DENOISING OF 3D POINT CLOUDS CONSTRUCTED FROM LIGHT FIELDS

Christian Galea and Christine Guillemot

INRIA, Rennes, France
christian.galea@inria.fr, christine.guillemot@inria.fr

ABSTRACT

Light fields are 4D signals capturing rich information from a scene. The availability of multiple views enables scene depth estimation, that can be used to generate 3D point clouds. The constructed 3D point clouds, however, generally contain distortions and artefacts primarily caused by inaccuracies in the depth maps. This paper describes a method for noise removal in 3D point clouds constructed from light fields. While existing methods discard outliers, the proposed approach instead attempts to correct the positions of points, and thus reduce noise without removing any points, by exploiting the consistency among views in a light-field. The proposed 3D point cloud construction and denoising method exploits uncertainty measures on depth values. We also investigate the possible use of the corrected point cloud to improve the quality of the depth maps estimated from the light field.

***Index Terms*—** Light fields, point clouds, depth estimation, denoising

1. INTRODUCTION

Light field imaging has recently captivated the interest of both research and industry due to numerous applications such as view synthesis, augmented reality, and post-production. A light field can be seen as capturing an array of viewpoints leading to a 4D representation of the imaged scene, spanning spatial and angular dimensions [1, 2]. Light fields can be captured by rigs of cameras that can be bulky and not easy to use, or by plenoptic cameras. The captured multiple viewpoints enable the computation of scene depth that can then be used to construct a 3D model, e.g. in the form of a 3D point cloud, thanks to the camera parameters. However, 3D point cloud construction methods are prone to outliers caused by inaccuracies in the depth maps arising from deficiencies in the depth estimation method itself, from matching ambiguities, or from factors such as lens distortion and sensor noise [3].

Methods proposed in the literature attempt to solve the problem by either refining the depth maps, or the point clouds themselves. Notable methods include the anisotropic denoising method in [4] which uses L_0 minimisation to recover sparse sharp features, assuming that the surface of common objects is piece-wise smooth almost everywhere except for a small number of sharp features. Outliers are removed in [5] by employing a statistical analysis of the points' geometrical properties, while the approach in [3] employs geometric and photometric consistency measures. The method in [6] removes outliers using Robust Principal Component Analysis (RPCA) and denoises the resulting point cloud with a bilateral filter [7, 8]. However, these methods are usually designed in the context of improving a particular task in a 3D reconstruction pipeline, such as surface reconstruction, or make explicit assumptions on the input scene.

This paper considers the problem of denoising 3D point clouds derived from depth maps estimated from light fields. The method does not remove any points or perform explicit smoothing as done in existing methods. Instead, the proposed method attempts to *correct* the point positions by leveraging the consistency among views and by considering both geometric *and* photometric information. To the best of the authors' knowledge, the proposed approach: (i) is the first method that considers point clouds of whole scenes (rather than single objects only) generated from light field cameras, and (ii) attempts to improve the position of outlier points, rather than simply detect and remove them as done by existing methods. It is also shown that the proposed 3D point cloud denoising method can be used in turn to improve the quality of depth maps by re-projecting the ameliorated points into the set of 2D planes.

The rest of this paper presents a description of the proposed methods in Sections 2 and 3, followed by their evaluation in Section 4. Concluding remarks and proposals for future work are finally given in Section 5.

2. 3D POINT CLOUD CONSTRUCTION USING DEPTH UNCERTAINTY

The algorithm starts by first estimating disparity between views using the ProbFlow optical flow estimation algorithm [9], which also generates uncertainty measures on the estimated disparity values. The uncertainty estimation is exploited to combine the disparity maps obtained for each view such that a value is extracted from the disparity map having the highest estimated reliability.

This work was supported by the EU H2020 Research and Innovation Programme under grant agreement No 694122 (ERC advanced grant CLIM).

