Boundary Flow: A Siamese Network that Predicts Boundary Motion without Training on Motion

Peng Lei, Fuxin Li and Sinisa Todorovic
Oregon State University
Corvallis, OR 97331, USA
{leip, lif, sinisa}@oregonstate.edu

Abstract

Using deep learning, this paper addresses the problem of joint object boundary detection and boundary motion estimation in videos, which we named boundary flow estimation. Boundary flow is an important mid-level visual cue as boundaries characterize objects’ spatial extents, and the flow indicates objects’ motions and interactions. Yet, most prior work on motion estimation has focused on dense object motion or feature points that may not necessarily reside on boundaries. For boundary flow estimation, we specify a new fully convolutional Siamese network (FCSN) that jointly estimates object-level boundaries in two consecutive frames. Boundary correspondences in the two frames are predicted by the same FCSN with a new, unconventional deconvolution approach. Finally, the boundary flow estimate is improved with an edgelet-based filtering. Evaluation is conducted on three tasks: boundary detection in videos, boundary flow estimation, and optical flow estimation. On boundary detection, we achieve the state-of-the-art performance on the benchmark VSB100 dataset. On boundary flow estimation, we present the first results on the Sintel training dataset. For optical flow estimation, we run the recent approach CPM-Flow but on the augmented input with our boundary-flow matches, and achieve significant performance improvement on the Sintel benchmark.

1. Introduction

This paper considers the problem of estimating motions of object boundaries in two consecutive video frames, or simply two images. We call this problem boundary flow (BF) estimation. Intuitively, BF is defined as the motion of every pixel along object boundaries in two images, as illustrated in Fig. 1. A more rigorous definition will be presented in Sec. 3. BF estimation is an important problem. Its solution can be used as an informative mid-level visual cue for a wide range of higher-level vision tasks, including object detection (e.g.,[11]), object proposals (e.g.,[38]), video segmentation (e.g.,[21]), and depth prediction (e.g., [1]). This is because, in a BF, the boundaries identify objects’ locations, shapes, motions, local interactions, and figure-ground relationships. In many object-level tasks, BF can be computed in lieu of the regular optical flow, hence avoiding estimating motion on many irrelevant background pixels that may not be essential to the performance of the task.

Yet, this problem has received scant attention in the literature. Related work has mostly focused on single-frame edge detection and dense optical flow estimation. These approaches, however, cannot be readily applied to BF estimation, due to new challenges. In particular, low-level spatiotemporal boundary matching — which is agnostic
of objects, scenes, and motions depicted in the two video frames — is subject to many ambiguities. The key challenge is that distinct surfaces sharing a boundary move with different motions, out-of-plane rotations and changing occlusions. This makes appearance along the boundary potentially inconsistent in consecutive frames. The difficulty of matching boundaries in two images also increases when multiple points along the boundary have similar appearance.

Our key hypothesis is that because of the rich visual cues along the boundaries, BF may be learned without pixel-level motion annotations, which is typically very hard to come by (prior work resorts to simulations [24] or computer graphics [8], which may not represent realistic images).

While there are a few approaches that separately detect and match boundaries in a video, e.g., [22, 31, 32], to the best of our knowledge, this is the first work that gives a rigorous definition of boundary flow, as well as jointly detects object boundaries and estimates their flow within the deep learning framework. We extend ideas from deep boundary detection approaches in images [34, 36], and specify a new Fully Convolutional Siamese encoder-decoder Network (FCSN) for joint spatiotemporal boundary detection and BF estimation. As shown in Fig. 2, FCSN encodes two consecutive video frames into a coarse joint feature representation (JFR) (marked as a green cube in Fig. 2). Then, a Siamese decoder uses deconvolution and un-max-pooling to estimate boundaries in each of the two input images.

