Light Field Video for Immersive Content Production

Marco Volino*[0000−0002−1869−9257], Armin Mustafa*[0000−0002−1779−2775],
Jean-Yves Guillemaut[0000−0001−8223−5505], and
Adrian Hilton[0000−0003−4223−238X]

Centre for Vision, Speech and Signal Processing; University of Surrey, UK
{m.volino, a.mustafa, j.guillemaut, a.hilton} @surrey.ac.uk
https://www.surrey.ac.uk/centre-vision-speech-signal-processing
* Equal author contribution

Abstract. Light field video for content production is gaining both research and commercial interest as it has the potential to push the level of immersion for augmented and virtual reality to a close-to-reality experience. Light fields densely sample the viewing space of an object or scene using hundreds or even thousands of images with small displacements in between. However, a lack of standardised formats for compression, storage and transmission, along with the lack of tools to enable editing of light field data currently make it impractical for use in real-world content production. In this chapter we address two fundamental problems with light field data, namely representation and compression. Firstly we propose a method to obtain a 4D temporally coherent representation from the input light field video. This is an essential problem to solve that will enable efficient compression editing. Secondly, we present a method for compression of light field data based on the eigen texture method that provides a compact representation and enables efficient view-dependent rendering at interactive frame rates. These approaches achieve an order of magnitude compression and temporally consistent representation that are important steps towards practical toolsets for light field video content production.

Keywords: Light Fields · 4D Reconstruction · Representation · Compression

1 Introduction

The rise in popularity of virtual and augmented reality (VR/AR) has fueled an increasing demand for high-quality digital characters and environments to enable immersive storytelling. Currently immersive content is produced using either artist-driven computer generated imagery or 360 video capture. The use of computer generated imagery and game engines allow interactivity, such as head movement, but do not achieve photo-realism in a practical or cost effective manner. Alternatively 360 video and stereo 360 video achieve photo-realism from a fixed location but do not allow head movement, also known as 6 degrees-of-freedom (6DoF), or realistic parallax. Light field technology has emerged as a solution for capturing objects and scenes with photo-realism whilst allowing realistic changes in viewpoint with correct parallax [19,14]. However, a number of open problems must be addressed before light field technology can be practically deployed for real-world productions.

For a given scene, a light field describes the light intensity passing through every point in space and in every direction. In order to capture a light field it is necessary to densely sample a scene using hundreds [34] or even thousands [47,30] of images. To date, light fields have primarily been used for capture of static scenes due to the requirement for a large number of viewpoints. Recently, motivated by applications in immersive AR/VR content production, arrays of video cameras have be employed to acquire light fields of dynamic scenes [44,21]. However, image and video based techniques for compression and editing cannot be directly applied to light field data as they fail to exploit the spatial and temporal redundancy present in the light field. Hence there is a need for an efficient light field video representation that is capable of exploiting the spatio-temporal redundancy inherent within light field data to enable compression and facilitate editing.

In this chapter we address two main issues that limit the use of light field video for immersive content production: Representation and Compression. We perform reliable temporal alignment of partial surfaces for complex dynamic scenes exploiting properties of the light field to obtain robust 4D temporal representation of the scene. Sparse temporal correspondence tracks are obtained for each view for the dynamic sequence using feature matching. These sparse temporal correspondence tracks are used to initialize a dense flow estimation. A novel sparse-to-dense flow is proposed exploiting Epipolar Plane Images (EPI) using oriented light field windows to obtain a temporally coherent dense 4D representation of the scene.

For compression, we introduce a compact light field representation that enables up to a 95% decrease in data size and offers efficient rendering at interactive frame rates on commodity graphics hardware whilst maintaining the visual quality of the captured light field. To summarize, the contributions of this chapter are:

- Temporally coherent 4D reconstruction of dynamic light field video for efficient editing.
- EPI retrieval from sparse light field video for spatio-temporal correspondence.
- Sparse-to-dense scene flow exploiting EPI image information.
- A novel Eigen texture representation for light fields which is compact and preserves the view-dependent photo-realism of the captured light field.
- 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 commodity graphics hardware.

The work presented in this chapter is based on the following published papers: '4D Temporally Coherent Dynamic Light Field Video' [27] and 'Light Field Compression using Eigen Textures' [39]. The remainder of the chapter is structured as follows: Section 2 provides a background on light field capture, compression and 4D scene reconstruction; Section 3 presents the proposed methods for establishing 4D temporal correspondence for light field video sequences and representation and compression of light field data using eigen textures; Section 4 presents quantitative and qualitative evaluation of the proposed methods for 4D correspondence and light field representation and compression; finally Section 5 concludes the chapter and proposes how these strands of work can be extended and combined to create a robust and compressible 4D representation for light field video.

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 [18]. 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 [19] and Gortler *et al*. [14] 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.

The general principle of light field photography was pioneered in 1908 by Gabriel Lippmann under the name of integral photography. However, practical methods for light field acquisition have only started to emerge in recent years using digital camera technology. Early approaches to capture light fields performed a dense sampling of the 3D space by moving a camera around the scene. These approaches used either a camera mounted on a gantry [19] or more recently a hand-held camera [9]. In these approaches, each frame effectively captures a 2D slice of the 4D light field. Due to the need to physically move the camera around the scene, these approaches are limited to static scenes. Alternatively, a micro lens array can be placed in front of a conventional image sensor [28] (e.g. Lytro Illum and Raytrix cameras). Micro lens arrays trade off spatial resolution of the image sensor to angular resolution of the lens array. These approaches use image-based rendering techniques to synthesise new views by interpolating the information from the captured views. As they do not explicitly capture the scene geometry, a large number of camera views (typically about 100) are required in order to densely sample the light field.

2.2 4D Scene Reconstruction

For conventional single view depth sequences and multiple view reconstruction of dynamic scenes techniques have been introduced to align sequences using correspondence information between frames. Methods have been proposed to obtain sparse [25,16,49,35] and dense [23,48,4] correspondence between consecutive frames for entire sequences. Existing sparse correspondence methods work independently on a frame-by-frame basis for a single view [35] or multiple views [16] and require a strong prior initialization [49]. Existing feature matching techniques either work in 2D [35] or 3D [23] or for sparse [16,49] or dense [48] points. Other methods are limited to RGBD data [48] or stereo pairs [23] for dynamic scenes. Dense matching techniques include scene flow methods. Scene flow techniques [42,4] typically estimate the pairwise surface or volume correspondence between reconstructions at successive frames but do not extend to 4D alignment or correspondence across complete sequences due to drift and

failure for rapid and complex motion. In this paper we propose sparse-to-dense temporal alignment exploiting the high spatio-temporal redundancy in light fields to robustly align light field video captured with sparse camera arrays.

2.3 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*. [38] for an overview of a number of techniques. Numerical methods that have been commonly employed to compress light field data include vector quantization [19,45], wavelet transforms [8], non-negative matrix factorization [7] and principal component analysis (PCA) [45,7] . in this section we focus on light field compression schemes that utilize scene geometry to aid compression [45,7,30].

Surface light fields are an alternative approach to parameterize a 4D light field that defines the radiance with respect to every point on a surface in all directions [24,45,7]. Wood *et al*. [45] 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 [7] 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 image compression techniques have been inspired by video coding and compression [30,22,6]. 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 frames or *p-frames*. These methods are capable of achieving high compression ratios and capitalize on work in video compression.

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 [37]. Later, Nishino *et al*. [29] proposed the Eigen texture method that allowed compression and synthesis of novel views from a sparse set of viewpoints. More recently, Boukhayma *et al*. [5] 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.

3 Method

3.1 Overview

An overview of the framework to obtain compressed light field video for immersive content production is shown in Figure 1. A 4D temporally coherent representation is created from the input light field video, explained in Section 3.3. This is followed by

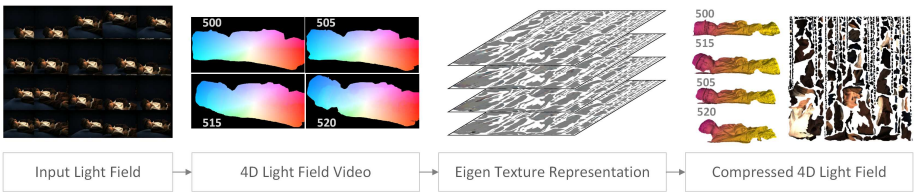

Fig. 1: Light field video for Immersive Content Production.

compression of light field video using eigen textures explained in Section 3.4. This gives a compressed 4D light field at each time instance that can be used for immersive content production with efficient storage and could facilitate editing operations.

3.2 Light Field Capture

In order to capture light field video, an array consisting of 20 FLIR Grasshooper3 machine vision cameras [12] was constructed. The cameras were uniformly spaced in a 5×4 grid configuration over an area of 50cm×50cm. Synchronised video streams were captured with 5MP resolution at 30 frames per second. The overall dimensions of the camera rig were designed to capture video over a range of viewpoints that a person could comfortably achieve from a seated position.

The intended use case of this work is to produce content for seated VR experiences. The camera array was used to capture human actors performing short theatrical scenes against a chroma key background. This simplified object isolation for reconstruction purposes and also for the final composition into virtual environments. Throughout this chapter the sparse 5x4 camera array is used to capture light field video of human actors, as shown in Figure 2.

