OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images

Charles-Olivier Artizzu, Haozhou Zhang, Guillaume Allibert, Cédric Demonceaux

OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images

Charles-Olivier Artizzu*, Haozhou Zhang†, Guillaume Allibert* and Cédric Demonceaux†

*Université Côte d'Azur, CNRS, I3S, France. Emails: artizzu,allibert@i3s.unice.fr

†VIBOT, ERL CNRS 6000, ImViA, Université Bourgogne Franche-Comté, France. Email: cedric.demonceaux@u-bourgogne.fr

***Abstract*—Spherical cameras and the latest image processing techniques open up new horizons. In particular, methods based on Convolutional Neural Networks (CNNs) now give excellent results for optical flow estimation on perspective images. However, these approaches are highly dependent on their architectures and training datasets. This paper proposes to benefit from years of improvement in perspective images optical flow estimation and to apply it to omnidirectional ones without training on new datasets. Our network, OmniFlowNet, is built on a CNN specialized in perspective images. Its convolution operation is adapted to be consistent with the equirectangular projection. Tested on spherical datasets created with Blender[1] and several equirectangular videos realized from real indoor and outdoor scenes, OmniFlowNet shows better performance than its original network without extra training.**

I. Introduction

With the arrival on the market of more affordable and more accurate 360 cameras, their use is proliferating. The latest generation of cameras, such as Instax 360 One X, Ricoh Theta Z1, and Samsung Gear 360, offers representative spherical images with up to 4K resolution. Thanks to their wide field of view, they can capture their entire environment in a single image. These sensors are therefore used in many mobile robotics tasks, such as image-based navigation [1], monitoring [2] or simultaneous visual localization and mapping (SLAM) [3].

The equirectangular projection is a simplified representation of images taken by spherical cameras, particularly to facilitate human interpretation. In this projection, the latitude and longitude of spherical images are projected in horizontal and vertical coordinates on a 2D plane. The equirectangular projection is different from classical perspective images and suffers from important distortions in the vicinity of the polar regions, as shown in Fig. 1. The methods developed for perspective have to be rethought to fit with the distortions. Distortions imply that an object will appear differently depending on its latitude due to the non-linearity of the equirectangular projection [4]. As a result, the image processing methods traditionally used for perspective images cannot be applied to equirectangular images, especially in the case of optical flow estimation.

The optical flow consists of estimating the displacement of the scene pixels between two frames. Precisely estimated, it gives information about the environment and the sensor's motion. Several methods are used to solve the aperture problem induced by the brightness constancy equation. The oldest

[1]https://www.blender.org

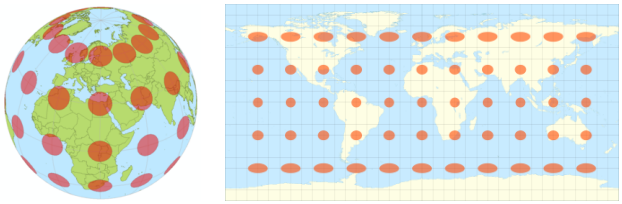

Fig. 1. Spherical and equirectangular projection of the globe. In orange (Tissot's indicatrix) is the amplitude of distortions [5].

is the variational method trying to constrain the problem to be as smooth as possible [6]. Other approaches propose a phase-based model [7]. In recent years, significant progress has been made in the accuracy of optical flow estimation with the apparition of convolutional neural networks (CNNs) [8]–[11]. They now dominate optical flow evaluation in the famous perspective datasets [12], [13]. Still, CNNs critically rely on extensive and representative training data sets.

To deal with equirectangular distortions, networks could be trained on spherical datasets. However, building a complete optical flow set would be very time-consuming and expensive. On the other hand, virtual ones often lack representativeness.