Our network trains only on boundary annotations in one frame and predicts boundaries in each frame, so at first glance it does not provide motion estimation. However, the Siamese network is capable of predicting different (but correct) boundaries in two frames, while the only difference in the two decoder branches are max-pooling indices. Thus, our key intuition is that there must be a common edge representation in the JFR layer for each edge, that are mapped to two different boundary predictions by different sets of max-pooling indices. Such a common representation enables us to match the corresponding boundaries in the two images. The matching is done by tracking a boundary from one boundary prediction image back to the JFR, and then from the JFR to boundaries in the other boundary prediction image. This is formalized as an excitation attention-map estimation of the FCSN. We use edgelet-based matching to further improve the smoothness and enforce ordering of pixel-level boundary matching along an edgelet.

Since FCSN performs boundary detection and provides correspondence scores for boundary matching, we say that FCSN unifies both boundary detection and BF estimation within the same deep architecture. In our experiments, this approach proves capable of handling large object displacements in the two images, and thus can be used as an important complementary input to dense optical flow estimation.

We evaluate FCSN on the VSB100 dataset [12] for boundary detection, and on the Sintel training dataset [8] for BF estimation. Our results demonstrate that FCSN yields higher precision on boundary detection than the state of the art, and using the excitation attention score for boundary matching yields superior BF performance relative to reasonable baselines. Also, experiments performed on the Sintel test dataset show that we can use the BF results to augment the input of a state-of-the-art optical flow algorithm – CPM-Flow [15] – and generate significantly better dense optical flow than the original.

Our key contributions are summarized below:

- We consider the problem of BF estimation within the deep learning framework, give a rigorous definition of BF, and specify and extensively evaluate a new deep architecture FCSN for solving this problem. We also demonstrate the utility of BF for estimating dense optical flow.
- We propose a new approach to generate excitation-based correspondence scores from FCSN for boundary matching, and develop an edgelet-based matching for refining point matches along corresponding boundaries.
- We improve the state-of-the-art on spatiotemporal boundary detection, provide the first results on BF estimation, and achieve competitive improvements on dense optical flow when integrated with CPM-Flow [15].

2. Related Work

This section reviews closely related work on boundary detection and dense optical flow estimation. The literature on semantic video segmentation and semantic contour detection is beyond our scope.

Boundary Detection. Traditional approaches to boundary detection typically extract a multitude of hand-designed features at different scales, and pass them to a detector for boundary detection [2]. Some of these methods leverage the structure of local contour maps for fast edge detection [10]. Recent work resorts to convolutional neural networks (CNN) for learning deep features that are suitable for boundary detection [13, 29, 6, 5, 34, 36, 23]. [18] trains a boundary detector on a video dataset and achieved improved results. Their network is defined on a single frame and does not provide motion information across two frames. The approach of [36] is closest to ours, since they use a fully convolutional encoder-decoder for boundary detection on one frame. However, without a Siamese network their work cannot be used to estimate boundary motion as proposed in this paper.

Optical Flow Estimation. There has been considerable efforts to improve the efficiency and robustness of optical flow estimation, including PatchMatch [4] and extensions [20, 14, 3]. They compute the Nearest Neighbor Field (NNF) by random search and propagation. EpicFlow [28] uses DeepMatching [33] for a hierarchical matching of image
patches, and its extension Coarse-to-fine Patch-Match (CPM-Flow) [15] introduces a propagation between levels of the hierarchical matching. While EpicFlow [28] propagates optical flow to image boundaries, it still does not handle very abrupt motions well, as can be seen in many of the fast-moving objects in the Sintel benchmark dataset. In this paper, we do not focus on dense optical flow estimation, but demonstrate the capability of boundary flow estimation in supplementing optical flow, which is beneficial in large displacements and flow near boundaries. As our results show, we improve CPM-Flow when using our boundary flow estimation as a pre-processing step. Boundary motion estimation was first considered in [22], and then in [35] where dense optical flow was initialized from an optical flow computed on Canny edges. However, in both of these papers, the definition of their edge flow differs from our boundary flow in the following. First, they do not consider cases when optical flow is not defined. Second, they do not have a deep network to perform boundary detection. Finally, they do not evaluate edge flow as a separate problem.

3. Boundary Flow

This section defines BF, introduces the FCSN, and specifies finding boundary correspondences in the two frames using the FCSN’s excitation attention score.

