Seminar on Current Works in Computer Vision

Hierarchical Discrete Distribution Decomposition for Match Density Estimation

Presenter: Saurav Shanu

Authors
Zhichao Yin
Trevor Darrell
Fisher Yu
Table of contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
Introduction

- Find dense correspondences for
  - Stereo Matching
  - Optical Flow
with match density estimation
Introduction -- HD$^3$

- Method is “Hierarchical Discrete Distribution Decomposition.”
- A general probabilistic frame-work for match density estimation.
- Model-inherent match density estimation.
HD$^3$

- Hierarchical decomposition of the image.
- Calculate match density at each level.
- Calculate full match density.
Why decomposing Match Density?

- Estimating full match density can be computationally expensive.
- For a $1000 \times 1000$ image,
  - With displacement range of motion vector $f_{ij}$, $[-50, 50]$.
  - Total number of cells to compute, $10^{10}$. 

30.01.2020
Seminar on Current Works in Computer Vision
Table of contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
Related Work

- Effective deep learning models for flow estimation are available.
- Some of them do uncertainty estimation as well.
- They lack non-parametric match density.

Ilg et al. 2016
Early work on uncertainty estimation

- Need for confidences established by Barron et al. IJCV 1994.
- Confidence measure was mostly based on the input image like
  - Gradient of the image sequence
  - Hessian of the image sequence
  - the trace or smallest eigenvalue of the structure tensor.
- For example, in the Lucas-Kanade method confidence is based on the gradient of the image.
Other methods for uncertainty estimation

- Kondermann et al. (ECCV, 2008) gave uncertainty estimation through hypothesis testing.
  - Hypothesis - “The central flow vector of a given flow field patch follows the underlying conditional distribution given the remaining flow vectors of the patch.”

- Mac Aodha et al. trained a classifier to assess the prediction quality in terms of end-point-error.
  - Confidence for each flow vector as the probability of that flow being below some specified error threshold.
Gast et al. CVPR 2018 proposed probabilistic output layers.

For computational tractability, they assumed Gaussian noise and adopted a parametric distribution.

Performance is only competitive with the deterministic counterparts.
Introduction
Related Work
Method
Experiments
Summary
Network Architecture – Single layer
Extract multi-scale features from images

- Extract features $F_1$ and $F_2$ from $I_1$ and $I_2$ via DLA (Fisher et al. CVPR 2018)
DLA – Deep layer aggregation
Upscale motion vector

- Get motion vector from previous layer.
- Upsample it to current layer size.
Find correlation

- Find correlations between
  - reference frame features.
  - warped target frame features.
Feed to Density Decoder

- Feed cost volume, features $F_1$ and density embedding to the density decoder.
- Use density embedding from previous layer.
Density decoder

- Density decoder is a ConvNet which gives density embedding and match density.
Retrieving Motion vector

- Motion vector is calculated from the D2V method using match densities.
D2V- Density to Vector

- Converts match density to vector.
Calculate Loss

- Ground truth motion vector is converted to match density by V2D method.
- Loss between both densities is then calculated.
V2D – Vector to Density

- Converts ground truth motion vector to ground truth match density.
Loss

- Loss is represented as Kullback-Leibler divergence.

\[ \mathcal{L} = \sum_{l} \sum_{g \in R_{g_l}} p_{gt}^l(g) \left( \log p_{gt}^l(g) - \log p_{res}^l(g) \right). \]

where,

- \( g^l = f^l - \varphi(f^{l-1}) \)
- \( f^l \): flow vector at layer \( l \).
- \( \varphi \): upsampling operator.
- \( l \): layers in network.
- \( R_{g_l} \): Support set of \( g \).
Table of contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
General setup

- Entire network is trained in an end-to-end manner.
- The weights of pyramid feature extractor are initialized from the ImageNet pre-trained model.
- Same scheme is applied to both stereo matching and optical flow with some modifications.
Stereo Matching on KITTI - Training

- **Pretraining on FlyingThings3D**
  - Pretraining for 200 epochs.
  - Batch size – 32.
  - Initial learning rate – $2 \times 10^{-4}$.

- **Finetuning on KITTI**
  - Training is performed for 2000 epochs.
  - Batch size – 16.
  - The initial learning rate is $1 \times 10^{-5}$.
    - decayed by 0.5 at the 1000th and the 1500th epoch.

