Dataset and Pipeline for Multi-View Light-Field Video

N. Sabater[1], G. Boisson[1], B. Vandame[1], P. Kerbiriou[1], F. Babon[1],
M. Hog[1,2], R. Gendrot[1], T. Langlois[1], O. Bureller[1], A. Schubert[1] and V, Allié[1].
[1] Technicolor Research & Innovation
[2] INRIA
Name.Surname@technicolor.com

Abstract

The quantity and diversity of data in Light-Field videos makes this content valuable for many applications such as mixed and augmented reality or post-production in the movie industry. Some of such applications require a large parallax between the different views of the Light-Field, making the multi-view capture a better option than plenoptic cameras. In this paper we propose a dataset and a complete pipeline for Light-Field video. The proposed algorithms are specially tailored to process sparse and wide-baseline multi-view videos captured with a camera rig. Our pipeline includes algorithms such as geometric calibration, color homogenization, view pseudo-rectification and depth estimation. Such elemental algorithms are well known by the state-of-the-art but they must achieve high accuracy to guarantee the success of other algorithms using our data. Along this paper, we publish our Light-Field video dataset that we believe may be of special interest for the community [1]. We provide the original sequences, the calibration parameters and the pseudo-rectified views. Finally, we propose a depth-based rendering algorithm for Dynamic Perspective Rendering.

1. Introduction

Since the introduction of the concept of *integral photography* [20], tremendous advances on Light-Fields have been done on the computational photography community. In particular, the availability of plenoptic cameras such as Lytro[1] or Raytrix[2] has originated the bloom of new research on the field during the last years, being now a very mature topic. Besides plenoptic cameras, Light-Fields can also be captured with camera arrays [33], robotic arms [19] or hand held cameras [7]. However, each acquisition system samples the plenoptic function (light rays in the three-dimensional space) [2] very differently. Indeed, plenoptic cameras produce great angular resolution at the cost of reducing the spatial resolution. On the contrary, multi-view systems have good spatial resolution but usually do not have many available views. Existing multi-view Light-Field systems with a great number of views are generally impractical due to the amount of data and the complexity of the capturing system. So, in general, plenoptic cameras capture *dense* and *narrow-baseline* Light-Fields and multi-view systems capture *sparse* and usually *wide-baseline* Light-Fields. While all Light-Field acquisition systems share the same theoretical principle, depending on the application, one type of acquisition or another would be preferred. Indeed, the baseline, the resolution and the number of views makes each acquisition system very specific and suitable for the needs of a given application. As a consequence, due to the data variability, processing algorithms need to be specifically tailored for each acquisition system. Besides the spatio-angular resolution, another particularity of the Light-Field acquisition systems is the capacity to capture video.

In terms of applications, the availability of wide-baseline Light-Field videos opens the door to new possibilities compared to conventional cameras. For example, 3D Television (3DTV), Free-view Television (FTV) [36], or mixed and augmented reality as proposed by MagicLeap[3] or Microsoft Hololens[4]. In particular, Light-Field videos are fundamental when the inserted content is not Computer Generated (CG) and the goal is to produce a plausible and immersive experience.

In this paper we focus on camera arrays as a video Light-Field capturing system. In particular we present a 4×4 synchronized camera rig. Our system belongs to the multi-view category and shares the same assumptions than [10] concerning the captured scenes. This is, we assume to capture Lambertian textured surfaces. However, we would like to make the difference between the general multi-view frame-

[1] www.lytro.com
[2] www.raytrix.de
[3] www.magicleap.com
[4] www.microsoft.com/microsoft-hololens

work in [10] and a Light-Field multi-view setup. The difference remains in the density of views (and the number of light rays imaging each point of the scene) which requires different algorithms in order to optimally process the data.

2. Related Work

Existing Light-Field datasets, [14] [33] [25] [18] [26], either synthetic or from real acquisition systems (plenoptic cameras, camera arrays or gantries) are essentially still Light-Fields. The only exception is the video Light-Field dataset recently proposed by Dabala et al. [6] which turns out to be the closest work to ours, since it also presents a pipeline for camera rigs. Our pipeline, though, takes into account color homogenization and the precise geometric position of the cameras given by calibration, which allows to relax some constraints on the depth estimation. The paper of Basha et al. [3] also deals with multi-view video. The authors propose to jointly estimate the 3D motion (scene flow) and the 3D reconstruction of the scene captured with a camera rig assumed to be calibrated and having a small baseline.

Pipelines for plenoptic cameras have also been proposed [13] [29] [5] but due to the different nature of plenoptic Light-Field data compared with camera rigs, the algorithms are sorely different.

Geometric calibration is a well studied problem [12] but it is generally not addressed by multi-view pipelines even if it is of paramount importance for the accuracy of the sequel processing. Camera manufacturers do not provide this information neither, specially when their cameras are used to build camera rigs. Previous work on multi-camera calibration includes [31] that studies a calibration method for planar camera arrays and [34] that assumes a more general camera setup but imposes a rigid constraint between the viewpoints. Other techniques specifically developed for Structure from Motion (SfM) such as Sparse Bundle Adjustment [21] can also provide multi-view calibration.

Regarding color calibration, when the cameras are not known, a family of algorithms using image correspondences allows to tonally stabilize videos [9] or to color homogenize different cameras of the same scene [32]. With the same philosophy, [23] uses spatio-temporal correspondences for multi-view video color correction. Nevertheless, in our pipeline we exploit the fact that we have full knowledge of our cameras and the capture setup. So we have an approach more similar to [17] in which a method for calibrating large camera arrays is presented.

Camera arrays have more capabilities compared to conventional images or video as it has been proved in a number of related papers. For instance, tracking through occlusions [16], multi-object detection [28], reconstructing occluding surfaces [30] [27] or creating All-In-Focus images [35]. Another application of multi-view systems concern Synthetic Aperture refocusing but the reduced number of available views creates angular aliasing. In [15] an algorithm for fast realistic refocusing for sparse Light-Fields is presented.

