Joint NeuraL Representation For Multiple Light Fields

Guillaume Le Guludec, Christine Guillemot

JOINT NEURAL REPRESENTATION FOR MULTIPLE LIGHT FIELDS

Guillaume Le Guludec and Christine Guillemot

Inria Rennes-Bretagne Atlantique, Rennes, France

ABSTRACT

Neural implicit representations have appeared as a promising technique for representing a variety of signals, among which light fields, offering several advantages over traditional grid-based representations, such as independence to the signal resolution. While some work has been done to find good initial representations for a given type of signal, usually via meta-learning approaches, exploiting the features shared between different scenes remains an understudied problem. We provide a step towards this end by presenting a method for sharing the representation between thousands of light fields, splitting the representation between a part that is shared between all light fields and a part which varies individually from one light field to another. We show that this joint representation possesses good interpolation properties, and allows for a more light-weight storage of a whole database of light fields, exhibiting a ten-fold reduction in the size of the representation when compared to using a separate representation for each light field.

***Index Terms*—** Light fields, Neural Implicit Representations, Singular value decomposition.

1. INTRODUCTION

Compared to traditional two-dimensional images, light fields provide a richer modelling of natural visual scenes by discriminating the received intensities not only spatially, but also angularly, meaning that different radiances are recorded for different directions of incident rays. Light fields, when viewed as collections of sub-aperture images, are therefore very costly to store, as the number of pixels grows quadratically in both the spatial and the angular resolution. While signals are traditionally represented as discrete arrays of data points, *neural implicit representations* have recently gained a lot of attention as efficient alternative means for representing and storing signals.

Since their success in the context of view synthesis [1] and shape representation [2, 3], neural implicit representations have been applied to a great variety of tasks for many types of signals, including to light field representation [4, 5, 6]. Those methods typically rely on some encoding of the input coordinates [7], followed by a multi-layer perceptron that predicts the value of the signal at the given coordinate. Feng *et al.* were able to achieve excellent representation quality of dense light fields using an architecture which they dubbed SIGNET [4]. Their method, however, can only be used to represent a single light field at a time. Building upon their architecture, we show that it is possible to jointly represent an entire database of dense light fields, and demonstrate our approach on a collection of two thousand light fields of natural scenes captured using a Lytro Illum.

This work was supported by the french ANR research agency in the context of the artificial intelligence project DeepCIM.

2. LIGHT FIELD REPRESENTATION

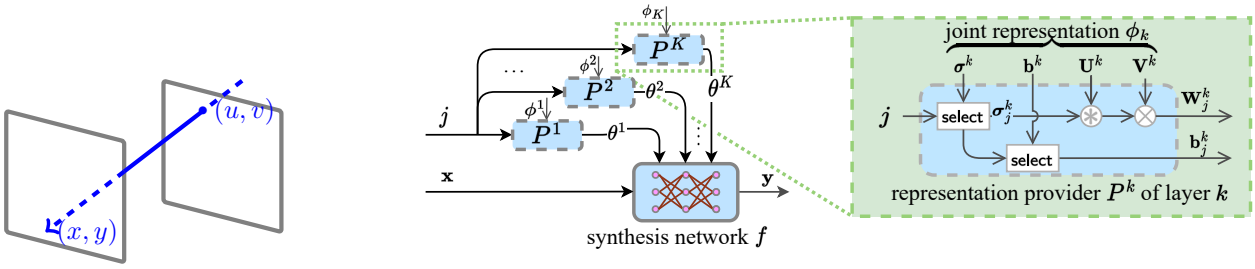
2.1. Light fields parameterization

A light field (LF) can be viewed as a function that associates a radiance in each of the color channels of interest to any geometrical ray in space. We consider the widespread *two-plane parameterization* of light fields. In this parameterization, each ray in space is uniquely characterized by the coordinates of its intersection with two fixed planes, the *spatial* and *angular* planes, corresponding usually (albeit not necessarily) to the plane of the camera sensor and the plane of the camera aperture respectively. In such a parameterization, illustrated in Fig. 1a, any ray is described by a tuple (x, y, u, v). We consider RGB light fields; a LF is thus simply a function $L : \mathbb{R}^4 \rightarrow \mathbb{R}^3$ from the space of coordinates to the space of color values. In the remainder of the article, for the sake of brevity, we use the symbol $\mathbf{x}$ to denote the 4D vector of spatio-angular coordinates. We denote the space of spatio-angular coordinates for which data is available by X.

