Joint Recovery of Dense Correspondence and Cosegmentation in Two Images

Tatsunori Taniai  
The University of Tokyo

Sudipta N. Sinha  
Microsoft Research

Yoichi Sato  
The University of Tokyo

Abstract

We propose a new technique to jointly recover cosegmentation and dense per-pixel correspondence in two images. Our method parameterizes the correspondence field using piecewise similarity transformations and recovers a mapping between the estimated common “foreground” regions in the two images allowing them to be precisely aligned. Our formulation is based on a hierarchical Markov random field model with segmentation and transformation labels. The hierarchical structure uses nested image regions to constrain inference across multiple scales. Unlike prior hierarchical methods which assume that the structure is given, our proposed iterative technique dynamically recovers the structure along with the labeling. This joint inference is performed in an energy minimization framework using iterated graph cuts. We evaluate our method on a new dataset of 400 image pairs with manually obtained ground truth, where it outperforms state-of-the-art methods designed specifically for either cosegmentation or correspondence estimation.

1. Introduction

Recovering dense per-pixel correspondence between image regions in two or more images is a central problem in computer vision. While correspondence estimation for images of the same scene (stereo, optic flow, etc.) is well studied, there has been growing interest in the case where the images portray semantically similar but different scenes or depict semantically related but different object instances [36]. Due to the variability in appearance, shape and pose of distinct object instances, camera viewpoint, scene lighting and backgrounds in the images, the task is quite challenging in the unsupervised setting. Yet, correspondence estimation enables fine-grained image alignment crucial in tasks such as non-parametric scene parsing and label transfer [36], 3D shape recovery [51], image editing [19] and unsupervised visual object discovery [11, 42, 45, 50].

In parallel to advances in correspondence estimation, there has also been rapid progress in image cosegmentation [17, 26, 41, 46] methods that automatically segment similar “foreground” areas in two or more images. These methods often require the foregrounds depicting common objects to have similar region statistics. Most cosegmentation methods do not explicitly recover dense pixel correspondence and alignment in the region labeled foreground. On the other hand, correspondence estimation methods [36, 29, 53, 23] align all the pixels without explicitly inferring which pixels in the two images actually have valid correspondence. Thus, recovering cosegmentation along with a dense alignment of the common foregrounds can be viewed as a holistic approach to solving both tasks.

In this paper, we present insight into how image cosegmentation and correspondence (or flow) estimation can be tackled within a unified framework by framing it as a labeling problem (Figure 1). We show that jointly solving the two tasks in this way can improve performance on both of them. This paper deals with the case where only two input images are given. The setting is unsupervised and we do not assume a priori information about the objects or the scene.

Our contributions are three folds. First, we propose a new hierarchical Markov random field (MRF) model for joint cosegmentation and correspondence recovery. The hierarchy is defined over nested image regions in the reference image and the nodes representing these regions take segmentation and flow labels. Precomputed hierarchical structures [29, 23, 27] are unsuitable for our task because pixels inferred as background must be excluded from matching.

Second, we propose a new optimization technique for the joint inference of the graph structure and labeling. Per-
forming exact inference jointly on the whole hierarchical structure is intractable. In the proposed approach, layers of the hierarchy are incrementally estimated with the labeling in an energy minimization framework using iterated graph cuts [8, 31] (alpha expansion moves).

Finally, we release a new dataset with 400 image pairs for which we provide ground truth cosegmentation masks and flow maps. The original images and some of the segmentation masks are taken from existing datasets [35, 42, 20]. The remaining segmentation masks were obtained from interactive image segmentation. The flow maps were obtained by selecting sparse keypoint correspondence with our interactive annotation tool and applying natural neighborhood interpolation [44] on the sparse data. Poor flow maps were discarded by visually inspecting the flow-induced image warping result. The ground truth flow maps makes it possible to directly evaluate dense image alignment. Even SIFT Flow [36] and other correspondence estimation methods [29, 54] are evaluated indirectly on tasks such as segmentation transfer and scene parsing using datasets that lack ground truth pixel correspondence.

The paper is organized as follows. We describe related work in Section 2 and our proposed model in Section 3. Section 4 presents our optimization method whereas implementation details and the images features used are described in Section 5. Finally, in Section 6, we report experimental evaluation and comparisons with existing approaches.

2. Related Work

We are not aware of any existing method that explicitly solves both cosegmentation and dense correspondence recovery together. However, the motivation behind our work is similar to that behind some recent cosegmentation methods [13, 17, 43]. We review those and other broadly related works on cosegmentation and correspondence estimation.

Cosegmentation. Rubio et al. [43] formulate cosegmentation in terms of region matching. However, the matches are computed independently using graph matching [16] and then exploited in their cosegmentation algorithm. Faktor and Irani [17] describe a model where common foregrounds in multiple images can be composed from interchangeable image regions. Although region matching is a key element of their method, it is primarily used to estimate unary potentials (foreground/background likelihoods) for a standard image segmentation method. While, Dai et al. [13] propose to cosegment images by matching foregrounds through a codebook of deformable shape templates, it involves learning a codebook requiring external background images. While a notion of correspondence implicitly exists in all these works, none of them explicitly compute dense correspondence maps between the cosegmented regions, which is an important distinction to our work.

