Dynamic Super-Rays for Efficient Light Field Video Processing

Matthieu Hog[1,2]
matthieu.hog@technicolor.com
Neus Sabater[1]
neus.sabater@technicolor.com
Christine Guillemot[2]
christine.guillemot@inria.fr

[1] Technicolor R&I
Rennes
France
[2] INRIA
Rennes
France

Abstract

One challenging problem with light field video editing is the dreadful volume of data to process. Image and video processing frequently rely on over-segmentation methods to reduce the computational burden of subsequent editing tasks. In this paper, we present the first approach for video light field over-segmentation called dynamic super-rays, which can be seen as temporally and angularly consistent superpixels. Our algorithm is memory efficient and fast. The results show timing close to editing time by leveraging GPU computational power.

1 Introduction

A light field is a description of the flow of light from and towards every direction in space. In most cases, it is captured by imaging a region of interest from varying viewpoints, yielding in a highly-redundant multiview scene representation. Because of this, a light field enables advanced processing such as depth estimation, refocusing, parallax shift, or super-resolution. The acquisition of such content is typically done with 3 classes of devices. Plenoptic cameras [18, 24] use a microlens array in front of a camera sensor to re-arrange by direction the light rays coming from the camera main lens. Camera arrays [40] are composed of a rig of cameras, often organized on a regular grid. Finally, camera gantries (e.g. Stanford gantry [33]) have a mechanical system to move a single camera along a plane, taking photos at regular intervals. Despite being aimed at different applications and having quite diverse spatial and angular resolutions, the content captured by these devices is often described using a representation called lumigraph [12], which describes all light rays as a set of views taken with a regular baseline on a single plane.

Currently, most of light field content available is static. This is perhaps due to the difficulties of capturing a dynamic light field for either of the aforementioned devices. Specifically, camera gantries are by design unable to record dynamic light fields, and the available plenoptic cameras are either limited to static acquisition (e.g. [19]) or cannot easily produce a lumigraph (e.g. type 2 plenoptic cameras [26]). Camera arrays require a synchronized camera acquisition system, which is a real technical challenge. However, latest advances in light field video acquisition systems [8, 31] show that it is possible to capture quite voluminous light fields in real time. The problem of efficiently processing the captured 5D light field content is then twofold. First, the amount of data to handle, already a problem for static light fields, reaches a critical point for video light fields. As a consequence, computational efficiency is a core aspect in many light fields processing tasks, such as editing. Secondly, to enable editing via user interaction on few key frames, algorithms must consistently propagate the edits angularly and temporally to the rest of the dynamic light field.

In this paper we present an over-segmentation scheme that exploits the light field redundancy in both the temporal and angular dimensions and leverages GPU computational power to enable easy and fast light field editing from a single reference view and frame. The proposed over-segmentation approach generalizes the concept of *super-rays*, introduced in [14] for static light fields, to *dynamic super-rays* for video light fields. Super-rays, seen as the light field analog of superpixels [2], are a grouping of rays captured in different views of the light field that are similar in appearance and coming from the same scene area.

Several constraints are taken into consideration in the design of the proposed method. First, the approach is parallelizable to take advantage of a GPU implementation. Second, it processes different frames sequentially and on-the-fly, so the memory footprint is reasonable, in contrast to methods operating on sliding windows. This is mandatory as the amount of data per frame is much greater than a conventional 2D video. Finally, super-rays are consistent across views *and* across frames in the temporal dimension.

Specifically, our contributions are: 1. An end to end approach to create an angularly and temporally consistent light field over-segmentation that we call dynamic super-rays 2. A new update term for creating and deleting superpixels and super-rays specially tailored for dynamic content 3. A new strategy for creating a temporal over-segmentation that is consistent with the scene movement.

2 Related Work

We focus on the two areas of the literature that are related to the proposed method, namely image and video over-segmentation and light field editing.

