Learning Scale-Aware Optical Flow

Xiaoming Wen

Technische Universiteit Delft

TU Delft

This page intentionally left blank.

Learning Scale-Aware Optical Flow

by

Xiaoming Wen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday November 28, 2018 at 2:00 PM.

Student number:	4608682	
Project duration:	January 1, 2018 – November 28, 2018	
Thesis committee:	Prof. dr. M. Reinders,	TU Delft, responsible professor
	Dr. J. C. van Gemert,	TU Delft, supervisor
	Prof. dr. Alan Hanjalic,	TU Delft
	Dr. Marco. Loog,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

This page intentionally left blank.

Preface

This paper is the report of my master thesis project about the topic of estimating optical flow in presence of large scale change using convolutional neural network. This paper mainly contains a scientific paper in chapter 1, including the methods we proposed and the experiments on our datasets. The following other chapters provide the preliminary knowledge and supplemental details that help to better understand this work.

I would like to express my deepest appreciation to my supervisor Dr. J. C. van Gemert for his enthusiastic encouragement and patient guidance during the whole research process. I would like to express my gratitude to Dr. Silvia Pintea who are always ready to help and had given me a lot of advice. I truly appreciate the discussions I had with her on the research.

I am deeply grateful to my family for their continuous love and support. They selflessly encouraged me to be the best version of myself. Last but not least, many thanks go to Shuang Cheng and all my friends at TuDelft for supporting and accompanying me on this journey. Thanks for always being there for me.

Xiaoming Wen
Rotterdam, November 2018

This page intentionally left blank.

Contents

This page intentionally left blank.

1

Scientific Paper

Learning Scale-Aware Optical Flow

Xiaoming Wen
Delft University of Technology
Deflt, The Netherlands
X.Wen@student.tudelft.nl

Abstract

Optical flow is a representation of projected real-world motion of the object between two consecutive images. The optical flow measures the pixel displacement on the image coordinate plane. However, it does not reveal the motion in depth explicitly, which could be useful as input in some tasks such as vehicle tracking. To extend the original optical flow approach, we model the depth change of the object as the scale change of object in the image and present an approach to estimate the scale change and optical flow jointly. Considering the scenario that obvious scale change occurs between two images, the traditional convolution network fails because it lacks the scale invariance. According to the Scale-space theory and the idea of learning a combination of Gaussian derivative basis to approximate arbitrary filters, we build a Basis Convolution layer that allows the network to see the scale change between two images and make use of it to better capture the same feature with various scales on two images.

We test our models on our own optical flow datasets which involve obvious scale change. According to the experimental results, our method is capable of estimating the scale change between images, and it significantly improves the optical flow estimation by modelling the scale change explicitly.

1. Introduction

Optical flow estimation is an old problem in computer vision field and it has been widely used in many motion analysis tasks [11, 19, 20, 23]. Recently, the rise of convolution neural network(CNN) brings the optical flow estimation into new era. Many works have been introduced to estimate optical flow by training a network end-to-end [3, 9, 17, 22]. With the help of sufficient synthetic dataset, the CNN-based method reaches the state-of-art result on optical flow estimation problem [22]. However, the current CNN-based methods do not consider large scale change between images, which is frequently seen in the real world when large motion in depth occurs.

In the optical flow estimation problem, most of the CNN-based methods [3, 9, 15, 17, 22] rely on correspondences matching between images, and this matching behaviour is performed on the feature maps extracted from two images through the same convolution channel. In this feature extraction module, the convolution layer extracts the feature just by calculating the in-product of image batch and specific filter. In other words, the convolution layer is sensitive to scale change [8], which makes it hard to capture the same pattern with significant scale change in the next frame. Thus current CNN-based methods suffer degradation in performance on optical flow estimation in present of large scale change.

At the same time, the optical flow is the projection of real-word 3D motion on the image plane. Optical flow measures the pixel displacement between image pairs, however, it cannot directly tell the object motion in the direction of camera view. This motion in depth could be useful in motion analysis task involves large depth motion like on-board and handle camera analysis. It is obvious that the depth motion of rigid object brings scale changes in images. so the estimation of scale changes can be regarded as a supplement to optical flow, which implies depth motion between images.

In this paper, we explore a CNN-based model to estimate optical flow in presence of large scale change and to provide the scale change estimation from image pairs. Traditional convolution layer have poor scale-invariant property. To enable the network to perceive the scale change, we replace the conventional convolution layer with our Basis Convolution layer which is inspired by the idea of approximating filter through the linear combination of Gaussian derivatives [10][1]. The Basis Convolution layer learns the combinations of Gaussian derivative rather than the pixel weights in filters. If the scale change is known, the Basis Convolution layer is able to change the filter scale for each location in feature map by simply adjusting the scale of the Gaussian derivatives. In this way, the network is capable of capturing the same features appearing on images with different scales

and hence improve the optical flow estimation. Considering the strong relationship between optical flow and scale change in images, in our model, we add an additional output channel to estimate the scale changes, along with the optical flow.

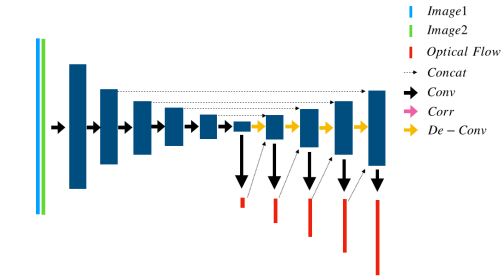
The availability of sufficient dataset is crucial for training the neural network in a supervised manner. The current optical flow datasets [3, 16, 18] is sufficiently large. However, they generally contain little scale change of object between images. Therefore, in this paper, we introduce our own optical flow datasets with scale change ground truth available. Each of our optical flow datasets contains 20k training images showing some random moving digits in front of a pure color background.

In this work, we have made the following contributions. i) We identify the performance degradation of current CNN-based method on optical flow estimation in presence of large scale change. ii) We evaluate the effectiveness of Basis Convolution layer which can perceive the scale change between images and efficiently adjust the filter scale for every pixel location. iii) We introduce a scale-aware model that estimates optical flow and scale change jointly from image pairs.

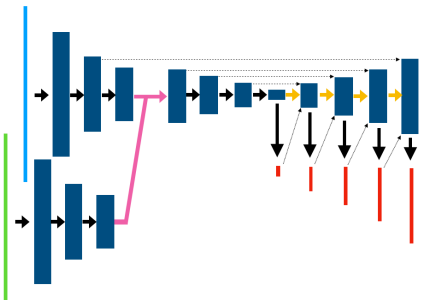
2. Related Work

2.1. Estimate Optical Flow with CNN

Traditionally, optical flow estimation is solved by optimizing an objective function that formularizes the constancy of some image properties and the local smoothness [7][25][21]. With the rise of the convolutional neural network(CNN) in computer vision field, recently, many efforts have been made to estimate optical flow with CNN. FlowNet [3] is the first CNN that directly learns the optical flow from two images. In this work, the author proposed two models, FlowNetSimple and FlowNetCorr. The difference between these two models is how they extract features from two images in the encoder part. In FlowNetSimple, two images input are concatenated as a six channels input, as shown in figure 1a. In FlowNetCorr, two images are first encoded separately through two shared-weight convolution channels, then the obtained feature maps are passed to a correlation layer to realize matching mechanism explicitly, as shown in figure 1b. After encoding, the rich feature representation of two images is input to the decoder part to provide optical flow estimations from coarse to fine. In the decoder part, several de-convolution layers of increasing size are used to upsample the feature maps. Several recent network architectures are proposed that surpass the FlowNet or even achieve state-of-art optical flow estimation [22]. Flownet 2.0 [9] connects multiple models in FlowNet by stacking, and get much better performance. SPy-net [17] and PWC-net [22] use the spatial-pyramid architecture that allows to refine the optical flow estimation at each pyramid level and greatly reduce the number of model parameters. In SPy-net and PWC-net, the researchers made use of the coarse optical flow estimation explicitly by warping the feature map with it. To avoid the occlusion problem in warping, The author in [15] calculated the cost volume from the feature map which is shifted by the pre-estimated optical flow. Despite the steady improvement of results, the extraction module in these models keeps the similar structure: passing two images through the same convolution channel separately and fusing the obtained two feature maps by a correlation layer. However, this weight-shared channel could fail when the features appearing with different scales in two images, which brings degradation in performance.



(a) FlowNetSimple.



(b) FlowNetCorr.

Figure 1: Network architectures of FlowNetSimple and FlowNetCorr. The dark blue squares are feature maps of different sizes. The blue and green squares on the left are two image inputs, frame 1 and frame 2, respectively. The red squares are the multi-scale optical flow estimations. Dark and yellow arrows denote convolution and deconvolution, respectively. Pink arrows represent the correlation operation. Dash lines imply how the feature maps are connected to others by concatenation. These two networks use similar encoder-decoder architecture, and they are capable of producing optical flow estimation at multiple scales.

2.2. Multi-Scale CNN

Recently, CNN has proven to be very successful in many computer vision tasks [5, 14, 24, 26]. A key advantage

of convolution layer over fully connected layer is its shift-invariance property due to the weight sharing at each location in the feature map. Unlike this architecture hard-coded shift-invariance property, CNN can hardly capture the same features with varying scales [6]. In last years, many efforts have been made to derive a multi-scale CNN and there are two main strategies. i) Single image multiple filters. This strategy utilizes multiple filters with different scales to capture features in a single image. Inception [24] used two sets of filters with different kernel sizes at each layer, learning the multi-scale filters in a brute force manner. In [28], the author derived multi-scale filters by transforming the canonical filter and proposed a network with predefined multi-scale filter columns. ii) Presenting the image to the network at multiple scales. Other works like [6, 12] use the fixed set of filter but provide multiple feature outputs for scaled images. Generally, these methods require multiple convolutions and resize operation on image or filter at different scales, leading to heavy computation and memory cost. Similar to single image multiple filters strategy, our network receives input at one scale and captures multi-scale features. However, unlike [28] who involves complex filter transformation in runtime, we obtain the filter of the desired scale by simply adjusting the scale of the Gaussian derivatives. In our work, this process is done offline which makes it more efficient.

3. Modelling the Scale in CNN

3.1. Scale-Space Theory and N-jet

Scale-space theory offers a model to represent the deep structure of an image by convolving the image with Gaussian filters of different scales [27]. The uniqueness of the Gaussian filter is that it will not introduce new structure in the signal after filtering [2]. For a image signal $f(x)$, its scale-space representation is defined as

$$L(x;\sigma) = g(x;\sigma) * f \tag{1}$$

where $g(x;\sigma)$ is the Gaussian kernel with scale σ, and $*$ represents the convolution operator. To obtain a local description of the signal in scale-space representation, we take the Taylor approximation around x_0 up to order N,

$$L(x;\sigma) = \sum_{n=1}^{N} \frac{L^n(x_0;\sigma)}{n!}(x - x_0)^n \tag{2}$$

Note that, due to the linearity of convolution, $L^n(x_0;\sigma) = g^n(x_0;\sigma) * f(x_0)$ and we have,

$$L(x;\sigma) = \sum_{n=0}^{N} \frac{g^n(x_0;\sigma) * f(x_0)}{n!}(x - x_0)^n \tag{3}$$

Equation (3) shows that to get the local approximation of an image at scale σ, we only need to convolve the original image $f(x)$ with the corresponding Gaussian derivatives. The set of responses to the derivative filters describing the local patch is called the N-jet [4].

