Received 3 October 2024; revised 11 February 2025; accepted 12 February 2025. Date of publication 25 February 2025; date of current version 14 March 2025. The review of this article was arranged by Associate Editor E. Vera.

Digital Object Identifier 10.1109/OJSP.2025.3545356

Unsupervised Angularly Consistent 4D Light Field Segmentation Using Hyperpixels and a Graph Neural Network

MARYAM HAMAD (Graduate Student Member, IEEE), CAROLINE CONTI (Member, IEEE), PAULO NUNES (Member, IEEE), AND LUÍS DUCLA SOARES (Senior Member, IEEE)

[1]Instituto de Telecomunicações, Instituto Universitário de Lisboa (ISCTE-IUL), 1649-026 Lisboa, Portugal

CORRESPONDING AUTHOR: MARYAM HAMAD (e-mail: maryam.hamad@lx.it.pt).

This work was supported in part by FCT/MECI through National Funds and EU Funds under Grant UID/50008 and in part by the Instituto de Telecomunicações under Project PTDC/EEI-COM/7096/2020.

This article has supplementary downloadable material available at https://doi.org/10.1109/OJSP.2025.3545356, provided by the authors.

ABSTRACT Image segmentation is an essential initial stage in several computer vision applications. However, unsupervised image segmentation is still a challenging task in some cases such as when objects with a similar visual appearance overlap. Unlike 2D images, 4D Light Fields (LFs) convey both spatial and angular scene information facilitating depth/disparity estimation, which can be further used to guide the segmentation. Existing 4D LF segmentation methods that target object level (i.e., mid-level and high-level) segmentation are typically semi-supervised or supervised with ground truth labels and mostly support only densely sampled 4D LFs. This paper proposes a novel unsupervised mid-level 4D LF Segmentation method using Graph Neural Networks (LFSGNN), which segments all LF views consistently. To achieve that, the 4D LF is represented as a hypergraph, whose hypernodes are obtained based on hyperpixel over-segmentation. Then, a graph neural network is used to extract deep features from the LF and assign segmentation labels to all hypernodes. Afterwards, the network parameters are updated iteratively to achieve better object separation using backpropagation. The proposed segmentation method supports both densely and sparsely sampled 4D LFs. Experimental results on synthetic and real 4D LF datasets show that the proposed method outperforms benchmark methods both in terms of segmentation spatial accuracy and angular consistency.

INDEX TERMS Light field, unsupervised segmentation, deep learning, angular consistency, graph neural network.

I. INTRODUCTION

Light field (LF) imaging has attracted increasing attention from researchers due to the rich information it includes and its potential for immersive applications [1], [2]. LFs contain information about both the intensity and direction of light rays and can be represented as an array of views captured from different perspectives. To represent that array of views, a 4D function $I(x, y, u, v)$ can be used, where (x, y) and (u, v) are, respectively, the spatial and angular coordinates of each view. By fixing one angular and one spatial coordinate, an Epipolar Plane Image (EPI) (i.e., the unique 2D spatio-angular LF slice typically containing a regular structure with several slanted lines [1]) can be obtained, which corresponds to the depth/disparity cues, as presented in Fig. 1. Depth/disparity cues in 4D LFs can help improve different computer vision tasks, such as in scene segmentation, by using these cues as a discriminative feature, notably, when visual information alone is not sufficient.

Image segmentation is a fundamental task that aims at dividing image data into perceptual and homogenous regions according to specific criteria. By segmenting an image, we can isolate and identify individual components or objects, which is essential for several applications, such as image compression, object detection, autonomous driving, medical imaging and scene understanding [3]. Image segmentation, in 2D images, has been widely investigated with different solutions

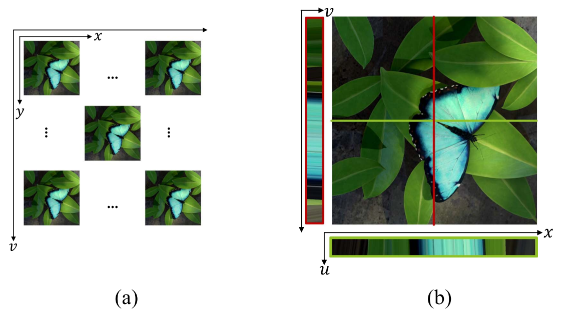

FIGURE 1. In 4D LFs, each LF view (i.e., a slice of 4D LF in a particular angular plane (u, v)) captures the scene from a different view perspective as in (a). This results in shifted light rays across views as can be seen in the EPIs with green and red borders, shown below and to the left of the central view in (b).

including traditional methods, e.g., clustering and graph-cut optimization techniques [3], or deep learning-based methods [3]. Most of the deep learning-based 2D image segmentation methods are supervised, relying on Ground Truth (GT) label images. Since generating pixel-wise annotations for large datasets can be labor-intensive and costly, the development of fully unsupervised methods or the fine-tuning of pre-trained foundation models that have been trained on large datasets to extract deep features for image segmentation tasks became a growing research direction with promising performance [3], [4].

Although 2D image segmentation is an active research area, 4D LF segmentation remains relatively unexplored, with additional challenges and performance requirements to be considered. While segmentation accuracy is important in 2D images, segmentation angular consistency in 4D LFs is also essential. More precisely, when segmenting 4D LFs, the corresponding pixels across all LF views must have the same segmentation label. Otherwise, the sudden label changes when navigating through the views can lead to unwanted flickering. Coupled with the huge amount of data involved, and the lack of 4D LF segmentation datasets for training and evaluation, this makes 4D LF segmentation a more complex task than conventional 2D image segmentation.

Existing 4D LF segmentation methods can be categorized into three main categories according to the level of the semantic meaning of the obtained segments (as detailed in Section II): i) Low-level unsupervised over-segmentation methods, where similar pixels are grouped into perceptually meaningful atomic regions, without the need for label annotations or user scribbles, e.g., [5], [6], [7], [8], [9], [10]; ii) Mid-level semi-supervised segmentation, where the semantic labels of the segmented objects are not included, e.g., [11], [12], [13], [14], [15]; and iii) High-level supervised semantic segmentation methods, e.g., [16], [17], [18], where semantic labels are also predicted for each pixel. This paper focuses on achieving mid-level multi-label segmentation in a fully unsupervised manner using a deep-learning approach.

Low-level and mid-level 4D LF segmentation methods typically rely on classical or basic machine learning techniques. On the other hand, high-level segmentation methods adopt deep learning techniques for training. Most deep learning-based 4D LF segmentation methods are applied only to the central view without considering other objects in the side views. However, when using sparse LFs, such as in immersive applications, other LF views, where disocclusions and additional objects may exist, must be considered. Moreover, the available deep neural networks for high-level semantic segmentation are often supervised, and, thus, inevitably demand pixel-wise GT segmentation labels for the training [16], [17], [19], which are challenging to obtain for all LF views, especially for real world LF datasets. Nevertheless, the use of deep learning has shown promising results in supervised 4D LF semantic segmentation and also in weakly-supervised and unsupervised 2D image segmentation [20]. Therefore, fully unsupervised 4D LF segmentation methods using deep learning are becoming increasingly appealing. The reason is that it may help in extracting relevant features from the LF and reduce the effort of manually defining precise features, which can be quite challenging. Another possible approach is to adapt pre-trained 2D foundation models, e.g., [21], for 4D LF data for extracting deep features and exploiting them for segmentation tasks. Pre-trained 2D foundation models capture rich semantic information from large-scale data, thus, by exploiting the pre-trained knowledge, zero-shot or few-shot image segmentation can be achieved. However, an adaptation is needed when applied for 4D LF data to consider ensuring angular consistency constraints. Applying unsupervised mid-level segmentation can overcome the GT availability limitation by learning deep features from the input itself and enabling segmentation based on intrinsic features. Moreover, pseudo-segmentation labels for segmentation are often generated in mid-level unsupervised segmentation, starting with many classes, and then the pixel labels and feature representations are jointly optimized by updating the network parameters using gradient descent.

To sum up, this paper aims to overcome the main limitations in most existing (mid/high-level) 4D LF segmentation methods, namely: i) Relying on the user scribbles or supervision; ii) Only supporting densely sampled 4D LFs; iii) Only applying segmentation to the central view; and iv) Not adequately exploiting LF view correlation or ensuring angular consistency across LF views. Accordingly, the main contributions of this paper are:

- **Proposal of a novel unsupervised angularly consistent 4D LF segmentation method for dense and sparse LFs** – In this paper, 4D LFs are segmented into (mid-level) objects without any prior supervision or user scribbles. To the best of the authors' knowledge, this is the first (mid-level) 4D LF segmentation technique that exploits deep features to segment objects without supervision for both dense and sparse LFs. Additionally, the segmentation is applied simultaneously to all LF views that compose a 4D LF, ensuring angular consistency throughout. This is achieved by initially over-segmenting 4D LFs into hyperpixels (where corresponding pixels across LF views are grouped according to their similarity in terms of color/texture, position and depth/disparity into the same hyperpixel) using the method proposed by the authors in Hamad et al. [10] to provide a compact LF representation.

- **Use of Graph Neural Networks (GNNs) for 4D LF segmentation** – In this paper, to efficiently deal with the large amount of 4D LF data, a novel hypergraph representation based on 4D LF over-segmentation is used. To exploit the advantage of deep learning techniques, a GNN is used on the graph-structured 4D LF. While GNNs have shown promising results in node classification and 2D image segmentation, to the best of the authors' knowledge, this is the first time a GNN has been applied to unsupervised 4D LF segmentation.
- **Proposal of a set of complementary metrics for evaluating segmentation angular consistency** – Although both spatial accuracy and angular consistency should be considered when evaluating 4D LF segmentation methods, existing 4D LF mid/high-level segmentation methods are often only evaluated in terms of spatial accuracy. Therefore, this paper proposes a set of complementary metrics that together enable evaluating the segmentation angular consistency, to be used in addition to spatial accuracy evaluation metrics. These metrics can be used for both dense and sparse LFs.

