This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Unsupervised Learning of Edges

Yin Li[1,2], Manohar Paluri[1], James M. Rehg[2], and Piotr Dollár[1]

[1]Facebook AI Research (FAIR)
[2]Georgia Institute of Technology

Abstract

Data-driven approaches for edge detection have proven effective and achieve top results on modern benchmarks. However, all current data-driven edge detectors require manual supervision for training in the form of hand-labeled region segments or object boundaries. Specifically, human annotators mark semantically meaningful edges which are subsequently used for training. Is this form of strong, high-level supervision actually necessary to learn to accurately detect edges? In this work we present a simple yet effective approach for training edge detectors without human supervision. To this end we utilize motion, and more specifically, the only input to our method is noisy semi-dense matches between frames. We begin with only a rudimentary knowledge of edges (in the form of image gradients), and alternate between improving motion estimation and edge detection in turn. Using a large corpus of video data, we show that edge detectors trained using our unsupervised scheme approach the performance of the same methods trained with full supervision (within 3-5%). Finally, we show that when using a deep network for the edge detector, our approach provides a novel pre-training scheme for object detection.

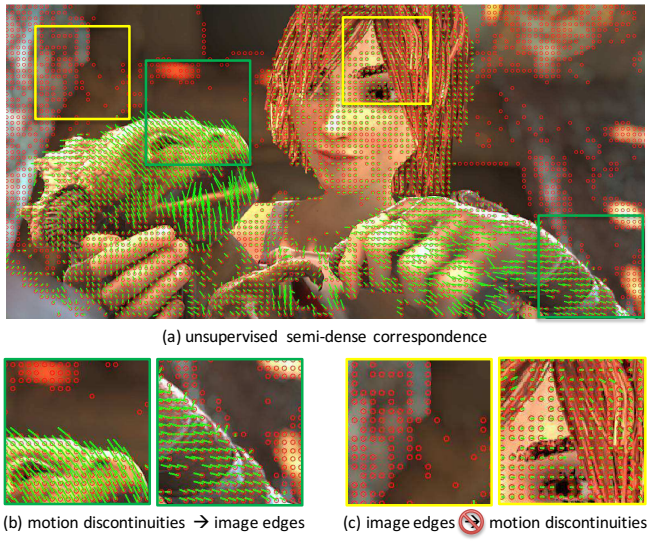


Figure 1. Our goal is to train an edge detector given only semi-dense matches between frames (a). While motion discontinuities imply the presence of image edges (b), the converse is not necessarily true as distinct image regions may undergo similar motion (c). In this work we exploit the sparsity of edges to overcome the latter difficulty. We show that the signal obtained from matches computed over a large corpus of video data is sufficient to train top-performing edge detectors.

1. Introduction

The human visual system can easily identify perceptually salient edges in an image. Endowing machine vision systems with similar capabilities is of interest as edges are useful for diverse tasks such as optical flow [31], object detection [40, 13], and object proposals [39, 46, 3]. However, edge detection has proven challenging. Early approaches [14, 8, 15] relied on low-level cues such as brightness and color gradients. Reasoning about texture [25] markedly improved results, nevertheless, accuracy still substantially lagged human performance.

The introduction of the BSDS dataset [2], composed of human annotated region boundaries, laid the foundations for a fundamental shift in edge detection. Rather than rely on complex hand-designed features, Dollár et al. [10] proposed a data-driven, supervised approach for learning to detect edges. Modern edge detectors have built on this idea and substantially pushed the state-of-the-art forward using more sophisticated learning paradigms [30, 23, 11, 44].

However, existing data-driven methods require strong supervision for training. Specifically, in datasets such as BSDS [2], human annotators use their knowledge of scene structure and object presence to mark semantically meaningful edges.[1] Moreover, recent edge detectors use ImageNet pre-training [5, 44]. In this paper, we explore whether this is necessary: *Is object-level supervision indispensable for edge detection? Moreover, can edge detectors be trained entirely without human supervision?*

[1]Human annotation of edge structure in local patches (without context) is quite noisy and is matched by machine vision approaches. Humans excel when given context and the ability to reason about object presence [47].

