Light Field Compression using Eigen Textures

Marco Volino Armin Mustafa Jean-Yves Guillemaut Adrian Hilton
Centre for Vision, Speech and Signal Processing
University of Surrey, UK
{m.volino,a.mustafa,j.guillemaut,a.hilton}@surrey.ac.uk

Abstract

Light fields are becoming an increasingly popular method of digital content production for visual effects and virtual/augmented reality as they capture a view dependent representation enabling photo realistic rendering over a range of viewpoints. Light field video is generally captured using arrays of cameras resulting in tens to hundreds of images of a scene at each time instance. An open problem is how to efficiently represent the data preserving the view-dependent detail of the surface in such a way that is compact to store and efficient to render. In this paper we show that constructing an Eigen texture basis representation from the light field using an approximate 3D surface reconstruction as a geometric proxy provides a compact representation that maintains view-dependent realism. We demonstrate that the proposed method is able to reduce storage requirements by $> 95\%$ while maintaining the visual quality of the captured data. An efficient view-dependent rendering technique is also proposed which is performed in eigen space allowing smooth continuous viewpoint interpolation through the light field.

1. Introduction

In virtual and augmented reality (VR/AR) there is increasing interest in creating photo-realistic cinematic experiences. The use of games engines and computer generated imagery allow interactivity, such as head movement, but does not achieve cinematic photo-realism, conversely 360 video and stereo capture achieve photo-realism for a fixed location but does not allow head movement or realistic parallax. Light field video capture offers a potential solution capturing a scene with full photo-realism whilst allowing change in viewpoint (head movement) within the camera aperture. However, the data volume associated with light field capture prohibits transmission and rendering on commodity graphics hardware for consumer applications.

In this paper we introduce a compact light field representation that enables up to a 95% decrease in data size and offers efficient rendering for interactive VR/AR viewing on commodity graphics hardware whilst maintaining the visual quality of the captured light field. To summarize, the contributions of this paper are:

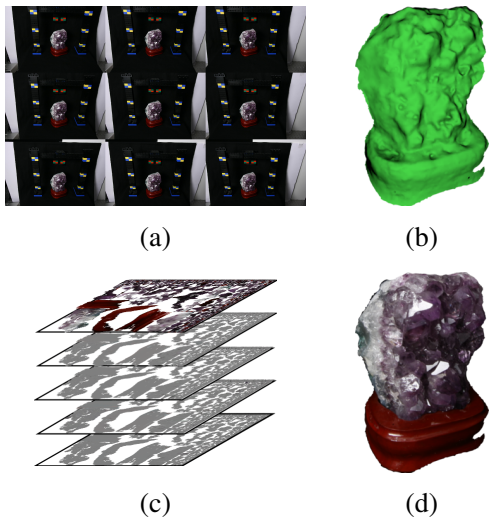


Figure 1: Light field compression using Eigen textures. (a) input light field, (b) scene reconstruction, (c) Eigen textures and (d) view-dependent render.

1. A novel Eigen texture representation for light fields which is compact and preserves the view-dependent photo-realism of the captured light field.

2. Efficient light field rendering and synthesis of novel views by interpolation in Eigen space to achieve photo realistic rendering with real-time interactive performance on a commodity graphics platform for interactive virtual and augmented reality applications.

We evaluate on public light field datasets of static scenes and sparse light field video frames. The approach is shown to reduce storage requirements by $> 95\%$ whilst maintaining the perceptual quality of the light field image.

2. Related Work

2.1. Light Fields

The origins of the light field can be traced back to the notes of Leonardo Da Vinci in which he postulated that the view of the world from any point in space is formed by the intersection of an infinite number of *radiant pyramids* from all directions [13]. This idea was later formalized by Adelson and Bergen [1] and became known as the plenoptic function, a seven dimensional function of 3D position, 2D viewing direction, observed wavelength, and time. The pioneering works of Levoy and Hanrahan [14] and Gortler *et al*. [10] showed that the plenoptic function could be reduced to four dimensions under the assumption that an object is observed from outside its convex hull and the object remained static. This allowed light fields to be represented by the intersection of light rays travelling between two planes. These assumptions reduced the dimensionality and inspired the design of light field acquisition hardware.

Light fields can be captured by a single camera with an array of micro lenses placed in front of a conventional image sensor [19] (e.g. Lytro Illum and Raytrix cameras), a single camera on a gantry [28,30], or using an array of multiple cameras [27]. Micro lens arrays trade off spatial resolution of the imaging sensor to angular resolution of the lens array. The range that can be rendered from a light field camera image is limited by the aperture of the camera, hence grids of cameras allow capture and rendering of intermediate viewpoints between the outer limits of the array. Recently, arrays of cameras have been used to capture light field video consisting of grids of 200-300 cameras [27]. Whilst this enables photo realistic rendering of dynamic scenes, the volume of data in a single light field image is 200-300 times that of a single conventional image resulting in a prohibitive volume of data for storage, transmission and rendering. It is therefore vital to develop compression methods suitable for light fields to ensure that this becomes a practical technology.

