

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

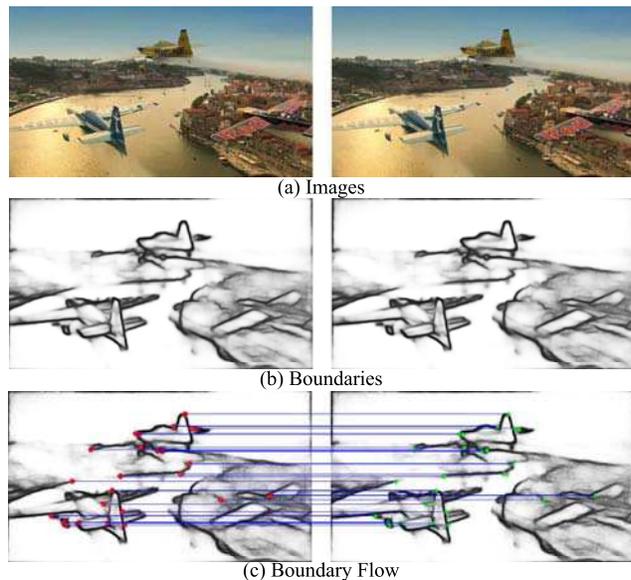


Figure 1: Boundary flow estimation. Given two images (a), our approach *jointly* predicts object boundaries in both images (b), and estimates motion of the boundaries in the two images (c). For clarity, only a part of boundary matches are shown in (c).

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 bound-

ary 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

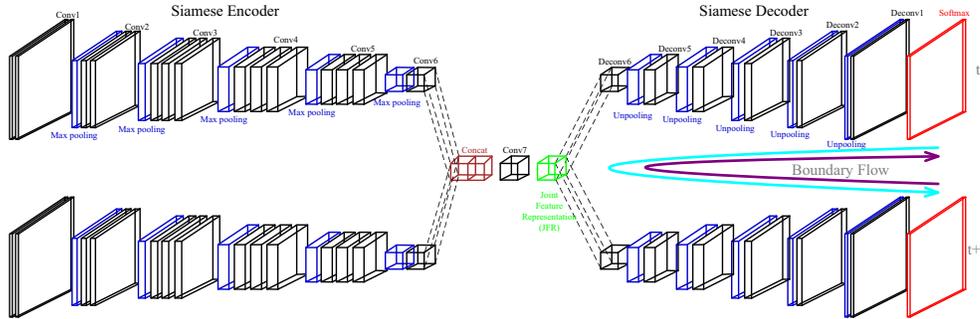


Figure 2: FCSN consists of a Siamese encoder and a Siamese decoder and takes two images as input. The two Siamese soft-max outputs of the decoder produce boundary predictions in each of the two input images. Also, the decoder associates the two Siamese branches via the decoder layers and the JFR layer (the green cube) for calculating the excitation attention score, which in turn is used for BF estimation, as indicated by the cyan and purple arrows. The convolution, pooling, softmax and concatenation layers are marked with black, blue, red and brown respectively. Best viewed in color.

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 $\text{OF}(\mathbf{x})$ denote the optical flow of a pixel \mathbf{x} in frame t , and $\mathbf{x} + \text{OF}(\mathbf{x})$ represent a mapping of pixel \mathbf{x} in frame $t + 1$. Boundary flow $\text{BF}(\mathbf{x})$ is defined as:

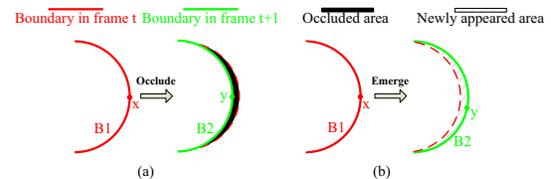


Figure 3: Fig. 3(a) shows the case when a boundary B_1 in frame t is occluded at time $t + 1$. Fig. 3(b) shows the case when a boundary B_1 in frame t is no longer a boundary at time $t + 1$ but its pixels are visible. In both cases BF is well-defined and always resides on the boundary.