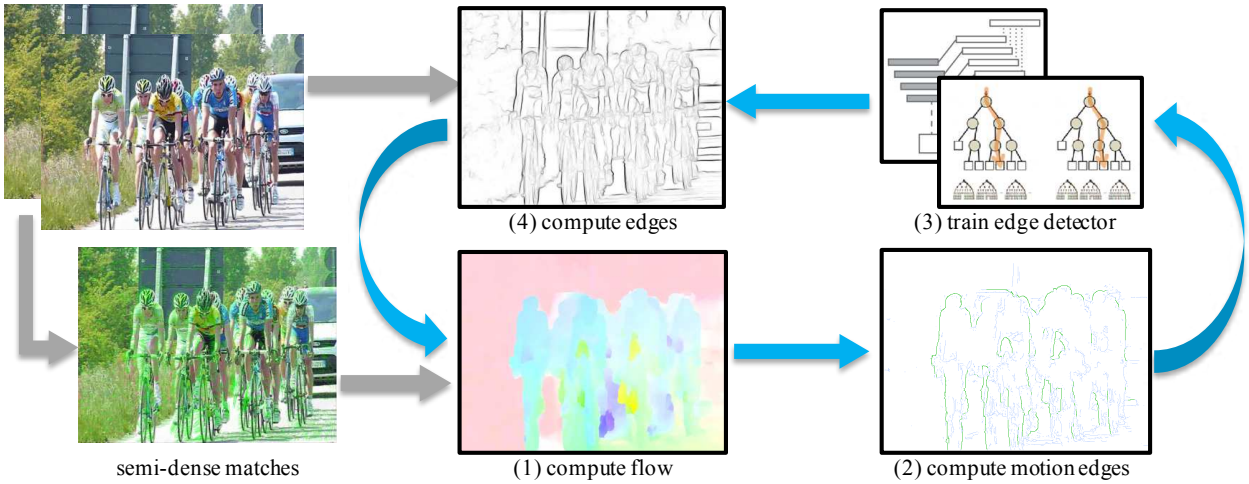


Figure 2. The only input to our approach is semi-dense matching results from [42]. During training we alternate between: (1) computing flow based on the matches and edge maps (initialized to simple gradients), (2) computing motion edges from the flow fields (green: positive edge samples; blue: discarded motion edges), (3) training an edge detector using the motion edges as supervision, and (4) recomputing image edges using the new detector. The process is iterated on a large corpus of videos leading to increasingly accurate flow and edges.

We propose to train edge detectors using motion in place of human supervision. Motion edges are a subset of image edges, see Figure 1. Therefore motion edges can be used to harvest positive training samples. On the other hand, locations away from motion edges may also contain image edges. Fortunately, as edges are sparse, simply sampling such locations at random can provide good negative training data with few false negatives. Thus, assuming accurate motion estimates, we can potentially harvest unlimited training data for edge detection.

While it would be tempting to assume access to accurate motion estimates, this is arguably an unreasonably strong requirement. Indeed, optical flow and edge detection are tightly coupled. Recently, Revaud et al. proposed EpicFlow [31]: given an accurate edge map [11] and semi-dense matches between frames [42], EpicFlow generates a dense edge-respecting interpolation of the matches. The result is a state-of-the-art optical flow estimate.

This motivates our approach. We begin with only semi-dense matches between frames [42] and a rudimentary knowledge of edges (simple image gradients). We then repeatedly alternate between computing flow based on the matches and most recent edge maps and retraining an edge detector based on signal obtained from the flow fields. Specifically, at each iteration, we first estimate dense flow fields by interpolating the matching results using the edge maps obtained from the previous iteration. Given a large corpus of videos, we next harvest highly confident motion edges as positives and randomly sample negatives, and use this data to train an improved edge detector. The process is iterated leading to increasingly accurate flow and edges. An overview of our method is shown in Figure 2.

We perform experiments with the Structured Edge (SE) [11] and Holistic Edge (HE) [44] detectors. SE is based on structured forests, HE on deep networks; SE is faster, HE more accurate. Both detectors achieve state-of-the-art results. The main result of our paper is that both methods, trained using our unsupervised scheme, approach the level of performance of fully supervised training.

Finally, we demonstrate that our approach can serve as a novel unsupervised pre-training scheme for deep networks [41, 9]. Specifically, we show that when fine-tuning a network for object detection [12], starting with the weights learned for edge detection improves performance over starting with a network with randomly initialized weights. While the gains are modest, we believe this is a promising direction for future exploration.

2. Related Work

Edge Detection: Early edge detectors were manually designed to use image gradients [14, 8, 15] and later texture gradients [2]. Of more relevance to this work are edge detectors trained in a data-driven manner. Since the work of [10], which formulated edge detection as a binary classification problem, progressively more powerful learning paradigms have been employed, including multi-class classification [23], feature learning [30], regression [35], structured prediction [11], and deep learning [20, 5, 44]. Recently, Weinzaepfel et al. [43] extended [11] to motion edge estimation. These methods all require strong supervision for training. In this work we explore whether unsupervised learning can be used instead (and as discussed select [11, 44] for our experiments).