2.2. Light Field Compression

Light field images obtained from multiple camera views inherently contain a large amount of redundancy. As such light field coding and compression has been well studied, see Viola *et al*. [25] for an overview of a number of techniques. Numerical methods that are commonly employed to compress light fields include vector quantization [14,28], wavelet transforms [7], non-negative matrix factorization [6] and principal component analysis (PCA) [6,28] . Here, we focus on light field compression schemes that utilize scene geometry to aid compression [6,21,28].

Surface light fields are an alternative approach to parameterize a 4D light field that define the radiance with respect to every point on a surface in all directions [6,17,28]. Wood *et al*. [28] extract and compress *lumispheres*, which store the directional radiance for every surface point mapped onto the surface of a sphere, using both vector quantization and PCA. Light Field Mapping [6] partitions the surface light field based on the elementary shape primitives of the 3D surface. Appearance variation is resampled on a per primitive basis, compressed using PCA and stored in surface and view map images giving further reduction through standard image compression.

Other light field compression techniques have been inspired by video coding and compression [5, 16, 21]. These works treat a subset of the sampled light field images as reference frames, or *i-frames*. Images within the local neighbourhood of each *i-frame* image of the light field array are compressed via predicted video frames or *p-frames*. These methods are capable of achieving high compression ratios and capitalize on work in video compression.

2.3. Multiple View Appearance Representation

More generally, an open research problem is how to efficiently represent appearance from multiple cameras. Light fields captured from large camera arrays make this a particularly challenging problem given the number of cameras and resulting size of the captured data. For an array of 10-100 cameras with UHD resolution and 8-bit colour depth, the raw image data requires approximately 240-2400 MB of storage at each time instance. Currently, there are no open standards for compression and transmission of light fields.

One method of appearance representation and compression that has not yet been explored for use with light fields is Eigen textures. Using PCA, a linear subspace can be computed from a set of images, enabling compression through dimensionality reduction. Appearance modelling through PCA was used as a method for face recognition [24]. Later, Nishino *et al*. [20] proposed the Eigen texture method that allowed compression and synthesis of novel views from a sparse set of viewpoints. More recently, Boukhayma *et al*. [4] extended this idea to handle dynamic objects by projecting the object's dynamic appearance onto a dynamic structured geometric proxy at each time instance. This approach considers dynamic appearance but does not preserve view-dependent surface appearance in the representation that is captured with light fields.

In this paper, we use a scene model and resample the light field appearance into a set of per camera UV maps. We process the per camera UV maps into Eigen textures to provide a compact representation for light fields that enables efficient storage and rendering whilst maintaining photo-realism and view-dependent appearance of the captured light field. This is achieved through the construction of a linear subspace that models the variation of the surface appearance across all viewpoints.

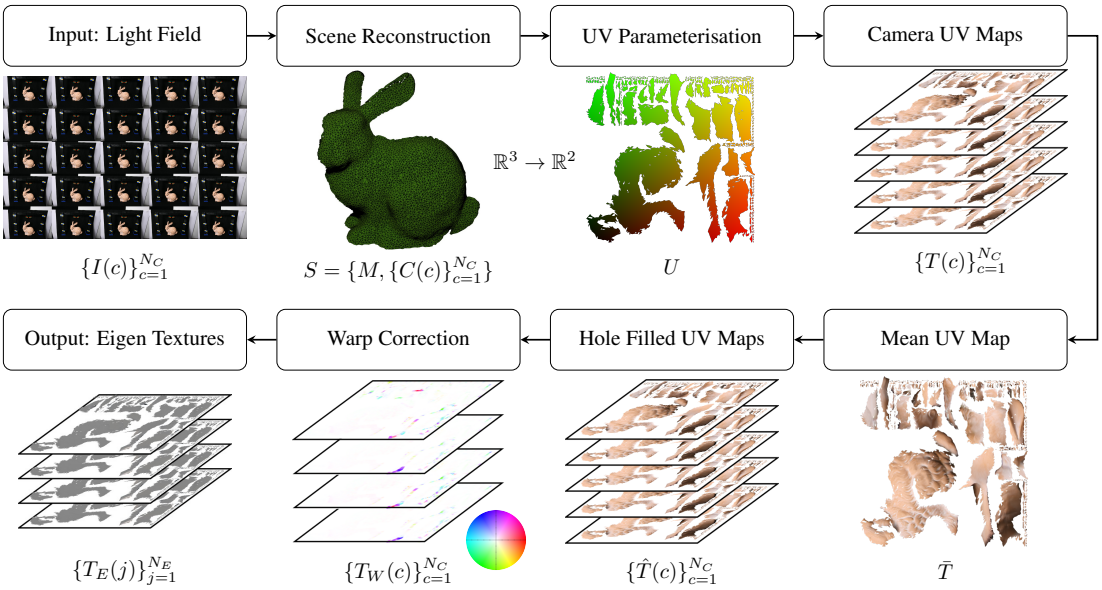


Figure 2: Eigen texture representation of light fields

3. Eigen Texture Representation

In this section we describe how the Eigen texture representation is extracted from a time instance in a multiple camera light field dataset. An overview of the pipeline is shown in Figure 2. Given the input light field images, a geometric proxy is reconstructed and UV coordinates are generated. Camera parameters and the geometric proxy are used to generate a UV map for each camera in the light field. Camera UV maps are processed to ensure there are a consistent number of pixels by filling in any holes from a mean UV map. The filled camera UV maps are used in a PCA based framework to generate the Eigen texture representation for light fields. Each of the stages are described in further detail throughout this section along with the view-dependent rendering pipeline.

