Fast Energy Minimization using Learned State Filters

Matthieu Guillaumin
ETH Zürich
guillaumin@vision.ee.ethz.ch

Luc Van Gool
ETH Zürich, KU Leuven
vangool@vision.ee.ethz.ch

Vittorio Ferrari
University of Edinburgh
ferrari@inf.ed.ac.uk

Abstract

Pairwise discrete energies defined over graphs are ubiquitous in computer vision. Many algorithms have been proposed to minimize such energies, often concentrating on sparse graph topologies or specialized classes of pairwise potentials. However, when the graph is fully connected and the pairwise potentials are arbitrary, the complexity of even approximate minimization algorithms such as TRW-S grows quadratically both in the number of nodes and in the number of states a node can take. Moreover, recent applications are using more and more computationally expensive pairwise potentials. These factors make it very hard to employ fully connected models. In this paper we propose a novel, generic algorithm to approximately minimize any discrete pairwise energy function. Our method exploits tractable sub-energies to filter the domain of the function. The parameters of the filter are learnt from instances of the same class of energies with good candidate solutions. Compared to existing methods, it efficiently handles fully connected graphs, with many states per node, and arbitrary pairwise potentials, which might be expensive to compute. We demonstrate experimentally on two applications that our algorithm is much more efficient than other generic minimization algorithms such as TRW-S, while returning essentially identical solutions.

1. Introduction

Models requiring the minimization of a discrete pairwise energy defined over a graph are very common in computer vision. Let $G=(\mathcal{V},\mathcal{E})$ be a graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Each node $n \in \mathcal{V}$ can take a state x_n from a finite set $\mathcal{L}_n$ and has an associated unary energy function $E_n : \mathcal{L}_n \to \mathbb{R}$. An edge $(n,m) \in \mathcal{E}$ connecting nodes n and m has an associated pairwise energy function $E_{n,m} : \mathcal{L}_n \times \mathcal{L}_m \to \mathbb{R}$. Then $\mathcal{L}=\mathcal{L}_1\times\cdots\times\mathcal{L}_{|\mathcal{V}|}$ is the domain of the energy function defined over the graph. With these definitions, the energy of a configuration of states $\mathbf{x} = [x_1, \cdots, x_{|\mathcal{V}|}] \in \mathcal{L}$ is

$$E_{\mathcal{L}}(\mathbf{x}) = \sum_{n\in\mathcal{V}} E_n(x_n) + \sum_{(n,m)\in\mathcal{E}} E_{n,m}(x_n, x_m). \quad (1)$$

The fundamental task of finding the configuration of states $\mathbf{x}^*_{\mathcal{L}} = [x^*_1, \cdots, x^*_{|\mathcal{V}|}] \in \mathcal{L}$ that minimizes (1) is often referred to *inference*, because of the relation to the probabilistic models that typically lead to such energies.

These models have been employed to address many low-level vision problems, where nodes are pixels and pairwise terms act as spatial regularization of the state configuration, e.g. denoising [28], binary segmentation [6], semantic segmentation [25, 16] and stereo matching [28, 30]. Recently, higher-level applications of such energies have emerged, where nodes represent a larger variety of visual elements: body parts [22, 24], superpixels [21, 31], image windows [2, 9]. Here, pairwise terms often express more complex interactions such as kinematic constraints [22], semantic consistency [21], or appearance similarity of whole objects [9, 32]. Matching objects across several images [2, 9, 32] and human pose estimation [22, 24] are examples of such recent developments.

A common trend in these state-of-the-art models is that they are becoming increasingly complex. The number of nodes, the density of the pairwise terms and the size of the state spaces increase. At the same time the pairwise terms are more and more arbitrary and computationally expensive. As an example, [16] uses fully connected graphs over all pixels in a image for semantic segmentation, using linear combination of Gaussians as pairwise potentials. In weakly supervised object localization, [9] uses fully connected graphs where nodes represent images and their state spaces have 100 to 1000 candidate windows for the location of an object. Computing the pairwise terms involves comparing high-dimensional appearance descriptors for all pairs of windows, such as bag-of-SURF, HOG and color histograms. Such pairwise terms are arbitrary in that they have no particular form. Analog models have been recently proposed for co-detection [2] and co-segmentation [32] where discrete energy minimization is used to match candidate windows or segments across images. Densely connected graphs are also emerging for human pose estimation [27], while at the same time the pairwise terms relating the body parts are becoming more expensive, evolving from simple truncated spatial priors [22] to similarity in appearance descriptors [24]. This adds to the computational burden already imposed by the very large state spaces typical for such models, as they correspond to all possible body part positions (10k-100k states [22, 24]).

In these settings, generic energy minimization algorithms become impractical, even approximate ones. For example, the popular TRW-S algorithm [13] poses no restrictions on connectivity, number of states, nor form of the pairwise potential, but its memory and computational complexity grow quadratically in the number of states and linearly in the number of edges (*i.e.* quadratic in the number of nodes in a fully connected model). The problem can be alleviated for certain subclasses of models by exploiting the specific form of their pairwise terms (*e.g.* submodular [6] or truncated convex [17], see sec. 2). Other authors exploit specific graph topologies, such as grids [5] or k-fans [8].

