CPU-efficient free view synthesis based on depth layering

CPU-EFFICIENT FREE VIEW SYNTHESIS BASED ON DEPTH LAYERING

Aleksandra Chuchvara, Mihail Georgiev, Atanas Gotchev

Tampere University of Technology, Tampere, Finland

ABSTRACT

In this paper, a new approach for depth-image based rendering (DIBR) based on depth layering is proposed. The approach effectively avoids the non-uniform to uniform resampling stage, which is otherwise inherent for classical DIBR. In contrast, the new approach employs depth layering, which approximates the scene geometry by a multi-planar surface, given the depth is defined within a closed range. Such an approximation facilitates a fast reverse coordinate mapping from virtual to reference view where straightforward resampling on a uniform grid is performed. The proposed rendering approach ensures an automatic z-ordering and disocclusion detection, while being very efficient even for CPU-based implementations. It is also applicable for reference and virtual views with different resolutions and as such can serve depth upsampling, view panning and zooming applications. The experimental results demonstrate its real-time capability, while the quality is comparable with other view synthesis approaches but for lower computational cost.

Index Terms – non-uniform, resampling, layered rendering, 2D+Z, DIBR, CPU.

1. INTRODUCTION

3D video is referred as to a type of visual media in which the viewer experiences depth perception of the recreated 3D scenery. Interest in 3D video production has notably increased recently along with the advances in 3D display technologies. Research in this area spans the whole media processing chain from capture to display [1]. 3D video technology can be applied in wide variety of applications, including home 3DTV and free view-point (FTV) entertainment and 3D video services for mobile devices. 3DTV and FTV would bring a more realistic visual experience to the home environment, i.e. users would be able to navigate through the scene and choose a different viewpoint at will. Contemporary auto-stereoscopic 3D displays are able to synthesize desired views using depth image-based rendering (DIBR). For DIBR algorithms depth information of a rendered scene need to be provided in a form of a depth map associated with color, e.g. in a view-plus-depth format [2] (cf Figure 1a).

In order to render virtual views from view-plus-depth 3D representation, a *3D warping* technique is used. It consists of the following steps. First, the points of the original view are projected into global world coordinates using reference camera projection matrix and depth associated with each point, the result is a 3D point-cloud in global coordinates. The second step is back projecting the real 3D points onto the image plane of the virtual view using the projection matrix of the virtual camera. The 3D warping projections result in data points containing known values from the original view scattered on the virtual camera image plane. This means that the unknown values at the regular pixel positions of the virtual view need to be estimated from irregular data thus imposing a non-uniform to uniform resampling. Accurate non-uniform to uniform resampling is still an open-research problem [3].

In this paper we propose a computationally efficient workaround of the "3D warp → non-uniform resampling" approach, avoiding the non-uniform resampling stage by employing depth layering, which facilitates a resampling at the uniform grid of the given reference camera. Another limitation of the classical approach is that non-uniform data cannot provide accurate estimation and masking of dis-occluded and hidden data (disocclusions detection and z-ordering), while the layered rendering naturally embeds z-ordering and disocclusion detection within the view-generation process. The proposed approach is also applicable for virtual views generation in the general case, i.e. in case of different camera parameters in terms of position, orientation, focal length and varying sensor spatial resolutions.

2. RELATED WORKS

Arbitrary view synthesis can be implemented in real time by GPU-based rendering means. The basic idea is to represent the depth data as a 3D triangulated surface where triangles can be formed in-between the neighborhood on the depth image grid vertices. After that, each pixel of the depth image is converted into 3D vertex using camera projection matrix and depth data [4]. The accompanying color image is applied as a texture to the 3D depth surface [5]. The result is a 3D textured surface of the scene containing no gaps: as the GPU rasterizes a triangle, it interpolates the depth and texture coordinates of the triangle's vertices across the triangle face using fast bilinear interpolation implemented in hardware. The textured surface can be then rotated, scaled and translated as needed to generate any arbitrary view (see Figure 1b for an example).

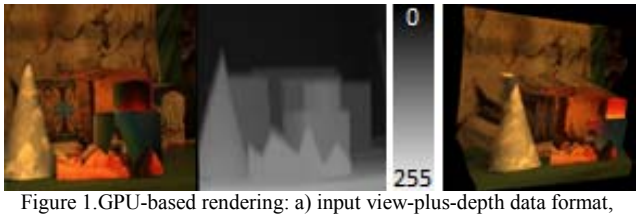

Figure 1.GPU-based rendering: a) input view-plus-depth data format, b) 3D textured surface.

