MAKING OF "WHO CARES?" HD STEREOSCOPIC FREE VIEWPOINT VIDEO

C. Lipski[1], F.Klose[1], K.Ruhl[1], M.Magnor[1]

[1]{lipski,linz,magnor}@cg.cs.tu-bs.de, Computer Graphics Lab, TU Braunschweig

Abstract

We present a detailed blueprint of our stereoscopic free-viewpoint video system. Using unsynchronized footage as input, we can render virtual camera paths in the post-production stage. The movement of the virtual camera also extends to the temporal domain, so that slow-motion and freeze-and-rotate shots are possible. As a proof-of-concept, a full length stereoscopic HD music video has been produced using our approach.

Keywords:

1 Introduction

The popularity of stereoscopic 3D content in movies and television creates a huge demand for new content. Three major methodologies for creating stereoscopic material are currently employed: purely digital content creation for animation films and games, content filmed with specialized stereoscopic cameras and the artificial enhancement of monocular content.

For animated films the creation of a stereoscopic second view is straightforward. Since the first camera is purely virtual, creating a second virtual camera poses no problem. The major drawback is that the creation of naturalistic real world images is extremely complex and time consuming.

The enhancement of monocular recordings suffers from a similar problem. Although the recorded footage in this case has the desired natural look, the creation of a proxy geometry or a scene model can be tedious. The depth map or proxy geometry used to synthesize a second viewpoint has to be created by skilled artists [8]. The complexity of the model creation directly depends on the complexity of the recorded scene.

While directly recording with a stereoscopic camera rig eliminates the need to create an additional scene model, it requires highly specialized and therefore expensive stereo-camera hardware. Leaving aside monetary constraints, the on set handling of the stereoscopic cameras poses a challenge. The view and baseline selection for example requires careful planning to give the viewer a pleasing stereoscopic experience. Changing the parameters in post production is difficult or even impossible.

Figure 1: "Who Cares?" set. Eleven HD camcorders captured various graffiti motifs. With our approach we are able to synthesize novel spatial and temporal in-between viewpoints.

We examine the stereoscopic content creation from a purely image-based-rendering perspective. Using only recorded images, a naturalistic image impression can be achieved without the need for manual modeling. Using multi-view datasets of only a few cameras, it becomes possible to interpolate arbitrary views on the input camera manifold. Thereby it is not only possible to create stereoscopic 3D views of an arbitrary scene, but also the control over stereo parameters such as baseline and convergence plane is kept during post-production. Since our image-based stereoscopic free-viewpoint video framework is capable of time and space interpolation, it combines the natural image impression from direct stereoscopic recording with the full viewpoint and time control of digitally created scenes.

In order to demonstrate the level of maturity of our approach, we incorporated our free-viewpoint framework into an actual post-production pipeline of a full-length music video, cf. Fig. 1. The goal of this project is to test how well our approach integrates into existing post-production pipelines and to explore how movie makers can benefit from image-based free-viewpoint technology.

In Sect. 2 we investigate the state-of-the art in free-viewpoint video and stereoscopy. In Sect. 3 we will discuss two different approaches for stereoscopic free-viewpoint video before we present our actual post-production pipeline in Sect. 4. Some exemplar results are provided in Sect. 5 before we conclude in Sect. 6.

2 Related Work

Since its invention in 1838, stereoscopy has been widely used in photography and film making industry. It has recently received renewed attention, partly because sophisticated stereoscopic equipment became available for the consumer market. Although the basic principle of stereoscopic image acquisition seems quite simple, many pitfalls exist that make the capture and edit of stereoscopic content a tedious task. Several books and articles [18, 4, 27] have been published that describe the different aspects of how to create pleasing stereoscopic 3D content. These aspects include perceptual factors like minimizing the viewer discomfort by window violations or objects that are far from screen and therefore hard to focus. These content related choices traditionally have to be made at capture time, some tasks can be done using post-production tools, eg. adapting the depth range in different scenes to manipulate 3D experience.

Typical stereoscopic editing task are image rectification, color balancing, disparity remapping and baseline editing. The latter one is especially interesting for our approach, since multi-view recordings often feature wide baselines and conversion to stereoscopic output material is not straight-forward.

Methods to synthesize new views often consist of two basic steps[19]. First a disparity estimation and then the synthesis of the new view. Since disparity estimation is often error-prone, Devernay et al. [6] proposed a novel view synthesis for altering interocular distance with on-the-fly artifact detection and removal.