In this paper, we propose a novel generic approximate minimization algorithm (i.e. with no assumptions on pairwise potential, connectivity, nor number of states). Our method *progressively filters* node states in an iterative fashion. At each iteration, the filter takes decisions based on the inference on a tractable subgraph. In addition to substantially reducing the cost of inference, our scheme makes further savings by avoiding computing many pairwise potentials. Filtering node states is made possible by *automatically specializing* to a particular family of problems during a training stage. This consists in learning filters specific to this problem class given as input some example instances along with a low-energy labeling of the nodes (e.g. different input images in semantic segmentation [31, 16, 25], or different input image sets in weakly supervised localization [9, 26, 20]). Importantly, this reference labeling is not a manually provided ground-truth, but the output of a good but slow optimization algorithm. Our method then learns filters to discard as many states as possible without losing any state in the reference labeling.

In this fashion, our method exploits the fact that the distribution of unary and pairwise energies will be similar on new problem instances, and is very efficient because it focuses where computational savings are likely to be obtained. Automatically learning a problem-specific optimizer is the key strength of our method, and it is the reason why it can achieve large speedups over a fixed, non-adaptive algorithm (*e.g.* TRW-S).

Through experiments on two challenging applications (weakly supervised object localization and semantic segmentation), we demonstrate that our minimization algorithm is on average $20\times$ faster than TRW-S [13] while returning essentially identical configurations of states (sec. 4).

2. Related Work

Below, we group discrete pairwise energy minimization algorithms according to their operating conditions.

Low treewidth graphs. A well-known class of tractable problems are energies defined over low treewidth graphs, with potentially many states in the nodes, and an arbitrary form of the pairwise terms. They can be solved exactly by message passing algorithms, e.g. Belief Propagation on trees has complexity $O(H^2 N)$, with H the (average) number of states in a node and N the number of nodes [4]. In general, the Junction Tree Algorithm can be applied for small treewidths, as the complexity is exponential in it.

Specific pairwise potentials. Various optimization algorithms have been proposed for accelerating inference by exploiting a specific form of pairwise potential. Graph-cut algorithms perform fast and exact inference in models with two states and submodular pairwise terms [14], regardless of their connectivity. Move-making extensions of Graph-cut such as α/β-swap, α-expansion [6] support more than two states (often less than 100 [31, 6, 3]). These methods are approximate and assume that pairwise terms are semi-metric and metric, respectively. Other forms of potentials that lends themselves for efficient inference include truncated convex [17], truncated to a small subset of state pairs [22], data independent [19], Gaussian [16], or where distance transforms are applicable [10, 11]. Sapp et al. [34, 24] propose a filtering cascade adapted to state spaces that present a coarse-to-fine decomposition.

Partial labelling. With two states and arbitrary pairwise terms, the roof duality algorithm [12, 23] produces partial solutions, where some nodes might be left unlabeled.

Generic. Some message passing algorithms can approximately minimize generic pairwise energies, e.g. Loopy Belief Propagation [4] and tree-reweighted message passing algorithms [33] such as the popular TRW-S [13]. As the complexity of these methods is quadratic in the number of states and linear in the number of edges, they eventually struggle as the number of nodes increases, the graph becomes denser, and as the state space grows. Komodakis [15] speeds up LP-based algorithms by fixing some nodes to one of their states, based on a hand-tuned dual-gap criterion.

These generic methods require computing *all* pairwise potentials, which for some models might be more expensive than inference itself [9]. In contrast, our filters are learned to be optimal for a particular problem class, and avoid unnecessary pairwise computations.

The meta-algorithm *Fusion moves* [18] optimally combines two good candidate labellings. This strategy heavily depends on having an algorithm that produces good, low-energy candidate labellings. A different algorithm has to be designed for each problem class.

3. Progressive state filtering

In this section we describe our approach for fast minimization of pairwise energy functions. We start by formally

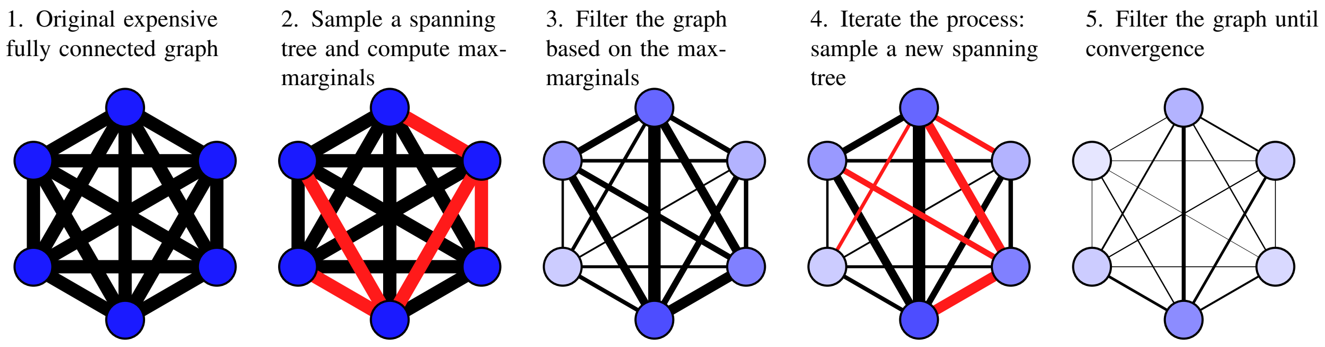
1. Original expensive fully connected graph
2. Sample a spanning tree and compute max-marginals
3. Filter the graph based on the max-marginals
4. Iterate the process: sample a new spanning tree
5. Filter the graph until convergence