2.2. Representing a single light field

We seek to approximate a LF using a *synthesis network* $f(.;\theta)$ parameterized by some $\theta \in \mathbb{R}^d$, called a *representation* of the LF. The quality of the representation depends on a chosen metric; in our experiments we choose to use the pixel-wise mean squared error as an objective metric to measure the distortion of the signal. Our goal of finding a good representation therefore comes down to solving the following regression problem:

$$\min_{\theta} \sum_{\mathbf{x} \in X} \|f(\mathbf{x};\theta) - L(\mathbf{x})\|^2 \qquad (1)$$



(a) (b)

Fig. 1: (a) Two-plane parameterization of a LF. (x, y) and (u, v) parameterize the intersection of the ray with the spatial and angular planes respectively. (b) Representation pipeline: the synthesis network f computes the RGB value $\boldsymbol{y}$ from ray coordinates $\mathbf{x}$ for a given LF, using the LF representation $\theta = (\theta^1, \ldots, \theta^K)$. The representation θ^k of layer k is produced by the representation provider P^k . $\boldsymbol{\sigma}^k$ is a tensor of size (J, r) that denotes the concatenation of the $\boldsymbol{\sigma}_j^k$ for all LFs; $\mathbf{b}^k$ is a tensor of size (J, m) concatenating the biases for all LFs; select denotes the operation of selecting a given row; $*$ denotes broadcast multiplication of a vector by a matrix, while $\times$ denotes matrix multiplication.

This problem is efficiently solved using a stochastic gradient descent solver, such as ADAM [8], combined with standard backpropagation for computing the gradient of the objective with respect to the representation θ.

2.3. Architecture of the representation network

Feng *et al.* [4] were able to obtain a good neural implicit representation of LFs by combining a fully connected multilayer perception with sinusoidal activation with a polynomial positional encoding. We briefly present here the architecture, which is based on the SIREN architecture [9]. The layers of their network are defined by:

$$\mathbf{a}^0 = \text{Encode}(\mathbf{x}) \tag{2}$$

$$\mathbf{y}^k = \sin(\mathbf{a}^k \mathbf{W}^k + \mathbf{b}^k) \tag{3}$$

$$\mathbf{z}^k = \mathbf{y}^k + \mathbf{a}^k \tag{4}$$

$$\mathbf{a}^{k+1} = \text{LayerNorm}(\mathbf{z}^k) \tag{5}$$

where Eq. 4 holds for all layers k except for the first and last ones, for which $\mathbf{z}^k = \mathbf{y}^k$ and Eqn. 5 holds for all layers k except for the last one, which satisfies $\mathbf{a}^{k+1} = \mathbf{z}^k$ instead. In our experiments, we reuse their architecture without any change, except for the last layer, in which the activation is replaced by a sigmoid. We have therefore $\theta = (\mathbf{W}^k, \mathbf{b}^k)_{k \leq K}$.

2.4. Positional encoding

In [4], the authors use the *Gegenbauer polynomials* G_α up to a certain order as the positional encoding (Eq. 2) and show that $\alpha = 0.5$ is empirically optimal, which corresponds to using the Legendre polynomials.

We found that replacing this polynomial positional encoding scheme with a random Fourier encoding [7] did not hurt the ability of the model to fit the LFs and reduces some artifacts, provided the (integer) input coordinates $\mathbf{x}$ are normalized to an appropriate scale. Formally, we use an encoder with trainable parameters $\mathbf{W}$ and $\mathbf{b}$ defined by:

$$\text{Encode}(\mathbf{x}) = \sin((\mathbf{x} - \mathbf{x}_0)\mathbf{W} + \mathbf{b}) \tag{6}$$

where $\mathbf{x}_0$ is the mean coordinates. $\mathbf{W}$ is a $4 \times n$ matrix in which the elements in the two first rows (corresponding to the spatial coordinates x and y) are randomly initialized by sampling the uniform distribution $\mathcal{U}(-s, s)$, while the elements in the last two rows (corresponding to the angular coordinates u and v) are initialized from a uniform distribution $\mathcal{U}(-t, t)$. We found that using $s = 0.2$ and $t = 0.06$ yielded good results. The elements of $\mathbf{b}$ are obtained by sampling from $\mathcal{U}(-\pi, \pi)$.