3.3 4D Temporal Alignment of Light Field Video

The high spatial and temporal redundancy in light field video makes it challenging to use in content production for AR/VR. The aim of the work presented in this section is to obtain a 4D temporally coherent representation of dynamic light field video exploiting spatio-temporal redundancy. This provides an efficient structured representation for light field compression, editing and use in immersive VR content production.

Overview

Given an independent per-frame surface or depth reconstruction from the light field video captured with a sparse camera array the problem is to simultaneously estimate the temporal correspondence of the input light field across all views for the entire sequence. This is achieved efficiently by estimating the temporal alignment of the reconstructed surface between each time frame and propagating this across all light field camera views. A coarse-to-fine approach is introduced that initially estimates temporal correspondence based on sparse features and then estimates dense scene flow initialised from the sparse features. To ensure robust tracking, key-frames are identified as

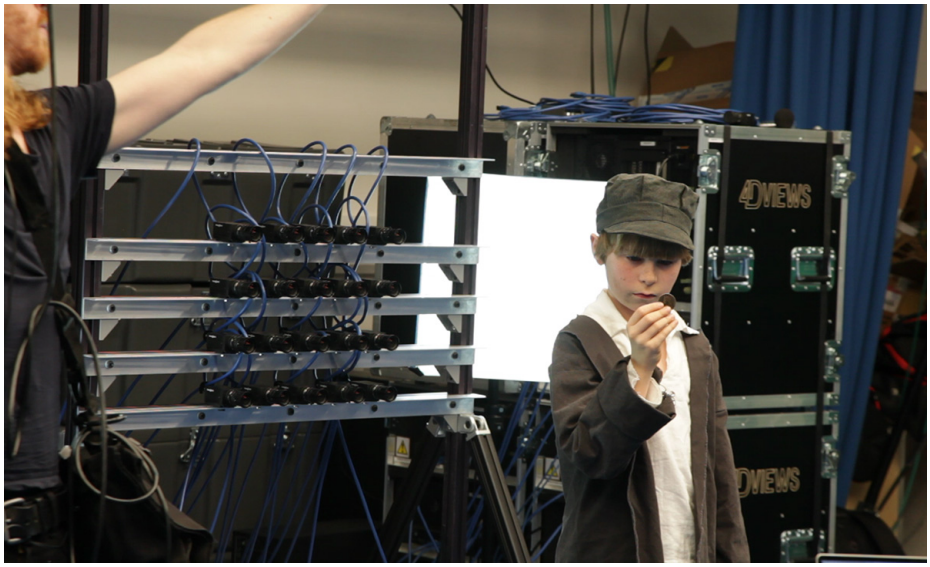

Fig. 2: Sparse light field video array used to capture an actor as part of Kinch & the Double World production tests - Photo Credit: Figment Productions.

a reference to minimise drift in long-term tracking. An overview of the 4D temporally coherent reconstruction of light field video is presented in Figure 3.

Mesh Reconstruction: Light field video of the dynamic scene is captured using a camera array. Multiple-view stereo is performed to obtain a per frame surface mesh reconstruction [32].

Object identification: The reconstructed point cloud is clustered in 3D [31] with each cluster representing a unique foreground object, shown in Figure 4.

Key-frame detection: Key-frames are detected for light field video exploiting redundant spatial information across views to identify a set of unique reference frames for stable long-term tracking. Surface tracking is performed between key-frames to reduce the accumulation of errors in sequential tracking due to large non-rigid motion for long sequences ($\approx$ 700 frames).

Sparse temporal feature correspondence tracks: Reconstructed 3D points projected at all frames are matched frame-to-frame across the sequence to estimate sparse temporal feature tracks for each dynamic object for each light field camera view. These sparse temporal correspondence tracks are used to initialize light field scene flow to handle occlusions and improve robustness.

Light field scene flow: We propose to estimate dense scene flow between images by exploiting EPI information from light field video based on oriented light field windows [33] initialised by the sparse feature correspondence per light field view at each time instant. This exploits the spatio-temporal redundancy in light fields to give 2D dense correspondences which are back-projected to the 3D mesh to obtain a 4D spatio-temporally coherent dynamic light field video representation.

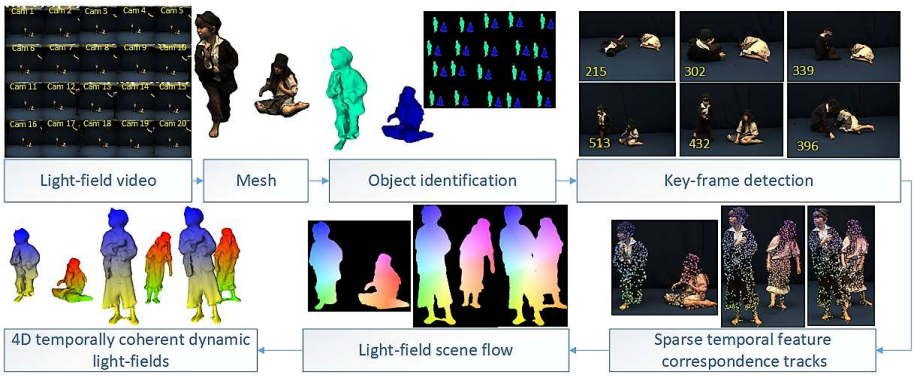

Fig. 3: 4D Light field video framework.

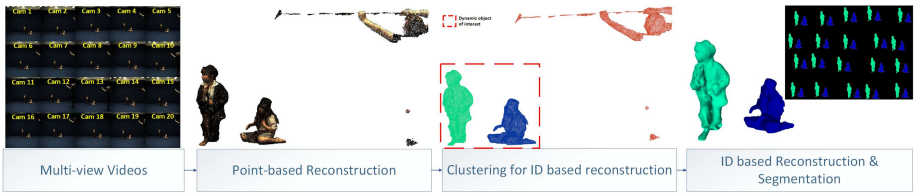

Fig. 4: Object identification example on light field frame.

Key-frame detection

Aligning per-frame reconstruction for long sequences leads to drift due to accumulation of errors in alignment between frames and failure is observed due to large non-rigid motion. To tackle this problem we detect key-frames across the sequence. Key-frame detection exploits the spatial redundancy in light field capture by fusing the appearance, distance and shape information across all views (N_v) in the sparse camera array. Temporal coherence is introduced between key-frames as explained in Section 3.3 and 3.3.

Appearance Metric ($M^c_{i,j}$): This measures appearance similarity between frame i and j for each object region in light field view c. It is the ratio of the number of temporal feature correspondences $Q^c_{i,j}$ to the total number of features in the object region at frame i, R^c_i and j, R^c_j: $M^c_{i,j} = \frac{2Q^c_{i,j}}{R^c_i + R^c_j}$

Distance Metric ($L^c_{i,j}$): This metric measures the distance between frame i and j for each object in each view c, it is defined as: $L^c_{i,j} = \frac{j-i}{D^c_{max}}$ where $j > i$ and D^c_{max} is the maximum number of frames between key-frames for view c. This term ensures that the distance between two key-frames does not exceed D^c_{max}. This is set to 100.

Shape metric ($I^c_{i,j}$): Gives the shape overlap between pairs of frames for each object in the light field video. It is defined as the ratio of the intersection of the aligned segmentation [10] $h^c_{i,j}$ to the union of the area $A^c_{i,j}$: $I^c_{i,j} = \frac{h^c_{i,j}}{A^c_{i,j}}$

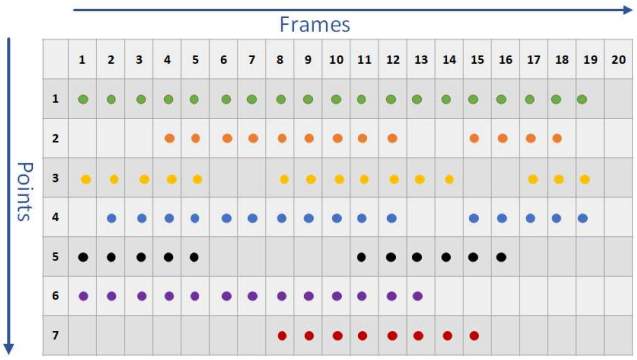

Fig. 5: Example of sparse feature correspondence tracks.

Key-frame similarity metric: The metrics defined above are used to calculate the similarity between frames as follows: $D_{i,j} = 1 - \frac{1}{3N_v}\sum_{c=1}^{N_v}(M_{i,j}^c + I_{i,j}^c + L_{i,j}^c)$. All frames with similarity > 0.75 are selected as key-frames defined as $K_i = \{k^{i+1}, k^{i+2}, ..., k^{N_f^i}\}$ where $i = 1$ to N_k (number of key-frames in light field video) and N_f^i is the number of frames between K_i and K_{i+1}.

Sparse temporal feature correspondence

Numerous approaches have been proposed to temporally align moving objects in 2D using either feature matching or optical flow. However these methods may fail in the case of occlusion, movement parallel to the view direction, large motions and visual ambiguity. To overcome these limitations we match sparse feature points from all light field camera views at each time instant for each object. This is used to estimate the similarity between the object surface observed at different frames of the light field video for key-frame detection and subsequently to initialize dense light field scene flow between frames.