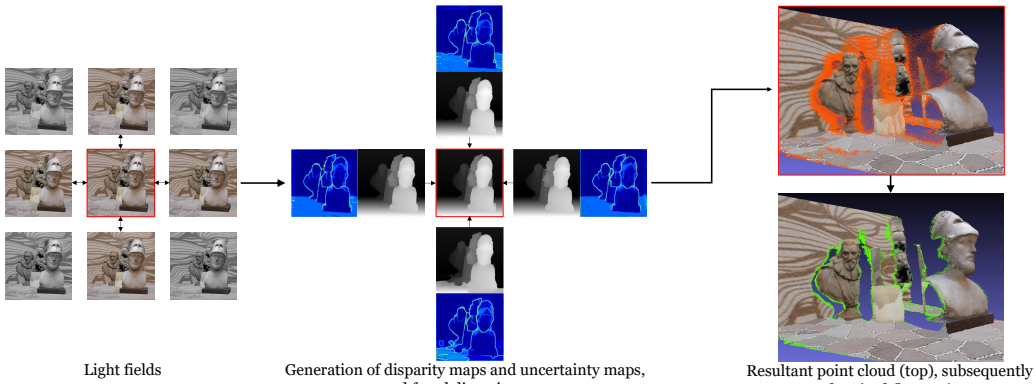

Fig. 1: Framework: for each image, disparity maps are extracted from four views in the horizontal and vertical directions along with uncertainty measures. The disparity maps are fused into a single map using uncertainty. The fused maps from all views are then projected into 3D to yield the 3D point cloud, which is denoised. Detected outliers are represented in orange, which are shown in green after the proposed method is applied.

Specifically, the disparity map for a given view can be estimated from its 4 vertical and horizontal neighbours (see Figure 1), yielding four candidate maps. For each pixel with coordinates $\mathbf{c} = (x, y)$, its final disparity value is taken from the ProbFlow-generated map having the most reliable estimates in a patch of size $p_r \times p_c$ centred on $\mathbf{c}$. From this disparity estimate $s_{\mathbf{c}}$, the depth value $Z_{\mathbf{c}}$ may be computed as follows:

$$Z_{\mathbf{c}} = \frac{b \cdot f \cdot fd \cdot max(H, W)}{s_{\mathbf{c}} \cdot fd \cdot ss + b \cdot f \cdot max(H, W)} \quad (1)$$

where W and H are the width and height of the captured image, respectively, b is the baseline (in millimetres, mm), f is the focal length (mm), fd is the focus distance (mm), and ss is the sensor size [10]. The $X_{\mathbf{c}}$- and $Y_{\mathbf{c}}$-coordinates may then be obtained as follows:

$$X_{\mathbf{c}} = \frac{(\frac{x}{W} - \frac{1}{2}) \cdot ss \cdot Z_{\mathbf{c}}}{f}, Y_{\mathbf{c}} = \frac{(\frac{y}{H} - \frac{1}{2}) \cdot ss \cdot Z_{\mathbf{c}}}{f} \quad (2)$$

3. 3D POINT-CLOUD DENOISING

The fused disparity maps still contain several discrepancies with respect to ground truth maps as a consequence of inaccuracies in both the original disparity maps and the uncertainty estimation. For instance, edges around objects tend to be blurred, corresponding to a smooth transition from foreground to background. This manifests in the point cloud as sparse points which do not belong to any object in the 3D space. To denoise the point clouds, the proposed method exploits the redundancy due to the fact that several areas of a scene are captured by multiple cameras. These regions which are assigned similar disparity values in the majority of views yield areas of high point densities in the 3D space that can be termed as *inliers*. On the other hand, any incorrect values are likely to be contained within a few disparity maps only, and are represented by points which lie in sparsely-populated areas that appear to float in free space, called *outliers* (see the orange points in Figure 1). While outliers could be removed, as done by most works in literature, the re-projection of the point cloud into 2D image planes would yield missing pixels. Instead, the proposed method attempts to improve the position of points deemed to be outliers, by comparing their geometric and photometric properties with those of inlier points.

3.1. Outlier detection

We first need to classify each point as being either an outlier or an inlier. This is done by exploiting information on local densities of points, utilising the Statistical Outlier Removal (SOR) filter proposed in [5] to exploit the differences in point densities exhibited by inliers and outliers. While this method was designed to remove outliers using statistical properties of the distances among points, in the proposed approach it is simply used to *detect* the outliers which need to be corrected. Formally, a point i is classed as an outlier if the distance d_i to its K nearest neighbours satisfies the following condition:

$$d_i \geq \mu(\vec{d_K}) + m \cdot \sigma(\vec{d_K}) \quad (3)$$

where $\vec{d_K}$ is a vector containing the average distance of each point to its K nearest neighbours, $\mu(\vec{x})$ and $\sigma(\vec{x})$ are functions computing the average and standard deviation, respectively, of the vector $\vec{x}$, and m is the standard deviation multiplier.