The remainder of the paper is organized as follows. Section II briefly reviews the related work on 4D LF segmentation. Section III describes the proposed unsupervised 4D LF segmentation method. Section IV presents the proposed segmentation angular consistency metrics. Section V includes experimental results to evaluate the proposed method. Finally, Section VI concludes the paper with final remarks and directions for future work.

II. RELATED WORK

In the past decade, several proposals have been made for 4D LF segmentation, which can be categorized into three main categories depending on the level of semantic meaning of the obtained segments, as briefly reviewed in this section:

Low-level unsupervised over-segmentation methods, e.g., [5], [6], [7], [8], [9], [10], mainly group pixels into atomic regions, which share similar characteristics, e.g., color/texture, position and depth/disparity, without the need for label annotations or user scribbles. These regions are often used as a pre-processing step for subsequent tasks. Available low-level 4D LF over-segmentation methods can be classified as either clustering-based or graph-based, depending on the approach used to divide 4D LFs into homogeneous regions. In the case of clustering-based methods, the K-means clustering algorithm is often used. K-means is usually applied to all 4D LF views with different approaches, such as starting K-means clustering in the central view and applying label propagation into all other LF views, as in [6], [7], or applying K-means clustering for the entire 4D LF as in the hyperpixels method [10]. In the case of graph-based methods, the 4D LF is represented by a weighted undirected 4D graph where each pixel is considered as a graph node, as proposed by Li et al. [8]. Afterwards, LF over-segmentation is achieved by maximizing the graph entropy in the 4D LF domain. While 4D LF over-segmentation can be achieved using graph techniques, applying graph optimization on a huge number of pixels requires extensive computational resources.

Mid-level semi-supervised segmentation methods group the pixels into objects without including semantic labels, e.g., [11], [14], [15]. In this case, user scribbles are usually inserted in the central view and the entire LF views are segmented accordingly [11], [12], [13], [14], [15]. For mid-level 4D LF segmentation, a common approach is to represent the 4D LF as a graph and apply classical graph-cut optimization assisted by the user scribbles (a.k.a. semi-supervised or scribble-supervised segmentation). However, representing each light ray as a graph node leads to a huge number of nodes, and thus can increase the processing complexity [12]. To reduce the number of graph nodes, corresponding pixels across LF views that represent the same 3D point (a.k.a. a ray bundle) are represented by a graph node [13]. To further reduce the graph size, the 4D LF can be represented by a hypergraph by exploiting the spatio-angular correlation across views [14], [15]. To achieve that, low-level 4D LF over-segmentation is first applied. Then, a hypergraph is created where 4D segments (i.e., corresponding pixels in all views that locally share similar criteria and represent the same 3D region) are represented by a hypernode. Although these methods reduce the graph size significantly, the over-segmentation methods they use to create hypergraphs are only suitable for dense LFs but not adequate for sparse LFs with large occlusions. Mid-level unsupervised segmentation methods can also be found in the literature, e.g., [22], [23], for specific applications such as transparent object segmentation and soft color segmentation, which are out of the scope of this paper.

High-level supervised semantic segmentation methods, e.g., [16], [17], [18], also predict semantic labels for each pixel. However, due to the lack of available LF datasets with GT segmentation labels for training deep neural network models, this has been a challenging research field in the past. New datasets for LF semantic segmentation have been proposed recently to support this research direction [17], [19], enabling the use of deep learning for this task [19], [17], [16], [18], [24]. To achieve supervised semantic segmentation, LF datasets with label annotations are required for training and evaluation. Therefore, a dataset with 400 real world LFs annotated for three foreground objects was created by Jia et al. [19] to train a Convolutional Neural Networks (CNN) based model. Later, Shen et al. [17] proposed a new dense 4D LF dataset for urban scenes (UrbanLF) annotated for 14 semantic classes. After the UrbanLF dataset was published, various 4D LF supervised semantic segmentation methods were proposed for urban scenes, e.g., [16], [18], [24]. Existing methods in this category can segment only specific objects (e.g., cars, buses and people). Additionally, they rely on supervision using GT

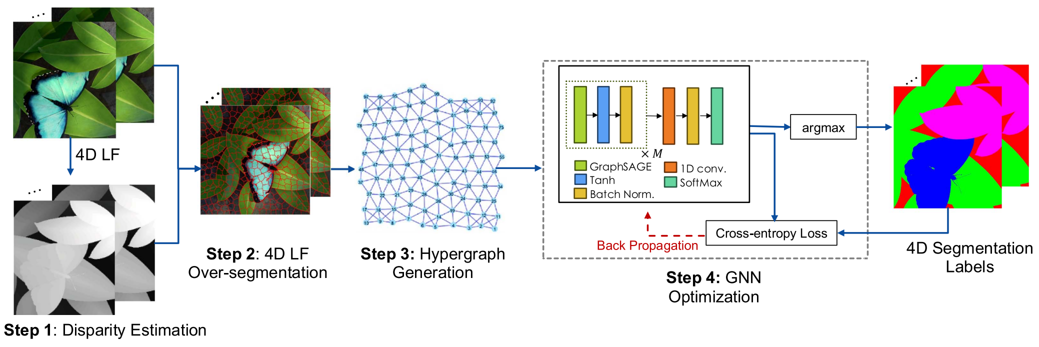

FIGURE 2. Main steps of the proposed 4D LF segmentation method. Given a 4D LF, the corresponding disparity maps for all views are initially estimated (Step1), next, the 4D LF is over-segmented into hyperpixels (Step2); after that, a hypergraph is generated, where each hyperpixel is represented as a hypernode (Step3); finally, a GNN optimization is performed in an unsupervised manner to obtain the 4D segmentation labels (Step4).

segmentation labels of densely sampled LFs. As the exploitation of weakly-supervised approaches to achieve high-level semantic segmentation shows promising results in 2D images [20], adapting these approaches for 4D LF could reduce the reliance on expensive and time-consuming fully annotated data.

III. PROPOSED 4D LIGHT FIELD SEGMENTATION METHOD

This paper proposes an unsupervised and angularly consistent mid-level 4D LF segmentation method for both dense and sparse static LFs. Given a 4D LF, the proposed method consists of four main steps, as summarized in Fig. 2. Each step is detailed in the following subsections. It is worth noting that the first two steps are considered pre-processing steps using existing methods, and the third and fourth steps include the contributions of this paper:

1) Disparity Estimation – Angularly consistent disparity maps are estimated for all LF views using an efficient disparity propagation method [25].
2) 4D LF Over-segmentation – The 4D LF is over-segmented into hyperpixels, which are consistent over the entire LF [10].
3) Hypergraph Generation – Once the hyperpixels are obtained, the 4D LF is represented as an undirected hypergraph, where each 4D hyperpixel is represented by a hypernode and two neighboring hyperpixels are connected by a hyperedge. Each hypernode is represented by a feature vector.
4) GNN Optimization – Finally, using the hypergraph as input, a GNN model is initialized and iteratively optimized, generating this way an unsupervised 4D LF segmentation, i.e., assigning a label for each hyperpixel, without any annotation effort or using user scribbles for supervision.

In this paper, we consider 4D LFs, however, the proposed method can be adapted and applied to other imaging modalities that can be represented by a graph (e.g., point clouds) and the GNN model may be adapted as well for other LF applications, such as LF inpainting and color editing.

A. DISPARITY ESTIMATION

As shown in Fig. 2 (Step 1), initially, disparity maps are estimated for all 4D LF views (with respect to its adjacent right view). Disparity information is inversely related to object depth and represents the difference in position of the same 3D point between two views. Therefore, it is a rich feature to guide the segmentation in terms of reducing edge ambiguity, especially when objects have similar colors or texture but different depths. Integrating disparity with other features during the segmentation helps deep learning models to learn spatial structures, leading to better performance, namely in terms of segmentation accuracy and angular consistency. The accuracy of the used disparity maps can affect the subsequent steps in terms of accuracy and angular consistency. Therefore, in this paper, to ensure disparity map angular consistency (i.e., corresponding pixels across views that represent the same 3D point must have the same disparity value), the disparity map of the central view is computed first using the method proposed by Shi et al. [26]. After that, the disparity of all other LF views is consistently propagated using the proposed disparity propagation method in [25]. For this step, any disparity estimation method that generates angularly consistent disparity maps for all views can be used.

B. 4D LF OVER-SEGMENTATION

As shown in Fig. 2 (Step 2), given the input 4D LF and the estimated disparity maps for all LF views, 4D LF over-segmentation is applied as a pre-processing step using the proposed method in Hamad et al. [10] to generate "hyperpixels". This step is useful for the proposed LFSGNN method since it handles the spatial shifts across LF views, due to the viewing angle, by grouping corresponding and similar pixels into hyperpixels in both dense and sparse LFs. As mentioned above, 4D LF over-segmentation into hyperpixels enables generating a more compact graph representation and reduces the number of nodes significantly (e.g., compared to using each pixel/light ray as a graph node). Moreover, using regular square segments of an image (a.k.a batches) to represent graph nodes as proposed in [4] may result in non-smooth borders; instead, it is more robust to use homogenous regions that adhere well to object boundaries and ensure angular consistency across all LF views. Since applying traditional 2D over-segmentation to each LF view independently will not ensure angular consistency, 4D hyperpixel over-segmentation is adopted in this paper [10].