All the above applications share the fact that they estimate and exploit the depth map of the captured scene. More precisely, in [27], depth estimation is formulated as an energy minimization problem with an intensity consistency and a smoothness term. In [35] a Light-Field visibility term is also considered in the energy. In [30] different cost functions for large camera arrays are compared in terms of robustness to occlusions. It is interesting to point out that most of the proposed methods estimate the depth for a viewpoint that is not necessarily one of the available viewpoints in the Light-Field. We have observed that this strategy is more prone to errors and instead we estimate a depth map for each available view-point in the Light-Field.

Finally, in Lu et al. [22] a survey on multi-view video synthesis and editing is presented.

3. Pipeline

In this paper we consider the 2 plane parameterization as in [11][19]. So, the 4D Light-Field $L(s,t,u,v)$ is parameterized such that each view (s,t) has pixels (u,v). We also consider that the light rays coming from the same scene point should be captured with the same radiance in the different views when the object is Lambertian. This is, corresponding pixels from different views should have the same color. As a consequence, we have included in our pipeline a color homogenization step. Besides, our camera rig has carefully been calibrated. Calibration parameters are used to project the images to a reference plane while removing its distortion. We call such images pseudo-rectified images to differentiate them from epipolar rectified images. Our strategy has the advantage that point correspondences between images can be found with simple translations without the need to deproject in the space and project in a new view each image point, which accelerates our pipeline considerably.

Our pipeline also includes a depth estimation step, which provides a depth for each camera. Our algorithm is multiscale and uses all images for estimating each depth map.

Finally, we propose a real time algorithm for Light-Field rendering which estimates intermediate (virtual) views of the light-field, so the captured Light-Field sequence can be watched with a dynamic parallax.

3.1. Capture

Our camera rig is made of 16 cameras whose sensors are manufactured by CMOSIS (CMV2000) and packaged by IDS. The 16 cameras are controlled through the UEye API. Our multi-view Light-Field video is fully synchronized. Fig. 1 shows our camera setup.

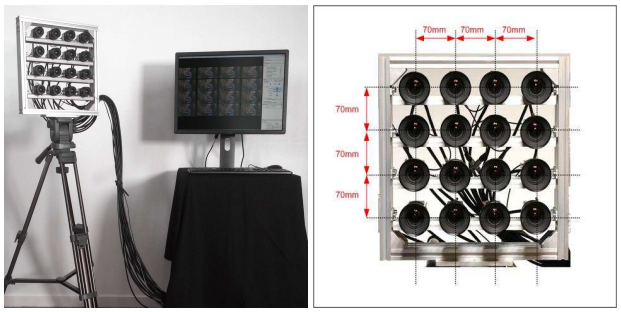


Figure 1. Our camera rig setup.

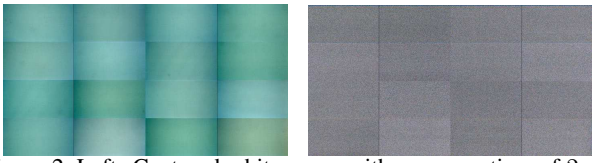


Figure 2. Left: Captured white scene with exposure time of 2ms. Images are demosaicked for the sake of visualization. Before correction the cameras capture a greenish color with many differences for each camera. Right: raw white images after bias and gain map correction. The corrected intensities have been clipped to $[200, 210]$ to better evaluate the similarity. Vignetting is also corrected by applying the gain map.

3.2. Color homogenization and Demosaicking

Let $RAW_{\mathbf{c}} : \Omega \subset \mathbb{N}^2 \to \mathbb{N}^3$ be the $\mathbf{c}$-th captured raw image. In particular,

$$RAW_{\mathbf{c}} = (RAW^r_{\mathbf{c}}, RAW^g_{\mathbf{c}}, RAW^b_{\mathbf{c}}), \tag{1}$$

where $RAW^g_{\mathbf{c}}(u,v) = RAW^b_{\mathbf{c}}(u,v) = 0$ if (u,v) is a red pixel in the Bayer pattern (and respectively for green and blue pixels).

Our goal is to homogenize the color of all captured images with respect to a reference camera $\mathbf{c}_0$. In order to do so, we describe here all the steps that need to be done before and after capturing the sequences.

- **Black level setting** - The black level is a hardware parameter that allows to control the pixel sensitivities in total darkness. It is important to tune this parameter for each camera to correctly capture intensities in low light conditions. Indeed, if the black level is set to 0, the sensor looses information because it records an intensity of 0 in dark scenes instead of low intensities. In order to avoid this to happen, we set our camera rig in total darkness (covering the cameras) and we increment the black level of each camera until 95% of pixels record an intensity different of 0. We have observed that after this manipulation, the black level hardware parameter of each camera is close to 4 and all captured images in total darkness have an intensity between 0 and 10.

- **Bias map estimation** - After the black level setting, we capture a dozen of raw images in total darkness for each camera $\mathbf{c}$. Averaging such captured images for each camera $\mathbf{c}$, we obtain the bias map $B_{\mathbf{c}}$, which records the minimum count for each pixel.

- **Aperture rig calibration** - In order to calibrate the apertures of the cameras, we first set the desired aperture on the reference camera $\mathbf{c}_0$. Afterwards, a flat illuminated led panel covered with a diffuser (white scene) is placed in front of the camera rig, so all cameras observe it while white raw images $W_{\mathbf{c}}$ are captured with the same exposure time. After subtracting the corresponding bias map to the raw white images, the average intensity $\mu_{\mathbf{c}}$ is computed:

$$\mu_{\mathbf{c}} = \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} \sum_{\text{ch}=r,g,b} \big(W^{\text{ch}}_{\mathbf{c}}(u,v) - B^{\text{ch}}_{\mathbf{c}}(u,v)\big). \tag{2}$$

  Finally, the aperture rings of the other cameras $\mathbf{c} \neq \mathbf{c}_0$ are tuned until $\frac{\mu_{\mathbf{c}}}{\mu_{\mathbf{c}_0}} = 1 \pm 0.02$.