Some studies propose other approaches to deal with the equirectangular distortions. Several methods modify the input images using the Fourier transform [14] or spherical polyhedron [15]. However, this modification of the input images requires redefining the whole feature map for a good estimation of the optical flow. Other methods directly modify the shape of the kernel used for convolution. In [16], the kernel sizes are roughly increased near the polar regions. This method gives promising results but neglects the spherical projection and its distortions. Therefore, [17], [18] extend this idea by taking into account the equirectangular transformation. The size of each sample grid is modified using a local perspective projection on the sphere. Consequently, the shape of the kernel is related to the amplitude of the distortion at its latitude.

In this paper, the distortion-aware convolution method is used to implement a solution that is as universal as possible and consistent with the equirectangular projection. The proposed approach preserves the feature map and architecture of high-performance perspective CNNs. The general idea consists in locally projecting the spherical image onto its perspective equivalent. The equirectangular convolution then replaces the

usual 2D convolution by modifying the coordinates of the kernel points used for the convolution. The main contribution of this paper is to benefit from years of improvement in perspective optical flow estimation and to transfer it to equirectangular images. Comparison tests are performed on spherical datasets to demonstrate the performance of the new network, OmniFLowNet. These datasets were built with Blender and are the second contribution of this paper. The last contribution of this article is the evaluation of optical flow estimation network on real equirectangular images.

The remainder of the paper is organized as follows. Section II reviews the optical flow estimation process with CNNs, especially FlowNet. The proposed solution, OmniFlowNet, is described in Section III. Section IV compares the performances of the new network and its initial version on generated datasets. Finally, Section V presents the results of OmniFlowNet on real equirectangular images.

II. OPTICAL FLOW ESTIMATION

In the last decades, CNNs have shown their superiority in many image processing tasks. Especially, these networks are on top of the evaluation rankings of famous optical flow perspective datasets [12], [13]. One of the earliest CNN precursors is FlowNet [8]. It outperforms traditional methods of optical flow estimation and many recent networks are based on it [9]–[11].

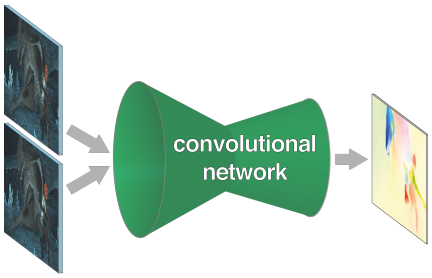

Fig. 2. Two main parts of FlowNet models: the contracting part and the expanding part. [8]

A. FlowNet

In the first version, two architectures called FlowNet-S and FlowNet-C are proposed and compared. FlowNet-S is the simplest network that directly stacks the two input images and transmits them via a CNN architecture. On the other hand, FlowNet-C first extracts the features of the two input images separately and then combines them in a correlation layer before transmitting to the CNN network. These convolutional neural networks are trained end-to-end using a large data set of image pairs and ground truth optical flow. The input of the network is a pair of successive perspective images, and the output is an estimate of the flow fields (u_f, v_f), as shown in Fig. 2. FlowNet architectures contain a contracting and an expanding part, both using back-propagation. In the expanding part, several "up-convolutional" layers increase the resolution of the feature maps. The contracting and expanding parts are connected to preserve both high-level information of the coarser feature maps and low-level information of the contracting part.

B. FlowNet2

In FlowNet2, several networks (FlowNet-S/FlowNet-C) are stacked to form a deeper network. At the end of each network, the second image I_2 is warped to the first image I_1 using the optical flow estimated by the previous network. This way, the next network can only focus on the remaining optical flow between I_1 and I_2 after warping. Each network in the stack has a particular architecture to perform a specialized task: some focus more on large displacements, others on small ones. Each network is trained solo while keeping the other CNNs weights fixed during the process to ensure better performance.

Due to its great performances in 2017, FlowNet2 is one of the most cited networks and has given birth to an important family. Thus, LiteFlowNet2, the most recent network of the FlowNet family available on GitHub[2], is selected as baseline in this paper to obtain the best performance for optical flow estimation. Our solution is developed to be as universal as possible, easily implemented on every network.