Disparity remapping also recently received considerable attention: Non-linear disparity mapping operators to alter perceived scene depth, necessary for content adaption to different viewing geometries have been proposed [13]. Targeting the same application, Devernay et al. [5] proposed a disparity-remapping scheme that does not distort objects with respect to perceived depth. To aid the stereographer with the capture on the set, Zilly et al. [29] presented the Stereoscopic analyzer, a tool for online validation of stereoscopic capture, including image rectification, detection of window violations, and optimal interocular distance proposal. Most similar to our proposed approach to stereoscopic content creation is the work of Guillemaut et al. [10]. They reconstruct a high-quality scene geometry from wide-baseline multi-view footage and use this geometry for stereoscopic view synthesis.

Free-Viewpoint Video Systems. Free viewpoint video systems render novel views from multi-video data. Although many approaches with convincing results have been proposed so far, rendering of stereoscopic views remains an open problem.

The first category of free-viewpoint video systems relies on a geometric reconstruction of the scene. Although stereoscopic rendering is straightforward if the scene geometry is known, they all suffer from typical drawbacks of geometry-based systems: Zitnick et al. [30] presented a system for view interpolation from synchronously captured multi-view video data. Unfortunately, time interpolation is not possible with their approach and cameras have to be densely spaced. De Aguiar et al. [3] presented a high-quality performance-capturing that requires the exact knowledge of the 3D geometry of the performing actor. Eisemann et al. [7] showed that misaligned reprojected textures can be corrected on-the-fly with image-based techniques. However, they assume that the overall object silhouette is faithfully preserved. Scene flow estimation enables the reconstruction of object movement [25]. Klose et al. [11] designed a scene flow reconstruction that is able to cope with unsynchronized multi-view recordings. However, their estimation produces only quasi-dense information and does not recover a valid model in textureless regions.

Recently, image-based approaches have been introduced to the community that circumvent the problems of geometry reconstruction. Germann et al. [9] represent soccer players as a collection of articulated billboards. Their approach is restricted to scenes with background that have known geometry and color distributions, e.g., soccer stadiums. Ballan et al. [1] presented an image-based view interpolation that uses billboards to represent a moving actor. Although they produce good results for a single rendered viewport, their billboard-based technique is not suitable for stereoscopic rendering. Zhang et al. [28] showed a system that can infer per frame depth maps by structure from motion techniques and is then able to create video special effects such as depth of field or time-freeze effects. However, due to the structure from motion technique, the recording camera has always to be in motion for the algorithm to work.

Lipski et al. proposed an image-based free-viewpoint framework [15] that is based upon multi-image-morphing. They accept unsynchronized multi-view recordings as their input and interpolate both viewpoints as well as time. Their approach has also been extended to produce a variety of visual effects [14]. We adapted their approach to fit into a post-production pipeline. We also extend their framework to render stereoscopic output as well as disparity-based effects.

3 Two approaches to stereoscopic free-viewpoint video

Before we dive into the details of our free-viewpoint rendering approach, we would like to evaluate the possibilities to create stereoscopic content from within a free-viewpoint system. Although we chose to incorporate the approach of Lipski et al. [15], the following discussion is applicable to most image-based free-viewpoint video approach.

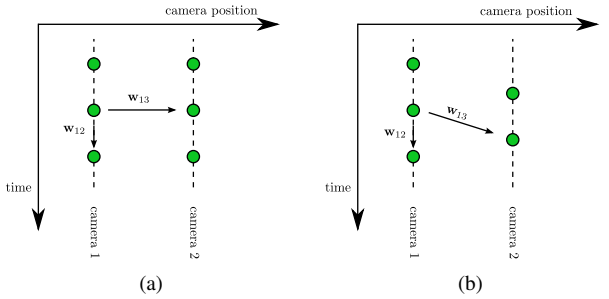


Figure 2: The layout of images shown as green dots in the space-time plane with different camera configurations: (a) synchronized cameras (b) unsynchronized cameras

Since it is important to understand that all required spatial information for stereoscopic rendering is already implicitly available, we will investigate the correspondence fields used in our approach.

We then present two possible ways to render stereoscopic video and juxtapose their benefits.

3.1 Correspondence Fields

To be able to create image-based stereoscopic images, it is important to get an insight into the nature of the information contained in the correspondence fields. A correspondence field is a dense vector field $\mathbf{w}_{ij}$ directed from source image I_i to a destination image I_j. One option of creating those correspondence maps is the application of optical flow algorithms to the source and destination image.

In order to get a clear understanding of the information encoded within $\mathbf{w}_{ij}$, we assume a two camera setup. Figure 2 shows such a simplified setup where the cameras are restricted to a 1D movement along the horizontal axis. The vertical axis is the time and each green dot corresponds to an image acquired at that specific time and place. Dotted lines connect images taken by the same camera. In the following we investigate the information contained in the correspondence fields $\mathbf{w}_{12}$ and $\mathbf{w}_{13}$ when different constraints are posed on the recording modalities.