3.1. Definition of Boundary Flow

BF is defined as the motion of every boundary pixel towards the corresponding boundary pixel in the next frame. In the case of out-of-plane rotations and occlusions, BF identifies the occlusion boundary closest to the original boundary pixel (which becomes occluded). We denote the set of boundaries in frame $t$ and $t + 1$ as $B_1$ and $B_2$, respectively. Let $OF(x)$ denote the optical flow of a pixel $x$ in frame $t$, and $x + OF(x)$ represent a mapping of pixel $x$ in frame $t + 1$. Boundary flow $BF(x)$ is defined as:

(i) $BF(x) = \arg\min_{y \in B_2} \|y - (x + OF(x))\|_2 - x$, if $OF(x)$ exists;
(ii) $BF(x) = OF(\arg\min_{y \in B_2} \|y - x\|_2)$, if $OF(x)$ does not exist ($x$ occluded in frame $t + 1$);
(iii) $BF(x)$ is undefined if argmin in (i) or (ii) does not return a unique solution.

In (i), BF is defined as optical flow for translations and elastic deformations, or the closest boundary pixel from the optical flow for out-of-plane rotations (see Fig. 3(b)). In (ii), BF is defined as the closest occlusion boundary of the pixel which becomes occluded (see Fig. 3(a)). Thus, BF can be defined even if optical flow is not defined. Since optical flow is often undefined in the vicinity of occlusion boundaries, BF captures shapes/occlusions better than optical flow. In (iii), BF is undefined only in rare cases of fast movements with symmetric occluders (e.g., a perfect ball) resulting in multiple pixels as the argmin solution.

3.2. Fully Convolutional Siamese Network

We formulate boundary detection as a binary labeling problem. For this problem, we develop a new, end-to-end trainable FCSN, shown in Fig. 2. FCSN takes two images as input, and produces binary soft-max outputs of boundary predictions in each of the two input images. The fully con-
volutionary architecture in FCSN scales up to arbitrary image sizes.

FCSN consists of two modules: a Siamese encoder, and a Siamese decoder. The encoder stores all the pooling indices and encodes the two frames as the joint feature representation (JFR) (green box in Fig. 2) through a series of convolution, ReLU, and pooling layers. The outputs of the encoder are concatenated, and then used as the input to the decoder. The decoder takes both the JFR and the max-pooling indices from the encoder as inputs. Then, the features from the decoder are passed into a softmax layer to get the boundary labels of all pixels in the two images.

The two branches of the encoder and the two branches of the decoder use the same architecture and share weights with each other. However, for two different input images, the two branches would still output different predictions, since decoder predictions are modulated with different pooling indices recorded in their corresponding encoder branches. Each encoder branch uses the layers of VGG net [30] until the fifth layer. The decoder decodes the JFR to the original input size through a set of unpooling, deconvolution, ReLU and dropout operations. Unlike the deconvolutional net [27] which uses a symmetric decoder as the encoder, we design a light-weight decoder with fewer weight parameters than a symmetric structure for efficiency. Except for the layer right before the softmax layer, all the other convolution layers of the decoder are followed by a ReLU operator and a dropout layer. A detailed description of the convolution and dropout layers is summarized in Tab. 1.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter</th>
<th>Dropout rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deconv1</td>
<td>1(\times)1(\times)512</td>
<td>0.5</td>
</tr>
<tr>
<td>Deconv2</td>
<td>5(\times)5(\times)512</td>
<td>0.5</td>
</tr>
<tr>
<td>Deconv3</td>
<td>5(\times)5(\times)256</td>
<td>0.5</td>
</tr>
<tr>
<td>Deconv4</td>
<td>5(\times)5(\times)128</td>
<td>0.5</td>
</tr>
<tr>
<td>Deconv5</td>
<td>5(\times)5(\times)64</td>
<td>0.5</td>
</tr>
<tr>
<td>Deconv6</td>
<td>5(\times)5(\times)32</td>
<td>0.5</td>
</tr>
<tr>
<td>Softmax</td>
<td>5(\times)5(\times)1</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: The configuration of the decoder in FCSN.

### 3.3. Boundary Flow Estimation

This section first describes estimation of the excitation attention score, used as a cue for boundary matching, and then specifies our edgelet-based matching for refining point matches along the boundaries.