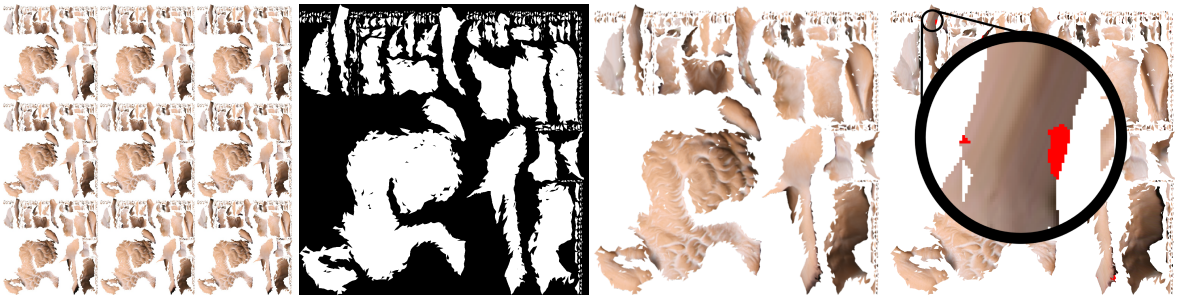
3.1. Input Data and Pre-processing

The input to our approach is a collection of images $\{I(c)\}_{c=1}^{N_C}$ captured from N_C cameras arranged in a rectangular array, as is common in light field video capture [23,27]. Scene geometry is modelled through explicit chart-based camera calibration, dense multiple view stereo and Poisson surface reconstruction. If calibration is not available, photogrammetry is employed to estimate camera calibration and model scene geometry [3, 9, 29]. Scene geometry, $S = \{M, \{C(c)\}_{c=1}^{N_C}\}$, consists of a triangular mesh model M and cameras $\{C(c)\}_{c=1}^{N_C}$. $C(c)$ represents the c^{th} camera of N_C cameras consisting of a 3x4 projection matrix that maps 3D world coordinates to 2D image coordinates. The triangular mesh model is defined as $M = \{V, W, U\}$, where V is the 3D mesh vertices, W is the mesh connectivity and U defines a $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ mapping from mesh vertices to UV coordinates. U are generated automatically using least-squares conformal maps [15].

3.2. Camera UV Map Processing

Using the reconstructed scene model M and camera parameters $\{C(c)\}_{c=1}^{N_C}$, we resample the observed surface appearance for each camera into a set of UV maps $\{T(c)\}_{c=1}^{N_C}$ based on the UV coordinates U of M. To achieve this, a 3D mesh vertex is projected into the image domain of camera c using the camera projection matrix and the colour of the pixel copied to the UV coordinate location associated with the vertex. To handle occlusions in the scene, depth testing is performed using a depth map rendered using the scene geometry and camera parameters. A binary value is stored in the alpha channel (α) of the UV map which indicates if a vertex is visible from the camera's viewpoint. This encodes surface appearance and visibility into the c^{th} camera UV map $T(c)$, an example of which shown in Figure 3b. This is performed in a custom graphics shader that interpolates vertex and UV coordinate values to give appearance and visibility across the complete mesh surface in each camera UV map, see Figure 3 for an example. This process results in N_C UV maps in which observations of a point on the 3D surface are mapped into the same pixel location in the 2D texture domain across all UV maps.

As each camera has variations in surface visibility, each UV map has a different number of visible pixels. To construct the Eigen texture representation, described in Section 3.3, it is required that all UV maps have an equal number of visible pixels. To this end, we first construct an average



(a) (b) (c) (d)

Figure 3: Examples of (a) Camera UV maps $\{T(c)\}_{c=1}^{N_C}$, (b) binary visibility map (stored in the alpha channel of the camera UV maps), (c) mean UV map $\bar{T}$ and (d) filled camera UV map $\hat{T}(c,i)$, with pixel that require filling highlighted in red.

UV map $\bar{T}$ which is then used to fill holes in $T(c)$. An average UV map $\bar{T}$ is generated as shown in Equation 1. This is computed on a per-pixel basis and results in a UV map in which the pixel values have been averaged over all visible pixels, resulting in an appearance value for all surface points in all UV maps $T(c)$.

$$\bar{T}(i) = \frac{1}{\sum_{c=1}^{N_C} v(c,i)} \sum_{c=1}^{N_C} v(c,i) T(c,i) \tag{1}$$

where $\bar{T}$ is the average over all N_C camera UV maps, $\bar{T}(i)$ is the average value for the i^{th} pixel, $v(c,i)$ is a binary value determined by the visibility encoded in the alpha channel of the camera UV map. Hole filling of the $T(c)$ takes place according to the conditions in Equation 2.

$$\hat{T}(c,i) = \begin{cases} T(c,i), & \text{if } v(c,i) = 1 \\ \bar{T}(i), & \text{otherwise} \end{cases} \tag{2}$$

where $T(c,i)$ and $\hat{T}(c,i)$ are values of the i^{th} pixel of camera c for the camera UV map and hole filled UV map, respectively.