3D points corresponding to each dynamic object are projected at each frame to ensure spatial light field coherence. The features for each light field camera view are defined as: $F_i^c = \{f_1^c, f_2^c, ..., f_{R_i^c}^c\}$, where $c = 1$ to N_v and $N_v = 20$ in our case. R_i^c are the 3D points visible at each frame i. Nearest neighbour matching is used to establish matches between features. The ratio of the first to second nearest neighbour descriptor matching score is used to eliminate ambiguous matches ($ratio < 0.85$). This is followed by a symmetry test which employs the principle of forward and backward match consistency to remove erroneous inconsistent correspondences. To further refine the sparse matching and eliminate outliers we enforce local spatial coherence in the matching. For matches in an $m \times m$ ($m = 11$) neighbourhood of each feature we find the average Euclidean distance and constrain the match to be within a threshold ($\pm\eta < 2 \times$ Average Euclidean distance).

Sparse temporal correspondence tracks are obtained by performing exhaustive matching between all frames N_f^i for each key-frame for the entire sequence. Feature matching is performed between frames such that features at view c frame i, F_i^c are matched

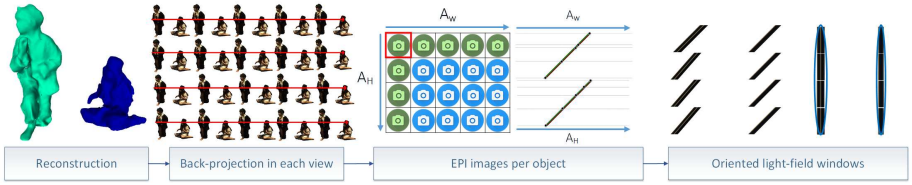

Fig. 6: An example of EPI representation of sparse light field (blue ellipse represents Gaussian weighted window).

to features at view c to frames $j = \{i+1, ..., N_f^i\}$. This gives us correspondences for all the frames N_f^i with key-frame K_i. Point tracks are constructed from this correspondence information for key-frame K_i. The same process is repeated for key-points which are not part of point tracks at the corresponding key-frame for all frames $j = \{i+1, ..., N_f^i\}$. Any new point-tracks are added to the list of point tracks for key-frame K_i. The exhaustive matching between frames per key-frame handles reappearance and disappearance of points due to occlusion or object movement. An example of sparse feature correspondence tracks in shown in Figure 5. Frame 1 is a key-frame and each point track is represented by a unique colour with missing points showing partial feature correspondence tracks. Sparse correspondences are also obtained between all key-frames to initialise dense light field flow.

Light field scene flow

Dense temporal correspondence is estimated using a novel light field scene flow method using pairwise dense correspondence which is estimated using the light field oriented window matching. This combines information across all light field views to achieve robust temporal correspondence. The sparse feature correspondences provides a robust initialisation of the proposed dense flow for large non-rigid shape deformation in the light field video.

The high-level of inherent redundancy in the light field images is used to improve the robustness and reliability of dense scene flow by using oriented light field windows on the EPI for matching. This has been shown to improve the temporal correspondences for scene flow [33]. In this paper we propose a novel method to obtain EPIs from sparse light field data to improve the quality of dense scene flow. Our approach uses oriented light field windows [33] to represent surface appearance information for matching as illustrated in Figure 6.

Epipolar Plane Image (EPI) Traditional light field data is captured with a plenoptic camera or a dense camera array typically capturing 200-300 views [34]. An example dataset is illustrated in Figure 7 (a) captured with 289 images. Given a dense set of views the EPI provides a representation for estimating correspondence across views from the regular structure of the image i.e. slant lines correspond to the same surface point. The 2D slice of such representation has the same width as the captured image and height is given by the number of views in a camera array row (17 in Figure 7 (a)). With dense sampling the disparity between adjacent views is typically sub-pixel giving slant lines in the EPI. Approaches have been proposed to estimate depth and segmentation

from this dense EPI representation [47,41]. However for sparse camera sampling the disparity between views may be several pixels making it difficult to directly establish surface correspondence from the EPI, Figure 7 (b). In our case of sparse views with just 20 cameras, the height of this EPI reduces to 5 pixels which makes it challenging to utilize the regularity of the information from EPI obtained from light fields.
In this paper we aim to use the EPIs to introduce spatio-temporal coherence in dynamic light field video. We propose a novel method to create an EPI parametrized representation from sparse light field capture. We assume that the calibration is known. All the images at each time instant are undistorted and rectified with respect to the reference camera. The depth information at each time instant is used to resample the light field to create an EPI representation of sparse light field data. The algorithm to obtain EPI from image, calibration and depth information is illustrated in Figure 6; the stages are as follows:

- The dense point-cloud P of each dynamic object is projected on the undistorted and rectified images. $B^c_{l,k}$ is the set of points in view c. For each row k of the projected points on the dynamic object a 2D EPI is obtained.
- The size of 2D EPI is set to $W \times H$, where W is the width of input images and $H = N_w \times \mu$ is estimated corresponding to the number of cameras in each row (N_w). μ is a constant introduced to increase the distance between views due to sparse camera sampling. It is set to 50 in our case to maximize performance. For each row of the camera array, H_o 2D EPIs are obtained for each dynamic object, where H_o is the height of the object. The corresponding projected points $B^c_{l,k}$ for each point $P_{l,k}$ are plotted on the 2D EPI with x coordinates the same as that of input images and y coordinates are estimated using the translation information from the calibration, defined as: $\frac{H \times \text{Distance between consecutive cameras}}{\text{Maximum distance between pair of cameras}}$
- Given the set of image samples corresponding to a given surface point we fit a line in the EPI due to imperfect camera array. A scene point is represented by a line in the 2D EPI. A similar process is repeated for the entire point-cloud P to obtain multiple 2D EPIs. Given a 2 dimensional camera array there are 2 sets of EPIs obtained by re-sampling along epipolar lines in the vertical and horizontal directions

The EPIs obtained from sparse camera arrays are used to constrain dense light field flow using oriented light field windows.
Oriented light field windows Oriented light field windows exploit the regularity information in the EPI to enable more robust and accurate pixel comparisons [33] over spatial windows that suffer from defocus blur and loss of precision. In this section we propose to use these oriented light field windows on the EPIs obtained from the sparse light field data to estimate the light field scene flow temporal correspondence.
Each scene point can be represented by an oriented window in the light field ray space. The shear of the ray is related to the depth of the 3D point and the size of the window is defined by spatial and angular Gaussian weights. However the ray moves with object motion, hence there is a need to account for shear and translation. The oriented light field window corresponding to a scene point for a light field, L is computed as follows:

$$O_{d,x0,y0}(x, y, u, v) = (W * S_d * T_{x0,y0})\,[L] \tag{1}$$

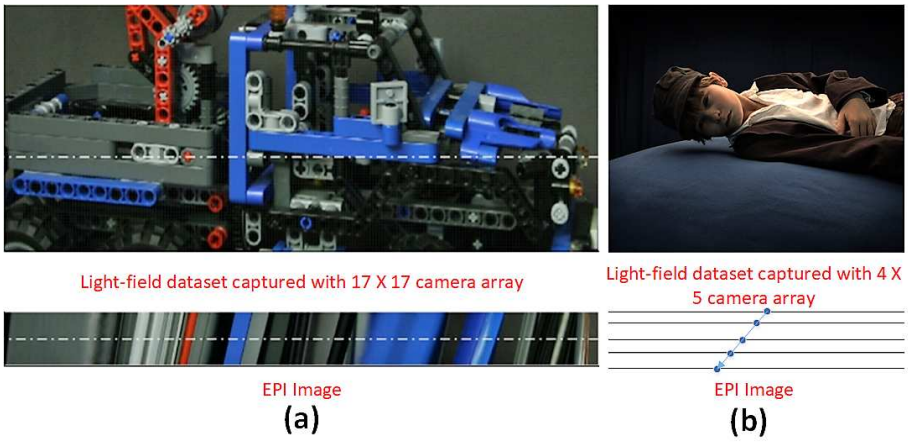

Fig. 7: Illustration of EPI from (a) dense Stanford dataset [34] (EPI from [41]) and (b) sparse light field capture.

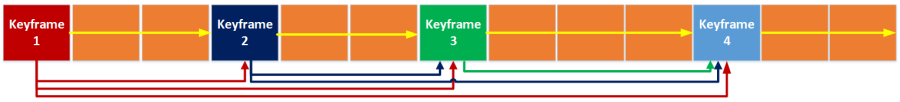

Fig. 8: Illustration of dense correspondence between frames and key-frames for full sequence 4D alignment: arrows indicate sparse correspondence tracks and dense correspondence.

where, x_0, y_0 is the centre of the window, x, y represents the image plane and u, v represents the camera plane. S_d is the shear operation for depth d, defined as $S_d\left[L\right] = L_d(x, y, u, v) = L(x + u\left(1 - \frac{1}{d}\right), y + v\left(1 - \frac{1}{d}\right), u, v)$, $T_{x0,y0}$ is the translation operator defined as, $T_{x0,y0}\left[L\right] = L_{x0,y0}(x, y, u, v) = L(x + x0, y + y0, u, v)$ and W is the Gaussian weighted windows defined as: $W\left[L\right] = L(x, y, u, v)N(x, y; \sigma^2_{xy})A(u, v)$. $N()$ is the 2D Gaussian distribution windows and $A(u, v)$ is the corresponding row and column of the camera array (shown as green cameras in Figure 6, 3^{rd} step) of the reference light field view ((u_c, v_c) shown in red box), defined as:

$$A(u, v) = \begin{cases} 1, & \text{if } v = v_c \text{ or } u = u_c \\ 0, & otherwise \end{cases}$$

This formulation could be easily extended to dense camera array. We propose to use this oriented light field window to robustly estimate the dense scene flow for light field video.