#### 3.3.1 Excitation Attention Score

A central problem in BF estimation is to identify the correspondence between a pair of boundary points \((x^t_i, y^t_{i+1})\), where \(x^t_i\) is a boundary point in frame \(t\), and \(y^t_{i+1}\) is a boundary point in frame \(t + 1\). Our key idea is to estimate this correspondence by computing the excitation attention scores in frame \(t + 1\) for every \(x^t_i\) in frame \(t\), as well as the excitation attention scores in frame \(t\) for every \(y^t_{i+1}\) in frame \(t + 1\). The excitation attention scores can be generated efficiently using excitation backpropagation (ExcitationBP) [37] – a probabilistic winner-take-all approach that models dependencies of neural activations through convolutional layers of a neural network for identifying relevant neurons for prediction, i.e., attention maps.

The intuition behind our approach is that the JFR stores a joint representation of two corresponding boundaries of the two images, and thus could be used as a “bridge” for matching them. This “bridge” is established by tracking the most relevant neurons along the path from one branch of the decoder to the other branch via the JFR layer (the cyan and purple arrows in Fig. 2).

In our approach, the winner neurons are sequentially sampled for each layer on the path from frame \(t\) to \(t + 1\) via the JFR, based on a conditional winning probability. The relevance of each neuron is defined as its probability of being selected as a winner on the path. Following [37], we define the winning probability of a neuron \(a_m\) as

\[
p(a_m) = \sum_{n \in \mathcal{P}_m} p(a_m | a_n) p(a_n)
\]

where \(w^+_{mn} = max\{0, w_{mn}\}\), \(\mathcal{P}_m\) and \(\mathcal{C}_n\) denote the parent nodes of \(a_m\), and the set of children of \(a_n\) in the path traveling order, respectively. For our path that goes from the prediction back to the JFR layer, \(\mathcal{P}_m\) refers to all neurons in the layer closer to the prediction, and \(\mathcal{C}_n\) refers to all neurons in the layer closer to the JFR layer.

ExcitationBP efficiently identifies which neurons are responsible for the final prediction. In our approach, ExcitationBP can be run in parallel for each edgelet (see next subsection) of a predicted boundary. Starting from boundary predictions in frame \(t\), we compute the marginal winning probability of all neurons along the path to the JFR. Once the JFR is reached, these probabilities are forward-propagated in the decoder branch of FCSN for finally estimating the pixel-wise excitation attention scores in frame \(t + 1\). For a pair of boundary points, we obtain the attention score \(s_{i \rightarrow j}\). Conversely, starting from boundary predictions in frame \(t + 1\), we compute the marginal winning probability of all neurons along the path to JFR, and feed them forward through the decoder for computing the excitation attention map in frame \(t\). Then we can obtain the attention score \(s_{j \rightarrow i}\). The attention score between a pair of boundary points \((x^t_i, y^t_{i+1})\) is defined as the average of \(s_{i \rightarrow j}\) and \(s_{j \rightarrow i}\), which we denote.
as \( s_{ij} \). An example of our ExcitationBP is shown in Fig. 4.

### 3.3.2 Edgelet-based Matching

After estimating the excitation attention scores \( s_{ij} \) of boundary point pairs \((x^i_t, y^j_{t+1})\), as described in Sec. 3.3.1, we use them for matching corresponding boundaries that have been predicted in frames \( t \) and \( t+1 \). While there are many boundary matching methods that would be suitable, in this work we use the edgelet-based matching which not only finds good boundary correspondences, but also produces the detailed point matches along the boundaries, as needed for our BF estimation. To this end, we first decompose the predicted boundaries into smaller edgelets, then apply edgelet-based matching to pairs of edgelets.

**From predicted boundaries to edgelets.** Given the two input images and their boundary predictions from FCSN, we oversegment the two frames using sticky superpixels \([10]\), and merge the superpixels to larger regions as in \([16]\). Importantly, both oversegmentation and superpixel-merging use our boundary predictions as input, ensuring that contours of the resulting regions strictly respect our predicted boundaries, as illustrated in Fig. 5(a). We define an edgelet as all the points that lie on a given boundary shared by a pair of superpixels. Fig. 5(b) shows two examples of matching edgelet pairs in frames \( t \) and \( t+1 \).

