Hardware-accelerated Dynamic Light Field Rendering

Bastian Goldlücke, Marcus Magnor, Bennett Wilburn*

Max-Planck-Institut für Informatik
Graphics - Optics - Vision
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
Email: bg@mpii.de

Abstract

We present a system capable of interactively displaying a dynamic scene from novel viewpoints by warping and blending images recorded from multiple synchronized video cameras. It is tuned for streamed data and achieves 20 frames per second on modern consumer-class hardware when rendering a 3D movie from an arbitrary eye point within the convex hull of the recording camera's positions.

The quality of the prediction largely depends on the accuracy of the disparity maps which are reconstructed off-line and provided together with the images. We generalize known algorithms for estimating disparities between two images to the case of multiple image streams, aiming at a minimization of warping artifacts and utilization of temporal coherence.

1 Introduction

Three-dimensional television is currently experiencing a surge in research activity [1, 2, 3, 4]. Acquisition hardware, computer vision algorithms, and rendering techniques have reached a level of affordability, robustness and sophistication, respectively, that enable building a system to record, reconstruct, and render real-world dynamic scenes in their three-dimensional nature. The development of 3D-TV follows advances recently made in image-based rendering (IBR). In IBR, conventional photographs are used to capture the visual appearance, the *light field* of a scene. Given enough images from different viewpoints, any view of the scene from outside of the convex hull of the recording positions can be reconstructed [5]. Unfortunately, light field rendering quality depends on the number of photographs. Very large numbers of images are necessary to attain convincing rendering results [6]. However, if 3D scene structure can be reconstructed from the image data, e.g. per-image depth maps [7] or a complete 3D scene geometry model [8], hybrid model/image-based rendering methods achieve realistic rendering results from only a relatively small number of images. Furthermore, programmable graphics hardware can be used to accelerate image-based rendering by warping and resampling the recorded images [9, 10].

Until recently, image-based rendering techniques were restricted to static scenes. To record the dynamic light field of a temporally varying scene, the necessary hardware effort is considerably higher. Multiple synchronized video cameras are needed to capture the scene from different viewpoints. In [11], six cameras are used in conjunction with three PCs to reconstruct approximate depth maps at near-interactive frame rates. An off-line approach based on volumetric reconstruction is presented in [12] where a dynamic voxel model is used to render the scene from novel viewpoints.

In this paper, a rendering system for dynamic light fields is presented. An array of multiple synchronized cameras is used to capture an animated scene from different viewpoints. Depth maps are reconstructed from the recorded video streams. In contrast to [11] we use an off-line algorithm and focus on the hardware accelerated rendering and blending, where we achieve interactive rates of up to 20 frames per second, including frame data loading. The system makes possible interactive viewing of 3D movies from arbitrary viewpoint positions within the window spawned by the camera positions.

* Computer Systems Lab, Stanford University, USA

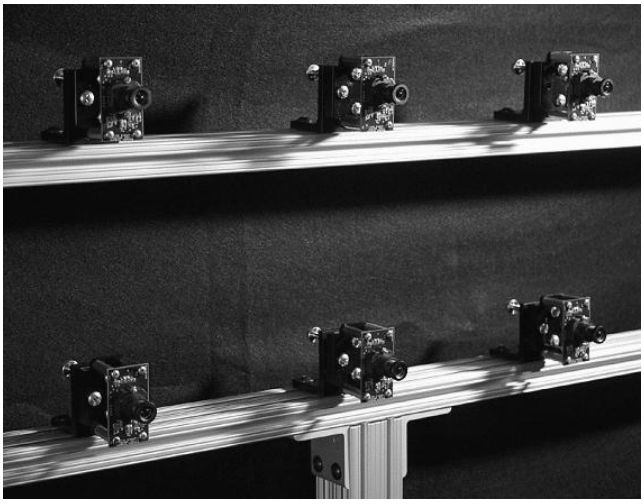

Figure 1: The Stanford Light Field Video Camera consists of numerous CMOS camera heads, arranged in a planar matrix and with aligned optical axes.