Cosegmentation methods originally proposed by Rother et al. [41] have been applied in broader settings [5, 24, 25, 32, 28, 52] and also on large sets of Internet images [42, 11]. Interesting convex formulations have also been proposed for a variant of cosegmentation—the object co-localization task [26, 46], which aims to find a bounding box around related objects in multiple images.

Correspondence Estimation. SIFT Flow [36] generalizes optic flow to images of different scenes and estimates complete flow maps with 2D translations at every pixel. Their energy function uses local matching costs based on dense SIFT features, and smoothness terms encoding standard pairwise potentials. SIFT Flow uses loopy belief propagation (BP) [18] for inference in a coarse-to-fine pipeline but other inference techniques [19, 53] have also been explored. HaCohen et al. [19] propose an extension of PatchMatch [3, 4] that handles images of identical scenes with large motions. However, their method is often unable to handle different scenes as it lacks regularization on correspondence fields. As another extension, DAISY filter flow (DFF) [53] proposes to use efficient cost-volume filtering [38] for enforcing smoothness, instead of adding explicit regularization. Deformable spatial pyramid (DSP) matching [29] and its generalization [23] propose hierarchical regularization using a regular grid-cell pyramid for flow estimation. Correspondence maps are parameterized using similarity transformations in [23] similarly to our work. Images with scale differences are handled by [39, 21, 23]. Cosegmentation has been used to guide sparse feature correspondence recovery [10]. However, such methods do not aim to accurately segment common regions.

Hierarchical Models. To exploit multi-scale image cues or to add flexible regularization, hierarchical conditional random fields (CRFs) have been proposed for single image segmentation [22], image matching [49], stereo correspondence [33], and much recently for optic flow [34, 27] and more general correspondence estimation [29, 23]. These methods use precomputed hierarchical structures such obtained by an external hierarchical oversegmentation method [2], or spatial pyramids as used in DSP [29, 23].

Optimization Techniques. Discrete optimization is commonplace in stereo but often problematic in general dense correspondence estimation because of the large label spaces involved. For this issue, SIFT Flow [36] performs hierarchical BP [18] on the image pyramid from coarse to fine levels using limited translation ranges. Recently, inspired by randomization search and label propagation schemes of PatchMatch [3, 4, 7], optimization methods using BP [6] or graph cuts [48, 47] have been proposed for efficient inference in pairwise MRFs with large label spaces. However, they are not directly applicable to our hierarchical model. We extend graph cut techniques [48, 47] for our inference task where we recover both the graph structure as well as the labeling.
3. Proposed Model

Given two images I and I’ our goal is to find dense correspondence and cosegmentation of a common object shown in the two images. The reference image I is represented by a set of superpixel nodes \( i \in V \) where \( \Omega_i \subseteq \Omega \) denotes a superpixel region in the image domain \( \Omega \subseteq \mathbb{Z}^2 \).

In the reference image, we seek a labeling involving a geometric transformation \( T_i \in \mathcal{T} \) and a foreground alpha-matte value \( \alpha_i \in [0, 1] \) for each superpixel node \( i \in V \). We formulate this as a mapping function \( f_i : V \rightarrow \{ T \times [0, 1] \} \) that assigns each node a pair of labels \( f_i = (T_i, \alpha_i) \). Here, \( \alpha_i \) is continuous during inference and binarized at the final step\(^1\). \( T_i \) denotes a similarity transform parameterized using a quadruplet \( (t_u, t_v, s, r) \). Slightly abusing the notation, we express the warped pixel location of \( p' \) in the other image as follows.

\[
p' = T_i(p) = sR_r(p - c_i) + c_i + t.
\]

Here, \( c_i \) is the centroid of pixels in region \( \Omega_i \), and centering at this point, \( p \) is rotated by the 2D rotation matrix \( R_r \) of angle \( r \) and scaled by \( s \), and then translated by \( t = (t_u, t_v) \).

In following sections we present the proposed model, by first defining a standard 2D MRF model in Section 3.1 and later generalizing it to a hierarchical model in Section 3.2. We discuss the allowed hierarchical structure in Section 3.3.

3.1. Single Layer Model

Let \( L = (V, E) \) be a graphical representation of the image I, where nodes \( i \in V \) and edges \( (i, j) \in E \) represent superpixels and spatial neighbors, respectively. Given this graph, our single layer model is defined as a standard 2D MRF model:

\[
E_{\text{mur}}(f|L) = \lambda_{\text{flo}} \sum_{i \in V} E_{\text{flo}}(f_i) + \lambda_{\text{reg}} \sum_{i \in V} E_{\text{reg}}(f_i) + \sum_{(s,t) \in E} w_{st} E_{\text{reg}}(f_s, f_t),
\]

(2)

which consists of the flow data term, cosegmentation data term and the spatial regularization term described below. **Flow Data Term.** \( E_{\text{flo}}(f_i) \) measures similarity between corresponding regions in the image pair. We define it as

\[
E_{\text{flo}}(f_i) = \sum_{p \in \Omega_i} \left[ \alpha_i \rho(p, p') + \bar{\alpha}_i \lambda_{\text{occ}} \right],
\]

(3)

where \( \bar{\alpha}_i = 1 - \alpha_i \) and \( \lambda_{\text{occ}} \) is a constant penalty for background pixels to avoid trivial solutions where all pixels are labeled background. The \( \rho(p, p') \) robustly measures visual dissimilarity between \( p \) and its correspondence \( p' \) as

\[
\rho(p, p') = \min \{ \| D(p) - D'(p') \|_2^2, \tau_D \},
\]

(4)

\(^1\)To avoid degenerate flow solutions, we set \( \alpha_i \) always larger than 0.1 during inference. See supplementary for a detailed explanation.