**Edgelet matching.** We apply edgelet-based matching to each edgelet pair, \( e_t \) in frame \( t \) and \( e'_{t+1} \) in frame \( t+1 \), that fall within a reasonable spatial neighborhood (empirically set to 100 pixels around the edgelet as sufficient to accommodate for large motions). For each edgelet pair, \( e_t \) in frame \( t \) and \( e'_{t+1} \) in frame \( t+1 \), all the similarities between all the pixel pairs on these edgelet pairs are summed up and normalized to obtain the similarity between the edgelet pair. The similarity between points \((x^i_t, y^j_{t+1})\) on \( e_t \) and \( e'_{t+1} \) is expressed in terms of their respective excitation attention scores as \( s_{ij} \).

For an edgelet \( e_t \) in frame \( t \), we keep the top-10 most similar edgelets in frame \( t+1 \) as its matching candidates. These candidate edgelet pairs are further filtered by their normals, with only edgelets with an angle not more than 45
degrees retained. The normals are computed as the average direction from pixel coordinates on one side of the edge to corresponding pixel coordinates on the other side of the edge. This also helps to determine which superpixel pair falls on the same side of the edge in the two images. As shown in Fig. 5(c), superpixels \( s_1 \) and \( s_1' \) fall on the left side of edges \( e_t \) and \( e_{t+1} \), respectively, thus superpixel pair \( \{s_1, s_1'\} \) fall on the same side of the edges.

After filtering by angle, a greedy matching algorithm is performed to approximate bipartite matching of edgelets in frame \( t \) to edgelets in the frame \( t+1 \). This further reduces the number of edgelet pairs retained.

For the final boundary flow placement, we observe that some boundary points will be placing on the incorrect side of the edgelet. We utilize normalized region similarity defined by color to assign the motion to superpixels pairs that are more similar to each other in color. As shown in Fig. 5(c), point \( x_t \) is on the right side of edge \( e_t \) but the corresponding point \( y_{t+1} \) is on the left side of edge \( e_{t+1} \). Our approach moves \( x_t \) to the other side of \( e_t \), resulting in \( x'_{t+new} \). After moving the points, we obtain pixel-level matches which are the final boundary flow result.

4. Training

FCSN is implemented using Caffe [17]. The encoder weights are initialized with VGG-16 net and fixed during training. We update only the decoder parameters using the Adam method [19] with learning rate \( 10^{-4} \). We train on VSB100, a state-of-the-art video object boundary dataset, which contains 40 training videos with annotations on every 20-th frame, for a total of 240 annotated frames. Because there are too few annotations, we augment the training with the PASCAL VOC 12 dataset, which contains 10582 still images (with refined object-level annotations as in [36]). In each iteration, 8 patches with size \( 224 \times 224 \) are randomly sampled from an image pair of VSB100 (or two duplicated frames of PASCAL VOC) and passed to the model.

The loss function is specified as the weighted binary cross-entropy loss common in boundary detection [34].
Table 2: Results on VSB100.

<table>
<thead>
<tr>
<th>Method</th>
<th>ODS</th>
<th>OIS</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEDN [36]</td>
<td>0.563</td>
<td>0.614</td>
<td>0.547</td>
</tr>
<tr>
<td>FCSN</td>
<td>0.597</td>
<td>0.632</td>
<td>0.566</td>
</tr>
</tbody>
</table>

Table 3: Results on VSB100 with fine-tuning on both BSDS500 and VSB100 training sets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EPE</td>
<td>23.158</td>
<td>20.874</td>
<td>25.476</td>
<td>9.856</td>
</tr>
</tbody>
</table>

Table 4: Quantitative results of boundary flow on Sintel training dataset in EPE metric.