30.01.2020
Seminar on Current Works in Computer Vision
Stereo matching on KITTI – Results

<table>
<thead>
<tr>
<th></th>
<th>KITTI 2012</th>
<th>KITTI 2015</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Out-Noc</td>
<td>Out-All</td>
<td>D1-bg D1-fg D1-all</td>
</tr>
<tr>
<td>SPS-st [46]</td>
<td>3.39</td>
<td>4.41</td>
<td>3.84 12.67 5.31</td>
</tr>
<tr>
<td>Displets v2 [14]</td>
<td>2.37</td>
<td>3.09</td>
<td>3.00 5.56 3.43</td>
</tr>
<tr>
<td>MC-CNN-acrt [51]</td>
<td>2.43</td>
<td>3.63</td>
<td>2.89 8.88 3.88</td>
</tr>
<tr>
<td>SGM-Net [35]</td>
<td>2.29</td>
<td>3.50</td>
<td>2.66 8.64 3.66</td>
</tr>
<tr>
<td>L-ResMatch [37]</td>
<td>2.27</td>
<td>3.40</td>
<td>2.72 6.95 3.42</td>
</tr>
<tr>
<td>GC-Net [24]</td>
<td>1.77</td>
<td>2.30</td>
<td>2.21 6.16 2.87</td>
</tr>
<tr>
<td>EdgeStereo [39]</td>
<td>1.73</td>
<td>2.18</td>
<td>2.27 4.18 2.59</td>
</tr>
<tr>
<td>PDSNet [41]</td>
<td>1.92</td>
<td>2.53</td>
<td>2.29 4.05 2.58</td>
</tr>
<tr>
<td>PSMNet [8]</td>
<td>1.49</td>
<td>1.89</td>
<td>1.86 4.62 2.32</td>
</tr>
<tr>
<td>SegStereo [47]</td>
<td>1.68</td>
<td>2.03</td>
<td>1.88 4.07 2.25</td>
</tr>
<tr>
<td>HD$^3$S (Ours)</td>
<td><strong>1.40</strong></td>
<td><strong>1.80</strong></td>
<td><strong>1.70 3.63 2.02</strong></td>
</tr>
</tbody>
</table>

All of the numbers denote percentages of disparity outliers.
Optical Flow

- Network pretrained on FlyingChairs and FlyingThings3D.
  - FlyingChairs –
    - batch size – 64.
    - initial learning rate - $4 \times 10^{-4}$.
  - FlyingThings3D –
    - batch size – 32.
    - initial learning rate - $4 \times 10^{-5}$.
Finetuning on Sintel

- Training is performed for 1200 epochs.
- Batch size – 32.
- The initial learning rate is $2 \times 10^{-5}$.  
  - decayed by 0.5 at the 600th and the 900th epoch.
## Optical flow results – Sintel

<table>
<thead>
<tr>
<th>Methods</th>
<th>Training</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>Final</td>
</tr>
<tr>
<td>PatchBatch [11]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EpicFlow [34]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPM-flow [18]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FullFlow [9]</td>
<td>-</td>
<td>3.60</td>
</tr>
<tr>
<td>MRFlow [44]</td>
<td>1.83</td>
<td>3.59</td>
</tr>
<tr>
<td>DCFLOW [45]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SpyNet-ft [33]</td>
<td>(3.17)</td>
<td>(4.32)</td>
</tr>
<tr>
<td>FlowNet2 [21]</td>
<td>2.02</td>
<td>3.14</td>
</tr>
<tr>
<td>FlowNet2-ft [21]</td>
<td>(1.45)</td>
<td>(2.01)</td>
</tr>
<tr>
<td>LiteFlowNet [19]</td>
<td>2.52</td>
<td>4.05</td>
</tr>
<tr>
<td>LiteFlowNet-ft [19]</td>
<td>(1.64)</td>
<td>(2.23)</td>
</tr>
<tr>
<td>PWC-Net [40]</td>
<td>2.55</td>
<td>3.93</td>
</tr>
<tr>
<td>PWC-Net-ft [40]</td>
<td>(2.02)</td>
<td>(2.08)</td>
</tr>
<tr>
<td>HD$^3$F (Ours)</td>
<td>3.84</td>
<td>8.77</td>
</tr>
<tr>
<td>HD$^3$F-ft (Ours)</td>
<td>(1.70)</td>
<td>(1.17)</td>
</tr>
</tbody>
</table>