- **Gain map estimation** - When the apertures of all cameras are homogeneous, we capture new raw white images for each camera $W_{\mathbf{c},i},\ i = 1, \cdots, N$ and we compute the gain map as:

$$G_{\mathbf{c}}(u,v) = \frac{1}{N} \sum_{i=1}^{N} \frac{W_{\mathbf{c},i}(u,v) - B_{\mathbf{c}}(u,v)}{\mu_{\mathbf{c}_0}}. \tag{3}$$

  Then, the raw image corrected with the bias and gain maps is computed as

$$\overline{RAW}_{\mathbf{c}} = \frac{RAW_{\mathbf{c}} - B_{\mathbf{c}}}{G_{\mathbf{c}}}, \tag{4}$$

  The resulting image is vignetting-free and homogeneous in colors with the remaining images. Fig. 2 shows the 16 captured images with our rig of a white scene before and after bias and gain correction. Note that the intensity values of all cameras after correction are very similar and independently of the color channel, meaning that all color channels are homogeneous. However the bias and gain correction may not be sufficient to have homogeneous colors with a different exposition. Indeed, the gain map is estimated with a reference exposure time but not all cameras have the same response with a different exposure. For this reason, we have a last step in our color homogenization method.

- **Color correction** - In order to be robust to the different exposures, we measure the average $\mu^{ch}_{\mathbf{c}}$ of each color

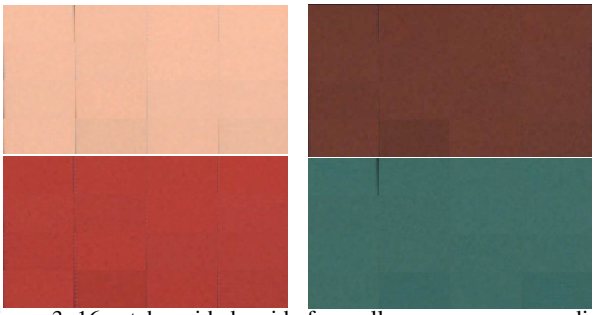


Figure 3. 16 patches side by side from all cameras corresponding to 4 different colors in the MacBeth color chart.

channel ch and each camera $\mathbf{c}$ for different exposures exp. Then, we estimate the regression line via least squares fitting of $\mu_{\mathbf{c}}^{ch}(exp)$ for each ch and $\mathbf{c}$. Let $\alpha_{\mathbf{c}}^{ch}$ and $\beta_{\mathbf{c}}^{ch}$ be the slope and offset of each regression line. In this manner, the color corrected raw images are defined as

$$\widehat{RAW}_{\mathbf{c}}^{ch}(u,v) = \alpha_{\mathbf{c}}^{ch} \cdot \overline{RAW}_{\mathbf{c}}^{ch}(u,v) + \beta_{\mathbf{c}}^{ch}\,. \quad (5)$$

After color homogenization, the images $\widehat{RAW}_{\mathbf{c}}$ are demosaicked using the algorithm in [8] which has proved to outperform with respect to the state-of-the-art. In the sequel, the resulting demosaicked images are noted $I_{\mathbf{c}}^{color}$.

In order to measure the accuracy of our colorimetric correction we have captured a MacBeth color chart. After correcting the images with the aforementioned processing, we have measured the color average of 25×25 homogeneous patches for each color in the MacBeth color chart. We have measured the standard deviation among all views. We have observed that the red channel has a slightly less accurate homogenization ($\sigma^r = 2.2$ compared to $\sigma^g = 0.8$ and $\sigma^b = 0.9$). See Fig. 3 illustrating the color correction for some of the MacBeth colors.

Note that the described method needs to be done once for all. Then, the bias and gain maps, as well as the 16×3 slopes and offsets α and β are registered and used during each capture. However, if the aperture has to be changed, the homogenization of the aperture needs to be done again before the capture. It is worth noting that the procedure for color correction defines a linear correction which follows the assumption of linear sensitivity of the pixels. It is also interesting to point out that our method aims to homogenize the colors and intensities of all cameras with respect to a reference camera but we have not tried to calibrate our rig to a referent illuminant. In the case that we require such a calibration, it would be enough to calibrate the reference camera with the desired illuminant before we run our homogenization method.

3.3. Calibration and Geometry Processing

Our rig has carefully been arranged trying to place the cameras in the same plane, having parallel principal axis and being equidistant (same horizontal and vertical baseline). However, the manual alignment not being perfect, a calibration has been implemented in our pipeline. Intrinsic and extrinsic parameters are estimated with Sparse Bundle Adjustement, based on the software package in [21]. The cameras are calibrated to fit a distorted pinhole projection model similar to the one proposed in [4]. In particular, the calibration module considers a set of corner pixel positions computed from several checkerboard captured images. Considering a camera, we denote $\mathbf{P} = [\mathbf{R} \quad \mathbf{T}] \in \mathbb{R}_{3\times4}$ the camera pose matrix in the World coordinate system and $\mathbf{Q} = [\mathbf{R}^{-1} \quad -\mathbf{R}^{-1}\mathbf{T}] \in \mathbb{R}_{3\times4}$ its extrinsic matrix. Now if $\mathbf{X}_w$ is a 3D point in the World coordinate system and $\mathbf{X}$ is the corresponding point in the camera coordinate system, then $\mathbf{X}_w = \mathbf{P} \cdot \binom{\mathbf{X}}{1}$ and $\mathbf{X} = \mathbf{Q} \cdot \binom{\mathbf{X}_w}{1}$.

Let $\mathbf{K} = \begin{pmatrix} f & \gamma & c_u \\ 0 & \lambda \cdot f & c_v \\ 0 & 0 & 1 \end{pmatrix}$ be the intrinsic matrix of the camera, where f is the distance from the optical center to the sensor expressed in pixels, (c_u, c_v) is the principal point, λ is the aspect ratio, and γ is the skew coefficient. Let $\mathbf{W}$ be the distortion warping operator that affects 3D points projections in the cameras coordinate system. The radial distortion is expressed as a polynomial function in the plane $z = 1$m as $\mathbf{W}\binom{p}{q} = (1 + a_1 r^2 + a_2 r^4)\binom{p}{q}$, where $r = \sqrt{p^2 + q^2}$.