Optical Flow: The estimation of optical flow is a classic problem in computer vision [18, 24]. A full overview is out-

side of our scope, instead, our work is most closely related to methods that leverage sparse matches or image edges for flow estimation [6, 42, 31]. In particular, as in [31], we use edge-respecting sparse-to-dense interpolation of matches to obtain dense motion estimates. Our focus, however, is not on optical flow estimation, instead, we exploit the tight coupling between edge and flow estimation to train edge detectors without human supervision.

Perceptual Grouping using Motion: Motion plays a key role for grouping and object recognition in the human visual system [21]. In particular, Ostrovsky et al. [27] studied the visual skills of individuals recovering from congenital blindness and showed that motion cues were essential to help facilitate the development of object grouping and representation. Our work is inspired by these findings: we aim to learn an edge detector using motion cues.

Learning from Video: There is an emerging interest for learning visual representations using video as a supervisory signal, for example by enforcing that neighboring frames have a similar representation [26], learning latent representations for successive frames [38], or learning to predict missing or future frames [29, 36]. Instead of simply enforcing various constraints on successive video frames, Wang and Gupta [41] utilized object tracking and enforce that tracked patches in a video should have a similar visual representation. The resulting network generalizes well to surface normal estimation and object detection. As we will demonstrate, our approach can also serve as a novel unsupervised pre-training scheme. However, while in previous approaches the training objective was used as a surrogate to encourage the network to learn a useful representation, our primary goal is to train an edge detector and the learned representation is simply a useful byproduct.

3. Learning Edges from Video

We start with a set of low level cues using standard tools in computer vision, including point correspondences and image gradients. We use DeepMatching [42] to obtain semi-dense matches M between two consecutive frames (I, I'). DeepMatching computes correlations at different locations and scales to generate the matches. Note that contrary to its name, the method involves no deep learning. For the rest of the paper, we fix the matching results M.

Our proposed iterative process is described in Figure 2 and Algorithm 1. We denote the edge detector at iteration t by $\mathcal{E}^t$. For each image I_j, we use E_j^t and G_j^t to denote its image edges and motion edges at iteration t. We initialize E_j^0 to the raw image gradient magnitude of I_j, defined as the maximum gradient magnitude over color channels. The gradient magnitude is a simple approximation of image edges, and thus serves as a reasonable starting point.

At each iteration t, we use EpicFlow [31] to generate edge-preserving flow maps F_j^t given matches M_j and previous edges E_j^{t-1}. We next apply $\mathcal{E}^{t-1}$ on a colored version of F_j^t to get an estimate of motion edges G_j^t. G_j^t is further refined by aligning to superpixel edges. Next, for training our new edge detector $\mathcal{E}^t$, we harvest positives instances using a high threshold on G_j^t and sample random negatives away from any motion edges.

Algorithm 1 Iterative Learning Procedure

Require: Pairs of frames (I_j, I'_j), matches M_j

$\mathcal{E}^0$ = gradient magnitude operator, $E_j^0 = \mathcal{E}^0(I_j)\ \forall j$

for t in $1...T$ **do**

  Estimate flow F_j^t using previous edge maps E_j^{t-1}

  $F_j^t = EpicFlow(I_j, I'_j, M_j, E_j^{t-1})\ \forall j$

  Detect motion edges G_j^t by applying $\mathcal{E}^{t-1}$ to F_j^t

  $G_j^t = \mathcal{E}^{t-1}(FlowToRgb(F_j^t))\ \forall j$

  Train new edge detector $\mathcal{E}^t$ using motion edges G_j^t

  $\mathcal{E}^t = TrainEdgeDetector(\{I_j, G_j^t\})$

  Apply edge detector $\mathcal{E}^t$ to all frames

  $E_j^t = \mathcal{E}^t(I_j)\ \forall j$

end for

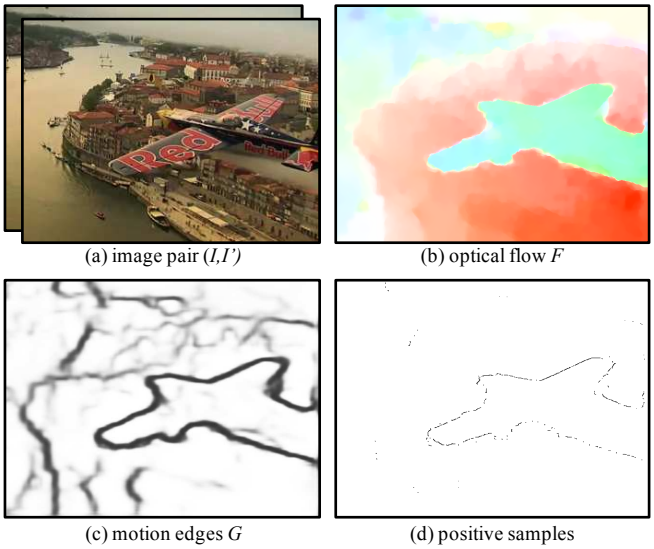
return $\mathcal{E}^T$ and $\{E_j^T, F_j^T, G_j^T\}$

The above process is iterated until convergence (typically 3 to 4 iterations suffice). At each iteration the flow F_j^t and edge maps E_j^t and G_j^t improve. In the following sections we describe the process in additional detail.