Figure 1. *Illustration of our progressive filtering on a fully-connected graph with 6 nodes. The color of a node represents its number of remaining states (darker = more states). The thickness of an edge illustrates the number of remaining pairwise potentials on it. Red edges denote sampled spanning trees. The red edges account for all the pairwise potentials that are computed by our method. They compound to only a small fraction of all pairwise potentials in the original graph (especially in fully connected models with many nodes).*

defining the problem in sec. 3.1, followed by an overview of our filtering framework in sec. 3.2, and then we explain the components of our method in detail: the type of filter functions we learn (sec. 3.3) and how they are learned (sec. 3.5).

3.1. Problem definition

We want to minimize functions of the form

$$E_{\mathcal{L}}(\mathbf{x}) = \sum_{n \in \mathcal{V}} E_n(x_n) + \sum_{(n,m) \in \mathcal{E}} E_{n,m}(x_n, x_m), \qquad (2)$$

and we are particularly interested in the cases where $\mathcal{V}$ is large, $\mathcal{L}_n$ is large for all $n \in \mathcal{V}$, where $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ and where $E_{n,m}(x_n, x_m)$ is computationally expensive. In these cases even just computing all the pairwise potentials can become impractical, although they are typically needed for generic message passing algorithms [4, 13, 33].

3.2. Overview of our method

To minimize Eq. (2), our method sparsifies the original problem so as to contain much fewer pairwise potentials, while retaining the same optimal configuration. More precisely, we iteratively *filter* the state space of a node by discarding states that are very unlikely to belong to the optimal configuration $\mathbf{x}^*_{\mathcal{L}}$. This reduces the number of pairwise terms involving this node. Applying this idea to all nodes makes the number of pairwise terms drastically smaller, which has a positive impact both on the memory and the computational complexity of inference.

Consider the first iteration of filtering. For a node n and its state space $\mathcal{L}_n$, a filter $f^{(1)}$ is a function that partitions $\mathcal{L}_n$ in two subsets: the states $\mathcal{L}_n^{(1)}$ to keep and the states $\mathcal{L}_n \setminus \mathcal{L}_n^{(1)}$ to discard. After filtering, the energy (2) remains the same, but the configuration space becomes $\mathcal{L}^{(1)} = \mathcal{L}_1^{(1)} \times \cdots \times \mathcal{L}_{|\mathcal{V}|}^{(1)}$. If the filter retains the optimal configuration of the original graph, *i.e.* $\mathbf{x}^*_{\mathcal{L}} \in \mathcal{L}^{(1)}$, then the optimal configuration of the filtered graph is the same: $\mathbf{x}^*_{\mathcal{L}^{(1)}} = \mathbf{x}^*_{\mathcal{L}}$. This leads to a central idea in our work: we optimize the filter parameters on a small set of 'training' instances of a problem class (*e.g.* different input images in semantic segmentation) so as to learn to discard as many states as possible, while preserving states likely to belong to the optimum of the original model. For this, we use a reference low-energy configuration obtained from a (slower) generic inference algorithm [13].

Clearly, filtering is only useful if its cost is much smaller than the cost of minimizing (2). For this purpose, we propose filters that are functions of the min-marginals of a node computed on a spanning tree of the original graph (sec. 3.3). The key intuition is that such a min-marginal already contains reliable indications about which states are definitely *not* part of the optimal configuration of the full model.

After applying the filter $f^{(1)}$, in the second iteration we repeat the operation on $\mathcal{L}^{(1)}$ with a second filter $f^{(2)}$, leading to the pruned state spaces $\mathcal{L}_n^{(2)}$. This second filter is based on a *new* spanning tree. In this fashion, as the iterations proceed, the method progressively explores all edges, discovering more and more information. As illustrated in fig. 1 at each iteration the node states spaces become smaller and consequently the edges become 'thinner' (*i.e.* they contain fewer pairwise terms). Therefore, the edges evaluated in an iteration cost much less than those evaluated in a previous iteration. This progressive 'discard & explore' paradigm is the key to the success of our algorithm, as it eventually acquires a full view of the original energy function, *without* paying the price of evaluating all its pairwise potentials.

Our filtering process stops after an iteration where no state was filtered, then we use a standard generic message passing algorithm [13] to perform inference on the heavily thinned model. Typically, at this stage only a few states are left for each node (*e.g.* 3-5 in our experiments on semantic segmentation) making inference extremely rapid and leaving only few pairwise potentials to compute.

In the following sections, we describe our filter functions (sec. 3.3 and sec. 3.4) and how we learn them (sec. 3.5).