3.2. Filters Approximation by Gaussian Derivatives

N-jet [4] shows how the complete base locally approximate any image, as it is explained in equation 3. Considering the filters in CNN as this local patch, we can derive the approximation of arbitrary filters in CNN as the linear combination of Gaussian derivatives up to order N [10].

$$F = \sum_{i=0,j=0}^{i+j=N} w_{i,j} * g^{i,j}(x, y; \sigma) \tag{4}$$

where $g^{i,j}(x, y; \sigma)$ is the Gaussian derivative in 2D form, i and j are the derivative order of x and y dimension respectively, and $w_{i,j}$ is the weight of corresponding Gaussian derivative. For the sake of simplicity, this set of weights of derivatives is denoted as α. It is worth noting that this filter approximation disentangles filter's shape and scale. More specifically, the parameter α determines the shape of the filter and parameter σ determines the scale separately, as shown in figure 2.

When utilize this filter approximation in the convolution layer, the layer tries to optimize the weights of Gaussian derivatives, unlike the traditional convolution layer that directly learns the pixel weights of the filters.

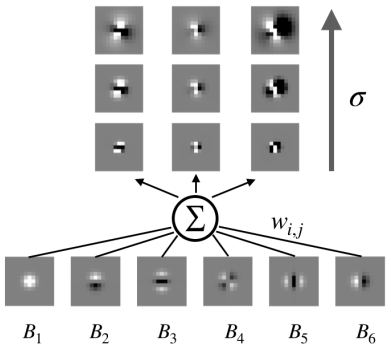


Figure 2: Linear combination of Gaussian derivatives up to order 2. σ and $w_{i,j}$ determine the shape and scale of the filter respectively. The figure is adopted from [1].

4. Scale-Aware Basis Convolution Layer

To solve the problem that the convolution layer cannot capture the same pattern with various scales, we introduce our Basis Convolution layer that can efficiently adjust the scale of filters for every location in the image.

4.1. Implementation of Basis Convolution Layer

The idea of using the linear combination of Gaussian derivatives to approximate arbitrary filters [10][4] enables us to easily model the scale of the filter as the width of its Gaussian derivative. By simply adjusting σ, one can obtain filter with arbitrary scale, surviving from complex filter transformation [28].

Furthermore, the filter approximation through Gaussian derivatives disentangles the scale and shape of the filter. Therefore, we can calculate the derivative basis with the possible discrete scales before the convolution and construct a derivative basis bank. To construct this basis bank, the continuous scale is discretized into N discrete values and we calculate the Gaussian derivatives of these discrete scales up to a predefined maximum order. Normally, the scale change between two frames is limited. A comparatively small N can easily meet the precision requirement in scale discretization.

Apart from the feature map input, this Basis Convolution layer accepts additional scale map input. Before doing the convolution, we calculate the desire scale for each position by multiplying the default scale with the input from the scale map. Then the derivative basis with the closest scale is fetched from the basis bank to construct the filter, rather than calculating the derivative basis of proper scale in runtime. This precalculated derivatives bank greatly reduces the computation burden of calculating unique Gaussian derivatives for tons of locations, which makes the Basis Convolution layer practical. Figure 3 shows the details of the Basis Convolution layer.

4.2. Scale-Invariant Property

The pattern that a filter wants to capture is fixed by the weights of Gaussian derivatives, the α, as it is shown in equation (4). By changing σ, we expect the filter to capture the same pattern but being downscaled or upscaled. Therefore, the filters must maintain the invariant-by-scaling property as it is discussed in [28]. Suppose we have a filter F of scale σ that detects specific feature in image I by the convolution $\text{conv}(I, F(\sigma))$. S is the resize operation and $S(I, k)$ represents the scaled image I by a factor k. To capture the scaled feature in image $S(I, k)$, we adjust the scale of filter F to $k\sigma$, generating the convolution result $\text{conv}(S(I, k), F(k \times \sigma))$. These two convolution results should be equivalent, except being scaled by the factor k, and this relationship is formalized as

$$S(\text{conv}(I, F(\sigma)), k) = \text{conv}(S(I, k), F(k \times \sigma)) \quad (5)$$

In other words, this invariant-by-scaling property means when the scaled filter captures the feature which is scaled by the same factor, the convolution should give the same activation value in the output feature map. However, it is known

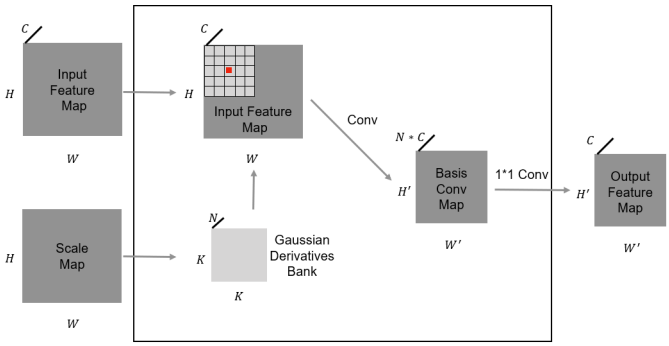


Figure 3: An illustration of the Basis Convolution layer. Unlike the traditional convolution layer, it takes an additional Scale Map as input, which denotes the scale change at each position in the feature map input. Before doing the convolution, it receives the scale change information of each position and chooses the Gaussian derivative basis with proper sigma from the derivatives bank. After convolving the feature map with the Gaussian derivatives, a traditional 1×1 convolution is utilized to provide the linear combination of these activations. And the learnable parameters in this depth-wish convolution are the weights in equation 4. The Basis Convolution layer is implemented on CUDA for speed up.

that the amplitude of Gaussian derivative decreases exponentially with scale [13], which does not hold this property. To solve this problem, we normalize the Gaussian derivatives using γ-normalized method introduced by Lindeberg [13], as shown in figure 4. For the sake of simplicity, we choose the γ equals to 1. In this case, the gamma-normalized is equivalent to L_1 normalization. Finally, we can write the filter as a weighted sum of normalized Gaussian derivatives in equation 6.

$$F = \sum_{i=0,j=0}^{i+j=N} w_{i,j} * g^{i,j}_{L1-norm}(x, y; \sigma) \quad (6)$$

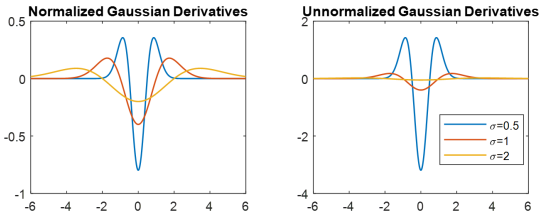


Figure 4: An example of the second order Gaussian derivatives with different σ. After normalization the amplitude is inversely proportional to σ, and the invariant-by-scaling property is preserved

Other implementation details of the Basis Convolution layer is available in the Appendix.

5. Proposed Networks

In this section, we propose several networks to achieve the goal of estimating the scale-aware optical flow and scale change from images.

5.1. Baseline Method

FlowNetCorr [3] has a simple network architecture and it is proved to be effective in optical flow estimation problem [3]. Furthermore, it implements a two-channel extraction module, which is generally used in CNN-based methods of estimating optical flow. So, we derive our baseline method from FlowNetCorr by deleting some layers of it to fit our own dataset. This baseline method is a light version of FlowNetCorr, and thus we call it FlowNet_Light. More details are provided in the Appendix.

5.2. Replacing Traditional Convolution Layer

To enable the network to perceive the scale change between images, we replace the first three traditional convolution layers in the FlowNet_Light with our Basis Convolution layers and obtain the network called FlowNet_Basis.

The FlowNet_Basis accepts additional scale map input to adjust the scale of filters in two channels. The channel for frame 2 is treated as the reference channel. It accepts a constant scale map of ones and convolves the input feature map with filters of the default scales. If the scale ratio of frame 1 to frame 2 is available, the channel 1 can adjust the filter scale for every pixel location accordingly. In this way, the features with various scales in two images are correctly captured by two channels. The network architecture of FlowNet_Basis is shown in figure 6.

The encoder channel consists of three convolution layers with stride 2, which means the size of feature map is reducing as the representation level goes up, as shown figure 5. So, we downsample the scale map through bilinear interpolation to fit the reducing size of feature map.

Training Loss. We apply the same multi-scale training loss introduced in FlowNet [3]. It is a weighted sum of the average endpoint errors defined on multi-scale optical flow estimations. As an adjustment for our own dataset, we separate the loss on background and foreground and normalize them with the ground truth optical flow range, as it is shown in equation 7.

$$Loss_F = \sum_i W_i \frac{AEE\_F_f^i + AEE\_F_b^i}{\max_x(F^i(x)) - \min_x(F^i(x))} \tag{7}$$

$$AEE\_F_f^i = \underset{x}{avg}(|F_f^i(x) - \hat{F_f^i}(x)|_2) \tag{8}$$

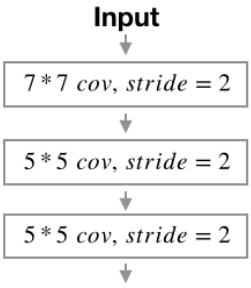


Figure 5: An illustration of the feature extraction of FlowNet_Light.It contains three convolution layers and each layer have stride of 2.

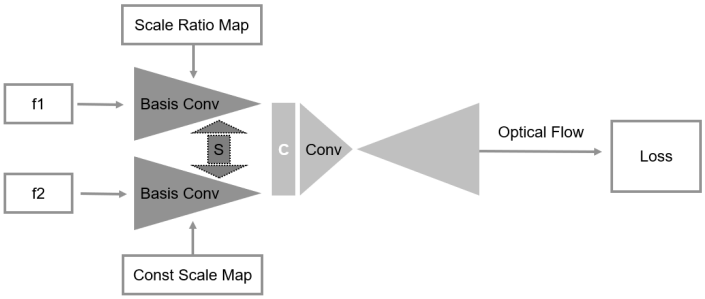


Figure 6: Network architecture of FlowNet_Basis. Its network architecture is the same as FlowNet_Light, except that the first three traditional convolution layers are replaced by the Basis Convolution layers. f1 and f2 represent two input frames. S denotes that two convolution channels share the α parameter. C represents the correlation layer. The triangles represent the encoder or decoder. The second convolution channel is the reference channel, and channel 1 accepts scale ratio map ground truth to adjust the scale of the filters.

$$AEE\_F_b^i = \underset{x}{avg}(|F_b^i(x) - \hat{F_b^i}(x)|_2) \tag{9}$$

$F^i(x)$ is the i^{th} scale level flow field with location variable x. The subscript f and b denote foreground and background, respectively. $AEE\_F_f^i$ and $AEE\_F_b^i$ represent the average endpoint error of foreground and background optical flow at i^th scale level. $|\cdot|_2$ represents the L_2 norm. The parameter W_i is the weight for multi-scale flow estimations. We assign higher weight to the estimation of the finest scale level, and the values for W_i are [1, 1, 1, 2], from coarse to fine.