The 4D hyperpixel over-segmentation method is used since it outperforms other existing 4D LF over-segmentation methods in terms of spatial accuracy and angular consistency [10]. Moreover, it enables a flexible and adaptive over-segmentation over the entire 4D space for both dense and sparse LFs. It is shown in [10] that using accurate disparity maps improves the 4D hyperpixel over-segmentation and enables better adherence to the object boundaries, subsequently, improving the final mid-level segmentation, as explained in Section V. The default hyperpixel size in [10] (i.e., 20) is used when applying 4D LF over-segmentation.

Each hyperpixel is represented by different features including color, texture, and disparity of the original 4D LF. Both RGB and CIELAB color spaces are used to represent the color feature since it has been shown that there is an advantage in combining those two color spaces [27]. With respect to the CIELAB color space, in this paper, only the chromatic components are used (i.e., a and b channels) to reduce the impact of variations in illumination. Moreover, a texture feature is computed using the Local Binary Pattern (LBP) texture descriptor [28] over the grayscale version of the original RGB 4D LF.[1] The LBP computes the local variation of pixels with high discriminative power and robustness to illumination changes, being a widely adopted effective approach in many computer vision applications. The LBP adopted in this paper uses up to 256 unique patterns and the texture descriptor of each hyperpixel is the pixel histogram over these 256 bins. Moreover, for the RGB, CIELAB and disparity features, the arithmetic mean values computed for each hyperpixel are used as hyperpixel features. Thus, this results in a 262-dimension feature vector, $h = [\bar{R},\ \bar{G}, \bar{B}, \bar{a}, \bar{b}, \bar{D}, (LBP_{hist})_{\times 256}\ ]$, for each hyperpixel. Given the differences in feature ranges, all features are normalized to $[0,\ 1]$. Normalizing these features will be useful later for the GNN optimization step, ensuring stable learning and proper optimization.

C. HYPERGRAPH GENERATION

As shown in Fig. 2 (Step 3), the hyperpixels that were computed in the previous step are used to generate a hypergraph. Each hyperpixel is represented as a hypernode of a hypergraph denoted by $G = (\upsilon, \varepsilon)$, where υ represents the set of hypernodes and ε represents the set of undirected and unweighted hyperedges that represent the adjacent relationship between the nearest neighboring hyperpixels. Every hypernode is, therefore, represented by a 262-dimension feature vector as explained in the previous step. In this paper, all hyperpixels that share a common boundary are considered neighbors, as illustrated in Fig. 3.

[1][Online]. Available: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

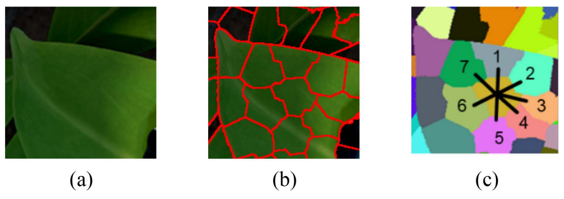

FIGURE 3. Example of a single hyperpixel neighbors (only a 2D slice is shown): a) Part of the original LF; b) Corresponding hyperpixels boundaries; c) Set of neighbors that share common boundaries with a given hyperpixel. This is illustrated for a single hyperpixel but applies to all hyperpixels.

The hypergraph, hypernode and hyperedge concepts are used in this paper since a hyperpixel typically has a 2D slice in some or all LF views. Hence, each slice can be considered as a classical graph node. However, since all hyperpixel slices represent the same hyperpixel, using the hypernode concept enables a more compact 4D LF representation. The hypergraph generation is flexible to any input resolution and the number of hypernodes may differ from one LF to another according to the input resolution and hyperpixel size. Therefore, there is no explicit adjustment needed to handle different 4D LF datasets. The hypergraph generated in this step is used as input to the GNN model, as detailed in the next step.

D. GNN OPTIMIZATION

The main goal of the proposed method is to classify all pixels in a 4D LF into an arbitrary number of classes c ($C_{\min} \leq c \leq C_{\max}$), where $C_{\min}$ is the minimum number of classes/labels and $C_{\max}$ is the initial maximum number of classes/labels, as detailed in Section V-B. To do so, the proposed method uses a GNN to achieve an unsupervised 4D LF segmentation by merging the initial hyperpixels and labeling the 4D LF with c unique labels based on the LF content.

1) GRAPH NEURAL NETWORK

A GNN is a neural network designed to process graph-structured data. The key idea behind GNNs is to enable each node to aggregate information from its neighbors through edges. Therefore, GNNs can capture complex dependencies within the graph data at different levels of abstraction. Each layer of a GNN typically comprises two primary operations: message passing and aggregation. In the message passing operation, node information is gathered and exchanged between neighboring nodes. In the aggregation operation, all the information gathered from the previous operation is fused for each node into one message to update its current state. GNNs have shown appealing performance in various applications, including node classification (e.g., in social networks and multi-label image segmentation). In this paper, the inductive and scalable Graph Sampling and Aggregation (GraphSAGE) framework is adopted which is widely used for node classification [29]. The key idea of GraphSAGE is to generate a node embedding (i.e., a low-dimensional vector representation of nodes in a graph) by learning an aggregation function from the representation of its neighbor nodes. The reason for using GraphSAGE lies in its ability to effectively capture local structural information of graph nodes and its scalability to process large-scale graph data and handle high-dimensional feature spaces [29]. Moreover, it enables sampling only a subset of neighboring nodes (which can be randomly selected or by using other advanced methods) to conduct propagation instead of using all the neighborhood information. This can help in reducing the computational complexity and makes the model less likely to overfit to specific structures in the training data. In GraphSAGE, the message passing operation can be considered a generalization of the traditional CNN on regular grids, where the convolution operation is replaced by the aggregator function. The mean aggregator (a.k.a. convolutional aggregator) is used in this paper, which processes each node in the graph as formulated in (1):

$$h_v^k = W_1 h_v^{k-1} + W_2 \sum_{u \in \mathcal{N}(v)} \frac{h_u^{k-1}}{|\mathcal{N}(v)|}, \tag{1}$$

where h_v^k is the current state of node v, h_v^{k-1} and h_u^{k-1} are the previous states of node $\boldsymbol{v}$ and $\boldsymbol{u}$, respectively, $\mathcal{N}(v)$ is the set of neighbors of node v ($|\mathcal{N}(v)|$ represents the number of its neighbors), and $W_1$and W_2 are matrices of the network parameters that need to be optimized. The main objective during training is to optimize W_1 and W_2 to make the node representations as informative and predictive as possible, based on their local graph structure. Optimizing W_1 and W_2 leads to minimizing the final loss through backpropagation.

2) PROPOSED GNN MODEL ARCHITECTURE

The high-level architecture of the proposed model is shown in Fig. 2 (Step 4) and contains M consecutive components, each of which contains a GraphSAGE operator, a hyperbolic tangent activation (tanh) function, and a batch normalization function. The batch size corresponds to one full LF hypergraph. The input of the proposed model is one matrix with the features of all hypernodes and the adjacency list (i.e., a list of neighbors for each hyperpixel). The size of the hidden channels of all GraphSAGE operators, as well as the output channels, is set to s. Therefore, s-dimensional feature maps are computed from the hypergraph G. The value of $\boldsymbol{s}$ is set in this paper to the same value as $C_{\max}$. Afterwards, the SoftMax operator is applied to the model output to obtain the probability distribution over predicted output classes. Since GT segmentation labels are not used in the proposed method, pseudo-segmentation labels are obtained using the argmax operator to find the dimension with maximum probability. The maximum probability value for each hypernode is selected and, when applying argmax for all hypernodes, the pseudo-segmentation labels are obtained. These pseudo- segmentation labels are used as target labels in the loss function as shown in Fig. 2. The loss is then computed between the model output and the pseudo-segmentation labels that are obtained in each epoch. For the loss function, L, the cross-entropy loss is used since it is effective and widely used in multi-label classification.

$$L(q,i) = -\sum_{c=1}^{C_{\max}} \log \frac{\exp(q_c)}{\sum_{j=1}^{C_{\max}} \exp(q_j)}, \tag{2}$$

where q_c represents the probability of the pseudo- segmentation label, c, and q_j denotes the probability of the j^{th} class and $i, j \in \{1, 2, \ldots C_{\max}\}$. The unsupervised 4D LF segmentation is achieved by applying both forward and backward passes with respect to a loss function to optimize model parameters. In the forward pass, the segmentation labels are predicted using fixed network parameters. However, in the backward pass, the network parameters are trained with fixed-label predictions as in [30]. The error signal is finally backpropagated to update the learnable parameters, which are initialized by default with Kaiming He initialization [31]. The model is iteratively trained until the maximum number of epochs or the minimum number of labels, $C_{\min}$, is reached. Finally, the predicted labels for the hypernodes are then mapped back to represent the 4D segmentation map. This is achieved by assigning the predicted label of the corresponding hypernode to all pixels in 4D space that belong to the corresponding hyperpixel.

IV. SEGMENTATION ANGULAR CONSISTENCY METRICS

Different from conventional 2D image segmentation where only the segmentation accuracy is important, in 4D LF segmentation angular consistency must also be considered.

To evaluate the segmentation angular consistency, we used the Labels per Pixel (LP) metric that was proposed to evaluate angular consistency for LF over-segmentation [7]. Initially, $LP = 1$ for all pixels in the central view (i.e., one unique label in the central view). To compute the LP metric, all LF views are warped into the central view using GT disparity maps. Afterwards, for each pixel position, the number of labels that have different values than the label in the central view is counted and then added to the initial LP value. Then, the $\overline{LP}$ value for all pixels in the central view is computed (higher value indicates worse angular consistency). However, the LP metric is adequate for dense LFs only, since when warping the views, all pixels in off-central views that are not seen in the central view due to the viewing angle are discarded. Hence, it is not adequate for sparse LFs with a large disparity range.

