Revisiting Optical Flow Estimation in 360 Videos

Keshav Bhandari, Ziliang Zong, Yan Yan
Department of Computer Science, Texas State University, USA

***Abstract*—Nowadays 360 video analysis has become a significant research topic in the field since the appearance of high-quality and low-cost 360 wearable devices. In this paper, we propose a novel LiteFlowNet360 architecture for 360 videos optical flow estimation. We design LiteFlowNet360 as a domain adaptation framework from perspective video domain to 360 video domain. We adapt it from simple kernel transformation techniques inspired by Kernel Transformer Network (KTN) to cope with inherent distortion in 360 videos caused by the sphere-to-plane projection. First, we apply an incremental transformation of convolution layers in feature pyramid network and show that further transformation in inference and regularization layers are not important, hence reducing the network growth in terms of size and computation cost. Second, we refine the network by training with augmented data in a supervised manner. We perform data augmentation by projecting the images in a sphere and re-projecting to a plane. Third, we train LiteFlowNet360 in a self-supervised manner using target domain 360 videos. Experimental results show the promising results of 360 video optical flow estimation using the proposed novel architecture.**

I. Introduction

The immersive 360 video technology shows promising growth in the past years. Services such as GoPro, VeeR, Visbit, Facebook360 and YouTube have become great platforms for 360 videos. 360 videos are shaping the future of content creation and sharing. Hence, 360 videos will be an important digital medium in near future. This adds newer challenges and opportunities in computer vision research. One of such challenge is the motion and optical flow estimation in 360 videos.

Motion and optical flow estimation is important for 360 video understanding. Motion information can significantly aid tasks such as saliency detection, saliency prediction, gaze prediction, video piloting in 360 videos [1], [2], [3], [4]. Similarly, optical flow based panorama video stitching has shown impressive results compared with other methods. Deep Learning based optical flow estimation methods have shown significant improvement over classical methods [5], [6]. Evolution of optical flow estimation methods from simple CNN based architecture to complex feature pyramid based architecture shows significant improvements as well [7], [8], [9]. However, regular CNN architectures are not suitable for 360 videos because of inherent distortion caused by projection of spherical videos to plane. We can use techniques such as [10], [11], [12], [13] to achieve spherical convolution. However, these methods have enormous overheads while converting existing complex pyramid based architecture to fit the needs of distortion free convolution for 360 videos. First, the training of the optical flow network is unstable. Having many transform convolution layers will lead entire process complicated as architectures becomes bigger. Second, we may not be able to guarantee that our model works even if we transform the architecture. We need some metrics such as EPE(End Point Error) to decide if our model works well. Since we lack labelled 360 video dataset for optical flow, the only method that fits our requirement is self-supervised methods. But how do we train this architecture in a self-supervised manner? The core part of self-supervised approach is calculating the loss between warped image and target image using predicted flow. Are warping techniques generalizable? In later sections we aim to answer these questions.

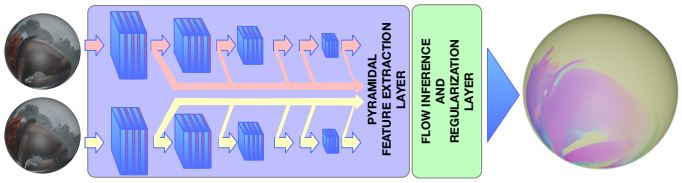

Fig. 1. Simple representation of LiteFlowNet360 network architecture. We put focus only on feature extraction block which is shown in detail. Flow inference and regularization layer is similar to the original implementation. Input to the network are equirectangular or spherical data. Each convolution layer in pyramidal network is transformed to adapt spherical convolution(shown in red color). Final output is optical flow in spherical domain.

Choosing a right architecture for our framework was the initial challenge we faced. However, we set certain requirements (like size, speed and efficiency) as a guide to choose the right architecture. There are many optical flow architectures to choose from, LiteFlowNet wins the competition. We will discuss more about this architecture in the following section. Framework we proposed would grow significantly as it includes significant changes as a part of perspective to a spherical domain transformation process. One more significant addition to this transformation process is the inclusion of special convolution to adapt the spherical nature for our dataset. This is important because the dataset we work on is an equirectangular plane, a sphere-to-plane projection. This planar projection incurs heavy distortion, which we have illustrated in Fig.2. These special convolution, termed as spherical convolution, are expensive in terms of computation, which voids our requirements. Therefore, we adopt techniques like kernel transformation using transformer network. One such architecture [14] dubbed as KTN has shown comparable improvement over later methods with less computation. We adopt KTN as our transformer network to learn spherical convolution.