Dense light field scene flow Oriented light field windows are used to estimate flow between consecutive frames and between key-frames, Figure 8, using the sparse temporal feature correspondence as initialization for each view. Flow is estimated on the object region for each light field view, as illustrated in Figure 9, to obtain dense temporal correspondence.

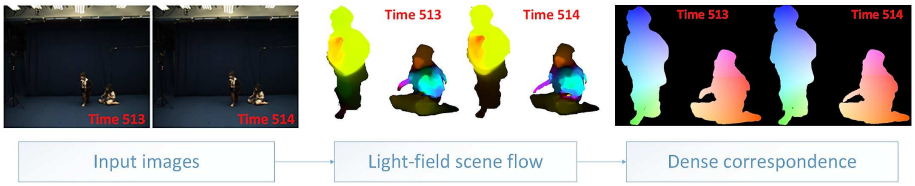

Fig. 9: Light field scene flow for two consecutive frames.

The flow is formulated as a translation of each pixel location $p = (x_p, y_p)$ in image I by $m_p = (\delta x_p, \delta y_p)$ in time. We formulate the computation of flow $\boldsymbol{M}$ per view for each dynamic object by minimization of the cost function:

$$E(\boldsymbol{M}) = \sum_{p \in I} \lambda_L E_L(p, m_p) + \lambda_C E_C(p, m_p) + \lambda_R E_R(p, m_p) \tag{2}$$

The cost consists of three terms: the light field consistency E_L for the oriented light field window alignment; the appearance term E_C for brightness coherency; and the regularization term E_R to avoid sudden peaks in flow and maintain the consistency. Colour and regularization terms are common to optical flow problems [36] and the light field consistency is introduced for the first time to improve dense flow for sparse light field video.

Light field Consistency: The 2D EPIs obtained from the sparse light field views are used to define oriented light field windows for each scene point. These windows encapsulate the observed multi-view light field appearance of the corresponding surface point and can be matched over time to estimate the temporal correspondence, defined as:

$$E_L(p, m_p) = \left\| O_{d_t, p}(p, u, v, t) - O_{d_{t+1}, p+m_p}(p, u, v, t+1) \right\|^2$$

where $O_{d_t, m}()$ is the oriented light field window at time t with depth d_t as defined in Equation 1.

Appearance Consistency: This adds the brightness consistency assumption to the cost function generalized for all N_v light field cameras for both time steps. This term is obtained by integrating the sum of three penalizers over the reference image domain. $e_C^T()$ penalizes deviation from the brightness constancy assumption in time for same views; $e_C^V()$ penalizes deviation from the brightness constancy assumption between the reference view and each of the other views at time $t+1$. $e_C^S()$ forces the flow to be close

to nearby sparse temporal correspondences.

$$E_C(p, m_p) = e_C^T(p, m_p) + e_C^V(p, m_p) + e_C^S(p, m_p)$$
$$e_C^T(p, m_p) = \sum_{i=1}^{N_v} \|(I_i(p, t) - I_i(p + m_p, t + 1))\|^2$$
$$e_C^V(p, m_p) = \sum_{i=2}^{N_v} \|(I_1(p, t) - I_i(p + m_p, t + 1))\|^2$$
$$e_C^S(p, m_p) = \sum_{i=1}^{N_v} e_C^S = \begin{cases} 0 & \text{if } p \in N \\ \infty & otherwise \end{cases}$$

where $I_i(p, t)$ is the intensity at point p and time t in camera i. This term denotes that the flow vector m is located within a window from a sparse constraint at p and it forces the flow to approximate the sparse 2D temporal correspondence tracks.
Regularization: This penalizes the absolute difference of the flow field to enforce motion smoothness and handle occlusions and areas with low confidence.

$$E_R(p, m_p) = \sum_{p,q \in N_p} \|\Delta m\|^2 \, (\lambda_R^L e_R^L(p, q, m_p, m_q) + \lambda_R^C e_R^C(p, q, m_p, m_q))$$
$$e_R^L(p, q, m_p, m_q) = \underset{q \in N_p}{mean} E_L(q, m_q) - \underset{q \in N_p}{min} E_L(q, m_q)$$
$$e_R^C(p, q, m_p, m_q) = \underset{q \in N_p}{mean} E_C(q, m_q) - \underset{q \in N_p}{min} E_C(q, m_q)$$

where $\Delta m = m_p - m_q$ and we compute e_R^L and e_R^C as the minimum subtracted from the mean data energy within the search window N_p for each pixel p.
Occlusions: To detect occlusions, we compare the forward flow from t to $t + 1$, and the backward flow from $t + 1$ to t. Occluded pixels are robustly indicated by large differences in the forward/backward motion and excluded as outliers.
Optimization: For each pixel p, the energy is estimated as defined in Equation 2 on a window N_p, as in the SimpleFlow algorithm [36] to estimate the flow vector m. The optical flow is optimized over a multi-scale pyramid with warping between pyramid levels, resulting in a coarse-to-fine strategy that allows the estimation of large displacements by minimizing the equation defined as: $F(\boldsymbol{M}) = \arg\min E(\boldsymbol{M})$

4D temporally coherent light field video The estimated dense flow for each view is back projected to the 3D visible surface to establish dense 4D correspondence between frames (N_f^i) and between key-frames K_i as seen in Figure 8 to obtain 4D temporally coherent light field video. Dense 4D correspondence is first obtained for the light field view with maximum visibility of 3D points. To increase surface coverage correspondences are added in order of visibility of 3D points for different sparse light field views. Dense temporal correspondence is propagated to new surface regions as they appear using the sparse feature correspondence tracks and respective dense light field scene flow. Dense scene flow is estimated between key-frames for robust long-term surface tracking.

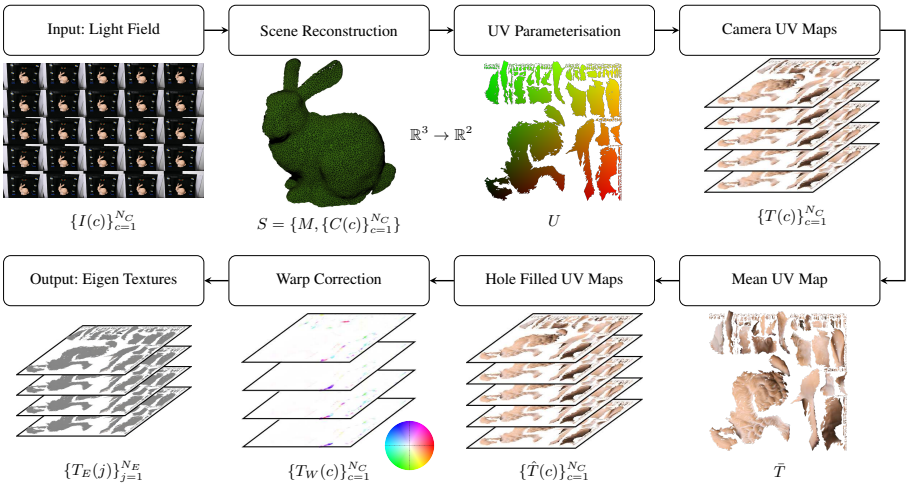

Fig. 10: Eigen texture representation of light fields.

3.4 Eigen Texture Representation of Light Field Images

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 10. Given the input light field images, a geometric proxy is reconstructed and UV coordinates are generated. Camera calibration parameters and the 3D reconstruction are used to generate a UV map for each camera used to capture the light field. Camera UV maps are processed to ensure there are a consistent number of pixels by filling in any holes caused by occlusion or viewpoint changes. 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.

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 [44,34]. 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 [46,13,3]. 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 $M = \{V, Y, U\}$, where V is the 3D mesh vertices, Y 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 [20].

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 11 (b). 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 11 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 3.4, it is required that all UV maps have an equal number of visible pixels. To this end, we first construct an average 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 3. 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 $T(c)$.

$$\bar{T}(l) = \frac{1}{\sum_{c=1}^{N_C} v(c,l)} \sum_{c=1}^{N_C} v(c,l) T(c,l) \tag{3}$$

where $\bar{T}$ is the average over all N_C camera UV maps, $\bar{T}(l)$ is the average value for the l^{th} pixel, $v(c,l)$ 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 4.

$$\hat{T}(c,l) = \begin{cases} T(c,l), & \text{if } v(c,l) = 1 \\ \bar{T}(l), & \text{otherwise} \end{cases} \tag{4}$$

where $T(c,l)$ and $\hat{T}(c,l)$ are values of the l^{th} pixel of camera c for the camera UV map and hole filled UV map, respectively. $\bar{T}(i)$ is the l^{th} pixel in the mean texture map and $v(c,l)$ returns the value in the alpha channel that determines the surface visibility.

To account for small errors in camera parameters and geometry estimation we employ optical flow image warping [11]. 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 $\hat{T}(c_{ref})$.