To account for small errors in camera parameters and geometry estimation we employ optical flow image warping [8]. We compute the optical flow field between a given camera UV map $\hat{T}(c)$ and a defined reference camera UV map $\hat{T}(c_{ref})$, typically a camera located in the centre of the camera array, resulting in an optical flow field per camera $\{T_W(c)\}_{c=1}^{N_C}$. The flow fields are then applied to each $\hat{T}(c)$ to correct for small errors and bring all $\{\hat{T}(c)\}_{c=1}^{N_C}$ into alignment with $\hat{T}(c_{ref})$.

3.3. Eigen Texture Construction

In this section, we describe the construction of the Eigen texture representation for light field data. We define the function $\hat{T}(c,i,\lambda)$ which returns a scalar value for the i^{th} pixel of wavelength λ in the hole filled UV map $\hat{T}(c)$ for camera c. As the input of the proposed method is RGB images, λ refers to the red, green and blue image channels. However, the approach is independent of colour representation and could also be used to compress other attributes. A pixel in a UV map $\hat{T}(c)$ is represented in row-vector form, as shown in Equation 3.

$$\mathbf{x}(c,i) = \Big[\hat{T}(c,i,r) - \bar{T}(i,r), \hat{T}(c,i,g) - \bar{T}(i,g), \hat{T}(c,i,b) - \bar{T}(i,b)\Big] \tag{3}$$

where $\mathbf{x}(c,i)$ returns intensity values for the red (r), green (g) and blue (b) image channels for the i^{th} pixel of the c^{th} camera in row-vector form. A complete image can then be represented in row-vector form, as shown in Equation 4.

$$\mathbf{x}(c) = [\mathbf{x}(c,1), \mathbf{x}(c,2) ..., \mathbf{x}(c,N_P)] \tag{4}$$

where $\mathbf{x}(c)$ returns a row-vector consisting of the RGB pixel intensities for all N_P visible pixels (i.e. $v(c,i) = 1$) in the hole filled UV map $\hat{T}(c)$. Subsequently, N_C UV maps can be compiled into matrix $\mathbf{X}$ where rows represent all valid pixels in a UV map and columns represent all samples of a point on the model's surface captured by N_C cameras, as shown in Equation 5.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(N_C) \end{bmatrix} \tag{5}$$

where $\mathbf{X}$ is a matrix of the RGB pixel intensities with dimensions N_C rows by $3N_p$ columns. Prior to vectorization of $\hat{T}(c)$, the mean texture $\bar{T}$ is subtracted allowing PCA to be performed. In this form, a UV map can be thought of as a point in a $3N_P$ dimensional space. To compute the Eigen texture representation, singular value decomposition (SVD) is performed on the matrix $\mathbf{X}^\top\mathbf{X}$ to find the Eigen vectors and Eigen values, as shown in Equation 6. As the number

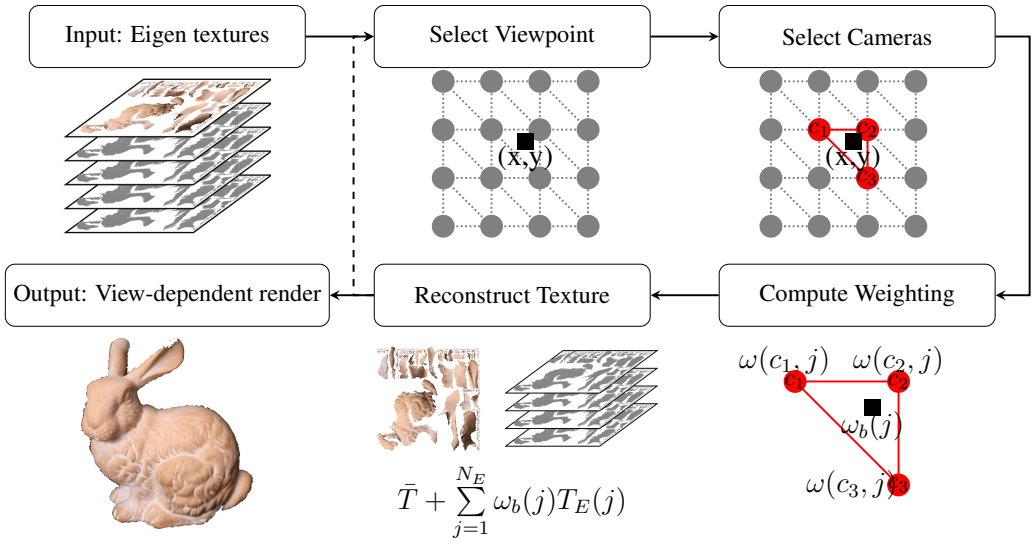


Figure 4: Light field Eigen texture rendering

of UV maps is less than the number of pixels, $N_C < 3N_P$, there are $N_C - 1$ rather than $3N_P$ non-zero values [24].

$$\mathbf{X}^\top \mathbf{X} = \mathbf{W} \Sigma \mathbf{W}^\top \tag{6}$$

where $\mathbf{W}$ is the orthogonal Eigen vectors and $\Sigma = diag(\sigma)_{1 \leq i \leq N_C}$ are the Eigen values. The rows of $\mathbf{W}$ are sorted in order of the magnitude of the Eigen values. The high amount of redundancy in the light field images allows the number of Eigen textures N_E to represent the view-dependent variation with $N_E \ll N_C$.