3. REPRESENTING MULTIPLE LIGHT FIELDS

3.1. Problem formulation

Let us now consider the more general case in which we have a collection of LFs (L_j) indexed by some $j \in J$. We call *representation provider* a function $P(j; \phi)$ parameterized by some vector ϕ, that maps the index j of a LF to a representation θ of that LF. The goal of the joint representation is to find a representation provider by learning the parameters ϕ, such that $P(j; \phi)$ provides on average a good representation of L_j. In other words, we want to minimize the following loss:

$$\mathcal{L}(\phi) = \sum_{j \in J} \sum_{\mathbf{x} \in X} \| f(\mathbf{x}; P(j; \phi)) - L_j(\mathbf{x}) \|^2 \tag{7}$$

The rationale for finding a joint representation relies on the notion that all representations θ lie on the same manifold. Since light fields of natural scenes share some common features (e.g parallax and laws of optics, common shapes, etc.), it is natural to expect some structure on this manifold. One approach is thus to learn this structure by training from a dataset of various LFs.

3.2. How to parameterize the representation manifold?

Hypernetworks [10] could be used as a means to parameterize this manifold. The hypernetwork might be used to map a discrete representation of a LF to a neural representation θ, *e.g.* using a CNN. It could also be an *auto-decoder* [3, 11] mapping some latent code $\mathbf{z}$ of a given LF, to a neural representation θ.

However, while this approach might be yielding good results in the case of simple signals such as low-resolution 2D images, it becomes intractable in practice when working with light fields even of average resolution. Indeed, representing a LF usually requires a larger network when compared to a 2D image due to a higher complexity. The output of the hyper-networks therefore becomes very large, potentially several millions, which makes the approach intractable.

As an alternative, *FiLM conditioning* [12, 13] is an approach that modulates the pre-activations in an affine manner using a scaling and offset coefficient. In the same vein as FiLM, we first designed an approach that shared the same weight matrices $\mathbf{W}^k$ between all LFs, while learning individual biases for each LF. While we found that this yields good results when the number of LFs in the dataset is small, as the number of LFs increases, the faithfulness of the reconstruction degrades.

In order to mitigate this problem, we increased the capacity of the representation provider by allowing the weight matrix in the representation to vary between LFs. In the following sections, we describe the structure of the representation provider.

4. PROPOSED REPRESENTATION

4.1. Structure of the joint representation

A joint representation ϕ is obtained by solving Eqn. 7 using stochastic gradient descent and gradient back-propagation. As f is a neural network, each forward and backward pass is costly. Therefore we want the representation provider to be as light-weight as possible.

The first idea is to split the parameterization ϕ of the representation provider into two parts: a shared part ϕ^{sh} and individual part $(\phi_j^{\text{ind}})_{j \in J}$. For a LF of index j, the two parts are combined by a simple procedure described below to provide the representation θ.

In our approach, the shared part of the joint representation is composed, for each layer k in the synthesis network with input dimension m and output dimension n, of a pair of matrices:

$$\phi^{\text{sh}} = \left((\mathbf{U}^k, \mathbf{V}^k)\right)_k \tag{8}$$

where $\mathbf{U}^k$ has size $m \times r$ and $\mathbf{V}^k$ has size $r \times n$, m and n being respectively the input and output dimensions of the layer; r is a hyper-parameter controlling the trade-off between the number of parameters and the quality of the joint representation. The individual parts of the representation are given by:

$$\left(\phi_j^{\text{ind}}\right)_j = \left(\left((\boldsymbol{\sigma}_j^k, \mathbf{b}_j^k)\right)_k \text{ for all } j \in J\right) \tag{9}$$

where $\boldsymbol{\sigma}^k$ and $\mathbf{b}_j^k$ are vectors of size r and n respectively. The full joint representation for the collection of LFs L_j is the pair