Therefore, inspired by the LP metric, we propose a set of complementary metrics to evaluate the 4D LF segmentation angular consistency for both dense and sparse LFs: i) Segmentation Angular Consistency (SAC); ii) Percentage of Inconsistent Pixels (IP); and iii) Average Local LP for Inconsistent Pixels ($\overline{LLP}_{IP}$). Each metric is explained below. All the proposed metrics rely on computing LP in a local window. To achieve that, LF views are warped into a locally central view within a local window of views (i.e., 3 × 3 views) and then the LP is computed locally, termed Local LP (LLP) in this paper. To consider all local windows in a 4D LF, this process is repeated by sliding the window one angular position

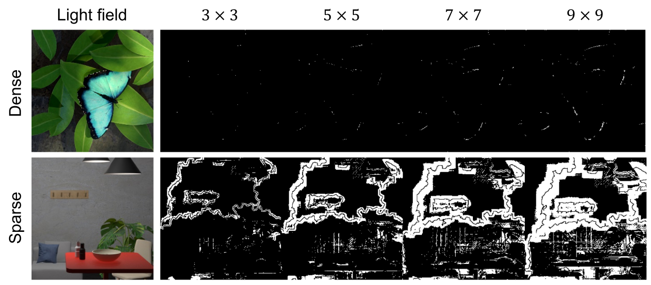

FIGURE 4. Visualization of the LLP for two LFs central views, where white pixels indicate pixel positions with $\overline{LLP}>1$, using different window sizes.

each time and computing the LLP for each window. To ensure accurate warping and adequately consider occlusions, we project a pixel from each view into the window's central view only if both have the same GT segmentation label. Moreover, as proposed in [7], when pixel overlapping occurs during the warping (i.e., projecting pixels of different objects from the off-central view into the same pixel position in the target view due to an occlusion), the foreground pixel is considered (i.e., the one with highest disparity). After warping all views of the window into the window's central view, the LLP metric is computed for each local window and the average LLP, $\overline{LLP}$, is calculated for all windows in the LF.

After computing the $\overline{LLP}$, the SAC metric is computed as formulated in (3). In a local window, N, represents the total number of views, which also corresponds to the maximum possible LLP value:

$$SAC = \frac{N - \overline{LLP}}{N - 1}. \tag{3}$$

This metric measures the segmentation angular consistency of light rays, where a higher value indicates better angular consistency. For example, if all views in a local window of $N = 3 \times 3$ have the same segmentation labels for corresponding light rays, i.e., $\overline{LLP} = 1$, then the SAC will have its highest value, i.e., $SAC = 1$, regardless of the N value; on the other hand, if each LF view has a different label for the same light ray (worst case scenario), then $\overline{LLP} = N$ and $SAC = 0$, which implies no angular consistency.

As a complement to the SAC metric, IP and $\overline{LLP}_{IP}$ are computed in this paper to highlight the percentage of inconsistent pixels across LF views (i.e., pixels where $\overline{LLP} > 1$) and the average LLP in those inconsistent pixels, respectively. As can be seen in Fig. 4, larger window sizes show more inconsistent pixels, especially in sparse LFs. The influence of using different window sizes on the proposed metrics is presented in Section V-E.

V. EXPERIMENTAL RESULTS

In this section, the proposed 4D LF Segmentation method using a GNN, from here on simply called LFSGNN, is evaluated both quantitively and qualitatively. Different 4D LF datasets are used in our experiments, including dense and sparse, synthetic, and real world LF datasets. Moreover, to evaluate the segmentation results in terms of spatial accuracy and angular consistency, different metrics that rely on the availability of the GT segmentation labels and disparity maps are considered. Since the real LF dataset does not have GT segmentation labels or disparity maps, only visual results are presented in this paper for this dataset. Regarding the segmentation angular consistency, in this paper, only the central view and central EPIs are shown to illustrate the angular consistency. However, to be able to observe the angular consistency across all LF views, we highly encourage the reader to observe the dynamic results, where LF views are scanned in serpentine order and presented as videos, in the supplemental materials available online for all test LFs.[2]

A. 4D LF DATASETS

The proposed unsupervised 4D LF segmentation method does not target a specific domain (e.g., urban scenes or medical images). Therefore, any 4D LF dataset can be used to evaluate the proposed method. However, to quantitatively evaluate the segmentation accuracy and consistency, the GT disparity maps and GT segmentation labels for all LF views are needed. Therefore, the synthetic HCI [32] and IT-4DLF [33] datasets are used since they provide the GT disparity maps and segmentation labels for all views of the dense and sparse LFs, respectively. Moreover, to validate our results on real world LFs, the MMSPG dataset captured with a Lytro Illum camera [33] is used, considering the central 9×9 views to eliminate the vignetting effects (i.e., saturation or darkening at the edges of a lenslet image compared to the center). It is worth noting that the 4D LF datasets designed for supervised semantic segmentation, summarized in [17] are not adequate for evaluating the proposed method. The reason is that multiple objects with different visual appearances may be classified to the same semantic label (e.g., blue and red cars have the same semantic label), which is not the case envisaged in the proposed method. A summary of the used LF datasets can be found in Table 1.

B. IMPLEMENTATION DETAILS

Firstly, the proposed method does not require splitting datasets for training and testing since it is unsupervised, and the learning parameters are optimized for each LF independently. Moreover, since hypergraphs are used to represent LFs, hypergraph generation is flexible to different input resolutions. In this paper, all experiments were run on a desktop computer with a 64-bit Ubuntu operating system, AMD Epyc 7282 16-core CPU, NVIDIA GeForce RTX 3090 and 256 GB RAM. Our network is implemented using Pytorch (2.1.1) and the network is optimized using a Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of 0.05. Momentum and weight decay are set to 0.9 and 0.01, respectively. The learning rate scheduler decays the learning rate by multiplying it by

[2][Online]. Available: https://github.com/MaryamHamad/LFSGNN

TABLE 1. Light Field Datasets Used in the Experimental Results

4D LF dataset	View resolution in pixels ($N_x \times N_y$)	Number of views ($N_u \times N_v$)	Disparity range
HCI dataset [32]: Papillon, Buddha, , StillLife and Horses	768×768 except for Horses: 1024×576	9×9	[-4, 4]
MMSPG dataset [33]: Poppies and Swans	625×434	15×15	[-1, 1]
IT-4DLF dataset [10]: Kitchen, Room and Antique	512×512	9×9	[-18, 18]

0.95 every 50 epochs. The maximum number of epochs is set to 1000. The number of M components of the network is set to 2. The maximum number of classes, $C_{\max}$, is set to 128 as a large number to start the segmentation process, and the minimum number of labels, $C_{\min}$, is set to 5 as a reasonable number of objects in the test datasets.

C. BENCHMARK METHODS

As there are currently no available methods that target fully unsupervised 4D LF mid-level segmentation, we compare our results with state-of-the-art unsupervised 2D image segmentation methods applied on 4D LF content without changing their model architectures. The first method is proposed by Kim et al. [30], which is fully unsupervised and adopts a conventional 2D CNN to extract deep features. Afterwards, segmentation labels are assigned according to the response vector using an argmax function. Then the segmentation labels are used as pseudo-segmentation labels to compute the final loss. Finally, image segmentation is achieved by iteratively minimizing the loss function until a maximum number of epochs or the minimum number of labels is reached. The second method is proposed by Aflalo et al. [4], in which deep features are extracted from an available pre-trained vision transformer. The used transformer divides an image into square patches (patch size is constant as detailed in [4]) where each patch represents extracted features. Those patches are then used to represent an image as a graph, where each patch represents a graph node. The created graph is then input to a lightweight GNN model with one graph convolutional layer to apply unsupervised segmentation for 2D images. Although this method exploits an existing pre-trained model to extract the features, the idea of representing an image as a graph based on local regions in the image and applying the GNN technique makes it directly related to the proposed method.

Both above-mentioned methods are designed for 2D images. To use them on 4D LF data, they are applied to each LF view independently. However, to promote view consistent segmentation, for each method, the same initial values for all training parameters are used for each LF view and the random values generated using a fixed seed to ensure reproducibility for each LF view. This ensures similar behavior for feature extraction according to the used initialization of the training parameters. Moreover, the benchmark methods were executed using their available Pytorch implementation and, for a fair comparison, the minimum number of segmentation labels is set to 5 for all methods.

This paper did not make comparisons with the high-level supervised 4D LF segmentation methods since they rely on segmenting objects according to their semantic labels, which is not the case considered in the proposed method.

D. EVALUATION METRICS

The segmented 4D LFs are evaluated, in this paper, in terms of spatial accuracy and angular consistency considering all views. As explained in Section V, the LP and SAC metrics are used to evaluate the non-local and local angular consistency, respectively, for both dense and sparse LFs. To evaluate the segmentation accuracy, the mean Intersection over Union (mIoU) metric is used, which is a common metric widely used to evaluate mid-level segmentation accuracy. The mIoU metric measures the amount of overlap between the GT and predicted segmentation labels. Since the proposed method is unsupervised, the predicted segmentation labels are not necessarily the same as the GT segmentation labels in terms of their values or number (this is valid for all benchmark methods). Therefore, to calculate the mIoU metric, for each label in the GT label images, the largest overlapping between that label and all the predicted labels in the corresponding location is considered, as described in [30].

E. INFLUENCE OF DISPARITY MAPS QUALITY ON THE PROPOSED METHOD

To highlight the importance of the disparity map quality on the segmentation accuracy and angular consistency, the proposed method is tested by using estimated and GT disparity maps (for both the proposed method and the hyperpixels over-segmentation on which it relies).