We start with the most restrictive camera setup with static cameras and synchronized camera shutters as shown in Fig. 2(a). The images are acquired on an axis aligned regular grid in the space-time plane. The correspondence field within the first camera $\mathbf{w}_{12}$ links two consecutive images of the video stream. Since the viewpoint does not change all image changes encoded within $\mathbf{w}_{12}$ represent motion of objects within the scene. In contrast the correspondence field $\mathbf{w}_{13}$ linking two adjacent cameras contains no object movement at all. Source and destination image have been acquired at the same point in time. All changes along $\mathbf{w}_{13}$ can be explained by the motion parallax due to the changed viewpoint between cameras. If the source and destination image of $\mathbf{w}_{13}$ were rectified, the correspondence field would amount to a disparity map.

When the recording constraints are relaxed to a setup where the camera shutters are no longer synchronized, the interpretation of $\mathbf{w}_{13}$ changes. While the intra camera correspondence field $\mathbf{w}_{12}$ still encodes only object motion, the inter camera correspondence field $\mathbf{w}_{13}$ is no longer horizontally aligned with the time axis. In the space-time plane this amounts to a sheering of the two camera paths as depicted in Fig. 2(b). As a result of the unsynchronized shutters the objects in a dynamic scene can move between the acquisition times and the variation between source and destination image is no longer only defined by the view point change.

Inherent to our model is the restriction to linear motion. This affects the object motion within the scene, because a correspondence field defines a line along which each point can move on the image plane.

3.2 Direct Stereoscopic Virtual View Synthesis

In this subsection, we recapitulate image-based virtual view synthesis and show how to use them to synthesize a stereoscopic image pair directly.

Following Lipski et al. [15], we embed all recorded images into a three-dimensional navigation space $\mathcal{N}$. The three dimensions represent two spatial degrees of (horizonatal/vertical movement of th camera) and the temporal dimension (recording time). This 3D point cloud is further partitioned into tetrahedra, so that every novel viewpoint lies exactly in one tetrahedron. We synthesize the left view $I_v^L(\varphi, \theta, t)$ for every point $\mathbf{v}$ inside the recording hull of the navigation space $\mathcal{N}$ by multi-image interpolation:

$$I_v^L(\varphi, \theta, t) = \sum_{i=1}^{4} \mu_i \tilde{I}_i, \tag{1}$$

where

$$\tilde{I}_i \left(\Pi_i \mathbf{x} + \sum_{j=1,\ldots,4, j\neq i} \mu_j (\Pi_j (\mathbf{x} + \mathbf{w}_{ij}(\mathbf{x})) - \Pi_i \mathbf{x}) \right) = I_i(\mathbf{x}) \tag{2}$$

are the forward-warped images [17] of the enclosing tetrahedron. $\{\Pi_i\}$ defines a set of re-projection matrices $\{\Pi_i\}$ that map each image I_i onto the image plane of $I_v^L(\varphi, \theta, t)$, as proposed by Seitz and Dyer [22]. Those matrices can be easily derived from camera calibration. Since the virtual image $I_v^L(\varphi, \theta, t)$ is always oriented towards the center of the scene, this re-projection corrects the skew of optical axes potentially introduced by our loose camera setup and also accounts for jittering introduced by dynamic cameras. Image re-projection is done on the GPU without image data resampling. As proposed by Stich et al. [24], disocclusion is detected on-the-fly by calculating local divergence in the correspondence fields. In contrast to their simple occlusion heuristic, we determine a geometrically valid disparity map. When applying the forward warp in a vertex shader program, we also determine where a given vertex would be warped to in the right eye's view. By subtracting both values, we derive

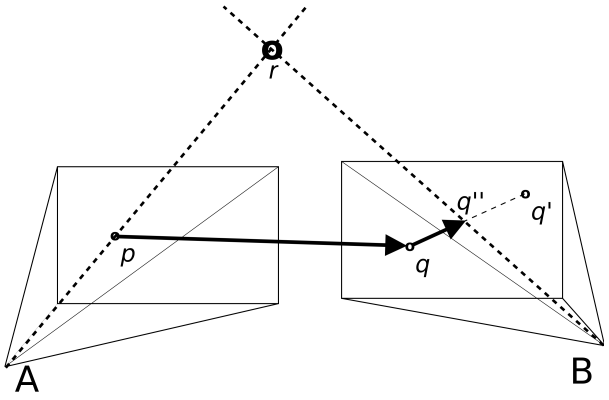


Figure 3: Per-pixel depth estimation. When computing the depth of point p in image A, we first determine the corresponding position in image B. let us assume that we know the corresponding pixel locations q of p in camera B. If p was observed at time t_p, q was observed at t_q and $t_p \neq t_q$, we have to exploit the temporal correspondeces: If we further captured the corresponding point q' at time t_{q+1} and $t_q < t_p < t_{q+1}$, we can estimate the corresponding point q''. We least-squares determine the point r that reprojects into both p and q''. The depth at p is set to the actual depth of r in A.