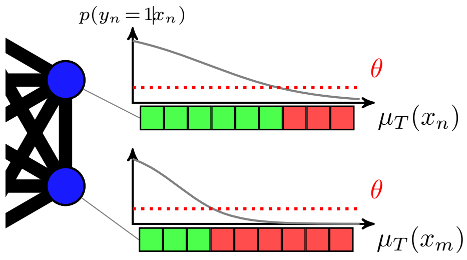

Figure 2. *Illustration of the filtering process for two nodes n and m. From the min-marginals $\mu_T(x_n)$ of the states, we estimate the probability $p(y_n = 1|x_n)$ that the state might belong to the optimum of the full model, and hence should be kept. We learn the threshold θ such that all states with $p(y_n = 1|x_n) \le \theta$ can be discarded (red) while keeping as few states as possible (green).*

3.3. Filter functions

At each iteration k of our algorithm, the filter function $f^{(k)}$ is a binary classifier that decides whether a state of a node should be kept or discarded. The filter acts on the current state space $\mathcal{L}_n^{(k)}$ of node n and it is a function of the min-marginal of the node computed on a random spanning tree T of the original graph. We denote this filter as $f_{T,n}^{(k)}$

$$f_{T,n}^{(k)}(x_n) = \left[\sigma\left(\mathbf{w}^{(k)} \cdot \mathbf{\Phi}_{T,n}^{(k)}(x_n)\right) \ge \theta^{(k)}\right], \qquad (3)$$

where $\sigma(u) = (1 + \exp(-u))^{-1}$ is the logistic function, $\Phi_{T,n}^{(k)}$ is a feature mapping of the min-marginal, $\mathbf{w}^{(k)}, \theta^{(k)}$ are the parameters of the filter, and $[a \ge b] = 1$ if $a \ge b$ and 0 otherwise. We sample a different tree at each iteration k to provides new information about the original graph.

In essence, we use logistic regression to estimate the probability $p_T^{(k)}(y_n = 1|x_n)$ that state x_n might belong to the global optimum of the original graph ($y_n = 1$), and therefore it should be kept

$$p_T^{(k)}(y_n = 1|x_n) = \sigma(\mathbf{w}^{(k)} \cdot \mathbf{\Phi}_{T,n}^{(k)}(x_n)). \qquad (4)$$

The threshold $\theta^{(k)}$ represents the probability value below which states can be safely discarded. Note how the filter parameters $\mathbf{w}^{(k)}, \theta^{(k)}$ are independent of the spanning tree and the state space of node n. Instead, the feature mapping $\mathbf{\Phi}$ depends on them. This enables to learn filters that generalize well over different spanning trees and can adapt to the different sizes of the state spaces of different nodes.

We illustrate the filter functions in fig. 2. The next subsection describes the feature mapping $\mathbf{\Phi}$, and sec. 3.5 explains how to learn the filter parameters.

3.4. Filter features

The key element for generalizing the filters over spanning trees and nodes is the choice of feature map $\Phi_{T,n}^{(k)}(x_n)$. A very good basis for this feature map are the min-marginals computed on the spanning tree T. On a general graph, the min-marginal of state x_n is the minimum energy that can be obtained over the whole configuration space $\mathcal{L}$ when node n has state x_n

$$\mu_n(x_n) = \min_{\hat{\mathbf{x}} \in \mathcal{L}, \hat{x}_n = x_n} E_{\mathcal{L}}(\hat{\mathbf{x}}). \qquad (5)$$

When restricted to a spanning tree T, the energy and min-marginals of a graph are simply summed only over the edges in T

$$E_T(\mathbf{x}) = \sum_{n \in \mathcal{V}} E_n(x_n) + \sum_{(n,m) \in T} E_{n,m}(x_n, x_m), \qquad (6)$$

$$\mu_{T,n}(x_n) = \min_{\hat{\mathbf{x}} \in \mathcal{L}, \hat{x}_n = x_n} E_T(\hat{\mathbf{x}}). \qquad (7)$$

The reasons for choosing spanning trees over a more complex subgraph are twofold. First, a spanning tree is the sparsest substructure that connects all the nodes in a graph. Hence, it minimizes the number of pairwise terms in eq. (6) that the algorithm needs to compute. Second, trees admit very efficient and exact algorithms for min-marginals, based on message passing. The cost of computing all min-marginals of all nodes of T is only about the same as finding the minimum configuration of states on T. This property has been successfully exploited by previous message passing algorithms on loopy graphs [13, 33].

Despite observing only part of the information in the original graph, the min-marginals of a spanning tree already contain powerful cues for identifying states unlikely to be part of the optimal configuration of the full graph. Our feature map $\mathbf{\Phi}_{T,n}^{(k)}(x_n)$ is composed of the following elements

1. The difference between the min-marginal of state x_n and the best state: $\mu_{T,n}(x_n) - \min_{\hat{x}_n} \mu_{T,n}(\hat{x}_n)$. If a state has a min-marginal much larger than the best state, then this might indicate it is not a suitable candidate. This feature is invariant to shifts in the range of min-marginal values.
2. The rank of x_n in the sorted list $\mu_{T,n}(\cdot)$ of all states of node n. If a state has a min-marginal that is among the highest in the node, it might be an indication that it can be safely discarded. This feature is invariant to positive linear transformations of the min-marginals.
3. A differential operator on sorted min-marginals: it takes the difference between the min-marginal for a state and the largest preceding value. This provides the filter with a 'sudden increase' detector which might indicate the end of a run of good candidate states. This is invariant to shifts and robust to small rescalings.
4. A constant value which enables to learn a bias term.