To evaluate the angular consistency, the proposed SAC, IP and $\overline{LLP}_{IP}$ metrics are computed considering different values of N (i.e., 3×3, 5×5, 7×7 and 9×9) to explore the window size influence. As presented in Table 2, the proposed metrics are influenced by the window size for both dense and sparse LFs due to the maximum possible number of labels in a window of views and the discarding of occluded or non-existent (due to the viewing angle) pixels in the central view when computing the LLP. Additionally, knowing that ideally, the GT segmentation labels have $\overline{LLP} = 1.0$ and $SAC = 1.0$ for a given N helps in identifying how close the angular consistency of the predicted labels is to the GT labels. In all upcoming experiments (if N value is not specified), the metrics are computed for $N = 3 \times 3$; to reduce discarding pixels that are occluded or non-existent in the window central view, especially for sparse LFs.

As has been demonstrated in [10], more accurate disparity maps can positively affect the hyperpixels angular consistency. Since the proposed method relies heavily on the

TABLE 2. Light Field Segmentation Angular Consistency Metrics With Different Window Sizes Using Estimated Disparity Maps

LF \ N	3×3			5×5			7×7			9×9		
	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$
Papillon	**1.00**	**0.04**	**2.35**	1.00	0.14	2.93	1.00	0.27	3.52	1.00	0.42	4.18
Buddha	**1.00**	**0.05**	**2.52**	1.00	0.15	3.15	1.00	0.29	3.91	1.00	0.48	4.21
StillLife	**1.00**	**0.13**	**2.55**	1.00	0.35	3.32	1.00	0.64	3.98	1.00	1.06	4.30
Horses	**1.00**	**0.11**	**2.77**	1.00	0.48	3.49	1.00	0.89	4.47	1.00	1.46	5.50
Kitchen	**1.00**	**0.11**	**2.99**	1.00	0.29	3.86	1.00	0.49	4.92	1.00	0.74	6.20
Room	**0.98**	**8.91**	**2.98**	0.97	19.15	5.34	0.95	28.78	8.59	0.94	38.65	12.52
Antique	**1.00**	**0.17**	**2.39**	1.00	0.54	2.79	1.00	1.09	3.17	1.00	1.70	3.64
Avg.	**1.00**	**1.36**	**2.65**	1.00	3.01	3.55	0.99	4.63	4.65	0.99	6.36	5.79

TABLE 3. Light Field Segmentation Angular Consistency Metrics With Different Window Sizes Using GT Disparity Maps

LF \ N	3×3			5×5			7×7			9×9		
	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$	SAC	IP [%]	$\overline{LLP}_{IP}$
Papillon	**1.00**	**0.02**	**2.60**	1.00	0.04	3.81	1.00	0.05	5.01	1.00	0.09	6.14
Buddha	**1.00**	**0.03**	**2.99**	1.00	0.05	4.46	1.00	0.06	5.92	1.00	0.10	6.36
StillLife	**1.00**	**0.18**	**2.72**	1.00	0.30	4.35	1.00	0.40	6.16	1.00	0.59	7.11
Horses	**1.00**	**0.04**	**3.07**	1.00	0.08	4.96	1.00	0.13	6.96	1.00	0.17	8.70
Kitchen	**1.00**	**0.07**	**2.94**	1.00	0.19	3.98	1.00	0.31	5.65	1.00	0.45	7.94
Room	**1.00**	**0.07**	**2.71**	1.00	0.17	3.39	1.00	0.33	4.57	1.00	0.70	6.15
Antique	**1.00**	**0.01**	**2.56**	1.00	0.03	3.18	1.00	0.05	4.07	1.00	0.08	4.44
Avg.	**1.00**	**0.06**	**2.80**	1.00	0.12	4.02	1.00	0.19	5.48	1.00	0.31	6.69

TABLE 4. Quantitative Results Using Estimated and GT Disparity Maps

	LFSGNN using estimated disparity		LFSGNN using GT disparity	
LF	mIoU	LP	mIoU	LP
Papillon	**0.63**	1.01	0.62	**1.00**
Buddha	0.75	1.02	**0.86**	**1.01**
StillLife	**0.51**	**1.04**	0.43	1.04
Horses	0.28	1.07	**0.35**	**1.01**
Kitchen	0.20	1.04	**0.32**	**1.03**
Room	**0.26**	5.45	0.16	**1.04**
Antique	0.17	1.04	**0.18**	**1.00**
Avg.	0.40	1.67	**0.42**	**1.02**

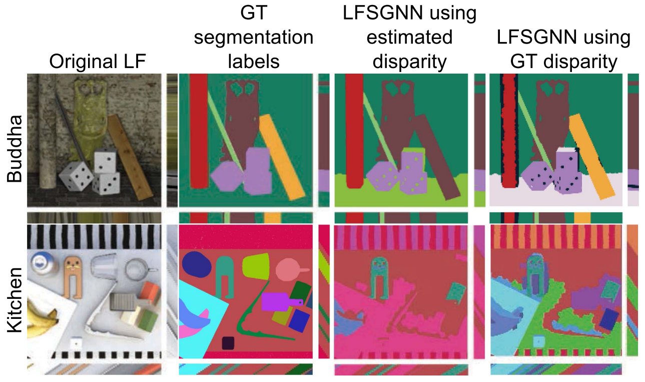

FIGURE 5. Examples of the influence of disparity map quality on the proposed method. The estimated disparity maps have smoother borders between objects and hence can merge different objects easier and faster especially when the $C_{\min}$ is less than the number of objects in the scene.

hyperpixels, the quality of the used disparity maps also influences the segmentation performance in terms of accuracy and angular consistency as shown in Tables 2, 3, 4 and Fig. 5. As can be noticed from Tables 2, 3 and 6 the proposed method in most cases achieves higher angular consistency when using more accurate disparity maps (e.g., GT disparity maps in this experiment), which indicates that most of the pixels in the central view of a given window have the same label across all views of that window. For sparse LFs, such as the Room LF (which contains large occlusions), the accuracy of the used disparity map significantly affects the segmentation angular consistency, as shown in Tables 2 and 3 (in this paper, bold style indicates better performance for all tables). Notice that in some cases (as in StillLife in Table 2), the angular consistency metrics show better performance when using estimated disparity maps. The reason for this is the smooth disparity values in estimated disparity maps can merge objects easier and faster (i.e., ends up segmentation with a fewer number of labels) compared to using sharp GT disparity maps, thus reducing the unique labels and warping error when computing the segmentation angular consistency metrics. Moreover, for the same reason, the $\overline{LLP}_{IP}$ is higher when using GT disparity maps in most LFs.

F. ABLATION STUDY

Before comparing our results to the benchmark methods, an ablation study to investigate different configurations of the

TABLE 5. Quantitative Results With and Without Using the Texture Feature During the Hypergraph Generation and GNN Optimization Steps

	LFSGNN with texture feature					LFSGNN without texture feature				
LF	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$
Papillon	**0.63**	**1.01**	**1.00**	**0.04**	**2.35**	0.62	1.04	1.00	0.05	2.39
Buddha	**0.75**	**1.02**	**1.00**	**0.05**	**2.52**	0.46	1.03	1.00	0.05	2.58
StillLife	**0.51**	**1.04**	**1.00**	**0.13**	**2.55**	0.48	1.05	1.00	0.23	2.60
Horses	0.28	**1.07**	**1.00**	**0.11**	2.77	**0.31**	1.07	1.00	0.12	**2.70**
Kitchen	0.20	1.04	**1.00**	0.11	2.99	**0.24**	**1.02**	**1.00**	**0.07**	**2.89**
Room	0.26	5.45	**0.98**	8.91	**2.98**	**0.33**	**5.27**	**0.98**	**8.48**	**2.98**
Antique	0.17	**1.04**	**1.00**	**0.17**	**2.39**	**0.22**	1.06	1.00	0.20	2.42
Avg.	**0.40**	1.67	**1.00**	1.36	**2.65**	0.38	**1.65**	1.00	**1.31**	**2.65**

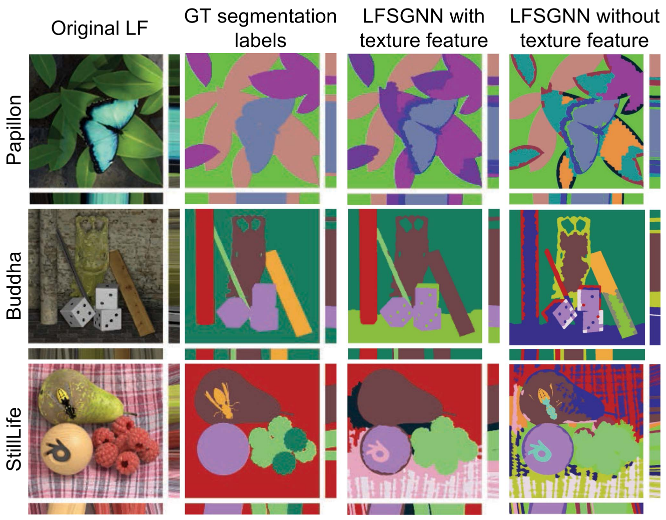

FIGURE 6. Examples of the influence of using the texture feature during the hypergraph generation and GNN optimization steps.

proposed method is presented. Initially, to study the influence of using the texture feature, the proposed method is tested with and without using the LBP texture descriptor in hypergraph generation and during the GNN optimization. As shown in Fig. 6, by incorporating texture features, the learning process in most LFs converged faster to $C_{\min}$ value, in contrast to when the texture feature was not used (converges slower, did not approach the $C_{\min}$ value and stopped based on the number of epochs). This can be noticed in Fig. 6 where the segmented LF has more labels than $C_{\min}$ when the texture feature was not used. Hence, it was stopped due to the epoch criteria being reached first. The rationale behind that is that the texture feature helps the model to learn a meaningful representation of the LF, and the evaluation metrics reflect that numerically, as in Table 5. The reason for Kitchen and Room achieving better accuracy when the texture feature is omitted is that the final segmented LF has more unique labels compared to when using the texture, which means considering most objects in the LF.