III. PROPOSED SOLUTION

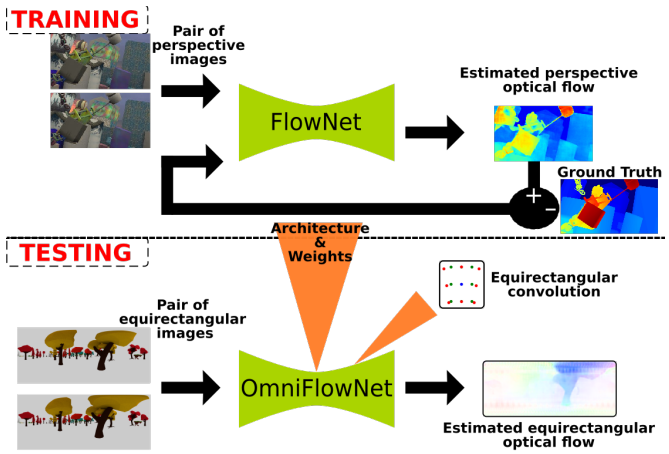

Fig. 3. Training: FlowNet is trained with perspective images and ground truth. Testing: OmniFlowNet is built with the architecture and weights of the previously trained FlowNet network and adapted with the equirectangular convolution.

Based on FlowNet networks, this paper proposes to build OmniFlowNet: a CNN trained on perspective images and used on spherical images for optical flow estimation. This strategy avoids generating large and complex spherical datasets for training and is robust against equirectangular distortions. The overall architecture and weights of OmniFlownet come from FlowNet models trained on perspective images and ground-truth optical flow, as shown in the first row of Fig. 3. The standard convolution is then replaced by the new distortion-aware convolution to deal with equirectangular inputs. Our

[2]https://github.com/twhui/LiteFlowNet2

solution is compatible with every size of kernel, stride, or padding. Once these code modifications realized, pairs of equirectangular images can be used in OmniFlowNet as input. The second row of Fig. 3 presents the testing process of the network.

A. Equirectangular convolution

Inspired by [19], this section presents the convolution adaptation. The usual perspective kernel is modified to fit the equirectangular distortions. To build a kernel of resolution r and angular size α centered in a location $p_{00} = (u_{00}, v_{00})$ in the equirectangular image, the center coordinates are first transformed to spherical system $p_{s,00} = (\phi_{00}, \theta_{00})$ using

$$\phi_{00} = \left(u_{00} - \frac{W}{2}\right)\frac{2\pi}{W}; \quad \theta_{00} = -\left(v_{00} - \frac{H}{2}\right)\frac{\pi}{H}, \tag{1}$$

where W and H are respectively the width and the height of the equirectangular image in pixels. Each point of the kernel is defined by

$$\hat{p}_{spher,ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{z}_{ij} \end{bmatrix} = \begin{bmatrix} i \\ j \\ d \end{bmatrix}, \tag{2}$$

where i and j are integers in the range $\left[-\frac{r-1}{2}, \frac{r-1}{2}\right]$ and d is the distance from the center of the sphere to the kernel grid. In order to cover the field of view α, the distance is set to $d = \frac{r}{2\tan\left(\frac{\alpha}{2}\right)}$. The coordinates of these points are computed by normalizing and rotating them to align the kernel center on the sphere. Therefore:

$$p_{spher,ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} = R_y\left(\phi_{00}\right) R_x\left(\theta_{00}\right) \frac{\hat{p}_{spher,ij}}{|\hat{p}_{spher,ij}|}, \tag{3}$$

where $R_a(\beta)$ stands for the rotation matrix of an angle β around the a axis. These coordinates are transformed to latitude and longitude in the spherical domain using

$$\phi_{ij} = \arctan\left(\frac{x_{ij}}{z_{ij}}\right); \quad \theta_{ij} = \arcsin\left(y_{ij}\right); \tag{4}$$

and finally back-projected to the original 2D equirectangular image

$$u_{ij} = \left(\frac{\phi_{ij}}{2\pi} + \frac{1}{2}\right) W; \quad v_{ij} = \left(-\frac{\theta_{ij}}{\pi} + \frac{1}{2}\right) H. \tag{5}$$