Learning the filter consists in finding the linear combination $\mathbf{w}$ of the min-marginal features and a threshold θ that best classify the states. We describe how we learn the parameters in the following subsection.

3.5. Learning filters

Here we explain how to learn the parameters $\mathbf{w}^{(k)}, \theta^{(k)}$ of a filter $f^{(k)}$ on training instances $\mathcal{G}$ of a family of problems. We run TRW-S [13] on these training graphs to obtain their optimal configuration of states (or a very good approximation if the global optimum is not achievable). This provides a reference low-energy positive ($y_n = 1$) state x_n for each node x (*i.e.* likely to belong to the optimal configuration) and $|\mathcal{L}_n^{(k)}|-1$ negative ones ($y_n = 0$).

A good filter should generalize well over the instances and over the spanning trees $\mathcal{T}_G$ that can be sampled on them. Hence, we learn $\mathbf{w}^{(k)}$ to maximize the likelihood of the correct prediction for all the states of all nodes of all instances, leading to the following objective

$$\ell(\mathbf{w}^{(k)}) = \sum_{G\in\mathcal{G}} \sum_{T\in\mathcal{T}_G} \sum_{n\in G} \sum_{x_n\in\mathcal{L}_n^{(k)}} \ell_T(\mathbf{w}^{(k)}|x_n), \quad (8)$$

$$\begin{aligned} \ell_T(\mathbf{w}^{(k)}|x_n) &= y_n \log p_T^{(k)}(y_n|x_n) \\ &\quad + \lambda(1-y_n) \log p_T^{(k)}(1-y_n|x_n). \end{aligned} \quad (9)$$

However, there are N^{N-2} trees in $\mathcal{T}_G$ for a graph G with N nodes, rendering this objective function intractable. We resort to sampling a limited number N_t of trees for each training instance G.

Note how the training set is imbalanced, as there are many more negative states in the reference labelling than positives. We correct for this by setting λ to a small value such that the total weight of all negative terms is the same as the total weight of positive terms. In addition to compensating class imbalance, λ also expresses the desirable behaviour that the loss for misclassifying negatives (*i.e.* keeping unoptimal states) should be much lower than the loss for misclassifying positives (*i.e.* discarding optimal states). The latter should be absolutely avoided, as discarded states can never be recovered and therefore cannot be in the final configuration output by our algorithm.

The maximum likelihood $\ell(\mathbf{w}^{(k)})$ of the logistic regression is concave. Therefore, any gradient descent algorithm will converge to the globally optimum. In practice, we use limited memory BFGS.

After learning $\mathbf{w}^{(k)}$, we set the threshold $\theta^{(k)}$ to ensure that no optimal state is discarded on newly sampled validation trees

$$\theta^{(k)} = \min_{G\in\mathcal{G}, T\in\mathcal{T}_G, n\in G, x_n\in\mathcal{L}_n^{(k)}, y_n=1} p(y_n = 1|x_n). \quad (10)$$

We first apply the learning procedure above to the original training graphs G to train the first filter $f^{(1)}$. Next, we apply $f^{(1)}$ on the training graphs themselves (on newly sampled trees). Here, it is important to sample a new, unique tree per training graph, as would be the case at test time. We now use the filtered training graphs to train the second filter $f^{(2)}$. We repeat this procedure for all iterations. It is important to train a separate filter per iteration, as the distribution of min-marginal features (sec. 3.4) can change considerably as the state spaces get smaller and the fraction of good states grows. Our procedure properly simulates the test scenario at training time, which leads to good filters. In particular, the randomized validation trees avoid overfitting the thresholds $\theta^{(k)}$ to the training graphs.

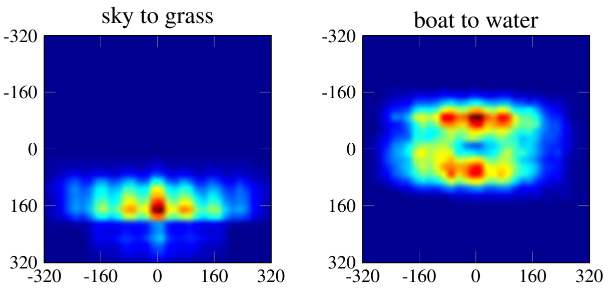

Figure 3. *Distribution of relative positions of superpixels for (sky,grass), left, and (boat,water), right, in the MSRC-21 training set. We show where the second label (grass/water) is most likely to occur relative to the position of the first label (sky/boat, at* $(0, 0)$*).*

4. Experiments

We evaluate our method on two vision problems involving large fully-connected models with pairwise potentials that have arbitrary form and are computationally expensive. The next two subsections introduce these tasks (weakly supervised object localization, sec. 4.1, and semantic segmentation, sec. 4.2). Section 4.3 reports the results.

4.1. Weakly supervised object localization

Given a set of images $\mathcal{I}$ of an unknown object class, weakly supervised object localization [7, 9, 26, 20] is the task of localizing instances of the class. This is typically a preprocessing step before learning an appearance model of the class (which we do not consider in this paper).