Hence, it can benefit the evaluation metrics (especially when the number of objects in the LF is larger than the minimum number of labels parameter) without necessarily improving the visual results. As can be seen in Table 5, there are no significant differences in segmentation angular consistency metrics in most LFs. The reason for this is that the segmentation angular consistency relies on the disparity estimation and 4D LF over-segmentation steps, and those steps do not rely on the texture feature. Hence, when using the same disparity maps, LFs are represented based on hyperpixels where the consistency is ensured similarly. The reason for the slight difference in the angular consistency metrics is typically due to the reached number of unique labels in the final segmented LFs (this can be noticed in Fig. 6 where segmented LFs without using the texture feature have more unique labels). Moreover, the rounding error in pixel projection when computing those metrics can also affect their results. To study the influence of the used number of components, M, in the proposed model (where the texture feature is included), different values of M are used and the results are reported in Table 6 which shows better performance in terms of angular consistency metrics when using larger M values without necessarily improving the segmentation accuracy (mIoU). This is typically because of a limitation in GNNs when increasing the GraphSAGE layers, the model over-smoothens the predicted segments, which leads to the merging of unrelated objects (resulting in a smaller number of labels). Accordingly, when the M value increases different objects are merged as shown in Fig. 7. Thus, the possibility of error occurrence in pixel projection when computing the angular consistency metrics is decreased. To avoid the over-smoothing effect, the value of $M = 2$ is adopted to lead to a reasonable balance between segmentation accuracy and angular consistency.

G. COMPARISON WITH THE BENCHMARK METHODS

Our results are compared with the benchmark methods on different LF datasets as shown in Figs. 8, 9 and Table 7. The proposed method and the benchmark methods are based on unsupervised learning; hence the values of the predicted labels differ from the GT segmentation labels. Therefore, to facilitate the visual comparisons, the labels of GT segmentation maps in the synthetic LFs are mapped to the labels of each method as shown in Fig. 9. The real world LF dataset does not have GT segmentation labels. Hence, the colors of the predicted labels

TABLE 6. Quantitative Results Using a Different Number of Network Model Components (M)

	LFSGNN when $M = 2$					LFSGNN when $M = 4$					LFSGNN when $M = 6$				
LF	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$
Papillon	**0.63**	**1.01**	**1.00**	0.04	2.35	0.57	1.01	**1.00**	0.04	2.35	0.53	1.01	**1.00**	**0.03**	**2.33**
Buddha	**0.75**	1.02	**1.00**	0.05	**2.52**	0.72	1.02	**1.00**	0.04	2.56	0.54	**1.01**	**1.00**	**0.04**	2.56
StillLife	0.51	1.04	**1.00**	0.13	2.55	0.42	1.05	**1.00**	0.20	2.62	**0.53**	**1.02**	**1.00**	**0.06**	**2.28**
Horses	0.28	1.07	**1.00**	0.11	2.77	**0.31**	1.05	**1.00**	0.10	**2.68**	0.27	**1.02**	**1.00**	**0.03**	2.73
Kitchen	0.20	1.04	**1.00**	0.11	2.99	**0.27**	1.06	**1.00**	0.14	2.81	0.26	**1.02**	**1.00**	**0.07**	**2.46**
Room	**0.26**	5.45	0.98	8.91	2.98	0.14	3.29	0.99	3.79	**2.89**	0.13	**2.61**	**0.99**	**3.10**	2.92
Antique	**0.17**	1.04	**1.00**	0.17	**2.39**	0.14	1.04	**1.00**	0.14	2.42	0.16	**1.02**	**1.00**	**0.08**	**2.39**
Avg.	**0.40**	1.67	**1.00**	1.36	2.65	0.37	1.36	**1.00**	0.64	2.62	0.35	**1.25**	**1.00**	**0.49**	**2.52**

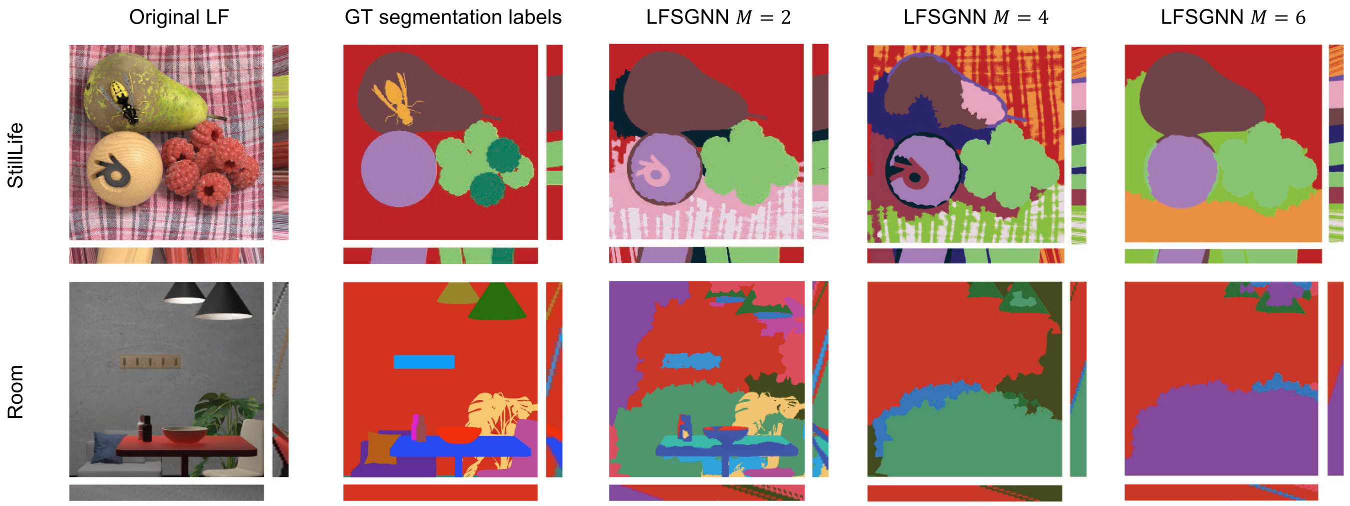

FIGURE 7. Examples of using the proposed method with a different number of components, *M*. In some LFs, such as Room, a higher value of *M* can result in over-smoothing and different objects can be merged which negatively affects the segmentation accuracy.

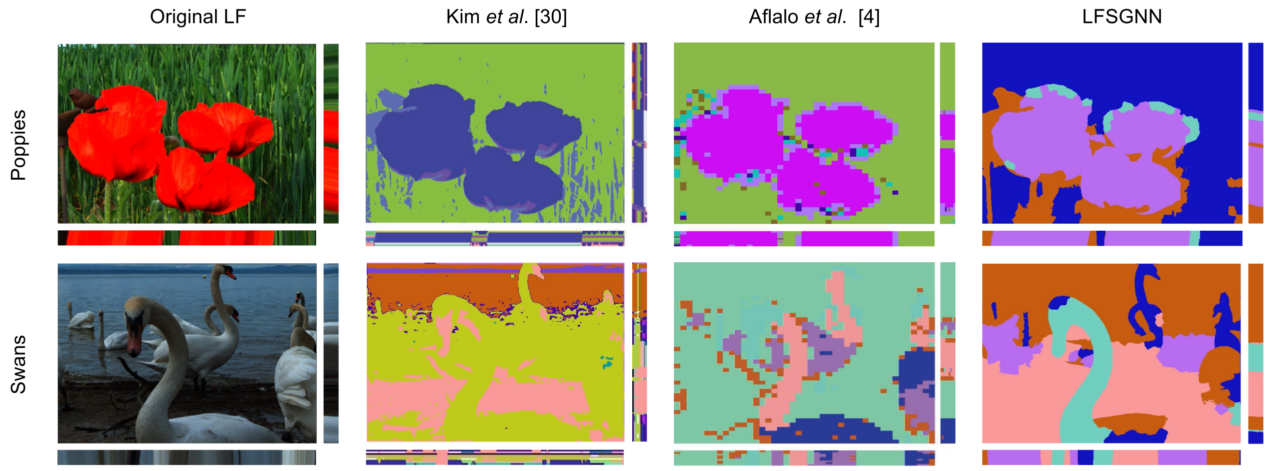

FIGURE 8. Examples of unsupervised 4D LF segmentations on real world LFs for different methods. For each LF, the central view and the central horizontal/vertical EPIs are presented to show the segmentation angular consistency across LF views. The minimum number of labels is set to 5. Our results ensure angular consistency as can be seen in the presented EPIs (composed of mostly regular slanted lines) and adhere to object boundaries according to the reached number of labels (e.g., the swan head).