5.3. Estimating the Scale Change

The Basis Convolution layer is capable of adjusting the scale of filter for each location by receiving the scale ratio map from frame 1 to frame 2. However, in the optical

flow estimation problem, this scale ratio map is generally not available and the only input to the model is the image pairs. So we propose a network that estimates the scale ratio of object directly from the image pair and makes use of it to better capture feature with various scales in two frames. Since it estimates flow and scale ratio jointly, we called this model FlowNet_FlowScale.

The scale ratio estimation and optical flow estimation both rely on feature extraction and feature matching mechanism, so we follow the similar encoder-decoder architecture in FlowNet_Basis. The scale change and optical flow between two frames have a strong relationship because they are all the observations of object motion from different perspectives. So we add an additional channel in output to give the dense scale ratio estimation, and the optical flow and scale ratio estimation can be refined jointly in the training process. Now, the prediction output has three channels. Two channels for the 2D flow field and one channel for the 1D scale ratio map.

The FlowNet_FlowScale is composed of two sub-networks. The estimation of scale ratio from the first network is presented to the next sub-network. By receiving the scale ratio input, the second network is expected to give better flow estimation. The network architecture of FlowNet_FlowScale is shown in figure 7.

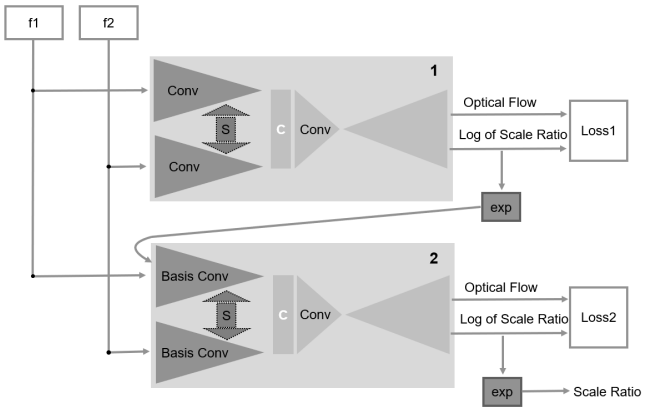


Figure 7: An illustration of the architecture of FlowNet_FlowScale. Basically, It is formed by stacking FlowNet_Light and FlowNet_Basis. By adding an extra channel in the output, these two sub-networks produce both flow and scale ratio estimation. The output scale ratio from the first sub-network is input to the Basis Convolution layer in the second sub-network, helping to capture features with different scales in two images. Note that an exponential function is used to make sure that the predicted scale ratio is within range (0, ∞). The loss for the scale ratio estimation is defined on the its logarithm.

Applying Exponential Function. The Scale Ratio Map is calculated as the ratio of the object's scale at the first frame to its corresponding at the second frame. Its value is within range (0, ∞). As shown in figure 7, the Scale Ratio Map output of the first network is the input to the second network, and during the training phase the value of Scale Ratio is unpredictable and does not likely fit the physical range of scale ration, which is especially true for non-ReLU activation. To avoid this problem, we apply an exponential function on the network's scale ratio output. It means the network is trying to estimate the logarithm of ground truth scale ratio. Since the scale ratio between two frames would typically locate in a small range like 0.5 to 2, so the scale ratio should be updated smoothly and a small base, 1.1, is used in the exponential function, as it is shown in figure 8.

What is more, the mapping from depth change to scale ratio is not linear. A traditional regression loss directly defined on scale ratio would be biased in favour of the estimation accuracy of scale ratio which is bigger than 1 and overlook those smaller than 1, while the loss defined on the logarithm would treat them equally.

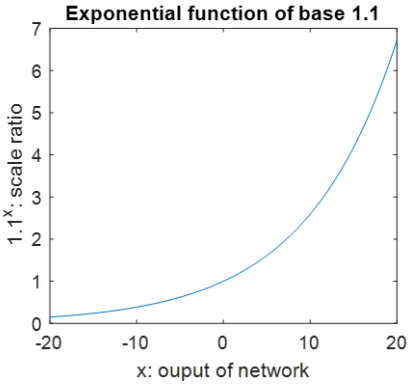


Figure 8: The effectiveness of applying exponential function to network's output, it limits the arbitrary network's output value to range (0, ∞).

Training Loss. Compared to FlowNet_Basis, the model FlowNet_FlowScale produces additional scale ratio estimation. Like the training loss for optical flow in equation 7 we define a similar training loss for the scale ratio estimation, as it is shown in equation 10

$$Loss_S = \sum_i W_i \frac{AEE\_S_f^i + AEE\_S_b^i}{\max_x(S^i(x)) - \min_x(S^i(x))} \tag{10}$$

$$AEE\_S_f^i = \underset{x}{avg}(|S_f^i(x) - \hat{S}_f^i(x)|_2) \tag{11}$$

$$AEE\_S_b^i = \underset{x}{avg}(|S_b^i(x) - \hat{S}_b^i(x)|_2) \tag{12}$$

The denotations in this equation are the same as equation 7, except that S denotes scale ratio estimation. The training loss of one sub-network combines these two loss terms,

$Loss_F$ and $Loss_S$. Finally, the sum of losses in two networks forms the final training loss for the whole model.

6. Training Datasets and Details

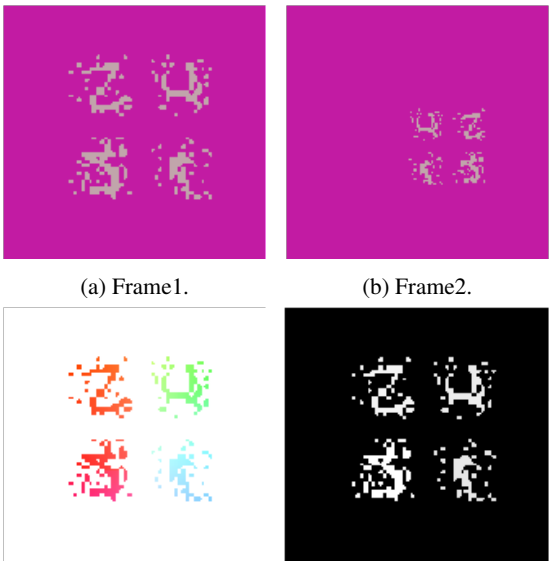
6.1. Training Datasets

It is hard to obtain the large optical flow dataset of the scene in the real world, since the dense displacement motion field has to be measured for every pixel. Recent work on optical flow estimation using learning method heavily relies on synthetic datasets [3, 16], where the motion field is easy to generate in picture synthesis. However, in general, there is no obvious scale change between images in these datasets. Therefore, to evaluate our methods, we create our own datasets with obvious scale change. These datasets show some random translation of digits in front of a pure background. Apart from the position, the size of digits in the next frame can also be quite different, indicating the motion toward or backward the camera. All the digits are randomly sampled from MNIST dataset but with heavy noise applied to them. For the detailed evaluation of our methods, we create 3 datasets considering different scenarios, and each dataset contains 20k training samples and 4k testing samples.

OF_Big dataset. This dataset considers a scenario where digits become far away from the camera. The image has a size of 192×192 , and it contains a group of four digits in the matrix arrangement. All the digits have the same colour differing from the background. After the translation which is uniformly distributed from -20 to 20, the digits appear in another position in the next frame and the arrangement of digits in the matrix is also shuffled. Because the digit moves further away from the camera, its size in the first frame is bigger than that in the second frame. Each digit in the same group has the similar size change. The size ratio of the digit in frame 1 to frame 2 is randomly distributed from 1.65 to 2.42. Figure 9 provides and example of OF_Big dataset.

OF_Small dataset. This dataset is very similar to the OF_Big dataset, but it assumes the group of digits moves closer toward the camera, resulting smaller digits in the first frame. This dataset can be treated as a reverse version of OF_Big dataset by simply swapping frame 1 and frame 2. The scale ratio of frame 1 to frame 2 now is randomly distributed from 0.4 to 0.6.

OF_Normal dataset. This dataset does not contain obvious scale change in images. Apart from the obvious scale change, the other settings are the same as the OF_Big and OF_Small dataset. Figure 10 provides an example of two frames in OF_Normal dataset. The scale ratio from frame 1 to frame 2 is randomly distributed from 0.9 to 1.1.



(a) Frame1. (b) Frame2. (c) Optical flow. (d) Size ratio.

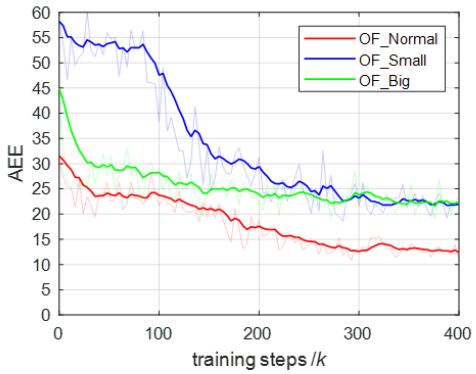
Figure 9: Example of OF_Big dataset, the size ratio is calculated as the ratio of digit in frame 1 to frame 2. In this paper we use the common Hue Saturation colour space for flow visualization. More details are in the Appendix.

(a) Frame1. (b) Frame2.

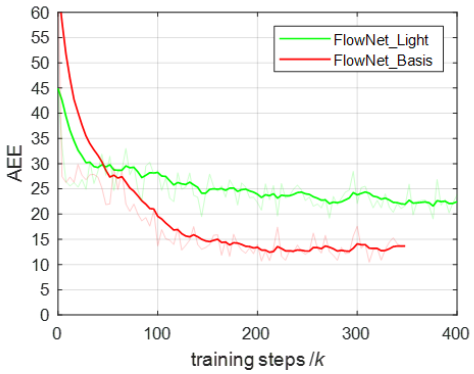
Figure 10: Example of OF_Normal dataset. The digits stay almost the same size in the next frame.

6.2. Training Details

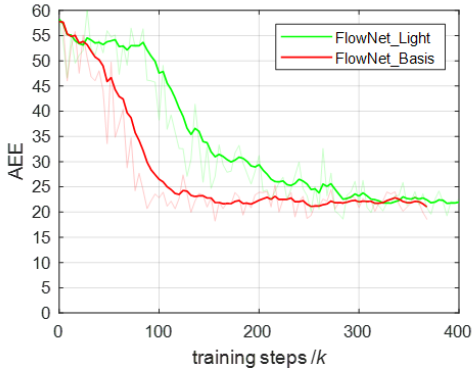
In our experiments, even though the size of training samples is quite large, we found that data augmentation is very crucial to avoid overfitting. We implement the same data augmentation process as the original FlowNet [3]. The data augmentation is performed online. During train phrase it applies random translation, rotation, zooming and noise to the image pairs. Not only the variety of images, the variety of flow is also achieved by adding extra transformation to frame 2 [3]. To further improve the generalization of our methods, we implement dropout layer of possibility 0.7 in the first three convolution layers. We used the Adam optimizer with β_1 and β_2 equal to 0.9 and 0.999 respectively.



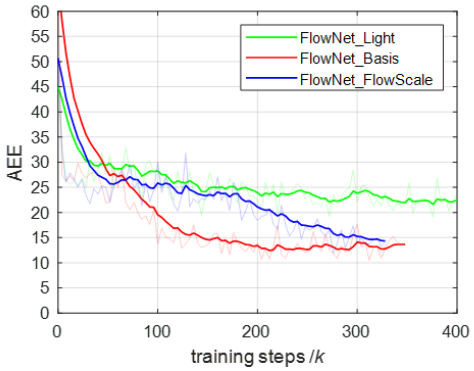
(a) Ex1: Testing accuracy of FlowNet_Light on different datasets.