Training an end-to-end optical flow architecture requires significant considerations of extra jobs like scheduling of

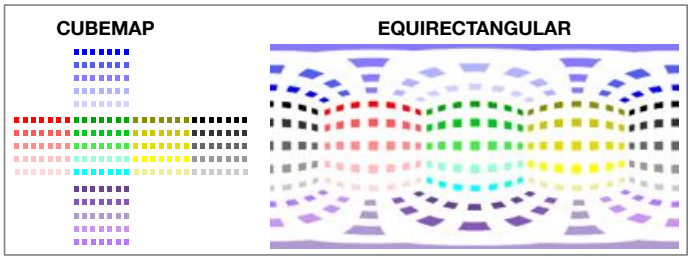

Fig. 2. Showing how regular kernel map does not work in equirectangular (right) projection. When kernel applied in cubemap (left) are mapped into equirectangular projection it suffers huge distortions.

training, implementing stacked architecture, considerations of motion magnitude and many other details to make it work on a par with the state-of-the-art results. Training architectures like this using better strategy to cope with gradually increasing task is an adoption of a popular philosophy, curriculum learning[15]. We have seen architecture like [8], [9] adopted this strategy to create a better model. Apart from traditional optical flow estimation strategies, we have additional requirements because of the spherical nature of the dataset. Therefore, end-to-end training of optical flow for 360 video is highly unstable as there are many parallel objectives to achieve. We need to make sure that our model size does not grow significantly. This can slow down the training and inference speed. We need to address the nature of optical flow in 360 domain which can change the interpretation of warping techniques, flow representation and many other aspects. To make it brief, training optical flow architecture in 360 videos is not straightforward. To achieve the stable training process and fulfill our requirements, we adopt a divide-and-conquer strategy, thus dividing the entire training process into three major stages.

First stage of LiteFlowNet360 is to train LiteFlowNet architecture in perspective video datasets like [16]. Then, we transform source CNN to target CNN layer wise. This transformation technique is progressive which means we need to do everything in-order. We will explain details about process of transformation in method section. After transforming into target CNN we will now further train entire model in an end-to-end fashion in supervised manner. To achieve this task we augment both source perspective videos and target optical flows into spherical distortion setting. When the second stage is complete, we will use a self-supervised scheme to further train our model in target videos. To do this, we need to perform a task like back-warping of frames using predicted flow. We use these predicted or warped frames as the basis of the training process by minimizing the similarity loss between ground frames and predicted frames. We adopt occlusion aware scheme inspired by [17].

In this paper we exploit the existing optical flow estimation techniques and distortion free convolution in 360 videos. Our contributions are three folds: **(i)** To the best of our knowledge, this is the first work to address deep learning based dense optical flow estimation in 360 videos. **(ii)** We present an algorithm inspired by [14] to transform learned representations from pre-trained network. **(iii)** We present a self-supervised learning approach since we do not have ground truth optical flow for 360 videos.

II. Related Work

Optical Flow Estimation. The classical optical flow estimation approaches [5], [6] used variational approaches to minimize energy based on brightness constancy and spatial smoothness. Recently, [7] proposed an end-to-end optical flow estimation with convolutional networks (FlowNet) using supervised scheme. Several other works based on CNN followed FlownNet including 3D convolution based approach [18], unsupervised approach [19], [20], [21] and pyramidal-coarse-to-fine approach [22], [23]. Recent variants such as [24], [25] used sparse matching by learning feature embedding. These methods were computationally expensive, making it impossible to train end-to-end fashion. FlowNet-2.0 [8] was an important addition in this series. It exploited curriculum learning approach [15]. In their work they address the weakness of FlowNet by addressing a smaller to larger range of displacement magnitude. However, the success of FlowNet-2.0 comes with a cost of over parametrization with around 160 Million parameters. [9] presented a more effective approach dubbed as LiteFlowNet. LiteFlowNet is around 30 times smaller and around 1.36 times faster. LiteFlowNet excelled FlowNet-2.0 by drilling down architecture details. LiteFlowNet proposed effective flow inference at each pyramid level, presented data fidelity and regularization as variational methods, whereas FlowNet only used a U-Net like architecture only. Self-supervised [17], [26] approaches for optical flow estimation are intuitive and reasonable approaches, as warping is one of the fundamental techniques used in successful deep learning based architecture. These techniques motivate our self-supervised learning scheme in the final stage.

CNN for 360 Video/Images. Performing direct convolution on spherical data led to inaccurate models [27], [28]. An intuitive approach to perform convolution on spherical data is to use convolution directly in cube map projection [29], [30]. This introduced less distortion but the model will have discontinuities, which led to sub-optimal model for several tasks. Another approach to learn rotation invariant CNN was to use graph convolution [31] techniques. This can be done by defining convolution in spectral domain [11], [12]. Similarly, this can also be done by projecting both feature maps and kernel in a spectral domain and apply regular CNN. These methods lose semantic information and were not useful in our case. In a recent year, several other spherical CNN based models have been proposed. Work such as [10], [13] considered distortion in sphere-to-plane projection of spherical images/videos. Recent method by [14], dubbed as Kernel Transformer Network(KTN) is a significant piece of work in this domain. This architecture efficiently transferred convolution kernels from perspective images to the equirectangular projection. Basically, KTN produced a function parametrised by a polar angle and kernel as output. This work preserved the