Back projecting the vectorised input filled camera UV maps into the Eigen space results in weighting coefficients $\{\{\omega(c,j)\}_{c=1}^{N_C}\}_{j=1}^{N_E}$ required to reconstruct the input. The weights are computed for each camera and stored for use at render-time. Eigen textures $\{T_E(j)\}_{j=1}^{N_E}$ are stored as image files along with the weighting coefficients to reconstruct the UV map from each camera.

3.4. Eigen Texture Rendering

Here we discuss the Eigen texture rendering pipeline, an overview is shown in Figure 4. The Eigen textures $\{T_E(j)\}_{j=1}^{N_E}$ and per camera reconstruction weights $\{\{\omega(c,j)\}_{c=1}^{N_C}\}_{j=1}^{N_E}$ are first loaded into memory. A viewing position (x, y), constrained to lie on an estimated plane and within the bounds of the camera array, is selected by the user. The three closest cameras to the viewing position measured by Euclidean distance are selected for rendering, denoted as c_1, c_2 and c_3. Given the viewing position and positions of the selected rendering cameras, a weighting based on barycentric coordinates is computed. These barycentric weights are then combined with the Eigen texture reconstruction weight for the selected cameras to give the interpolated reconstruction weights, as shown in Equation 7.

$$\omega_b(j) = b_1 \omega(c_1, j) + b_2 \omega(c_2, j) + b_3 \omega(c_3, j) \tag{7}$$

where $b_{\{1,2,3\}}$ are the barycentric weights, $\omega(c_{\{1,2,3\}}, j)$ are the Eigen texture reconstruction parameters for cameras $c_{\{1,2,3\}}$ and $\omega_b(j)$ is the barycentric weighted reconstruction parameters. The use of a barycentric weighting scheme ensures a smooth transition between camera reconstruction weights in both the horizontal and vertical directions. An alternative weighting scheme could be used, e.g. bi-linear. However, in the current implementation it would be subject to $\sum_{i=1}^{3} b_i = 1$. The final view-dependent texture T_V for viewpoint v is computed by the linear combination of the mean texture, Eigen textures and Eigen texture reconstruction weights, shown in Equation 8.

$$T_v = \bar{T} + \sum_{j=1}^{N_E} \omega_b(j) T_E(j) \tag{8}$$

where $\bar{T}$ is the mean texture, $T_E(j)$ are the computed Eigen textures, $\omega_b(j)$ are the barycentric weighted reconstruction weights corresponding to the j^{th} Eigen texture, and T_v is the resulting view-dependent texture. This enables efficient view-dependent rendering directly from the Eigen texture representation.

4. Results and Evaluation

In this section, we present a quantitative evaluation of the proposed approach. We compare UV maps reconstructed from the Eigen texture representation to the camera UV maps extracted from the scene model and camera images.

The storage requirements of the Eigen texture representation are also compared against the camera UV maps. Light field data was captured by a 5x4 camera array made up of 5MP PointGrey Grasshopper3 cameras (Boy and Girl sequences). To test the limits of the algorithm, we also use scenes from the Stanford light field dataset [23] captured by a single camera on a robotic gantry resulting in the equivalent of a 17x17 camera array.

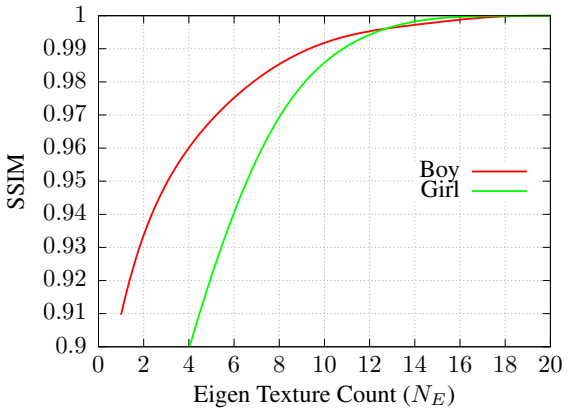
Scene geometry and camera parameters from the 5x4 camera array were computed using a chart-based calibration pipeline [2, 11], multiple view stereo [22] and Poisson surface reconstruction [12]. For the Stanford datasets [23], scene geometry and camera parameters were estimated using photogrammetry [3]. Scenes were chosen to give a mixture of objects and materials, e.g. Bunny is a Lambertian surface with few specular highlights whereas Amethyst and Chest contain metal and glass structures with strong specular highlights. In the case of the Tarot dataset, the scene was reconstructed with photogrammetry and the glass ball was manually modelled with a sphere as it is not currently possible to geometrically reconstruct such a complex object.

All presented results were generated using a desktop PC with an Intel i7 CPU, 64GB of RAM and a Nvidia Geforce GTX 1080 GPU. All code is written in C++ and makes use of Eigen, OpenCV and OpenSceneGraph libraries. OpenGL Shading Language (GLSL) are used to render depth maps, camera UV maps and to perform Eigen texture rendering.