Eigen Texture Construction In this section, we describe the construction of the Eigen texture representation for light field data. We define the function $\hat{T}(c,l,\lambda)$ which re-

turns a scalar value for the l^{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 5.

$$\mathbf{x}(c,l) = \Big[\hat{T}(c,l,\lambda_r) - \bar{T}(l,\lambda_r), \hat{T}(c,l,\lambda_g) - \bar{T}(l,\lambda_g), \hat{T}(c,l,\lambda_b) - \bar{T}(l,\lambda_b)\Big] \tag{5}$$

where $\mathbf{x}(c,l)$ returns intensity values for the red (λ_r), green (λ_g) and blue (λ_b) image channels for the l^{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 6.

$$\mathbf{x}(c) = [\mathbf{x}(c,1), \mathbf{x}(c,2)..., \mathbf{x}(c,N_P)] \tag{6}$$

where $\mathbf{x}(c)$ returns a row-vector consisting of the RGB pixel intensities for all N_P visible pixels (i.e. $v(c,k) = 1$) 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 7.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(N_C) \end{bmatrix} \tag{7}$$

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 8. As the number 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 [37].

$$\mathbf{X}^\top\mathbf{X} = \mathbf{W}\Sigma\mathbf{W}^\top \tag{8}$$

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

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

Fig. 11: 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.

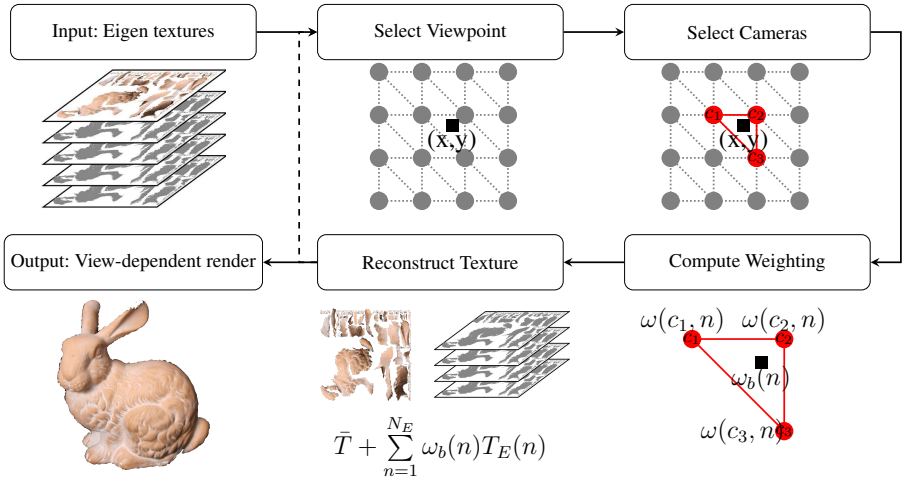

Fig. 12: Light field Eigen texture rendering.

Eigen Texture Rendering Here we discuss the Eigen texture rendering pipeline, an overview is shown in Figure 12. The Eigen textures $\{T_E(n)\}_{n=1}^{N_E}$ and per camera reconstruction weights $\{\{\omega(c,n)\}_{c=1}^{N_C}\}_{n=1}^{N_E}$ 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 9.

$$\omega_b(n) = b_1\omega(c_1,n) + b_2\omega(c_2,n) + b_3\omega(c_3,n) \tag{9}$$

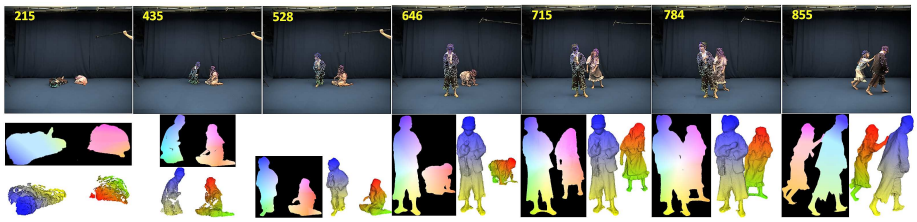

Fig. 13: Dense 2D and 4D correspondence for Walking sequence across key-frames.

where $b_{\{1,2,3\}}$ are the barycentric weights, $\omega(c_{\{1,2,3\}}, n)$ are the Eigen texture reconstruction parameters for cameras $c_{\{1,2,3\}}$ and $\omega_b(n)$ 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 10.

$$T_v = \bar{T} + \sum_{n=1}^{N_E} \omega_b(n) T_E(n) \tag{10}$$

where $\bar{T}$ is the mean texture, $T_E(n)$ are the computed Eigen textures, $\omega_b(n)$ are the barycentric weighted reconstruction weights corresponding to the n^{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 and qualitative evaluations of both the temporal alignment and eigen-texture representation of light field video. The eigen texture representation is evaluated in terms of storage reduction, rendering quality and rendering performance. All presented results were generated using a desktop PC with an Intel i7 CPU, 64GB of RAM and a Nvidia Geforce GTX 1080 GPU.

4.1 4D Light Field Video

The proposed approach is tested on various light field captures. The properties of the evaluation datasets are presented in Table 1. Algorithm parameters set empirically are constant for all results. Sparse and dense correspondence are obtained on the sparse light field dynamic data and the colour coded results are shown in Figure 13 for Walking dataset and in Figure 14 for Sitting and Waking up dataset using the method explained in Section 3.3. To illustrate the 2D dense alignment the silhouette of the dense mesh on key-frames is colour coded and the colours are propagated between frames using dense

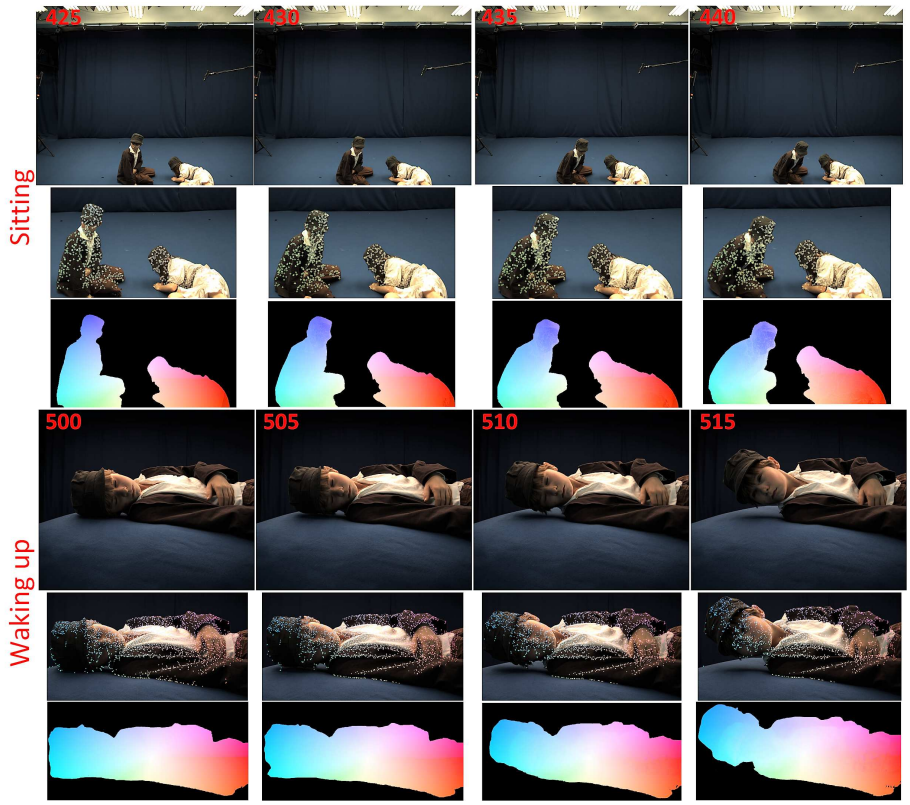

Fig. 14: Sparse temporal correspondences and dense flow results on 2 light field sequences: Sitting and Waking up.

scene flow explained in Section 3.3. Results of the proposed 4D temporal alignment, illustrated in Figure 15 shows that the colour of the points remains consistent between frames. The proposed approach is qualitatively shown to propagate the correspondences reliably over the entire light field video for complex dynamic scenes with large non-rigid motion.

Qualitative evaluation: For comparative evaluation we use:(a) state-of-the-art dense flow algorithm Deepflow [43]; (b) dense flow without light field consistency (DFwLF) in Equation 2; (c) a recent algorithm for alignment of partial surfaces (4DMatch) [26] and (d) Simple flow [36]. Qualitative results against DFwLF, 4DMatch, Deepflow and Simpleflow shown in Figure 16 indicate that the propagated colour map does not remain consistent across the sequence for large motion as compared to the proposed method (red regions indicate correspondence failure).

Quantitative evaluation: For quantitative evaluation we compare the silhouette overlap error (SOE). Dense correspondence over time is used to create propagated mask for each image. The propagated mask is overlapped with the silhouette of the projected surface reconstruction at each frame to evaluate the accuracy of the dense propagation.

Table 1: Properties of all datasets: Length is the length of the sequence, KF gives the number of key-frames detected for the dynamic sequence and Avg. sparse tracks gives the number of sparse temporal correspondence tracks averaged over the entire sequence for each object.

Datasets	Length	Views	Shot level	Resolution	KF	Avg. sparse tracks
Walking	667	20	Far	2448×2048	15	1934
Sitting	694	20	Mid-level	2448×2048	13	1046
Wakingup	270	20	Close-up	2448×2048	7	2083
Running	140	20	Far	2448×2048	5	1278
Magician	353	20	Close-up	2448×2048	6	1312

Table 2: Silhouette overlap error for all the datasets. Prop. represents proposed approach, 4DM is 4DMatch, DF is Deepflow and SF is Simpleflow.