2 Dynamic Light Field Acquisition

To record the light field of a static scene, it is customary practice to use one single still-image camera and consecutively record multiple views by moving the camera around the scene. For acquiring the temporally varying light field of a dynamic scene, however, numerous video cameras are needed to record the scene from multiple viewpoints simultaneously. In addition, all video cameras must be synchronized to maintain temporal coherence among the recorded images.

Because of the high hardware requirements, only a few laboratories are currently able to record dense dynamic light fields [1, 3, 4]. The dynamic light field data used in this work has been captured with the Light Field Video Camera (LFVC) [4] which is currently being built at Stanford University as part of the Immersive Television Project, Fig. 1. The LFVC consists of multiple low-cost CMOS imagers, each providing 640×480-pixel RGB resolution at 30 frames per second. The camera heads are aligned in parallel, capturing the scene's light field in the two-plane parameterization [5]. Custom-built driver boards enable on-line pre-processing as well as MPEG-2 compression of each video stream. At 5 MBytes/sec per camera, up to 60 MPEG-encoded video streams can be streamed to one PC via the IEEE1394 High Performance Serial Interface Bus where the data is stored on a SCSI hard drive.

The use of more than one camera inevitably leads to mismatches in hue, brightness and radial distortion among different camera images. These differences need to be minimized by careful calibration prior to further processing. In addition, due to the design of the Light Field Video Camera, only MPEG-compressed image data is available, causing quantization noise and blocking artifacts in the images. The depth estimation algorithm described in the following Section must be robust against these artifacts.

3 Disparity Map Estimation

Let I be an image from a source camera C. For $k = 0, \ldots, n$, let I_k be an image from a reference camera C_k related to the source image by the fundamental matrix $\mathbf{F}_k$. If $P \in \mathbb{R}^3$ is a fixed point in world coordinates and $\mathbf{p}$ its projection in the source image in homogenous coordinates, the corresponding point $\mathbf{p}'_k$ in the reference image I_k lies on the epipolar line of $\mathbf{p}$ satisfying the *epipolar constraint* [13, (1.9)]:

$$\mathbf{p}_k'^T \, \mathbf{F}_k \, \mathbf{p} \;=\; 0. \tag{1}$$

We use the prime to denote points whose coordinates are given in the image coordinate frame of C_k, all others are given in the image coordinate frame of camera C.

If C_k is obtained from C by a pure translation $\mathbf{d}_k$ parallel to the image plane, the fundamental matrix is given by [13, Sect. 1.13]:

$$\mathbf{F}_k \;=\; [\mathbf{d}_k]_\times \;=\; \begin{bmatrix} 0 & 0 & d_{k,y} \\ 0 & 0 & -d_{k,x} \\ -d_{k,y} & d_{k,x} & 0 \end{bmatrix}$$

and the epipolar constraint (1) yields an equation for a line which can be rewritten as

$$\mathbf{p}'_k \;=\; \mathbf{p} \;-\; \lambda \mathbf{d}_k. \tag{2}$$

Note that $\mathbf{d}_k$ is not normalized. Here $\lambda = \lambda(P) \geq 0$ is called the *disparity of* P and can be interpreted as the parallax of $\mathbf{p}$ for a unit camera movement on the eye point plane. We also note that it is a bijective function of the *depth* $\delta(P)$, the distance of P to the image plane: From Fig. 2 we deduce that *in the image coordinate frame of camera C,* $\mathbf{p}_k$ has coordinates

$$\mathbf{p}_k \;=\; \mathbf{p} + \frac{\delta(P)}{\delta(P) + f} \, \mathbf{d}_k, \tag{3}$$