Then, given a 3D point $\mathbf{X}_w$ in the World coordinate system, its projection in pixel coordinates (u, v) in the camera image plane is determined by

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \cdot \begin{pmatrix} \mathbf{W}\binom{p}{q} \\ 1 \end{pmatrix} \quad (6)$$

where

$$\begin{pmatrix} p \\ q \\ 1 \end{pmatrix} \equiv \mathbf{Q} \cdot \begin{pmatrix} \mathbf{X}_w \\ 1 \end{pmatrix}\,. \quad (7)$$

Note that, using homogeneous coordinates

$$\begin{pmatrix} p \\ q \\ 1 \end{pmatrix} \equiv \begin{pmatrix} x \\ y \\ z \end{pmatrix} \iff \begin{cases} p = \frac{x}{z} \\ q = \frac{y}{z} \end{cases} \quad (8)$$

Image Pseudo-Rectification:
After calibration, $\mathbf{K}$, $\mathbf{W}$ and $\mathbf{Q}$ are known for each camera $\mathbf{c}$ which allows to determine for a given depth z correspondent points in different images using Eq. 6 and Eq. 7. However, the projection and deprojection process has a high computational complexity due to the non linear distortion. In order

to accelerate our pipeline, our images are warped such that corresponding points between images are found with simple translations. This assumption stands in our setup because our cameras are almost coplanar.

Formally, let $I_{\mathbf{c}}^{color}$ be the original color images that have been color corrected and demosaicked, where $\mathbf{c} = (s,t); s,t = 0,\dots,3;$ is the camera index of the camera placed at the s-th column and t-th row of the camera array. From $I_{\mathbf{c}}^{color}$ we compute the so-called pseudo-rectified views $I_{\mathbf{c}}$, which are the projections onto a reference camera $\mathbf{c}_0$ at a reference depth z_0 of the original images $I_{\mathbf{c}}^{color}$. More precisely, the pseudo-rectified images $I_{\mathbf{c}}$ are defined at pixel $(u,v) \in \mathbb{N}^2$ as

$$I_{\mathbf{c}}(u,v) = I_{\mathbf{c}}^{color}(u',v')\,, \tag{9}$$

where

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \mathbf{K_c} \cdot \begin{pmatrix} \mathbf{W_c}\begin{pmatrix} p \\ q \end{pmatrix} \\ 1 \end{pmatrix} \tag{10}$$

and

$$\begin{pmatrix} p \\ q \\ 1 \end{pmatrix} \equiv \mathbf{Q_c} \cdot \begin{pmatrix} & \mathbf{P}_{\mathbf{c}_0} & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} z_0\, \widehat{\mathbf{K}_{\mathbf{c}_0}}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \\ 1 \end{pmatrix}, \tag{11}$$

$\mathbf{W_c}$, $\mathbf{K_c}$ and $\mathbf{Q_c}$ being the distortion, the intrinsic and extrinsic matrices of camera $\mathbf{c}$, $\mathbf{P}_{\mathbf{c}_0}$ the pose matrix of the reference camera $\mathbf{c}_0$ and $\widehat{\mathbf{K}_{\mathbf{c}_0}}$ the intrinsic matrix of a virtual pinhole camera derived from $\mathbf{K}_{\mathbf{c}_0}$ which skew coefficient and aspect ratio are respectively set to 0 and 1:

$$\widehat{\mathbf{K}_{\mathbf{c}_0}} = \begin{pmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{pmatrix}. \tag{12}$$

Note that in order to compute $I_{\mathbf{c}}$ in Eq. 9, the images $I_{\mathbf{c}}^{color}$ need to be interpolated since $(u',v') \in \mathbb{R}^2$. In our pipeline, a Lanczos kernel has been used for interpolation. Note also that the projection at a reference depth z_0 of each image $I_{\mathbf{c}}^{color}$ does not ensure the image domains to be equal. This is, (u',v') in Eq. 9 may not belong to the image domain of $I_{\mathbf{c}}^{color}$. In that case, empty pixels are colored with pure green RGB=$(0,255,0)$. Nevertheless, in order to minimize the number of such empty pixels, the reference depth z_0 has been set to an arbitrarily large distance from the cameras.

Using Pseudo-Rectified Images:
With the notations above, let $Z_{\mathbf{c}_0} : \mathbb{N}^2 \to \mathbb{R}$ be the depth map of the reference camera $\mathbf{c}_0$. Then, given a pixel $(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}) \in \mathbb{N}^2$ in the pseudo-rectified image $I_{\mathbf{c}_0}$, its correspondent point $(u_{\mathbf{c}}, v_{\mathbf{c}}) \in \mathbb{R}^2$ in $I_{\mathbf{c}}$ can be found with a simple pixel translation:

$$\begin{pmatrix} u_{\mathbf{c}} \\ v_{\mathbf{c}} \end{pmatrix} = \begin{pmatrix} u_{\mathbf{c}_0} \\ v_{\mathbf{c}_0} \end{pmatrix} + D(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}) \cdot \begin{pmatrix} \delta u_{\mathbf{c}} \\ \delta v_{\mathbf{c}} \end{pmatrix}, \tag{13}$$

where $D : \mathbb{N}^2 \to \mathbb{R}$ is defined as

$$D(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}) = \frac{\frac{1}{Z_{\mathbf{c}_0}(u_{\mathbf{c}_0}, v_{\mathbf{c}_0})} - \frac{1}{z_0}}{\frac{1}{z_1} - \frac{1}{z_0}}, \tag{14}$$

and $(\delta u_{\mathbf{c}}, \delta v_{\mathbf{c}})$ is the disparity shift which corresponds to the shift in pixels between the projected point at depth $z_1 \neq z_0$ and the projected point at the reference depth z_0.