3.1. Method Details

EpicFlow: EpicFlow [31] takes as input an image pair (I, I'), semi-dense matches M between the images, and an edge map E for the first frame. It efficiently computes approximate geodesic distance defined by E between all pixels and matched points in M. For every pixel, the geodesic distance is used to find its K nearest matches, and the weighted combination of their motion vectors determines the source pixel's motion. A final optimization is performed by a variational energy minimization to produce an edge-preserving flow map with high accuracy. We refer readers to [31] for additional details.

Motion Edge Detection: Detecting motion edges given optical flow estimates can be challenging, see Figure 3. Weinzaepfel et al. [43] showed that simply computing gradients over a flow map produces unsatisfactory results and instead proposed a data-driven approach for motion edge detection (for a full review of earlier approaches see [43]). In this work we employ a simpler yet surprisingly effective approach. We use an edge detector trained on image edges for motion edge estimation by applying the (image) edge detector to a color-coded flow map. The standard color-coding scheme for optical flow maps 2D flow vectors into a 3D color space by encoding flow orientation via hue and magnitude via saturation [4]. Motion edges become clearly visible in this encoding (3b) (we tried other color spaces but HSV worked best). Running an edge detector $\mathcal{E}$ on the



(a) image pair (*I,I'*) (b) optical flow F

(c) motion edges G (d) positive samples

Figure 3. Illustration of motion edge detection. (a) Input images. (b) Colorized EpicFlow results F on the input images. (c) Motion edges G computed by applying an edge detector $\mathcal{E}$ to the colorized flow. (d) Motion edges G after alignment, non-maximum suppression, and aggressive thresholding. The aligned motion edge map G serves as a supervisory signal for training an edge detector.

colored flow map gives us a simple mechanism for motion edge detection (3c). Moreover, in our iterative scheme, as both our edge detector $\mathcal{E}^{t-1}$ and flow estimate F^t improve with each iteration t, so do our resulting estimates of motion edges $G^t = \mathcal{E}^{t-1}(FlowToRgb(F^t))$.

Motion Edge Alignment: Motion edges computed from flow exhibit slight misalignment with their corresponding image edges. We found that this can adversely affect training, especially for HE which produces thick edges. To align the motion edges we apply a simple heuristic: after applying non-maximum suppression and thresholding, we align the motion edges to superpixels detected in the color image. Specifically, we utilize SLIC super-pixels [1], which cover over 90% of all image edges, and match motion and superpixel edge pixels using bipartite matching (also used in BSDS evaluation) with a tolerance of 3 pixels. Matched motion edge pixels are shifted to the superpixel edge locations and unmatched motion edges are discarded. This refinement, illustrated in Figure 3d, helps to filter out edges with weak image gradients and improves localization.

We emphasize that our goal is not to detect all motion edges. A subset with high precision is sufficient for training. Given a large video corpus, high-precision motion edges should provide a dense coverage of image edges. However, due to our alignment procedure our sampling is slightly biased. In particular, motion edges with weak corresponding image edges are often missing. This limitation and its impact on performance is discussed in Section 4.

Training $\mathcal{E}$: The aligned motion edge maps G^t serve as a supervisory signal for training an edge detector $\mathcal{E}^t$. Positives are sampled at locations with high scoring motion edges in G^t. Negatives are uniformly sampled from location with motion edges below a small threshold. Note that locations with ambiguous motion edge presence (G^t with intermediate scores) are not considered in training. As we will demonstrate, samples harvested in this manner provide a strong supervisory signal for training $\mathcal{E}$.

Video Dataset: For training, we combine videos from two different datasets: the Video Segmentation Benchmark (VSB) [16] and the YouTube Object dataset [28]. We use all HD videos ($100 + 155$) in both datasets. We drop all the annotations for YouTube object dataset. This collection of videos ($\sim$250) contains more than $500K$ frames and has sufficient diversity for training an edge detector.

Frame Filtering: Given the vast amount of available data, we apply a simple heuristic to select the most promising frames for motion estimation. We first fit a homography matrix between consecutive frames using ORB descriptor matches [32] (which are fast to compute). We then remove frames with insufficient matches, very slow motion (max displacement $<$2 pixels), very large motion (average displacement $>$15 pixels), or a global translational motion. These heuristics remove frames where optical flow may be either unreliable or contain few motion edges. For all experiments we used this pruned set of $\sim 50K$ frames.