Image and Video Over-Segmentation: Since the introduction of the term superpixel [28], many approaches have been proposed for still image over-segmentation. Because superpixels are often used as a pre-processing step to speed up other algorithms, it is unsurprising to see that superpixel methods yielding low run-time are more popular than historical approaches [16, 20, 36]. Among them, we can mention SEEDS [34]. Starting from a regular lattice, it iteratively reassigns pixel blocks to superpixels based on a color consistency term, in a coarse to fine multi-scale fashion. Simple Linear Iterative Clustering (SLIC) [2] adapts the Loyd's algorithm for k-means clustering of image pixels in terms of spatial and color distance in the Lab space. In order to speed up the algorithm, the clusters are contained in a small local window. Arguably, the strongest advantage of SLIC is its ability to leverage massively parallel computation. Because each assignment and update step can be carried out efficiently on the GPU, to the best of our knowledge, SLIC is the only approach providing real-time performances [27]. Many variations of this algorithm have been proposed [1, 17, 37], but all loosing somehow this computational advantage.

The problem of over-segmentation for multiview is a rather unexplored topic. Current approaches focus on generating superpixels on each view and group them afterwards to compute a consistent segmentation, rather than an over-segmentation [6, 21]. On the contrary, video over-segmentation is a more researched topic. In [2], volumetric superpixels are created by treating the temporal and spatial dimensions in the same way. Whereas in [10], superpixels are computed separately on each frame and later merged to extract object segments. In [7] a pre-computed dense optical flow is used along with a Gaussian process to update the labeling from a frame to another. The superpixels deletion or creation is done by inference using the same Gaussian process. In [35], the framework proposed in [34] is extended by performing the re-assignments as a new frame arrives. In [29], the most related work to ours, dynamic SLIC superpixels are computed in a sliding window of 20 frames. A dense flow is used to propagate the assignment from a frame to another and several SLIC iterations are run. The centroid color is shared between corresponding superpixels on several frames of the sliding window. The superpixel update criteria is solely based on the superpixel size evolution. Unfortunately, none of the aforementioned approaches are readily applicable for video light field over-segmentation for different reasons. Loading a large amount of the video frames as in [2, 29] is prohibitive in the case of a light field. Performing a late merge of frames superpixels as in [10] does not provide any temporal consistency. Methods taking the assumption that the temporal dimension is densely sampled, as in [7], will fail when large objects motion are involved, as in our case. Finally, all the approaches relying on a stack of granular operations as in [35], are ill-suited for a GPU implementation.

Light field editing: Because segmentation is the first step of many editing algorithms, it is natural to see most light field editing papers being centered around this topic. A level set method is proposed in [5] to extract objects layers in the scene. In [38] and [22], the authors use user scribbles on a reference view to learn a joint color and depth object model that is used to infer a label for each ray of the light field. These label assignments are further regularized inside and between views. In these two approaches, because the regularization does not scale well with the size of the input light field, the running times are rather high. To solve this issue, in [13] rays coming from the same scene point are represented in a single node on a ray-based graph structure. This reduces significantly the size of the graph and scales well with the number of input images. However, the quality and the run-time depends on the quality of the dense depth estimation on all views, which can be quite expensive to compute. Furthermore, as for conventional 2D images, the approach does not scale with the spatial resolution of the light field.

Recent works focused on providing a light field over-segmentation to overcome this problem. In [41], a depth estimation is used to propagate and refine an initial SLIC over-segmentation on a reference view to all the views of the light field. In [14], a method to compute angularly consistent superpixels, named *super-rays* is proposed. The approach does not require a dense depth estimation and focuses on speed and parallelism but still provides satisfactory segmentations. This last algorithm has shown to be a good trade-off between speed and accuracy for still light fields, so we consider it as a starting point for our over-segmentation method for light field videos. We summarize it in Sec. 3.

3 Static super-rays summary

In this section we briefly describe and give the main notations of the super-rays algorithm for static light fields. Further details can be found in [14].

Notations: Let r be a ray of the light field LF, and $(\mathbf{s},\mathbf{x})$ its coordinates using the two plane parametrization [12], where $\mathbf{s} = (s,t)$ and $\mathbf{x} = (x,y)$ are the angular and spatial coordinates respectively. Besides, each ray has an associated *CIELab* color value Lab_r. Let

$\mathbf{x}' := \mathcal{P}^d_{\mathbf{s}'}(\mathbf{x})$ be the spatial pixel position in view $\mathbf{s}'$ imaging the same scene point, at a distance d, as $\mathbf{x}$ in view $\mathbf{s}$. This is, $r = (\mathbf{s}, \mathbf{x})$ and $r' = (\mathbf{s}', \mathbf{x}')$ are corresponding rays imaging the same scene point in different views. Given a light field, *super-rays* are all the perceptually similar rays corresponding to the same scene area. That is, the mapping $A\colon LF \subset \mathbb{Z}^4 \to \mathbb{Z}$, such that each ray r is assigned a super-ray label c is computed. Let SR_c be the set of rays r such that $A(r) = c$. The super-ray computation is inspired by SLIC [2] and has the same main steps. One major difference is that each super-ray SR_c is characterized by a centroid ray r_c with angular coordinates corresponding to the reference view $\mathbf{s}_c$ and a depth d_c associated to it.