**Cosegmentation Data Term.** The foreground and background likelihoods for each node are defined as follows.

\[
E_{\text{seg}}(f_i) = -\sum_{p \in \Omega_i} \left[ \alpha_i \ln P(I_p|\theta^F) + \bar{\alpha}_i \ln P(I_p|\theta^B) \right].
\]

(5)

Here, \( P(\cdot|\theta) \) is likelihood given a foreground or background color model \( \{ \theta^F, \theta^B \} \) of the image I, which is implemented as \( 64^3 \) bins of RGB color histograms. The color models are estimated during initialization (Section 5.3).

**Spatial Regularization Term.** The term \( E_{\text{reg}} \) encourages flow and alpha values of neighboring nodes to be similar.

\[
E_{\text{reg}}(f_s, f_t) = \lambda_{\text{st}} \min \{ \alpha_s, \alpha_t \} \sum_{p \in B_{st}} \psi_{\text{st}}(p)/|B_{st}| + \lambda_{\text{st}} |\alpha_s - \alpha_t|.
\]

(6)

Here, \( B_{st} = \partial \Omega_s \cap \partial \Omega_t \) is the set of pixels on the boundary of two adjoining regions \( \Omega_s \) and \( \Omega_t \), and \( \psi_{\text{st}}(p) \) penalizes flow discontinuities at these pixels. It is defined as

\[
\psi_{\text{st}}(p) = \min \{ \| T_s(p) - T_t(p) \|_2, \tau_{st} \}.
\]

(7)

If \( \alpha \) were binary, then the first term in Eq. (6) would enforce flow smoothness when two adjoining regions are labeled foreground (\( \alpha_s = \alpha_t = 1 \)) and the second term would give a constant penalty \( \lambda_{\text{st}} \) only when \( \alpha_s \neq \alpha_t \). However, Eq. (6) generalizes this idea to continuous valued \( \alpha \).

3.2. Hierarchical Model

Now we introduce the notion of a layered graph and generalize the single layer model to a full hierarchical model. As illustrated in Figure 2, our hierarchical graph \( G = (V, E) \) consists of multiple layered subgraphs \( \{ L_0, L_1, \cdots, L_H \} \). Each layer \( L_i = (V_i, E_i) \) represents a

\(\text{Flow Data Term.}\)

\[
\rho(p, p') = \min \{ \| D(p) - D'(p') \|_2^2, \tau_D \},
\]

(4)

\(\text{Cosegmentation Data Term.}\)

\[
E_{\text{seg}}(f_i) = -\sum_{p \in \Omega_i} \left[ \alpha_i \ln P(I_p|\theta^F) + \bar{\alpha}_i \ln P(I_p|\theta^B) \right].
\]

(5)

Here, \( P(\cdot|\theta) \) is likelihood given a foreground or background color model \( \{ \theta^F, \theta^B \} \) of the image I, which is implemented as \( 64^3 \) bins of RGB color histograms. The color models are estimated during initialization (Section 5.3).

**Spatial Regularization Term.** The term \( E_{\text{reg}} \) encourages flow and alpha values of neighboring nodes to be similar.

\[
E_{\text{reg}}(f_s, f_t) = \lambda_{\text{st}} \min \{ \alpha_s, \alpha_t \} \sum_{p \in B_{st}} \psi_{\text{st}}(p)/|B_{st}| + \lambda_{\text{st}} |\alpha_s - \alpha_t|.
\]

(6)

Here, \( B_{st} = \partial \Omega_s \cap \partial \Omega_t \) is the set of pixels on the boundary of two adjoining regions \( \Omega_s \) and \( \Omega_t \), and \( \psi_{\text{st}}(p) \) penalizes flow discontinuities at these pixels. It is defined as

\[
\psi_{\text{st}}(p) = \min \{ \| T_s(p) - T_t(p) \|_2, \tau_{st} \}.
\]

(7)

If \( \alpha \) were binary, then the first term in Eq. (6) would enforce flow smoothness when two adjoining regions are labeled foreground (\( \alpha_s = \alpha_t = 1 \)) and the second term would give a constant penalty \( \lambda_{\text{st}} \) only when \( \alpha_s \neq \alpha_t \). However, Eq. (6) generalizes this idea to continuous valued \( \alpha \).
superpixel graph of the image. In addition to spatial edges within each layer \( E_l \), our hierarchy \( G \) contains parent-child edges \((p,c) \in E_{pc}^s\) that connect parent nodes \( p \in V_l \) to their children nodes \( c \in V_{l-1} \) (green edges in Figure 2).

Using a layered graph \( G \) and the model \( \mathcal{E}_{\text{mrf}}(f|L) \) defined in Eq. (2), we define our hierarchical model as

\[
\mathcal{E}(f, G) = \sum_{l=0}^{H} \left[ \mathcal{E}_{\text{mrf}}(f|L_l) + \mathcal{E}_{\text{reg}}^l(f|G) + \mathcal{E}_{\text{gra}}^l(V_l) \right].
\] (8)

Here, we treat the hierarchical graph \( G \) as a variable that is dynamically estimated together with \( f \). Our construction is fundamentally different from prior work [23, 27, 29], where the hierarchical structure is computed before flow inference.