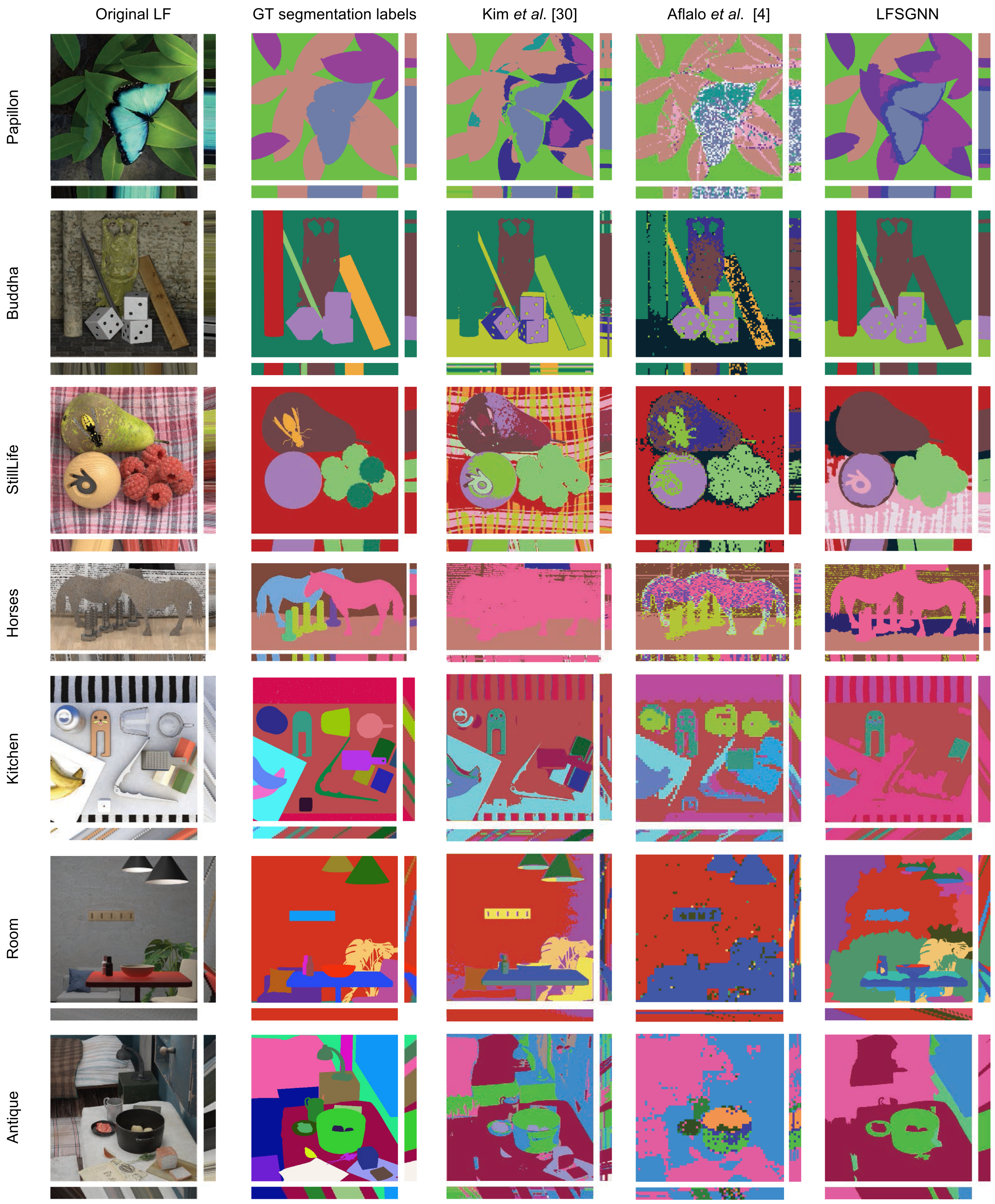

FIGURE 9. Examples of unsupervised 4D LF segmentation on synthetic LFs for different methods. For each LF, the central view and the central horizontal and vertical EPIs are presented to show the segmentation angular consistency across LF views. The minimum number of labels is set to 5. Depending on the reached number of labels in each LF view and the actual number of objects in each LF, benchmark methods in some 4D LFs achieve better visual accuracy in terms of adhering to the object boundaries. This happens especially in sparse LFs (e.g., Kitchen and Room) where the estimated disparity maps used for LFSGNN are not accurate for all pixels. However, for accurate disparity maps, such as in the dense LFs (e.g., Papillon, Buddha, StillLife and Horses), LFSGNN achieves better separation between objects that share similar color or texture but vary in their depth (e.g., leaves in Papillon and the left pillar in Buddha). The angular consistency is better in both sparse and dense LFs compared to the benchmark methods, as can be seen in the central EPIs (composed of mostly regular slanted lines).

TABLE 7. Quantitative Results Using Our Results and Benchmark Methods

	Kim *et al.* [30]					Aflalo *et al.* [4]					LFSGNN				
LF	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$	mIoU	LP	SAC	IP [%]	$\overline{LLP}_{IP}$
Papillon	0.52	12.98	0.93	22.91	3.41	0.53	10.72	0.89	26.87	4.32	**0.63**	**1.01**	**1.00**	**0.04**	**2.35**
Buddha	0.48	23.80	0.68	70.08	4.69	0.44	6.80	0.93	18.72	3.81	**0.75**	**1.02**	**1.00**	**0.05**	**2.52**
StillLife	0.41	3.16	0.97	8.56	3.50	**0.54**	7.12	0.94	18.32	3.57	0.51	**1.04**	**1.00**	**0.13**	**2.55**
Horses	0.24	11.73	0.91	19.89	4.36	**0.39**	10.10	0.91	24.66	3.77	0.28	**1.07**	**1.00**	**0.11**	**2.77**
Kitchen	**0.22**	18.70	0.77	47.91	4.85	0.20	31.63	0.78	61.60	3.86	0.20	**1.04**	**1.00**	**0.11**	**2.99**
Room	0.23	31.85	0.65	81.46	4.43	0.12	33.19	0.70	80.62	3.90	**0.26**	**5.45**	**0.98**	**8.91**	**2.98**
Antique	0.16	25.35	0.78	52.17	4.36	0.13	17.79	0.83	42.98	4.16	**0.17**	**1.04**	**1.00**	**0.17**	**2.39**
Avg.	0.32	18.22	0.81	43.28	4.23	0.34	16.76	0.86	39.11	3.91	**0.40**	**1.67**	**1.00**	**1.36**	**2.65**

across the used methods are not related to each other, as can be seen in Fig. 8.

In Table 7, the proposed segmentation method outperforms the benchmarks in terms of accuracy and consistency in most LFs. To visually notice the segmentation angular consistency, the central horizontal and vertical EPIs are presented for all methods. The segmentation angular consistency of all LF views can be more clearly noticed in the dynamic results in the supplemental material. Kim et al. [30] and Aflalo et al. [4] methods suffer from discontinuity in the predicted segments, where sparse pixels with wrong labels can be noticed in Fig. 9.

Although the same initial values are used for the training parameters in each of the benchmark methods to consider the same segmentation behavior for all LF views, the evaluation metrics still indicate inconsistent results with high values of LP. This highlights the importance of explicitly considering the angular correlation in 4D LFs and the effectiveness of the used hypergraph representation (which allows applying segmentation to the entire 4D LF simultaneously). As can be noticed in Table 7, benchmark results of mIoU in some LFs (e.g., in Kitchen) are the same or better than the proposed method but significantly worse in terms of angular consistency metrics. This is because the mIoU metric considers the largest overlapping between the GT segmentation label and all the predicted labels in LF views (as described earlier in this paper).

However, the angular consistency metrics are significantly affected by the inconsistency across LF views, which are essentially noticeable at the boundaries of objects. Moreover, for some LFs, the benchmark methods terminate before reaching the minimum number of labels, $C_{\min}$, since they have different architectures and differ in the used features.

Thus, they can end up with more unique labels than our method which may positively affect the used accuracy metric if the content of LF has more objects than the $C_{\min}$ value. Finally, using hyperpixels as the starting point allows exploiting the entire LF data during the segmentation and ensures segmentation angular consistency. Moreover, the proposed method is trained in an unsupervised manner which makes it suitable for different applications. One possible direction

TABLE 8. Average Running Time in Seconds Per View

LF dataset	Kim *et al.* [30]	Aflalo *et al.* [4]	LFSGNN				
			Step 1	Step 2	Step 3	Step 4	Total time
HCI dataset [32]	83.60	0.62	1.90	8.96	7.82	7.59	26.27
MMSPG dataset [33]	20.12	0.66	1.83	3.33	2.67	1.09	8.93
IT-4DLF dataset [10]	43.93	0.45	1.75	4.29	3.62	2.04	11.71

for improving the performance of the proposed method is to fine-tune the used features for domain-specific tasks (e.g., LF medical imaging). A major limitation of the proposed method is that it does not inherently determine the number of objects in an LF. Hence, a technique that adequately estimates the minimum number of labels based on the LF content would be extremely useful to further improve the segmentation results. Moreover, optimizing the implementation of the included steps can reduce the required computational time.

To compare the computational complexity between the proposed method and the benchmark methods, all methods were run on the same computer and the GPU is being used under similar conditions for all methods. The running times are reported in Table 8. The breakdown running time of all the steps of the proposed LFSGNN method is divided by the number of views to obtain running time per view for each step and reported in Table 8. The summation of all steps of the proposed method is also computed. In Table 8, although both our proposed method and Kim et al. [30] method iterate for 1000 epochs, our proposed method reduces the running time compared to Kim et al. [30] significantly. The method proposed by Aflalo et al. [4] iterates only for 10 epochs since it relies on an available pre-trained vision transformer to extract the deep features before performing image segmentation, thus it has the lowest running time in Table 8. Training their lightweight model beyond 10 epochs was also tried (i.e., up to 1000) but the performance did not show significant improvements. In fact, this shows one advantage, in terms of computational complexity, of extracting the deep features from pre-trained foundation models and then applying fine-tuning for a few iterations while performing a specific task as image segmentation. Considering the number of epochs, our proposed method has shown improvement in both segmentation accuracy, angular consistency and a significant reduction in the computational complexity.

VI. CONCLUSION

In this paper, a novel unsupervised angularly consistent 4D LF segmentation method is proposed for both dense and sparse LFs. Initially, the 4D LF is represented as a hypergraph based on 4D hyperpixel over-segmentation. Afterwards, a GNN model is designed to extract deep features of the hypergraph and to group the hypernodes into objects by applying message passing and aggregation iteratively until convergence is reached. Different from existing 4D LF segmentation methods, the proposed method is fully unsupervised, represents 4D LFs robustly and efficiently, exploits the power of deep learning of graph-structured data and supports both dense and sparse 4D LFs. Experimental results show outperforming segmentation performance for most dense and sparse 4D LFs in terms of segmentation accuracy, angular consistency, and computational complexity.