Thanks to the coplanar assumption, for each camera $\mathbf{c}$, the disparity shift $(\delta u_{\mathbf{c}}, \delta v_{\mathbf{c}})$ is constant over the whole image. Nevertheless, if the cameras are not coplanar, Eq. 13 does not stand. For this reason, since our rig may not be perfectly coplanar (Fig. 4-(a)), we have evaluated the different pixels positions when computing exact pixel correspondences via projection and deprojection (Eq. 6 and Eq. 7) and the approximate pixel correspondences via pixel translation (Eq. 13). We have measured that such difference is only of 0.32 pixels in average at a distance of 1.9m, which is totally acceptable, knowing the computational complexity drop of using Eq. 13 instead of Eq. 6 and Eq. 7. Fig. 4-(b) shows the largest and the average error per camera.

Therefore, Synthetic Aperture Refocusing can be computed easily. In particular, a refocused image at depth z can be computed with the provided images $I_{\mathbf{c}}$ for each pixel $(u,v) \in \mathbb{N}^2$ as

$$S^z(u,v) = \frac{1}{16} \sum_{\mathbf{c}} I_{\mathbf{c}}\big(u + d(z)\cdot\delta u_{\mathbf{c}},\; v + d(z)\cdot\delta v_{\mathbf{c}}\big)\,, \tag{15}$$

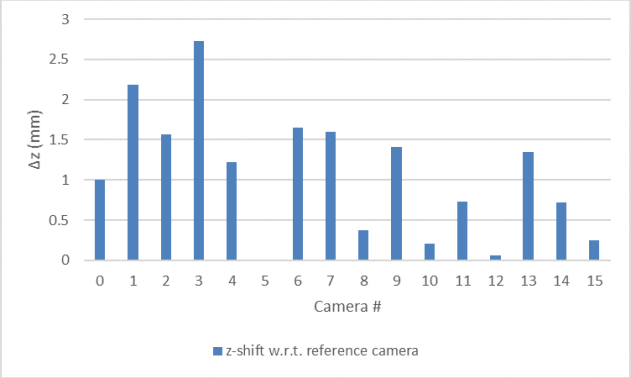
where

$$d(z) = \frac{\frac{1}{z} - \frac{1}{z_0}}{\frac{1}{z_1} - \frac{1}{z_0}}. \tag{16}$$

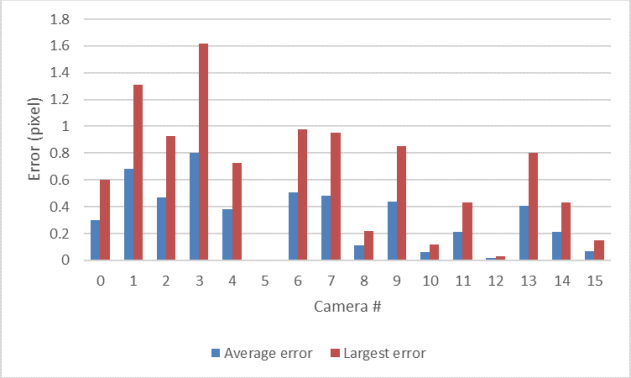
3.4. Depth Estimation

In order to estimate the depth, our pipeline has a multi-resolution matching approach that estimates a depth map for each image of the camera rig. The multi-resolution strategy allows to compute accurately the depth maps in a fast manner.

In this sense, the closest work to ours is [6]. However our algorithm uses a different similarity measure and does not impose a coherence matching among all views at each scale. Compared to other existing depth estimation methods [30][27][35], our approach is significantly different since depth estimates are not done for one single and virtual view. In our experiments we have observed that this is a key factor on the depth estimation quality.



(a)



(b)

Figure 4. (a) Shift in z with respect to the camera reference $\mathbf{c}_0 = 5$. The rig is almost coplanar since the biggest shift is 2.73mm. (b) In the reference camera $\mathbf{c}_0 = 5$, position differences (in pixels) between corresponding exact points (Eq. 6 and Eq. 7) and approximate points (Eq. 13) . The largest errors are located in the border of the images.

Correspondence matching:

Let us first present the correspondence matching done at each scale of our multi-resolution algorithm. So, we assume that the images are at the current resolution. Now, we consider the Zero-mean Normalized Cross-Correlation (ZNCC) as the similarity measure. More precisely, we note $\mu(I(u,v),n)$ the average of image I in a squared neighborhood of size $(2n+1)^2$ centered at (u,v), $\overline{I}(u,v) := I(u,v) - \mu(I(u,v),n)$ and $\sigma(\overline{I}(u,v),n)$ the standard deviation of image $\overline{I}$ in the same neighborhood. We also define

$$\widehat{I}(u,v,i,j) = \frac{I(u+i,v+j) - \mu(I(u,v),n)}{\sigma(\overline{I}(u,v),n)}\,. \qquad (17)$$

Then, given a reference view $\mathbf{c}_0$, the ZNCC is defined as

$$ZNCC(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}, z) = \frac{1}{15(2n+1)^2} \sum_{\mathbf{c}\neq\mathbf{c}_0} \sum_{i,j=-n}^{n} \widehat{I}_{\mathbf{c}_0}(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}, i, j) \cdot \widehat{I}_{\mathbf{c}}(u_{\mathbf{c}}, v_{\mathbf{c}}, i, j)\,. \qquad (18)$$

where $(u_{\mathbf{c}}, v_{\mathbf{c}}) = (u_{\mathbf{c}_0} + d(z)\cdot\delta u_{\mathbf{c}},\ v_{\mathbf{c}_0} + d(z)\cdot\delta v_{\mathbf{c}})$.

