Signal Processing: Image Communication 75 (2019) 76–88

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Consistent video projection on curved displays

Yangxintong Lyu [a,*], Shao-Ping Lu [b,a,**], Quentin Bolsee [a], Adrian Munteanu [a]

[a] Department of Electronics and Informatics, Vrije Universiteit Brussel, Pleinaan 2, B-1050 Brussel, Belgium
[b] TKLNDST, CS, Nankai University, Tianjin, China

ARTICLE INFO

Keywords:
Curved displays
Immersion
Video deformation

ABSTRACT

The production of curved displays has substantially increased in the past years. This new type of displays can enhance the feeling of immersion by surrounding the viewers with content. In this paper, we propose a content-aware and depth-aware video deformation method for projection of rectangular content on curved displays. Novel deformation and grid-based mapping methods are proposed to prevent over-preservation of foreground content for static and dynamic content and to enforce temporal coherence. The objective evaluation of the results shows that our approach efficiently and consistently preserves both static and dynamic content. A thorough subjective evaluation is also performed using a real-world curved screen as well as a virtual reality setup, showcasing our method against existing methods. Experimental results indicate that our method systematically outperforms the existing state-of-the-art, providing minimal content distortion and high temporal coherence between consecutive frames.

1. Introduction

Immersion, in the context of virtual reality, defines how convincing a viewing experience is for the user. This term is also used to generally describe a realistic viewing experience in the real world. A lot of research has been carried out in the recent past in order to substantially improve the immersive visual perception by developing new types of displays and means to consume the content. Ultra-High-Definition (UHD) displays were developed to improve the rendered spatio-temporal resolution of the video content. 3D displays based on the stereopsis of binocular vision have been devised to convey the perception of depth to viewers. Light field displays have been also explored as means of providing auto-stereoscopy and a genuine 3D experience. Last but not least, curved displays have been proposed to provide an immersive experience by surrounding the viewer with visual content; these displays can provide a realistic feeling when they cover a large part of a user's visual field. Cinemas have already tried introducing curved projection surfaces to improve the immersion when displaying a movie. However, with a single rectangular input source, the curvature of the projection surface in a cinema must be kept small to prevent extreme distortion effects. Curved screen TVs and curved smart-phones do not suffer from this limitation, and can be designed with somewhat larger curvature angles.

Projecting conventional rectangular video on curved displays is a significant challenge, in particular for large curvatures. Related work includes image and video retargeting methods of which an overview will be given in the next section. As detailed next, existing state-of-the-art methods have difficulties in simultaneously keeping geometric consistency for foreground and background objects and also in providing a temporally coherent video, free of visual artifacts at object boundaries. Properly addressing these problems and providing solutions to them is the main focus of this work. In this context, a novel grid-based mapping method for video retargeting onto a curved surface is proposed offering minimal content distortion and high temporal coherence between consecutive frames.

In summary, the main contributions of this paper are as follows:

- We propose a deformation scaling method which limits the over-preservation of foreground content and the distortion of the background.
- We introduce a grid-based mapping method which tackles the issue of flickering and waving artifacts between consecutive frames.
- We introduce an objective metric which helps in evaluating video deformation results based on the differences with respect to the input source, as well as to objectively evaluate the temporal coherence.
- We perform a thorough subjective evaluation of our proposed method against the state of the art, by employing a real-world curved screen display as well as a virtual reality setting which allows for simulating projection on curved displays of arbitrary curvatures.

* Corresponding author.
** Corresponding author at: TKLNDST, CS, Nankai University, Tianjin, China.
E-mail addresses: yangxintonglyu@icloud.com (Y. Lyu), slu@nankai.edu.cn (S.-P. Lu), qbolsee@etrovub.be (Q. Bolsee), acmuntea@etrovub.be (A. Munteanu).

https://doi.org/10.1016/j.image.2019.03.006
Received 6 October 2018; Received in revised form 5 March 2019; Accepted 11 March 2019
Available online 22 March 2019

The paper is structured as follows. In the next section, a state-of-the-art overview of existing retargeting methods is provided. Our proposed method is presented in detail in Section 3, while the experimental results and the comparison against the state-of-the-art methods are reported in Section 4. Finally, Section 5 draws the conclusions of our work.

2. Related work

2.1. Image retargeting

The simplest method to map an input rectangular image to devices with different shape surfaces/aspect ratios is to crop the input image directly. However, this will always result in a loss of content. Uniform scaling Ling et al. [1] can also be used in this context, but this will visibly introduce distortion of important content. Numerous content-aware retargeting methods have been developed in the recent years to enable the preservation of important content. Most of the image retargeting methods can be generally divided into two main categories: pixel-level and mesh-level methods.

Pixel-level methods are mainly based on seam carving proposed by Avidan et al. in [2]. These methods follow common steps to solve the deformation problem, that is: *(i)* they define an energy function to generate the importance/salient map, and *(ii)* they remove one-pixel-wide column/row with the lowest energy and proceed recursively until the deformed image reaches the target size. Battiato et al. [3] define the energy function by the properties of Gradient Vector Flow, incorporating thus gradient information in the deformation process. Qi et al. [4] segment each seam in order to produce irregularly-shaped output images. Shen et al. [5] point out that it is not easy to determine the important and salient objects that must be exempted from distortion. Thus, they use depth maps captured by depth cameras to determine foreground and to impose the protection of foreground content.

The problem of loss of content is still persistent in the seam-carving family of methods, which process pixels independently without considering the structure of content across neighboring pixels. An alternative solution for image retargeting is to overlay a mesh and to warp the image at mesh level, which means processing the pixels contained within rectangular or triangular shaped grids. While the seam-carving based methods only work with planar input and planar output, some of the mesh level methods can also produce a spherical or cylindrical output.

For the planar input/planar output image retargeting problem, Zhang et al. [7] propose a method which preserves the shape of important objects by supplementing the grid points by extra control points that operate both locally and globally. Chang et al. [8] emphasize the line structure preservation by formulating the problem in the Hough space. Concerning planar to spherical/cylindrical transformation problems, Carroll et al. [9] propose a method which projects a 3D scene onto a flat, wide angle image with user-specified constraints. This method preserves the highly subjective perception on the flat image after transformation.

Mansfield et al. [10] and Lee et al. [11] use depth maps to decompose the input image into different layers prior to solving the retargeting problem.

Raskar et al. [12] and Sajadi et al. [13] emphasize geometric calibration and alignment terms which solve the projection problem on curved screens for a multi-projector system. However, these last two methods do not consider the distortion of content occurring in the deformation process.

Lu et al. [6] proposed an efficient grid-based, depth-aware image adaptation method for image projection on curved surfaces. This method mainly preserves the shape and the size of foreground-content grids at the cost of background-content grids in order to fit the top and bottom boundaries of curved screens. However, when the curvature of the curved screen is large, both shape and size preservation of foreground-content grids in Lu et al. [6] easily lead to a visible distortion of background content (which can also be interpreted as over-preservation of foreground content), as illustrated in the example of Fig. 1. Moreover, when employed on video content, deforming individual frames with this method can easily result in temporal incoherence, characterized by flickering or waving artifacts at object boundaries caused by the motion of camera and/or objects in the scene.

2.2. Video retargeting

The common practice to solve the video retargeting problem is to extend image retargeting methods towards video by embedding the motion information as part of the image saliency map and by enforcing temporal coherence between consecutive frames. Based on Avidan et al. [2], Rubinstein et al. [14] extend the 2-D seam to 3-D time-space seam to enforce temporal coherence. Wang et al. [15] emphasize temporal consistency of the camera and object motion by aligning consecutive frames and detecting moving areas. Zhang et al. [16] overcome the difficulty of computing spatial domain importance maps and solve the video retargeting problem in the compressed domain. However, the retargeting problems solved by this prior works [14–16] are limited to planar-to-planar applications. Thus, such methods cannot be applied directly to our planar to curved projection problem. Unlike [14–16], Wei et al. [17] correct a fish-eye video from spherical domain to planar domain. However, our starting domain is planar and requires a completely different methodology.

All the above-cited video retargeting methods are guided by importance/saliency maps which can be computed in many different ways. However, there is no guarantee or evidence that any specific importance/saliency map computation method will always provide stable and visually pleasing results when applied on a wide variety of video sequences. Because of this, instead of computing importance maps, more stable results are obtained by relying on depth information instead based on which content is roughly classified into foreground and background; details are given next.

3. Proposed method

3.1. Problem description

The pipeline of our method is presented in Fig. 2. We follow the same projection model as that in Lu et al. [6] by projecting a deformed video from planar domain to cylindrical domain in order to fit the curvature of the curved screen. The projection is realized from the center of curvature of the screen (called **viewpoint** in the remainder of the paper), as illustrated in Fig. 3(a). We denote pixels in the original frame by $P(x, y)$. Pixels in the deformed frame and the output frame on the curved screen are denoted by $\hat{P}(\hat{x}, \hat{y})$ and $P_s(x_s, y_s)$, respectively.

Our proposed method is content-aware and depth-aware. The deformed result is a mapping that optimally deforms a given input frame so that it can be projected onto the image plane. The top and bottom boundaries of each deformed frame are symmetrical curves that are guided by the curvature boundaries of the curved screen. Meanwhile, left and right boundaries of each deformed frame should be kept vertical. In order to project the video onto the curved screen with high temporal coherence and minimal content distortion, we add four constraints on the deformation process:

- The shape of content in each frame should be preserved as much as possible.
- Both foreground and background content should be preserved in a balanced way, instead of focusing only on the foreground content.
- The boundaries (left/right/top/bottom) of each deformed frame should be constrained to fit the boundaries of the curved screen.