**Multi-layer Regularization Term.** Similar to the spatial regularization term in Eq. (6), the term \( \mathcal{E}_{\text{reg}}^l \) enforces smoothness between parent child pairs of \( V_l \) and \( V_{l-1} \) as

\[
\mathcal{E}_{\text{reg}}^l(f|G) = \sum_{(p,c) \in E_{pc}^s} w_{pc} \mathcal{E}_{\text{reg}}^{pc}(f_p, f_c),
\] (9)

where \( \mathcal{E}_{\text{reg}}^{pc} \) is defined using Eq. (7) and \( c \)'s centroid \( c_c \) as

\[
\mathcal{E}_{\text{reg}}^{pc}(f_p, f_c) = \lambda_{pc1} \min \{ \alpha_p, \alpha_c \} \psi_{pc}(c_c) + \lambda_{pc2} |\alpha_p - \alpha_c|.
\] (10)

**Graph Validity Term.** The term \( \mathcal{E}_{\text{gra}}^l \) measures validity of the layer structure \( V_l \) as

\[
\mathcal{E}_{\text{gra}}^l(V_l) = \lambda_{\text{mod}} \beta^l |V_l| - \lambda_{\text{col}} \sum_{i \in V_l} \sum_{p \in \Omega_i} \ln P(I_p|\theta^l).
\] (11)

The first term reduces nodes in the higher layers. We set \( \beta = 2 \) to reduce the node count approximately by half at each layer. The second term enforces color consistencies within each region \( \Omega_i \). \( \theta^l \) represents the RGB color histogram of the region \( \Omega_i \). Our definition of Eq. (11) is motivated by work in multi-region segmentation [15].

3.3. Hierarchical Structure

Here we describe the form of hierarchical graphs allowed in our method. The nodes \( i \in V_l \) in each layer divide the image domain \( \Omega \) into \( |V_l| \) connected regions \( \Omega_i \subseteq \Omega \). Our hierarchical superpixels have a nested (or tree) structure, i.e., a superpixel (parent) in a layer \( V_l \) consists of the union of superpixels (children) in its sublayer \( V_{l-1} \). The lowest layer \( V_0 \) named the pixel layer is special because each node \( i \in V_0 \) represents a pixel \( p_i \in \Omega \). The finest layer \( V_H \) has about 500 nodes which are set to SLIC superpixels [11].

For parent-child edges \((p,c) \in E_{pc}^s (l = 1, \cdots, H)\), the edge weights \( w_{pc} \) are assigned to the area of child regions

\[
w_{pc} = |\Omega_c|. \] (12)

At the two lowest layers \((l = 0, 1)\), edges \((s,t) \in E_t \) between adjoining nodes are assigned edge weights \( w_{st} \) as

\[
w_{st} = e^{-\|I_s-I_t\|^2/\kappa}, \] (13)

where \( I_i \in \mathbb{R}^3 \) is the mean color of the region \( \Omega_i \). Following [40], we set \( \kappa \) to the expected value of \( 2|I_s-I_t|^2 \) over \((s,t) \in E_t \). For the upper layers \((l \geq 2)\), the edge weights \( w_{st} \) are set to the sum of the children’s edge weights as

\[
w_{st} = \sum w_{s't'}, \] (14)

where \((s',t') \in E_{l-1} \) are children of \( s \) and \( t \), respectively.

4. Optimization

Optimizing \( \mathcal{E}(f, G) \) in Eq. (8) has two main difficulties. 1) The joint inference of \( f \) and \( G \) is intractable due to the dependency of \( f \) on \( G \). 2) The label space of \( f \) resides in a 5-dimensional continuous domain and the number of candidate labels is essentially infinite. To practically address these issues, we propose two-pass bottom-up and top-down optimizing procedures that approximately optimize the energy. In the bottom-up phase, we construct a hierarchical structure \( G \) by incrementally adding layers from lower to higher levels, while simultaneously estimating the labeling \( f \). In the top-down phase, we refine the labeling \( f \) while keeping the structure \( G \) fixed. The optimization procedure is summarized in Algorithm 1. Next we discuss the details.

4.1. Bottom-Up Hierarchy Construction

To formally describe our bottom-up procedure, we denote \( G^k = (V^k, E^k) \) as a hierarchy consisting of \( k+1 \) layered subgraphs \( \{L_0, \cdots, L_k\} \) where \( L_k = (V_k, E_k) \). We also define it sequentially, i.e., \( G^k \) and \( G^{k+1} \) share the same structure for the bottom \( k+1 \) layers.