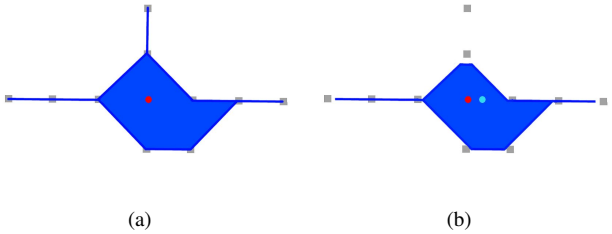


Figure 4: Navigation space boundaries, original camera positions are shown in grey: (a) when a single image (red) is rendered, the full volume (blue) can be accessed. Please note that for the horizontal and vertical arcs (left, top, right) only purely horizontal or vertical camera movement is feasible. Otherwise, error-prone long-range image correspondences would have to be used. (b) When an actual camera pair (red/cyan) is rendered, the navigation space volume is effectively eroded, since there has to be a minimum horizontal distance between the both views. This effect prohibits stereoscopic view interpolation if cameras are placed along an vertical arc (b, top).

a per-vertex disparity. This is interpolated in the fragment shader. Using this disparity value, occlusions can now be resolved by a simple depth buffer comparison. If necessary, the exact disparity of a parallel camera setup can be computed by applying the identical post-warping re-projection matrices. The view for the right eye $I_v^R(\varphi + \Delta, \theta, t)$ is synthesized similar to Eq. (1) by offsetting the camera position along the φ-direction. A common rule for stereoscopic capture is that the maximal angle of divergence between the stereo camera axes should not exceed 1.5 degrees. Otherwise, the eyes are forced to diverge to bring distant objects in alignment which usually causes discomfort. By nature of construction, our approach renders converging stereo pairs and angles of divergence between 0.5 and 1.5 degrees give the most pleasing stereoscopic results.

3.3 Depth-image Based Rendering

An alternative approach is to apply Depth-Image based Rendering (DIBR) as an intermediate step. Instead of rendering a left and right eye view as described in Sect. 3.2, only a center camera view is rendered. In addition, a per-image depth map is obtained. As described in Sect. 3.1, the spatial relationship between corresponding pixels is encoded in the correspondence maps. By assuming linear object motion we can determine where an object is located in two different views at the same point in time.

For every pixel, we determine its position in a neighboring view. If the two cameras did not capture the images synchronously, we determine the actual position of the object by assuming linear motion, cf. Fig. 3. We least-squares determine the three-dimensional location of this point in Euclidean space. We reproject this point into the original view and store the depth value. Similar to the disparity values introduced in Sect. 3.2, we can use the depth maps during the following image interpolation phase to resolve occlusion ambiguities.

It is also possible to render depth maps from novel views. Instead of interpolating RGB information for every pixel, we interpolate the Euclidean 3D position of each pixel. The location of each pixel is computed on-the-fly in the vertex shader using the depth map and the camera matrix. The final depth value is obtained by reprojecting the 3D position into the coordinate space of the synthetic virtual camera. The color images and depth maps are used to render both a left and a right eye view. This is done by using the naive Background Extrapolation Depth-Image Based Rendering techniques [20].

3.4 Comparison

Since both approaches seem appealing at first sight, we would like to compare the individual benefits.

The most striking advantage of the first approach is that it does not require any additional building blocks in the production pipeline. Basically, the scene is rendered twice with slightly modified view parameters. Since there is no second rendering pass (as there is in the Depth-Image Based method), no additional resampling or occlusion handling issues arise.

The most valuable advantage of the second is that no restriction is made to the camera placement. When using the direct approach, the two virtual views must fit into the navigation space boundaries, prohibiting a vertical placement of camera. In contrast, the DIBR method allows stereoscopic rendering even if the cameras are aligned in a vertical arc., cf. Fig. 4. In

addition, the whole navigation space can be used: While the DIBR method allows a placement of the center camera to the far left and far right of the navigation space, the direct method is restricted by the placement of the left or right eye view, cf. Fig. 4. Another benefit becomes obvious when the linear approximation of object movement is considered. Non-linear object motion will lead to unwanted vertical disparity when the direct method is applied. Although non-linear motion might lead to errors in the depth estimation, the DIBR rendering scheme prevents vertical disparity at the rendering stage. The additional re-rendering of the material (color images and depth maps are turned into a left eye and a right eye view) might seem as a disadvantage as first. This notion changes if the material is in some way edited between these two rendering passes. When any kind of image/video editing operation is applied to the center image, e.g., correction of rendering artifacts, it is automatically propagated to the left/right eye view. In the case of the direct method any post-production operations would have to be applied individually (and possibly incoherently) to both views.