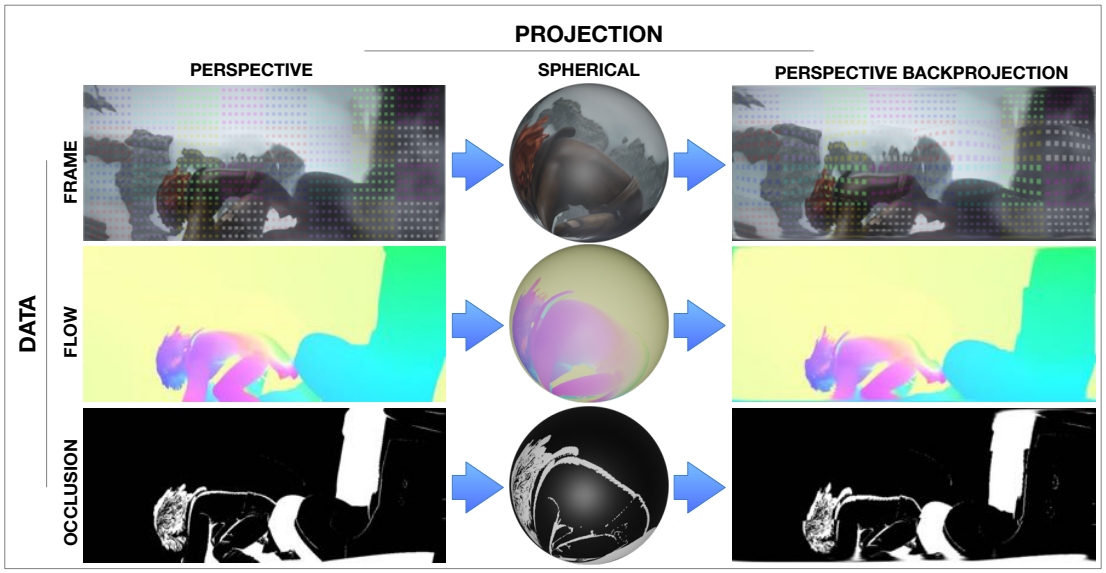

Fig. 3. Spherical Data Augmentation. Perspective videos are projected in a unit sphere and then back projected to equirectangular plane. This is intentionally lossy process to create distortion artifact(shown in top row) in perspective data.

source CNNs and maintained accuracy, meanwhile offering transferability and scalability. Our architecture is a modified version of KTN, which uses interleaving convolution techniques to reduce discontinuity during convolution.

360 Flow Estimation. [32] proposed an approach by back-projecting image points to a virtual curved retina intrinsic to the geometry of central panoramic camera. Their method could adopt to contemporary ego-motion algorithms. [33] implemented Lucas-Kanade based method for optical flow estimation in catadioptric images. They proposed new constraint based on motion model defined on perspective images. This new constraint-based model was used to compute optical flow for omnidirectional image sequences. [34] used multichannel spherical image decomposition techniques to compute optical flow for 360 image sequences. Similarly [35] proposed several variational regularization methods to estimate and decompose motion fields on the sphere. [36] implemented, adapted phase based method to compute optical flow using different treatments to account for 360 images.

Our work fall somewhere in the intersection of optical flow estimation and emerging domain of omnidirectional computer vision. However, none of the above methods address the optical flow estimation problem using deep learning methods.

III. Method

We choose LiteFlowNet [9] as the basis for our newly proposed LiteFlowNet360 (shown in Fig. 1) architecture because of its simpler design, lightweight (5.37M parameters) and highly efficient implementation. We represent our LiteFlowNet360 architecture in terms of two important blocks, feature extractor block (F) and regularization block (P) as shown in Eq. 1 where X is a sequence of two consecutive frames, k is the number of layer transform from 0^{th} to k^{th} layer in F and where n is an optimum number of layers eligible for transformation.

$$F_k(X) = \begin{cases} F_0(X) & : k = 0 \\ F_k(F_{k-1})(X) & : n > k > 0 \end{cases} \tag{1}$$

Feature extractor block F is transformed to adapt our need of 360 flow estimation as shown in Eq. 2. Each convolution layers $F_0, F_1, ..., F_{(n-1)}$ in feature extractor block is parametrised by a function $\Omega = g(\theta, \phi)$ in sphere, by generating different kernels for distortion above and below the equatorial region in sphere such that layers $F'_0, F'_1, ..., F'_{(n-1)}$ are our target layers. We compute location dependent kernel using polar and azimuthal angle θ and ϕ respectively.

We keep inference and regularization block as the same as LiteFlowNet. However, feature warping techniques at each pyramid level is transformed to address warping in 360 video domain. Further details will be explained in the following section.

$$F'_k(X) = \begin{cases} F'_0(F_0(X), \Omega) & : k = 0 \\ F'_k(F_k, \Omega)(F'_{k-1}(F_{k-1}, \Omega))(X) & : n > k > 0 \end{cases} \tag{2}$$

LiteFlowNet360 framework is an evolutionary architecture. We start from regular LiteFlowNet architecture and perform incremental transformation and training process to achieve final architecture. We formulate three important subsequent stages, transformation stage, intermediate refinement stage and final refinement.

A. Stage 1: Transformation

This stage starts with training the LiteFlowNet architecture with labeled data following [9]. The most important part of this stage is to transform convolution layers trained on perspective images to adapt to 360 images. We follow [14]

with some improvements, which we will present later in this section. Since we use equirectangular projection method, distortion depends only on polar angle. This leads to direct correspondence of the polar angle to the height of the input image, i.e., $y = \theta h/\pi$. This means we can utilize a single kernel for optimal row-group size(n_g) such that we have h/n_g projection matrices $P_i \in \mathbb{R}^{r_i \times k_h \times k_w}$, where $r_i = h_i \times h_w$ is target kernel for each row-group i and $(k_h \times k_w)$ is an original kernel size from source CNN as in [14]. Different from original implementation, we interleave these rows as shown in Algorithm-1 to maintain connectedness, where n_l is interleaving factor. We choose $n_l = 3$ in our case.