At a high level, our bottom-up procedure is presented as a sequence of subtasks, where given a current solution \( \{f, G^k\}\) we estimate \( \{f, G^{k+1}\} \) as illustrated in Figures 3 (a) and (d), respectively. We estimate \( \{f, G^{k+1}\} \) as approximate minimizers of \( \mathcal{E}(f, G^{k+1}) \) in Eq. (8). Here, \( \mathcal{E}(f, G^{k+1}) \) given \( G^k \) can be separated into two parts

\[
\mathcal{E}(f, G^{k+1}) = \mathcal{E}(f|G^k) + \mathcal{E}_{\text{top}}(f, L_{k+1}),
\] (15)

where \( \mathcal{E}(f|G^k) \) is energy involved in the known graph \( G^k \) while \( \mathcal{E}_{\text{top}}(f, L_{k+1}) \) refers to the unknown top layer \( L_{k+1} \).

\[
\mathcal{E}_{\text{top}}(f, L_{k+1}) = \mathcal{E}_{\text{mrf}}(f|L_{k+1}) + \mathcal{E}_{\text{reg}}^{k+1}(f|G^{k+1}) + \mathcal{E}_{\text{gra}}^{k+1}(V_{k+1}).
\] (16)

Jointly inferring \( G^{k+1} \) and its labeling \( f \) is difficult. Therefore, we assume a known temporary graph \( G^{k+1} \) for unknown \( G^{k+1} \), and we replace this joint problem by a simpler labeling problem \( \hat{f} \) on \( G^{k+1} \).

\[
\mathcal{E}(\hat{f}|G^{k+1}) = \mathcal{E}((\hat{f})|G^{k+1}) + A(\hat{f}).
\] (17)

Here, \( \mathcal{E}(\hat{f}|G^{k+1}) \) is equivalent to \( \mathcal{E}(f|G^k) \) in Eq. (15), and \( A \) is an approximation of the top layer energy \( \mathcal{E}_{\text{top}} \).

In following three sections, we detail lines 4–10 of Algorithm 1 and explain how we derive \( \mathcal{E}(\hat{f}|G^{k+1}) \), optimize it, and obtain the desired solution \( \{f, G^{k+1}\} \) from \( \{f, G^{k+1}\} \).
Energy Approximation using Temporary Graphs

We now briefly explain the conversion from $\mathcal{E}(f, G^{k+1})$ in Eq. (15) to $\hat{\mathcal{E}}(\hat{f}|G^{k+1})$ in Eq. (17). For detailed derivations, please refer to the supplementary material.

To relax the joint inference of $\mathcal{E}(f, G^{k+1})$, we create a temporary graph $\tilde{G}^{k+1}$ as an approximation of $G^{k+1}$ by duplicating the top layer of $G^k$ as $L_k^t = (V_k^t, E_k^t)$ (line 4 of Algorithm 1). We illustrate $G^k$ and $\tilde{G}^{k+1}$ in Figures 3 (a) and (b), respectively. Here, a labeling $\hat{f}$ on $\tilde{G}^{k+1}$ (or $V_k^t$) can equivalently express all possible $f$ on $G^{k+1}$ (or $V_k$), because $V_k^t$ is the finest form of any possible $V_k$.

The labeling $\hat{f}$ is copied from $f$ at line 5. Substituting $\tilde{G}^{k+1}$ for $G^{k+1}$ into Eq. (15) to derive its approximation $\hat{\mathcal{E}}(\hat{f}|G^{k+1})$ in Eq. (17) with following $\mathcal{A}$.

\[
\mathcal{A}(\hat{f}) = \mathcal{E}_{\text{err}}(\hat{f}|\hat{L}_k^t) + \mathcal{E}_{\text{reg}}^{k+1}(\hat{f}|\tilde{G}^{k+1}) + \mathcal{E}_{\text{gra}}^{k+1}(\hat{f}|V_k^t). \quad (18)
\]

The conversion from $\mathcal{E}_{\text{top}}$ in Eq. (16) to this $\mathcal{A}$ is provably exact except for only terms $\mathcal{E}_{\text{gra}}^{k+1}$ and $\mathcal{E}_{\text{reg}}^{k+1}$ in $\mathcal{E}_{\text{err}}$ of Eq. (2).

Here, conversion of $\mathcal{E}_{\text{gra}}^{k+1}$ is tricky because we need to convert variables from $V_k$ to a labeling $\hat{f}$ on $V_k^t$. We observe that the region of each node $i \in V_k$ should optimally 1) be a connected component, 2) assigned a single label unique from neighbors, and 3) be the union of regions in $V_k$. Thus, we can treat nodes $i \in V_k$ as connected components $C_i$ of nodes in $V_k^t$ assigned the same label, i.e.,

\[
V_{k+1} \equiv \left\{ C_i \mid \text{nodes } y \in C_i \text{ in } V_k^t \text{ are assigned the same label } \hat{f}_c \right\}. \quad (19)
\]

This property allows us to rewrite $\mathcal{E}_{\text{gra}}^{k+1}(V_{k+1})$ in Eq. (11) as a function of $\hat{f}$. To further make inference tractable, we relax the connectivity of $V_{k+1}$ and treat $|V_{k+1}|$ as label costs [15] of $\hat{f}$, i.e., the number of unique labels $\hat{f}_i$ in $V_k^t$ without considering their spatial connections. In this manner, the formulation of Eq. (11) becomes the same as that of multi-region segmentation [15]. Following their model fitting approach based on alpha expansion move [9], we treat the label costs and the likelihood terms of Eq. (11) as pairwise submodular terms and unary terms, respectively. See more discussions in the supplementary.