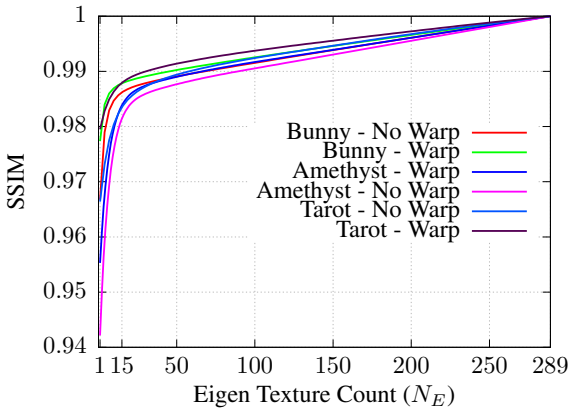
4.1. UV Map Reconstruction Quality

The evaluation is performed by comparing the hole filled UV maps $\{\hat{T}(c)\}_{c=1}^{N_C}$ to the UV maps reconstructed from the Eigen texture representation using the per camera reconstruction parameters. This is a direct comparison of the reconstructed output T_V to expected output $\{\hat{T}(c)\}_{c=1}^{N_C}$. The UV maps are compared using the structural similarity index measure (SSIM) [26] which is considered across all visible foreground surface points. SSIM values range from ± 1 with $+1$ achieved only when comparing identical images.

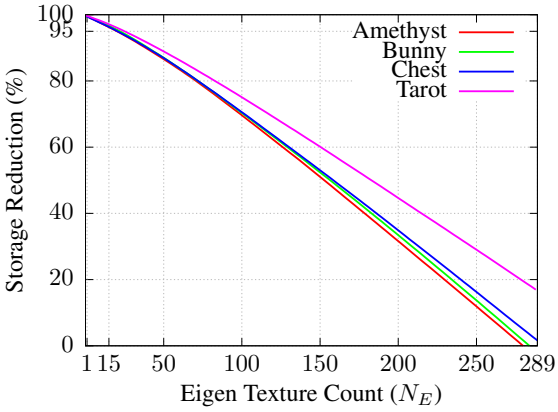
Figure 5a and 5b show SSIM values versus the number of Eigen textures used for rendering N_E for the evaluation of the datasets. We see that as N_E is increased, the difference between the reconstructed UV map T_V and the camera UV map decreases. In the Amethyst, Bunny and Chest dataset, there is an increase in quality, up to an SSIM 0.98, until $N_E = 15$ after which it is a linear increase. This represents a 20:1 reduction in the number of textures required. The same relationship is also observed in the Boy and Girl datasets, but in a less dramatic fashion requiring 7 and 9 Eigen textures for an SSIM of 0.98 for the Boy and Girl datasets, respectively. The reason for this is that the Amethyst, Bunny and Chest light field datasets consist of 289 images compared to the relatively sparse light field



(a)



(b)



(c)

Figure 5: Quantitative evaluation of the proposed approach: (a) Rendering quality against the number of Eigen textures with and without optical flow correction using scenes from the Stanford light field gallery [23]; (b) Rendering quality against the number of Eigen textures for 5x4 light field dataset; (c) Storage reduction for Stanford light field dataset.

with 20 images in the Boy and Girl datasets. This results in increased redundancy that is exploited by the Eigen texture representation and results in less Eigen textures to represent the variation in the captured images. Rendering results from the evaluation datasets can be found in Figure 6 and in the supplementary video.

4.2. Warping

An inaccurate geometric proxy and errors in camera parameters result in misalignment in the texture domain. These errors affect compression as they lead to a requirement for more Eigen textures to represent the surface variation. Figure 5b demonstrates this by comparing the eigen texture representation using the evaluation datasets with and without optical flow correction applied. It can be seen that performing the optical flow based warping allows a higher UV map reconstruction quality with fewer Eigen textures. This is due to the appearance being in alignment minimizing the surface variation.

4.3. Rendering Performance

A trade off must be considered between rendering quality and rendering complexity. As more components are added to the Eigen texture representation, the more computationally complex the rendering pipeline becomes. An evaluation of the rendering efficiency was performed by monitoring the rendering frame rate against N_E. The proposed approach is able to achieve and maintain interactive frame rates ($>$ 60 fps) for $N_E <= 40$ At $N_E >= 40$ the frame rate begins to fall in an almost linear fashion to 30 fps at $N_E = 100$.

4.4. Storage

Table 1 shows the storage requirements for the evaluation datasets and Figure 5c shows storage reduction against N_E. A comparison is made between the storage requirements of the camera UV maps $\{\bar{T}(c)\}_{c=1}^{N_C}$ and the Eigen texture representation consisting of a mean texture $\bar{T}$ and Eigen textures $\{T_E(j)\}_{j=1}^{N_E}$. This ensures we are comparing the data input and output to the Eigen texture representation. It can be seen that as N_E is increased the storage reduction decreases. Taking the values of N_E that result in a SSIM of 0.98, $N_E = 15$ for Amethyst, Bunny, Chest and Tarot datasets results in a storage reduction of $> 96\%$ and for the Boy and Girl datasets by approximately 60%.