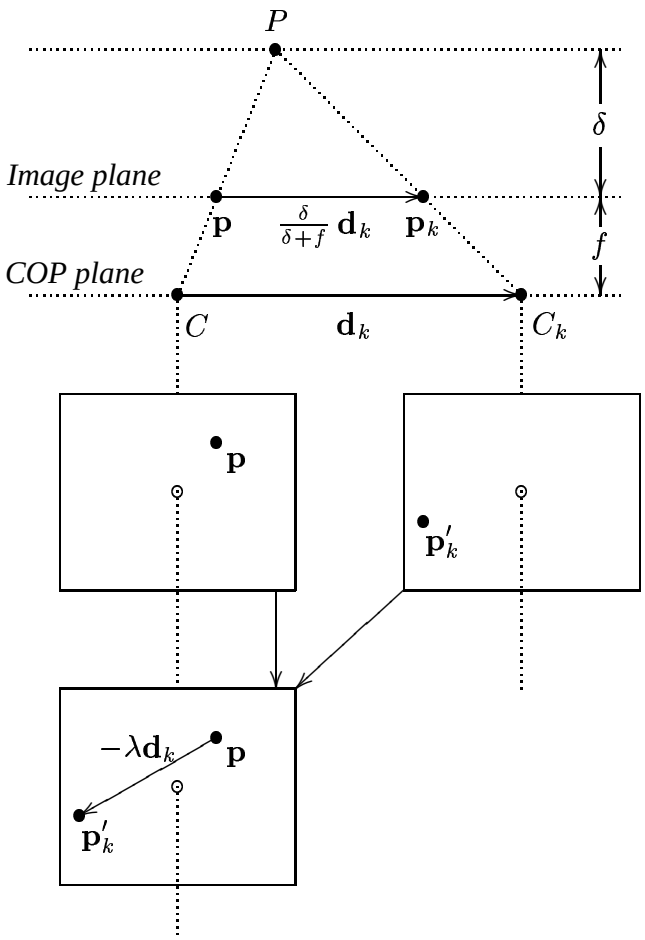

Figure 2: *Top*: Plane spawned in $\mathbb{R}^3$ by a point P and the camera's centers of projection (COP). *Below:* Image coordinate frames of camera C on the left and camera C_k on the right side.

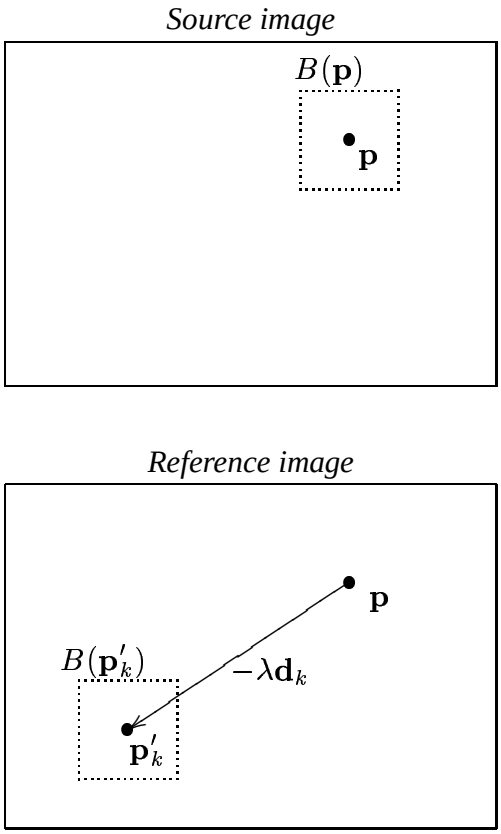

Figure 3: Relation between pixels in source and reference image.

where f denotes the camera's focal length. Since the coordinate frames in the image plane are related by $\mathbf{p}'_k = \mathbf{p}_k - \mathbf{d}_k$ we conclude by comparing (2) with (3) that

$$\lambda = 1 - \frac{\delta(P)}{\delta(P)+f} = \frac{f}{\delta(P)+f}$$

does not depend on the reference image, so it is well-defined. Equation (2) thus ensures that knowledge of λ is sufficient to derive the location of $\mathbf{p}$ in an image taken by a camera related to C by a pure translation parallel to the image plane.