Algorithm 1: Two-Pass Optimization Process

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{input}: Two images $I, I'$
\State \textbf{output}: Hierarchical graph $G$ and flow-alpha map $f$
\FlowInitialization{1}, $G \leftarrow G^1$
\FlowInitialization{2}, $f \leftarrow f^1$
\FlowFor{$k = 1, 2, \ldots$}
\FlowInitialization{3}, \textbf{bottom-up graph construction} \diamond
\FlowInitialization{4}, Create temporary $\tilde{G}^{k+1}$ by duplicating $V_k$ of $G^k$
\FlowInitialization{5}, Initialize temporary $\hat{f}$ by copying labels from $f$
\FlowInitialization{6}, \textbf{Perform local expansion moves} ($\hat{\mathcal{E}}(\hat{f}, \tilde{G}^{k+1}, V_k^t)$)
\FlowInitialization{7}, $\hat{f} \leftarrow \arg \min \hat{\mathcal{E}}(\hat{f}|\tilde{G}^{k+1})$
\FlowInitialization{8}, Create $G^{k+1}$ by merging nodes of $V_k^t$ in $\tilde{G}^{k+1}$
\FlowInitialization{9}, if rejection criteria is met then break;
\FlowInitialization{10}, Update solution: $\{f, G\} \leftarrow \{\hat{f}, G^{k+1}\}$
\FlowInitialization{11}, if any stopping criteria is met then break;
\FlowReturn{end}
\FlowEnd
\FlowFor{$k = H, \ldots, 1$} \diamond \textbf{top-down label refinement} \diamond
\FlowInitialization{12}, \textbf{Perform local expansion moves} ($\mathcal{E}(f, G, V_k)$)
\FlowInitialization{13}, $f \leftarrow \arg \min \mathcal{E}(f|G)$ with $f_i$ fixed for $\forall i \in V_{k-1}$
\FlowReturn {end}
\end{algorithmic}
\end{algorithm}

Algorithm 2: Local Expansion Moves [48, 47]

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{arguments}: (model $\mathcal{E}$, labeling $f$, graph $G$, target layer $V_T$)
\For{each target node $i \in V_T$}
\State Make neighborhood: $N_i \leftarrow \{i\text{'s neighbors}\} \cup \{i\}$
\State Make expansion region: $R_i \leftarrow \{N_i\text{'s descendants}\} \cup N_i$
\For{each candidate proposer}
\State Generate a candidate label $\ell = (T, \alpha)$
\State Apply a local expansion move using min-cut: $f \leftarrow \arg \min \mathcal{E}(f|G)$ with $f_j \in \{f_j, \ell\}$ for $j \in R_i$
\EndFor
\EndFor
\State return $f$;
\end{algorithmic}
\end{algorithm}

Optimization of Approximation Energy

In Figure 3 (c) and at line 6 of Algorithm 1, we minimize the approximation energy $\hat{\mathcal{E}}(\hat{f}|\tilde{G}^{k+1})$ of Eq. (17) with known $\tilde{G}^{k+1}$. To efficiently infer the continuous 5df domains in $\hat{f}$, we use the local expansion move method of [48, 47].
In its general form, the local expansion move algorithm repeatedly solves the following binary labeling problem for each target node $i \in V$ visited in sequence.

$$f^{(t+1)} = \text{argmin} \mathcal{E}(f_j | f_j \in \{ f_j^{(t)} \}, \ell) \quad \text{for} \quad j \in R_i. \quad (20)$$

Here, $R_i \subset V$ is a set of local nodes around the target node $i$ (named expansion region), and this operation tries to improve the labels of the local nodes $j \in R_i$ by assigning them either their current label $f_j^{(t)}$ or a candidate label $\ell$. We use graph cuts [8] to solve this binary problem.

Our version of local expansion moves is summarized in Algorithm 2. During the bottom-up process, we randomly visit all nodes in the top layer $V_k$ (i.e., target layer $V_T$) at line 1, and update the labeling of local nodes. In order to apply the local expansion move algorithm for our hierarchical MRF model, we extended it in two ways. First, the expansion region $R_i$ is extended from $i$’s neighbors ($N_i$ at line 2) to include all its descendants (line 3). Second, when generating candidate labels $\ell$ for the target node $i$ (line 5), we use four types of candidate proposers listed below.

- **Expansion proposer** generates a label by copying the current label as $\ell \leftarrow f_i$. This tries to propagate the current label $f_i$ to nearby nodes in $R_i$ as explained in [48].
- **Cross-view proposer** refers to the current labeling $f^t$ of the other image, and uses a label $f'_j$ that gives warping to the target node region $\Omega_i$ as a candidate $\ell$, using inverse warp of $f'_j$. This is similar to view propagation in [7, 6].
- **Merging proposer** generates labels $\ell \leftarrow w_i f_i + w_j f_j$ as weighted sums of $i$’s current label $f_i$ and its neighbors’ labels $f_j$, $j \in N_i$. The weights $w_i, w_j \in [0, 1]$ are proportional to their region sizes $|\Omega_i|, |\Omega_j|$. This is a new extension for promoting better region merging.
- **Perturbation proposer** generates labels $\ell \leftarrow f_i + \Delta$ by randomly perturbing the current label $f_i$. Similarly to [48, 7], we iterate between lines 5 and 6 several times while reducing the perturbation size $|\Delta|$ by half.