Layer-based representations of multi-modal (e.g. depth) data have been studied in a number of works [6, 7]. In [6], a layer-based method for an arbitrary view synthesis from a set of real views is presented. A depth layer model of the scene is generated from a collection of camera views and the plenoptic sampling theory is used to predict the required number of layers. Once the layer positions are decided, each pixel within an image is assigned to a particular layer based on disparity. The layer models are calculated for all input images. Then, with such geometry EPI line structure is used to interpolate a new image from existing images. In [7], an extension of texture mapping is

978-1-4799-4758-4/14/$31.00 ©2014 IEEE

given for efficient image based rendering. For a depth image with an orthogonal displacement in each pixel, it is decomposed by displacement into a series of layered textures with each one having the same displacement for all texels (texture pixel). The plane-to-plane texture mapping is then used to map these layered textures to produce novel views. While these methods address various aspects in layer-based representations for virtual view synthesis, the key difference of our proposed approach is that it uses a reversed mapping procedure: from virtual to the original view, which reduces the non-uniform resampling problem to a classical interpolation problem on a regular grid.

3. LAYERED RENDERING APPROACH

3.1 Layering in disparity domain

The input is formed by the aligned color and range images, where the range image represents the depth map $z(x,y)$. For the layer based rendering the given depth map is first transformed into a disparity map, which is subsequently divided into layers. The disparity map is calculated with respect to the targeted virtual view, which is determined by the new camera position and the corresponding baseline b between the reference and the new camera positions. The disparity displacement is calculated in pixels as follows:

$$d(x,y)=\frac{b\cdot f}{z(x,y)}, \tag{1}$$

where f is the focal length of the cameras. The required number of layers depends on the minimum and maximum disparity displacements D_{min} and D_{max}. A layered map with L layers, l=1…L, is constructed by assigning pixels with disparity displacements within the interval [D_{min}, D_{min}+*1*) to the first layer, within the interval [D_{min}+*1*, D_{min}+*2*) to the second layer, and so on until $D_{min}+L \geq D_{max}$ and the interval [D_{min}+*L-1*, D_{max}) corresponds to the last layer. Figure 2c illustrates a layered disparity map. The layering is equivalent to considering the scene as formed by planes, where each layer corresponds to a plane in space perpendicular to the Z-axis of the scene and located at distance z_l (world coordinate system origin is considered placed at the reference camera position), so that the plane equation is $Z=z_l$, for l=1..L. z_l can be chosen as the average depth value between minimum and maximum depth values of the pixels belonging to the layer. Such flat layered representation of the scene geometry allows approximating the 3D transformations as a series of 2D planar perspective projections. This feature is used to render an arbitrary view as described in the next section.

Figure 2.Test scene: a) color, b) depth map, c) layered map

3.2 View synthesis

Having the scene depth surface approximated by a set of planes, a projective transformation, i.e. a *homography,* exists for each plane, which maps the points of the plane from the reference camera image plane to the virtual camera image plane. Thus, the 3D warping procedure can be approximated as a series of 2D perspective projections performed in a reversed way. Consider the *pinhole model* camera model [12]. The 3×3 projection matrices for the reference and the virtual cameras are defined as $\mathbf{P}=\mathbf{K}\cdot[\mathbf{I}|\mathbf{0}]$ (world origin is at the camera) and $\mathbf{P}'=\mathbf{K}'\cdot[\mathbf{R}|\mathbf{t}]$ correspondingly. Here $\mathbf{K}$ and $\mathbf{K}'$ are the 3×3 calibration matrices of the form:

$$\mathbf{K}_{3x3}=\begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

where f_x, f_y are the focal length components measured in pixels, p_x, p_y are the principal point coordinates; and $[\mathbf{R}|\mathbf{t}]$ is the combined 3×3 rotation matrix and the 3×1 translation column vector of the virtual camera with respect to the reference camera. The 3×3 homography $\mathbf{H}_l$ which maps points of the plane $Z=z_l$ from reference image plane to the virtual image plane is:

$$\mathbf{H}_l=\mathbf{K}'\cdot\left(\mathbf{R}+\mathbf{t}\cdot[0\,0\,1/z_l]\right)\cdot\mathbf{K}^{-1}. \tag{3}$$