(b) Ex2:Testing accuracy of FlowNet_Light and FlowNet_Basis on OF_Big datasets.



(c) Ex2:Testing accuracy of FlowNet_Light and FlowNet_Basis on OF_Small datasets.



(d) Ex3: Testing accuracy of three models on OF_Big datasets.

Figure 11: Experimental results of optical flow estimation. (a) shows that on the dataset with large scale changes, the FlowNet_light suffers severe degradation performance. (b) shows that on OF_Big dataset, FlowNet_Basis significantly outperforms FlowNet_Light. (c)shows that on OF_Small dataset, FlowNet_Basis has similar performance with FlowNet_Light. (d) shows the two scale-aware networks have better performance than the original FlowNet_Light on OF_Big dataset.

The learning rate is initialized as 0.0002 and becomes two times smaller after 100k steps.

Weight Decay The weight decay is set to be 0.0001 for all learnable parameters in FlowNet_Light. However, the FlowNet _Basis and FlowNet_FlowScale contain Basis Convolution Layer whose variables, α, are quite different from the other layers. In filter approximation, the α is multiplied by the L_1 normalized derivatives that have fairly small constant values. So to construct an effective filter the Basis Convolution layer requires much bigger parameter than the other layers. In other words, the regularization on α should be smaller than other parameters. In our experiment, we found this special adjustment of weight decay parameter is decisive to train the our model successfully. The weight decay of α we use is about 30 times smaller than conventional parameters.

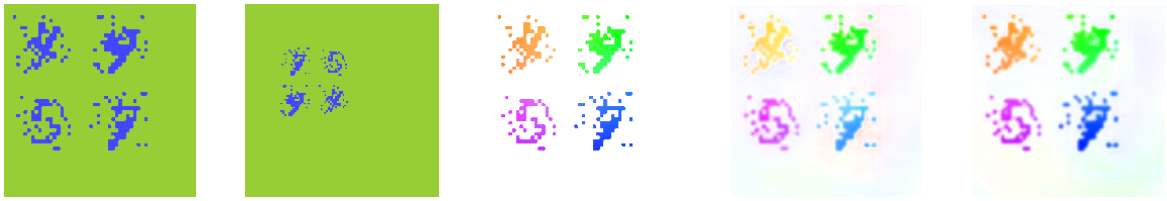
7. Experiments and Results

7.1. Ex1: Evaluating baseline method

In this experiment, we train the FlowNet_Light with OF_Normal, OF_Small and OF_Big dataset separately, and figure 11a shows the optical flow average endpoint error of digits in the testing phase. The errors on OF_Big dataset and OF_Small dataset are significantly higher than the error on OF_Normal dataset, which proves that the current architecture with two weight-shared extracting channels suffers from performance degradation in presence of large scale change.

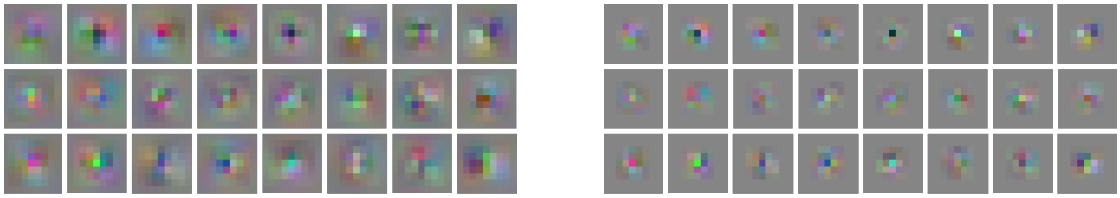
7.2. Ex2: Evaluating of Basis Convolution layer

To enable the network to perceive the scale change between images, we propose our Basis Convolution layer and create the network called FlowNet_Basis that accepts



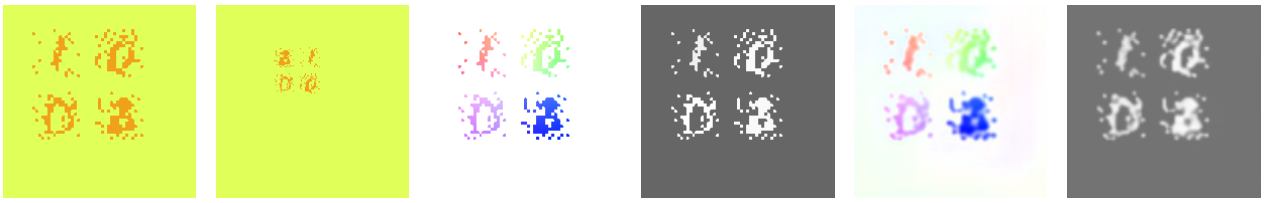
(a) Frame 1. (b) Frame 2. (c) Optical flow GT. (d) FlowNet_Light. (e) FlowNet_Basis.

Figure 12: Comparison of FlowNet_Basis between FlowNet_Light on OF_Big datasets. The flow estimation of FlowNet_Basis is better than FlowNet_Basis.



(a) Learned Filters of default scales. (b) Filters that downscaled by the factor 2.

Figure 13: Visualization of learned filters of the first convolution of FlowNet_Basis which is trained on OF_Small datasets. The filters are not smooth, and the downscaling introduces obvious distortion due to the discretization error. The full set of filters is available in the Appendix.



(a) Frame 1. (b) Frame 2. (c) Optical flow GT. (d) Scale ratio GT. (e) Pred Flow. (f) Pred scale ratio.

Figure 14: An example of estimated result of FlowNet_FlowScale on OF_Big dataset. (e) and (f) show the predicted optical flow and scale ratio respectively. It is able to produce correct scale ratio and optical flow estimation.

additional scale map ground truth input. In this experiment, we train the FlowNet_Basis on our OF_Big and OF_Small dataset separately and compare its performance with FlowNet_Light.

According to the estimation error in the testing phase, shown in figure 11b, the FlowNet_Basis significantly outperforms FlowNet_Light on OF_Big dataset, which proves the effectiveness of Basis Convolution layer. The estimated result of a random test example is provided in figure 12. More testing examples are available in the Appendix.

Unfortunately, these two networks have similar performance on OF_Small dataset, as it is shown in figure 11c. The Basis Convolution layer is not effective when it tries to adjust the filters with smaller scale. The reason might due to the information loss in the downscaling operation, especially for filter with small kernel size, as it is shown in figure 13.

7.3. Ex3: Evaluating final network

In this experiment, we train our final Network FlowNet_FlowScale on the OF_Big dataset. Unlike FlowNet_Basis that requires scale map ground truth as input, FlowNet_FlowScale estimates the scale map by itself. Figure 11d shows the flow error comparison of FlowNet_FlowScale, FlowNet_Light and FlowNet_Basis. Even though FlowNet_FlowScale is slightly worse than FlowNet_Basis, it still shows a big improvement over FlowNet_Light. Figure 15 shows the average endpoint error of scale ratio estimation. The scale ratio estimation is not quite accurate and the error is about 0.4. That could be the reason why FlowNet_FlowScale performs worse than FlowNet_Basis who accepts the scale ratio ground truth. Figure 15 shows the result of a random testing example of FlowNet_FlowScale. More examples are available in the Appendix.

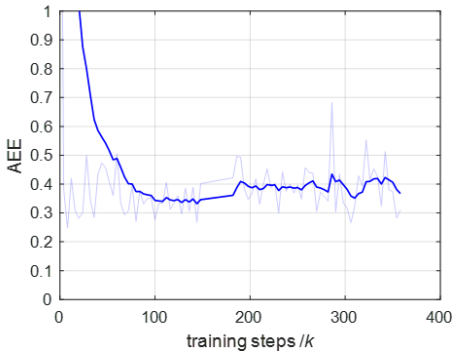


Figure 15: The average endpoint error of scale ratio estimation from FlowNet_FlowScale trained on OF_Big dataset. The estimation error is about 0.4, which is not quite accurate.

8. Conclusion and Future Work

In this work, we propose the Basis Convolution layer that can introduce the scale-ware capability to the network, and it can be easily integrated by replacing the traditional convolution layer. By using this Basis Convolution layer, we construct a network that focuses on the optical flow estimation in presence of scale change. The experiment shows that it provides much better optical flow estimation result and correct scale change estimation directly from image pairs.

However, this Basis Convolution layer suffers filter distortion when it try to downscale the filter, especially for those filters with small kernel size. Thus, our network is not effective when it is test on the OF_Small dataset. Instead of taking frame 2 as the reference, an alternative approach would be taking the feature of smaller scale as the reference, so that the filter downscaling is avoided. Further more, the dataset where we evaluate our methods is fairly simple. More experiments on complex dataset are still needed.

References

[1] Anonymous. N-jetnet: Learning the resolution of deep convolutional neural networks. unpublished, 2018.

[2] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):26–33, 1986.

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[4] L. Florack, B. T. H. Romeny, M. Viergever, and J. Koenderink. The gaussian scale-space paradigm and the multiscale local jet. *International Journal of Computer Vision*, 18(1):61–75, 1996.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[7] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[8] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin, and P.-A. Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. *arXiv preprint arXiv:1804.00433*, 2018.

[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. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, page 6, 2017.

[10] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders. Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619, 2016.

[11] A. Jaegle, S. Phillips, and K. Daniilidis. Fast, robust, continuous monocular egomotion computation. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 773–780. IEEE, 2016.

[12] A. Kanazawa, A. Sharma, and D. Jacobs. Locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1412.5104*, 2014.

[13] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[15] Y. Lu, J. Valmadre, H. Wang, J. Kannala, M. Harandi, and P. H. Torr. Devon: Deformable volume network for learning optical flow. *arXiv preprint arXiv:1802.07351*, 2018.

[16] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[17] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 2. IEEE, 2017.

[18] S. R. Richter, Z. Hayder, and V. Koltun. playing for benchmarks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2232–2241, 2017.

[19] L. Sevilla-Lara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black. On the integration of optical flow and action recognition. *arXiv preprint arXiv:1712.08416*, 2017.

[20] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and

M. Abidi. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*, 11(3):204–218, 2005.

[21] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[22] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[23] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2233–2240. IEEE, 2011.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[25] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1663–1668. IEEE, 2009.

[26] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[27] A. Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984.

[28] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014.

2 Introduction

Powered by the massive amount of available data and rapid growth of computing performance, in recent years, the convolutional neural network (CNN) has achieved impressive success in many vision-oriented tasks like image classification [29], action recognition [33], image reconstruction [23], segmentation [2].

In optical flow estimation problem given two consecutive images, there is also a noticed shift from energy-based method to the learning-based method with CNN recently. In 2015, Dosovitskiy *etal*. [5] made the first attempt to train the CNNs end-to-end that learn the optical flow directly from image pairs. The model they proposed is called FlowNet which has an encoder-decoder architecture. Although at that time the traditional energy-based method [19] was still dominating with regard to optical flow accuracy, the FlowNet shows that the optical flow estimation can be treated as a learning problem. Many other efforts have been made to estimate optical flow with CNN since the work of FlowNet. Ilg *etal*. improve the estimation result significantly by stacking multiple FlowNets. In SPy-net[18] and PWC-net [27], large motions are dealt by the spatial-pyramid architecture. This network architecture greatly simplifies the network since it only needs to estimate small displacement at each pyramid level.