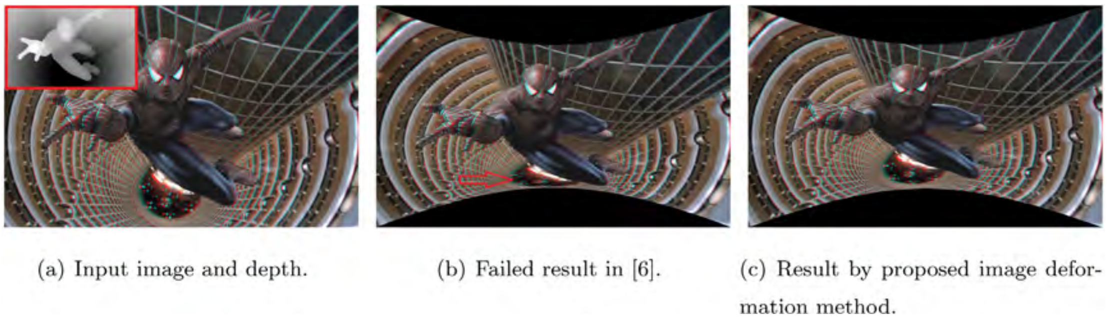

(a) Input image and depth.
(b) Failed result in [6].
(c) Result by proposed image deformation method.

Fig. 1. (b) is the failure case in Lu et al. [6]. (c) is the image deformation result of proposed method.

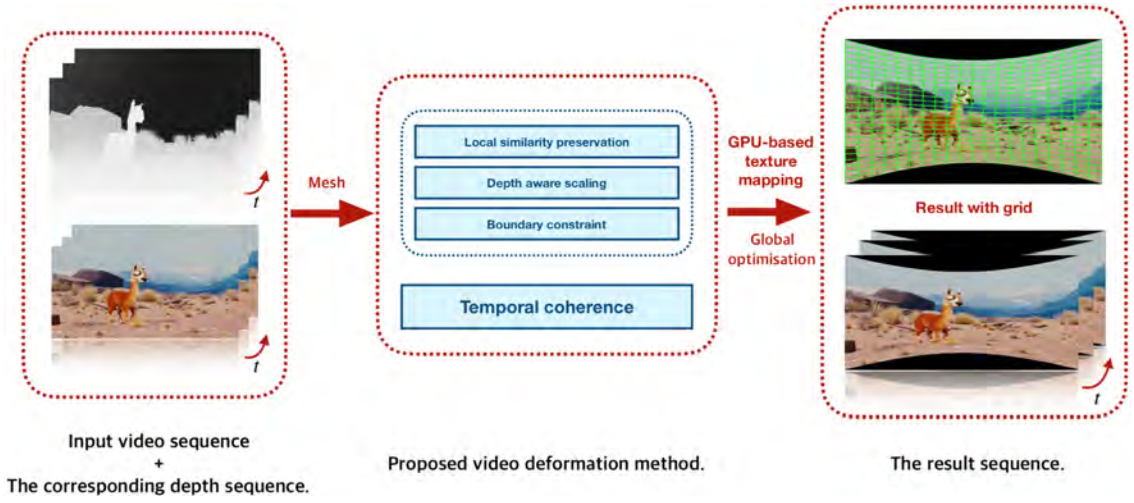


Fig. 2. The framework of our proposed method.

- Temporal coherence should be imposed between consecutive frames when motion is detected.

Each frame is covered by a quad mesh. We design an energy function that imposes these constraints on the coordinates of grid vertices. Determining the appropriate deformation process that meets these constraints is thus formulated and solved as a global optimization problem of which the details are given next.

3.2. Local similarity preservation

Our shape preservation energy E_S is defined to preserve the content shape within each grid. Intuitively, angles should be locally preserved after deformation, hinting towards a *conformal mapping*. We choose to apply a *similarity transformation*, which is a special case of conformal mapping. Zhang et al. [7] prove that the shape preservation energy can be associated to a linear least square minimization when such a similarity transformation is applied for the grid q, following:

$$E_S = \sum_q \left\| (A_q(A_q^T A_q)^{-1} A_q^T - I)\hat{B}_q \right\|^2, \tag{1}$$

where I is the 8×8 identity matrix and

$$A_q = \begin{bmatrix} x_a & -y_a & 1 & 0 \\ y_a & x_a & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_d & -y_d & 1 & 0 \\ y_d & x_d & 0 & 1 \end{bmatrix}, \quad \hat{B}_q = \begin{bmatrix} \hat{x}_a \\ \hat{y}_a \\ \vdots \\ \hat{x}_d \\ \hat{y}_d \end{bmatrix}. \tag{2}$$

Each frame is covered by a quad mesh $M = \{F, V\}$, where $F = \{q_m, 1 \leqslant m \leqslant N_q\}$ is the set of rectangular grid elements which uniformly segment the frame, and $V = \{v_k = (x_k, y_k), 1 \leqslant k \leqslant N_v\}$ is the set of grid vertices. Each grid element q, is defined by its four vertices v_a, v_b, v_c and v_d. In this paper, we use v_a for the top-left vertex, and v_b, v_c, v_d, respectively, denote the other three vertices in clockwise order.

3.3. Depth aware scaling

The depth constraint used in Lu et al. [6] emphasizes too much on the preservation of the shape and size of foreground content. When the curvature angle of the screen is large, the distortion of background content nearby foreground content is easily noticed, especially in the middle part of the frame. As shown in the example of Fig. 1(b), the circle next to Spider-Man's right leg is heavily distorted because of the over-preservation of foreground content.

Furthermore, for video deformation, if the foreground objects have motion or the camera moves, the distortion in each frame caused by these movements and over-preservation of foreground content will become easily noticeable when displaying the video sequence. This causes flickering and waving artifacts to appear at object boundaries which are very disturbing.

In order to tackle the over-preservation of foreground content, we firstly introduce a deformation impact factor $f_\theta(x)$ which is determined by the X-axis coordinate. This factor indicates how much a column on the curved screen has been deformed in the vertical direction for each given x on the curved screen. The closer the x is to $\frac{w}{2}$ (the center of a frame), the more deformation should be put on both foreground and background content, rather than only background content. Besides x, the deformation factor $f_\theta(x)$ is dependent on the curvature angle θ of the curved surface. When θ gets larger, less pixels are valid when projected from the image plane to the curved surface. For a specific curved display, θ is fixed. Thus, we define our deformation impact factor as follows:

$$f_\theta(x) = \frac{h_\theta(x)}{h}, \tag{3}$$

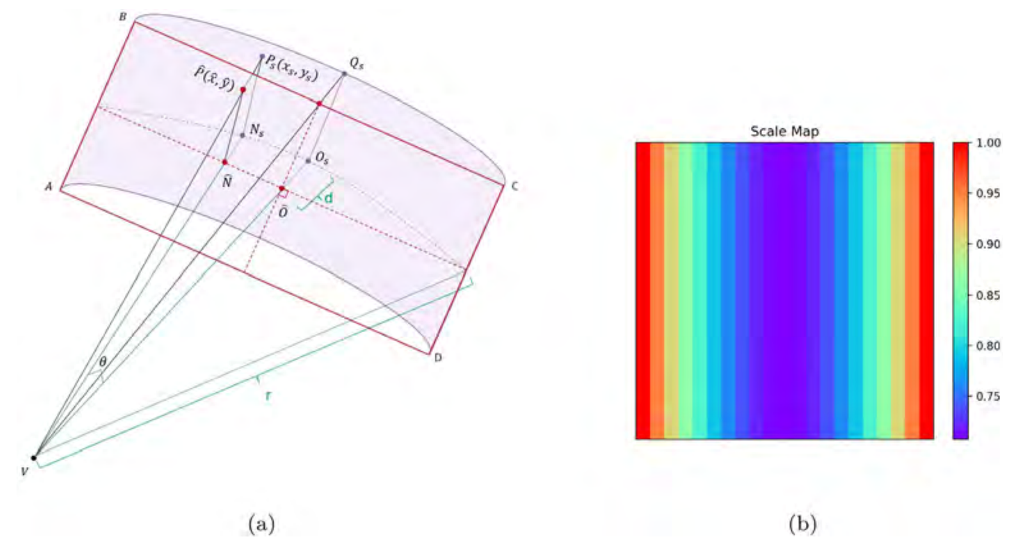


Fig. 3. (a) shows geometrical model between the viewpoint V, the image plane (Plane $ABCD$) and the curved output screen (Cylindrical surface $ABCD$) Lu et al. [6]. (b) shows deformation scale map with a curvature angle $\theta = 90°$.

where $h_\theta(x)$ is the number of valid pixels whose $X = x$ in the deformed image and h is the height of input image. Fig. 3(b) shows the deformation scaling factor when $\theta = 90°$. From Fig. 3(b), the values of $h_\theta(0)$ and $h_\theta(w)$ are 1, and the minimal value is reached when $x = \frac{w}{2}$. The term $h_\theta(x)$ depends on the boundary functions we use (details in Section 3.4) which follows $h_\theta(x) = |y_B(x) - y_T(x)|$, where $y_B(x)$ and $y_T(x)$ are coordinates on Y-axis on the top and bottom boundaries forming curves C_T and C_B respectively.