Model. We adopt the model of [9], which aims at finding exactly one bounding-box on an instance of the class in each image $I_n \in \mathcal{I}$. Each image is a node n in a fully connected graph G and its states represent candidate windows for the location of the object. There are $S = 100$ candidate windows per image sampled according to their 'objectness' probability of containing an object of any class [1]. The objectness probability is also used as a unary potential E_n. The pairwise potentials $E_{n,m}(x_n, x_m)$ express the dissimilarity between two windows (x_n, x_m) in different images I_n and I_m. The dissimilarity is a linear combination of an appearance distance (Euclidean on HOG descriptors) and the difference in aspect-ratios between the windows. As the appearance descriptor is high-dimensional, the pairwise potentials are expensive to compute (30 minutes on average

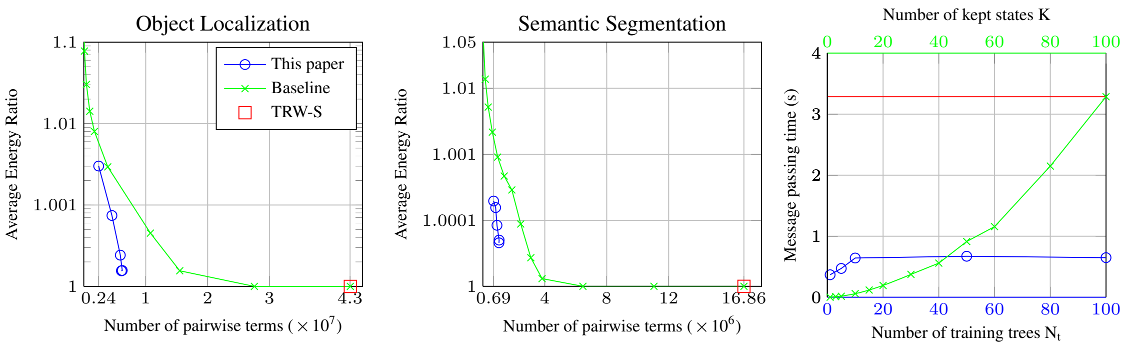

Figure 4. *Quantitative evaluation of our energy minimization algorithm on object localization (left) and semantic segmentation (center). The curves illustrate the trade-off between accurate solutions (low average energy ratio) and computational cost (computed pairwise terms). For our approach, we vary the number of training trees ($N_t = 1, 6, 10, 50, 100$). For the baseline, we vary the number of kept states ($K = 1 \ldots S$). TRW-S uses all S states. We also report the average time to perform message passing for object localization (right). The averages are computed over 18 localization problems and 247 semantic segmentation problems.*

for an image set from our experiments). Minimizing this energy will select windows that are likely to contain objects and form a recurring visual pattern over the input images.

Experimental setup. We follow the setup of [9, 20, 26]. Each input image set is a class-viewpoint combination from the PASCAL VOC 2007 training dataset (the ground-truth bounding-boxes are not given to the algorithm). We use 23 class-viewpoint combinations from 8 classes (car, motorbike, person, train, aeroplane, tvmonitor, bus, boat) with $\sim$3 viewpoints each. An image set contains between 27 and 150 images. We use 5 image sets for training our filters (bus-left, train-frontal, aeroplane-left, car-frontal and person-right) and the remaining 18 image sets for testing.

4.2. Semantic segmentation

Semantic segmentation is a common application of energy minimization algorithms [16, 25, 29, 31]. The task is to assign a label from a set of known classes to every pixel in a test image (e.g. car, road, grass, cow).

Model. A node represents a pixel [25] or a superpixel [29, 31] in the test image. Unary potentials are the likelihoods of the $S = 21$ different classes according to appearance classifiers, which are typically trained beforehand from a set of training images with pixel-level labels [16, 25, 29]. We adopt the model of [31], based on superpixels to offer good spatial support to extract advanced features [29].

Most existing works use simple grid graphs where each node is only connected to its neighbors. However, this can model only short-range interactions such as label smoothness [25, 31] or contextual interactions between adjacent pixels [21]. The interest of using fully connected graphs for semantic segmentation has been recently shown by [16]. Connecting distant (super-)pixels allows to model longer-range interactions. Moreover, our method allows the use of arbitrary, complex pairwise potentials. This enables to fully exploit higher-level contextual information.

We propose to exploit the relative position between any two superpixels in an image to favour or penalize the choice of labels. The intuition is that superpixels of *sky* should appear above superpixels of *street*, and this should be favoured even if superpixels of *car* or *building* appear in-between (which cannot be achieved with grid graphs). We adopt a data-driven model. For each pair (n, m) of superpixels in a test image, we consider the 2D displacement vector $\vec{p}_{n,m}$ between their centers. Then, for all superpixel pairs (i, j) in the training set, we accumulate a score for their ground-truth labels (x_i, x_j) that depends on the Euclidean distance between $\vec{p}_{n,m}$ and $\vec{p}_{i,j}$. The new pairwise potential says that the more similar $\vec{p}_{n,m}$ and $\vec{p}_{i,j}$ are, the more likely (n, m) should have the same labels as (i, j)

$$E_{n,m}(x_n, x_m) = \sum_{(i,j)} g(\vec{p}_{n,m}, \vec{p}_{i,j})[x_n \neq x_i][x_m \neq x_j]. \quad (11)$$

where g is a Gaussian kernel on the Euclidean distance. In fig. 3, we show the distribution of displacement vectors $\vec{p}_{i,j}$ in the training data for two label pairs.