3.2. Volume segmentation

Each 3D point must lie on specific rays emanating from the camera viewpoints (i.e. from the pixels in the image planes of the cameras). The position on this line is given by the depth value of the 3D point. Hence, the true position of the outlier is constrained to lie on this line, thereby significantly reducing

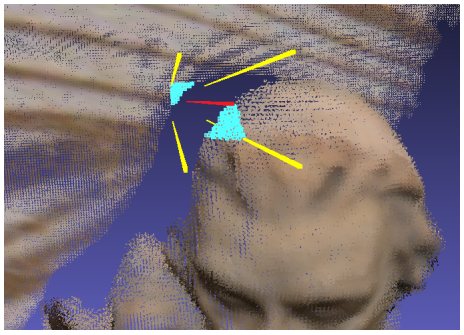

Fig. 2: Outliers reside on the red ray, and are located in a frustum bounded by the yellow rays. Two clusters of inliers are found within this frustum, depicted in blue.

the number of possibilities in the 3D space. The parametric equation $L(t)$ of a line in 3D may be given by:

$$L(t) = \vec{R} + \vec{V}t \tag{4}$$

where $\vec{R} = \vec{p_1}$ is a point on the line, $\vec{V} = \vec{p_2} - \vec{p_1}$ is a vector parallel to the line containing the points $\vec{p_1}$ and $\vec{p_2}$ which correspond to the end-points, and t is a scalar parameter representing the position on the line [11]. Another observation is that a point is likely to have a similar colour to neighbouring points. Lastly, points should also be relatively close in terms of their geometric distance to other visually similar points.

Given the above observations, the proposed method first determines the ray on which an outlier point must lie, and then determines nearby inlier points by constructing a frustum around this line as shown in Figure 2. The projection in the 3D space of the four neighbouring pixels (located at the corners of the $n \times n$ window centred on the considered pixel) in the disparity map yield four rays corresponding to the boundary edges of the frustum.

The histogram of the number of points along the depth axis is then computed, where a peak represents a cluster of points. Due to the similarity in terms of colour among neighbouring points, the RGB values of an outlier point are compared to those of the inlier points (within each group) using Euclidean distance, retaining only those points having a distance in this colour space that is less than an upper limit l_c.

3.3. Point position improvement

Given the clusters of inliers, three approaches are then used to correct an outlier's position. The first approach determines if the ray intersects an object that can be well represented with a plane, such as a wall or floor. This is done by attempting to fit planes to the points in a scene using the Random Sample Consensus (RANSAC) method as implemented in the Point Cloud Library [12]. Given a plane P defined as follows:

$$ax + by + cz = q \tag{5}$$

where q is a scalar, and a, b, c are the coefficients of the normal to the plane, the intersection of the plane P with the ray $L(t)$ is then found by first computing t:

$$t = \frac{q - (ax_1 + by_1 + cz_1)}{ax_2 + by_2 + cz_2} \tag{6}$$

Substitution of t in Equation (4) then yields the coordinates of the point where the ray $L(t)$ and the plane P intersect, corresponding to the position of an object on the surface. Otherwise, if the distance D_i between the intersection of the plane and the inliers is greater than a threshold l_{D1}, the point is determined to be too far away from similar inliers and the outlier point is kept unchanged. Given points $\vec{p_1}$ and $\vec{p_2}$ defined in Section 3.2 having coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), respectively, the squared (perpendicular) distance between an inlier $\vec{p_0} = (x_0, y_0, z_0)$ and the ray $L(t)$ is given by [13]:

$$D_i^2 = [(x_1 - x_0) + (x_2 - y_1)t]^2 + [(y_1 - y_0) + (y_2 - y_1)t]^2 + [(z_1 - z_0) + (z_2 - z_1)t]^2 \tag{7}$$

While the previous approach attempted to determine whether an outlier belongs to a plane fitted to a large set of points, the second approach uses a similar methodology but uses a plane that is locally fitted to the inliers within each group using Singular Value Decomposition (SVD) [14]. The same threshold l_{D1} is used to determine whether the new point coordinates are too far away from the inliers.

Finally, if the above two approaches fail, the inliers are not well-represented with a plane and may thus form part of a curved surface. RPCA is employed to find the new point coordinates in this scenario, which decomposes a matrix into low-rank and sparse matrices as follows:

$$\mathbf{M} = \mathbf{L} + \mathbf{S} \tag{8}$$

where $\mathbf{M}$ is the matrix representing the point cloud, $\mathbf{L}$ is the low-rank matrix that can be considered to correspond to the inliers, and $\mathbf{S}$ is a sparse matrix corresponding to the outliers [6]. A threshold l_{D2} is used to determine whether the new point coordinates are close enough to the inliers.