The CNN-based method has been proved to provide accurate and robust estimation in synthetic datasets or real-world video [9, 27]. However, the situation that involves big scale change in images is not well studied. In this work, we focus on the optical flow estimation with CNN in this situation and attempt to estimate the scale change between images as well.

2.1. Motivation

The scale matters in estimating optical flow with CNN because of the poor scale-invariance property of the convolution layer. In other words, convolution layer can hardly capture pattern appearing with various scales. The convolution layer captures the desired pattern by convolving the image input with the filter. This convolution is calculated as the inner product of filter and image batch, and it would not preserve the same activation when the scale of image changes. The current CNN-based methods rely on extracting features from two images through the same convolution channel. Given two image with obvious scale change, the CNN-based method is likely to suffer severe performance degradation since it is not able to identify the same feature with various scales in the beginning.

The scale change is very common in image sequence especially in those scenarios where obvious motion occurs in the direction of the camera, like video from handle camera and onboard camera. The motion analysis tasks in images generally use optical flow as a fundamental input. Optical flow represents the pixel displacement on the image plane. However, it does not reveal the depth motion of the observed object directly. The scale change in images implies the depth motion of the rigid object in images, which could be the supplementary measurement to optical flow in motion analysis tasks.

2.2. Research Questions

- Do the existing CNN-based methods suffer performance degradation while estimating optical flow from image pairs in presence of larger scale change?
- In the convolutional neural network, how to efficiently model the scale change in images pair and make

the network capable of detecting the feature of various scale?

- Is it possible to build a model that can estimate scale change from image pair and receive this scale change to improve the optical flow estimation?

2.3. Outline

The following Chapters are the supplementary materials to the scientific paper above. In Chapter 3, we introduce the background knowledge about optical flow and artificial neural network. The details of our model are given in Chapter 4. Chapter 5 allows a closer look the experimental results. Finally, Chapter 6 provides a reference to an unpublished paper.

3

Preliminaries

3.1. Introduction to Optical Flow

3.1.1. Definition of Optical Flow

The motion in the image sequence is brought by the relative motion between the objects of the observed scene and the observer. Optical flow is a common representation to describe the motion between images since it is the lowest-level characterization of motion in consecutive image frames. It measures the displacement of each visual pixel in the image. Once this displacement motion field of images is measured, it can be used as an black-box input in comprehensive motion analysis tasks, like object tracking [25], boundary detection [28], action recognition [24], Ego-Motion estimation [11].

Figure 3.1 gives an example of the 2D displacement vector of one pixel. Assume that there is an image coordinate $(\mathbf{u},\mathbf{v})$. At time t_1, a point is captured in the image at the 2D location (u_1, v_1). At t_2 it moves to other position due to the camera motion or the motion itself. Then the optical flow of this pixel point is a vector calculated as $(u_2 - u_1, v_2 - v_1)$.

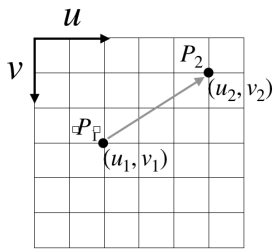


Figure 3.1: The pixel displacement in the image coordinate. (u_1, v_1) and (u_2, v_2) is the pixel location at t_1 and t_2. The arrow denotes the 2D displacement vector.

To better visualize the 2D motion field, this paper uses the common HS color space to encode the optical flow field. Figure 3.2 provides an example of visualization of optical flow from Scene Flow Dataset [15]. The hue and saturation represent the direction and magnitude of the motion vector, respectively.

3.1.2. Optical Flow Estimation

Optical flow estimation is an old problem and many methods have been introduced to improve the estimation result since the work of Horn and Schunck [8]. The original HS method assumes that the pixel brightness is preserved between images and the optical flow is smooth over the whole image. The optical flow estimation problem then is converted to the objective function optimization problem which considers these two constraints, as it is shown in equation 3.1.

$$\sum_{i,j}(I_1(i,j) - I_2(i+u, j+v))^2 + \alpha(\nabla^2 u + \nabla^2 v) \tag{3.1}$$

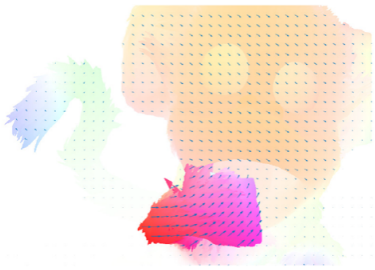
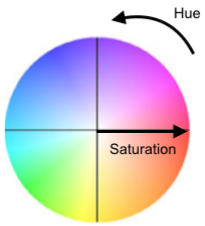


Figure 3.2: Optical flow example from Scene Flow dataset[15]. It describes the scene that monkey is putting its hand into the mouth.



HS color space.

Figure 3.3: The optical flow coding in this paper. The hue and the saturation represent the direction and magnitude of the motion vector, respectively.

I_1 and I_2 is the brightness of frame 1 and frame 2 respectively. α is the balance parameter and (u, v) is the 2D motion field. The classic methods following HS method keep the similar structure but introduce many improvements. Different forms of data constraint [26, 30] and smooth constraint [6, 31] were considered. Since the quadratic loss function in HS formula suffers a lot from outliers, many other robust loss functions are presented [3, 21, 26].

Recently, artificial neural networks have greatly improved the state of art result of many computer vision tasks including optical flow estimation problem [4]. In the learning-based method with the neural network, the optical flow estimate problem is solved in an end-to-end manner. By defining the loss between the optical flow estimation and ground truth, the neural network is able to update its variables to provide better estimation through the gradient descent method. After thousands of training steps, the neural network learns to estimate the final smooth optical flow directly from two images input. Nowadays these CNN-based method is dominating in optical flow estimation problem with regard to speed and accuracy [27].

3.2. Background about Artificial Neural Networks

Artificial Neural Networks(ANN) is a part of machine learning method, which learns the representation of data through multiple connected layers. It is inspired by the biological neural networks in the human brain, and it intends to imitate the way how a human learns. Recent years, the methods based on the neural network reach state-of-art result in many fields like computer vision, machine translation, bioinformatics [14].

3.2.1. Neurons and Multilayer Perceptron

A simple ANN is composed of three layers, one input layer, one hidden layer, and one output layer, as it is shown in figure 3.4. This simple ANN is called Multilayer Perceptron (MLP). The input layer represents the input data to the ANN and each layer following the input layer is formed by one or some artificial neurons. An artificial neuron is the elementary unit in ANN. What it does is just provide the weighted sum of all the received inputs. Its mathematical model is illustrated in figure 3.5. Usually, to introduce non-linearity, the output of a neuron is passed to a nonlinear activation function, and the most common activation functions are TanH and ReLU [16].

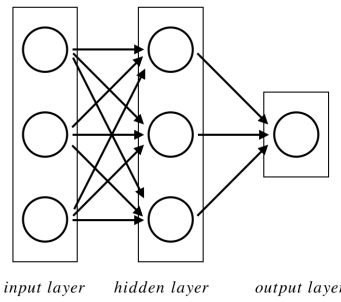


Figure 3.4: A simple example of ANN. It consists of three layers, input layer, hidden layer and output layer.

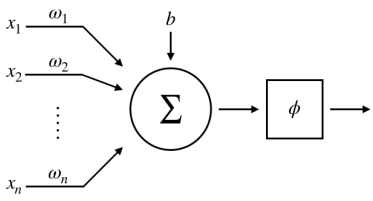
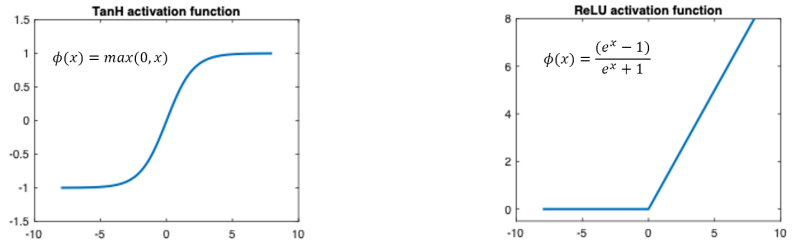


Figure 3.5: Model of a neuron. It pass the weighted sum of received inputs to the activation function and output the final result.

The final output of ANN is used to define a task-determined loss function. In general, mean square error (MSE) for regression problem and cross entropy loss for classification problem. The learnable variables of ANN are the weights and bias inside each neuron. During the training phase, these learnable variables are updated through the gradient descent method by taking the derivatives of loss function[22]. TanH is a zero-center function and it restricts the output from -1 to 1, as it is shown in figure 3.6. However, it saturates when the input is far away from 0, in which case the gradient is very close to 0 and the learnable variable will not get effective update, leading to gradients vanish problem. Therefore Tanh usually converges slower than ReLU [12].



(a) TanH activation function.

(b) ReLU activation function.

Figure 3.6: The Tanh and ReLU activation function. Tanh suffers from gradient vanishing problem.

3.2.2. Convolutional Neural Network

The fully connected layer described above takes all the outputs of the previous layer as inputs. So it is impractical to apply this architecture while analyzing the image which has tons of input pixels. Unlike the fully connected layer, the neuron in convolution layer receives inputs only from a restricted square area of the previous layer, and this area is called receptive field. Another key difference is that when the neurons calculate their outputs, they share the same weights and bias among all the locations. This share-weights architecture not only reduces the number of learnable variables greatly but also provides the crucial translation-invariant property in image analyzing. An example of convolution layer is shown in figure 3.7

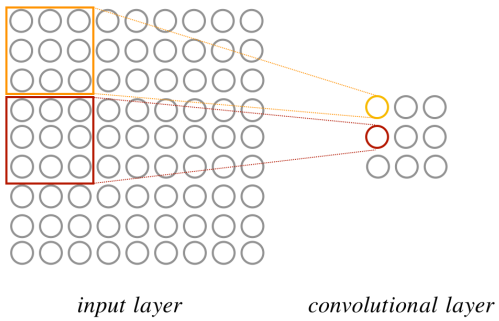


Figure 3.7: Visualization of convolutional layer. The kernel size is 3, stride is 3.

Figure 3.8 shows the typical CNN architecture that receives image input. After several convolution and pooling layers, the high-level representation of the image is obtained, and this high-level representation is passed through the fully connected layers for final reasoning like recognition [13].