F-score of FCSN is 0.60 while 0.56 for CEDN. FCSN yields higher precision than CEDN, and qualitatively we observe that FCSN generates visually cleaner object boundaries. As shown in Fig. 6, CEDN misses some of the boundaries of background objects, but our FCSN is able to detect them. Due to limited training data, both FCSN and CEDN obtain relatively low recall. Tab. 2 shows that FCSN outperforms CEDN in terms of the optimal dataset scale (ODS), optimal image scale (OIS), and average precision (AP).

**5. Results**

This section presents our evaluation of boundary detection, BF estimation, and utility of BF for optical flow estimation.

**5.1. Boundary Detection**

After FCSN generates boundary predictions, we apply the standard non-maximum suppression (NMS). The resulting boundary detection is evaluated using precision-recall (PR) curves and F-measure.

**VSB100.** For the benchmark VSB100 test dataset [12], we compare with the state-of-the-art approach CEDN [36]. We train both FCSN and CEDN using the same training data with 30000 iterations. Note that CEDN is single-frame based. Nevertheless, both FCSN and CEDN use the same level of supervision, since only isolated single frame annotations apart from one another are available. Fig. 7a shows the PR-curves of object boundary detection. As can be seen, the two branches still can output different predictions, since decoder predictions are modulated with different pooling indices recorded in the corresponding encoder branches. Due to the imbalances of boundary pixels and non-boundary pixels, we set $\lambda_1$ to 1 and $\lambda_2$ to 0.1, respectively.

**5.2. Boundary Flow Estimation**

Boundary flow accuracies are evaluated by average end-point error (EPE) between our boundary flow prediction and the ground truth boundary flow (as defined in Sec. 3.1) on the Sintel training dataset.

In order to identify a good competing approach, we have tested a number of the state-of-the-art matching algorithms on the Sintel training dataset, including coarse-to-fine PatchMatch (CPM) [15], Kd-tree PatchMatch [14] and Deep-Matching [33], but have found that these algorithms are not suitable for our comparison because they prefer to find point matches off boundaries.

![Figure 8](image-url)
Therefore, we compare our edgelet-based matching algorithm with the following baselines: (i) greedy nearest-neighbor point-to-point matching, (ii) RANSAC [7], (iii) FLANN, a matching method that uses SIFT features. The quantitative results are summarized in Table 4. Our edgelet-based matching outperforms all the baselines significantly.

5.3. Dense Optical Flow Estimation

We also test the utility of our approach for optical flow estimation on the Sintel testing dataset. After running our boundary flow estimation, the resulting boundary matches are used to augment the standard input to the state of the art CPM-Flow [15], as shown in Fig. 8. Such an approach is denoted as CPM-AUG, and compared with the other existing methods in Table 5. As can be seen, CPM-AUG outperforms CPM-Flow and FlowFields. Note the results we submitted on "Sintel clean" under the name CPM-AUG was not actually results of CPM-AUG, actually it was just our implementation of CPMFlow[15], which is a bit lower than the public one on the Sintel dataset. However these are the best results we can obtain using the public implementation of the algorithm. In principle, the augmented point matches should be able to help other optical flow algorithms as well as it is largely orthogonal to the information pursued by current optical flow algorithms.

Fig. 9 shows qualitative results of CPM-AUG on Sintel testing dataset with comparison to two state-of-the-art methods: CPM-Flow and EpicFlow. As it can be seen, CPM-AUG performs especially well on the occluded areas and benefits from the boundary flow to produce sharp motion boundaries on small objects like the leg and the claws as well as the elongated halberd.

<table>
<thead>
<tr>
<th>Method</th>
<th>EPE all</th>
<th>EPE matched</th>
<th>EPE unmatched</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPM-AUG</td>
<td><strong>5.645</strong></td>
<td>2.737</td>
<td><strong>29.362</strong></td>
</tr>
<tr>
<td>CPM-Flow[15]</td>
<td>5.960</td>
<td>2.990</td>
<td>30.177</td>
</tr>
</tbody>
</table>

Table 5: Quantitative results on Sintel final test set.

Acknowledgement. This work was supported in part by DARPA XAI Award N66001-17-2-4029.

References

[1] M. R. Amer, S. Yousefi, R. Raich, and S. Todorovic. Monocular extraction of 2.1 d sketch using constrained convex...