Initialization: The spatial positions $\mathbf{x}_c$ of the centroid rays are initialized on a regular grid of step S in the reference view. The corresponding *CIELab* color values on such positions are the initial color values of the centroid rays Lab_{r_c}, and the depth d_c of each centroid ray r_c is estimated via block-matching.

Assignment step: At each iteration, each ray $r = (\mathbf{s}, \mathbf{x})$ of the light field is assigned a super-ray label $A(r)$. First, the depth estimated in the previous step is used to compute $r'_c = (\mathbf{s}', \mathcal{P}^{d_c}_{\mathbf{s}'}(\mathbf{x}_c))$, the corresponding rays of r_c. Then, each ray in a neighborhood $N_S(r'_c)$ of size S around r'_c is assigned to the super-ray SR_c if it minimizes the color and spatial distances:

$$A(r) = \arg\min_c \left(\Delta_{Lab}(r, r_c) + \lambda\, \Delta_{xy}(r, r'_c) \right), \tag{1}$$

where $\Delta_{Lab}(r, r_c) = ||Lab_r - Lab_{r_c}||^2$, $\Delta_{xy}(r, r'_c) = ||\mathbf{x} - \mathcal{P}^{d_c}_{\mathbf{s}'}(\mathbf{x}_c)||^2$ and λ is the parameter weighting the color and spatial distances.

Update step: Following the assignment step, the spatial coordinates of the ray centroid and its corresponding *Lab* values are updated. The new color value of r_c is the average of the color values of all rays in SR_c and the new spatial coordinates are the average coordinates of all light rays $r = (\mathbf{s}, \mathbf{x})$ in SR_c projected on the reference view using the depth d_c:

$$\mathbf{x}_c = \frac{1}{|SR_c|} \sum_{r \in SR_c} \mathcal{P}^{d_c}_{\mathbf{s}_c}(\mathbf{x}). \tag{2}$$

Note that the angular coordinates $\mathbf{s}_c$ of the centroid rays do not change along the iterations, while the spatial coordinates are updated.

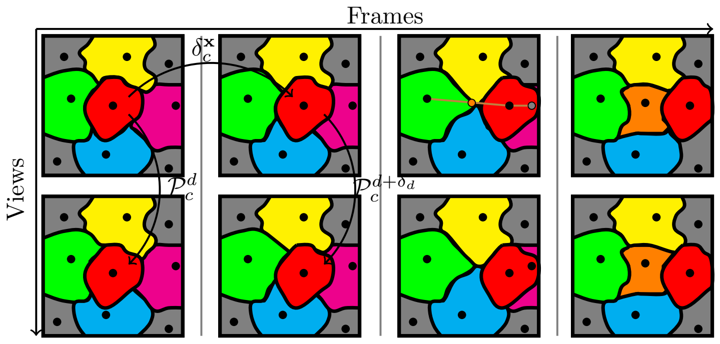
Iterations: As in SLIC, the two previous steps are repeated until the centroids are stable. This happens within 10 to 15 iterations. A cleanup step is optionally run to reassign labels to disconnected rays.

4 Dynamic super-rays

The proposed approach is inspired from techniques proposed to generate temporally consistent superpixels [7, 29], that can be decomposed into three main steps: 1. initialize the current frame segmentation by temporally propagating the segmentation of previous frames 2. adapt the segmentation to changes in geometry 3. create and delete segments to take into account occlusions and objects entering or leaving the scene. Our algorithm is summarized in Alg. 1 in the supplementary material and illustrated in Fig. 1.