$$\left(\phi^{\text{sh}}, \left(\phi_j^{\text{ind}}\right)_j\right)$$

4.2. Representation provider P

The representation provider $P(.;\phi)$ that maps the index j of a LF to its representation θ is defined by:

$$P(j;\phi) = \left((\mathbf{W}_j^k, \mathbf{b}_j^k)\right)_k \tag{10}$$

where the weight matrix $\mathbf{W}_j^k$ is given by

$$\mathbf{W}_j^k = \mathbf{U}^k \text{diag}(\boldsymbol{\sigma}_j^k)\mathbf{V}^k \text{ for all } j \in J \text{ and layer } k \tag{11}$$

where $\text{diag}(x)$ is the diagonal matrix whose diagonal elements are those of x. The matrix product $\mathbf{U}^k \text{diag}(\boldsymbol{\sigma}_j^k)$ is computed by broadcast multiplication of the rows of $\mathbf{U}^k$ by the elements of $\boldsymbol{\sigma}_j^k$. The complete representation pipeline is illustrated in Fig. 1b. We motivate our choice of architecture in the section below.

4.3. Inspiration from the singular value decomposition

Any matrix $\mathbf{W}$ of size $m \times n$ can be written as its well-known singular values decomposition (SVD):

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V} \tag{12}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices of size $m \times r$ and $r \times n$ respectively and $\boldsymbol{\Sigma}$ is a diagonal matrix of size $r \times r$ containing the non-zero singular values of $\mathbf{W}$, where $r \leq \min(m, n)$ is the rank of $\mathbf{W}$, equal to the number of non-zero elements in $\boldsymbol{\Sigma}$.

Assuming that we have weight matrix $\mathbf{W}_j^k$ for a layer k and a LF $j \in J$, we could separately apply the SVD to each matrix as:

$$\mathbf{W}_j^k = \mathbf{U}_j^k \boldsymbol{\Sigma}_j^k \mathbf{V}_j^k \tag{13}$$

However, this is not going to lead to any joint representation. Our idea was to assume that the weight matrices can be jointly decomposed across LFs as

$$\mathbf{W}_j^k = \mathbf{U}^k \boldsymbol{\Sigma}_j^k \mathbf{V}^k \tag{14}$$

with $\mathbf{U}^k$ and $\mathbf{V}^k$ now being shared between the different LFs. Since an arbitrary collection of matrices is unlikely to be amenable to joint decomposition, we relax the constraint that r be less than or equal to $\min(m, n)$.

It is worth noting that such a parameterization enforces the representation manifold to be a vector space. The shared part of the representation effectively provides a basis for this space, while the individual parts correspond to coordinates in the representation manifold.

Algorithm 1 Joint training forward pass

Require: LF index j, coordinates $\mathbf{x}$
  $\mathbf{a} = \text{Encode}(\mathbf{x})$
  for $k \in [0..K-1]$ **do**
    $\mathbf{b} \leftarrow \mathbf{b}_j^k$
    $\mathbf{W} \leftarrow \mathbf{U}^k \text{diag}(\boldsymbol{\sigma}_j^k)\mathbf{V}^k$
    $\mathbf{a} \leftarrow \text{Layer}_k(\mathbf{a}, \mathbf{W}, \mathbf{b})$
  end for
  return a

5. EXPERIMENTS

5.1. Methodology

We performed experiments using the architecture for the synthesis network described in section 2.3 and the representation provider described in section 4.2. We experimented with values for hyper-parameter r ranging in $\{4096, 8192, 16384\}$. Our dataset is composed of the first 2,000 LFs of the dataset introduced by [14], consisting of $375 \times 540 \times 8 \times 8$ RGB dense light fields of flowers captured using a Lytro Illum.

To be consistent with the singular value decomposition idea, for each layer k, the matrices $\mathbf{U}^k$ and $\mathbf{V}^k$ are initialized as random orthogonal matrices, while the elements of $\boldsymbol{\sigma}_j^k$ are drawn from the uniform distribution $\mathcal{U}(0, \sqrt{6})$. During training however, neither $\mathbf{U}^k$ nor $\mathbf{V}^k$ are constrained to be orthogonal, and the elements of $\boldsymbol{\sigma}_j^k$ can become negative. We used $K = 10$ layers for our synthesis network, with 512 hidden channels per layer.