3.2.3. Convolutional Encoder-Decoder

In many computer vision tasks, like image segmentation, depth estimation, the desired output has the same size as the input image. To solve this problem in CNN ending with fully connected layers, the convolutional encoder-decoder architecture is introduced. The convolutional encoder-decoder contains an encoder part and a decoder part, as it is shown in figure 3.9. The encoder part follows the typical architecture of the convolutional neural network in figure 3.8. It transforms the input image to high-level feature map. To gradually

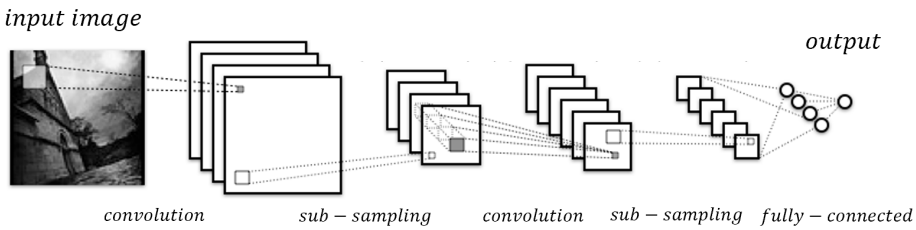


Figure 3.8: Typical CNN architecture. It contains several convolution and sub-sampling layers to extract high-level feature map. Fully connected layer is used to produce final output in the end.

increase the resolution of feature map from encoder part and finally produce the output, an inverse version of encoder part is supplemented, called decoder part, where unpooling and de-convolution layers are just the reverse operation of pooling and convolution layer respectively. Additionally, it is possible to introduce shortcut from the encoder part to decoder part, like U-Net [20], so that the high-resolution feature maps from encoder part help to generate precise output in the expanding part.

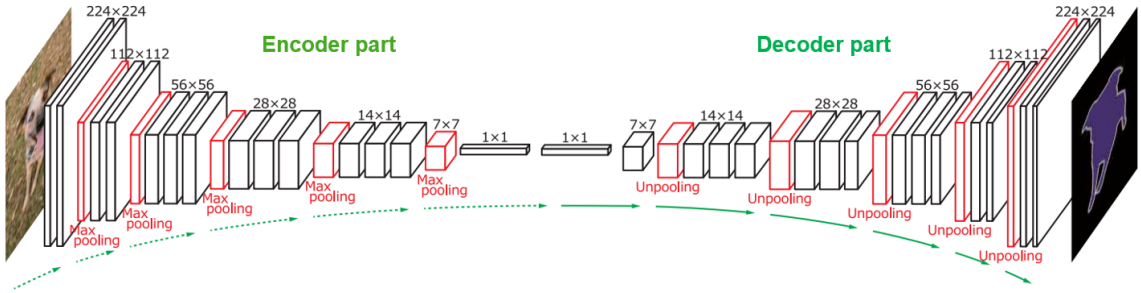


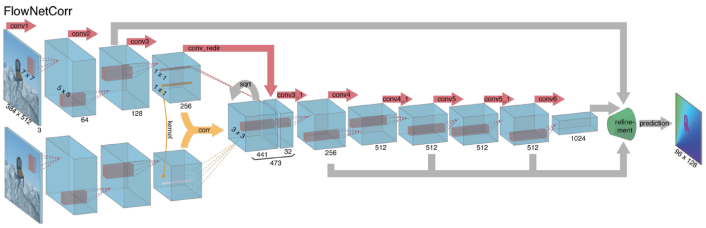
Figure 3.9: The network architecture of convolutional encoder-decoder [17]. The encoder part is the same as the typical CNN. The decoder part consists of several de-convolution and unpooling layers to perform refinement.

4

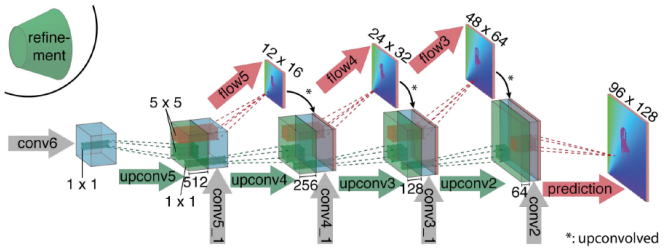
Network Details

4.1. Network Architecture of FlowNetCorr

This paper is based on the work of FlowNet [5], and FlowNetCorr is chosen as the baseline method, so it is necessary to introduce its architecture into details. The FlowNetCorr consists of two parts, the contractive part and the refinement part, as it is shown in figure 4.1.



(a) Contractive part



(b) Refinement part

Figure 4.1: Architecture of FlowNetCorr. In the contractive part, two images are passed through two weight-shared convolution channels. The outputs then are fused by the correlation layer to perform feature matching. The refinement part mainly consists of de-convolution layers. It is worth noting that the coarse estimation is concatenated to its next feature map.

Contractive part is mainly used to extract feature and perform feature matching between two images. Two input images are passed through two share-wight channels to contract meaningful representations. Each channel contains three convolutional layers of stride 2 and their kernel size are 7,5 and 5, respectively. Then the outputs of these two streams are combined by a correlation layer that performs multiplicative batch matching between two feature maps. Given feature maps f_1 and f_2 from two channels, the single comparison between two batch centered at x_1 in f_1 and centered at x_2 in f_2 is calculated as

$$Corr(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} < f_1(x_1 + o), f_2(x_2 + o) > \tag{4.1}$$

where, k is the batch radius and <> represents the inner product of two vectors.

The batch correlation measures the similarity of batches between f_1 and f_2. In theory, the corresponding pixel point can go everywhere in f_2. Thus, we need to do $(H * W)^2$ comparisons for all the combination of

batches. To save the computation burden, the batch comparison is only performed within a square region center at x_1 instead of searching in the whole image. Given the maximum displacement d, for every location x_1 in frame 1, the batch location x_2 in f_2 is limited to be x_1's neighbour such that $x_1 - x_2 \in [-d, d] \times [-d, d]$. By defining the maximum displacement, the number of batch comparisons is reduced to $H \times W \times (2d + 1)$. The output of the correlation layer should be four dimensions, two dimensions for the x_1 location and two dimensions for the relative displacement. However, in practice, the 2D displacement is flatted into one dimension. Therefore, the output feature map has the same size of input feature map but with a different depth of $(2d + 1)$. Then, the output of correlation layer is passed through several convolution layers to extract even higher-level feature map.

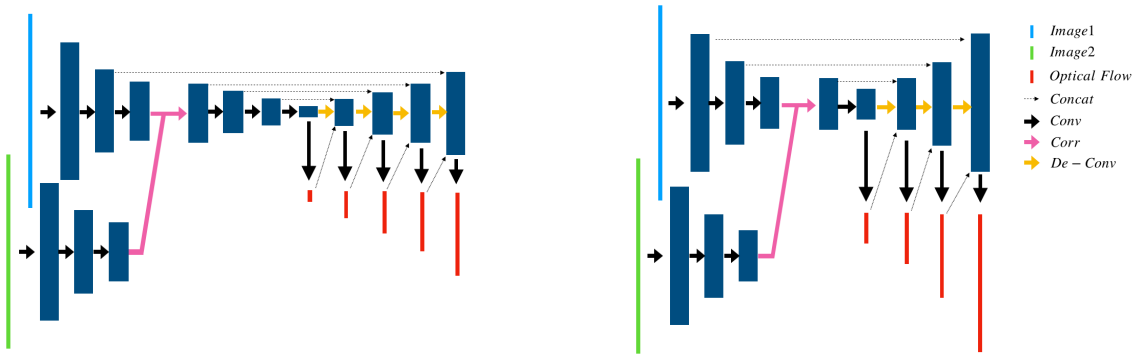
Refinement part aims to recover the high-resolution feature map from the high-level output of the contractive part. Like many other works [2, 17], it learns a deep de-convolutional network to perform the refinement. Additionally, the convolution layer is applied to different level feature maps in contractive part to generate coarse to fine optical flow estimations. The coarse optical flow is upsampled and then copied to its next stage by concatenation. Similar to UNet, FlowNet also introduces connection from contractive part to refinement part, which preserves both the high-level information from coarser feature maps and local details provided in lower layer feature maps [5].

As a trade-off between performance and computational complexity, only four de-convolution layers are used in the refinement part, which means the size of the finest optical flow estimation is still 4 times smaller than the original input, as shown in 4.1. To recover the full image resolution, a simple bilinear interpolation is implemented in the end. Since it gives multi-level optical flow estimations, the final loss function is a weighted sum of average endpoint errors between all these optical flow estimations and the downsampled ground truths.

4.2. Light Version of FlowNetCorr

The original FlowNetCorr is evaluated on the MPI Sintel Flow [4] dataset and its own Flying Chairs dataset where the image sizes are 1024×436 and 512×382, respectively. The model runs slow on the image of such big size. More specifically, one training step with a batch size of 8 takes about 1.5 seconds, and it takes above 10 days to finish the whole training process, 600k steps (Tensorflow 1.2 + GTX1080 Ti + Intel Xeon CPU E5-2620 v4 @ 2.10GHz).

Considering the limited computational resource and time, in this paper, we are working on our own optical dataset with much smaller image size, for example, 192×192. Therefore, we need to modify the FlowNetCorr to fit the smaller image and derive the light version of FlowNetCorr, called FlowNet_Light. This is mainly done by deleting two convolution layers in the contractive part and one de-convolution layer in the refinement part, as it is shown in figure 4.2. After these modifications, some of the encoder-to-decoder connections also need to be adjusted. In FlowNet_Light, the finest optical flow estimation is 2 times smaller than the original input. Thus, the same bilinear upsampling operation is used to generate the full-resolution output.



(a) Architecture of Light Version FlowNetCorr

(b) Architecture of Light Version FlowNetCorr

Figure 4.2: Comparison between FlowNetCorr and its light version, FlowNet_Light. The main change is the deletion of several layers.

4.3. Implementation Details of Pixel-Wise Scale Adjusted Convolution Layer

Following Scale-space theory [32] and Njet [7] we are able to approximate the filter weight by the weighted sum of a finite set of Gaussian derivatives [1] [10]. The biggest benefit brought by this approximation is that the scale and the shape of the filter are decoupled, controlled by two separate parameters σ and α. σ represents the scale of the Gaussian derivatives and α is a set of weights determined the contribution of different Gaussian derivatives. By combining the filter approximation and the traditional convolution layer, we obtained the Basis Convolution layer that can easily model the scale of the filter. Apart from the input feature map, the Basis Convolution Layer accepts a scale map that implies additional scale info from every pixel location. Before doing the convolution, the Basis Convolution layer constructs the filter with the scale adjusted for every pixel location based on the scale map, as it is shown in figure 4.3. So the Basis Convolution layer can capture the feature of the desired scale at each location while the traditional convolution layer can only capture feature of exactly the same size.

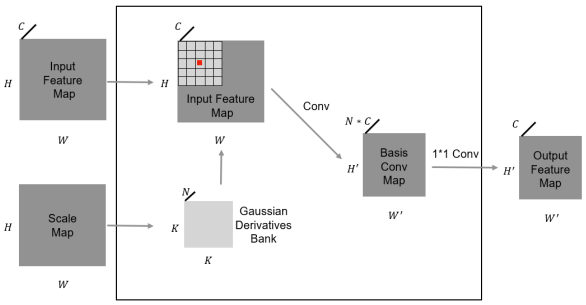


Figure 4.3: An illustration of the Basis Convolution layer. Before doing the convolution, it receives the scale input of each pixel position and fetches the proper Gaussian derivative from the derivative bank. After the convolution with Gaussian derivatives, a traditional depth-wise convolution is implemented to calculate the weighted sum, similar to [10].

Maximum Order of Gaussian derivatives. A finite set of Gaussian derivatives up to a maximum order are used to approximate the filter. It has been proven [12] that 3^{rd} or 4^{th} order is sufficient to capture all the local image variation perceivable by humans, so in the Basis Convolution layer we only consider the derivatives up to order 4.