With the notations above, depth estimation at each image point $(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}) \in I_{\mathbf{c}_0}$ is performed minimizing the cost function

$$Z_{\mathbf{c}_0}(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}) = \underset{z\in[z_{min}, z_{max}]}{\arg\min}\ ZNCC(u_{\mathbf{c}_0}, v_{\mathbf{c}_0}, z)\,. \qquad (19)$$

Multi-Resolution strategy:

Multi-resolution is a well-know strategy in stereo matching [24]. Here, we have considered a pyramid in which, by definition, $I_{\mathbf{c}}^{(0)} = I_{\mathbf{c}}$, $\forall\mathbf{c}$, and at each scale $k = 0,\ldots,K$, the image $I_{\mathbf{c}}^{(k)}$ is a downsampling of $I_{\mathbf{c}}^{(k-1)}$ by a factor of 2. Now, if we aim to estimate the depth of a reference view $\mathbf{c}_0$, we start estimating the depth $Z^{(K)}$ at the coarsest scale K using Eq. 19 (for the sake of simplicity we avoid writing the index $\mathbf{c}_0$). The cost function is tested for all $z = z_{min}^{(K)} + l\cdot\Delta z^{(K)};\ l = 0,...,L$ where $\Delta z^{(K)} = (z_{max}^{(K)} - z_{min}^{(K)})/L$.

Then, for the estimation of $Z^{(K-1)}$ we minimize again Eq. 19 but using a different depth range depending on the pixel position. Indeed, for each $(u,v) \in I^{(K-1)}$, we consider the depth estimated values in the previous scale in a given neighborhood

$$z_{i,j}^{(K)} = Z^{(K)}(u/2 + i, v/2 + j); \quad i,j = -n,...,n \qquad (20)$$

and the depth ranges $\left[z_{i,j}^{(K)} - \frac{\Delta z^{(K)}}{2},\ z_{i,j}^{(K)} + \frac{\Delta z^{(K)}}{2}\right]$. So, our algorithm minimizes Eq. 19 for all

$$z = z_{i,j}^{(K)} - \frac{\Delta z^{(K)}}{2} + m\cdot\frac{\Delta z^{(K)}}{M}, \quad i,j = -n,...,n; \quad m = 0,...,M. \qquad (21)$$

The same reasoning is valid for the next scales until the finest scale $k = 0$. In our implementation we have fixed a squared neighborhood of size 3×3 (n=1), we consider $K = 4$, $L = 50$ (subdivisions at the coarsest scale) and $M = 2$ (subdivisions for the other scales). Note that the initial depth range in the coarsest scale $[z_{min}^{(K)}, z_{max}^{(K)}]$ varies for each Light-Field sequence.

It is interesting to point out that our depth estimation benefits from the fact that the previous algorithms in our pipeline are extremely accurate, so our simple but efficient algorithm produces precise depth estimates. Our pipeline does therefore not include any particular filtering of the depth maps. Also, our video sequences are processed independently for each view and without temporal coherence constraints. This strategy allows us to capture and process in a very fast manner Light-Field videos.

Figure 5. Novel virtual view rendered at an intermediate position of our camera rig.

3.5. Rendering

After depth estimation, a Multi-View plus Depth (MVD) video is available. Different options for depth-based image rendering are possible. While Synthetic Aperture Refocusing has been proposed in the literature, when the Light-Fields have been captured with a camera rig the resulting images suffer from angular aliasing due to the poor angular sampling. Instead, we believe that sparse Light-Fields are better adapted to Dynamic Perspective Rendering. This is, the estimated depth is used to render novel views different from the captured available views. To this end the MVD data is turned into a point cloud $\{\mathbf{X}_w(\mathbf{c},u,v);I_{\mathbf{c}}(u,v)\}_{\mathbf{c},u,v}$ as follows:

$$\mathbf{X}_w(\mathbf{c},u,v)=\mathbf{P}_{\mathbf{c}}\cdot\begin{pmatrix} Z_{\mathbf{c}}(u,v)\cdot\widehat{\mathbf{K}_{\mathbf{c}}}^{-1}\cdot\begin{pmatrix}u\\v\\1\end{pmatrix}\\ 1\end{pmatrix}. \quad (22)$$

Then the novel view is rendered by projecting the point cloud onto a virtual pinhole camera defined by its intrinsic and extrinsic matrices $\mathbf{K}_R$ and $\mathbf{Q}_R$.

4. Dataset and Experimental Results

We provide a set of synchronized Light-Field video sequences captured by a 4×4 camera rig at 30 fps. Each camera has a resolution of 2048×1088 pixels and a 12mm lens. The Field Of View (FOV) is $50^\circ\times37^\circ$. Fig. 7 shows one camera image of one frame of the Light-Field sequences we have captured. Our dataset has a number of close-ups sequences that are interesting for some specific use cases such as realistic telepresence. Indeed, recovering 3D accurate information of faces is still a challenging problem because very small errors may create unpleasant results. We have also captured medium angle scenes (*Painter, Birthday*) and other animated scenes where the movement does not come from a human (*Automaton, Theater, Train*).

In our dataset, we consider the reference camera $\mathbf{c}_0=(1,1)$. This is $s=t=1$. For each Light-Field sequence, we will provide the intrinsic matrix of the reference pinhole camera $\widehat{\mathbf{K}_{\mathbf{c}_0}}$, the reference depth z_0 and the chosen depth z_1 for which the shifts $(\delta u_{\mathbf{c}},\delta v_{\mathbf{c}})$ in Eq. 13 are computed.

For example, for the sequence *Painter*,

$$\widehat{\mathbf{K}_{\mathbf{c}_0}}=\begin{pmatrix}2340.14 & 0 & 1043.09\\ 0 & 2340.14 & 480.46\\ 0 & 0 & 1\end{pmatrix}, \quad (23)$$

we have chosen $z_0=100$m and $z_1=1.630$m and Table 1 shows the shifts $(\delta u_{\mathbf{c}},\delta u_{\mathbf{c}})$. Note that, a different shift table has to be computed for each sequence with different calibration settings. Besides, given the geometric position of our cameras in he rig and the fact that they are physically well aligned, the computed shifts are close to be equispaced. For example, if $s=0$, the correspondent points at $z=z_1$ have a row shift of respectively 98.28, 98.14, 98.07 and 97.35 pixels. And at any z, the corresponding points have a row shift of respectively $98.28\cdot d(z)$, $98.14\cdot d(z)$, $98.07\cdot d(z)$ and $97.35\cdot d(z)$ pixels. Considering these shifts is more accurate than considering a perfect epipolar rectification of the images.

s \ t	0	1	2	3
0	$\binom{100.00}{98.28}$	$\binom{-0.36}{98.14}$	$\binom{-97.19}{98.07}$	$\binom{-195.55}{97.35}$
1	$\binom{98.67}{-1.73}$	$\binom{0.00}{0.00}$	$\binom{-96.18}{0.74}$	$\binom{-197.85}{0.11}$
2	$\binom{99.17}{-99.93}$	$\binom{0.21}{-99.11}$	$\binom{-98.33}{-101.12}$	$\binom{-197.00}{-99.07}$
3	$\binom{99.08}{-197.68}$	$\binom{-1.22}{-198.14}$	$\binom{-99.26}{-198.89}$	$\binom{-198.36}{-199.37}$

Table 1. Values of the shifts $(\delta u_{\mathbf{c}},\delta v_{\mathbf{c}})$ for the sequence *Painter* with the reference camera $\mathbf{c}=(1,1)$.