Datasets	Prop.	DFwLF	4DM	DF	SF
Walking	**0.45**	0.59	0.58	0.81	1.05
Sitting	**0.51**	0.73	0.71	1.13	1.83
Waking up	**0.39**	0.56	0.53	0.89	1.17
Running	**0.65**	0.87	0.92	1.23	1.95
Magician	**0.59**	0.82	0.83	1.05	1.67

Table 3: Temporal coherence evaluation for Walking dataset against existing methods: S.D. is the standard deviation.

Methods	Frame-to-frame		Keyframe-to-frame	
	Mean	S.D.	Mean	S.D.
Proposed	**3.78**	1.45	**4.34**	1.74
DFwLF	5.93	2.60	7.41	3.73
4DM	5.30	2.12	6.95	3.12
DF	6.28	3.77	15.79	6.36
SF	7.82	4.31	21.92	8.45

The error is defined as: $SOE = \frac{1}{N_v \times N_F} \sum_{i=1}^{N_v} \sum_{c=1}^{N_F} \frac{\text{Area of intersection}}{\text{Area of back-projected mask}}$. Evaluation against the different techniques is shown in Table 2 for all datasets. As observed the silhouette overlap error is lowest for the proposed approach showing relatively high accuracy.

We evaluate the temporal coherence across Walking sequence, by evaluating the variation in appearance for each scene point between frames and between key-frames and frames for state-of-the-art methods, defined as: $\sqrt{\frac{\Delta r^2 + \Delta g^2 + \Delta b^2}{3}}$, where Δ is the difference operator. Evaluation shown in Table 3 against state-of-the-art methods demonstrates the stability of long term temporal tracking for proposed method. Evaluation of the proposed method against dense flow without light field consistency (DFwLF)

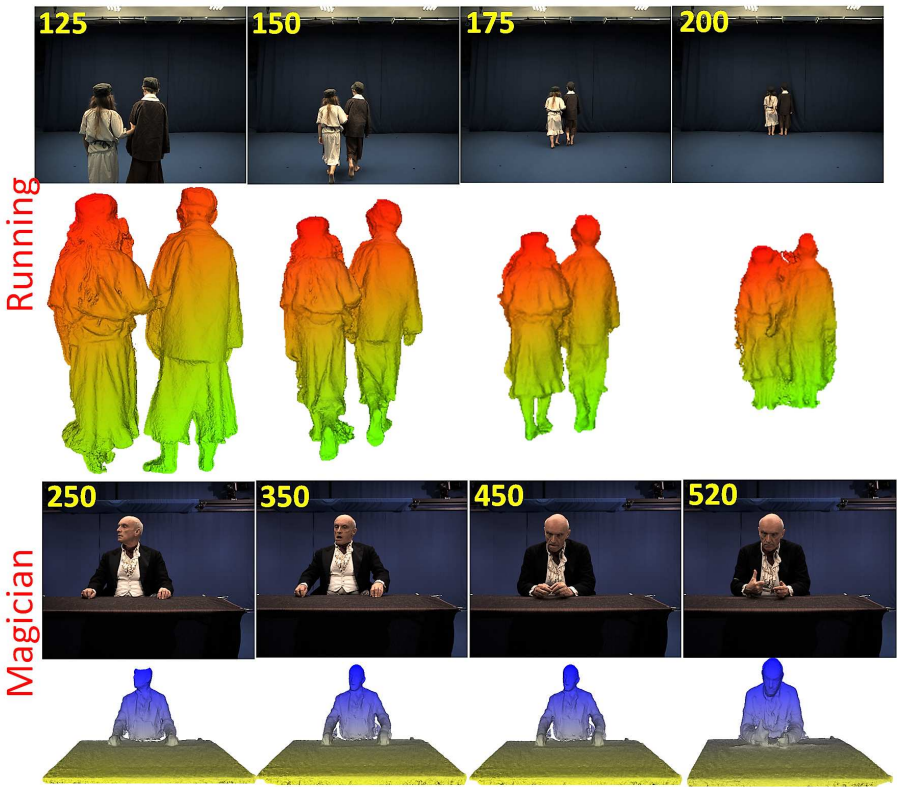

Fig. 15: 4D temporal alignment between frames for Walking and Magician dataset.

demonstrates the usefulness of information from the EPIs in the dense flow in Section 3.3.

Light field camera array configuration: We evaluate the performance of the proposed 4D temporal alignment with different light field camera configurations shown in Figure 17. The completeness of the 3D points at each time instant for all camera configurations as observed in Table 4 is defined as:

$$\frac{100}{MN}\sum_{i=1}^{N}\sum_{c=1}^{M}\frac{\text{Number of 3D points propagated in each configuration}}{\text{Number of 3D points propagated with full camera array}}$$

The evaluation demonstrates a drop in 3D correspondence with the reduction in number of cameras in different camera configurations, specially for close-up shots (Wakeup and Magician). However configuration 1 and 3 with cameras in the corners provide a better coverage compared to 2 and 4.

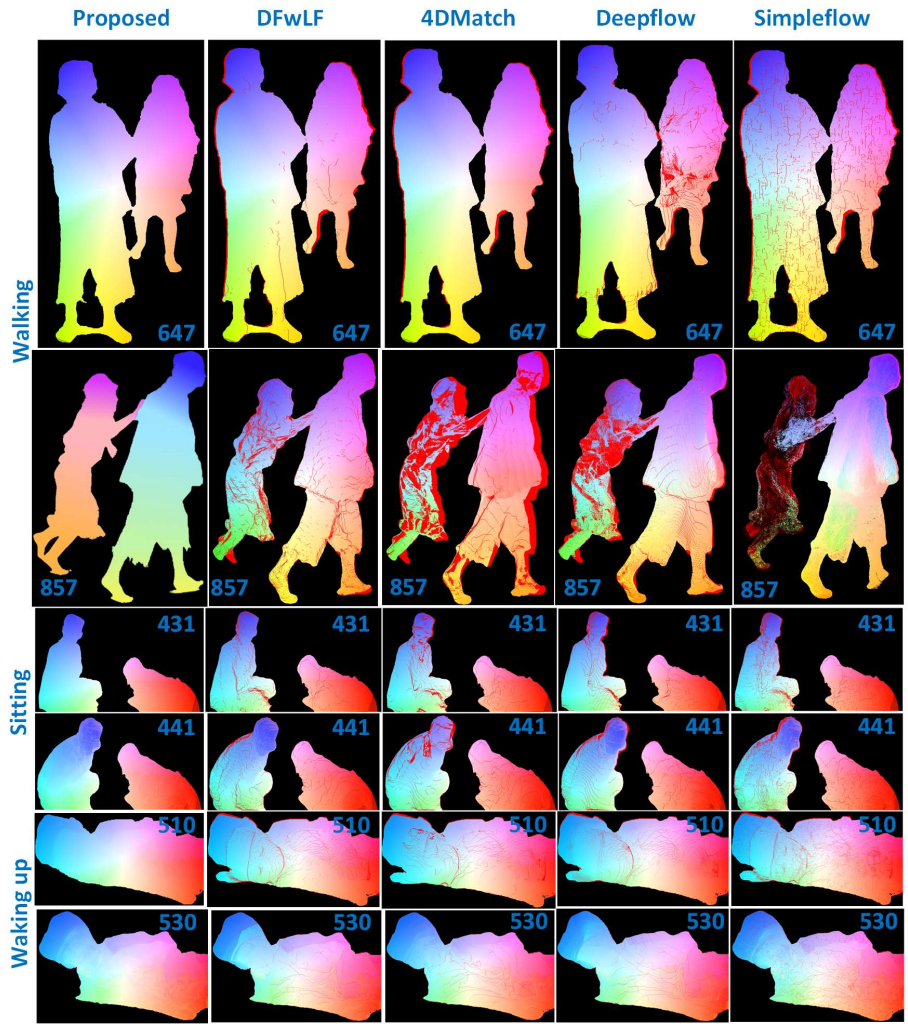

Fig. 16: Dense flow comparison results on different light field sequences.

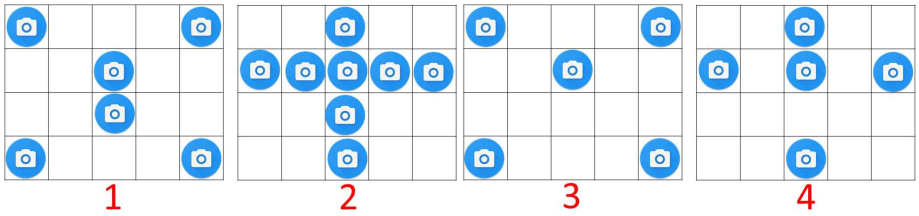

Fig. 17: Different light field camera array configurations.

Table 4: Completeness of dense 3D correspondence averaged over the entire sequence in % for different camera configurations.

Config	Walk	Sit	Wake	Run	Magician
1	95.15	92.58	89.64	91.03	90.82
2	91.33	86.73	86.98	88.78	89.90
3	90.76	85.15	86.05	87.21	86.21
4	87.82	82.40	85.16	86.56	80.91

4.2 Eigen Texture Representation

In this section, we present a quantitative evaluation of the Eigen texture representation. 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. The evaluation is performed on individual frames from the light field video boy and girl datasets. To test the limits of the algorithm, we also use scenes from the Stanford light field archive [34] 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 [15,2], multiple view stereo [32] and Poisson surface reconstruction [17]. For the Stanford datasets [34], 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.