For any point $\mathbf{q}$, let $B(\mathbf{q})$ be a square block around $\mathbf{q}$ of a chosen fixed size. The distance between two blocks is defined as the sum of the absolute differences between the R-, G- and B-values of each pixel. We estimate the disparity λ per-pixel by computing the distance $r_k(\lambda)$ between $B(\mathbf{p})$ and $B(\mathbf{p}_k)$ for all values of $\lambda \in [0, \lambda_{max}]$. The initial disparity estimate is then given by the value of λ where $r_k(\lambda)$ attains its minimum, Fig. 3. However, if we use only this well-known correlation scheme, the result suffers from a number of per-pixel discontinuities caused by false matches, Fig. 4b. These lead to strong artifacts when (2) is used for forward-predictive warping. In order to eliminate outliers and further improve the disparity estimate, we use the technique presented in [14]. In this algorithm, λ is recovered as the asymptotic state of a parabolic differential equation, which can be approximated using an iteration process. We use our initial estimate to precondition this iteration and obtain new disparity maps with the desireable property that per-pixel discontinuities have vanished while discontinuities along image edges are preserved, Fig. 4. Another advantage of this method is that the temporal coherence in image streams can easily be exploited: The disparity map obtained for the current frame is a very good initial estimate for the disparity map of the next frame, so no further computation of correlations is required.

So far, we have only considered a single reference image. In the ideal case this is sufficient, since λ does not depend on k. In reality, we usually obtain different λ_k for the different reference images.

Figure 4: Original image, its correlation-based disparity map and the processed final map.

The discrepancy is due to false matches caused by occlusions as well as quantization noise and blocking artifacts present in the MPEG-encoded image data used. Additionally, the transformation relating the cameras is usually not a perfect translation, and the response to colors may be different for different cameras. The latter problem is alleviated by calibrating each camera's color reproduction using calibration matrices obtained from MacBeth color chart images. In a second step, the images are equalized which greatly improves the quality of the correlation process.

Further errors are reduced by computing the final disparity map λ of the source image I by solving

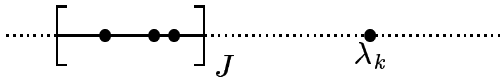

Figure 5: Outlier elimination scheme.

the least-squares problem

$$\min_{\lambda} \sum_{k} (\lambda_k - \lambda)^2 \tag{4}$$

for a selection of different λ_k. The values which go into the computation are chosen according to the following principles:

- If the distance function r_k exceeds a certain threshold value, λ_k is discarded according to the assumption that $\mathbf{p}_k$ is most likely obscured by some other object in the reference image I_k and thus no depth information can be gained from this image.
- To further eliminate outliers, we place an interval J covering a small disparity range on the λ-axis in such a way that it contains the largest possible number of disparity values, Fig. 5. For the computation of (4) we use only the values of λ_k within J.

4 Interactive DLF Rendering

In the rendering step, a number of images $\{I_k\}_{k=1,\ldots,m}$ with precalculated dense disparity maps $\{\lambda_k\}$ are warped and blended in order to predict a view of the scene from a new viewpoint C. For interactive frame rates, one cannot transform each pixel seperately as this would consume too much time. The method described in this section exploits the polygon processing capabilities of OpenGL as well as hardware texturing and blending provided by modern graphics hardware.

We create a regular triangle mesh covering the area of the source image and assign to each vertex $\mathbf{v}$ of the mesh a disparity value $\lambda(\mathbf{v})$ computed as the average of its surrounding pixels in λ_k. This process essentially downscales the disparity maps and reduces the number of transformations required during rendering. The downscaled maps can also be precomputed and stored on hard drive for different resolutions of the triangle mesh. An additional benefit of downscaled disparity maps is their smaller size which speeds up loading while displaying sequences of movie frames.