4.5. Limitations

In the presence of gross geometric errors the proposed method will be unable to effectively compress the appearance using the eigen texture method. In practice, it is not possible to get accurate geometry as the light field scenes used in the evaluation contain complex specular objects and transparent surfaces that exhibit refraction, e.g. Amethyst, Chest and Tarot. Despite the approximate geometric reconstruction, the proposed approach is able to achieve high quality renderings with a $> 95\%$ reduction in data size for dense light fields. This can be observed in the supplementary video particularly in the Tarot dataset where the compressed Eigen texture representation is able to reproduce the complex refraction within the glass ball.

Table 1: Storage requirements of light field datasets in represented as images, per camera texture maps and eigen textures.

Dataset	N_C	Storage (MB)		
		$\{I(c)\}_{c=1}^{N_C}$	$\{\hat{T}(c)\}_{c=1}^{N_C}$	$\{T_E(j)\}_{j=1}^{N_E}$ †
Amethyst	289	769.4	359.6.2	13.4 (96%)
Bunny	289	770.3	276.5	9.5 (97%)
Chest	289	756.4	387.7	14.8 (96%)
Tarot	289	1100	425.5	12.0 (97%)
Boy	20	97.7	16.4	5.7 (65%)
Girl	20	105	21.7	7.2 (67%)

†For Amethyst, Bunny, Chest and Tarot datasets $N_E = 15$, for Boy and Girl datasets $N_E = 7$.

5. Conclusions

In this paper, we have proposed and demonstrated for the first time that Eigen textures can be effectively used to represent and render light fields. We have described how to construct an Eigen texture representation from a light field camera array and how this Eigen texture basis can be used to perform view-dependent rendering. A quantitative evaluation was performed in which it was shown that a storage saving of $> 95\%$ could be obtained for dense light fields using the Eigen texture representation with a minimal loss in quality. It was also shown that in datasets that used large numbers of cameras, a higher storage reduction was achieved as the representation was able to exploit the increased redundancy.

Future work will investigate extending this approach to dynamic scenes in a way that exploits the large redundancy both spatially and temporally while preserving both dynamic and view-dependent surface appearance. This requires solving the challenging problem of surface correspondence [18] to enforce a consistent mesh topology over time.

Acknowledgements

This work was supported by the following sources 'ALIVE: Live action light fields for immersive virtual reality experiences' (InnovateUK 102686), 'Polymersive: Immersive Video Production Tools for Studio and Live Events' (InnovateUK 105168), EPSRC Audio-Visual Media Research Platform Grant (EP/P022529/1) and Royal Academy of Engineering Research Fellowship (RF-201718-17177).