In Fig. 4, some example of kernels at different latitude and longitude are presented. The blue point defines the center of the kernel $p_{00} = (u_{00}, v_{00})$. The red points are the positions of the elements $p_{ij} = (u_{ij}, v_{ij})$ in the adapted kernel, defined as previously. The green points are the positions of elements in a standard perspective kernel given by:

$$u_{persp,ij} = u_{00} + ir; \quad v_{persp,ij} = v_{00} + jr. \tag{6}$$

B. OmniFlowNet convolution

The OmniFlowNet architecture is based on the implementation of open-source software shared by the LiteFlowNet2 team [11]. It is based on the Caffe framework [20] and allows GPU acceleration for training and testing. In the Caffe engine, the function *im2col* contains the convolution operation. Updating this function with the previously proposed models and building the distribution allows OmniFlownet to compute the spherical convolution.

In *im2col* function, the coordinates of elements on each spatial patch are modified according to patch latitude. Therefore, the distortion of the output matrix is the same as the equirectangular image. In OmniFlowNet, the new distortion-aware convolution replaces all standard convolutions in the contracting or expanding part of the array.

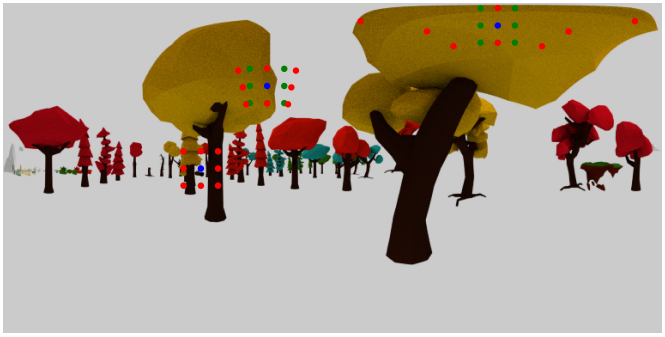

Fig. 4. Example of kernels with different latitude and longitude. In blue is the center of the kernel, in green the perspective kernel and in red the equirectangular one. Wide distortions are visible near the poles.

IV. Results on virtual datasets

First, the performance of OmniFlowNet and its initial versions are tested on equirectangular data sets with a ground truth optical flow, specially built for this study. To the best of our knowledge, all the famous datasets available with optical field truth flow, such as MPI Sintel [12], KITTI [13] or FlyingThings3D [21] contains only perspective images. Therefore, omnidirectional datasets are constructed to compare networks.

A. Equirectangular datasets

1) Scene creation: Inspired by the realization of the MPI Sintel dataset, Blender is used. This tool is a free and open-source 3D computer graphics software for the creation of animated films, visual effects, 3D printed models, interactive 3D applications, and video games. Three different scenes called `Cartoon Tree`, `Forest` and `LoyPolyModel` are generated with free 3D models available online (Fig. 5). An equirectangular camera, simulated by Blender, moves in these fixed scenes with different orientations given by Euler angles (roll ϕ, pitch θ, yaw ψ). The sets are shown in TABLE II. The image resolution is 768×384 to maintain a width to height ratio of 2.

TABLE I

AE (DEFINED IN EQ.7, IN DEGREES) AND EE (DEFINED IN EQ.8 COMPUTED FOR BOTH LITEFLOWNET2 AND OMNIFLOWNET NETWORKS ON THE THREE SCENES. THE RESULTS ARE AVERAGED ON CASES.