3.2. Edge Detector Details

We experiment with the Structured Edge (SE) [11] and Holistic Edge (HE) [44] detectors, based on structured forests and deep networks, respectively. SE has been used extensively due to its accuracy and speed, e.g. for flow estimation [31] and object proposals [46, 3]. HE is more recent but achieves the best reported results to date. When trained using our unsupervised scheme, both methods approach similar performance as when trained with full supervision.

Structured Edges (SE): SE extracts low-level image features, such as color and gradient channels, to predict edges. The method learns decision trees by using structured labels (patch edge maps) to determine the split function at each node. During testing, each decision tree maps an input patch to a local edge map. The final image edge map is the average of multiple overlapped masks predicted by each tree at each location, leading to a robust and smooth result. We use the same parameters as in [11] for training. The forest has 8 trees with maximum depth of 64. Each tree is trained using a random subset (25%) of 10^6 patches, with equal number of positives and negatives. During training, we convert a local edge map to a segmentation mask as required by SE by computing connected components in the edge patch. We discard patches that contain edge fragments that do not span the whole patch (which result in a single connected component). During each iteration of training,

the forest is learned from scratch. During testing, we run SE over multiple scales with sharpening for best results.

Holistic Edges (HE): HE uses a modified VGG-16 network [34] with skip-layer connections and deep supervision [22]. Our implementation generally follows [44]. We remove all fully connected layers and the last pooling layer, resulting in an architecture with 13 conv and 4 max pooling layers. Skip-layers are implemented by attaching linear classifiers (1×1 convolutions) to the last conv layer of each stage, their outputs are averaged to generate the final edge map. In our implementation, we remove the deep supervision (multiple loss functions attached to different layers) as we found that a single loss function has little performance penalty (.785 vs .790 in ODS score) but is easier to train.

We experimented with both fine-tuning a network pre-trained on ImageNet [33] and training a network from scratch (random initialization). For fine-tuning, we use the same hyper-parameter as in [44] with learning rate $1e{-}6$, weight decay .0002, momentum .9, and batch size 10. When training from scratch, we add batch normalization [19] layers to the end of every conv block. This accelerates training and also improves convergence. We also increase learning rate $(1e{-}5)$ and weight decay $(.0005)$ when training from scratch. We train the network for 40 epochs in each iteration, then reduce learning rate by half. Unlike for SE, we can reuse the network from previous iterations as the starting point for each subsequent iteration.

The somewhat noisy labels, in particular missing positive labels, prove to be challenging for training HE. The issue is partially alleviated by discarding ambiguous samples during back propagation. Furthermore, unlike in [44], we randomly select negative samples ($40\times$ as many negatives as positives) and discard negatives with highest loss (following the same motivation as in [37]). Without these steps for dealing with noisy labels convergence is poor.

4. Experiments and Results

Our method produces motion edges G^t, image edges E^t, and optical flow F^t at each iteration t. We provide an extensive benchmark for each task tested with two different edge detectors (SE and HE). Our main result is that the image edge detectors, trained using videos only, achieve comparable results as when trained with full supervision. As a byproduct of our approach, we also generate competitive optical flow and motion edge results. Finally, we show that pre-training networks using video improves their performance on object detection over training from scratch.

4.1. Motion Edge Detection

While our focus is not on motion edge detection, identifying motion edges reliably is important as motion edges serve as our only source of supervision. Thus our first experiment is to benchmark motion edges.

Method	ODS	OIS	AP	P20
HUMAN	.63	.63	-	-
SE-IMAGE	.45	.48	.33	.39
HE-IMAGE	.47	.52	.35	.49
EPICFLOW	.39	.47	.33	.55
GALASSO [16]	.34	.43	.23	.34
WEINZAEPFEL [43]	.53	.55	.37	**.71**
SE-VIDEO	.44	.48	.34	**.67**
HE-VIDEO	.45	.47	.32	**.66**

Table 1. Motion edge results on the VSB benchmark. See text.

We use the Video Segmentation Benchmark (VSB) [16] which has annotated ground truth motion edges every 20 frames. We report results on the 282 annotated frames in the test set (we remove frames without motion edges and the final frame of each video as [43] requires 3 frames). We evaluate using three standard metrics [2]: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP). As we are concerned about the high precision regime, we introduce an additional measure: precision at 20% recall (P20). Non-maximum suppression is applied to all motion edges prior to evaluation.