If plausible coordinates are found by any above method, the coordinates which are the closest to the corresponding cluster is selected and the outlier is set as an inlier. On the other hand, if no method produces satisfactory coordinates, the point is checked again in subsequent iterations (where an iteration spans all the non-corrected outliers). Thus, each iteration increases the number of inliers that can be used to correct the remaining outliers.

4. EVALUATION

Five scenes from the Sparse Light Field Dataset (SLFD)[1] are considered, each containing a 9×9 grid of synthetic images at a resolution of 512×512 pixels.

[1]Database is available at `http://clim.inria.fr/DataSoftware.html`

Table 1: Quality evaluation of estimated point clouds with results averaged over all views. The best results are in boldface.

Light fields	Accuracy (1.0)			Accuracy (5.0)			Completeness (1.0)			Completeness (5.0)		
	A1	A2	A3	A1	A2	A3	A1	A2	A3	A1	A2	A3
Aretheuse	0.107	0.111	**0.153**	0.423	0.439	**0.566**	0.056	**0.059**	**0.059**	0.270	**0.285**	**0.285**
Chair	0.120	0.125	**0.137**	0.418	0.428	**0.465**	0.066	**0.068**	**0.068**	0.274	0.277	**0.278**
Lion	0.065	0.059	**0.079**	0.283	0.261	**0.337**	0.030	**0.036**	**0.036**	0.146	**0.180**	0.178
Philosophers	0.320	0.262	**0.335**	0.759	0.729	**0.862**	**0.192**	0.155	0.158	0.665	0.649	**0.669**
Toys_cloud	0.202	0.210	**0.253**	0.613	0.618	**0.725**	0.133	0.140	**0.141**	0.549	0.569	**0.585**
Average	0.163	0.153	**0.191**	0.499	0.495	**0.591**	**0.095**	0.092	0.093	0.381	0.392	**0.399**

Table 2: Quality evaluation of estimated disparity maps with results averaged over all views. The best results are in boldface.

Light fields	MSE			BadPix (0.01)			BadPix (0.03)			BadPix (0.07)			Q25		
	A1	A2	A3	A1	A2	A3	A1	A2	A3	A1	A2	A3	A1	A2	A3
Aretheuse	**0.57**	0.63	0.59	69.39	68.16	**67.59**	33.00	32.12	**30.88**	14.53	13.70	**12.55**	0.80	0.76	**0.75**
Chair	5.69	**5.58**	7.68	66.06	63.93	**63.38**	32.26	29.61	**29.02**	14.99	13.33	**13.26**	0.70	0.66	**0.65**
Lion	0.51	0.52	**0.40**	76.12	71.68	**71.24**	42.67	33.05	**32.03**	19.96	13.47	**12.63**	1.06	0.88	**0.87**
Philosophers	1.63	0.70	**0.45**	**71.63**	78.52	77.49	**42.57**	47.74	45.56	23.97	23.18	**20.54**	**0.87**	1.18	1.12
Toys_cloud	0.68	0.75	**0.60**	78.90	78.12	**77.70**	48.48	46.80	**45.60**	24.84	23.00	**20.96**	1.21	1.16	**1.13**
Average	1.82	**1.64**	1.94	72.42	72.08	**71.48**	39.80	37.86	**36.62**	19.66	17.34	**15.99**	0.93	0.93	**0.90**

The ground truth and reconstructed 3D point clouds are compared using the accuracy and completeness measures as described in [15, 16]. The results are shown in Table 1, where method A1 represents the point clouds derived from the projection of the disparity maps computed by ProbFlow [9], method A2 represents the point clouds derived from the projection of the disparity maps fused using uncertainty estimation, and method A3 represents the point cloud derived after the proposed denoising method has been applied.

Each type of disparity map is compared to the corresponding ground-truth disparity map, using the commonly used Mean Square Error (MSE), Q25, and BadPix(e) measures, where the latter computes the percentage of pixels having an error larger than e. The results are shown in Table 2, where methods A1 and A2 represent the disparity maps obtained from ProbFlow and the disparity maps fused using uncertainty estimation, respectively, while method A3 denotes the re-projected disparity maps from the 3D point clouds corrected with the proposed method.

All measures are averaged over all views in a scene. Values of the parameters are set to $n = 17$, $l_c = 20$, $l_{D1} = 40$, $l_{D2} = 20$, $pr = pc = 15$, $K = 50$, and $m = 1$.