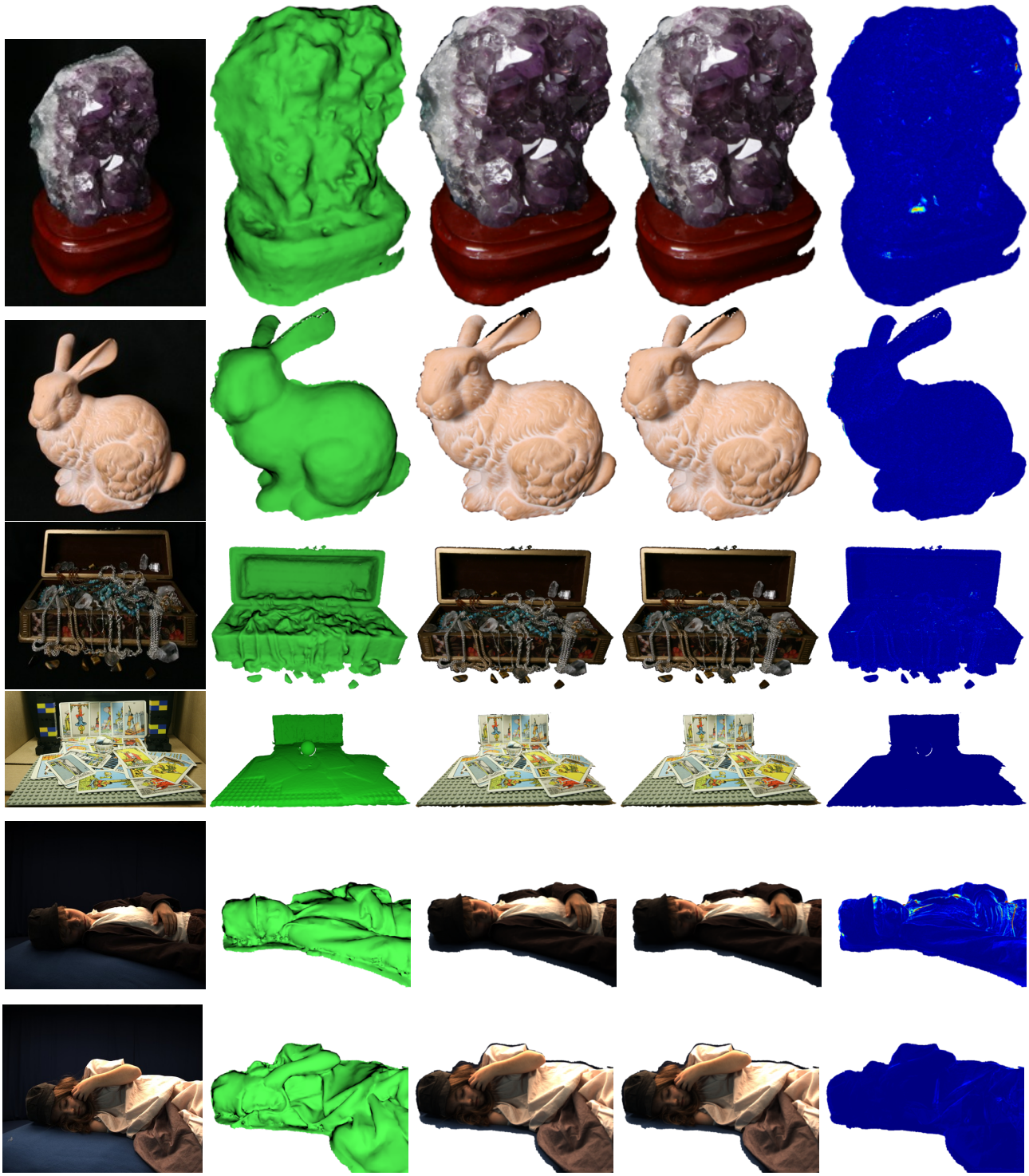


Figure 6: Results: Rows (top to bottom) show Amethyst, Bunny, Chest, Tarot, Boy, and Girl datasets, respectively. From left to right, selected image from dataset, scene reconstruction model, render using camera UV map, Eigen texture render using $N_E = 15$ for Amethyst, Bunny, and Chest, and $N_E = 7$ for Boy and Girl datasets. Heat maps show the normalized error in RGB pixel values, for illustrative purposes this error has been scaled 5 times.

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, 1991. 2

[2] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org. 6

[3] Agisoft. Agisoft Photoscan v1.3.2. 3, 6

[4] A. Boukhayma, V. Tsiminaki, J.-S. Franco, and E. Boyer. Eigen Appearance Maps of Dynamic Shapes. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 230–245. Springer International Publishing, 2016. 2

[5] Cha Zhang and Jin Li. Compression of lumigraph with multiple reference frame (mrf) prediction and just-in-time rendering. In *Proceedings DCC 2000. Data Compression Conference*, 2000. 2

[6] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02. ACM, 2002. 2

[7] Chuo-Ling Chang, Xiaoqing Zhu, P. Ramanathan, and B. Girod. Light field compression using disparity-compensated lifting and shape adaptation. *IEEE Transactions on Image Processing*, 15(4):793–806, 2006. 2

[8] G. Farneb. Two-Frame Motion Estimation Based on Polynomial Expansion. *Lecture Notes in Computer Science*, 2749(1):363–370, 2003. 4

[9] S. Fuhrmann, F. Langguth, and M. Goesele. MVE-A Multi-View Reconstruction Environment. *Eurographics Workshop on . . .*, pages 11–18, 2014. 3

[10] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *SIGGRAPH*, pages 43–54, 1996. 2

[11] Itseez. Open source computer vision library v2.4. http://opencv.org/, 2017. 6

[12] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70. Eurographics Association, 2006. 6

[13] M. Kemp. *Leonardo on painting : anthology of writings by Leonardo da Vinci, with a selection of documents relating to his career as an artist*. New Haven ; London : Yale Nota Bene, 2001., 2001. 2

[14] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 31–42. ACM Press, 1996. 2

[15] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002. 3

[16] M. Magnor and B. Girod. Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, 2000. 2

[17] G. Miller, S. Rubin, and D. Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*. Springer Vienna, 1998. 2

[18] A. Mustafa, M. Volino, J. Guillemaut, and A. Hilton. 4d temporally coherent light-field video. In *2017 International Conference on 3D Vision (3DV)*, pages 29–37, 2017. 7

[19] R. Ng, M. Levoy, G. Duval, M. Horowitz, and P. Hanrahan. Light Field Photography with a Hand-held Plenoptic Camera. *Computer Science Technical Report CSTR*, 2005. 2

[20] K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-texture method: Appearance compression and synthesis based on a 3D model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1257–1265, 2001. 2

[21] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Trans. Graph.*, 37(6):197:1–197:15, 2018. 2

[22] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, pages 519–528, 2006. 6

[23] Stanford Graphics Laboratory. The (New) Stanford Light Field Archive. 3, 6

[24] M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, Jan. 1991. 2, 5

[25] I. Viola, M. ebek, and T. Ebrahimi. Comparison and evaluation of light field image coding approaches. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1092–1106, 2017. 2

[26] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[27] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Trans. Graph.*, 24(3):765–776, 2005. 2, 3

[28] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3D photography. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–296, 2000. 2

[29] C. Wu. Towards linear-time incremental structure from motion. *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*, pages 127–134, 2013. 3

[30] K. Yücer, A. Sorkine-Hornung, O. Wang, and O. Sorkine-Hornung. Efficient 3D Object Segmentation from Densely Sampled Light Fields with Applications to 3D Reconstruction. *ACM Transactions on Graphics*, 35(3):1–15, 2016. 2