Moreover, in order to avoid a stretching effect near left/right boundaries, we propose to keep the ratio of grid width and height to be the same before and after deformation.

Based on these ideas, our depth constraint is introduced:

$$\begin{aligned} E_F = \sum_q \gamma(q) \big[& (d_x(\hat{v}_a, \hat{v}_b) - d_x(v_a, v_b))^2 \\ & + s\ f_\theta(x_b)(d_y(\hat{v}_c, \hat{v}_b) - d_y(v_c, v_b))^2 \\ & + (d_x(\hat{v}_a, \hat{v}_d) - d_x(v_a, v_d))^2 \\ & + s\ f_\theta(x_d)(d_y(\hat{v}_c, \hat{v}_d) - d_y(v_c, v_d))^2 \big], \end{aligned} \tag{4}$$

where s is the aspect ratio of the grid (width divided by height). $d_x(\cdot,\cdot)$ and $d_y(\cdot,\cdot)$ are distances in the X-axis and Y-axis, respectively. $\gamma(q)$ is a sigmoid-like function introduced in Lu et al. [6], which follows:

$$\gamma(q) = \frac{2}{e^{-\frac{\kappa(q)}{40}} + 1} - 1, \tag{5}$$

where $\kappa(q)$ is the mean of depth value of all the pixels in q.

3.4. Boundary constraints

In order to project the deformed video perfectly onto the curved surface, we also need additional constraints which restrict the vertices on the mesh boundaries to be mapped to the corresponding boundaries of the curved screen. For the left and right boundaries, they should be vertically preserved, leading to the energy term:

$$E_V = \sum_{v_i \in V_L} \hat{x}_i^2 + \sum_{v_i \in V_R} (\hat{x}_i - w)^2, \tag{6}$$

where $v_i = (x_i, y_i)$ is a boundary vertex and w is the width of input image. V_L and V_R are the corresponding vertex sets of left and right boundaries, respectively.

For the top and bottom boundaries, the boundary preservation energy is given by:

$$E_H = \sum_{v_i \in V_T} (\hat{x}_i - \tilde{x}_i)^2 + (\hat{y}_i - \tilde{y}_i)^2 + \sum_{v_i \in V_B} (\hat{x}_i - \tilde{x}_i)^2 + (\hat{y}_i - \tilde{y}_i)^2, \tag{7}$$

where $\tilde{v}_i = (\tilde{x}_i, \tilde{y}_i)$ is a reference point on curve C_T or C_B such that $\tilde{x}_i = x_i$, i.e. we keep the same horizontal sampling as the original grid. As stated before, C_T and C_B are functions (or curves) that describe the top and bottom boundaries, respectively. The coordinates of pixels on the top and bottom boundaries should follow:

$$\begin{cases} C_T : (r-d)^2 + (\hat{x} - \frac{w}{2})^2 - r^2(1 - \frac{2\hat{y}}{h})^2 = 0 \\ C_B : (r-d)^2 + (\hat{x} - \frac{w}{2})^2 - r^2(1 + \frac{2\hat{y}}{h})^2 = 0\,. \end{cases} \tag{8}$$

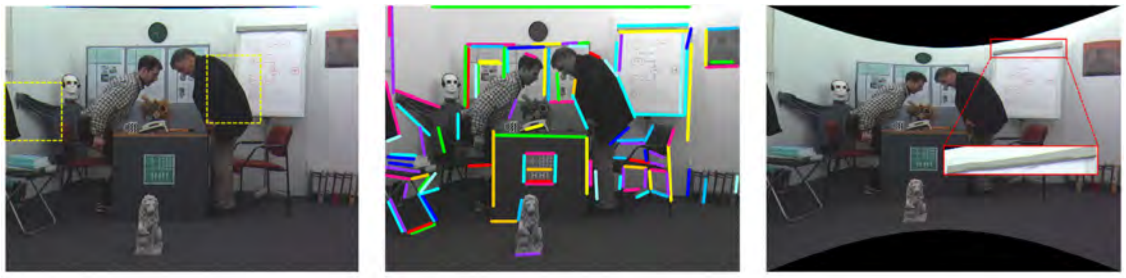
We do not consider the straight-line preservation constraint in our method, since that:

1. Lines detected by a line detector without semantic information as pre-condition are not always the straight lines which need to be preserved. For example, the line detector of [18] used in Lu et al. [6] mainly detects lines based on the image gradient. Within the yellow dash rectangular regions of Fig. 4(a), the lines detected on the boundaries of the model's sleeves and the back of the man on the right side in Fig. 4(b) should not necessarily be preserved. Moreover, detecting real lines with semantic information is not our main research target.
2. The preservation of lines across grids for the curved screens by simply keeping the skew angle the same before and after deformation easily results in visible segments. In Lu et al. [6], we find that for horizontal and vertical lines, the preservation works well, while lines that have a skew angle different from 0° or 90° are segmented by different grids and these segments become easily visible by observers after deformation. In Fig. 4(c), the top boundary of the white board is affected by segmentation.
3. Time cost needs to be taken into consideration. Some complex line preservation methods like Chang et al. [8] could preserve lines well at the cost of additional processing time. For example, the method proposed by Chang et al. [8] takes 2–40s per frame for straight-line preservation, which is significantly higher than our whole pipeline.

3.5. Temporal coherence

When the scene changes suddenly, the flickering is not easily noticed [15]. Thus, our temporal coherence assumes that all the frames of one sequence are from one scene.

If we impose temporal coherence on moving objects only, the flickering effect will propagate to the static content nearby. Thus, we take



(a) Input image. (b) Line detection result of the input image. (c) Unexpected line segments.

Fig. 4. (a) is the input image (with 2 yellow dash rectangular highlight regions). (b) shows the line detection result of the line segment detector [18]. (c) shows the unexpected wave effect of previous straight-line preservation Lu et al. [6].

both dynamic and static content into consideration. When an object moves, it can appear or disappear from one frame to the next, or move from one coordinate to another. Detecting and tracking individual dynamic objects among frames then keeping them temporally stable would be an ideal approach. But accurate object detection and tracking involve multi-label segmentation and pose estimation methods, which are challenging tasks requiring more advanced approaches [19–21]. However, because our method is grid-based and already preserves the shape of content based on the depth map, we do not need accurate segmentation of moving objects.

By taking both dynamic and static content into consideration, we propose a grid-based mapping approach for our temporal coherence constraint. Wang et al. [15] apply a feature-based method to evaluate the camera motion between consecutive frames for the purpose of alignment. But this feature-based method does not perform well in our grid mapping. This is due to the fact that the features detected by [22–24] mainly take color variation into consideration and they are sensitive to texturing. This leads to failure when matching texture-sparse content, like the sky, where very few feature points can be detected. Moreover, in texture-dense regions like vegetation, too many feature points are detected. That is, if we only base the correspondence between grids on the number of matched feature points, it will lead to mis-matched results.

Thus, we consider the luminance, contrast and structure of each grid. $SSIM$ [25] is a method for measuring the similarity between two images and is widely used for quality assessment with ground truth. In our work, we use this metric to evaluate the similarity between two grids. We calculate the $SSIM$ [25] value between corresponding neighboring grids at the previous frame. For a quad mesh at time t, let $M_t = \{F_t, V_t\}$, where $F_t = \{q_{t,m}, 1 \leqslant m \leqslant N_{q_t}\}$ is the set of rectangular grids which cover the input image, and $V_t = \{v_{t,k} = (x_{t,k}, y_{t,k}), 1 \leqslant k \leqslant N_{v_t}\}$ is the set of vertices. The grid mapping rule follows:

$$map(t+1,j) = \arg\max_{m\in\mathcal{N}(q_{t,j})} SSIM(q_{t+1,j}, m), \tag{9}$$

where $SSIM(a,b)$ is the $SSIM$ value between grids a and b. The 8-connected neighborhood grids of q_j and itself are denoted $\mathcal{N}(q_{t,j})$ (see Fig. 5). The result, $map(t+1,j)$, is the grid index at time t which matches best with grid $q_{t+1,j}$. After getting the mapping result, the offset (u and v, see Fig. 6) between matched grids from planar domain can be transformed to the cylindrical domain ($\hat{u}$ and $\hat{v}$, see Fig. 6), $\hat{u}$ and $\hat{v}$ in horizontal and vertical direction, respectively.

