[12] A. Ohashi, F. Yamano, G. Masuyama, K. Umeda, D. Fukuda, K. Irie, S. Kaneko, J. Murayama, and Y. Uchida, "Fisheye stereo camera using equirectangular images," in *Proc. 17th Int. Conf.*, 2016, pp. 284–289.
[13] K.-T. Ng, S.-C. Chan, and H.-Y. Shum, "Data compression and transmission aspects of panoramic videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 82–95, Jan. 2005.
[14] M. Coban, G. Van der Auwera, and M. Karczewicz, *AHG8: Adjusted Cubemap Projection for 360-Degree Video*, document JVET-F0025, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Apr. 2017.
[15] V. Zakharchenko, K. P. Choi, and J. H. Park, "Quality metric for spherical panoramic video," *Proc. SPIE Opt. Photon. Inf.*, vol. 9970, Sep. 2016, Art. no. 99700C.
[16] H.-C. Lin, C.-Y. Li, and J.-L. Lin,*AHG8: An Efficient Compact Layout for Octahedron Format*, document JVET-D0142, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Oct. 2016.
[17] V. Zakharchenko, E. Alshina, and K. P. Choi, *AHG8: Icosahedral Projection for 360-Degree Video Content*, document JVET-D0028, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Oct. 016.
[18] C. Zhang, Y. Lu, and J. Li, *AHG8: Segmented Sphere Projection (SSP) For 360-Degree Video Content*, document JVET-D0030, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Oct. 2016.
[19] A. Abbas and D. Newman, *AHG8: Rotated Sphere Projection for 360 Video*, document JVET-F0036, Hobart, AU, USA, Mar. 2017.

[20] Y.-H. Lee, J.-L. Lin, and S.-K. Chang, *CE13: Modified Cubemap Projection*, document JVET-J0019 JVET-K0131, Ljubliana, SI, USA, Jul. 2018.
[21] W. Chengjia, Z. Haiwu, and S. Xiwu, "Octagonal mapping scheme for panoramic video encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2402–2406, Sep. 2018.
[22] B. Choi, Y.-K. Wang, and M. M. Hannuksela, *Study of ISO/IEC DIS 23000-20 Omnidirectional Media Format*, document ISO/IEC JTC1/SC29/WG11, N16950, Jul. 2017.
[23] S. E. Chen, "QuickTime VR: An image-based approach to virtual environment navigation," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn.*, 1995, pp. 29–38.
[24] B. N. Madhukar and R. Narendra, "Lanczos resampling for the digital processing of remotely sensed images," in *Proc. Int. Conf. VLSI, Commun., Adv. Devices, Signals Syst. Netw. (VCASAN)*, 2013, pp. 403–411.
[25] K. Turkowski, "Filters for common resampling tasks," in *Graphics Gems*. New York, NY, USA: Academic, 1990.
[26] Y. Ye, E. Alshina, and J. Boyce, *Algorithm Descriptions of Projection Format Conversion and Video Quality Metrics in 360Lib Version 4*, document ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G1003, Joint Video Exploration Team, Jul. 2017.
[27] D. Zorin and A. H. Barr, "Correction of geometric perceptual distortions in pictures," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn.*, 1995, pp. 257–264.
[28] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T SG16 Q, VCEG-M33, vol. 6, Apr. 2001.
[29] Y. Sun, A. Lu, and L. Yu, *AHG8: WS-PSNR for 360 Video Objective Quality Evaluation*, document JVET-D0040ITU-T SG16 WP3 ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Chengdu, China, Oct. 2016.
[30] M. Yu, H. Lakshman, and B. Girod, "A framework to evaluate omnidirectional video coding schemes," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, Sep. 2015, pp. 31–36.
[31] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves ITU-T Study Group, Video Coding Expert Group (VCEG)*, document VCEG-M33, vol. 16, Austin, TX, USA, Apr. 2001.
[32] J. Boyce, E. Alshina, A. Abbas, *JVET Common Test Conditions and Evaluation Procedures for 360° Video*, document JVET-F1030 ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Joint Video Exploration Team, Hobart, AU, USA, Mar. 2017.

SAIPING ZHANG (Student Member, IEEE) received the B.E. degree from Xidian University, Xi'an, China, in 2016, where she is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Services Networks. Her research interests include video coding and multimedia communication.



FUZHENG YANG (Member, IEEE) received the B.E. degree in telecommunication engineering, the M.E. degree, and the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2000, 2003, and 2005, respectively. From 2006 to 2007, he served as a Visiting Scholar and a Postdoctoral Researcher with the Department of Electronic Engineering, Queen Mary University of London. He became a Lecturer and an Associate Professor at Xidian University, in 2005 and 2006, respectively. He has been a Professor of communications engineering with Xidian University, since 2012. He is also an Adjunct Professor with the School of Engineering, RMIT University. His research interests include video quality assessment, video coding, and multimedia communication.

SHUAI WAN (Member, IEEE) received the B.E. degree in telecommunication engineering and the M.E. degree in communication and information system from Xidian University, Xi'an, China, in 2001 and 2004, respectively, and the Ph.D. degree in electronic engineering from Queen Mary University of London, in 2007. She is currently a Professor with Northwestern Polytechnical University, Xi'an. Her research interests include scalable/multiview video coding, video quality assessment, and hyperspectral image compression.

PEIYUN DI received the M.E. degree from the Electronic Engineering School, Xidian University, Xi'an, China, in 2004 and 2007. In 2007, she joined the Media Laboratory, Huawei Technologies Company Ltd., Shenzhen, China, where she is currently a Senior Engineer. Her current research interests include video quality assessment, video communication, and video coding.