Since our project required a (partially) vertical camera layout and the possibility to edit the rendered material, we decided to use the Depth-Image Based Rendering approach. During our evaluation we also created some interesting preliminary results with the direct method which we present in Sect. 5.

4 "Who Cares?" Post-Production Pipeline

In this section we intend to give a hands-on description of our actual production pipeline, cf. Fig. 5.

In the "Who Cares?" project, we process HD input material featuring two timelines (live foreground action and background timelapse) captured with non-synchronized camcorders. To our knowledge it is the first project that makes use of this input material and allows horizontal, vertical and temporal image interpolation. The basic idea of the video is that two DJs appear on a stage and perform a live act with their audio controller. During the course of the music video, the background is painted over with various graffiti motifs (e.g., giant stereo speakers, equalizer bars or a night skyline). Although it would be possible to create and animate these graffiti motifs with traditional CGI tools, we decided to use an actual graffiti timelapse to maintain a credible "low-fi" look. We embedded our stereoscopic free-viewpoint video into a traditional 2D post-production timeline. By collaborating with independent film makers that were familiar with traditional 2D content post-production, we could explore the possibility to seamlessly integrate our system into an existing pipeline.

4.1 Input material and Initial Processing

The first deliverable in our project was the script that defined a rough mapping between the different graffiti motifs in the background and a basic choreography for the actors in the foreground. In addition, the approximate position and movement of the virtual camera was sketched, cf. Fig. 5 (top). The basic movement of the camera was either left-to-right or up-and-down. In some parts of the video, transitions between these two predominant moves were inserted. On some occasions, more complex two-dimensional fly-throughs of the virtual cameras were needed (e.g., the timefreeze scene included in this submission).

For the recordings, we used a custom-built capturing software than ran on four Linux PCs and controlled 11 Canon XHA-1 HD camcorders at 1440x1080p, 25 fps. Although the capture PCs were interconnected via ethernet, the dv1394 camera interface did not allow an accurate synchronization of the cameras.

The foreground action was shot in front of an all-green background. Later, chroma-keying was used to extract an RGBA representation as well as a "shadow layer" of the foreground using Adobe After Effects and Keylight. The shadow information was obtained by investigating brightness differences in the background areas between the actual shots and a clean-plate version of the background.

After completion of the foreground capture, the graffiti timelapse was recorded over a period of five days. The capturing software was set to a timelapse mode and unwanted frames were later removed manually. Using conventional 2D animation techniques, some animation effects, e.g. pumping equalizer bars or vibrating boombox speakers were later inserted into the footage.

Although manual white-balance was applied to all cameras, some color balance differences were observed in the material. These were corrected in Adobe After Effects.

4.2 Static Background

First, we used an off-the-shelf structure-from-motion system [23] to obtain the intrinsic and extrinsic parameters of the cameras. To improve the calibration, we modified the SIFT feature matching scheme so that features of different graffiti motifs were used for the same calibration run. As described in [15], we used the Euclidean 3D positions of the cameras to parametrize their horizontal and vertical position in our navigation space $\mathcal{N}$. We imported the resulting camera parameters and the sparse reconstruction of the scene geometry into blender [2]. We fitted the scene geometry manually to the reconstructed point cloud. Although methods exists to perform this task automatically, e.g., [21], the simplicity of the scene geometry made the manual approach feasible.

The camera calibration together with the reconstructed scene geometry allowed us to compute depth maps for each view and also correspondence maps between the different camera positions. Using OpenGL and GLSL shaders, we rendered both correspondence and depth maps.

4.3 Dynamic Foreground

The calibration data obtained in Sect. 4.2 together with the foreground footage allowed us to to turn the roughly defined camera movement from the script into a valid spacetime