- (i) $\text{BF}(\mathbf{x}) = \arg \min_{\mathbf{y} \in B_2} \|\mathbf{y} - (\mathbf{x} + \text{OF}(\mathbf{x}))\|_2 - \mathbf{x}$, if $\text{OF}(\mathbf{x})$ exists;
- (ii) $\text{BF}(\mathbf{x}) = \text{OF}(\arg \min_{\mathbf{y} \in \text{OF}(\mathbf{y})} \|\mathbf{y} - \mathbf{x}\|_2)$, if $\text{OF}(\mathbf{x})$ does not exist (\mathbf{x} occluded in frame $t + 1$);
- (iii) $\text{BF}(\mathbf{x})$ is undefined if $\arg \min$ 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 $\arg \min$ 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-

volutional 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 *fc6* 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.

Layer	Filter	Dropout rate
Deconv1	$1 \times 1 \times 512$	0.5
Deconv2	$5 \times 5 \times 512$	0.5
Deconv3	$5 \times 5 \times 256$	0.5
Deconv4	$5 \times 5 \times 128$	0.5
Deconv5	$5 \times 5 \times 64$	0.5
Deconv6	$5 \times 5 \times 32$	0.5
Softmax	$5 \times 5 \times 1$	-

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 $\langle \mathbf{x}_t^i, \mathbf{y}_{t+1}^j \rangle$,

where \mathbf{x}_t^i is a boundary point in frame t , and \mathbf{y}_{t+1}^j 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 \mathbf{x}_t^i in frame t , as well as the excitation attention scores in frame t for every \mathbf{y}_{t+1}^j 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

$$\begin{aligned}
 p(a_m) &= \sum_{n \in \mathcal{P}_m} p(a_m | a_n) p(a_n) \\
 &= \sum_{n \in \mathcal{P}_m} \frac{w_{mn}^+ a_m}{\sum_{m' \in \mathcal{C}_n} w_{m'n}^+ a_{m'}} p(a_n)
 \end{aligned} \tag{1}$$

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 $\langle \mathbf{x}_t^i, \mathbf{y}_{t+1}^j \rangle$ is defined as the average of $s_{i \rightarrow j}$ and $s_{j \rightarrow i}$, which we denote

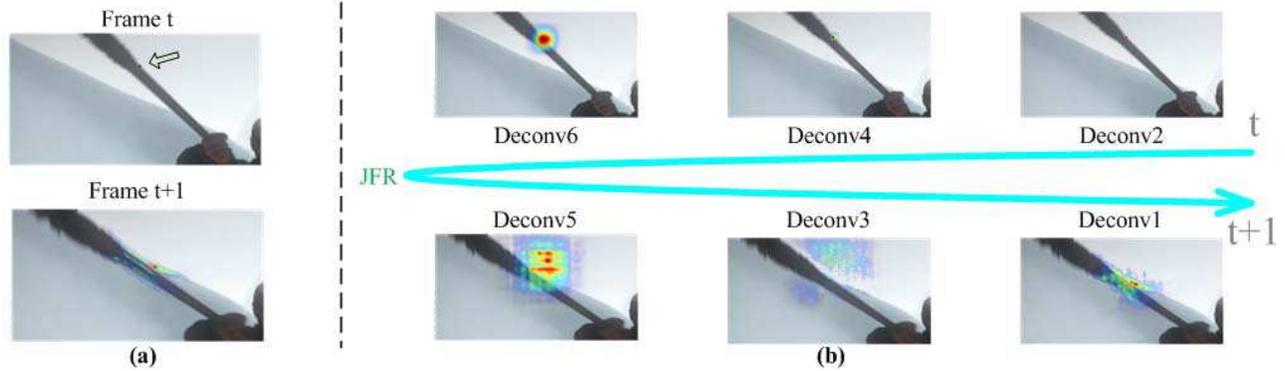


Figure 4: (a) Estimation of the excitation attention score in frame $t + 1$ (bottom) for a particular boundary point in frame t (top; the point is indicated by the arrow). The attention map is well-aligned with the corresponding boundary in frame $t + 1$, despite large motion. (b) Visualization of attention maps at different layers of the decoders of FCSN along the excitation path (cyan) from a particular boundary point in frame t to frame $t + 1$ via the JFR. For simplicity, we only show the attention maps in some of the layers from the decoder branch at time t and $t + 1$. As can be seen, starting from a pixel on the predicted boundary in frame t , the attention map gradually becomes coarser along the path to the JFR. Then from the JFR to boundary prediction in frame $t + 1$, the excitation attention scores gradually become refined and more focused on the most relevant pixels in frame $t + 1$. (Best viewed in color)

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 $\langle \mathbf{x}_t^i, \mathbf{y}_{t+1}^j \rangle$, 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