To render a view at the virtual camera, projections of the points on the virtual camera grid onto the reference camera grid are calculated, so that the values of the projected points can be interpolated within the regular grid of the reference image. To calculate the projections, for each depth z_l, points of the virtual view image grid are back-projected onto the reference image plane using the inverse homography $\mathbf{H}_l^{-1}$. This way we obtain projections of the virtual image grid points as if they all are located on the plane $Z=z_l$ in space, or, according to our layering, if they all belong to the layer l of the reference image. In order to decide which of the projected points do belong to the layer l and how their values can be interpolated, we analyze the neighborhood of the projected points, as illustrated in Figure 3. First, we analyze the four-point neighborhood of the projected point, which is referred as *layer support*. If the layer support of the projected point *(u,v)* contains pixels from *current, next* or *previous* layers and there is at least one pixel from the *current* layer, the value P of the projected point *(u,v)* can be interpolated. Second, the $n\times n$-point neighborhood is checked where n is determined by the chosen interpolator's support. If *P(u,v)* can be interpolated within the current layer, use for the interpolation only those pixels from the $n\times n$ resampling support which belong to the layers *l-n/2..l+n/2* qualified as valid for interpolation. If all pixels within the resampling support are valid, the desired interpolator is directly used. If there are some invalid pixels, an interpolator with a smaller support is applied.

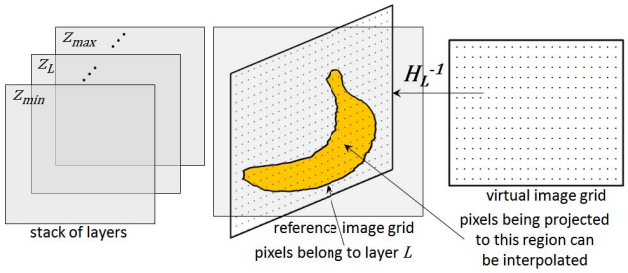

Figure 3.Layered rendering.

The obtained interpolated values are used to fill in the corresponding values on the virtual view image grid. In this way, by processing all layers, a novel view can be rendered as illustrated in Figure 4a. By processing layers from back to front z-ordering is maintained. Disoccluded regions are determined by pixels where no value was assigned, see Figure 4b. Edge pixels of disoccluded regions can also be detected as pixels without values that were projected in such a way, that they have pixels from *current* layer in their layer support, but some of the pixels in the layer support do not belong to the *current, next* or *previous* layers (cf Figure 4c). The projection incorporates all

parameters of a two-camera system; therefore the rendering script can be configured for different camera arrangements, asymmetric camera parameters in terms of f, $\mathbf{R}$, $\mathbf{t}$, and varying sensor spatial resolutions, which works also as rescaling or zooming of the rendered view.

The described method can be regarded as plane sweeping [8, 9] but implemented in a reverse manner. The plane sweeping approach uses layers to estimate depth, while the proposed method uses depth to find optimal depth layering. Furthermore, the inverse projections from the virtual to the reference view position the pixels being synthesized within the reference camera grid, thus avoiding non-uniform resampling.

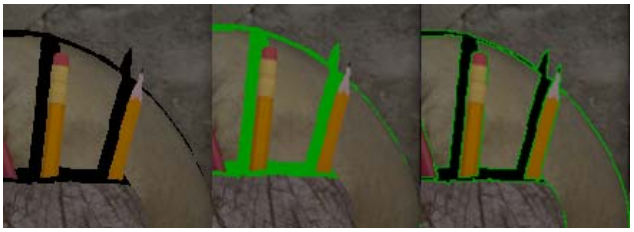

Figure 4.Layered rendering: a) result for translation vector (0.1, 0.1, 0.1) in meters, b) disoccluded regions, c) edge detection.

3.3 Backprojection optimizations

The virtual image grid is back-projected for each value of z_1. Instead of applying the homography $\mathbf{H}_i^{-1}$ every time, it is more efficient to compute the projection of the virtual image grid on some initial plane, e.g. Z=1, and then modify this projection to calculate the projection of the virtual image grid on any other plane $Z=z_i$ [9]. Consider two projections $\mathbf{H}_1$ and $\mathbf{H}_i$ that maps points from the planes Z=1 and $Z=z_i$ correspondingly to the virtual image grid:

$$\begin{aligned} \mathbf{H}_1 &= \mathbf{K}' \cdot \left(\mathbf{R} + \mathbf{t} \cdot [0\,0\,1]\right) \\ \mathbf{H}_i &= \mathbf{K}' \cdot \left(\mathbf{R} + \mathbf{t} \cdot [0\,0\,1/z_i]\right) \end{aligned}. \tag{4}$$