4.1 Sparse temporal propagation

Computing a dense and accurate optical flow for light fields can be a quite tedious task, especially when memory and time requirements are taken into account. Moreover, because



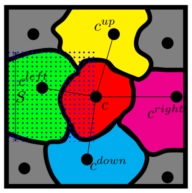


Figure 1: Proposed dynamic super-ray scheme. Each super-ray is represented by a solid color and its centroid by a black dot. (a): Illustration of our algorithm in a simple case. The red foreground super-ray is tracked over the consecutive frames of a 2×1 light field. Other super-rays do not move since the background is static. The depth d is used to enforce angular consistency from a view to another, while the scene flow $(\delta^{\mathbf{x}}, \delta^d)$ guarantees temporal consistency. On the third frame, the moving red super-ray becomes too close of the pink super-ray and too far from the green one, triggering the creation of the orange super-ray and the deletion of the pink one on the next frame. (b): Super-ray neighborhood. The search area for the left neighbor c^{left} of the red super-ray c is represented by the blue dotting, and the final neighborhood connections of c by the black lines.

super-rays embed a depth information per segment, the problem we aim to resolve is a scene flow estimation problem. That is, we aim to find the displacements of 3D points in the scene rather than pixels shifts in the image plane. Fortunately, in the case of super-rays, the scene flow estimation needs to be estimated only for centroids and not for all rays of the light field. One way of computing the scene flow $(\delta_c^{\mathbf{x}}, \delta_c^d)$ is to compute exhaustively a cost function for each possible motion vector (as in Eq. 1 in the supplementary material). However, this 3-dimensional cost function being quite expensive to minimize, we have split the problem into optical flow and depth estimation, like other methods for light field scene flow estimation in the literature [4, 32].

Now, in state of the art optical flow estimation methods Deep Flow [39] stands out for its performance in terms of quality and run-time [3]. Deep flow first searches for sparse matches using Deep Match [30] between downsampled versions of the input frames, and then the matches are densified by regularizing a selected set of sparse matches. Deep Match has many properties which are interesting for our problem. It is robust and efficient since it can be implemented on GPU [15] and the matches are searched in a limited local window. Thus, we solve the sparse flow estimation using deep matches. In contrast to Deep Flow, we do not seek to obtain a dense and precise optical flow, but rather a robust and fast sparse flow for each centroid.

We compute the set of deep matches [30] from two downsampled frames f and $f+1$. Then, the estimated flow $\delta_m^{\mathbf{x}} = \mathbf{x}_m^{f+1} - \mathbf{x}_m^f$, using the deep matches in the full resolution coordinate system, is used to compute the flow of each centroid $\delta_c^{\mathbf{x}}$ using a simple and fast bilinear interpolation. Precisely, $\delta_c^{\mathbf{x}}$ is the distance-weighted average flow $\delta_m^{\mathbf{x}}$ of its 4 nearest matches. Using the notation above, the depth is updated using the same strategy as in [14]:

$$\delta_c^d = \arg\min_{\delta \in D} \left\{ \min_{o \in \Omega} \sum_{\mathbf{s}'} o(\mathbf{s}')\ \Delta_{RGB}^{B}(r_c^{f+1}, r_c'^{f+1}) \right\}, \tag{3}$$

where D is the small range of potential depth movements and Ω is a family of spatio-angular patches.

4.2 Centroid creation and deletion

Because of object movements in the scene, the super-ray topology can change in time. For instance, parts of the super-rays can be occluded or disoccluded, or completely appear or disappear due to objects entering or leaving the scene. For this reason, creating and deleting super-rays might be necessary. While the superpixel size or color consistency has been used to determine the creation or deletion in other research works, we propose to leverage the depth information associated to the super-ray to detect occlusions and disocclusions.

In particular, a new super-ray is candidate to be created at the midpoint of two super-rays when their centroid distance exceeds a given threshold $S * \tau$. Conversely, a super-ray will be a candidate to be deleted if two super-rays are too close from each other, i.e. their centroid distance is lower than a threshold S/τ. In particular, the occluded super-ray (with the smallest disparity or biggest depth) is the candidate for deletion. For the sake of efficiency, and to avoid duplicates, we search the candidate centroids to be deleted or created in a 4-nearest neighborhood, computed as illustrated in Fig. 1(b). Specifically, the approximate neighborhood of a centroid c is defined as $\mathcal{N}(c) = \{c^{left}, c^{right}, c^{up}, c^{down}\}$ where

$$c^{left} = \arg\min_{\hat{c}} \left\{ |y_{\hat{c}} - y_c| \text{ s.t. } x_{\hat{c}} < x_c,\ |y_{\hat{c}} - y_c| < S \right\}, \tag{4}$$

and similarly for the other neighbor centroids.