$$\begin{aligned} Y_k &= F_k(X), Y'_k = F'_k(F_k(X), \Omega) \\ L_k &= ||Y'_k - Y_k||^2 \end{aligned} \tag{3}$$

$$L'_k = \frac{1}{n_g} \sum_{i}^{n_g} L_k(\Omega(Y'^i_k), Y^i_k) \tag{4}$$

We train each layer of feature extractor evolutionarily. We feed augmented images created by warping perspective image sequence as inputs to transformed layer and original image sequence as inputs to source CNN. Warping process is done by plane-to-sphere and sphere-to-plane projection of perspective image sequences. This back projection trechnique introduce distortion in perspective videos, as shown in Fig.3 (top row). We train each layer with objective function presented in Eq.3 to minimize the L2 norm between feature map generated by source CNN and transformed CNN layers, where Y_k and Y'_k represent output at k_{th} layers in source and target architecture respectively and L_k is an L2 norm between feature map from source and target CNN.

We project feature map row-group wise in tangential plane and compute loss with respect to corresponding feature map row-group from perspective source CNN. We combine these losses by averaging all the losses in row-group as shown in Eq.4, where i refers to i-th row-group, and L'_k is an L2 norm averaged over row-group.

B. Stage 2: Intermediate Refinement

Though first stage can transform convolution layers to adapt spherical images, it does not guarantee that estimated flow are well represented. A common problem will arise when we try to warp inferred flow around the sphere. This is due to the nature of sphercial coordinates. We can observe in Fig. 4 that the size and the shape of the patches decreases as we move away from the equatorial region. This means u and v component(shown in figure) changes as we move away from the equatorial region. This is because of the difference between idea behind the optical flow representation in perspective domain vs spherical domain. Optical flow in perspective domain is represented by the displacement in terms of euclidean distance. However, in spherical domain flow information makes sense only if we represent flow in terms of angular displacement. This means we need to present u and v in terms of u_θ and v_ϕ component. Instead of obtaining a direct solution, which is beyond the scope of our work, we introduce some correction factor on original u and v and project it in spherical domain.

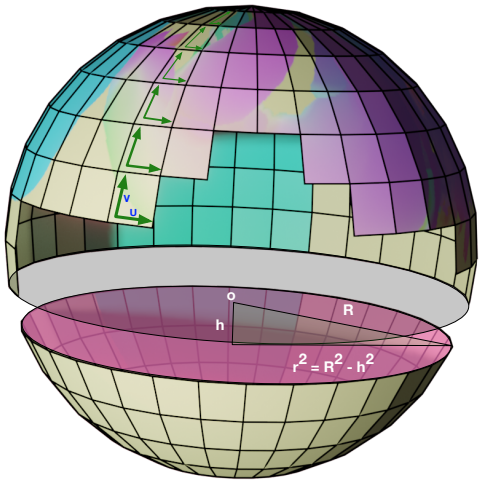

Fig. 4. Flow representation in spherical domain. (u, v) component changes as we move away from equator.

Algorithm 1: Interleaving Convolution

$Y \leftarrow tensor()$;
$X \leftarrow input$;
$tied\_weights \leftarrow n_g$;
$n\_transform \leftarrow h/n_g$;
for *each* $row \in [0, n\_transform]$ **do**
  $start \leftarrow row \times tied\_weights$;
  if $row < (n\_transform - 1)$ **then**
    $end \leftarrow start + tied\_weights + n_l$;
  else
    $end \leftarrow start + tied\_weights$;
  end
  $Y[start : end] \leftarrow$
   $\sum_{i,j} K_{row}[i, j] * X_{row}[x - i, y - j]$;
end

The second stage is to refine the representation learning of optical flow in spherical domain. The intermediate refinement process is all about end-to-end training of the transform network. The training process is supervised as we want to make sure our network learn the actual representation. The problem with this scheme is that we do not have labelled dataset. Core part of the intermediate refinement stage is to use data augmentation techniques to convert labelled data, both images and optical flow in a spherical domain. We show sample augmented image sequences and corresponding optical flow in Fig.3.

Equirectangular plane is expressed from $(-\pi, \pi)$, $(-\pi/2, \pi/2)$ for length and height respectively, leading the aspect ratio of $length : height = 2 : 1$. We resize our original image and optical flow with the nearest interpolation scheme to maintain required aspect ratio. Then, we use simple projection techniques given by $(\phi = 2 \times \pi \times u, \cos\theta = 2 \times v - 1)$

