UNIVERSITÄT DES SAARLANDES
Prof. Dr.-Ing. Eddy Ilg
Computer Vision and Machine Perception Lab
ilg@cs.uni-saarland.de

January 16, 2024

3D Computer Vision

Assignment 5: Estimating Optical Flow with FlowNet

Submission Deadline: **Jan 26th 8am**

Optical flow is one of the fundamental problems in classical computer vision and serves as an important building block for many downstream tasks. Traditionally, optical flow was solved by explicitly formulating assumptions about correspondences based on encodings of appearance similarity and neighborhood consistency [3]. In this assignment, you will be implementing the seminal deep learning works FlowNetS and FlowNetC [2].

Privacy Note: Please note that you are not allowed to publish the solutions to any of the excercises publicly.

Part 1 - Evaluating Optical Flow

- Q1: [2 Points] The most common metric to evaluate flow fields is the mean endpoint error. For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$, the endpoint error is defined as their Euclidean distance:

$$EPE(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2. \tag{1}$$

  The average endpoint error (AEPE) over the flow field is defined as the mean over the EPE for every pixel. In the file `eval.py` you need to implement the function `AEPE`. You can verify your results with the jupyter notebook `visualization.ipynb`. The used color coding for the visualization is shown in Figure 1.

Part 2 - Variational Optical Flow

Before deep learning methods took over in the field of optical flow, variational methods were the top-performing methods. The seminal work of Horn and Schunck [3] introduced the first algorithm to estimate optical flow in the variational framework. Since then, a lot of improvements have been made to further advance variational methods, including the introduction of descriptor matches in Large Displacement Optical Flow (LDOF) [1]. We supply two executables `HornSchunck` and `LDOF` that provide an implementation of those individual methods.

- Q1: [2 Points] Run the two methods by executing the corresponding cell in the notebook `visualization.ipynb`. Inspect the flow fields and reason about their qualities by writing 4-5 bullet points about the relative strengths and weaknesses of the methods. Place your answer as text in the jupyter notebook.

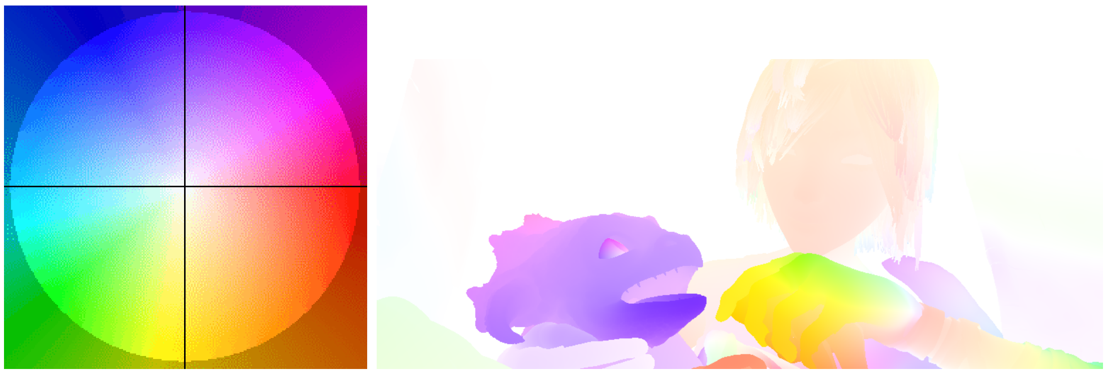
(a) Color code (b) Visualization of flow field

Figure 1: Supplementary information for Part 1

Part 3 - Optical Flow with Deep Learning

(a) **FlownetS:** This is a general architecture that was used to estimate optical flow. It comprises of a generic UNet [4] with no specialized operations that are specific to the optical flow problem. In this exercise, you need to complete the supplied FlownetS architecture in the files `models/FlowNetS.py` and `models/blocks.py` by implementing the layers and the forward pass in the missing modules. Please refer to the FlowNet paper (FlowNet) about the details of the architecture. **Hint:** The architecture figure in the paper provides all the necessary parameters and implementation details. Please note that you may not change the names of the corresponding layers in the networks, or otherwise the weights will not match correctly with your implementation.

- Q1: [3 Points] Implement `ConvLayer` in the file `models/blocks.py`. Pay attention to applying the correct leaky ReLU.
- Q2: [3 Points] Implement the FlowNetS encoder in the file `models/FlowNetS.py`.
- Q3: [3 Points] Implement `UpConvLayer` in the file `models/blocks.py`. Pay attention to applying the correct leaky ReLU.
- Q4: [8 Points] Implement the decoder in the file `models/blocks.py`.
- Q5: [3 Points] Finally, put all pieces together in the FlowNetS `forward()` function in the file `models/FlowNetS.py`. Once implemented, test your implementation with the jupyter notebook.

(b) **FlownetC:** In this exercise, you will implement and estimate flow fields with the FlownetC architecture. We again supply a skeleton implementation where we require you to implement the missing modules. FlowNetC uses some of the building blocks from FlowNetS architecture, but has a specialized architecture tailored towards optical flow. The relevant source code is located in the file `FlowNetC.py`.

- Q1: [2 Points] Start by implementing the Siamese feature extractor `FeatureExtractor` that provides individual encodings of each image in the file `models/FlowNetC.py`.
- Q2: [10 Points] Now you can implement the centerpiece of FlowNetC that correlates the feature maps of each image in `CorrelationLayer` in the file `models/FlowNetC.py`. Please refer to the FlowNet paper (FlowNet) for details. The correlation between two positions $\boldsymbol{x}_1, \boldsymbol{x}_2$ in the feature map is defined as:

$$c(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{\boldsymbol{o} \in [-k,k] \times [-k,k]} \langle \boldsymbol{f}_1(\boldsymbol{x}_1 + \boldsymbol{o}), \boldsymbol{f}_2(\boldsymbol{x}_2 + \boldsymbol{o}) \rangle. \quad (2)$$

However, this full operation is not computationally feasible for all possible combinations of positions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Therefore, there are some simplifying assumptions. First, we set $k = 0$ to compare single feature vectors instead of patches (note that a feature vector already encodes data from a patch). Second, we only consider target locations within a certain neighborhood $D := 2d + 1$ around the considered position $\boldsymbol{x}_1$ in $\boldsymbol{f}_2$ with $d = 20$, resulting in a total neighborhood size of 41×41. Finally, there are two stride parameters $s1 = 1$, $s2 = 2$. The crucial one here is $s2$, which impacts the sampling positions in the second feature map. For each position $\boldsymbol{x}_1$ in feature map $\boldsymbol{f}_1$, we compute the dot product with $\lceil 41/s2 \rceil^2 = 21^2 = 441$ positions $\boldsymbol{x}_2$ in feature map $\boldsymbol{f}_2$.

- Q3: [2 Point] Implement the encoder in the file `models/FlowNetC.py`.
- Q4: [4 Points] Finally, put all pieces together in the FlowNetC encoder in the file `models/FlowNetc.py` and test your implementation by running the corresponding cell in the notebook. To test the functionality of your correlation layer, we provide a sample output in the file `data/correlation_layer_output.pth` that you can test by calling `test_correlation` in the model wrapper.

Best of Luck!

References

[1] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010.

[2] 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.

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

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.