As shown in Tables 1 and 2, the fusion of disparity maps using uncertainty estimation does not always lead to improvement in performance. This is mostly due to the measure of uncertainty which is not always sufficiently precise. However, in most cases, it is evident that uncertainty estimation can indeed improve the quality of both 3D point clouds and disparity maps, and is thus an effective manner of combining multiple candidate disparity maps.

Comparing the denoised point clouds and corresponding disparity maps with the original versions (i.e. methods A3 and A2, respectively), it is also evident that the proposed denoising method is effective in improving the quality of the point clouds and disparity maps. Indeed, the BadPix measures (which use low error tolerances) indicate that improvements can be made even at the sub-pixel level, despite the light fields containing a substantial number of relatively large disparities (in the range $[-26, 12]$ pixels).

It is interesting to note that the proposed denoising method is quite robust given that the objects within the scenes considered are typically quite similar in terms of colour, thus making it difficult to determine the object to which an outlier should be assigned. This is also true for the 'Chair' light field, which yields the highest MSEs for all methods. One reason is that this scene exhibits the largest disparities compared to the other scenes. Moreover, since even the disparities computed by ProbFlow contain substantial errors, the inliers are not highly robust and some outliers are undetected by the SOR filter, which impede the effectiveness of the proposed approach. Nevertheless, improvements in terms of other evaluation metrics are still evident.

5. CONCLUSIONS AND FUTURE WORK

A method for noise removal in 3D point clouds acquired from light fields has been described. The proposed approach first exploits uncertainty measures to have better estimates of depth values and then corrects the positions of outlier points by exploiting geometric and photometric properties together with known camera parameters. This is in contrast to existing methods which simply remove outliers. The proposed pipeline was shown to achieve promising performance in both 2D and 3D, thus leading the way for future avenues of research. Future work includes the use of a more advanced method to detect outliers, and the consideration of non-Lambertian surfaces.

6. REFERENCES

[1] Marc Levoy and Pat Hanrahan, "Light field rendering," in *23rd Annual Conf. on Computer Graphics and Interactive Techniques*. 1996, SIGGRAPH '96, pp. 31–42, ACM.

[2] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen, "The lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 43–54.

[3] Katja Wolff, Changil Kim, Henning Zimmer, Christopher Schroers, Mario Botsch, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung, "Point cloud noise and outlier removal for image-based 3d reconstruction," in *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 118–127.

[4] Yujing Sun, Scott Schaefer, and Wenping Wang, "Denoising point sets via L0 minimization," *Computer Aided Geometric Design*, vol. 35-36, pp. 2 – 15, 2015, Geometric Modeling and Processing 2015.

[5] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927 – 941, 2008, Semantic Knowledge in Robotics.

[6] Gerasimos Arvanitis, Aris S. Lalos, Konstantinos Moustakas, and Nikos Fakotakis, "Real-time removing of outliers and noise in 3d point clouds applied in robotic applications," in *ICR*, 2017.

[7] Youyi Zheng, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, Oct 2011.

[8] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or, "Bilateral mesh denoising," in *ACM SIGGRAPH 2003 Papers*, New York, NY, USA, 2003, SIGGRAPH '03, pp. 950–953, ACM.

[9] Anne S. Wannenwetsch, Margret Keuper, and Stefan Roth, "Probflow: Joint optical flow and uncertainty estimation," in *Proceedings of the Sixteenth IEEE International Conference on Computer Vision*, 2017.

[10] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke, "A dataset and evaluation methodology for depth estimation on 4dÂ light fields," in *Computer Vision – ACCV 2016*, Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato, Eds., Cham, 2017, pp. 19–34, Springer International Publishing.

[11] "Lines and planes in $\mathbb{R}^3$," `http://www.math.harvard.edu/~wboney/fall15/LinesPlanes.pdf`, last visited on Oct. 25, 2018.

[12] "Plane model segmentation," `http://pointclouds.org/documentation/tutorials/planar_segmentation.php`, last visited on Oct. 25, 2018.

[13] Eric W. Weisstein, "Point-Plane Distance, url = http://mathworld.wolfram.com/point-planedistance.html," last visited on Oct. 10, 2018.

[14] Inge Söderkvist, "Using svd for some fitting problems," `https://www.ltu.se/cms_fs/1.51590!/svd-fitting.pdf`, last visited on Oct. 25, 2018.

[15] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, June 2006, vol. 1, pp. 519–528.