UV Map Reconstruction Quality The UV map reconstruction quality is performed by comparing the hole filled UV maps $\{\hat{T}(c)\}_{c=1}^{N_C}$ to the UV maps reconstructed using the Eigen texture representation with 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) [40] which is considered across all visible foreground surface points. SSIM values range from ± 1 with $+1$ achieved only when comparing identical images.

Figure 18 (a) and (b) 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

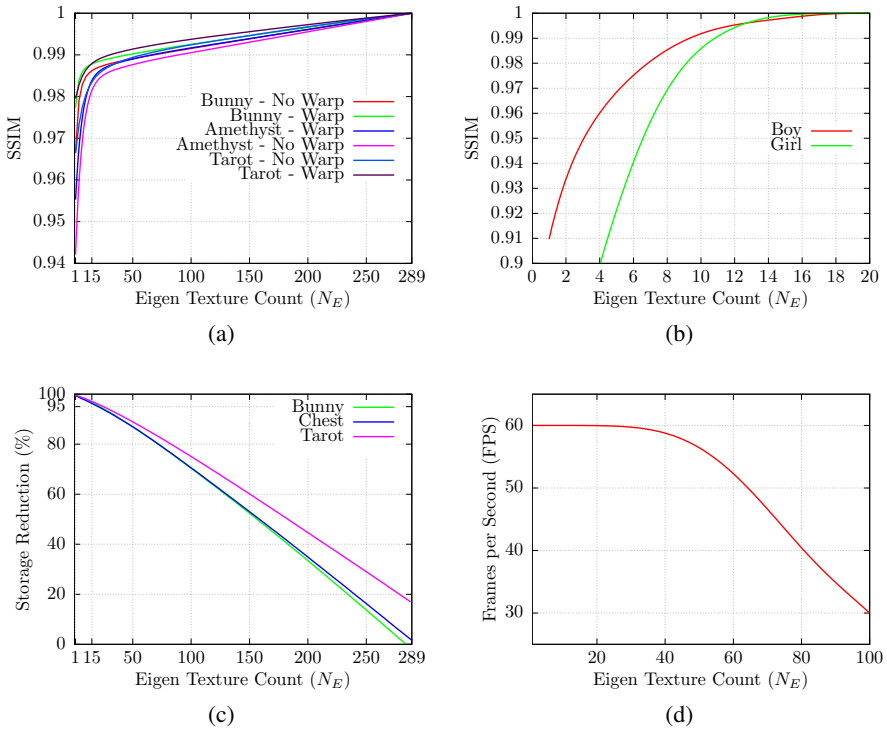

Fig. 18: Quantitative evaluation of the proposed approach: (a) Rendering quality against the number of Eigen textures with and without optical flow correction using scenes from the Stanford light field gallery [34]; (b) Rendering quality against the number of Eigen textures for 5x4 light field dataset; (c) Storage reduction for Stanford light field dataset; (d) Rendering performance against number of Eigen textures.

Amethyst, Bunny and Chest light field datasets consist of 289 images compared to the relatively sparse light field 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 19.

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 18 (a) 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.

Table 5: Storage requirements of light field datasets in represented as images, per camera texture maps and eigen textures.†For Amethyst, Bunny, Chest and Tarot datasets $N_E = 15$, for Boy and Girl datasets $N_E = 7$.

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%)

Storage Table 5 shows the storage requirements for the evaluation datasets and Figure 18 (c) 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%.

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. Figure 18 (d) shows the Eigen texture representation 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 $N_E = 100$.

4.3 Limitations

The proposed method for 4D light field video fails for fast spinning objects; scenes with uniform appearance and highly crowded dynamic environments. This is due to the failure of sparse and dense correspondence due to high ambiguity.

The proposed Eigen texture representation will be unable to effectively compress the appearance in the presence of gross geometric errors. 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 Tarot dataset where the compressed Eigen texture representation is able to reproduce the complex refraction within the glass ball.

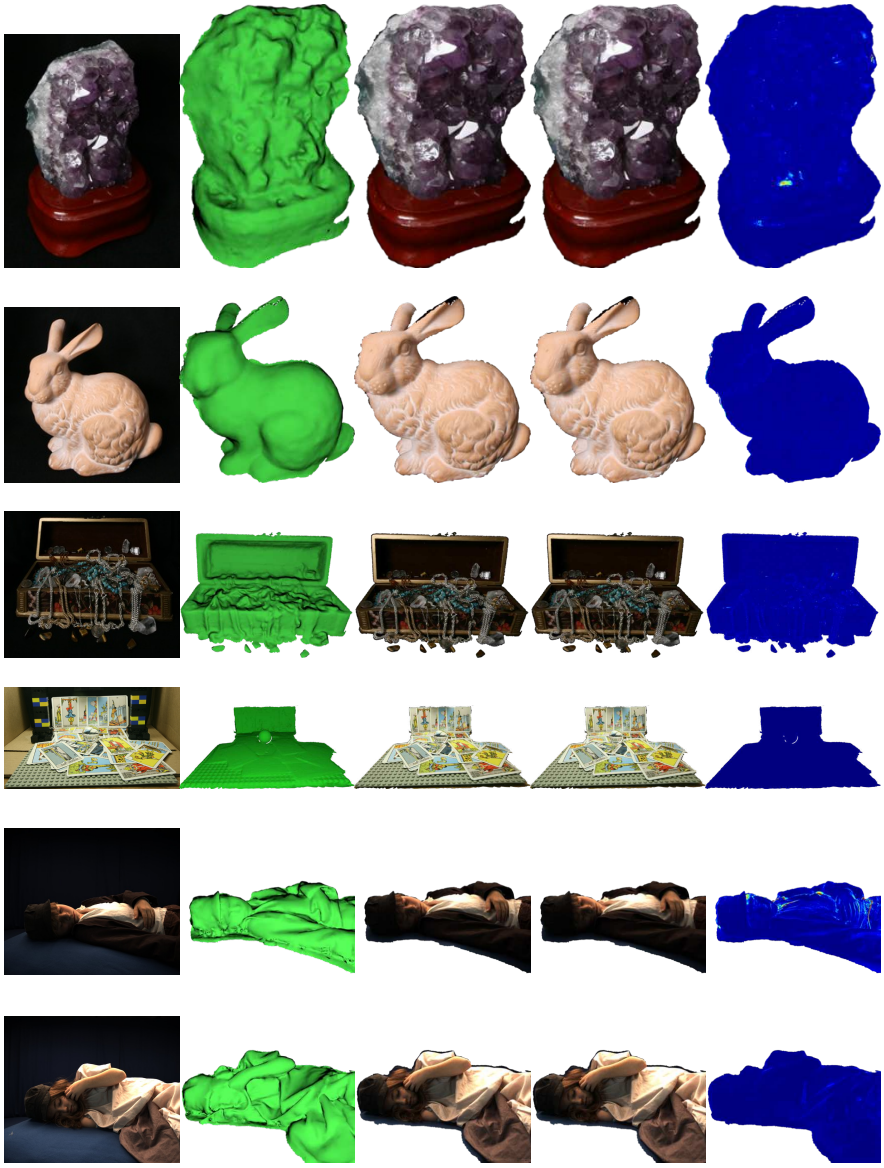

Fig. 19: Results: Rows (top to bottom) show Amethyst, Bunny, Chest, Tarot, Boy, and Girl datasets, respectively. From left to right, selected image from dataset, scene reconstruction model, render using camera UV map, Eigen texture render using $N_E = 15$ for Amethyst, Bunny, and Chest, and $N_E = 7$ 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.

5 Conclusions and Future Work

Light fields have the potential to revolutionise the way we create immersive content. However they also introduce a new set of challenges. Firstly, the light field representation is significantly larger than conventional image/video representations due to the increased dimensionality. This poses challenges in terms of data storage and transmission. An important aspect to ensure acceptance of the technology will therefore be standardisation as well as the development of efficient compression techniques tailored to light field data. Another challenging aspect relates to the development of creative tools dedicated to light field editing. There is currently a lack of software tools for light field post production. Finally, giving the user the ability to freely explore and move in a scene introduces new challenges in terms of storytelling and narratives for VR content production.

This chapter has introduced the first algorithm to obtain a 4D temporally coherent representation of dynamic light field video. A novel method to obtain EPIs from sparse light field video for spatio-temporal correspondence is proposed. Sparse-to-dense light field scene flow is introduced exploiting information from the EPIs. Dense correspondence is fused spatially for 4D temporally coherent light field video. The proposed approach is evaluated on various light field sequences of complex dynamic scenes with large non-rigid deformations to obtain a temporally consistent 4D representation and demonstrating accuracy of the resulting 4D alignment. 4D light field video provides a spatio-temporally coherent representation to support subsequent light field video compression or editing to replicate the functionality of conventional video editing allowing the propagation of edits both spatially across views and temporally across frames.

We have also proposed and demonstrated that the Eigen texture method can be used effectively 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 in representation will investigate creating a deformable mesh for each object for the dynamic sequence in between key-frames. This would create a more complete and efficient 4D representation for immersive content production. Extending this method to more complex crowded scenes will also be investigated by exploiting human pose information. Future work in compression 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.