Average EPE results on MPI Sintel dataset
- “-ft” means finetuning on the Sintel training set
- numbers in the parenthesis are results on data the method has been trained on.
Finetuning on KITTI

- Training is performed for 2000 epochs.
- Batch size – 16.
- The initial learning rate is $1 \times 10^{-5}$.
  - decayed by 0.5 at the 1000th and the 1500th epoch.

Same parameters as stereo matching.
Optical flow results – KITTI

<table>
<thead>
<tr>
<th>Methods</th>
<th>KITTI 2012</th>
<th></th>
<th></th>
<th></th>
<th>KITTI 2015</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AEPE</td>
<td>AEPE</td>
<td>F1-Noc</td>
<td></td>
<td>AEPE</td>
<td>F1-all</td>
<td>F1-all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>train</td>
<td>test</td>
<td>test</td>
</tr>
<tr>
<td>EpicFlow [34]</td>
<td>-</td>
<td>3.8</td>
<td>7.88%</td>
<td>-</td>
<td>-</td>
<td>26.29%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FullFlow [9]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>23.37%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PatchBatch [11]</td>
<td>-</td>
<td>3.3</td>
<td>5.29%</td>
<td>-</td>
<td>-</td>
<td>21.07%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlowFields [2]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>19.80%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCFlow [45]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>15.09%</td>
<td>14.83%</td>
<td></td>
</tr>
<tr>
<td>MirrorFlow [20]</td>
<td>-</td>
<td>2.6</td>
<td>4.38%</td>
<td>-</td>
<td>-</td>
<td>9.93%</td>
<td>10.29%</td>
<td></td>
</tr>
<tr>
<td>PRSM [42]</td>
<td>-</td>
<td>1.0</td>
<td>2.46%</td>
<td>-</td>
<td>-</td>
<td></td>
<td>6.68%</td>
<td></td>
</tr>
<tr>
<td>SpyNet-ft [33]</td>
<td>(4.13)</td>
<td>4.7</td>
<td>12.31%</td>
<td>-</td>
<td>-</td>
<td>35.07%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlowNet2 [21]</td>
<td>4.09</td>
<td>-</td>
<td>-</td>
<td>10.06</td>
<td>30.37%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlowNet2-ft [21]</td>
<td>(1.28)</td>
<td>1.8</td>
<td>4.82%</td>
<td>(2.30)</td>
<td>(8.61%)</td>
<td>10.41%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LiteFlowNet [19]</td>
<td>4.25</td>
<td>-</td>
<td>-</td>
<td>10.46</td>
<td>29.30%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LiteFlowNet [19]</td>
<td>(1.26)</td>
<td>1.7</td>
<td>-</td>
<td>(2.16)</td>
<td>(8.16%)</td>
<td>10.24%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWC-Net [40]</td>
<td>4.14</td>
<td>-</td>
<td>-</td>
<td>10.35</td>
<td>33.67%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWC-Net-ft [40]</td>
<td>(1.45)</td>
<td>1.7</td>
<td>4.22%</td>
<td>(2.16)</td>
<td>(9.80%)</td>
<td>9.60%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HD$^3$F (Ours)</td>
<td>4.65</td>
<td>-</td>
<td>-</td>
<td>13.17</td>
<td>23.99%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HD$^3$F-ft (Ours)</td>
<td>(0.81)</td>
<td>1.4</td>
<td>2.26%</td>
<td>(1.31)</td>
<td>(4.10%)</td>
<td>6.55%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- “-ft” means fine-tuning on the KITTI training set.
- Numbers in parenthesis are results on data the network has been trained on.
Multiscale predicted flowmaps
Multiscale Error and Confidence maps

Warm color = inaccurate  White = more confident
Flow error map comparison with PWC-Net
Summary

- This approach decomposed the match density into multiple scales.
- Learned the decomposed match densities in an end-to-end manner.
- The predicted match densities can be converted into point estimate.
- Provides model-inherent uncertainty measures.
Thank you!