Learnable parameters. Unlike the traditional convolutional layer that directly learns the pixel weights of the filters, the Basis Convolution layer learns the weight parameters for each unique Gaussian derivative. The numbers of learnable parameters of Basis Convolution layer and traditional convolution layer are $C \times (\frac{(N+1)*N}{2}+N+1)$, $C \times K^2$, respectively. K is kernel size. C is the number of output channels of the convolutional layer. $\frac{(N+1)*N}{2}+N+1$ is the total number of derivatives up to order N. The number of learnable parameters in Basis Convolution layer remains the same as the maximum order is fixed, so the Basis Convolution layer is less complex than convolution layer especially with large kernel size, with regard to the number of model parameters.

Default scales. By default (in case the scale doesn't need to be adjusted), the Basis Convolution layer use derivatives of scales 1, 2 and 3 to approximate filters. Our experiment and [10] show that multi-scale derivatives are crucial to enhance the encoding capability of the convolution layer with filter approximation by Gaussian derivatives. When it is about to do the convolution it calculates the proper scale for each pixel position by multiplication of default scale and the received input from scale map, so the location with scale input of 1 will use the default scales.

Derivatives bank. Calculating the derivatives with proper scale for every pixel in feature map is extremely computationally expensive. Fortunately, the precision requirement for scale is slow, which allows us to maintain a small derivative bank with possible discrete scales. Instead of calculating the derivative in runtime, the Basis Convolution layer fetches the suitable derivative from this pre-calculated derivatives bank. The scale is evenly discretized in the logarithmic scale of a small base 1.1, from -20 to 20 with step of 1. Which means the derivative bank contains derivatives of 41 sigmas from $\sigma = 0.15$ (1.1^{-20}) to $\sigma = 6.7$ (1.1^{20}).

Kernel size. If the scale increases or decreases, the Gaussian derivative will expands or shrinks accordingly, as well as the kernel size. Therefore the kernel size should be adjustable for every pixel location depends on its scale input. We choose the kernel size equals to $2 \times \lceil 1.3\sigma \rceil + 1$, where $\lceil\ \rceil$is the roundup operation. The range is sufficiently large to hold a big part of the Gaussian derivative and also keep the odd number kernel

size. Thus, the kernel size of default scales 1, 2 and 3 are 5, 7 and 9 respectively.

5

Supplementary Results

5.1. Example of Data Augmentation

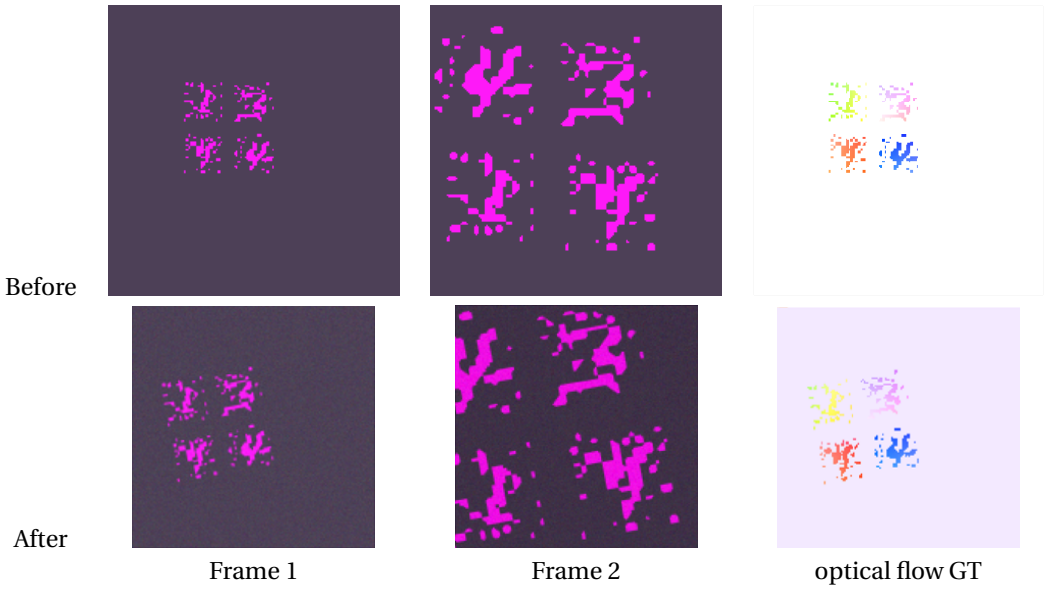


Figure 5.1: An example of data augmentation result. The data augmentation introduces variety to the images as well as the flow field.

5.2. Results of Testing Examples

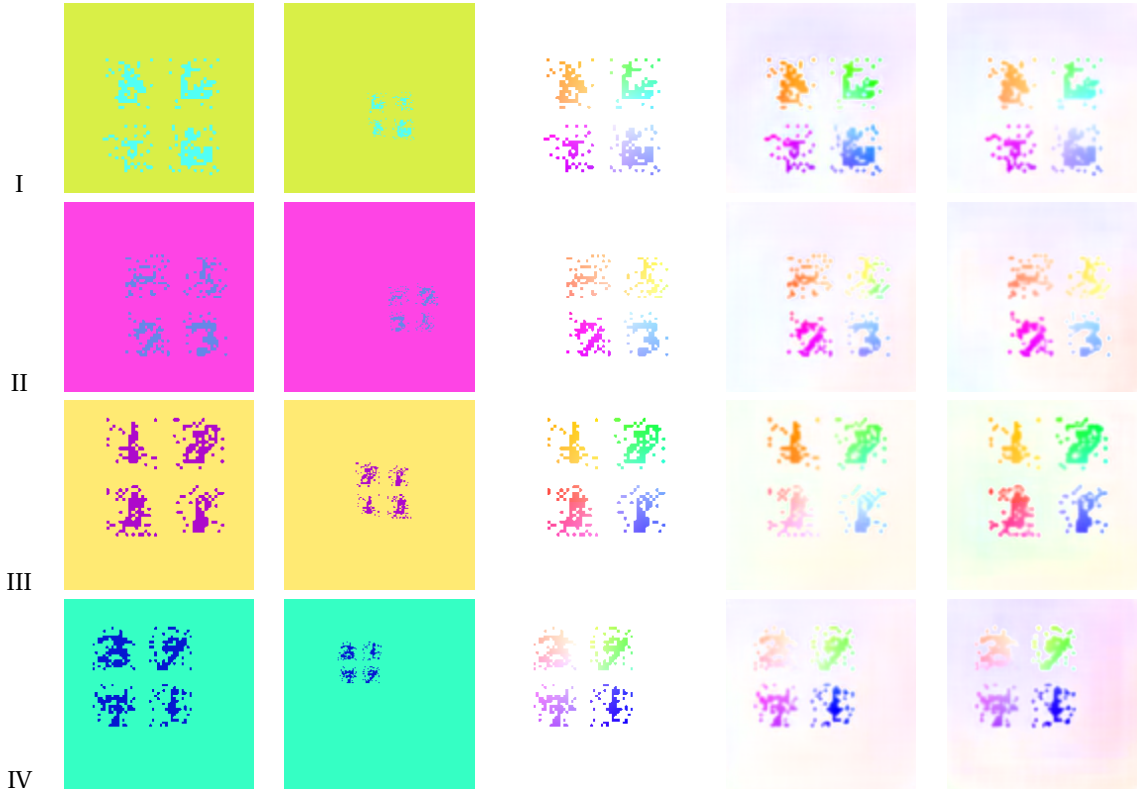


Figure 5.2: Testing result of optical flow of FlowNet_Light and FlowNet_Basis trained on OF_Big dataset. FlowNet_Basis provides more accurate result generally.

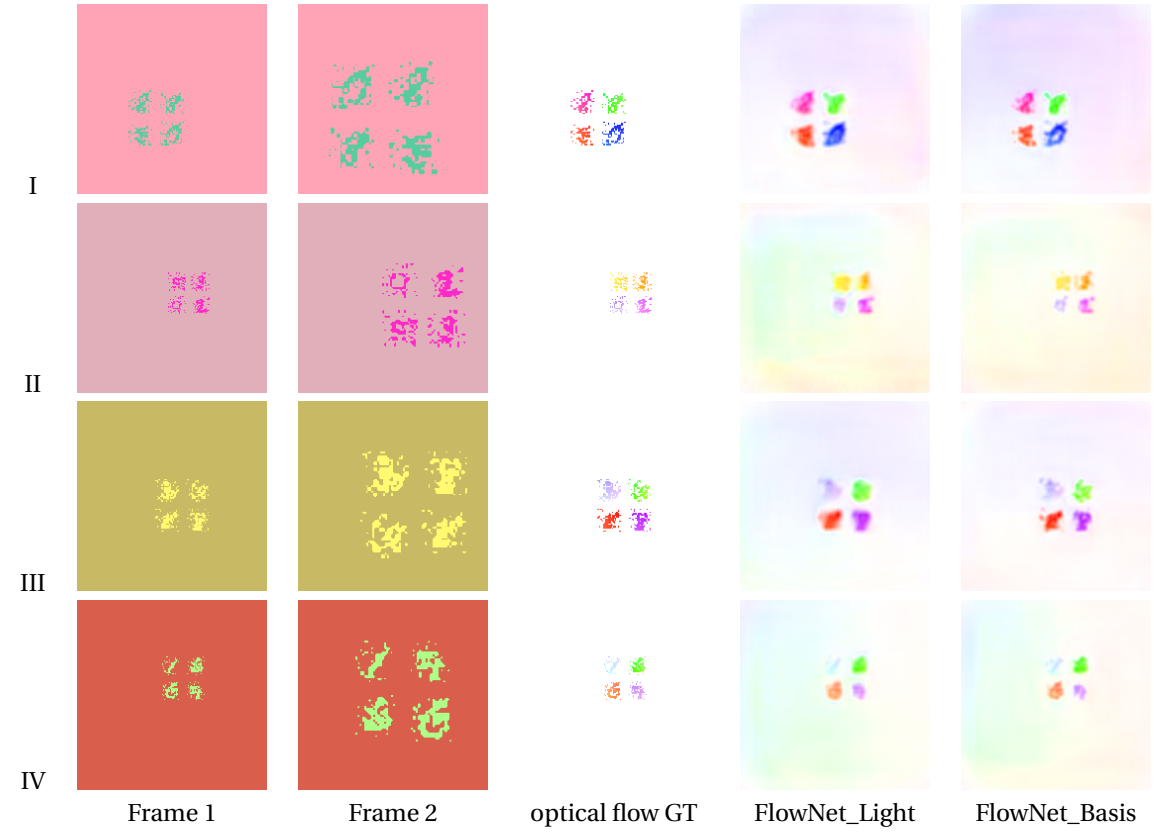


Figure 5.3: Testing results of FlowNet_Light and FlowNet_Basis trained on OF_Small dataset. No obvious difference between these two methods.