	Cartoon Tree				Forest				Low Poly Model			
	LiteFlowNet2		OmniFlowNet		LiteFlowNet2		OmniFlowNet		LiteFlowNet2		OmniFlowNet	
	AE	EE	AE	EE	AE	EE	AE	EE	AE	EE	AE	EE
Case 1	61.67	3.76	**53.87**	**3.05**	58.81	16.56	**54.47**	**15.65**	58.22	7.29	**54.62**	**6.87**
Case 2	62.65	4.22	**56.07**	**3.16**	57.91	8.23	**53.99**	**7.37**	57.66	6.89	**53.54**	**6.41**
Case 3	64.16	7.41	**55.89**	**6.03**	57.73	8.47	**56.32**	**7.64**	57.55	7.51	**55.42**	**6.99**
Case 4	63.78	7.01	**53.14**	**5.7**	57.99	9.18	**55.42**	**8.23**	58.68	8.96	**57.25**	**8.65**
Average on Cases	63.07	5.60	**54.74**	**4.49**	58.11	10.61	**55.05**	**9.72**	58.03	7.66	**55.21**	**7.23**
	AE diff		EE diff		AE diff		EE diff		AE diff		EE diff	
Average on Cases	**8.32**		**1.11**		**3.06**		**0.89**		**2.82**		**0.43**	

TABLE II

CAMERA'S ORIENTATION FOR DIFFERENT DIRECTIONS (GIVEN AS EULER ANGLES) IN DIFFERENT SCENES.

Case	Cartoon Tree	Forest	LowPolyModel
1	$(-\frac{3\pi}{4}, \pi, 0)$	$(-\frac{3\pi}{4}, \pi, -\frac{\pi}{2})$	$(\frac{\pi}{4}, 0, 0)$
2	$(-\frac{\pi}{4}, \pi, 0)$	$(-\frac{\pi}{2}, \pi, -\frac{\pi}{2})$	$(\frac{3\pi}{4}, 0, 0)$
3	$(-\frac{\pi}{2}, \frac{3\pi}{4}, 0)$	$(-\frac{\pi}{2}, \frac{3\pi}{4}, -\frac{\pi}{2})$	$(\frac{\pi}{2}, -\frac{\pi}{4}, 0)$
4	$(-\frac{\pi}{2}, \frac{5\pi}{4}, 0)$	$(-\frac{\pi}{2}, \frac{5\pi}{4}, -\frac{\pi}{2})$	$(\frac{\pi}{2}, \frac{\pi}{4}, 0)$

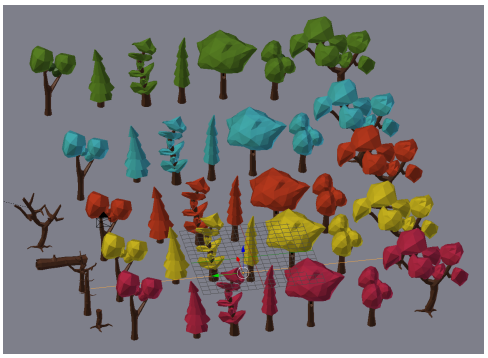

Fig. 5. 3D models used to generated Forest datasets.

2) Ground truth extraction: The *Vector Pass* given by the Blender Cycles Renderer is used to extract the ground truth optical flow, as presented in [22]. This render pass is usually helpful to produce motion blur by giving the motion of every pixel in the image. Here, the *Vector Pass* returns the pixel displacement in the horizontal and vertical directions perpendicular to the camera axis, the ground truth optical flow. Fig. 6 presents an example of images generated by the scene Forest and their ground truth optical flow.

B. Evaluation

1) Metrics: To evaluate OmniFlowNet, two classical metrics are used [23]: the average Angle Error (AE) and the average Endpoint Error (EE). The AE between a flow vector (u_f, v_f) and the ground-truth flow (u_{fgt}, v_{fgt}) is the angle in 3D space between $(u_f, v_f, 1.0)$ and $(u_{fgt}, v_{fgt}, 1.0)$. This error is computed by taking the dot product of the vectors, dividing by the product of their lengths, and then taking the inverse cosine:

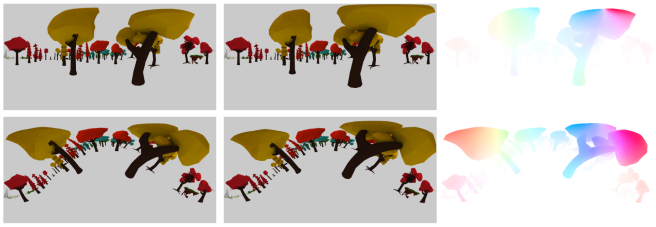

Fig. 6. Example of two image pairs and their optical flow ground truth generated by the scene Forest with different camera orientations.

$$AE = cos^{-1}\left(\frac{1 + u_f \cdot u_{fgt} + v_f \cdot v_{fgt}}{\sqrt{1 + u_f^2 + v_f^2}\sqrt{1 + u_{fgt} + v_{fgt}}}\right) \quad (7)$$

The goal of the AE is to provide a relative measure of performance that avoids the "divide by zero" problem for zero flows. Errors in large flows are penalized less in AE than errors in small flows. We also compute an absolute error, the EE:

$$EE = \frac{1}{N}\sum_N \sqrt{(u_{fgt} - u_f)^2 + (v_{fgt} - v_f)^2} \quad (8)$$

Although the use of AE is common, the EE measure is also appropriate to understand the magnitude of flow errors.

For ease of reading and comparison, the AE difference is set by comparing the AE values between FlowNet and OmniFlowNet. The result is an angle, positive if OmniFlowNet performs better than FlowNet, negative if not.

$$AE_{diff} = AE_{FlowNet} - AE_{OmniFlownet}. \quad (9)$$

The EE difference is also defined as the difference between the EE of FlowNet and OmniFlowNet. If OmniFlowNet performs better than FlowNet, the gain is a positive number of pixels, otherwise negative.

$$EE_{diff} = EE_{FlowNet} - EE_{OmniFlownet}. \quad (10)$$

2) Results and Analysis: Network weights come directly from LiteFlowNet2 [11]. The SINTEL set [12] is chosen to fit the synthetic images coming from the Blender rendering. From this trained model, the OmniFlowNet network is created

by adapting the equirectangular convolution, as described in Fig.3.

Both networks are tested on previously constructed equirectangular datasets. The angular error and the final error are compared for all results. Table I presents all AE (Eq.7, degrees) and EE (Eq.8, pixels) results and their average on orientation cases for OmniFlowNet and its initial LiteFlowNet2 network on all scenes. Second part of Table I directly shows the differences in AE (Eq.9) and EE (Eq.9) between the networks.

In all averaged cases, OmniFlowNet outperforms LiteFlowNet2 on the average angular error, which means that it measures better the direction of the motion vectors. As for the average endpoint error, OmniFlowNet has better results in almost all of them (small under-performance in 2 configurations in `Forest`). The overall gain expected, on average, is around 4.7 degrees in AE and 0.8 pixels in EE.

V. Results on real datasets

To the best of our knowledge, there is no omnidirectional dataset with optical flow ground truth available. So we compared the performances of LiteFlowNet2 and OmniFlowNet directly on real equirectangular images. These were taken with a RICOH THETA Z1. Two situations were mainly studied: one with a fixed camera in a moving scene and the other with a moving camera in an essentially rigid scene.

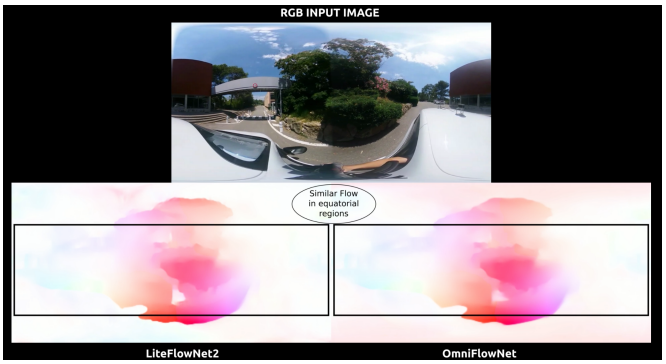

Fig. 7. *Car 1* case: RGB input image (top) and optical flow computed by LiteFlowNet2 (bottom-left) and OmniFlowNet (bottom-right). The optical flow estimated in the equatorial region is similar. The two networks behaves in the same way in low distortion areas.