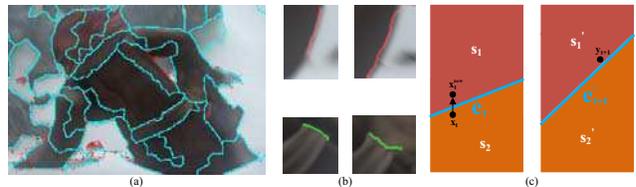


Figure 5: Overview of edgelet matching. The matching process consists of three phases: superpixel generation, edgelet matching, and flow placement. The two frames are first oversegmented into large superpixels using the FCSN boundaries. (a) most of the boundary points (in red color) are well aligned with the superpixel boundaries (in cyan color); (b) Example edgelet matches. In the second case, it can be seen clearly that the appearance only matches on one side of the edgelet. (c) The process of matching and flow placement. Sometimes, because of the volatility of edge detection, \mathbf{x}_t and \mathbf{y}_{t+1} falls on different sides of the boundary, we will need to then move \mathbf{x}_t so that they fall on the same side. Note that s_1 and s_2 , s'_1 and s'_2 denote the superpixel pairs falling on the two sides of the edgelets.

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 $\langle \mathbf{x}_t^i, \mathbf{y}_{t+1}^j \rangle$ 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

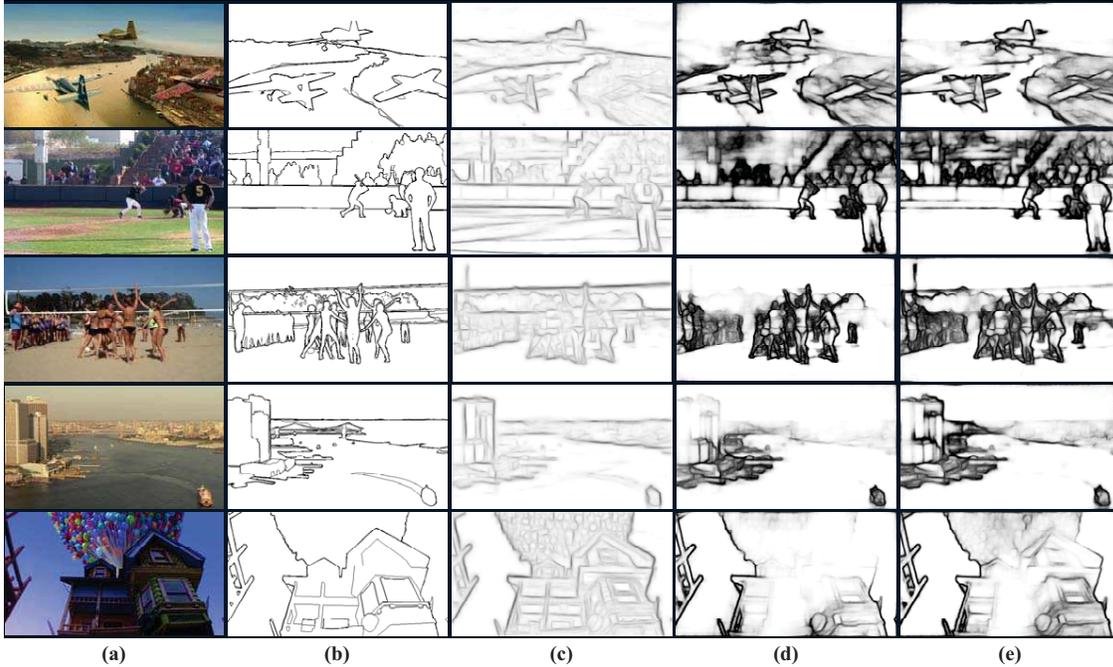


Figure 6: Example results on VSB100. In each row from left to right we present (a) input image, (b) ground truth annotation, (c) edge detection [10], (d) object contour detection [36] and (e) our boundary detection.

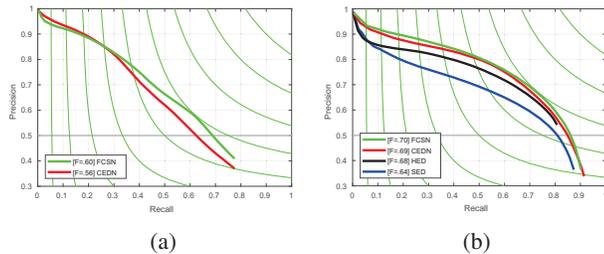


Figure 7: (a) PR curve for object boundary detection on VSB100. (b) PR curve for object boundary detection on VSB100 with fine-tuning on both BSDS500 and VSB100 training sets.