We trained each model for 16 million iterations, with batch size of 4096. This is equivalent to 2.5 epochs of training; that is, each pixel of each LF is fed on average 2.5 times to the network. At each iteration, we randomly sample a LF index j along with a random batch of coordinates $\mathbf{x}$. The forward pass is described in Algorithm 1. Following [4], we use a learning rate with cosine decay from 10^{-5} to 10^{-8}. Training the largest model (r = 16384) took 20 days on a Nvidia RTX 2080 Ti.

5.2. Parameters reduction

To demonstrate the ability of our approach to effectively reduce the total number of parameters required to fit an entire dataset, we separately trained a model (as defined by Eqs. 2 to 6) on several LFs of the whole dataset. We considered a random subset J' of 100 LFs and computed the average PSNR as an estimate of the average PSNR on the whole dataset. We trained a new set of weights for each LF $j \in J'$ for 3 epochs, so as to provide a fair comparison with the joint training. We adjusted the capacity of the individual models so that the reconstruction quality after 3 epochs approximately matches that of the joint training. To do so, we chose a depth of 5 layers for the synthesis network.

model	# parameters / LF	PSNR (dB)
joint r = 4096	65K	36.15
joint r = 8192	126K	37.65
joint r = 16384	248K	38.85
separate	1.1M	37.24

Fig. 2: Reconstruction quality and model size for the different models.

Fig. 3: Central views along with EPIs for 3 LFs in the *Flowers* dataset [14]. Top row are the LFs produced by the joint model with rank r = 16384; bottom row corresponds to a model specifically fitted for one LF.

5.3. Results

Table 2 shows the average PSNR on the whole dataset for different values of r, as well as the size of the models per LF (*i.e.* the total number of parameters divided by the number of LFs), and additionally provides a comparison with the separately trained models. We see that, for the same reconstruction quality, the joint representation results in a ten-fold reduction in the number of parameters. Fig. 3 provides a visual illustration of the reconstruction quality of our joint representation. In addition to the central view, we computed interpolated epipolar plane images (EPI) by querying the model on non-integer coordinates. More precisely, we interpolated by a factor 8, converting the 8×8 LF into a 64×64 LF. The smoothness and quality of the interpolated EPIs show the ability of the joint representation to capture parallax and perform angular interpolation on the learned LFs.

6. CONCLUSIONS

We have presented a method to jointly represent a large collection of light fields using neural implicit representations. We have shown that our technique yields good reconstruction quality both visually and in terms of PSNR. Additionally, we showed that the joint representation greatly reduces the number of parameters required to represent an entire collection of light fields when compared a set of scene-by-scene representations.

7. REFERENCES

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, Eds., Cham, 2020, pp. 405–421, Springer International Publishing.

[2] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019.

[3] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein, *Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations*, Curran Associates Inc., Red Hook, NY, USA, 2019.

[4] Brandon Yushan Feng and Amitabh Varshney, "Signet: Efficient neural representation for light fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 14224–14233.

[5] Paramanand Chandramouli, Hendrik Sommerhoff, and Andreas Kolb, "Light field implicit representation for flexible resolution reconstruction," *CoRR*, vol. abs/2112.00185, 2021.

[6] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim, "Learning neural light fields with ray-space embedding," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 19787–19797.

[7] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2022, NIPS'20, Curran Associates Inc.

[8] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[9] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein, "Implicit neural representations with periodic activation functions," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020, NIPS'20, Curran Associates Inc.

[10] David Ha, Andrew M. Dai, and Quoc V. Le, "Hypernetworks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017, OpenReview.net.

[11] Jeong Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," 06 2019, pp. 165–174.

[12] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio, "Feature-wise transformations," *Distill*, 2018.

[13] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville, "Film: Visual reasoning with a general conditioning layer," 2018, AAAI'18/IAAI'18/EAAI'18, AAAI Press.

[14] Pratul P. Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng, "Learning to synthesize a 4d rgbd light field from a single image," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2262–2270, 2017.