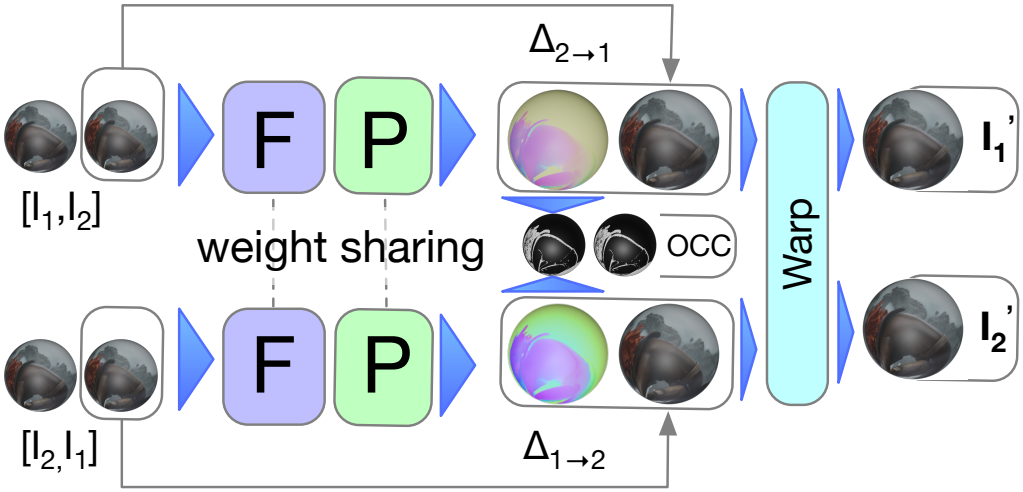

Fig. 5. Final Refinement process. Network from second stage is extended to have two parallel weight sharing architecture.

Algorithm 2: Spherical Data Augmentation

$\Delta_{I_1\to I_2} \leftarrow input();$
$(I_1, I_2) \leftarrow I \leftarrow input();$
$(h, w) \leftarrow dim(I_1);$
$(r_w, r_h) \leftarrow (\frac{h}{4\pi}, \frac{w}{2\pi});$
for *each* $i \in [-h/2, h/2]$ **do**
  $\Delta_u = \Delta_{I_1\to I_2}[i,:] \times \frac{2\pi\sqrt{r_w^2 - |r_w - i|^2}}{w};$
end
for *each* $j \in [-w/2, w/2]$ **do**
  $\Delta_v = \Delta_{I_1\to I_2}[:,j] \times \frac{2\pi\sqrt{r_h^2 - |r_h - j|^2}}{h};$
end
$\Delta'_{I_1\to I_2} \leftarrow \omega(\Omega(\Delta));$
$I' \leftarrow (I'_1, I'_2) \leftarrow \omega(\Omega(I));$

Algorithm 3: Boundary Condition

$(g_\theta, g_\phi) \leftarrow ([-180, +180], [-90, +90]);$
$G \leftarrow mesh\_grid(g_\theta, g_\phi);$
$(\tilde{\Delta}_u, \tilde{\Delta}_v) \leftarrow \tilde{\Delta} \leftarrow G + \Delta_{1\to2};$
$\tilde{\delta_u} = \frac{\tilde{\Delta_u}}{|\tilde{\Delta_u}|}(|\tilde{\Delta_u}| - 360);$
$\tilde{\delta_v} = \frac{\tilde{\Delta_v}}{|\tilde{\Delta_v}|}(180 - |\tilde{\Delta_v}|);$
$$\tilde{\Delta}_u = \begin{cases} \tilde{\Delta}_u, & \tilde{\Delta_u} \in [-180, 180] \\ -\tilde{\Delta}_u, & \tilde{\Delta_v} \notin [-90, 90] \\ \tilde{\delta_u}, & \tilde{\Delta_u} \notin [-180, 180] \end{cases};$$
$$\tilde{\Delta}_v = \begin{cases} \tilde{\Delta}_v, & \tilde{\Delta_v} \in [-90, 90] \\ \tilde{\delta_v}, & \tilde{\Delta_v} \notin [-90, 90] \end{cases};$$
$\tilde{\Delta} \leftarrow (\tilde{\Delta}_u, \tilde{\Delta}_v);$

for unit sphere to perform forward projection (i.e., perspective to spherical projection) followed by restoration using backward projection (i.e., spherical projection to backward projection).

As we discussed, issues regarding projecting perspective optical flow directly into sphere requires a correction factor. We apply this correction factor separately for u and v component of original perspective flow. The idea behind these factors is to scale displacement magnitude to be fair all over the points in spherical representation. For example, u is corrected by scaling each row with the ratio of central circumference (corresponding to the actual width w in perspective plane) and circumference $w_i = 2\pi r_i$ at each row. Regarding calculation of r_i, see Fig.4, where radius of a pixel-row i at distance $h_i = |R - i|$ from center can be calculated using simple law of triangle $r_i^2 = R^2 - h_i^2$ where $R = \frac{w}{2\pi}$, where $i \in (-R, R)$. We finally define function $\Omega_{(x,y)}$ to perform perspective to spherical projection, $\omega_{(r,\theta,\phi)}$ to perform back projection and ζ as correction function. Now we present image augmentation I as $I' = \omega(\Omega(I))$ and optical flow Δ as $\Delta' = \omega(\Omega(\zeta(\Delta))$. We present spherical data augmentation algorithm in Algorithm-2.

The training objective is to minimize end point error $||\Delta' - \Delta''||$ between predicted flow Δ'' and ground truth flow Δ' in conjunction with brightness error $||(I'_1 + \Delta'_{I_1\to I_2}) - I''_2||$ between the warped image and source image. We follow routine prescribed by [9] to train our network, but we limit our training process to significantly fewer amount of epochs compared to original implementation, as this is only a refinement process and network plateaus in terms of error rate. Our model is now ready to cope with the spherical domain, but we need to adapt our model to real-world data. To adapt our model to real-world data, we move into ultimate refinement stage.