A. Field protocol

Different real indoor and outdoor scenes were filmed with the omnidirectional camera to observe motion in the polar regions. In the cases of *Apple 1* and *Ball 3*, a sphere is moving next to a fixed camera in an indoor scene. In *Ball 1* and *Ball 2*, a moving ball is also observed but in an outdoor scene. In Car 1 and Car 2, the camera is held outside a moving car while passing under an archway. Once acquired, the equirectangular videos are reconstructed from the fisheye inputs using the RICOH THETA Movie Converter application[3]. Then, the videos are cut into multiple frames using *FFmpeg*[4].

[3]https://support.theta360.com/en/download/

[4]https://ffmpeg.org/

B. Optical flow results

The same networks as in the virtual dataset evaluation were used: LiteFlowNet2 with SINTEL weights and its OmniFlowNet version. A video compiles[5] all the experimental results and shows the comparison between the two networks.

In all cases, the optical flow estimated in the equatorial region by LiteFlowNet2 and OmniFlowNet are similar as shown in Fig.7. Our network behaves as its perspective version in low distorted areas.

Fig. 8. *Ball 1* case: RGB input image (top) and optical flow computed by LiteFlowNet2 (bottom-left) and OmniFlowNet (bottom-right). OmniFlowNet better estimates the optical flow in the top polar region is better. The ball is clearly more visible and smoother in that case.

Fig. 9. *Apple 1* case: RGB input image (top) and optical flow computed by LiteFlowNet2 (bottom-left) and OmniFlowNet (bottom-right). OmniFlowNet better estimates the optical flow in the top polar region is better. The arm is smoother.

In polar regions, the optical flow estimated by OmniFlowNet is smoother and, all the time, more coherent than the network in perspective, as shown in Fig.8. Whereas the ball motion estimated by LiteFlowNet2 is a shredded mark, OmniFlowNet predicts a complete ball with coherent motion. In Fig.9, the arm moving above the north pole of the camera has a smoother predicted optical flow by the spherical network than the perspective one. The equirectangular convolution helps the network to better understand and calculate motion in highly distorted areas.

[5]http://www.i3s.unice.fr/~allibert/Videos/icpr20_video.mp4

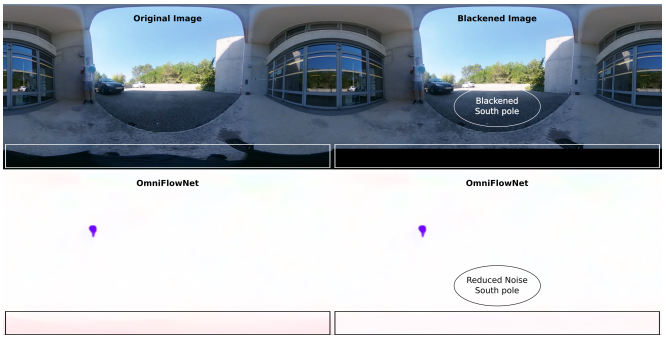

Fig. 10. *Ball 1* case with blackened south pole: original RGB input image (top-left), new RGB input image with blackened south pole (top-right) and respective optical flow computed by OmniFlowNet (bottom-left) and OmniFlowNet (bottom-right). The optical flow computed with the blackened images shows less noise in the south pole. Thus camera tripod and equirectangular reconstruction add noise to the optical flow estimation.

The south polar region often presents optical flow noise in the OmniFlowNet estimation. This is related to the artifacts created by the tripod holding the camera. When reconstructing the equirectangular image from the two fish-eye lenses, this highly distorted region certainly brings some noise to the final image. While LiteFlowNet interprets it as white noise, OmniFlowNet with the equirectangular convolution probably detects coherent motion. The Fig.10 shows that by blackening the entire tripod region, the induced noise is reduced.