For future work, the proposed LFSGNN method can be adapted for other imaging modalities, such as point clouds and multi-view images. Additionally, the inclusion of pre-trained foundation models in the 4D LF segmentation task can be also an interesting direction for future work. Moreover, the resulting 4D LF segmentation can be used for other applications such as in augmented reality where a segmented 4D object can be inserted in other 4D LFs. Finally, extending the proposed LFSGNN method to LF videos by considering the temporal dimension is also an interesting research direction that requires further investigation.

REFERENCES

[1] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 31–42.
[2] M. Zhao et al., "A survey for light field super-resolution," *High-Confidence Comput.*, vol. 4, no. 1, Mar. 2024, Art. no. 100206.
[3] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022.
[4] A. Aflalo, S. Bagon, T. Kashti, and Y. Eldar, "DeepCut: Unsupervised segmentation using graph neural networks clustering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 32–41.
[5] M. Hog, N. Sabater, and C. Guillemot, "Superrays for efficient light field processing," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 7, pp. 1187–1199, Oct. 2017.
[6] H. Zhu, Q. Zhang, Q. Wang, and H. Li, "4D light field superpixel and segmentation," *IEEE Trans. Image Process.*, vol. 29, pp. 85–99, Dec. 2020.
[7] N. Khan, Q. Zhang, L. Kasser, H. Stone, M. H. Kim, and J. Tompkin, "View-consistent 4D light field superpixel segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7810–7818.
[8] R. Li and W. Heidrich, "Hierarchical and view-invariant light field segmentation by maximizing entropy rate on 4D ray graphs," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–15, Nov. 2019.
[9] M. Hamad, C. Conti, P. Nunes, and L. D. Soares, "ALFO: Adaptive light field over-segmentation," *IEEE Access*, vol. 9, pp. 131147–131165, Sep. 2021.
[10] M. Hamad, C. Conti, P. Nunes, and L. D. Soares, "Hyperpixels: Flexible 4D over-segmentation for dense and sparse light fields," *IEEE Trans. Image Process.*, vol. 32, pp. 3790–3805, Jul. 2023.
[11] S. Wanner, C. Straehle, and B. Goldluecke, "Globally consistent multi-label assignment on the ray space of 4D light fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1011–1018.
[12] H. Mihara, T. Funatomi, K. Tanaka, H. Kubo, Y. Mukaigawa, and H. Nagahara, "4D light field segmentation with spatial and angular consistencies," in *Proc. IEEE Int. Conf. Comput. Photography*, 2016, pp. 1–8.
[13] M. Hog, N. Sabater, and C. Guillemot, "Light field segmentation using a ray-based graph structure," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 35–50.
[14] X. Lv, X. Wang, Q. Wang, and J. Yu, "4D light field segmentation from light field super-pixel hypergraph representation," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 9, pp. 3597–3610, Sep. 2021.
[15] M. Hamad, C. Conti, A. M. De Almeida, P. Nunes, and L. D. Soares, "SLFS: Semi-supervised light-field foreground-background segmentation," in *Proc. Telecoms Conf.*, 2021, pp. 1–6.
[16] D. Yang, T. Zhu, S. Wang, S. Wang, and Z. Xiong, "LFRSNet: A robust light field semantic segmentation network combining contextual and geometric features," *Front. Environ. Sci.*, vol. 10, Oct. 2022, Art. no. 1443.
[17] H. Sheng, R. Cong, D. Yang, R. Chen, S. Wang, and Z. Cui, "UrbanLF: A comprehensive light field dataset for semantic segmentation of urban scenes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7880–7893, Nov. 2022.
[18] Y. Li et al., "Multi-view semantic information guidance for light field image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 3454–3462.
[19] C. Jia et al., "Semantic segmentation with light field imaging and convolutional neural networks," *IEEE Trans. Instrum. Meas.*, vol. 70, Sep. 2021, Art. no. 5017214.
[20] K. Zhu, N. N. Xiong, and M. Lu, "A survey of weakly-supervised semantic segmentation," in *Proc. IEEE 9th Int. Conf. Big Data Secur. Cloud, IEEE Int. Conf. High Perform. Smart Comput., IEEE Int. Conf. Intell. Data Secur.*, 2023, pp. 10–15.
[21] M. Caron et al., "Emerging properties in self-supervised vision transformers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 9630–9640.
[22] Y. Xu, H. Nagahara, A. Shimada, and R. –I Taniguchi, "TransCut2: Transparent object segmentation from a light-field image," *IEEE Trans. Comput. Imag.*, vol. 5, no. 3, pp. 465–477, Sep. 2019.
[23] P. Matysiak, M. Grogan, W. Aenchbacher, and A. Smolic, "Soft colour segmentation on light fields," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 2621–2625.
[24] R. Cong, D. Yang, R. Chen, S. Wang, Z. Cui, and H. Sheng, "Combining implicit-explicit view correlation for light field semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 9172–9181.
[25] M. Hamad, C. Conti, P. Nunes, and L. D. Soares, "Efficient propagation method for angularly consistent 4D light field disparity maps," *IEEE Access*, vol. 11, pp. 63463–63474, Jun. 2023.
[26] J. Shi, X. Jiang, and C. Guillemot, "A framework for learning depth from a flexible subset of dense and sparse light field views," *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 5867–5880, Dec. 2019.
[27] A. Borji and L. Itti, "Exploiting local and global patch rarities for saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 478–485.
[28] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Jun. 2017, pp. 1025–1035.
[30] W. Kim, A. Kanezaki, and M. Tanaka, "Unsupervised learning of image segmentation based on differentiable feature clustering," *IEEE Trans. Image Process.*, vol. 29, pp. 8055–8068, Jul. 2020.

[31] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing Human-level performance on ImageNet classification,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[32] S. Wanner, S. Meister, and B. Goldlücke, “Datasets and benchmarks for densely sampled 4D light fields,” *Vis., Model. Vis.*, vol. 13, pp. 225–226, Sep. 2013.

[33] M. Rerabek and T. Ebrahimi, “New light field image dataset,” in *Proc. 8th Int. Conf. Qual. Multimedia Experience*, 2016, pp. 1–2.

MARYAM HAMAD (Graduate Student Member, IEEE) received the B.E. degree in computer systems engineering from Palestine Technical University-Kadoorie, Tulkarm, Palestine, in 2018, covered by an excellence scholarship. She is currently working toward the Ph.D. degree with Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal. During her degree, she spent one semester as an Exchange Student with Middle East Technical University, Ankara, Türkiye in the ERASMUS+ Program, Türkiye. She completed her professional internship in information science and technology with the IAESTE Program as a Researcher with the Multimedia Signal Processing Group, Instituto de Telecomunicações, Portugal, where she is also a Researcher. Her research interests include immersive visual technologies, such as light field imaging, digital image processing, and computer vision. She is a member of the IEEE Women in Engineering Society, the IEEE Signal Processing Society, and the IEEE Young Professionals Group. She is a reviewer for the IEEE ACCESS, *Signal, Image and Video Processing*, and *IEEE Transactions on Image Processing Journals*.

PAULO NUNES (Member, IEEE) received the degree in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade de Lisboa, Lisbon, Portugal, in 1992, and the M.Sc. and Ph.D. degrees in electrical and computers engineering from IST, in 1996 and 2007, respectively. He is currently a Senior Researcher with Multimedia Signal Processing Group, Instituto de Telecomunicações, Portugal. He is also an Associate Professor with the Department of Information Science and Technology, Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon. He has coordinated and participated in various national and international (EU) funded projects and has acted as a Project Evaluator for the European Commission. His research interests include 2D/3D image and video processing and coding, namely light field image and video processing and coding. He is a reviewer for various ACM, EURASIP/Elsevier, IEEE, IET, SPIE, and Springer conferences and journals and as a member of the program and organizing committees of various international conferences. He has contributed more than 70 papers to international journals and conferences in these areas.

CAROLINE CONTI (Member, IEEE) received the B.Sc. degree in electrical engineering from Universidade de São Paulo, São Paulo, Brazil, in 2010, and the Ph.D. degree in information science and technology from Instituto Universitário de Lisboa (ISCTE- IUL), Lisbon, Portugal, in 2017. She is currently a Senior Researcher with Multimedia Signal Processing Group, Instituto de Telecomunicações, and an Assistant Professor with the Department of Information Science and Technology, Instituto Universitário de Lisboa. Her research interests include immersive visual technologies and image and video processing, including light field processing and coding. She has contributed more than 25 papers to international journals and conferences in these areas. She is an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING. She has been the Guest Editor of *Signal Processing: Image Communication* (Elsevier). She actively participates as a reviewer for various IEEE and EURASIP journals and conferences.

LUÍS DUCLA SOARES (Senior Member, IEEE) received the Licenciatura and Ph.D. degrees in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade de Lisboa, Lisbon, Portugal, in 1996 and 2004, respectively. He is currently a Senior Researcher with Multimedia Signal Processing Group, Instituto de Telecomunicações, Portugal. He is also an Associate Professor with the Department of Information Science and Technology, Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon. His research interests include coding and processing of visual information modalities, including light fields. He has contributed more than 70 papers to international journals and conferences in these areas. In addition, he has participated in the development of the MPEG-4 Visual standard, as well as in several national and international projects. He is a member of the editorial board of the EURASIP *Signal Processing* (Elsevier) Journal. He is a reviewer for several IEEE, IET, and EURASIP journals and conferences.