C. Stage 3: Final Refinement

We replicate our initial network from stage-2 into two channel siamese network as shown in Fig. 5 to estimate forward $\Delta_{1\to2}$ and backward $\Delta_{2\to1}$ flow. We use forward and backward flow to estimate occlusion $\tilde{O} = (\tilde{O}_{2\to1}, \tilde{O}_{1\to2})$

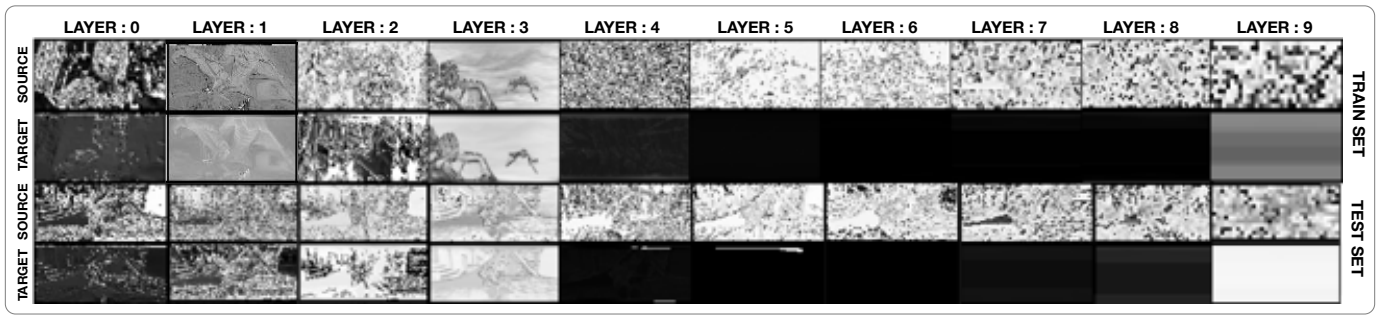

Fig. 6. Showing randomly picked output individual channel from output of different layers in source and target(stage 2) architectures. First two rows represent source source and transform CNN respectively on train set, similarly second two rows represent test set.

using Eq.5 where $\epsilon \approx 10^{-2}$, $(i, j) = (1, 2)$ for forward flow and vice versa.

$$M_i = \begin{cases} 0 \\ 1 \quad if, |\Delta_{i\to j}| \leq \epsilon \end{cases}$$
$$\tilde{O}_{i\to j} = M_i \odot ((1 - M_j) + \tilde{O}_{j\to i}) \tag{5}$$
$$L_p = \sum_{i,j} \frac{\sum \psi(I_i - I'_i) \odot (1 - O_{i\to j})}{\sum 1 - O_{i\to j}}$$

Similarly, we use predicted optical flow to warp target image. Apart from traditional warping techniques, we modify warping technique as shown in Algorithm-3. This warping technique is necessary to address the continuous nature of 360 images, i.e., whenever pixel displacement occurs beyond the boundary condition the pixel is displaced somewhere within the equirectangular plane. For example, if a pixel is displaced beyond the right boundary, the pixel will be displaced on the left side of the equirectangular plane. This is not true with perspective flow, where we consider this as a boundary condition and put the pixel into boundary. This is well preserved and more accurate assumption for smaller displacement in the border area.

We present final refinement process as further training steps to adapt to the target domain. We use dataset from our ongoing work Egok360, an egocentric activity recognition dataset for 360 videos as target dataset. The training process is self-supervised based on photometric loss as shown in Eq.5 where $\psi = (|x| + \epsilon)^q, \epsilon \approx 10^{-2}, q \approx 1 \times 10^{-1}$.

IV. Results

We present our result mainly on augmented Sintel [16] dataset, which we termed as Sintel360. We performed spherical data augmentation on original Sintel training set, which we divided into 9:1 train-val set. This train-val set has ground truth optical flow information. We compared 4 different models as shown in Table I using commonly used end point error (EPE) metrics using validation set. To make comparision fair, we augmented original sintel test set. We used this test set to compute photometric loss (L_p), defined in Eq.5.

Quantitative Results. Table I summarizes our experiments. We found that exhaustive layer replacement task is unnecessary. The convergance rate dramatically decreases as we go deeper, as shown in Table I, EPE is significanly higher ($\geq$ 17) for more than 4 layers replacement. This creates a domino effect, which propagate errors in subsequent layers. We illustrate this effect in Fig.6. We can see that beyond layer 4, the output of transformed layers are different. Instead of reproducing the source CNN these layers learn nothing even after training for 30 epochs, using same techniques that was used to train previous layers.

TABLE I
RESULTS ON SINTEL360 DATASET.

Model	Data	#Layers	EPE	L_p^*
LiteFlowNet[9]	Sintel360	0	$\sim$ 6.35	$\sim$ 1.30
LiteFlowNet +[14]	Sintel360	> 4	$\geq$ 17	$\geq$ 3.06
Ours, Stage-2	Sintel360	4	$\sim$ 6.35	$\sim$ 0.70
Ours, Final	Sintel360	4	$\sim$ 3.95	$\sim$ 0.60

Our model on Stage 2 performs on par with original implementation. Though original method seems fine, representation for optical flow in spherical video is not fair. We can explain the lower EPE on the original model with the large number of flow information correspondence between real and augmented data in central region of equirectangular plane. With a final refinement stage, we improve our model significantly bringing EPE to 3.95 from 6.35 on val-set and photometric loss from 0.70 to 0.60 on a test set.