Overall, on several diverse real data sets, OmniFlowNet outperforms its perspective version in estimating better and smoother optical flow in polar regions and shows the same performance in the equatorial region. Due to space limitation, only brief but most representative parts of total results are presented in the video[6].

VI. CONCLUSION

In this paper, we present OmniFlowNet, a CNN for optical flow estimation on omnidirectional images. Built on a proven reference CNN, our network benefits from many previous studies on optical flow estimation. It adapts to equirectangular distortions using distortion-aware convolution. The high-end methods for optical flow estimation on perspective images are applied locally on the sphere. Thus it only uses pre-trained weights on perspective images and does not require extra training on equirectangular datasets. OmniFlowNet outperforms its original network on omnidirectional sets built with Blender and on real equirectangular sets with various scenes and motion. Our code implementation and omnidirectional dataset is available on GitHub[7].

REFERENCES

[1] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese, "Deep Visual MPC-Policy Learning for Navigation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3184–3191, 2019.

[6]http://www.i3s.unice.fr/~allibert/Videos/icpr20_video.mp4

[7]https://github.com/coatz/OmniFlowNet

[2] K. Saitoh, T. Machida, K. Kiyokawa, and H. Takemura, "A 2d-3d integrated interface for mobile robot control using omnidirectional images and 3d geometric models," *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 173–176, 2006.

[3] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct SLAM for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 141–148.

[4] F. Pearson, *Map projections : theory and applications*. CRC Press Boca Raton, Fla, 1990.

[5] E. Gaba. (2008) Tissot indicatrix equirectangular projection. [Online]. Available: https://commons.wikimedia.org/wiki/File:Tissot_indicatrix_world_map_equirectangular_proj.svg

[6] B. K. Horn and B. G. Schunck, "Determining Optical Flow," in *Artificial Intelligence*, vol. 17, 08 1981, pp. 185–203.

[7] A. Radgui, C. Demonceaux, E. Mouaddib, M. Rziza, and D. Aboutajdine, "Optical flow estimation from multichannel spherical image decomposition," *Computer Vision and Image Understanding*, vol. 115, pp. 1263–1272, 09 2011.

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Haüsser, C. Hazırbas, and V. Golkov, "FlowNet: Learning Optical Flow with Convolutional Networks," *ICCV*, 04 2015.

[9] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1647–1655, 07 2017.

[10] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018, pp. 8981–8989.

[11] ——, "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 02 2020, pp. 1–1.

[12] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, 2012, pp. 611–625.

[13] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," *ICLR*, 01 2018.

[15] Y. K. Lee, J. Jeong, J. S. Yun, C. W. June, and K. jin Yoon, "SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360 degree Images," in *CVPR*, 11 2018.

[16] Y.-C. Su and K. Grauman, "Learning Spherical Convolution for Fast Features from 360 Imagery," in *Advances in Neural Information Processing Systems 30*, 08 2017, pp. 529–539.

[17] K. Tateno, N. Navab, and F. Tombari, "Distortion-Aware Convolutional Filters for Dense Prediction in Panoramic Images," *The European Conference on Computer Vision (ECCV)*, pp. 732–750, 09 2018.

[18] B. Coors, A. P. Condurache, and A. Geiger, "SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images," *The European Conference on Computer Vision (ECCV)*, pp. 525–541, 09 2018.

[19] C. Fernandez, J. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. Guerrero, "Corners for Layout: End-to-End Layout Recovery from 360 Images," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1255–1262, 01 2020.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, 06 2014.

[21] N. Mayer, E. Ilg, P. Haüsser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, 06 2016.

[22] A. Ranjan, D. Hoffmann, D. Tzionas, S. Tang, J. Romero, and M. Black, "Learning Multi-Human Optical Flow," *International Journal of Computer Vision*, 01 2020.

[23] S. Baker, S. Roth, D. Scharstein, M. J. Black, J. P. Lewis, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," *International Conference on Computer Vision*, vol. 92, 01 2007.