In Table 1 we report results of four baselines and the motion edges G^T obtained from the final iteration of our approach (SE/HE-VIDEO). The baselines include: image edges (SE/HE-IMAGE), gradient magnitude of optical flow (EPICFLOW), a method which combines superpixel segmentation with motion cues (GALASSO [16]), and a recent data-driven supervised approach (WEINZAEPFEL [43]).

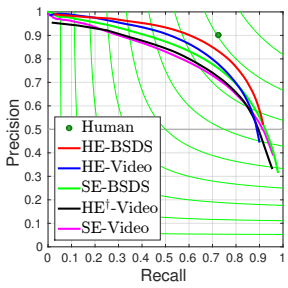
Our method, albeit simple, has a precision .66$\sim$.67 at 20% recall, only slightly worse than [43], even though it was not trained for motion edge detection. It substantially outperforms all other baselines in the high precision regime. While our goal is not motion edge detection per-se, this result is important as it enables us to obtain high quality positive samples for training an image edge detector.

4.2. Image Edge Detection

We next investigate edge detection performance. Results are reported on the Berkeley Segmentation Dataset and Benchmark (BSDS) [25, 2], composed of 200 train, 100 validation, and 200 test images. Each image is annotated with ground truth edges. We again evaluate accuracy using the same three standard metrics: ODS, OIS and AP.

Can an image edge detector be trained using motion edges? Our first experiment tests this question. We use all *ground truth* motion edges available in VSB (591 images) to train both SE and HE. The results are reported in Table 2 (SE-VSB, HE-VSB). For both methods, results are within 2-4 points ODS compared to training with image edge supervision (SE-BSDS, HE-BSDS). Our results suggest that using motion edges to learn an image edge detector is feasible.

We next present results using videos as the supervi-



Method	ODS	OIS	AP
HUMAN	.80	.80	-
SE-BSDS	.746	.767	.803
SE-VSB	.722	.744	.757
SE-VIDEO	.724	.748	.763
HE-BSDS	.785	.803	.791
HE-VSB	.745	.772	.769
HE-VIDEO	.748	.770	.772
HE†-BSDS	.760	.774	.790
HE†-VSB	.719	.735	.751
HE†-VIDEO	.726	.745	.761

Table 2. Edge detection results on BSDS test set. We report results for SE and HE using three training scenarios: BSDS, VSB, and VIDEO (unsupervised). HE uses the VGG network pre-trained on ImageNet, HE† indicates that network is trained from scratch.

sory signal (SE-VIDEO, HE-VIDEO). SE-VIDEO achieves an ODS of .724 compared to .746 for the supervised case (SE-BSDS). Results for HE are similar (.748 versus .785). As these results show, using video supervision achieves competitive results (within 3-5%). Interestingly, learning from video slightly outperforms training using the ground truth motion edges. We attribute this to the small size of VSB.

For HE, we experiment with starting with an ImageNet pre-trained model (HE) and training from scratch (HE†). Pre-training on ImageNet benefits HE across all training scenarios. This is encouraging as it implies that object level knowledge is useful for edge detection. On the other hand, our video supervision scheme also benefits from ImageNet pre-training, thus implying that in our current setup we are not training the model to its full potential.

To probe how performance evolves, we plot the ODS scores at each iteration for both methods in Figure 4. Raw image gradient at iteration 0 has ODS of .543 (not shown). Our iterative process provides a significant improvement from the image gradient, with most of the gains coming in the first iteration. Performance saturates after about 4 iterations (for the last iteration, we use 4 million samples for SE and 80 epochs for HE, which slightly increases accuracy).

We provide visualizations of edge results (before NMS) in Figure 5. SE misses some weak edges but edges are well aligned to the image content. HE generally produces thicker edges due to use of downsampled conv feature maps which makes it difficult to produce sharp image edges. HE-VIDEO/HE†-VIDEO results have thinner edges than HE-BSDS/HE†-BSDS, potentially due to the sampling strategy used for training with motion edges. When training using video, we also observe that the edge detection output is less well localized and more likely to miss weak edges, which likely accounts for much of the performance differences.

4.3. Optical Flow

We benchmark optical flow results on the Middlebury [4] and MPI Sintel [7] datasets. Middlebury is widely used and

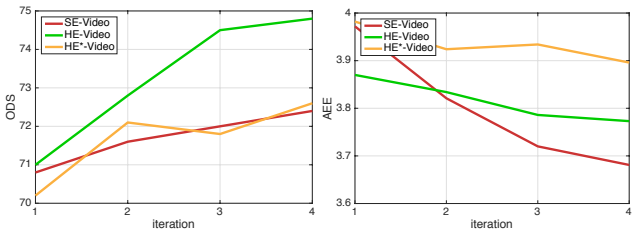


Figure 4. Convergence of ODS and AEE over iterations. See text.