Qualitative Result. Fig.7 shows qualitative results from our experiment compared with baseline LiteFlowNet. We also present qualitative results on our target 360 video dataset. To understand the fairness of the flow predicted, we used flow information to predict the next frame. We observe that the warping of flow preserves the spherical nature. In another word, it preserves the artificial artificat we introduced in original dataset (please note patches in different colors in target dataset shown in Fig.7). However, there are cases where none of these models work as expected. We show such case in the last row of estimated optical flow on target dataset. We believe this can be improved further by allowing the model to have longer training times with further hyperparameter exploration.

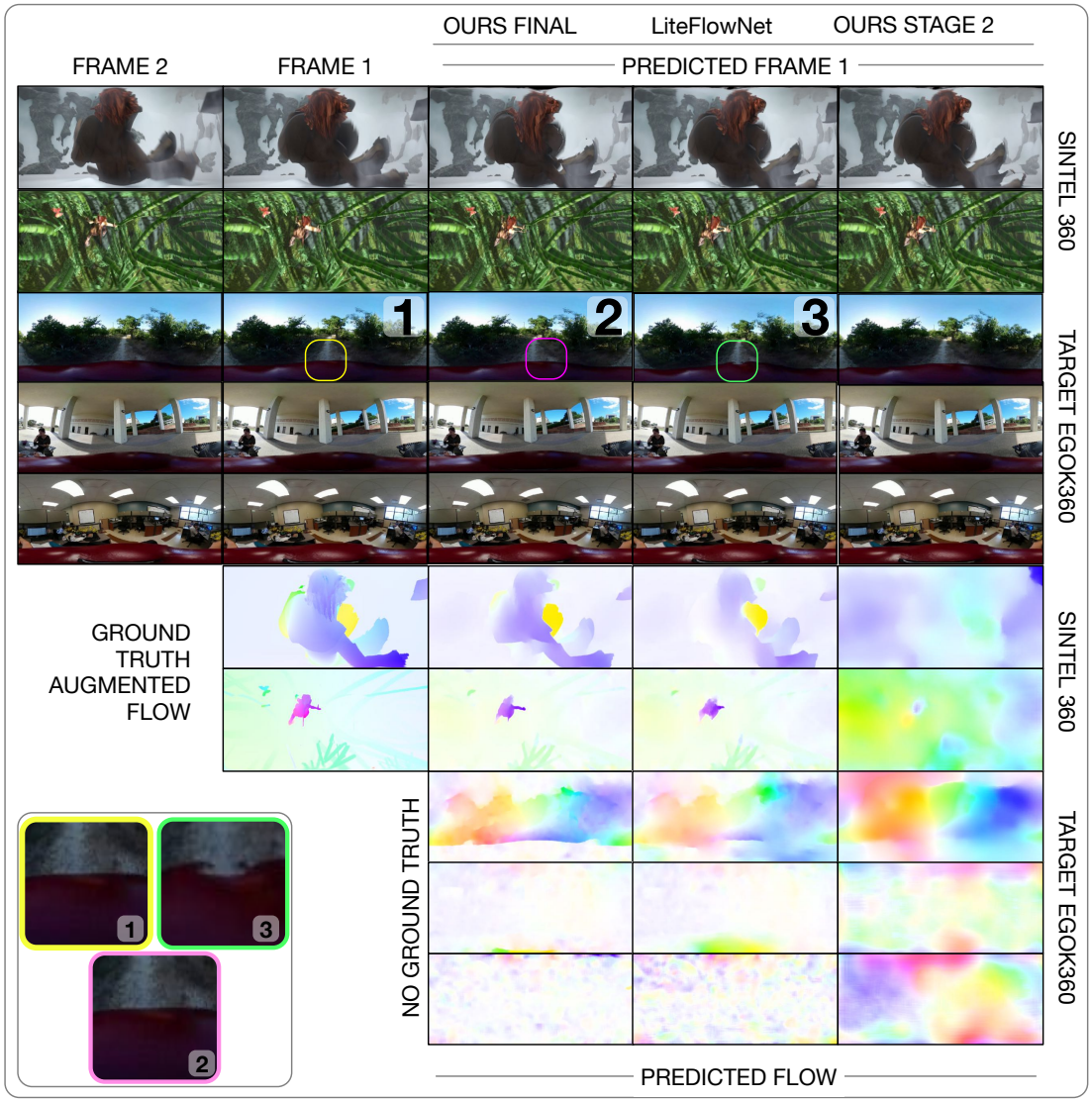

Fig. 7. Qualitative results on augmented Sintel 360 dataset and target video dataset. First two row represents randomly picked frames and second two row represents corresponding optical flow information. We predict frame-1 using forward flow from each architecture. We randomly pick patches from same location from predicted(patch-2,patch-3) and ground truth(patch-1) frame-1 as shown in bottom left corner. Patch 2,3 are from liteflownet360 and liteflownet respectively. We can see liteflownet360 results are comparatively better. Note: We encourage digital reader to zoom in for detail view.