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 224×224 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]

Method	ODS	OIS	AP
CEDN [36]	0.563	0.614	0.547
FCSN	0.597	0.632	0.566

Table 2: Results on VSB100.

Method	ODS	OIS	AP
SE [10]	0.643	0.680	0.608
HED [34]	0.677	0.715	0.618
CEDN [36]	0.686	0.718	0.687
FCSN	0.698	0.729	0.705

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

Method	FLANN [26]	RANSAC [7]	Greedy	Our Matching
EPE	23.158	20.874	25.476	9.856

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

$L(w) = -\frac{1}{N} \sum_{i=1}^N [\lambda_1 y_n \log \hat{y}_n + \lambda_2 (1-y_n) \log(1-\hat{y}_n)]$ where N is the number of pixels in an iteration. Note that the loss is defined on a single side of the outputs, since only single frame annotations are available. The two decoder branches share the same architecture and weights, and thus can be both updated simultaneously with our one-side loss. 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 λ_1 to 1 and λ_2 to 0.1, respectively.

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,

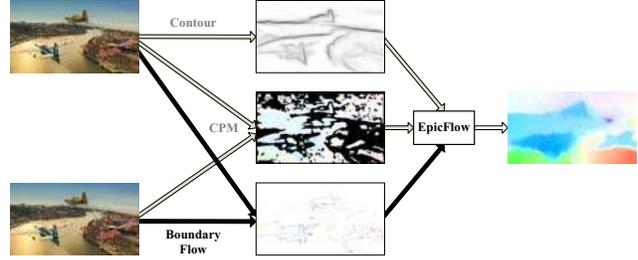


Figure 8: Overview of augmenting boundary flow into the framework of CPM-Flow. Given two images, we compute the standard input to CPM-Flow: matches using CPM matching [15] and the edges of the first image using SE [10]. Then we augment the matches with our predicted boundary flow (i.e., matches on the boundaries), as indicated by black arrows.

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).

Finetuning on BSDS500 and VSB100. We also evaluate another training setting when FCSN and CEDN are both fine-tuned on the BSDS500 training dataset [2] and VSB100 training set for 100 epochs with learning rate 10^{-5} . BSDS has more edges annotated hence allows for higher recall. Such trained FCSN and CEDN are then compared with the state-of-the-art, including structured edge detection (SE) [10], and holistically-nested edge detection algorithm (HED) [34]. Both SE and HED are re-trained with the same training setting as ours. Fig. 7b and Tab. 3 present the PR-curves and AP. As can be seen, FCSN outperforms CEDN, SE and HED in all metrics. We presume that further improvement may be obtained by training with annotated boundaries in both frames.

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-art matching algorithms on the Sintel training dataset, including coarse-to-fine PatchMatch (CPM) [15], Kd-tree PatchMatch [14] and DeepMatching [33], but have found that these algorithms are not suitable for our comparison because they prefer to find point matches off boundaries.

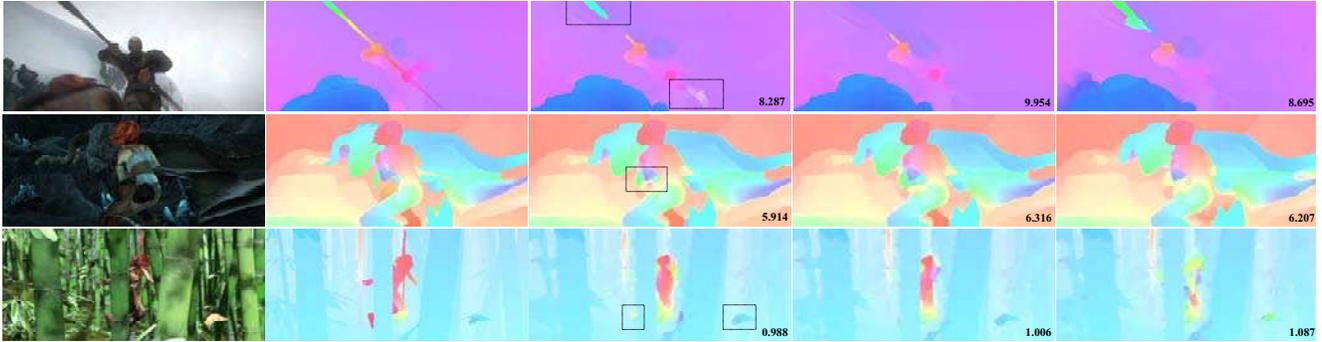


Figure 9: Example results on MPI-Sintel test dataset. The columns correspond to original images, ground truth, CPM-AUG (i.e., our approach), CPM-Flow [15] and EpicFlow[28]. The rectangles highlight the improvements and the numbers indicate the EPEs.