Fig. 6 shows the depth maps for the first frame of the sequence *Painter*. The scene has many different objects and a person walking on it. Fig. 8 shows the point clouds obtained with our pipeline for the sequences *Face1* and *Rugby*. In particular, since our rig has been calibrated, our 16 depth maps are all projected into a precise point cloud. Our pipeline does not have a proper filtering of the depth maps. Instead, the only manipulation that has been done in the point clouds is to remove completely isolated points and points that have not been coherently estimated by at least half of the cameras (8 cameras). While our camera rig is not intended to provide complete 3D points clouds of objects as 360-camera rigs would do, the visualization of the point clouds from different viewpoints allows to assess the accuracy of our depth estimates. Finally, Fig. 5 shows an image rendered from a virtual position different from the camera positions of the camera rig using Eq. 22.

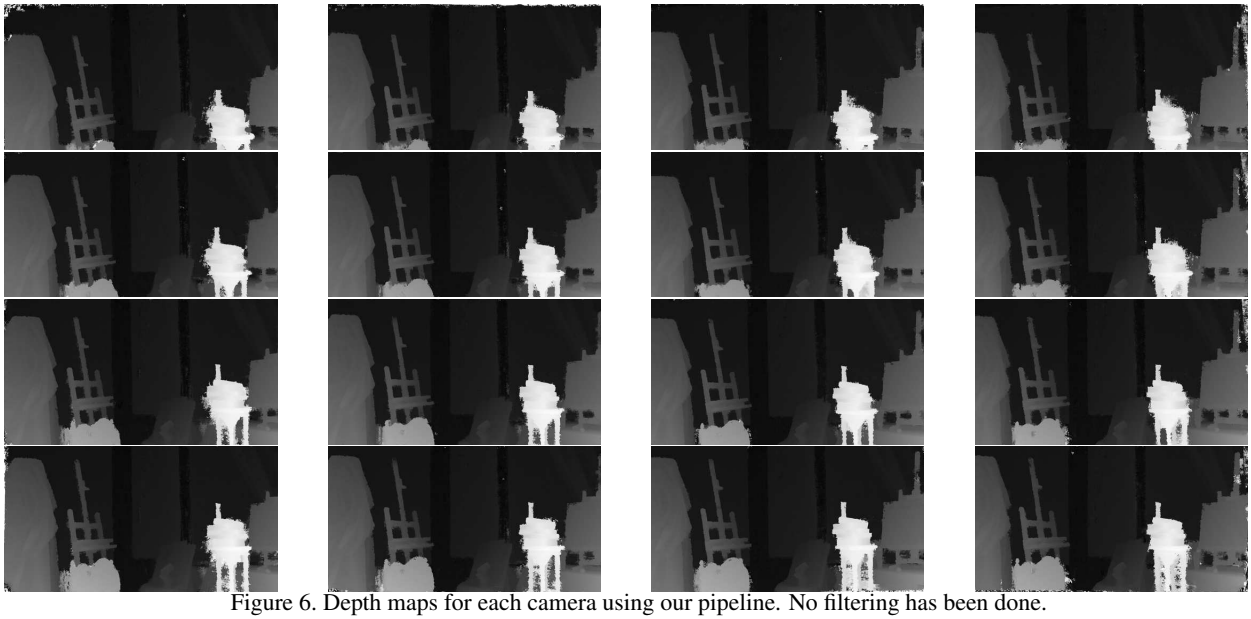


Figure 6. Depth maps for each camera using our pipeline. No filtering has been done.

The rendering of such images allows to render the scene with dynamic parallax.

Computational time Our first goal is to implement an accurate pipeline that precisely captures and manipulates data. We have also implemented our pipeline in GPU to meet the computational time requirements of some applications. In particular, our fast implementation captures data in real time using the registered geometry and color calibration parameters. Demosaicking is done with a linear algorithm in this case. Depth estimation, the step with highest complexity, is performed at 22fps at the full image resolution (2048×1088 color images) on an NVidia GTX 1080 Ti and at 32fps on a Nvidia Quadro P6000. Our image rendering for dynamic parallax is achieved in real time in GPU.

5. Conclusion

In this paper we have presented a complete pipeline for accurately capture and process Light-Fields. Our pipeline is suitable for real-time applications. We have also created a dataset available at [1] that is our major contribution and we believe will be of interest for the scientific community.

At this moment we believe that one of the major challenges for the community is to address the problem of Light-Field video compression. Indeed, many capturing systems generating a huge amount of data and many applications having very constrained transmission requirements, compression is of utmost importance for the technology to become popular.

Furthermore, while image and video editing is a well-known problem an many tools exist, not many solutions to edit Light-Field video have been studied. Such methods would need to handle a big amount of data and to guarantee inter-view coherence to succeed.