V. Conclusion

In this paper, we presented a novel framework for 360 optical flow estimation, dubbed as LiteFlowNet360. This framework is an adaptation of existing best practices from both of the world, "optical flow estimation for perspective videos" and "spherical convolution for 360 videos/images". We presented our work as three major subsequent stages, transformation stage, intermediate refinement stage and final refinement stage. We started with the process of transformation, which includes evolutionary learning of spherical convolution based on transformer network. Apart from the success of these methods in other field, we empirically showed that exhaustive layer transformation from source to target CNN is insignificant in the context of optical flow estimation. We present second stage to address the correct representation of 360 flow. This stage requires further training as a refinement task. To train our model, we introduce a lossy data augmentation techniques to exploit existing labelled datasets. This technique allowed us to introduce artifacts related to spherical distortion in perspective videos, meanwhile transforming optical flow information in a spherical domain. We presented final stage as a domain transfer stage, where we use unlabelled target 360 video data to train our model in a self-supervised manner. Empirical and qualitative results showed the potential of this work. We believe this work will inspire others to investigate this area of optical flow estimation.

Acknowledgements: This research was partially supported by NSF CSR-1908658 and NeTS-1909185. This article solely reflects the opinions and conclusions of its authors and not the funding agents.

REFERENCES

[1] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360 videos," in *Proceedings of the European Conference on Computer Vision (ECCV), 2018*.

[2] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proceedings of the 26th ACM international conference on Multimedia, 2018*.

[3] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, "Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, 2017*.

[4] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze prediction in dynamic 360 immersive videos," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018*.

[5] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*. International Society for Optics and Photonics, 1981.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, 1981*.

[7] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," *arXiv e-prints, 2015*.

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition, 2017*.

[9] T.-W. Hui, X. Tang, and C. Change Loy, "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation," *arXiv e-prints, 2018*.

[10] Y.-C. Su and K. Grauman, "Learning spherical convolution for fast features from 360imagery," in *Advances in Neural Information Processing Systems 30, 2017*.

[11] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," *arXiv e-prints, 2018*.

[12] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) Equivariant Representations with Spherical CNNs," *arXiv e-prints, 2017*.

[13] B. Coors, A. P. Condurache, and A. Geiger, "Spherenet: Learning spherical representations for detection and classification in omnidirectional images," in *European Conference on Computer Vision (ECCV), 2018*.

[14] Y.-C. Su and K. Grauman, "Kernel Transformer Networks for Compact Spherical Convolution," *arXiv e-prints, 2018*.

[15] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning, 2009*.

[16] 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*, A. Fitzgibbon et al. (Eds.), Ed.

[17] P. Liu, M. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019*.

[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Deep end2end voxel2voxel prediction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2016*.

[19] A. Ahmadi and I. Patras, "Unsupervised convolutional neural networks for motion estimation," *arXiv e-prints, 2016*.

[20] J. Wulff and M. J. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015*.

[21] J. J. Yu, A. W. Harley, and K. G. Derpanis, "Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness," *arXiv e-prints, 2016*.

[22] D. Teney and M. Hebert, "Learning to Extract Motion from Videos in Convolutional Neural Networks," *arXiv e-prints, 2016*.

[23] A. Ranjan and M. J. Black, "Optical Flow Estimation using a Spatial Pyramid Network," *arXiv e-prints, 2016*.

[24] D. Gadot and L. Wolf, "PatchBatch: a Batch Augmented Loss for Optical Flow," *arXiv e-prints, 2015*.

[25] C. Bailer, K. Varanasi, and D. Stricker, "CNN-based Patch Matching for Optical Flow with Thresholded Hinge Embedding Loss," *arXiv e-prints, 2016*.

[26] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki, "Self-supervised learning of motion capture," in *Advances in Neural Information Processing Systems, 2017*.

[27] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, "Deep 360 Pilot: Learning a Deep Agent for Piloting through 360 Sports Video," *arXiv e-prints, 2017*.

[28] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, "Semantic-driven Generation of Hyperlapse from 360° Video," *arXiv e-prints, 2017*.

[29] W. Boomsma and J. Frellsen, "Spherical convolutions and their application in molecular modelling," in *Advances in Neural Information Processing Systems 30, 2017*.

[30] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, T.-L. Liu, and M. Sun, "Cube Padding for Weakly-Supervised Saliency Prediction in 360 Videos," *arXiv e-prints, 2018*.

[31] R. Khasanova and P. Frossard, "Graph-Based Classification of Omnidirectional Images," *arXiv e-prints, 2017*.

[32] O. Shakernia, R. Vidal, and S. Sastry, "Omnidirectional egomotion estimation from back-projection flow," in *Conference on Computer Vision and Pattern Recognition Workshop, 2003*.

[33] A. Radgui, C. Demonceaux, E. M. Mouaddib, D. Aboutajdine, and M. Rziza, "An adapted lucas-kanade's method for optical flow estimation in catadioptric images, 2008."

[34] A. Radgui, C. Demonceaux, E. Mouaddib, M. Rziza, and D. Aboutajdine, "Optical flow estimation from multichannel spherical image decomposition," *Computer Vision and Image Understanding, 2011*.

[35] C. Kirisits, L. F. Lang, and O. Scherzer, "Decomposition of optical flow on the sphere," *GEM-International Journal on Geomathematics, 2014*.

[36] B. Alibouch, A. Radgui, M. Rziza, and D. Aboutajdine, "Optical flow estimation on omnidirectional images: an adapted phase based method," in *International Conference on Image and Signal Processing, 2012*. Springer.