The homography $\mathbf{H}_{i1} = \mathbf{H}_i^{-1} \cdot \mathbf{H}_1$ maps points between planes Z=1 and $Z=z_i$ directly, by first applying forward projection form Z=1 to the virtual image grid, and then back-projecting them onto the $Z=z_i$. $\mathbf{H}_{i1}$ has a rather simple structure: if (u_1, v_1, w_1) is a point projected onto the plane Z=1, the corresponding point on the plane $Z=z_i$ is:

$$\begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix} = \mathbf{H}_i^{-1}\mathbf{H}_0 \begin{pmatrix} u_1 \\ v_1 \\ w_1 \end{pmatrix} = \begin{pmatrix} (1+c_3\delta)u_1 + (1-\delta)c_1 w_1 \\ (1+c_3\delta)v_1 + (1-\delta)c_2 w_1 \\ (1+c_3)w_1 \end{pmatrix}. \tag{5}$$

Here $\delta=1/z$ and $(c_1, c_2, c_3) = (\mathbf{r}_1\mathbf{t}, \mathbf{r}_2\mathbf{t}, \mathbf{r}_3\mathbf{t})$, where $\mathbf{t}$ is the translation column vector, and $\boldsymbol{r}_i$ is a row vector of transposed rotation matrix $\mathbf{R}^{\mathrm{T}}$. This way all projections can be calculated as linear combination of some pre-calculated projection of the virtual image grid.

In a direct approach, all virtual view pixels need to be processed for each layer, or L times in total. However, one can exclude pixels which value has been filled during the previous iteration, so that the current iteration processes the remaining pixels. In this case, to maintain z-ordering the layers have to be rendered in the order from front to back, so that the front objects are rendered first and, as the filled image regions are not re-render during the following iterations, the occluded parts of the backward objects are simply skipped, which gives the right result in case of occlusions without any depth testing.

4. EXPERIMENTAL RESULTS

We demonstrate the proposed resampling technique by an experiment performed on a photorealistic scene synthetically rendered by the Blender software [10]. A rendering script has been configured for different camera arrangements, asymmetric camera parameters in terms of f, $\mathbf{R}$, $\mathbf{t}$, and varying sensor spatial resolutions. We have designed a scene of high depth contrast varying within the range of 0.5÷7 m, which resembles a typical range of the available ToF depth sensing devices [13]. The scene contains objects of different reflection surfaces, materials and facing directions, and illumination noise. The scene is given in Figure 2a and 2b.

Using the script, a rather extreme case has been simulated. A horizontally aligned camera setup has been misaligned in forward/backward direction within the range 0 ÷ 0.25m for an arbitrary chosen relative pose in terms of $\mathbf{R}$ and $\mathbf{t}$, and for different resolution downscaling of the reference view, both for width and height from two to ten times in order to evaluate upsampling capabilities of the method. We compared the rendered results of the proposed approach against previous approaches by measuring the difference to ground-truth data in terms of Peak signal-to-noise ratio (*PSNR*) for the rendered color view. The comparative tests included a direct bilinear plane-fit resampling (bilinear triangulation) as used in a graphic accelerator hardware supported by OpenGL, and a method proposed in our previous work [4], where an intermediate resampling step by bilinear triangulation is used to place the virtual camera to the desired camera position, and a 2D image back-projective up-sampling to the desired camera grid is followed (denoted as VC method). Ground truth data (GT) was rendered in Blender with the same tested baseline, relative pose and misalignment, and with dis-occluded pixels being masked. All re-sampling approaches were programmed in Matlab. For the 2D interpolation (VC) and the proposed method (Proposed), bi-cubic interpolation was implemented [11]. The results plotted in Figure 5, show that for each tested case, our approach achieves comparable or better quality. Visual results for some scene details are given in Figure 6.

To measure the performance of the proposed method, it was implemented in C++ and tested on a computer with Intel Core i7-3770 CPU. The performance results are measured for the 480×640 resolution of reference and virtual image grids and can be seen in Figure 7. The results demonstrate that pure C++ implementation of the proposed approach without any processor-specific optimization gives fair real-time performance (~20 fps), the approach has superior quality and in contrast to the other approach it also inherits dis-occlusion detection. Otherwise, the disocclusion handling would require additional re-sampling passes of projected data thus slowing the overall performance.