The blending process requires m passes of the scene. In the kth pass, the source image I_k is loaded

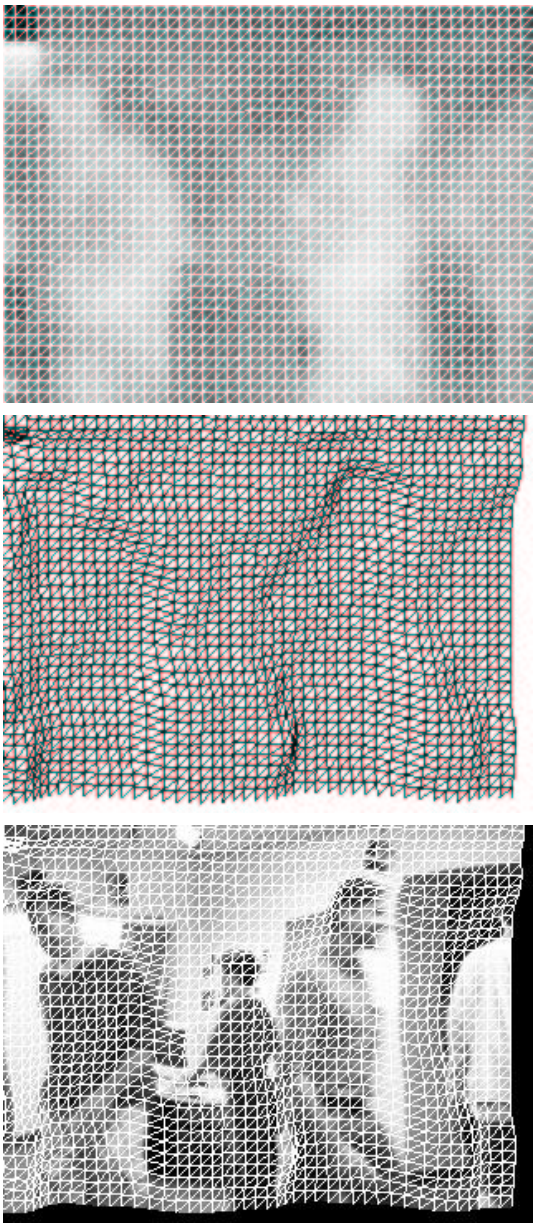

Figure 6: Disparity map with triangle mesh, warped triangle mesh and resulting warped image.

to the texture target TEXTURE_RECTANGLE_NV. This OpenGL extension is required since the width and height of the source images is usually not a power of two. The mesh is rendered as a sequence of triangle strips, which gives far superior performance compared to quads. Final vertex positions and texture coordinates are computed in hardware by a vertex program which performs the following tasks:

- Use the position of the vertex directly as the texture coordinates in the source image. Note that texture coordinates for rectangular textures are not homogenous.
- Compute the position of the vertex $\mathbf{v}$ in the warped image according to

$$\mathbf{v}_{opos} \;=\; \mathbf{v} \;+\; \lambda(\mathbf{v})\,\mathbf{d}_k,$$

  where $\mathbf{d}_k$ is the translation from C_k to the new viewpoint C.

The initial weight for each image is given by

$$\omega_k \;:=\; \exp(-c\,\mathbf{d}_k \cdot \mathbf{d}_k). \qquad (5)$$

ω_k decreases with greater distance to the source image. The use of the Gaussian is not mandatory, it is but one of the functions having the desired property of smoothness and a function value of one when $\mathbf{d}_k$ equals zero. To further speed up rendering, images with a weight below a small treshold value $\epsilon > 0$ are not used since their contribution is too small to be visible. The constant c is chosen so that ω_k falls just below ϵ when $\mathbf{d}_k$ equals the minimum distance between two cameras. Thus, if the position of the camera for the predicted view coincides with one of the source cameras, the original image of the camera is reproduced exactly without distortions from other source images.