**Incremental Layer Construction**

In Figure 3 (d) and at line 7 of Algorithm 1, we create a new graph $G^{k+1}$ by merging nodes of $V_k'$ in $G^{k+1}$. Here, $V_{k+1}$ is created from $V_k'$ and $\hat{f}$ using the variable conversion of Eq. (19). After merging regions, we check the number of new foreground regions at the top layer. If it is zero (line 8), we reject the new solution and stop the graph construction process. Otherwise, we adopt the new solution $\{ f, G^{k+1} \}$ as $\{ f, G \}$ (line 9). Later we check the foreground count again and if it is one or not reduced from the previous iteration (line 10), we stop the graph construction process.

**4.2. Top-Down Labeling Refinement**

After the bottom-up phase, we further refine the labeling $f$ during the top-down phase shown at lines 12–14 of Algorithm 1. Since $G$ is held fixed during this step, we can directly optimize $\mathcal{E}(f | G)$ using local expansion moves without requiring the energy conversion described in Sec. 4.1. During this phase, we visit layers $V_i$ in $G$ in top-down order (from $k = H$ to $k = 1$) and apply local expansion moves with $V_k$ as a target layer $V_T$. Here, the labeling $f$ for the higher layers $V_i$ ($i > k$) does not change, because the expansion regions $R_i$ only contain nodes in layers $V_k$ and below.

**5. Implementation Details**

We now discuss initialization steps and features used in our method. See the supplementary material for details.

**5.1. Local HOG Features**

The images are first resized so that their larger dimension becomes 512 pixels. A Gaussian pyramid is then built for each image (we use 1 octave and 1 sub-octave). From each pyramid layer, we densely extract local histogram of gradient (HOG) feature descriptors [14]. These features are extracted at every pixel on the image grid from patches of size $27 \times 27$ pixels. Our HOG descriptors are 96-dimensional.

We use a $3 \times 3$ cell grid for each patch and 16 equally spaced bins for the oriented gradient histograms. Each gradient histogram thus has 16 bins for signed gradients and 8 bins for unsigned gradients. The histograms for each contiguous $2 \times 2$ block of the $3 \times 3$ cell grid are aggregated to form a 24-dimensional vector. These are then L2-normalized followed by element-wise truncation (using a threshold of 0.5). Four such vectors are concatenated to form the final 96-dimensional HOG descriptor. These HOG features are used to compute the flow data terms $E^t_{\text{flo}}$ described earlier. They are also used to construct bag of visual words (BoW) histogram features required during the initialization stage.

**5.2. BoW Histogram Features**

Each HOG descriptor is vector-quantized using a K-means codebook of size 256. Next, BoW histograms are computed from several overlapping image patches of size $64 \times 64$ pixels. These patches are sampled every 4 pixels (both horizontally and vertically) in the image. We use integral images (one per visual word) to speed up the BoW histogram computation. All the visual words are aggregated into a histogram. This is repeated for $2 \times 2$ sub-regions. The five BoW histograms are then L2-normalized followed by element-wise square root normalization\footnote{This is equivalent to using a Hellinger kernel instead of the Euclidean distance to measure the similarity of two feature vectors.}. The 256-dimensional histograms are concatenated to form 1280-dimensional BoW histogram features.

**5.3. Initialization**

During initialization, initial flow candidates and foreground/background color models for each image are com-
puted as follows. First, dense matching is done using the BoW features at three levels in the image pyramid. The Euclidean distances between each pixel feature in the first image and features for all pixels within a search window in the second image are computed. Fortunately, this is quite fast due to the sparsity of the BoW features. The best match is stored as a flow candidate. The ratio of the Euclidean distances of the best and worst match is computed as follows. First, dense matching is done using the BoW features at three levels in the image pyramid. The foreground. Geodesic distances from the image boundary (right) are used to add background clues (black regions).

![Figure 4. Min/Max ratios of BoW feature matching distances in local windows (middle). Low ratios (blue) are likely to suggest foreground. Geodesic distances from the image boundary (right) are used to add background clues (black regions).](image)

5.4. Efficient Implementation

Three key ideas allow our optimization method to be efficiently implemented. First, unary terms $E^i_{f_p}(f_i)$ and $E^i_{seg}(f_i)$ in Eq. (2) can be efficiently computed using the tree structure of $G$. Specifically, the unary cost $E^p(\ell)$ of a node $p \in V_l$ is computed as the sum $\sum_c E^c(\ell)$ over its children $c \in V_{l-1}$, if their labels $\ell$ are the same. This constant-label property is satisfied during local expansion moves because the candidate label $\ell$ is the same for all nodes in an expansion region $R_l$. Thus, at line 6 of Algorithm 2, we compute the unary cost $E^j(\ell)$ for $j \in R_l$ by sequentially summing them up from bottom to top layer nodes. Second, we exclude the pixel layer $L_0$ from the graph $G$ during the main iterations. We add it to $G$ just before the last refinement step in the top-down phase ($k = 1$ at line 13 of Algorithm 1). Finally, we use efficient graph cuts [8] at line 6 of Algorithm 2, instead of QPBO [30]. This is possible because our energy is submodular under (local) expansion moves [48, 47]. The proofs are in the supplementary.

6. Experiments

We evaluate our method for flow and segmentation accuracy and compare it to existing methods on our new dataset. **Dataset.** Our dataset comprises of 400 image pairs divided into three groups -- FG3DCar contains 195 image pairs of vehicles from [35]. JODS contains 81 image pairs of airplanes, horses, and cars from [42]. PASCAL contains 124 image pairs of bicycles, motorbikes, buses, cars, trains from [20]. See Figure 6 for some examples from each group.