For temporal coherence, we define an energy $E_{t,T}$ for each mesh at time $t+1$, which follows:

$$E_{t,T} = \sum_{q_{t+1,j}\in F_{t+1}} \left\| \hat{V}'(q_{t+1,j}, q_{t,map(t+1,j)}) - \hat{V}_{t+1,j} \right\|^2, \tag{10}$$

where $q_{t+1,j}$ is the jth grid of mesh at time $t+1$. $V_{t,j}$ is the vertices set of jth grid on mesh F_t. $\hat{V}'(q_{t+1,j}, q_{t,map(t+1,j)})$ is a matrix that contains the transformed vertices from grid $q_{t,map(t+1,j)}$ to $q_{t+1,j}$ in cylindrical domain (see Fig. 6):

$$\hat{V}'(q_{t+1,j}, q_{t,map(t+1,j)}) = \left[\Phi(\hat{u},\hat{v},\hat{v}_{t,j,a})^T, \Phi(\hat{u},\hat{v},\hat{v}_{t,j,b})^T, \Phi(\hat{u},\hat{v},\hat{v}_{t,j,c})^T, \Phi(\hat{u},\hat{v},\hat{v}_{t,j,d})^T\right]^T, \tag{11}$$

where $\hat{v}_{t,j,(\cdot)}$ is the corresponding vertex of the jth grid at frame t, and $P_2 = \Phi(u, v, P_1)$ is a coordinate-transform function from P_1 to P_2 with cylindrical offset $\hat{u}$ and $\hat{v}$ ($\hat{u}$ and $\hat{v}$ can be computed from the grid mapping results). $\Phi(\hat{u},\hat{v},P_1)$ follows:

$$\Phi(\hat{u},\hat{v},P_1(x_1,y_1)) = [r\cos\theta\tan(\arctan(\frac{x_1}{r\cos\theta}) - \frac{\hat{u}}{r}), \frac{(y1+\hat{v})\cos\theta}{\cos(\arctan(\frac{x_1}{r\cos\theta}) - \frac{\hat{u}}{r})}]^T, \tag{12}$$

where r and θ are curvature radius and curvature angle, respectively. $\hat{V}_{t,j}$ is a 8×1 coordinate matrix for 4 vertices of jth grid at time t on the deformed image.

$$\hat{V}_{t,j} = \left[\hat{x}_{t,j,a}, \hat{y}_{t,j,a}, \hat{x}_{t,j,b}, \hat{y}_{t,j,b}, \hat{x}_{t,j,c}, \hat{y}_{t,j,c}, \hat{x}_{t,j,d}, \hat{y}_{t,j,d}\right]^T. \tag{13}$$

3.6. *Total energy*

Our total deformation energy E for each frame at time t, E_t is the linear combination of shape preservation energy E_S, depth energy E_F, boundary constraints E_V and E_H, and the temporal coherence constraint $E_{t,T}$ with the corresponding weighting factor β_f, β_b and β_t:

$$E_t = E_{t,S} + \beta_f E_{t,F} + \beta_b(E_{t,H} + E_{t,V}) + \beta_t E_{t,T}. \tag{14}$$

We set the high value of $\beta_b = 10^6$ to impose hard boundary constraint. For β_f and β_t, our proposed method follows similar parameter settings as in other multi-term deformation methods [6,8,9]. We find that $\beta_f = 1$ and $\beta_t = 500$ work well during experiments, and that small changes do not affect results significantly. We use these fixed settings for the remaining of this paper. The final minimization problem is a linear quadratic function and can be solved by a linear least-squares minimization. We use Pardiso [26] to solve our linear system.

4. Experimental results

4.1. *Implementation*

We implemented our method on the Microsoft Visual C++ 2012 platform with a 3.2 GHz Quad-Core Intel-i7 CPU and 16 GB of RAM. Each input video is processed frame by frame. We first normalize

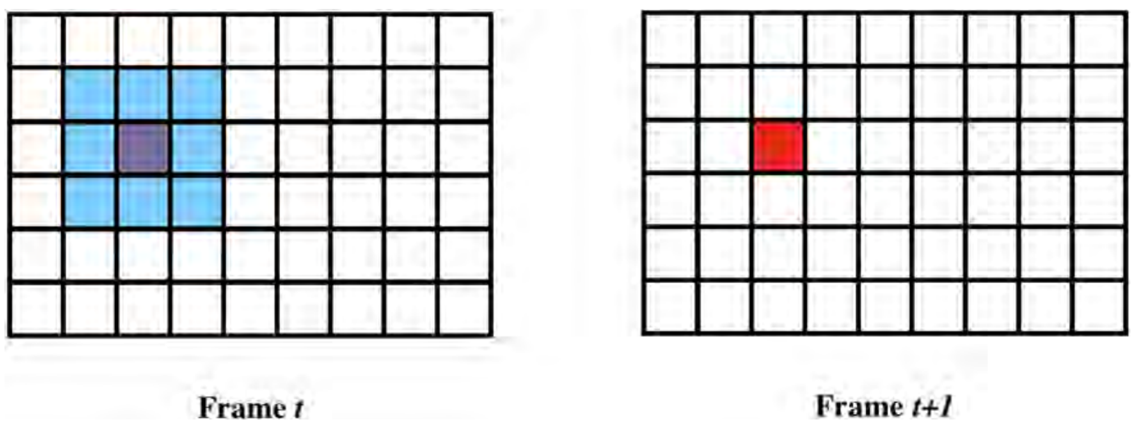


Fig. 5. For each grid at time $t+1$ (*i.e. the red grid*), blue grids at time t would be searched.

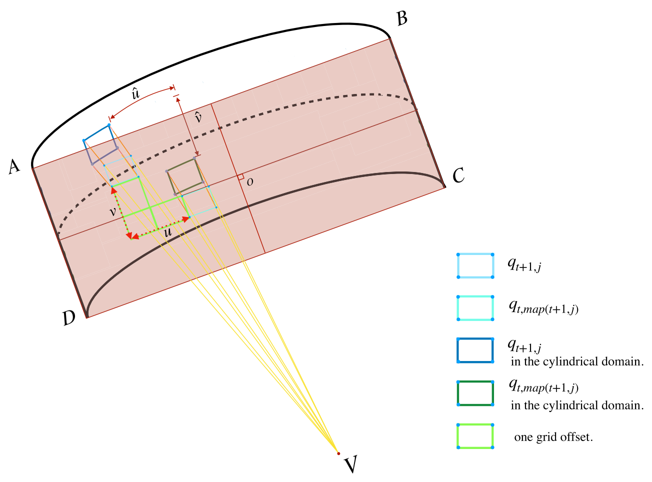

A
B
C
D
V
o
$\hat{u}$
$\hat{v}$
u
v
$q_{t+1,j}$
$q_{t,map(t+1,j)}$
$q_{t+1,j}$ in the cylindrical domain.
$q_{t,map(t+1,j)}$ in the cylindrical domain.
one grid offset.

Fig. 6. Horizontal and vertical offsets (u and v) between grid $q_{t+1,map(t+1,j)}$ and grid $q_{t,j}$ in the planar domain can be geometrically transformed into cylindrical offsets $\hat{u}$ (horizontal) and $\hat{v}$ (vertical). In Fig. 6, $q_{t+1,map(t+1,j)}$ translates 2 and 2 grids in the planar domain, horizontally and vertically. ($u = 2w_g$ and $v = 2h_g$. w_g is the width of each planar grid. h_g is the height of each planar grid.).

the depth map $\xi(i,j)$ to the range $[0,255]$ and use the following pre-processing:

$$\xi'(i,j) = \frac{255}{2}\left(1 + \tanh\frac{2\xi(i,j)}{255}\right). \tag{15}$$

After computing the globally optimal deformation, the output video is rendered with a GPU-based standard texture mapping. The time cost mainly lies in the grid mapping, the optimization of the deformation and the GPU-based texture mapping. We uniformly cover the grid with size 64 × 30 pixels on each frame. For our *cam* scene (1280 × 720), these steps take around 0.75 s and 0.09 s, respectively, and the total execution time is less than 0.91 s for each frame.

Concerning the processing time and the need for the depth map as input, our method could be used during movie making process rather than the movie playback. For computer graphics based movies, the depth map is readily available during rendering. For real-world movies, the depth map could be captured by the RGBD camera, or from a stereo camera pair. As our method does not require a very accurate depth map, depth estimation from video sequence [27] or single frames [28] could also be applied.

4.2. Data

The real-world video *Breakdancers* (we refer to it as *dancer*) we used was produced by Interactive Visual Media group at Microsoft Research [29]. We render the other CG videos from 3D models in Blender 2.79. Except for Butterfly, other CG cases are rendered as stereoscopic videos which can be watched with 3D glasses. All the CG blender projects are open projects on Blender Cloud (*cam*, *cam1b* and *dweets*).

4.3. Deformation results

We compare our deformation results with those produced by *Perspective mapping*, *Uniform mapping* and Lu et al. [6] with curvature angle $\theta = 90°$. Fig. 7 shows a llama moving from the left side to the middle of the scene. When using *perspective mapping* (see Fig. 7(b)), some content near the top and bottom boundaries is cropped. In these two frames, some grass and the top of the background hill are missing. The results of the other two mapping methods do not lead to a visible loss of content, though there is visible distortion of the fence and the llama in the result of Lu et al. [6] (see Fig. 7(d)). The distant fence is straight at frame 30, while it is bent at frame 42. As for *Uniform mapping* (see Fig. 7(c)), the face of the llama is stretched seriously at frame 30. At frame 42, the back-legs of the animal are stretched, while its head and front-legs are correct. For our proposed method, unlike *uniform mapping*, we preserve the shape of foreground objects without too much distortion (see our result at frame 42). Unlike Lu et al. [6] where static background objects are more distorted in order to preserve the shape and size of foreground objects (see the fence at frame 42 next