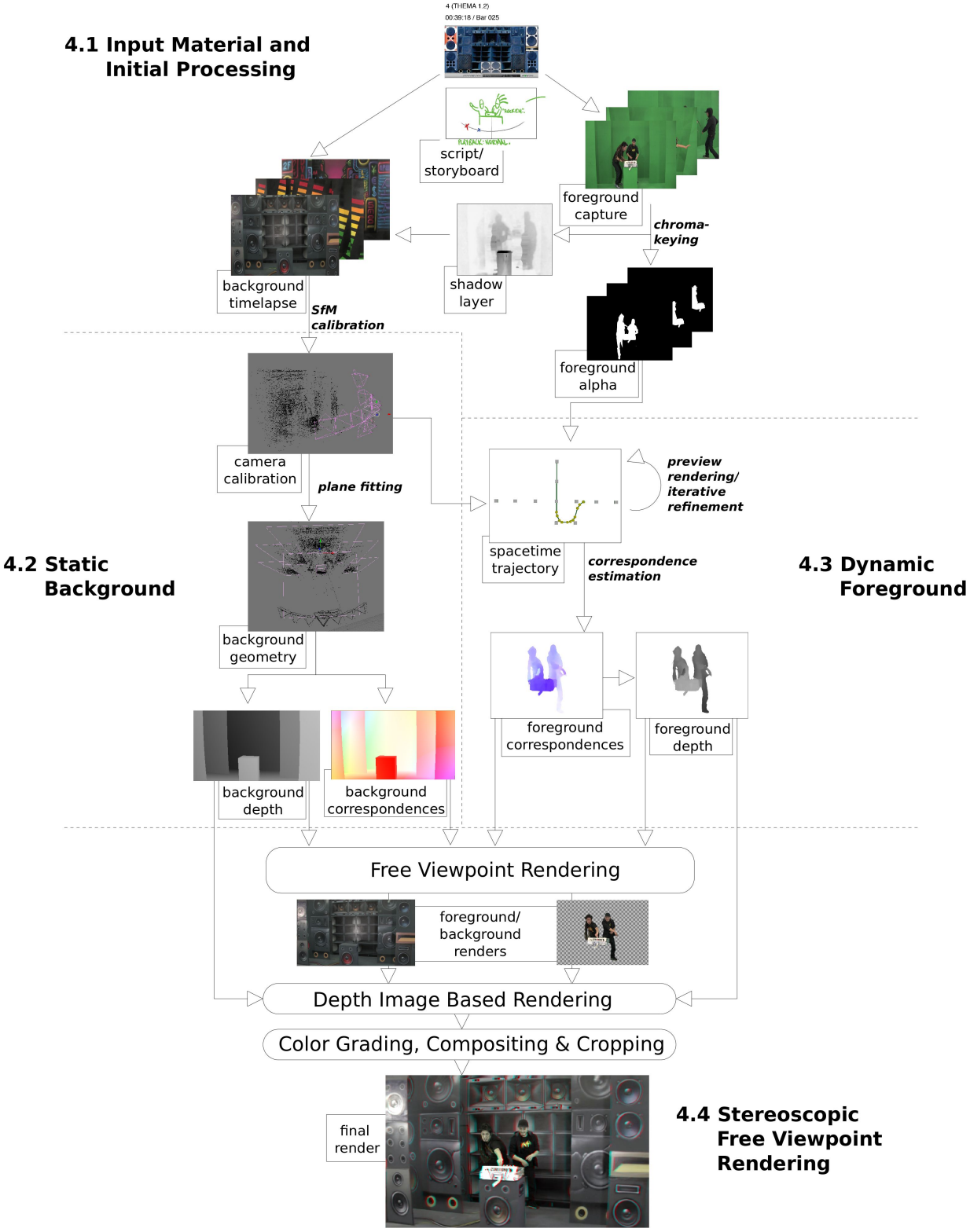


Figure 5: An overview of the production pipeline. After an initial processing stage (top), foreground and background footage are processed independently. While background depth and correspondence maps are derived from a static geometric model (left), foreground data is computed using state-of-the-art correspondence estimation algorithms (right). The processed footage is passed to a free-viewpoint renderer before the actual left and right eye view are rendered using Depth-Image Based Rendering (bottom).

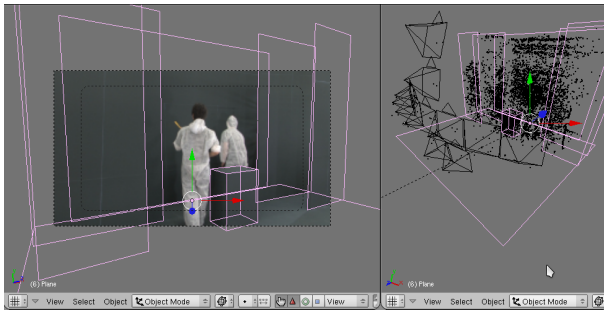


Figure 6: Fitting a geometric model to the static background. The camera orientations and a sparse scene geometry are imported into blender (right). Single quads are fitted manually to the point cloud. Using the actual footage as reference (left), a pixel-exact geometric proxy consisting of 10 faces is created.

trajectory. Using a graphic trajectory editor, key points in navigation space $\mathcal{N}$ were mapped to every dedicated frame in the final video. For in-between frames, linear interpolation or Catmull-Rom splines were used. Since the free-viewpoint framework used at the core of our system allowed to change the trajectory freely in post-production, many variations of the camera trajectory could be explored. In order to get live feedback, we used low-res, foreground-only preview renderings. Since no high-quality correspondence maps of the foreground were available at this stage of the production pipeline, we used the real-time optical flow by Werlberger et al.[26].