**Flow accuracy.** We evaluate flow accuracy by the percentage of pixels in the true foreground region that have an error measure below a certain threshold. Here, we compute the absolute flow endpoint error (i.e., the Euclidean distance between estimated and true flow vectors) in a normalized scale where the larger dimensions of images are 100 pixels.

**Segmentation accuracy.** We use the standard intersection-over-union ratio metric for segmentation accuracy. As existing flow estimation methods do not recover common foreground regions, we compute them by post-processing the estimated flow maps. Specifically, given the two flow maps, we do a left-right consistency check with a suitable threshold and treat pixels that pass this test as foreground.

**Settings.** We strictly fixed all the parameters throughout the experiments as follows. For the data and graph term parameters, we set $\{\theta_{fp}, \theta_{seg}, \lambda_{seg}, \lambda_{mod}, \lambda_{col}\} \leftarrow \{0.25, 2.4, 6.5, 0.8, 125, 1\}$. For regularization parameters $\{\lambda_{st1}, \lambda_{st2}, \lambda_{st}, \lambda_{pc1}, \lambda_{pc2}, \lambda_{pc}\}$ associated with the pixel layer (edges $E_p$ and $E^{st}_p$) we use $\{0.5, 20, 0.005, 10, 200\}$, and for the other edges we use $\{0.1, 4, 20, 0.04, 8, 200\}$. See the supplementary material for our strategy of tuning parameters. Our method is implemented using C++ and run by a single thread on a Core i7 CPU of 3.5 GHz.

6.1. Comparison with Existing Approaches

For correspondence, we compare our method with SIFT Flow [36], DSP [29] and DFF [53]. We also evaluate our method using only the single layer model without hierarchy, which can be done by skipping the bottom-up construction step in Algorithm 1. This single layer method can be seen as a variant of [48]. For cosegmentation, we compare our method with Joulin et al. [24] and Faktor and Irani [17] based only on segmentation accuracies. We summarize average accuracy scores for each subset in the upper part of Table 1, where flow accuracy is evaluated using a threshold of 5 pixels. The plots in Figure 5 show average flow accuracies with varying thresholds. As shown here, our method achieves the best performance on all three groups at all thresholds. Our average flow accuracies for FG3DCar, JODS and PASCAL, respectively, are up to 45%, 19% and 34% higher than SIFT Flow (best existing method).

4We omit results of HaCohen et al. [19] for its low performance on our dataset. It could not find any correspondence for many image pairs.

5For Joulin et al. [24] that cannot identify the “foreground” label from $\{0, 1\}$, we refer to ground truth and choose for each image pair either 0 or 1 to maximize the scores. Results of their extension method [25] are omitted since we could not observe improvements over [24] in our settings.

6We omit results of Dai et al. [13] as it did not work for many image pairs. The method seems to fail in finding matches with learned templates.
Table 1. Benchmark results. FAcc is flow accuracy rate for an error threshold of 5 pixels in a normalized scale. SAcc is segmentation accuracy by intersection-over-union ratios. SAcc scores (⋆) of optic flow methods are computed by post-processing using left right consistency check.

<table>
<thead>
<tr>
<th>Optic flow / cosegment. Methods</th>
<th>FG3DCar</th>
<th>JODS</th>
<th>PASCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>0.829</td>
<td>0.756</td>
<td>0.439</td>
</tr>
<tr>
<td>Our single layer [48]</td>
<td>0.727</td>
<td>0.727</td>
<td>0.413</td>
</tr>
<tr>
<td>SIFT Flow [36]</td>
<td>0.604</td>
<td>0.420</td>
<td>0.352</td>
</tr>
<tr>
<td>DSP [29]</td>
<td>0.487</td>
<td>0.285</td>
<td>0.465</td>
</tr>
<tr>
<td>DFF [53]</td>
<td>0.493</td>
<td>0.326</td>
<td>0.303</td>
</tr>
<tr>
<td>Faktor and Irani [17]</td>
<td>–</td>
<td>0.689</td>
<td>–</td>
</tr>
<tr>
<td>Joulin et al. [24]</td>
<td>–</td>
<td>0.461</td>
<td>–</td>
</tr>
</tbody>
</table>

Figure 5. Average flow accuracies evaluated by endpoint errors with varying thresholds. Ours always shows best scores. DFF [53] is not robust due to lack of explicit regularization.

Figure 6. Dataset.

Figure 7. Correspondence results.

Figure 8. Cosegmentation results.

7. Conclusion

We have presented a joint method for cosegmentation and dense correspondence estimation in two images. Our method uses a hierarchical MRF model and jointly infers the hierarchy as well as segmentation and correspondence using iterated graph cuts. Our method outperforms a number of methods designed specifically either for correspondence recovery [36, 29, 19, 53] or cosegmentation [24, 25, 17, 13]. We provide a new dataset for quantitative evaluation. Enforcing left-right consistencies on flow and segmentation maps for two images, or by using multiple images [12, 55, 25] are promising avenues for future work.

Acknowledgments. We thank Richard Szeliski, Pushmeet Kohli and Tianfan Xue for valuable feedback. T. Taniai was partially supported by JSPS KAKENHI Grant Number 14J09001 and a Microsoft Research Asia PhD fellowship.
References