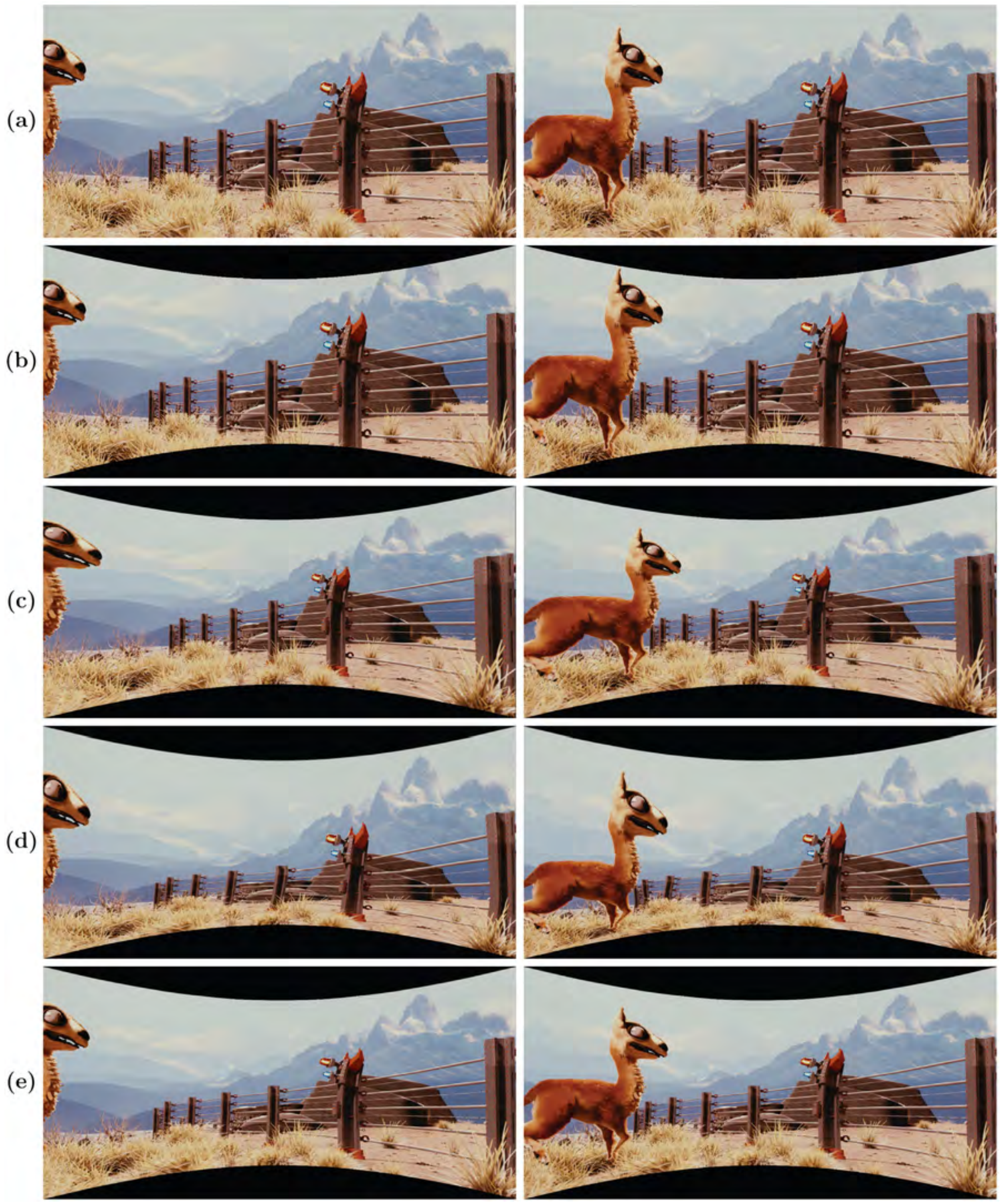


Fig. 7. 2 input video frames of case *cam1b (a)* and the corresponding deformation results with $\theta = 90°$ by perspective mapping *(b)*, uniform mapping *(c)*, Lu et al. [6] *(d)* and our proposed method *(e)*. (frame 30 and frame 42) ©Blender Foundation.

to the llama's neck), our method (see Fig. 8(e)) balances the importance of the moving foreground objects and static background objects.

Fig. 8 is a video showing 5 persons, and each of them has their own individual motion in the scene. When it comes to shape preservation, people close to the left and right boundaries are stretched visibly by means of *uniform mapping* (see Fig. 8(c)). Moreover, the face of people close to the right side is distorted differently within the selected 2 frames. The face at frame 0 is wider than that at frame 21. When using *perspective mapping* (see Fig. 8(b)), the right hand of the central dancer at frame 0 and the feet at frame 21 are missing and can be easily noticed. The left-side floor at frame 21 is distorted more heavily (bent effect) than those at frame 0 which leads to flickering effects in the result of Lu et al. [6] (see Fig. 8(d)). Clearly, while exhibiting little distortion on the wall and floor, our proposed method (see Fig. 8(e)) still preserves the shape of dynamic objects and the static content with temporal stability.

Unlike the first 2 cases, the method of Lu et al. [6] do not work well for *cam*. Lu et al. aim to keep the shape of foreground objects (stones, desert and the llama). The llama is close to the bottom boundary in this scene. Thus, the head and body part are well preserved by Lu et al. [6], while the legs near the bottom are badly distorted, and this is clearly visible. Besides loss of *perspective mapping*, the stretching effect near the left and right sides can be easily observed in *uniform mapping* (stones on the left side).

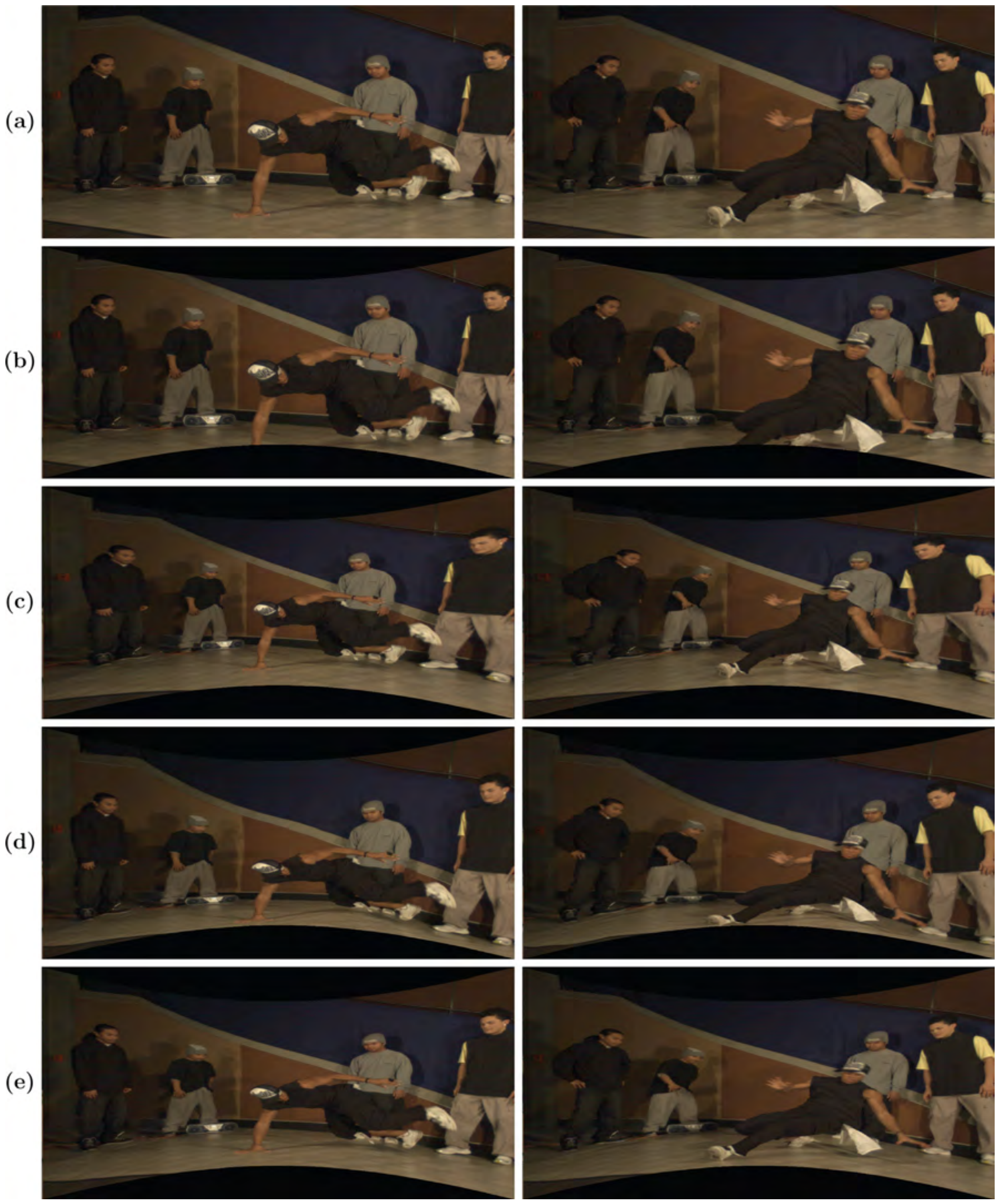


Fig. 8. 2 input video frames of *dancer (a)* and the corresponding deformation results with $\theta = 90^{\circ}$ by perspective mapping *(b)*, uniform mapping *(c)*, Lu et al. [6] *(d)* and proposed method *(e)*. (frame 0 and frame 21).

4.4. Evaluation

Objective evaluation:

The object shape changes in a movie scene are not basic enough to be described using complex mathematical formulas shown in [31–33]. Moreover, it is not easy to evaluate the temporal coherence of dynamical content without precise segmentation results. Inspired by Lu et al. [6] which use the centroid of grids to evaluate shape preservation, we introduce an objective metric aiding us to evaluate the deformed video from two aspects: *(i)* shape distortion of the deformed result; *(ii)* temporal coherence of the deformed result compared to the input sequence.