After all initial weights are known, the final blending weight w_k used in the OpenGL blend equation is then computed according to a cumulative normalization by

$$w_k \;=\; \frac{\omega_k}{\sum_{i=1}^{k}\,\omega_i}.$$

The stencil buffer is used to ensure that a weight of 1 is used in areas where no image data has yet been written. That way it is guaranteed that all pixels are blended with the correct weight relative to the images already warped and that for each pixel the sum of all weights after every pass is equal to one.

Backfacing triangles are culled during rendering since their pixels are obscured by a nearer object. An example of the original triangle mesh, the depth map and the resulting warped mesh is shown in Fig. 6.

5 Results

The measurements in the following tables were performed on a 1.7GHz Pentium Xeon with an nVidia GeForce 4 graphics card. Four source images with a resolution of 320×240 pixels taken from cameras in the corners of a square are warped together to render a predicted view with a resolution of 640×480 pixels, see the figures on the color plate.

Block size [pixel]	Mesh res. [blocks]	Triangles [#]	Frame rate [Hz]
8	40×30	9600	72.99
4	80×60	38400	32.15
2	160×120	145200	10.87

Table 1: Frame rates for different mesh resolutions without loading times.

Block size	Time per 100 frames used for Rendering	Loading images	Loading depth maps
8	1.37 s 26.4 %	3.68 s 70.8 %	0.15 s 2.8 %
4	3.11 s 42.0 %	3.68 s 49.7 %	0.61 s 8.3 %
2	9.98 s 61.9 %	3.68 s 22.8 %	2.45 s 15.3 %

Table 2: Profile for different tasks while displaying a movie, assuming the theoretical average transfer rate of 25 MByte/s.

Frame rates achieved for a static image with different triangle mesh resolutions are denoted in Table 1, where block size corresponds to triangle leg length in pixel.

In the case of dynamic scenes, the task of loading the images and depth maps becomes the bottleneck, as can be concluded from Table 2. Indeed, at this resolution about 1 MByte of image data and 0.25 MByte of disparity data have to be transferred per frame from hard drive to the graphics card. Modern standard IDE drives achieve average loading rates of 25 MByte per second, which limits the theoretically possible frame rate to 20Hz. In practice, the transfer rate on our system seldom exceeds 15 MByte/s, probably because the different data files to be read are not stored linearly on the drive.

Thanks to the use of graphics hardware for rendering, the predicted image can be scaled to an arbitrary size with no impact on performance. Smaller block sizes result in a more complex triangle mesh and require more bandwidth and rendering time, but improve the quality of the image only marginally. This is shown quantitatively in Table 3, where we predict the view from the top-left camera by warping the images of the other three cameras. The mean squared error per pixel between the original image

Figure 7: Residual error of disparity compensation: The error is concentrated in edges of the image, where large differences in color stem from only small inaccuracies in disparity.

Block size	Root mean squared error
8	16.50
4	16.35
2	16.30

Table 3: Per-pixel error in a view warped from three other images. Pixel values range from 0 to 255.

and the predicted image serves as a measure for the warped image quality. Fig. 7 shows a visualization of the error distribution.

The predicted image in the worst possible case where the camera lies in the center of the square is displayed on the color plate. Some strong blurring artifacts are visible in the areas circled in red. The errors in the upper right corner result from the fact that some part of it is visible in only one image, so no correct depth information can be derived for it. In general, the depth information near the boundary is not as accurate as in central regions, which is a common problem in disparity map estimation [14]. The blurriness in the legs of the person is due to the motion blur already present in the source images, which leads to bad disparity estimates. However, the algorithm reconstructs well features such as the corners in the wall and the computer monitors, circled green. A movie showing our system in action is available for download on our web page[1].

6 Conclusions and Future Work

The system we have presented is capable of rendering 3D movies from an arbitrary viewpoint within

[1] *http://www.mpi-sb.mpg.de/∼bg/3dtv.html*

the recording window at interactive frame rates on today's consumer-class hardware. Image quality largely depends on the accuracy of the disparity maps provided with the recorded video streams. The algorithm we have proposed for off-line reconstruction of these maps is a generalization of known single-reference-image methods to the case of multiple image streams. It minimizes warping artifacts and exploits the temporal coherence of the data.