5. CONCLUSIONS

We have proposed a layered resampling approach for free-viewpoint rendering from view-plus-depth data suitable for general rendering scenario of asymmetric camera parameters in terms of f, $\mathbf{R}$, $\mathbf{t}$, and varying sensor spatial resolutions. The main benefits of the proposed method can be summarized as follows: a high quality texture resampling, avoiding non-uniform to uniform resampling, on-the-fly dis-occlusion detection and z-ordering. The speed of proposed method has similar processing time compared to some other 2D image resampling methods, and is attractive for CPU-based applications. The method is limited to relatively small free-view shifts (e.g. <0.25m). For larger shifts, the number of layers increases and more pixels are treated as boundary conditions, which degrades the performance, which

becomes comparable to the performance of other simpler approaches.

6. REFERENCES

[1] A. Smolic, "3D video and free viewpoint video," *in J. of Patt. Recogn*, vol. (44)9, pp. 1958-1968, 2011.

[2] Dimenco Displays, "3D Content Creation Guidelines,"; "3D Interface Specification," www.dimenco.eu (2011).

[3] Sankaran H., Georgiev M., Gotchev A., Egiazarian K., "Non-uniform to uniform image resampling utilizing a 2D farrow structure," *Proc. of SMMSP*, 2007.

[4] A. Chuchvara, M. Georgiev, A. Gotchev, "A speed-optimized RGB-Z capture system with improved denoising capabilities," Proc. SPIE 9019, Image Processing: Algorithms and Systems XII, March, 2014.

[5] Mark Segal, et al. Fast shadows and lighting effects using texture mapping. In Proceedings of SIGGRAPH '92, pages 249- 252, 1992

[6] J. Pearson, M. Brookes, P-L. Dragotti, "Plenoptic layer-based modelling for image based rendering," *IEEE Transactions on Image Processing*, vol. 22, pp.1-15, 2013.

[7] W. Wang, K. Li, X. Zheng, E. Wu, "Layered Textures for Image-Based Rendering," *Journal Computer Science Technologies,* vol. 19(5), pp. 633-642, 2004.

[8] C. L. Zitnick, S. B. Kang, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," In *SIGGRAPH*, pp. 600–608, August, 2004.

[9] R. Szeliski, "Computer Vision: Algorithms and Applications", Springer, 2010.

[10] Blender Foundation, "Free and open source 3D animation suite," 3D rendering software in www.blender.org.

[11] R. Keys, "Cubic convolution interpolation for digital image processing," Tran. on Signal Processing, Acoustics, Speech, and Signal Processing 29(6), 1153–1160, 1981.

[12] R. Hartley, A. Zisserman, "Multiple-view geometry in computer vision, 2nd Edition", in book of Cambridge University Press, ISBN 0-521-54051-8, 2003.

[13] A. Kolb, E. Barth, R. Koch, R. Larsen, "Time-of-flight cameras in computer graphics," *Computer Graphics Forum* 29(1), pp. 141-159, 2010.

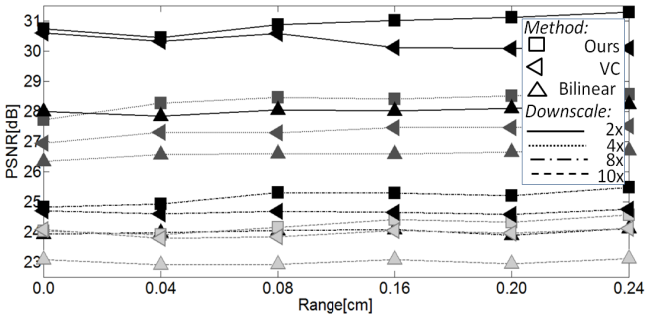

Figure 5.Performance comparison for asymmetric camera setup between proposed (Ours), general re-sampling approach (Bilinear) and virtual camera inter-view rendering approach (VC).

Figure 6.Visual results on scene details for 4x asymmetric downscale and 0.25m setup misalignment of methods(by columns):a) Bilinear, b) VC, c) proposed, d) GT

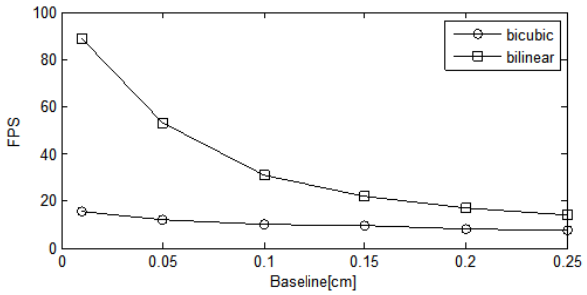

Figure 7.Performance results of the method for bilinear and bicubic interpolation kernel.