Contour	MPI-Sintel	Middlebury
GT motion edges	3.588	-
SE-BSDS	3.686	.380
SE-VIDEO	3.681	.385
HE-BSDS	3.608	.298
HE-VIDEO	3.773	.308
HE†-VIDEO	3.896	.390

Table 3. Accuracy of EpicFlow with various edge maps (AEE).

consists of complex motions with small displacements. Sintel is obtained from animated sequences and features large displacements and challenging lighting conditions. We use the 'final' version of Sintel and test on the train set with public ground truth. As our goal is to test the quality of generated edge maps, we focus only on variants of EpicFlow [31], the highest performing method on Sintel as of CVPR 2015.

Table 3 shows the average endpoint error (AEE) of EpicFlow using different edge maps for Sintel and Middleburry. Most edge maps give rise to relatively similar results (AEE around 3.6~3.8) on Sintel. In particular, originally EpicFlow used SE-BSDS edges; the results with SE-VIDEO edges are nearly identical. Top results are obtained with HE-BSDS, while HE-VIDEO and HE†-VIDEO are slightly worse. On Middleburry the method rankings are similar.

As an upper bound, we also include EpicFlow given ground truth (GT) motion edges (derived from the ground truth flow). Accuracy is only slightly better than with the best learned edge maps. This suggests that the performance of EpicFlow is saturated given the current matches.

Finally, in Figure 4 we plot AEE on Sintel for each iteration. All methods improve over the initial flow (AEE 4.016, not shown) and results again saturate after a few iterations.

4.4. Object Detection

Finally, we test whether our unsupervised training scheme for edge detection can be used to pre-train a network for object detection. The question of whether strong supervision is necessary for learning a good visual representation for object detection is of much recent interest [41, 9]. While not the focus of our work, we demonstrate that our scheme can likewise be used for network initialization.

For these experiments, we use the HE† edge detector (without ImageNet pre-training). We perform experiments using PASCAL VOC 2007 [12] and the Fast R-CNN [17]

pre-training	mAP
IMAGENET	66.9
NONE	15.6
HE†-BSDS	42.1
HE†-VIDEO	44.2

pre-training	mAP
IMAGENET	58.6
NONE	38.2
HE†-BSDS	41.4
HE†-VIDEO	41.1

Table 4. Object detection results (mean AP) on PASCAL VOC 2007 test using VGG (left) and ZF (right). See text for details.

object detector with proposals from [39]. Results are evaluated by mean Average Precision (mAP). We compare results using two networks, VGG [34] and ZF [45], and four pre-training schemes: ImageNet pre-training; no pre-training; pre-trained on BSDS (HE†-BSDS), and pre-trained using video (HE†-VIDEO). All networks are fine-tuned using the train-val set for $40K$ iterations ($120K$ iterations when training from scratch). Results are summarized in Table 4.

VGG Results: We attempted to train VGG [34] from scratch on VOC (with various learning rates plus batch normalization and dropout) but failed to obtain meaningful results. Even after $120K$ iterations detection performance remains poor ($\sim$15 mAP). When the network is pre-trained on BSDS for edge detection, we were able to achieve 42.1 mAP on PASCAL. Interestingly, when training using video, we observe a further 2 point improvement in mAP (even though the same network is inferior for edge detection).

ZF Results: We also experimented with training a smaller ZF network [45] which has only 5 convolutional layers. We modify the network slightly for edge detection to facilitate the alignment between outputs from different layers[2]. With ImageNet pre-training, Fast R-CNN with our modified ZF network achieves 58.6 mAP on PASCAL, which is close to the ZF result originally reported in [17]. With no pre-training, mAP drops to 38.2. Pre-trained for edge detection, either with or without supervision, improves results by $\sim$3 mAP over training from scratch.

Overall we conclude that pre-training for edge detection (with or without supervision) improves the performance of training an object detector from scratch. However, ImageNet pre-training still achieves substantially better results.

4.5. Limitations

Why doesn't unsupervised training outperform supervised training for edge detection? In theory, a sufficiently large corpus of video should provide an unlimited training set and an edge detector trained on this massive corpus should outperform the much smaller supervised training set. However, there are a number of issues that currently limit performance. (1) Existing flow methods are less accurate at weak image edges, in addition, our alignment scheme also removes weak edges. Thus weak image edges are missing from our training set. (2) Further improving image edges does not improve optical flow, see Table 1. We conjecture that the matches between frames are the limiting factor for EpicFlow and until these are improved neither optical flow nor edges will improve in the current scheme. (3) Training is harmed by noisy labels, and in particular, the missing positive labels. The false negatives, if not handled properly, tend to dominate the gradients in the late stages of training.