Acknowledgements

This work was supported by ‘Live action light fields for immersive virtual reality experiences’ (InnovateUK 102686), EPSRC Audio-Visual Media Research Platform Grant

(EP/P022529/1), Royal Academy of Engineering Research Fellowship (RF-201718-17177) and 'Polymersive: Immersive Video Production Tools for Studio and Live Events' (InnovateUK 105168). The authors would also like to thank project partners from Figment Productions and Foundry for the creative direction of the data capture used in this work.

References

1. Adelson, E.H., Bergen, J.R.: The plenoptic function and the elements of early vision. Computational Models of Visual Processing (1991)
2. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org
3. Agisoft: Agisoft Photoscan v1.3.2 (2017), http://www.agisoft.com/
4. Basha, T., Moses, Y., Kiryati, N.: Multi-view scene flow estimation: A view centered variational approach. In: CVPR. pp. 1506–1513 (2010)
5. Boukhayma, A., Tsiminaki, V., Franco, J.S., Boyer, E.: Eigen Appearance Maps of Dynamic Shapes. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) European Conference on Computer Vision. pp. 230–245. Lecture Notes in Computer Science, Springer International Publishing (2016). doi:10.1007/978-3-319-46448-0_14, http://link.springer.com/10.1007/978-3-319-46448-0{_}14
6. Cha Zhang, Jin Li: Compression of lumigraph with multiple reference frame (mrf) prediction and just-in-time rendering. In: Proceedings DCC 2000. Data Compression Conference (2000). doi:10.1109/DCC.2000.838165
7. Chen, W.C., Bouguet, J.Y., Chu, M.H., Grzeszczuk, R.: 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). doi:10.1145/566570.566601, http://doi.acm.org/10.1145/566570.566601
8. Chuo-Ling Chang, Xiaoqing Zhu, Ramanathan, P., Girod, B.: Light field compression using disparity-compensated lifting and shape adaptation. IEEE Transactions on Image Processing **15**(4), 793–806 (2006). doi:10.1109/TIP.2005.863954
9. Davis, A., Levoy, M., Durand, F.: Unstructured light fields. Comput. Graph. Forum **31**(2pt1), 305–314 (May 2012). doi:10.1111/j.1467-8659.2012.03009.x, http://dx.doi.org/10.1111/j.1467-8659.2012.03009.x
10. Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 1858–1865 (2008)
11. Farneb, G.: Two-Frame Motion Estimation Based on Polynomial Expansion. Lecture Notes in Computer Science **2749**(1), 363–370 (2003). doi:10.1007/3-540-45103-X_50, http://link.springer.com/chapter/10.1007/3-540-45103-X{_}50
12. FLIR: Grasshopper3. https://www.flir.co.uk/products/grasshopper3-usb3/?model=GS3-U3-51S5C-C
13. Fuhrmann, S., Langguth, F., Goesele, M.: MVE-A Multi-View Reconstruction Environment. Eurographics Workshop on ... pp. 11–18 (2014). doi:10.2312/gch.20141299, http://diglib2.eg.org/EG/DL/WS/GCH/GCH2014/011-018.pdf.abstract.pdf;internal{&}action=action.digitallibrary.ShowPaperAbstract
14. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. pp. 43–54. SIGGRAPH '96, ACM, New York, NY, USA (1996). doi:10.1145/237170.237200, http://doi.acm.org/10.1145/237170.237200
15. Itseez: Open source computer vision library v2.4. http://opencv.org/ (2017)

16. Joo, H., Liu, H., Tan, L., Gui, L., Nabbe, B., Matthews, I., Kanade, T., Nobuhara, S., Sheikh, Y.: Panoptic studio: A massively multiview system for social motion capture. In: ICCV (2015)
17. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing. pp. 61–70. SGP '06, Eurographics Association (2006), http://dl.acm.org/citation.cfm?id=1281957.1281965
18. Kemp, M.: 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), https://search.library.wisc.edu/catalog/999923957902121
19. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. pp. 31–42. SIGGRAPH '96, ACM, New York, NY, USA (1996). doi:10.1145/237170.237199, http://doi.acm.org/10.1145/237170.237199
20. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics **21**(3), 362–371 (2002). doi:10.1145/566654.566590, http://dl.acm.org/citation.cfm?id=566590
21. Lytro: Lytro immerge. https://www.lytro.com/immerge
22. Magnor, M., Girod, B.: Data compression for light-field rendering. IEEE Transactions on Circuits and Systems for Video Technology **10**(3), 338–343 (2000). doi:10.1109/76.836278
23. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: CVPR (2015)
24. Miller, G., Rubin, S., Ponceleon, D.: Lazy decompression of surface light fields for precomputed global illumination. In: Drettakis, G., Max, N. (eds.) Rendering Techniques '98. Springer Vienna (1998)
25. Mustafa, A., Hilton, A.: Semantically coherent co-segmentation and reconstruction of dynamic scenes. In: CVPR (2017)
26. Mustafa, A., Kim, H., Hilton, A.: 4d match trees for non-rigid surface alignment. In: ECCV (2016)
27. Mustafa, A., Volino, M., Guillemaut, J., Hilton, A.: 4d temporally coherent light-field video. In: 2017 International Conference on 3D Vision (3DV). pp. 29–37 (2017). doi:10.1109/3DV.2017.00014
28. Ng, R., Levoy, M., Duval, G., Horowitz, M., Hanrahan, P.: Light Field Photography with a Hand-held Plenoptic Camera. Computer Science Technical Report CSTR (2005). doi:10.1.1.163.488
29. Nishino, K., Sato, Y., Ikeuchi, K.: 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). doi:10.1109/34.969116
30. Overbeck, R.S., Erickson, D., Evangelakos, D., Pharr, M., Debevec, P.: A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. ACM Trans. Graph. **37**(6), 197:1–197:15 (2018). doi:10.1145/3272127.3275031, http://doi.acm.org/10.1145/3272127.3275031
31. Rusu, R.B.: Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. Ph.D. thesis, Computer Science department, Technische Universitaet Muenchen, Germany (2009)
32. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: CVPR. pp. 519–528 (2006)
33. Srinivasan, P., Tao, M., Ng, R., Ramamoorthi, R.: Oriented light-field windows for scene flow. In: ICCV (December 2015)
34. Stanford Graphics Laboratory: The (New) Stanford Light Field Archive (2008), http://lightfield.stanford.edu/
35. Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In: ECCV. pp. 438–451 (2010)

36. Tao, M.W., Bai, J., Kohli, P., Paris, S.: Simpleflow: A non-iterative, sublinear optical flow algorithm. Computer Graphics Forum (Eurographics 2012) **31**(2) (May 2012), http://graphics.berkeley.edu/papers/Tao-SAN-2012-05/
37. Turk, M., Pentland, A.: Eigenfaces for recognition. J. Cognitive Neuroscience **3**(1), 71–86 (Jan 1991). doi:10.1162/jocn.1991.3.1.71, http://dx.doi.org/10.1162/jocn.1991.3.1.71
38. Viola, I., Řeřábek, M., Ebrahimi, T.: Comparison and evaluation of light field image coding approaches. IEEE Journal of Selected Topics in Signal Processing **11**(7), 1092–1106 (2017). doi:10.1109/JSTSP.2017.2740167
39. Volino, M., Mustafa, A., Guillemaut, J.Y., Hilton, A.: Light field compression using eigen textures. In: International Conference on 3D Vision (3DV) (2019)
40. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). doi:10.1109/TIP.2003.819861, http://www.ncbi.nlm.nih.gov/pubmed/15376593http://ieeexplore.ieee.org/document/1284395/
41. Wanner, S., Goldluecke, B.: Globally consistent depth labeling of 4D light fields. In: CVPR. pp. 41–48 (June 2012)
42. Wedel, A., Brox, T., Vaudrey, T., Rabe, C., Franke, U., Cremers, D.: Stereoscopic scene flow computation for 3D motion understanding. IJCV **95**, 29–51 (2011)
43. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: Large displacement optical flow with deep matching. In: ICCV. pp. 1385–1392 (2013)
44. Wilburn, B., Joshi, N., Vaish, V., Talvala, E.V., Antunez, E., Barth, A., Adams, A., Horowitz, M., Levoy, M.: High performance imaging using large camera arrays. ACM Trans. Graph. **24**(3), 765–776 (2005). doi:10.1145/1073204.1073259, http://doi.acm.org/10.1145/1073204.1073259
45. Wood, D.N., Azuma, D.I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D.H., Stuetzle, W.: Surface light fields for 3D photography. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques pp. 287–296 (2000). doi:10.1145/344779.344925, http://doi.acm.org/10.1145/344779.344925
46. Wu, C.: Towards linear-time incremental structure from motion. Proceedings - 2013 International Conference on 3D Vision, 3DV 2013 pp. 127–134 (2013). doi:10.1109/3DV.2013.25
47. Yücer, K., Sorkine-Hornung, A., Wang, O., Sorkine-Hornung, O.: Efficient 3D object segmentation from densely sampled light fields with applications to 3D reconstruction. ACM Transaction in Graphics **35**(3), 22:1–22:15 (March 2016). doi:10.1145/2876504, http://doi.acm.org/10.1145/2876504
48. Zanfir, A., Sminchisescu, C.: Large displacement 3d scene flow with occlusion reasoning. In: ICCV (2015)
49. Zheng, E.and Ji, D., Dunn, E., Frahm, J.M.: Sparse dynamic 3D reconstruction from unsynchronized videos. In: ICCV (2015)