Now, in order to maintain the number of super-rays constant, we create the same number of super-rays we delete. If the number of candidates for deletion is smaller (resp. bigger) than the number of candidates for creation, only the centroids with the biggest (resp. smallest) centroid distance are created (resp. deleted). Finally, because objects can move inside or outside of the reference view, the super-rays near the image borders are treated as follows. New super-rays are created in the reference view between the image borders and the closest centroids. For instance, if a centroid c does not have a neighbor c^{right}, a new centroid will be $(\frac{x_c+M}{2}, y_c)$, M being the reference view width. Super-rays that leave the reference view image plane are automatically deleted.

Note that the centroid neighborhood can be used for further processing, as it is a convenient way of representing the super-rays structure.

4.3 New frame over-segmentation

In the new frame, after defining the set of centroids in the reference view, all the rays of the light field are assigned to a centroid. Similarly to super-rays, the assignment is done using Eq. 1 in an iterative process with color and position centroid updates. While the centroid color is updated with the same color average strategy as super-rays, the centroid position update changes for dynamic super-rays. So Eq. 2 becomes

$$\mathbf{x}_c^{f+1} = \left(\frac{p}{|SR_c^{f+1}|} \sum_{r \in SR_c^{f+1}} \mathcal{P}_{\mathbf{s}_c}^{d_c}(\mathbf{x}_r^f) \right) + (1-p)(\mathbf{x}_c^f + \delta_c^{\mathbf{x}}), \tag{5}$$

where p is a parameter controlling how much the super-rays are allowed to move from their theoretical position. When $p = 1$, this step corresponds to the same SLIC iteration as in Eq. 2, and when $p = 0$, the super-ray centroids are not updated at all, providing the best temporal consistency. Newly created centroids (as described in Sec. 4.2) are updated using $p = 1$, allowing them to adapt to scene changes.

In [29], 5 SLIC iterations are run, where the centroids are allowed to move freely. As a consequence, superpixels of static objects tend to move since they are affected by the creation, deletion and movements of nearby superpixels. On the contrary, our dynamic super-rays movement is congruous with the objects movement in the scene, providing a more consistent temporal over-segmentation.

5 Experiments

As an effort to quantitatively assess the efficiency of the proposed approach, we use the *Monka* dataset proposed in [25], composed of ground truth images, disparity, optical flow and object labels for pairs of cameras. This can arguably be considered as a small light field, but to the best of our knowledge such dataset does not exists for more than 2 views. We use the standard superpixel quality metrics [23] (ASA ,BR, UE) along with a Temporal Consistency (TC) measure, which is the analog of the angular consistency in [14] in the temporal dimension.

We compare our approach with statics super-rays [14] computed on each frame and put into correspondence using the ground truth optical flow such that the TC is maximized. In average, we find the dynamic super-ray to be slightly better for classical over-segmentation metrics but not significantly so. This can be explained by the fact that at each new frame, the previous over-segmentation is re-used, meaning that each frame iteration is refining the previous segmentation. However, we see that super-rays computed separately give worst temporal constancy, despite being advantaged by the fact that super-pixel matching is perfect. Detailed results for all the sequences can be found in the supplementary material.

Currently, two datasets for video light fields captured with camera arrays are available: 1. The Fraunhofer dataset [8], with full-HD sequences of 9 to 16 views and between 150 and 400 frames; 2. the Technicolor dataset [31] with $2K$ sequences of 16 pseudo-rectified views and between 300 and 390 frames. The cameras baseline is quite important, especially for the second dataset.

We use a fixed set of hyper parameters, that we did not fine tune for each sequence. The values are described in the supplementary material.

Fig. 2 shows the output of our algorithm for a small area of the dataset *Birthday* [31]. For the sake of visualization, results are only shown for two views, the reference view $\mathbf{s}_c = (1,1)$ and another view $\mathbf{s} = (1,0)$, and three non-consecutive frames $f = 95, 100, 105$. For each view, we show the input image with the optical flow only on the reference view (top left), the color-coded assignment (top right), the super-ray average color (bottom left) and finally the super-ray contours (bottom right).