We divide the grids at frame t into dynamic grids set $DG(t)$ and static grids set $SG(t)$ respectively by Algorithm 1. The Euclidean distance $d_{(\cdot)}(t)$ between the centroids of two consecutive frames (frame t and $t+1$, where $(\cdot)$ can be either d – dynamic, or s – static) can describe the temporal movements of $DG(t)$ and $SG(t)$, respectively. Then we calculate the absolute difference $D_{(\cdot)}(t)$ between $d_{(\cdot)}^{deformed}(t)$ and $d_{(\cdot)}^{original}(t)$ from $DG(t)$ and $SG(t)$.

The **mean** value of $D_{(\cdot)}(t)$ can evaluate the shape distortion temporally and the **variance** value of $D_{(\cdot)}(t)$ can evaluate the temporal coherence.

Results are given for 7 cases: ***butterfly*** (10s, $12fps$), ***cam*** (12s, $12fps$), ***cam1b*** (13s, $12fps$), ***dweets*** (10s, $12fps$) and ***dancer*** (5s, $12fps$).

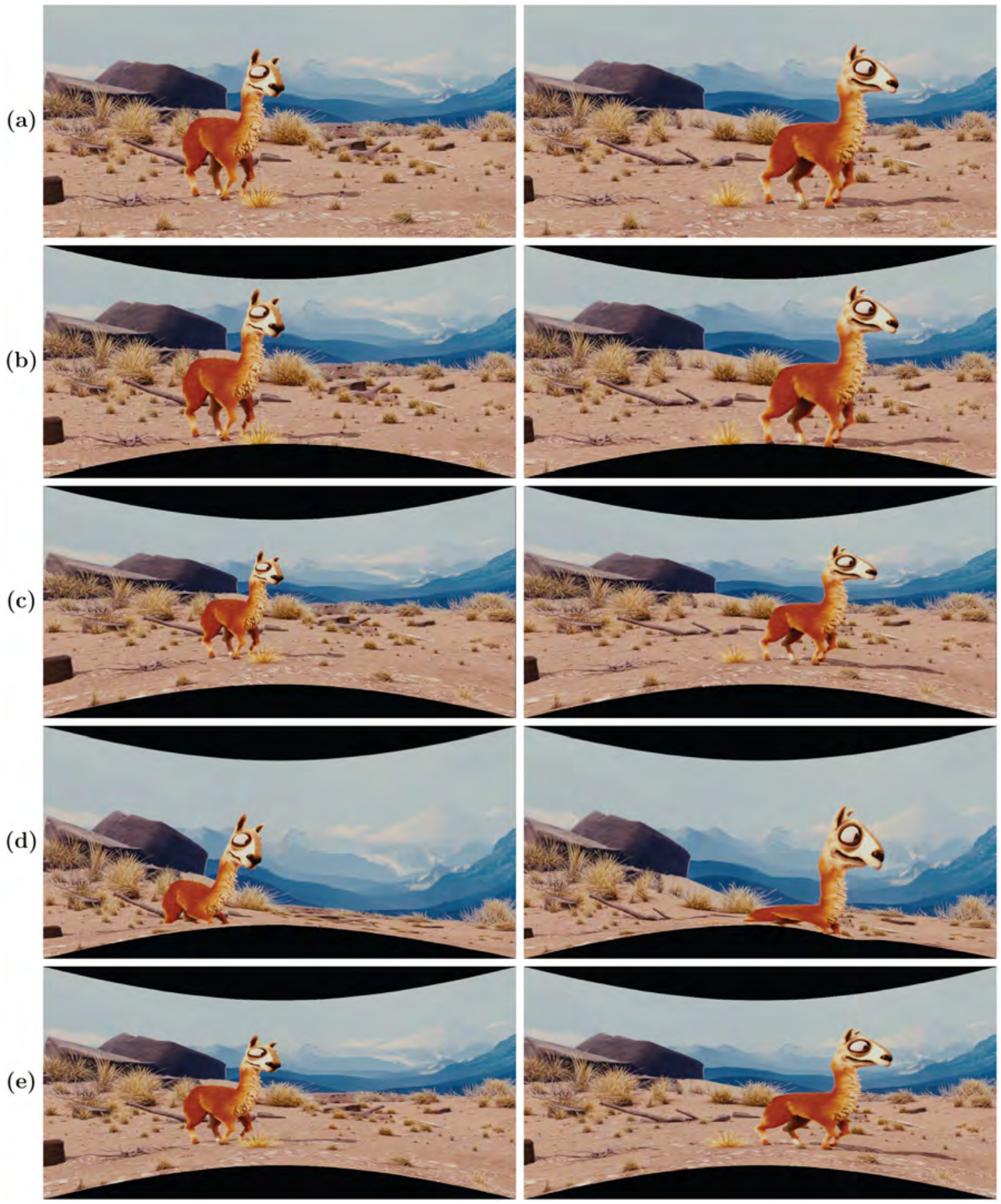
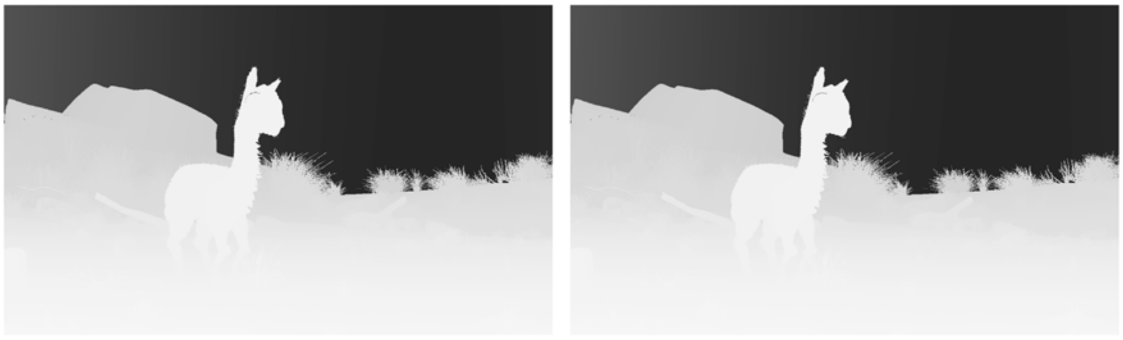


Fig. 9. 2 input video frames of *cam (a)* and the corresponding deformation results with $\theta = 90°$ by perspective mapping *(b)*, uniform mapping *(c)*, Lu et al. [6] *(d)* and our proposed method *(e)*. ©Blender Foundation.

Moreover, *cam*, *cam1b* and *dweets* are stereoscopic videos with left and right views (labeled as L and R in the tables). We calculate the mean and variance values of $D_d(t)$ and $D_s(t)$ with $\theta = 40°$ and $\theta = 90°$. Because *perspective mapping* loses content, we compare only *uniform mapping* and Lu et al. [6] with our proposed method.

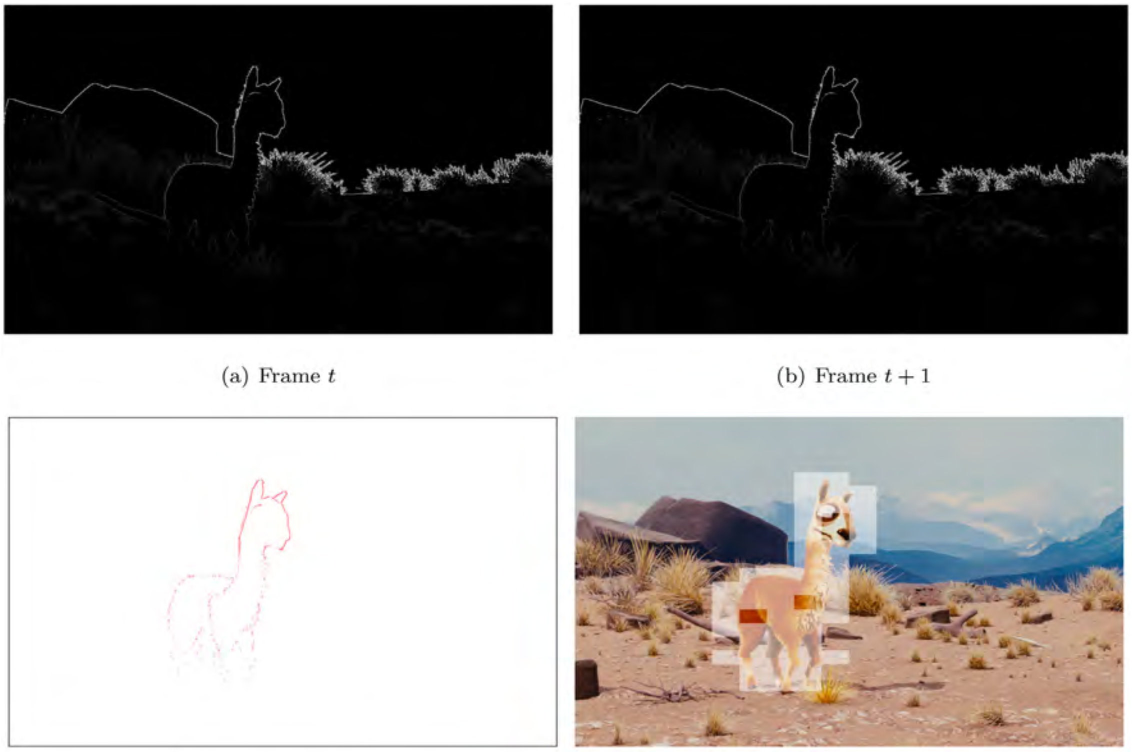
In Tables 1 and 3, when $\theta = 40°$, the mean values of all methods are competitive and our proposed method has only a slight advantage compared the other two, except for *cam* left view and right view. Indeed, when the value of deformation angle θ is not too large, the deformed results by means of these three methods are very close to the corresponding ground truth. Both dynamic content and static content are preserved well respectively. This can also be concluded from Tables 2 and 4. The centroids of each frame are as stable as they are in the ground truth. Scene *cam* (L and R) is an exception for Lu et al. [6] with $\theta = 40°$ (see Fig. 12(c) and the second last row of Fig. 9). The shape of dynamic objects is distorted seriously (the legs of the llama are shortened). Besides dynamic objects, the right side ground is heavily compressed. Both dynamic content and static content are too far off from ground truth and they are not stable enough.