After the final trajectory was chosen, high-quality correspondence maps were computed using the approach by Lipski et al. [16]. We used a 30-core Linux cluster to obtain the final maps. In order to increase computation speed we did only compute correspondence values on pixels with non-zero alpha values. Spurious mismatches between images were corrected with the tool presented by Klose et al. [12].

Afterwards, a depth map was computed for every image needed for rendering, cf. Sect. 3.3.

If more than one neighboring view was available, we averaged the depth values. Since the resulting disparity did not exceed a few dozen pixels, byte-precision depth maps proved to be sufficiently accurate.

4.4 Stereoscopic Free Viewpoint Rendering

Incorporating the original foreground and background images, the correspondence and depth maps as well as the space-time trajectory, we rendered both color and depth maps for foreground and background. We used GIMP to manually correct rendering artifacts, e.g. streaking and ghosting, in some images.

Finally, Depth-Image Based Rendering was used to obtain a left and right eye view for each frame.

As a last step, foreground, shadow layer and background were composited and a final color grading was applied. Since rendering artifacts typically occur at the image borders, we cropped these regions, 10% of image width and height were discarded. We also downsampled the final video to 720p to account for the slight blurring introduced by the forward-warping-and-blending scheme of the renderer.

5 Results

Using our stereoscopic free-viewpoint video framework, we rendered a 3-minute music video at a resolution 1280x720px, 25p, cf. Fig. 7 for a selection of renderings. A total of more than 3000 bidirectional correspondence maps had to be computed, each one took about 10 minutes on a single core. The final video is available online at `http://graphics.tu-bs.de/projects/vvc`

5.1 Visual Effects and Disparity Effects

In order to show the versatility of our approach, we included some early results on other data sets using the direct stereoscopic view synthesis.

Our system is able to produce disparity-based effects, since a disparity map (cf. Sect. 3.2) or depth map (cf. Sect. 3.3) has been calculated for occlusion handling Together with the output images, it can be used to create visual effects in a second rendering pass, e.g., depth-of-field or fog.

6 Conclusion

We presented a stereoscopic free-viewpoint video production pipeline that can be effectively incorporated into HD content creation. To demonstrate the level of maturity, we used our framework to produce a full-length HD music video.

We plan to further streamline and automatize the process, requiring less manual interaction. Since no temporal consistency is enforced during correspondence estimation, temporal flickering artifacts may manifest which are hard to correct. This issue has also got be be addressed.

7 Acknowledgements

This work has been funded by the European Research Council ERC under contract No.256941 “Reality CG” and by the German Science Foundation, DFG MA 2555/1-3.

References

[1] Luca Ballan, Gabriel J. Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 29(3):87:1–87:11, July 2010.

[2] blenderinstitute. Blender website. `http://www.blender.org/`, July 2011.

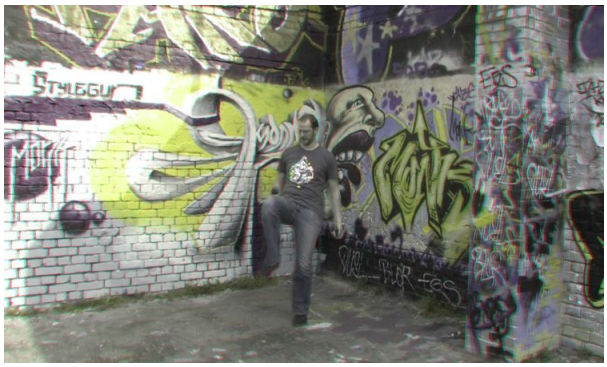
[3] Edilson de Aguiar, Carsten Stoll, Christian Theobalt,

Figure 7: Results: Exemplar stills from the "Who Cares?" music video. Please view with red/cyan (left/right) anaglyph glasses.

(a)

(b)

(c)



(d)

Figure 8: Earlier results obtained with direct stereoscopic view synthesis. Please view with red/cyan (left/right) anaglyph glasses.

Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance Capture from Sparse Multi-View Video. *ACM Trans. on Graphics*, 27(3):1–10, 2008.

[4] Frédéric Devernay and Paul Beardsley. *Image and Geometry Processing for 3-D Cinematography*, volume 5, chapter Stereoscopic cinema, pages 11–51. Springer-Verlag, 2010.

[5] Frédéric Devernay and Sylvain Duchêne. New view synthesis for stereo cinema by hybrid disparity remapping. In *Proc. of IEEE International Conference on Image Processing (ICIP'10)*, pages 5–8, 2010.