In our current implementation the predicted view can only be rendered for translated cameras. The correlation algorithm used for preconditioning the disparity map estimation also assumes that the cameras in the array are related to each other by pure translations. Our next goal is to generalize our software towards arbitrary recording geometry and arbitrary positions used for prediction. The additional hardware-accelerated per-vertex computations will not decrease the overall frame rate significantly. Since the real bottleneck is the time needed for data loading, we will investigate compression techniques to speed up the transfer.

We also plan to improve the quality of the prediction by using a triangle mesh adapted to image features instead of a regular mesh. By matching triangle edges with edges in the depth maps, pixels belonging to the same physical object will be bound to an independent subset of the mesh, which further improves rendering quality.

References

[1] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Siggraph 2000, Computer Graphics Proceedings*, Kurt Akeley, Ed. 2000, Annual Conference Series, pp. 369–374, ACM Press / ACM SIGGRAPH / Addison Wesley Longman.

[2] M. Op de Beeck and A. Redert, "Three dimensional video for the home," *Proc. EUROIMAGE International Conference on AUgmented, Virtual Environments and Three-Dimensional Imaging (ICAV3D'01)*, Mykonos, Greece, pp. 188–191, May 2001.

[3] R. Ooi, T. Hamamoto, T. Naemura, and K. Aizawa, "Pixel independent random access image sensor for real time image-based rendering system," *Proc. International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, pp. 193–196, Oct. 2001.

[4] B. Wilburn, M. Smulski, K. Lee, and M. Horowitz, "The light field video camera," *SPIE Proc. Media Processors 2002*, vol. 4674, Jan. 2002.

[5] M. Levoy and P. Hanrahan, "Light field rendering," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, New Orleans, USA, pp. 31–42, Aug. 1996.

[6] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000)*, New Orleans, USA, pp. 307–318, July 2000.

[7] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, New Orleans, USA, pp. 43–54, Aug. 1996.

[8] D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle, "Surface light fields for 3D photography," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000)*, New Orleans, USA, pp. 287–296, July 2000.

[9] P.-P. Sloan, M. Cohen, and S Gortler, "Time critical lumigraph rendering," in *1997 Symposium on Interactive 3D Graphics*, Michael Cohen and David Zeltzer, Eds. ACM SIGGRAPH, Apr. 1997, pp. 17–24, ISBN 0-89791-884-3.

[10] H. Schirmacher, W. Heidrich, and H.-P. Seidel, "High-quality interactive lumigraph rendering through warping," in *Proceedings of Graphics Interface 2000*, 2000, pp. 87–94.

[11] H. Schirmacher, L. Ming, and H.-P. Seidel, "On-the-fly processing of generalized lumigraphs," *Computer Graphics Forum*, vol. 20, pp. C165–C173;C543, 2001.

[12] S. Vedula, S. Baker, and T. Kanade, "Spatio-temporal view interpolation," Tech. Rep. CMU-RI-TR-01-35, Carnegie Mellon University, Sept. 2001.

[13] O. Faugeras and Q.-T. Luong, *The Geometry of Multiple Images*, The MIT Press, Cambridge, MA, 2001.

[14] L. Alvarez, R. Deriche, J. Sánchez, and J. Weickert, "Dense disparity map estimation respecting image discontinuities: A PDE and scale-space based approach," *INRIA Rapport de Recherche, No.3874*, Jan. 2000.

Figure 8: Source images taken by four cameras positioned at the vertices of a square. Note the artifacts circled red caused by motion blur, which lead to blurred reproduction in the predicted view below.

Figure 9: The predicted view from a viewpoint in the center between the four cameras. Note the sharp reproduction of features circled green. The visible blurriness in areas marked red is caused partly by motion blur in the source images and partly by inaccuracies in the depth maps as explained in the main text.