When $\theta = 90°$, the mean values of $D_d(t)$ by uniform mapping do not perform as well as Lu et al. [6] and our proposed method. This is because dynamic content moving from the left side to the right side is more stretched when it is close to the sides. Mean values of $D_d(t)$ in *cam* are smaller with Lu et al. [6] than those of uniform mapping,



(a) Frame t (b) Frame $t+1$

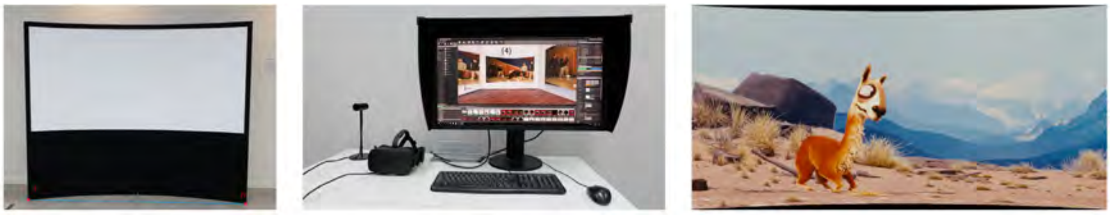
Fig. 10. Input depth maps at time t and $t+1$.



(a) Frame t (b) Frame $t+1$

(c) Temporal gradient map. (d) Dynamic grids.

Fig. 11. Corresponding contour maps (a)(b) of Fig. 10 calculated by Canny edge detector [30], temporal gradient map (c) and dynamic grids (d), shown in white. ©Blender Foundation.



(a) Real-world curved surface. (b) Virtual scene and Oculus Rift. (c) Deformation result

Fig. 12. (a) shows the real-world curved screen with curvature angle $\theta = 40°$. (b) shows the virtual scene we build within which the curved screens with curvature angle $\theta = 90°$. (c) shows the deformation result of scene *cam* frame 0 by Lu et al. [6] with $\theta = 40°$.

Algorithm 1 Algorithm for *static grids* and *dynamic grids*.

```
Require: Meshed depth map $depth(t)$ at time $t$ and meshed depth map
    $depth(t+1)$ at time $t+1$;
Ensure: dynamic grids set $DG(t)$ and static grids set $SG(t)$;
1: Detect the contour map $contour(t)$ and $contour(t+1)$ from $depth(t)$
   and $depth(t+1)$ by Canny edge detector [30] (see Fig. 11.);
2: Compute the temporal gradient map $gradient(t)$ of $contour(t)$ and
   $contour(t+1)$ (see Fig. 11(c).);
3: for all $grid \in gradient(t)$ do
4:     if gradient of $grid \geq 0$ then $grid$ belongs to $DG(t)$;
5:     else $grid$ belongs to $SG(t)$;
6:     end if
7: end for
8: return $SG(t)$, $DG(t)$;
```

Table 1
Objective evaluation: mean of $D_d(t)$ for various scenes and configurations.

Scene	θ	Mean of $D_d(t)$		
		Uniform	Lu *et al.* [6]	Proposed method
butterfly	40°	0.566302	0.658396	**0.262247**
cam (L)	40°	0.547710	1.959574	**0.337895**
cam (R)	40°	0.714413	1.745877	**0.414786**
cam1b (L)	40°	0.551407	0.759210	**0.352015**
cam1b (R)	40°	0.618535	0.735290	**0.414190**
dweets (L)	40°	0.575184	0.531805	**0.344856**
dweets (R)	40°	0.514265	0.509262	**0.307742**
dancer	40°	0.791693	0.882472	**0.523511**
butterfly	90°	2.783716	1.573083	**0.972567**
cam (L)	90°	2.722349	2.537118	**1.332737**
cam (R)	90°	3.548114	2.581514	**1.643616**
cam1b (L)	90°	2.788562	1.553388	**1.371216**
cam1b (R)	90°	3.123621	1.708179	**1.627980**
dweets (L)	90°	2.879390	1.559232	**1.300870**
dweets (R)	90°	2.553689	1.419156	**1.165210**
dancer	90°	3.896591	2.330255	**2.140830**

Table 2
Objective evaluation: variance of $D_d(t)$ for various scenes and configurations.

Case	θ	Variance		
		Uniform mapping	Lu *et al.* [6]	Proposed method
butterfly	40°	0.223306	0.385706	**0.047679**
cam (L)	40°	0.396945	13.119368	**0.203644**
cam (R)	40°	0.369766	6.887471	**0.160362**
cam1b (L)	40°	0.587323	0.795642	**0.196472**
cam1b (R)	40°	0.517805	0.729009	**0.190646**
dweets (L)	40°	0.220330	0.245499	**0.086291**
dweets (R)	40°	0.179446	0.222211	**0.084249**
dancer	40°	0.198892	0.487549	**0.183241**
butterfly	90°	5.711604	1.685794	**0.773003**
cam (L)	90°	9.906974	19.700668	**3.638118**
cam (R)	90°	9.152859	10.212986	**2.856850**
cam1b (L)	90°	14.875714	3.950359	**3.280384**
cam1b (R)	90°	13.380880	3.548495	**3.054187**
dweets (L)	90°	5.524043	2.111583	**1.382643**
dweets (R)	90°	4.432388	2.033815	**1.477977**
dancer	90°	4.984207	4.717884	**3.842195**

and this is due to $\theta = 90°$ being a relatively large curvature angle, and the content being close to the sides. The centroid coordinate is mainly influenced by the top and bottom boundaries rather than the shape of the dynamic content. But in Table 2, the variance values in scene *cam* (L and R) by Lu et al. [6] are much higher than those of uniform mapping, which can explain the failure of Lu et al. [6] in scene *cam* (L and R). Moreover, for dynamic content, our method performs the best when compared with the ground truth and is temporally stable enough, with the lowest mean and variance values simultaneously.

For static content, uniform mapping and Lu et al. [6] have competitive results, but our proposed method performs the best among

Table 3
Objective evaluation: mean of $D_s(t)$ for various scenes and configurations.

Case	θ	Mean		
		Uniform mapping	Lu *et al.* [6]	Proposed method
butterfly	40°	0.112230	0.120356	**0.043974**
cam (L)	40°	0.093427	0.824781	**0.043148**
cam (R)	40°	0.120436	0.856510	**0.060556**
cam1b (L)	40°	0.131180	0.251062	**0.086717**
cam1b (R)	40°	0.128691	0.260109	**0.089005**
dweets (L)	40°	0.137989	0.208948	**0.077300**
dweets (R)	40°	0.128665	0.198681	**0.069185**
dancer	40°	0.346851	0.962201	**0.214769**
butterfly	90°	0.549204	0.251153	**0.157225**
cam (L)	90°	0.464607	0.941727	**0.152238**
cam (R)	90°	0.599748	0.956593	**0.225033**
cam1b (L)	90°	0.657464	0.507315	**0.341004**
cam1b (R)	90°	0.646345	0.536356	**0.359439**
dweets (L)	90°	0.685396	0.451480	**0.289625**
dweets (R)	90°	0.642425	0.411031	**0.255143**
dancer	90°	1.709182	1.449238	**0.858678**

Table 4
Objective evaluation: variance of $D_s(t)$ for various scenes and configurations.

Case	θ	Variance		
		Uniform mapping	Lu *et al.* [6]	Proposed method
butterfly	40°	0.007072	0.026133	**0.001000**
cam (L)	40°	0.013391	3.230232	**0.001381**
cam (R)	40°	0.015433	2.470027	**0.005163**
cam1b (L)	40°	0.083865	0.375567	**0.072037**
cam1b (R)	40°	0.084109	0.460068	**0.081295**
dweets (L)	40°	0.010071	0.073540	**0.003026**
dweets (R)	40°	0.007804	0.074220	**0.002646**
dancer	40°	0.031434	0.797369	**0.026745**
butterfly	90°	0.174683	0.051787	**0.014210**
cam (L)	90°	0.335818	4.000705	**0.017767**
cam (R)	90°	0.385220	2.645810	**0.097365**
cam1b (L)	90°	2.160249	2.197491	**1.344434**
cam1b (R)	90°	2.176553	2.677143	**1.589080**
dweets (L)	90°	0.245445	0.163865	**0.048643**
dweets (R)	90°	0.192821	0.161413	**0.046976**
dancer	90°	0.750359	1.891118	**0.555956**

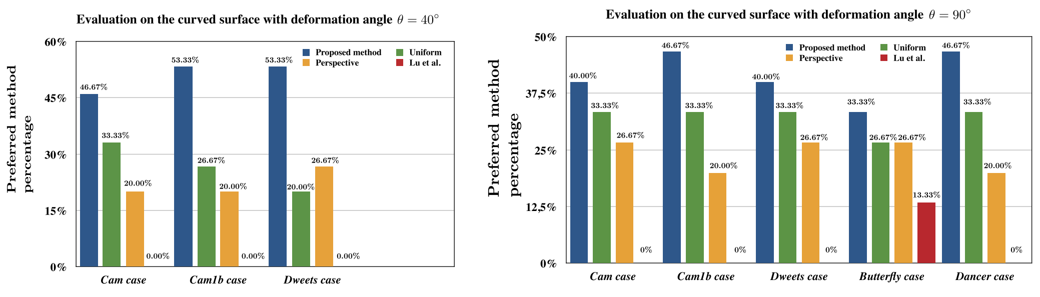
them. Moreover, our temporal coherence preservation keeps not only dynamic content but also static content stable, as can be seen in Table 4.