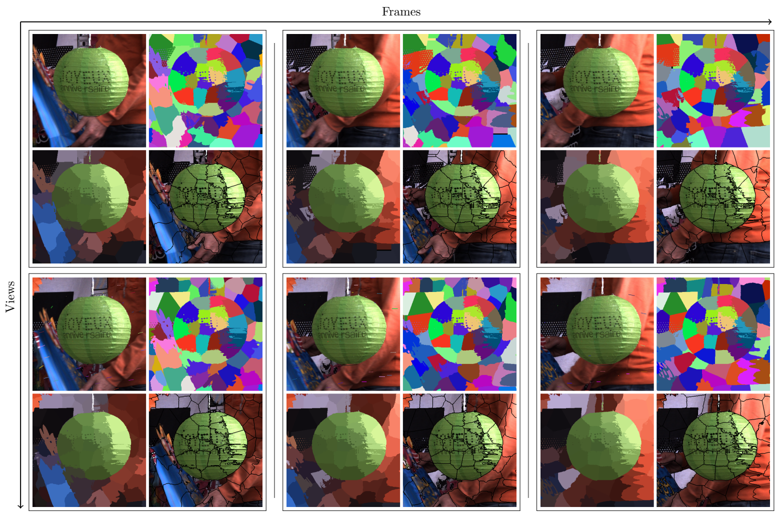


Figure 2: Video super-rays for 3 frames and 2 views of the dataset *Birthday* [31].

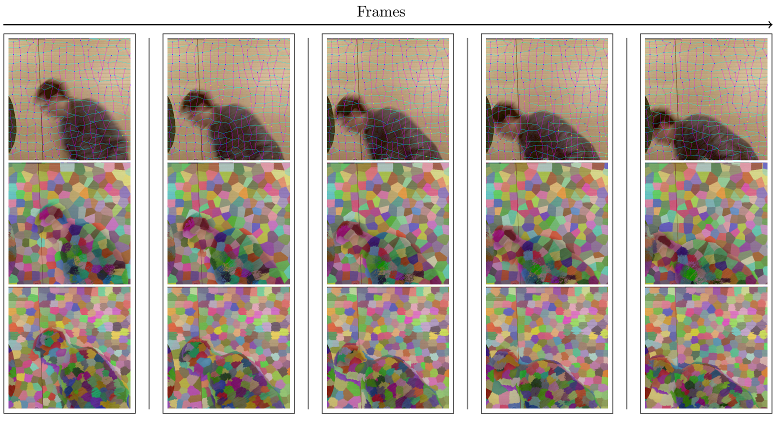


Figure 3: Over-segmentation comparison with [29] on the reference view over 5 frames. Our dynamic super-rays (second row) are consistent with the scene movement while the superpixels in [29] (third row) move in static regions.

Please note that it is hard to evaluate our over-segmentation results on paper due to the reduced number of frames or views we can illustrate compared with the full light-field videos. We strongly encourage the reader to visualize all our results on our web-page[1].

The videos show concatenated views with usual visualization methods, namely, the super-ray average color, the color-coded super-ray labels, and the super-ray contours (as in Fig. 2). We also visualize the value of the flow for each centroid, as well as the coarse depth of each super-ray (each ray has assigned its super-ray depth) and finally, the centroids

[1] Project website : https://www.irisa.fr/temics/demos/DynamicSuperrays/index.html. Alternatively the videos can be viewed on YouTube : https://www.youtube.com/channel/UCHFkXPUSiV3UFxlABRmQkNA/videos

neighborhood structure (as in Fig. 3).

We compare our method with the algorithm in [29] which is the state of the art for temporal consistent superpixels on videos. Note that in this experiment we focus on the temporal aspect since it has already been shown [14] that computing superpixels on each of the views separately does not guarantee angular consistency. Fig. 3 shows this comparison on five frames, $f = 260, 262, 263, 264, 265$, of the central view. In particular, on the top row, we show the neighborhood structure described in Sec. 4.2. Each centroid appears as a blue dot, and horizontal and vertical neighborhoods are illustrated with cyan and magenta edges respectively. Centroids of deleted super-rays are represented in red, while new super-rays are represented in yellow. The second and third rows correspond to our results and the results of [29] respectively. The input view and the color segments are blended on theses images. We observe that the update step in [29] allows the superpixels on the static background to move freely. On the contrary, our super-rays are not moving so the scene movement is consistent with the super-rays movement. We believe this is a major benefit if dynamic super-rays are to be used in further editing tasks. We invite the reader to view the videos in the supplementary material or on our website, where temporal consistency is more visible.