[6] Frédéric Devernay and Adrian Ramos Peon. Novel view synthesis for stereoscopic cinema: Detecting and removing artifacts. In *Proc. of ACM Workshop on 3D Video Processing (3DVP'10)*, pages 25–30, 2010.

[7] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum (Proc. Eurographics EG'08)*, 27(2):409–418, 4 2008.

[8] Prime Focus. View-D website. `http://www.primefocusworld.com/services/products/view-d`, August 2011.

[9] Marcel Germann, Alexander Hornung, Richard Keiser, Remo Ziegler, Stephan Würmlin, and Markus Gross. Articulated billboards for video-based rendering. *Comput. Graphics Forum (Proc. Eurographics)*, 29(2):585–594, 2010.

[10] Jean-Yves Guillemaut, Muhammad Sarim, and Adrian Hilton. Stereoscopic content production of complex dynamic scenes using a wide-baseline monoscopic camera set-up. In *Proc. of IEEE International Conference on Image Processing (ICIP'10)*, pages 9–12. IEEE Computer Society, September 2010.

[11] Felix Klose, Christian Lipski, and Marcus Magnor. Reconstructing Shape and Motion from Asynchronous Cameras. In *15th International Workshop on Vision, Modeling and Visualization (VMV)*, pages 171–177, Siegen, Germany, November 2010.

[12] Felix Klose, Kai Ruhl, Christian Lipski, and Marcus Magnor. Flowlab - an interactive tool for editing dense image correspondences. In *Proc. European Conference on Visual Media Production (CVMP) 2011*, 2011.

[13] Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus Gross. Nonlinear disparity mapping for stereoscopic 3d. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 29(3):75:1–75:10, 2010.

[14] Christian Linz, Christian Lipski, Lorenz Rogge, Christian Theobalt, and Marcus Magnor. Space-Time visual effects as a Post-Production process. In *ACM Multimedia 2010 Workshop - 3DVP'10*, pages 1–6, 10 2010.

[15] Christian Lipski, Christian Linz, Kai Berger, Anita Sellent, and Marcus Magnor. Virtual video camera: Image-based viewpoint navigation through space and time. *Computer Graphics Forum*, 29(8):2555–2568, 2010.

[16] Christian Lipski, Christian Linz, Thomas Neumann, and Marcus Magnor. High resolution image correspondences for video Post-Production. In *CVMP 2010*, page to appear, London, 2010.

[17] William Mark, Leonard McMillan, and Gary Bishop. Post-Rendering 3D Warping. In *Proc. of Symposium on Interactive 3D Graphics*, pages 7–16, 1997.

[18] Bernard Mendiburu. *3D Movie Making: Stereoscopic Digital Cinema from Script to Screen*. Focal Press, 2009.

[19] Sammy Rogmans, Jiangbo Lu, Philippe Bekaert, and Gauthier Lafruit. Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity GPUs. *Signal Processing: Image Communication*, 24(1-2):49 – 64, 2009. Special issue on advances in 3DTV and video.

[20] Michael Schmeing and Xiaoyi Jiang. Time-consistency of disocclusion filling algorithms in depth image based rendering. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1–4, 2011.

[21] Christopher Schwartz, Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Photopath: Single image path depictions from multiple photographs. *Journal of WSCG*, 18(1-3), February 2010.

[22] Steven M. Seitz and Charles R. Dyer. View Morphing. In *Proc. of ACM SIGGRAPH'96*, pages 21–30, New York, 1996. ACM Press/ACM SIGGRAPH.

[23] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. on Graphics*, 25(3):835–846, 2006.

[24] Timo Stich, Christian Linz, Christian Wallraven, Douglas Cunningham, and Marcus Magnor. Perception-motivated Interpolation of Image Sequences. In *Proc. of ACM APGV'08*, pages 97–106, Los Angeles, USA, 2008. ACM Press.

[25] S. Vedula, S. Baker, and T. Kanade. Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events. *ACM Trans. on Graphics*, 24(2):240–261, 2005.

[26] Manuel Werlberger, Thomas Pock, and Horst Bischof. Motion estimation with non-local total variation regularization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, June 2010.

[27] Lucy Wilkes. The role of ocula in stereo post production. *The Foundry, Whitepaper*, 2009.

[28] G. Zhang, Z. Dong, J. Jia, L. Wan, T.T. Wong, and H. Bao. Refilming with depth-inferred videos. *IEEE Transactions on Visualization and Computer Graphics*, pages 828–840, 2009.

[29] Frederik Zilly, Marcus Müller, Peter Eisert, and Peter Kauff. The stereoscopic analyzer an image-based assistance tool for stereo shooting and 3d production. In *Proc. of IEEE International Conference on Image Processing (ICIP'10)*, pages 4029–4032. IEEE Computer Society, 2010.

[30] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon A. J. Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.