Subjective evaluation:

We invited *15* viewers to perform user case study. The test includes 2 different configurations:

1. A real-world curved screen with curvature angle $\theta = 40°$, as shown in Fig. 12(a).
2. A virtual room including 5 screens (4 curved + 1 flat for reference) with curvature angle $\theta = 90°$, as shown in Fig. 12(b).

Each viewing experiment was performed at the curvature center of the corresponding curved displays. For the real-world curved screen, each participant stood at the curvature center directly. For the virtual scene, each participant was asked to wear an Oculus Rift headset and place themselves on the curvature center of each virtual display. Test 1 took around 10 minutes per viewer and Test 2 took around 20 minutes per viewer. For each test, we measured the choice percentage of each method. In Test 1, 3 sequences were displayed. The results of each deformation method (uniform mapping, perspective mapping, Lu et al. [6] and our proposed method) were shown in a random order. In Test 2, 5 sequences were displayed. Four deformed results were simultaneously displayed onto the curved surfaces. For reference, one



(a) Curvature angle $\theta = 40°$. (b) Curvature angle $\theta = 90°$.

Fig. 13. Subjective test results with different curvature angles. In the case $\theta = 90°$, the screen is simulated and viewed through the Oculus Rift.

flat screen showing the original input video was also present in the virtual scene. The viewers were invited to pick their preferred method. Fig. 13 shows the subjective results. In all cases, our proposed method was picked the most, above uniform mapping, perspective mapping and Lu et al. [6]. The deformed results processed by Lu et al. [6] were least chosen in most scenes, because they cause visible waving artifacts between frames. In *butterfly* scene, the flickering effect is not so noticeable, which explains why some viewers selected the deformed result by Lu et al. [6] as their preference.

5. Conclusions

In this paper, we present a video deformation method to solve the problem of video projection onto a curved display with minimal distortion and high temporal coherence between consecutive frames. Our method avoids the over-preservation of foreground content and the distortion of background. Moreover, the flickering or waving artifacts are removed between consecutive frames. Both objective and subjective evaluation show that our proposed method outperforms every other existing work for video retargeting onto a curved surface.

To make our work more broadly applicable, future works include depth estimation from stereoscopic video input so that we can remove the depth map as a requirement. Furthermore, adaptive mesh representation formats can be used to improve the objective metric result rather than the uniform grid used in our present work.

Acknowledgments

This work is supported by Innoviris (DrIvINg), FWO (PhD fellowship Q. Bolsee), and Tianjin Natural Science Foundation (No. 18JCYBJC41300 and No. 18ZXZNGX00110). We would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions which have improved this paper. We would also like to show our gratitude to the Blender open project authors for their excellent 3D animation models.

Conflict of interest

This is the first submission of this manuscript and no parts of this manuscript are being considered for publication elsewhere. All authors have approved this manuscript. No author has financial or other contractual agreements that might cause conflicts of interest.

References

[1] W. Ling, Mathematics and the Sciences of the Heavens and the Earth, Cambridge University Press, 1970.
[2] S. Avidan, A. Shamir, Seam carving for content-aware image resizing, in: ACM Trans graphics (TOG), Vol. 26, ACM, 2007, p. 10.
[3] S. Battiato, G.M. Farinella, G. Puglisi, D. Ravi, Saliency-based selection of gradient vector flow paths for content aware image resizing, IEEE Trans. Image Process. 23 (5) (2014) 2081–2095.
[4] S. Qi, J. Ho, Seam segment carving: retargeting images to irregularly-shaped image domains, in: European Conference on Computer Vision, Springer, 2012, pp. 314–326.
[5] J. Shen, D. Wang, X. Li, Depth-aware image seam carving, IEEE Trans. Cybern. 43 (5) (2013) 1453–1461.
[6] S.-P. Lu, R.-M. Florea, P. Cesar, P. Schelkens, A. Munteanu, Efficient depth-aware image deformation adaptation for curved screen displays, in: Proceedings of the on Thematic Workshops of ACM Multimedia 2017, ACM, 2017, pp. 442–450.
[7] G.-X. Zhang, M.-M. Cheng, S.-M. Hu, R.R. Martin, A shape-preserving approach to image resizing, in: Computer Graphics Forum, Vol. 28, Wiley Online Library, 2009, pp. 1897–1906.
[8] C.-H. Chang, Y.-Y. Chuang, A line-structure-preserving approach to image resizing, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1075–1082.
[9] R. Carroll, M. Agrawala, A. Agarwala, Optimizing content-preserving projections for wide-angle images, ACM Trans. Graph. 28 (3) (2009) 43–1.
[10] A. Mansfield, P. Gehler, L. Van Gool, C. Rother, Scene carving: scene consistent image retargeting, in: European Conference on Computer Vision, Springer, 2010, pp. 143–156.
[11] K.-Y. Lee, C.-D. Chung, Y.-Y. Chuang, Scene warping: layer-based stereoscopic image resizing, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 49–56.
[12] R. Raskar, J. Baar, T. Willwacher, S. Rao, Quadric transfer for immersive curved screen displays, in: Computer Graphics Forum, Vol. 23, Wiley Online Library, 2004, pp. 451–460.
[13] B. Sajadi, A. Majumder, Autocalibrating tiled projectors on piecewise smooth vertically extruded surfaces, IEEE Trans. Vis. Comput. Graph. 17 (9) (2011) 1209–1222.
[14] M. Rubinstein, A. Shamir, S. Avidan, Improved seam carving for video retargeting, ACM Trans. Graph. (TOG) 27 (3) (2008) 16.
[15] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, H.-P. Seidel, Motion-aware temporal coherence for video resizing, ACM Trans. Graph. 28 (5) (2009) 127.
[16] J. Zhang, S. Li, C.-C.J. Kuo, Compressed-domain video retargeting, IEEE Trans. Image Process. 23 (2) (2014) 797–809.
[17] J. Wei, C.-F. Li, S.-M. Hu, R.R. Martin, C.-L. Tai, Fisheye video correction, IEEE Trans. Vis. Comput. Graphics 18 (10) (2012) 1771–1783.
[18] R.G. Von Gioi, J. Jakubowicz, J.M. Morel, G. Randall, Lsd: a fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 722–732.
[19] D. Tsai, M. Flagg, A. Nakazawa, J.M. Rehg, Motion coherent tracking using multi-label mrf optimization, Int. J. Comput. Vis. 100 (2) (2012) 190–202.
[20] B.-W. Hong, J.-K. Koo, S. Soatto, Multi-Label Segmentation via Residual-Driven Adaptive Regularization, arXiv preprint arXiv:1702.08336.
[21] D.C. Luvizon, D. Picard, H. Tabia, 2d/3d pose estimation and action recognition using multitask deep learning, arXiv preprint arXiv:1802.09232.
[22] P.F. Alcantarilla, A. Bartoli, A.J. Davison, Kaze features, in: European Conference on Computer Vision, Springer, 2012, pp. 214–227.
[23] H. Bay, T. Tuytelaars, L. Van Gool, Surf: speeded up robust features, in: European Conference on Computer Vision, Springer, 2006, pp. 404–417.
[24] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[25] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[26] O. Schenk, M. Bollhöfer, R.A. Römer, On large-scale diagonalization techniques for the anderson model of localization, SIAM Rev. 50 (1) (2008) 91–112.
[27] M. Martinello, P. Favaro, Depth estimation from a video sequence with moving and deformable objects, in: IET Conference on Image Processing, 2012, pp. 1–6.
[28] A. Saxena, S.H. Chung, A.Y. Ng, Learning depth from single monocular images, in: Advances in Neural Information Processing Systems, 2006, pp. 1161–1168.
[29] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, in: ACM Trans. Graph. (TOG), Vol. 23, ACM, 2004, pp. 600–608.

[30] J. Canny, A computational approach to edge detection, in: Readings in Computer Vision, Elsevier, 1987, pp. 184–203.
[31] R. Basri, L. Costa, D. Geiger, D. Jacobs, Determining the similarity of deformable shapes, Vis. Res. 38 (15–16) (1998) 2365–2385.
[32] S. Antani, D. Lee, L.R. Long, G.R. Thoma, Evaluation of shape similarity measurement methods for spine x-ray images, J. Vis. Commun. Image Represent. 15 (3) (2004) 285–302.
[33] R.C. Veltkamp, L.J. Latecki, Properties and performance of shape similarity measures, in: Data Science and Classification, Springer, 2006, pp. 47–56.