In addition to this new term, our pairwise potential also includes a traditional contrast-sensitive label smoothness term that encourages nearby, similar-looking superpixels to take the same label [16, 25, 31]. Building the fully connected graph with these pairwise terms takes about 75 seconds per image on average.

Experimental setup. We use the standard MSRC-21 dataset [25, 31]. It contains 788 images that we divide in three subsets: (i) 532 images with ground-truth labels used to learn appearance models (unary terms) and displacement-based label penalty (pairwise terms); (ii) 9 images without ground-truth labels to learn our filters for inference; (iii) the remaining 247 images to test our approach.

4.3. Results

We measure the computational benefits of our progressive state filtering algorithm in three ways: (1) The number of all pairwise potentials evaluated by our method. This

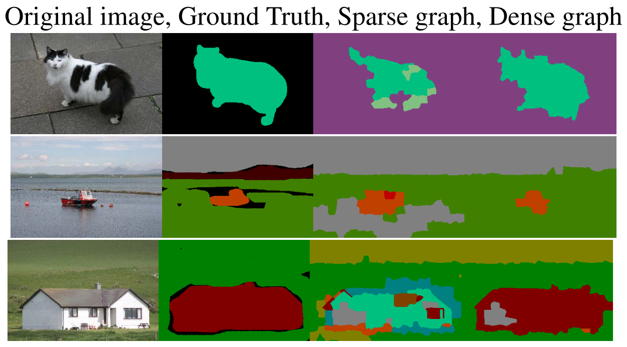

Figure 5. *Qualitative improvements of semantic segmentations thanks to long-range contextual interactions between labels in our fully connected models. We show the original image (far-left) and its ground-truth labels (center-left), and we compare sparse models [31] (center-right) to our fully connected models (far-right).*

is the dominant computation cost, and therefore gives the *speed-up factor* made by our method. The overhead caused by our method (sampling the tree, computing the min-marginals and filtering) is indeed negligible (around 1s); (2) The energy ratio of the best configuration of the graph filtered by our method over the best configuration of the original graph. The latter is delivered by TRW-S [13], which is an excellent[1], but much slower generic minimization algorithm used in various computer vision applications [9, 28]. This measures the *quality of the solution* produced by our algorithm. The best possible value for this ratio is 1. When states belonging to the optimal configuration of the original graph are erroneously discarded by our method, this ratio increases; (3) The computation time of the message-passing passing algorithm [13] that we run after filtering, compared to the time that would be spent on the original graph.

To obtain an insight on the efficiency-vs-quality trade-off of our approach, we have learnt state filters with various numbers of random training spanning trees ($N_t = 1..100$). Because the threshold θ is set to discard no optimal state over all training trees (eq. (10)), we expect a greater N_t to lead to higher quality solutions in return for filtering fewer states. As a baseline, we consider keeping the K labels for each node that have the best unaries, and vary the value of K from 1 to S, the number of states in the original problem. Figure 4 shows the results. Each point in a plot is the average over all test graphs (18 test image sets for object localization, and 247 test images for semantic segmentation). We observe: (1) For all values of N_t, our algorithm successfully computes only a small fraction of the pairwise potentials, thanks to discarding many states of many nodes.

[1]In addition to returning a solution and its energy, TRW-S also gives a lower bound on the energy. The global optimum is in-between. The ratio between the energy of the returned solution and its corresponding lower bound, averaged over all problem instances is 1.0008 for semantic segmentation, i.e. the solution by TRW-S is within 0.08% of the global optimum. For object localization it is within 0.06%. Hence, TRW-S is delivering solutions very close to the global optimum on these problems.

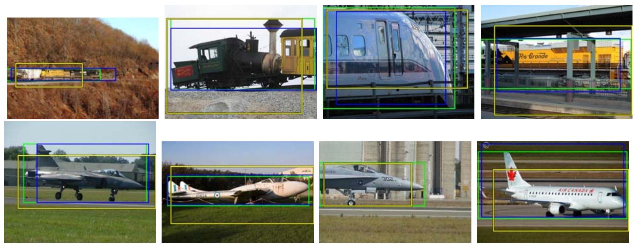

Figure 6. *Qualitative results for localizing trains and aeroplanes. The green windows show the ground-truth boxes, the blue windows show the best window among the $S = 100$ candidates [1]. The yellow windows are those selected by TRW-S and by our algorithm (which are identical).*

This leads to computing up to 20× fewer pairwise potentials ($N_t = 1$). Thanks to this, it now takes only a fraction of the original time to compute the potentials for object localization (on average about 100s vs 1800s) and semantic segmentation (about 3s vs 75s). (2) Discarding states also has a positive impact on the time required to perform message passing, with 3× to 4× speed-up on this component of the energy minimization procedure. However, on these problems, this is negligible compared to the cost of the pairwise potentials. (3) The energy of the best configuration of the filtered graphs are essentially identical to those of the original graphs. Even for $N_t = 1$, on average the energy is less than 0.5% higher. This means that the speedups brought by our method comes at no loss of accuracy of the output solution, compared to TRW-S [13]. (4) For object localization, the baseline with low K leads to solutions with substantially larger energy than what delivered by TRW-S (for $K=3$, +15% on average, and up to +25%). For semantic segmentation, it is reasonably good (+0.4% on average and up to +7%). On both problems, our approach obtains lower energy solutions for any given number of pairwise terms. Moreover, for the baseline, the user would have to manually set the value for K, which is heavily dependent on the problem class.