References

[1] Technicolor light-field dataset. http://www.technicolor.com/en/innovation/scientific-community/scientific-data-sharing/light-field-dataset. 1, 8

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

[3] T. Basha, S. Avidan, A. Hornung, and W. Matusik. Structure and motion from scene registration. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1426–1433. IEEE, 2012. 2

[4] J.-Y. Bouguet. www.vision.caltech.edu/bouguetj/calib_doc. 4

[5] D. Cho, M. Lee, S. Kim, and Y.-W. Tai. Modeling the calibration pipeline of the lytro camera for high quality light-field image reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3280–3287, 2013. 2

[6] Ł. Dabała, M. Ziegler, P. Didyk, F. Zilly, J. Keinert, K. Myszkowski, H.-P. Seidel, P. Rokita, and T. Ritschel. Efficient multi-image correspondences for on-line light field video processing. In *Computer Graphics Forum*, volume 35-7, pages 401–410. Wiley Online Library, 2016. http://resources.mpi-inf.mpg.de/LightFieldVideo/. 2, 5

[7] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012. 1

[8] J. Duran and A. Buades. A Demosaicking Algorithm with Adaptive Inter-Channel Correlation. *Image Processing On Line*, 5:311–327, 2015. 4

Face1

Face2

Face3

Face4

Face5

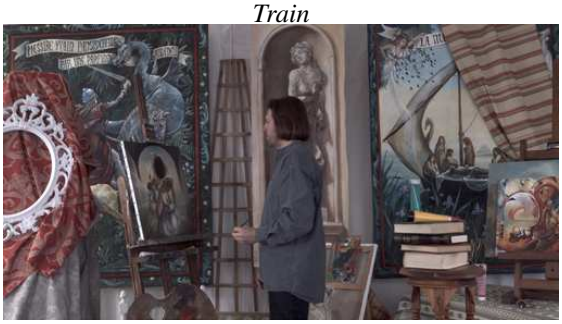
Rugby

Hands

Automaton

Train

Birthday



Painter

Theater

Figure 7. Reference images for the first frame of the Technicolor Light-Field dataset.

Figure 8. Point clouds from different viewpoints using one frame of the Light-Field sequences *Face* and *rugby*. Background has been removed for the sake of visualization.

[9] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Motion driven tonal stabilization. *IEEE Transactions on Image Processing*, 25(11):5455–5468, 2016. 2

[10] Y. Furukawa, C. Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 1, 2

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996. 2

[12] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[13] M. Hog, N. Sabater, B. Vandame, and V. Drazic. An image rendering pipeline for focused plenoptic cameras. *IEEE Transactions on Computational Imaging*, 2017. To appear. 2

[14] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*, pages 19–34. Springer, 2016. http://www.lightfield-analysis.net. 2

[15] C.-T. Huang, J. Chin, H.-H. Chen, Y.-W. Wang, and L.-G. Chen. Fast realistic refocusing for sparse light fields. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1176–1180. IEEE, 2015. 2

[16] N. Joshi, S. Avidan, W. Matusik, and D. J. Kriegman. Synthetic aperture tracking: tracking through occlusions. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 2

[17] N. Joshi, B. Wilburn, V. Vaish, M. L. Levoy, and M. Horowitz. *Automatic color calibration for large camera*

arrays. [Department of Computer Science and Engineering], University of California, San Diego, 2005. 2

[18] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. H. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32(4):73–1, 2013. https://www.disneyresearch.com/project/lightfields. 2

[19] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996. 1, 2

[20] G. Lippmann. Epreuves reversibles donnant la sensation du relief. *J. Phys. Theor. Appl.*, 7(1):821–825, 1908. 1

[21] M. I. Lourakis and A. A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2, 2009. 2, 4

[22] S. Lu, T. Mu, and S. Zhang. A survey on multiview video synthesis and editing. *Tsinghua Science and Technology*, 21(6):678–695, 2016. 2

[23] S.-P. Lu, B. Ceulemans, A. Munteanu, and P. Schelkens. Spatio-temporally consistent color and structure optimization for multiview video color correction. *IEEE Transactions on Multimedia*, 17(5):577–590, 2015. 2

[24] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*. Vancouver, BC, Canada, 1981. 6

[25] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Transactions on Graphics (TOG)*, 32(4):46, 2013. http://web.media.mit.edu/~gordonw/SyntheticLightFields/index.php. 2

[26] A. Mousnier, E. Vural, and C. Guillemot. Partial light field tomographic reconstruction from a fixed-camera focal stack. *arXiv preprint arXiv:1503.01903*, 2015. https://www.irisa.fr/temics/demos/lightField/index.html. 2

[27] Z. Pei, Y. Zhang, X. Chen, and Y.-H. Yang. Synthetic aperture imaging using pixel labeling via energy minimization. *Pattern Recognition*, 46(1):174–187, 2013. 2, 5

[28] Z. Pei, Y. Zhang, T. Yang, X. Zhang, and Y.-H. Yang. A novel multi-object detection method in complex scene using synthetic aperture imaging. *Pattern Recognition*, 45(4):1637–1658, 2012. 2

[29] N. Sabater, M. Seifi, V. Drazic, G. Sandri, and P. Pérez. *Accurate Disparity Estimation for Plenoptic Images*, pages 548–560. Springer International Publishing, Cham, 2015. 2

[30] V. Vaish, M. Levoy, R. Szeliski, C. L. Zitnick, and S. B. Kang. Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2331–2338. IEEE, 2006. 2, 5

[31] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane+parallax for calibrating dense camera arrays. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004. 2

[32] J. Vazquez-Corral and M. Bertalmío. Color stabilization along time and across shots of the same scene, for one or several cameras of unknown specifications. *IEEE Transactions on Image Processing*, 23(10):4564–4575, 2014. 2

[33] 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. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 765–776. ACM, 2005. http://lightfield.stanford.edu. 1, 2

[34] Y. Xu, K. Maeno, H. Nagahara, and R.-i. Taniguchi. Camera array calibration for light field acquisition. *Frontiers of Computer Science*, 9(5):691–702, 2015. 2

[35] T. Yang, Y. Zhang, J. Yu, J. Li, W. Ma, X. Tong, R. Yu, and L. Ran. All-in-focus synthetic aperture imaging. In *European Conference on Computer Vision*, pages 1–15. Springer, 2014. 2, 5

[36] C. Zhang. Multiview imaging and 3dtv. *IEEE signal processing magazine*, 24(6):10–21, 2007. 1