Besides the qualitative comparison with [29] we have also observed considerable differences in terms of computational complexity. Depending on the datasets, the algorithm in [29] takes several hours and up to one day (using the original implementation) to run for all the frames of a single view video. In our case, the biggest advantage is the GPU friendliness. Indeed, the SLIC-based iterations, the deep flow computation and the super-ray creation and deletion, are highly parallelizable. On the same machine (equipped with a *Nvidia GTX 1080* GPU hosted by an *Intel Xeon E5-2630* CPU) our current *Python/PyOpencl* implementation gives an average running time for each iteration of 0.157s and 0.059 to 0.083s (depending on the input size), respectively on [31] and [8]. Further improvements are to be expected by a more optimized implementation.

Dynamic super-rays with the neighborhood structure presented in Sec. 4.3 offer a useful representation of the scene captured by the light field videos. Temporal super-rays can be seen as a powerful tool for efficient light-field video editing in which the edits in one reference view of the light-field can be easily propagated to other frames and views. For example, in [14] it is presented how to use super-rays to generate intermediate views to correct the angular aliasing caused by the poor angular sampling of sparse light fields. Similarly, dynamic super-rays can be used for temporal image interpolation without *flickering* caused by an inconsistent interpolation. Other examples of temporal super-rays applications include light-field video compression (e.g. adapting the approach in [9]), or light field color transfer (e.g. using the algorithm in [11]).

Limitations: We have observed that our approach has some limitations, in particular, when the depth or the flow estimation becomes erroneous, the super-ray consistency is not guaranteed from one view to another. Such failure case is visible in the dataset *Newspeaker* [8], where a very uniform green background challenges both the depth and flow estimations. When the depth is inconsistent, the centroids are wrongly projected, leading to large areas with no nearby centroid for the rays to be assigned to. The other failure case involves small moving objects, because of our sparse flow computation strategy, the optical flow for small object can be wrongly evaluated to the flow value of its surrounding. This is visible on the dataset *Train* [31], where centroids struggle to follow the train wagons.

In conclusion, even if depth and flow estimation are mature research topics we have observed that challenging datasets may still produce inaccurate estimates. In particular, the images in the two datasets suffer heavily from motion blur, noise and over and under exposition. Furthermore, the dataset in [8] has some large texture-less areas. However, loosing consistency in flat areas is not critical. Indeed, if the zone to edit is totally uniform, the editing become trivial (e.g. using a simple color threshold), dismissing the needs for super-rays in the first place.

6 Conclusion

We presented an approach to generate angularly and temporally consistent light field video over-segmentation. Our algorithm design enabless a GPU implementation, allowing computational performances that are required to cope with the high volume of data. To the best of our knowledge, this is the first approach to deal with the problem of video light field editing.

References

[1] Radhakrishna Achanta and Sabine Süsstrunk. Superpixels and Polygons using Simple Non-Iterative Clustering. In *CVPR*, 2017.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011.

[4] T. Basha, S. Avidan, A. Hornung, and W. Matusik. Structure and motion from scene registration. In *CVPR*, pages 1426–1433, 2012.

[5] Jesse Berent and Pier Luigi Dragotti. Unsupervised extraction of coherent regions for image based rendering. In *BMVC*, pages 1–10, 2007.

[6] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Automatic object segmentation from calibrated images. In *Conference for Visual Media Production (CVMP)*, pages 126–137, 2011.

[7] Jason Chang, Donglai Wei, and John W. Fisher, III. A video representation using temporal superpixels. In *CVPR*, June 2013.

[8] Łukasz Dąbała, Matthias Ziegler, Piotr Didyk, Frederik Zilly, Joachim Keinert, Karol Myszkowski, H-P Seidel, Przemyslaw Rokita, and Tobias Ritschel. Efficient Multi-image Correspondences for On-line Light Field Video Processing. *Computer Graphics Forum*, 2016.

[9] Giulia Fracastoro, Francesco Verdoja, Marco Grangetto, and Enrico Magli. Superpixel-driven graph transform for image compression. In *ICIP*, pages 2631–2635, 2015.