Table 1 reports the pixel-level accuracies obtained for semantic segmentation with our fully connected graphs on the official test set of MSRC-21, compared to the sparse graphs of [31]. Although both models use the exact same unaries, we obtain an improvement on 16 classes out of 21 as well as on average over all classes. This demonstrates the benefits of using fully connected models with our long-range spatial context terms (fig. 5). Finally, the table also reports the results for our fully connected model, but using the slower TRW-S for inference instead of our method. The results are essentially identical, which shows that our method not only delivers solutions with near identical energy, but also near identical configurations of states. We observed the same behavior on the object localization task, where only 0.3% of the candidate windows selected by our approach differ from those selected by TRW-S (only 5 different windows over the

Class accuracy	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	mean
Sparse graph [31]	23.5	90.4	80.2	84.3	**96.4**	85.3	97.9	67.0	**86.4**	93.2	98.1	**79.3**	71.2	53.6	62.6	**59.0**	67.8	60.7	50.3	42.5	**48.0**	71.3
This paper (TRWS)	**31.9**	**91.5**	84.5	**89.2**	91.8	**89.7**	**99.0**	**68.0**	79.5	**94.1**	**98.3**	72.6	76.2	**55.3**	**64.9**	53.4	**74.2**	**62.7**	**53.4**	**54.7**	46.3	**72.9**
This paper (Filters)	**31.9**	**91.5**	**84.7**	**89.2**	91.8	**89.7**	**99.0**	**68.0**	79.5	**94.1**	**98.3**	72.6	**76.3**	**55.3**	**64.9**	53.4	**74.2**	**62.7**	**53.4**	**54.7**	46.3	**72.9**

Table 1. *Class accuracies on the MSRC-21 test set. We compare the performance of the sparse model of [31] and our fully-connected model with long-range spatial context interaction, using either TRW-S or our progressive filtering for inference. Using the same unaries, our model improves on 16 out of the 21 classes. Our fast inference algorithm obtains essentially the same results as the slower TRW-S.*

entire experiment involving 1413 images). We show qualitative results for object localization in fig. 6.

5. Conclusion

We presented a novel minimization algorithm for discrete pairwise energies that can handle densely connected graphs, large state spaces, and arbitrary pairwise potentials. We demonstrated experimentally its high computational efficiency over baseline filters and the TRW-S algorithm.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.

[2] S. Y. Bao, Y. Xiang, and S. Savarese. Object co-detection. In *ECCV*, 2012.

[3] D. Batra and P. Kohli. Making the right moves: Guiding alpha-expansion using local primal-dual gaps. In *CVPR*, 2011.

[4] C. Bishop. Pattern recognition and machine learning. *Springer*, 2006.

[5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on PAMI*, 23(11):1222–1239, 2001.

[7] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR*, 2007.

[8] D. J. Crandall and D. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*, 2006.

[9] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9), 2010.

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[12] A. Fix, J. Chen, E. Boros, and R. Zabih. Approximate mrf inference using bounded treewidth subgraphs. In *ECCV*, 2012.

[13] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. on PAMI*, 28(10):1568 – 1583, 2006.

[14] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Trans. on PAMI*, 26(2):147–159, 2004.

[15] N. Komodakis. Towards more efficient and effective lp-based algorithms for mrf optimization. In *ECCV*, 2010.

[16] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[17] M. P. Kumar, O. Veksler, and P. H. Torr. Improved moves for truncated convex models. *JMLR*, 2011.

[18] V. S. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. *IEEE Trans. on PAMI*, 32(8):1392–1405, 2010.

[19] J. McAuley and T. Caetano. Exploiting data-independence for fast belief propagation. In *ICML*, 2010.

[20] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.

[21] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007.

[22] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2006.

[23] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *CVPR*, 2007.

[24] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010.

[25] J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *CVPR*, 2008.

[26] P. Siva and T. Xiang. Weakly supervised object detector learning with model drift detection. In *ICCV*, 2011.

[27] M. Sun, M. Telaprolu, H. Lee, and S. Savarese. An efficient branch-and-bound algorithm for optimal human pose estimation. In *CVPR*, 2012.

[28] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. on PAMI*, 30(6):1068–1080, 2008.

[29] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *ECCV*, 2010.

[30] R. Timofte and L. Van Gool. Four colour theorem for fast early vision. In *ACCV*, 2010.

[31] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised semantic segmentation with multi image model. In *ICCV*, 2011.

[32] S. Vicente, C. Rother, and V. Kolmogorov. Object cosegmentation. In *CVPR*, pages 2217–2224, 2011.

[33] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

[34] D. Weiss, B. Sapp, and B. Taskar. Sidestepping intractable inference with structured ensemble cascades. In *NIPS*, 2010.