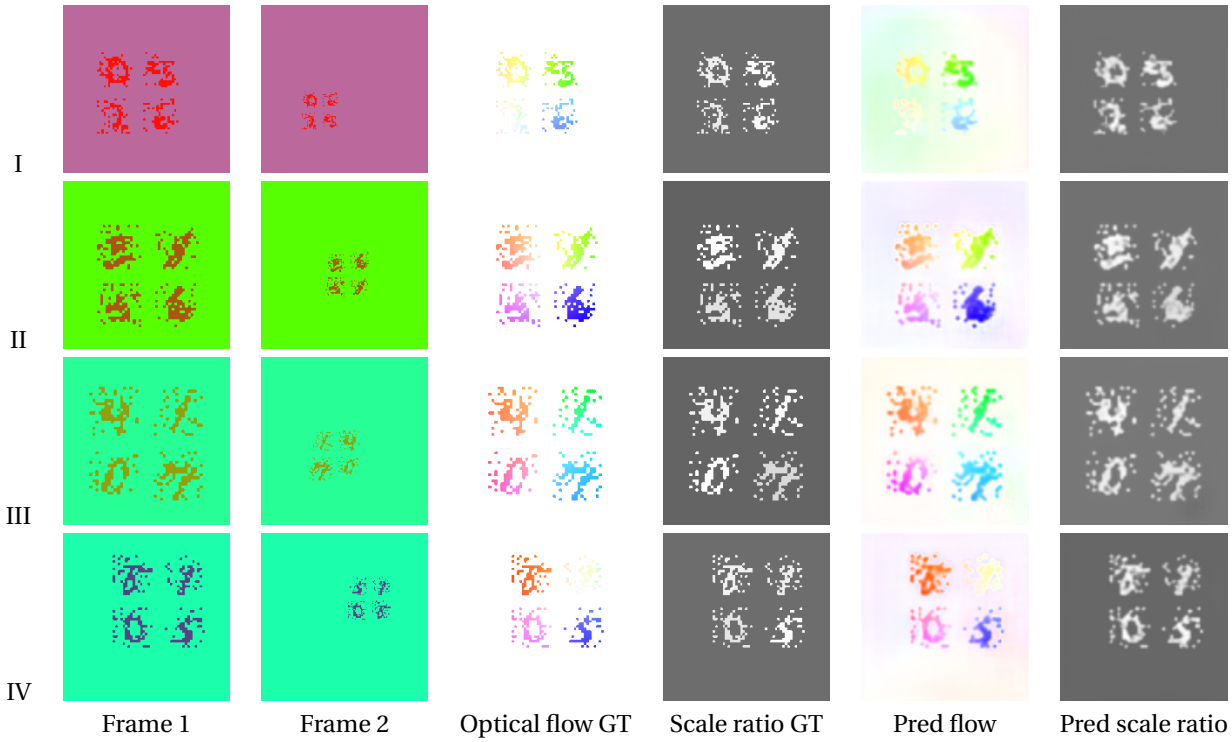
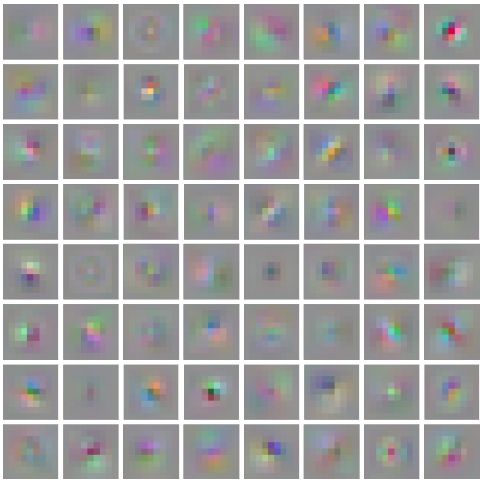
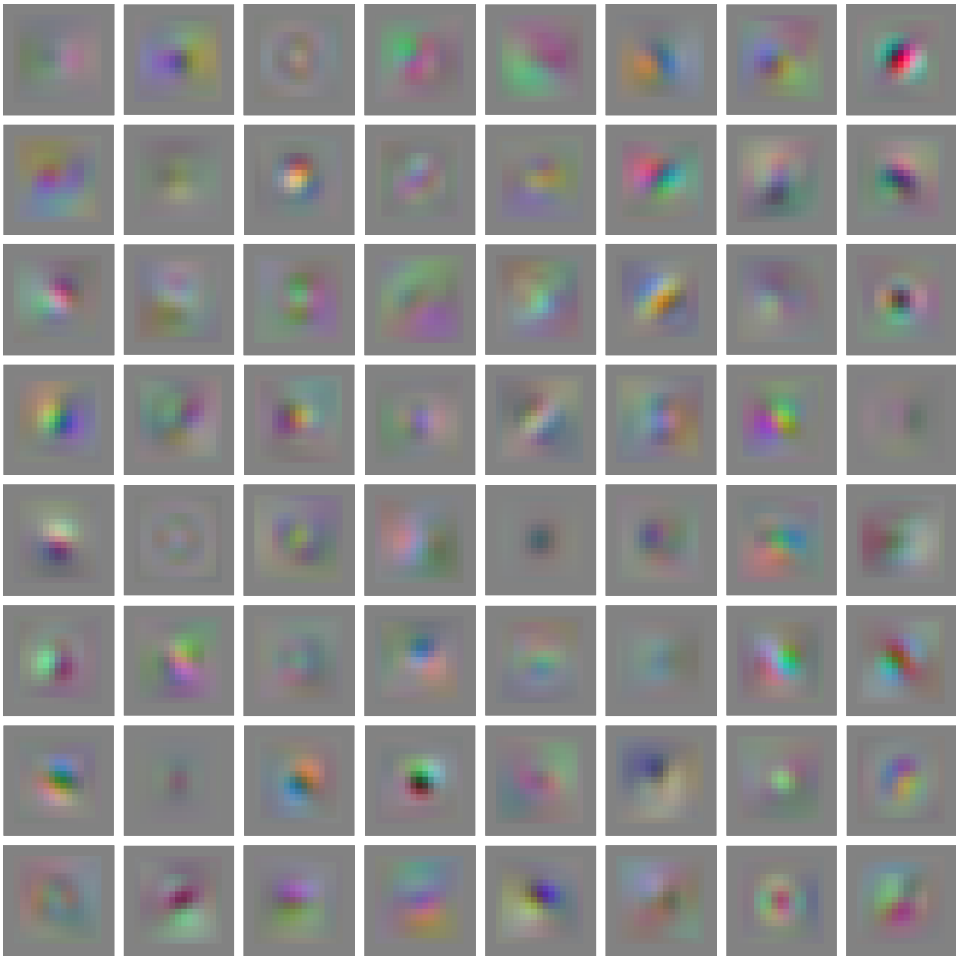


Figure 5.4: Testing results of FlowNet_FlowScale trained on OF_Big dataset. The FlowNet_FlowScale is capable of providing correct scale ratio and optical flow estimation jointly. But its flow estimation is not as good as Flownet_Basis.

5.3. Filters Visualization

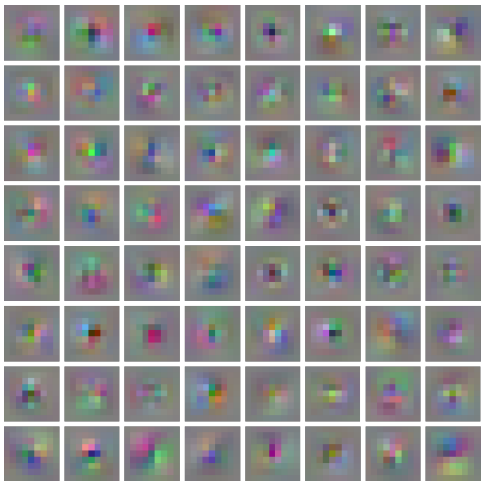


(a) Default scale.

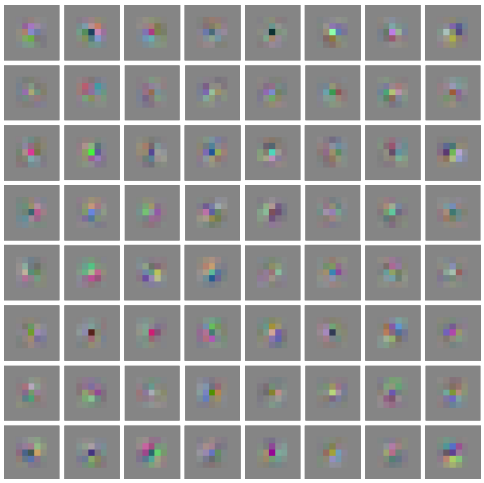


(b) With scale 2 times bigger.

Figure 5.5: The visualization of filters learned in the first layer of FlowNet_Basis on OF_Big dataset. The network learns zero, first and second order filter of different color combinations. And the filter is nicely upsampled by changing to a bigger scale.



(a) Default scale.



(b) With scale 2 times smaller.

Figure 5.6: The visualization of filters learned in the first layer of FlowNet_Basis on OF_Small dataset. when the scale becomes two times smaller, obvious distortion can be found.

Bibliography

[1] Anonymous. N-jetnet: Learning the resolution of deep convolutional neural networks. unpublished, 2018.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[5] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[6] Marius Drulea and Sergiu Nedevschi. Total variation regularization of local-global optical flow. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 318–323. IEEE, 2011.

[7] Luc Florack, Bart Ter Haar Romeny, Max Viergever, and Jan Koenderink. The gaussian scale-space paradigm and the multiscale local jet. *International Journal of Computer Vision*, 18(1):61–75, 1996.

[8] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[9] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, page 6, 2017.

[10] Jorn-Henrik Jacobsen, Jan van Gemert, Zhongyu Lou, and Arnold WM Smeulders. Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619, 2016.

[11] Andrew Jaegle, Stephen Phillips, and Kostas Daniilidis. Fast, robust, continuous monocular egomotion computation. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 773–780. IEEE, 2016.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[15] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[16] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[17] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[18] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 2. IEEE, 2017.

[19] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[21] Stefan Roth, Victor Lempitsky, and Carsten Rother. Discrete-continuous optimization for optical flow estimation. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 1–22. Springer, 2009.

[22] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[23] Jo Schlemper, Jose Caballero, Joseph V Hajnal, Anthony N Price, and Daniel Rueckert. A deep cascade of convolutional neural networks for dynamic mr image reconstruction. *IEEE transactions on Medical Imaging*, 37(2):491–503, 2018.

[24] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. *arXiv preprint arXiv:1712.08416*, 2017.

[25] Jeongho Shin, Sangjin Kim, Sangkyu Kang, Seong-Won Lee, Joonki Paik, Besma Abidi, and Mongi Abidi. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*, 11(3):204–218, 2005.

[26] Frank Steinbrücker, Thomas Pock, and Daniel Cremers. Advanced data terms for variational optic flow estimation. In *VMV*, pages 155–164, 2009.

[27] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[28] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2233–2240. IEEE, 2011.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[30] Andreas Wedel, Thomas Pock, Jürgen Braun, Uwe Franke, and Daniel Cremers. Duality tv-l1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6. IEEE, 2008.

[31] Joachim Weickert and Christoph Schnörr. A theoretical framework for convex regularizers in pde-based computation of image motion. *International Journal of Computer Vision*, 45(3):245–264, 2001.

[32] Andrew Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984.

[33] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G Hauptmann. Hidden two-stream convolutional networks for action recognition. *arXiv preprint arXiv:1704.00389*, 2017.