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 Tab. 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 Tab. 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.

6. Conclusion

We have formulated the problem of boundary flow estimation in videos. For this problem, we have specified a new end-to-end trainable FCSN which takes two images as input

Method	EPE all	EPE matched	EPE unmatched
CPM-AUG	5.645	2.737	29.362
FlowFields[3]	5.810	2.621	31.799
Full Flow[9]	5.895	2.838	30.793
CPM-Flow[15]	5.960	2.990	30.177
DiscreteFlow[25]	6.077	2.937	31.685
EpicFlow[28]	6.285	3.060	32.564

Table 5: Quantitative results on Sintel final test set.

and produces boundary detections in each image. We have also used FCSN to generate excitation attention maps in the two images as informative features for boundary matching, thereby unifying detection and flow estimation. For matching points along boundaries, we have decomposed the predicted boundaries into edgelets and applied edgelet-based matching to pairs of edgelets from the two images. Our experiments on the benchmark VSB100 dataset for boundary detection demonstrate that FCSN is superior to the state-of-the-art, succeeding in detecting boundaries both of foreground and background objects. We have presented the first results of boundary flow on the benchmark Sintel training set, and compared with reasonable baselines. The utility of boundary flow is further demonstrated by integrating our approach with the CPM-Flow for dense optical flow estimation. This has resulted in an improved performance over the original CPM-Flow, especially on small details, sharp motion boundaries, and elongated thin objects in the optical flow.

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

- optimization. *International Journal of Computer Vision*, 112(1):23–42, 2015. 1
- [2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011. 2, 7
- [3] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*, 2015. 2, 8
- [4] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009. 2
- [5] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015. 2
- [6] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *ICCV*, 2015. 2
- [7] M. Brown and D. G. Lowe. Recognising panoramas. In *ICCV*, 2003. 7, 8
- [8] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 2
- [9] Q. Chen and V. Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *CVPR*, 2016. 8
- [10] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2015. 2, 5, 6, 7
- [11] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 30(1):36–51, 2008. 1
- [12] F. Galasso, N. Nagaraja, T. Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 2, 7
- [13] Y. Ganin and V. Lempitsky. \mathcal{N}^4 -fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, 2014. 2
- [14] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012. 2, 7
- [15] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *CVPR*, 2016. 2, 3, 7, 8
- [16] A. Humayun, F. Li, and J. M. Rehg. The middle child problem: Revisiting parametric min-cut and seeds for object proposals. In *ICCV*, 2015. 5
- [17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6
- [18] A. Khoreva, R. Benenson, F. Galasso, M. Hein, and B. Schiele. Improved image boundaries for better video segmentation. In *ECCV 2016 Workshops*, 2016. 2
- [19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6
- [20] S. Korman and S. Avidan. Coherency sensitive hashing. In *ICCV*, 2011. 2
- [21] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011. 1
- [22] C. Liu, W. T. Freeman, and E. H. Adelson. Analysis of contour motions. In *NIPS*, pages 913–920, 2006. 2, 3
- [23] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. In *ECCV*, 2016. 2
- [24] 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 *CVPR*, 2016. 2
- [25] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *GCPR*, 2015. 8
- [26] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331–340):2, 2009. 7
- [27] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 4
- [28] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 2, 3, 8
- [29] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015. 2
- [30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4
- [31] W. B. Thompson. Exploiting discontinuities in optical flow. *International Journal of Computer Vision*, 30(3):163–173, 1998. 2
- [32] W. B. Thompson, K. M. Mutch, and V. A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4):374–383, 1985. 2
- [33] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 3, 7
- [34] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 2, 6, 7
- [35] T. Xue, M. Rubinstein, C. Liu, and W. T. Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)*, 34(4):79, 2015. 3
- [36] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016. 2, 6, 7
- [37] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *ECCV*, 2016. 4
- [38] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1