Is the unsupervised learning scheme capturing object-level information? The extent of an object is defined by its edges and conversely many edges can only be identified with knowledge of objects. Our results on both edge and object detection support this connection: on one hand, ImageNet pre-training is useful for edge detection, possibly because it injects object-level information into the network. On the other hand, pre-training a network for edge detection also improves object detection. In principle, an edge detection network has to learn high-level shape information, which might explain the effectiveness of pre-training. However, we observe that pre-training on ImageNet still benefits edge detection under all scenarios; moreover, ImageNet pre-training is still substantially better than video pre-training for object detection. Hence, perhaps unsurprisingly, the current unsupervised scheme is not as effective as ImageNet pre-training at capturing object level information.

5. Discussion

In this work, we proposed to harvest motion edges to learn an edge detector from video without explicit supervision. We developed an iterative process that alternated between updating optical flow using edge results, and learning an edge detector based on the flow fields, leading to increasingly accurate edges and flows.

The main result of our paper is that edge detectors, trained using our unsupervised scheme, approach the same level of performance as fully supervised training.

We additionally demonstrated our approach can serve a novel unsupervised pre-training scheme for deep networks. While the gains from pre-training were modest, we believe this is a promising direction for future exploration.

In the long run we believe that unsupervised learning of edge detectors has the potential to outperform supervised training as the unsupervised approach has access to unlimited data. Our work is the first serious step in this direction.

Acknowledgements We thank Saining Xie for help with the HE detector and Ahmad Humayun, Yan Zhu, and Yuandong Tian and many others for valuable discussions and feedback. The work was conducted when Yin Li was an intern at FAIR. Yin Li gratefully acknowledges the support of the Intel ISTC-PC while completing the writing of the paper at Georgia Tech.

[2]We change kernel size of all pooling layers to 2 and modify padding to 3 and 2 in conv1 and conv2, respectively. We also attach classifiers (1×1 convs) on all conv layers and up-sample and merge the results into a single edge map as in [44]. Finally, when training from scratch, we add batch normalization after every conv layer. The ZF network has an ODS of .715 when trained using BSDS and .681 when trained using videos.

Figure 5. Illustration of edge detection results on five sample images (same as used in [11]). The first two rows show the original image and ground truth. The second and third rows are results from SE, trained using BSDS or VIDEO. The remaining rows show the results of variants of HE on BSDS or VIDEO. HE† indicates that the network is trained from scratch. Use viewer zoom functionality to see fine details.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 4

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011. 1, 2, 5

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*,

2014. 1, 4

[4] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 3, 6

[5] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object feat. and its app. to high-level vision. In *ICCV*, 2015. 1, 2

[6] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 2011. 3

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 6

[8] J. Canny. A computational approach to edge detection. *PAMI*, 1986. 1, 2

[9] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2, 6

[10] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 1, 2

[11] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 2015. 1, 2, 4, 8

[12] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2014. 2, 6

[13] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 2008. 1

[14] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE TOC*, 1975. 1, 2

[15] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *PAMI*, 1991. 1, 2

[16] F. Galasso, N. S. Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 4, 5

[17] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 6, 7

[18] B. K. Horn and B. G. Schunck. Determining optical flow. In *1981 Technical symposium east*. International Society for Optics and Photonics, 1981. 2

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[20] J. J. Kivinen, C. K. Williams, and N. Heess. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, 2014. 2

[21] K. Koffka. *Principles of Gestalt psychology*. Routledge, 2013. 3

[22] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 5

[23] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 1, 2

[24] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 2

[25] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 2004. 1, 5

[26] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*, 2009. 3

[27] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 2009. 3

[28] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 4

[29] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. In *ICLR*, 2015. 3

[30] X. Ren and B. Liefeng. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 1, 2

[31] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*, 2015. 1, 2, 3, 4, 6

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *ICCV*, 2011. 4

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 5

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 5, 7

[35] A. Sironi, V. Lepetit, and P. Fua. Multiscale centerline detection by learning a scale-space distance transform. In *CVPR*, 2014. 2

[36] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 3

[37] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv:1412.1441*, 2014. 5

[38] G. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal feat. In *ECCV*, 2010. 3

[39] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recog. *IJCV*, 2013. 1, 7

[40] S. Ullman and R. Basri. Recognition by linear combinations of models. *PAMI*, 1991. 1

[41] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2, 3, 6

[42] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 2, 3

[43] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Learning to Detect Motion Boundaries. In *CVPR*, 2015. 2, 3, 5

[44] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 2, 4, 5, 7

[45] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 7

[46] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 4

[47] C. L. Zitnick and D. Parikh. The role of image understanding in contour detection. In *CVPR*, 2012. 1