[10] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. Video segmentation with superpixels. In *ACCV*, pages 760–774, 2013.

[11] Rémi Giraud, Vinh-Thong Ta, and Nicolas Papadakis. Superpixel-based color transfer. In *ICIP*, 2017.

[12] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[13] Matthieu Hog, Neus Sabater, and Christine Guillemot. Light Field Segmentation Using a Ray-Based Graph Structure. In *ECCV*, page 16, October 2016.

[14] Matthieu Hog, Neus Sabater, and Christine Guillemot. Super-rays for efficient light field processing. *J-STSP*, PP(99):1–1, 2017.

[15] Zaid Harchaoui Cordelia Schmid Jerome Revaud, Philippe Weinzaepfel. Deep-Matching: Deep Convolutional Matching. `http://lear.inrialpes.fr/src/deepmatching/`. [Online; accessed 29-mar-2018].

[16] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *PAMI*, 31(12):2290–2297, 2009.

[17] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *CVPR*, pages 1356–1363, 2015.

[18] Andrew Lumsdaine and Todor Georgiev. The focused plenoptic camera. In *ICCP*, pages 1–8, 2009.

[19] Lytro. Lytro website. `https://lytro.co`. [Online; accessed 29-mar-2018].

[20] Fernand Meyer and Petros Maragos. Multiscale morphological segmentations based on watershed, flooding, and eikonal pde. In *International Conference on Scale-Space Theories in Computer Vision*, pages 351–362, 1999.

[21] Branislav Mičušík and Jana Košecká. Multi-view superpixel stereo in urban environments. *IJVC*, 89(1):106–119, 2010.

[22] Hajime Mihara, Takuya Funatomi, Kenichiro Tanaka, Hiroyuki Kubo, Yasuhiro Mukaigawa, and Hajime Nagahara. 4d light field segmentation with spatial and angular consistencies. In *ICCP*, pages 1–8, 2016.

[23] Peer Neubert and Peter Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, pages 1–12, 2012.

[24] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report*, 2(11):1–11, 2005.

[25] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`. arXiv:1512.02134.

[26] Raytrix. Raytrix website. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/. [Online; accessed 29-mar-2018].

[27] Carl Yuheng Ren, Victor Adrian Prisacariu, and Ian D Reid. gSLICr: SLIC superpixels at over 250Hz. *ArXiv e-prints*, September 2015.

[28] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *ICCV*, page 10, 2003.

[29] Matthias Reso, Jorn Jachalsky, Bodo Rosenhahn, and Jorn Ostermann. Temporally consistent superpixels. In *ICCV*, pages 385–392, 2013.

[30] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *IJCV*, 120(3):300–323, 2016.

[31] N. Sabater, G. Boisson, B. Vandame, P. Kerbiriou, F. Babon, M. Hog, R. Gendrot, T. Langlois, O. Bureller, A. Schubert, and V. Allié. Dataset and pipeline for multi-view light-field video. In *CVPR Workshop*, pages 1743–1753, 2017.

[32] Pratul P Srinivasan, Michael W Tao, Ren Ng, and Ravi Ramamoorthi. Oriented light-field windows for scene flow. In *ICCV*, pages 3496–3504, 2015.

[33] Andrew Adams Vaibhav Vaish. The (new) Stanford fight field archive website. lightfield.stanford.edu/lfs.html. [Online; accessed 29-mar-2018].

[34] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *ECCV*, pages 13–26, 2012.

[35] Michael Van den Bergh, Gemma Roig, Xavier Boix, Santiago Manen, and Luc Van Gool. Online video seeds for temporal window objectness. In *ICCV*, pages 377–384, 2013.

[36] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, pages 705–718, 2008.

[37] Peng Wang, Gang Zeng, Rui Gan, Jingdong Wang, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. *IJCV*, 103(1):1–21, 2013.

[38] S. Wanner, C. Straehle, and B. Goldluecke. Globally consistent multi-label assignment on the ray space of 4d light fields. In *CVPR*, pages 1011–1018, 2013.

[39] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deep-Flow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[40] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Marc Levoy, and Mark Horowitz. High-speed videography using a dense camera array. In *CVPR*, volume 2, pages II–294, 2004.

[41] Hao Zhu, Qi Zhang, and Qing Wang. 4d light field superpixel and segmentation. In *CVPR*, 2017.