Integrating Graph Partition and Matching with Editing

Liang Lin, Xiaobai Liu and Song-Chun Zhu

Abstract

Many computer vision tasks can be posed as either a graph partitioning (segmentation) problem or a graph matching (correspondence) problem. In this paper, we present an integrated solution for simultaneous graph partition and matching with graph editing. Our objective is to find and match an unknown number of common graph structures (objects) in two images – a problem arising in recent object recognition research, such as object categorization and unsupervised learning. Unlike previous segmentation and correspondence problems motivated by large motion or wide baseline stereo, the scenes and objects in the two images may have quite different appearance but share similar graphic structures. Given two images, we first extract their primal sketches [11] as attributed graphs which represent both textures (by key points) and structures (by line segments and primitives). The points, line segments, and primitives are vertices of the graphs. Both graphs are partitioned into \( K + 1 \) layers of subgraphs so that \( K \) pairs of subgraphs are matched and the remaining layer contains unmatched background. Each pair of matched subgraphs represent a common object under geometric deformations and structural editing with graph operators. We design a data-driven Markov Chain Monte Carlo (DDMCMC) algorithm to explore the joint partition and matching space effectively. It has two iterative components: (i) A Swendsen-Wang Cut component in the partition space that can flip, in each step, a chunk (connected component) of the graph where the vertices are often strongly coupled due to proximity and similar appearance; (ii) A Gibbs sampling step for matching some connected components to the other graph. To prune the matching space, we grow a number of vertices to form some connected components so that each connected component has a small number of candidate matches. We apply the algorithm to a number of applications with comparison to the state of the art methods: (i) multi-object wide-baseline matching (underlying both rigid and non-rigid motions) with occlusions, (ii) automatic detection and recognition of common objects from templates, and (iii) human articulate motion analysis between images from a video sequence.

Index Terms

Graph Matching, Graph Partitioning, DDMCMC, Motion Analysis, Swendsen-Wang Cut.

This manuscript is submitted to IEEE Trans. on PAMI. A short version was published in CVPR2007 [14]. email contacts: llin@lotushill.org, xbliu@lotushill.org, sczhu@stat.ucla.edu
I. Introduction

A. Motivation and objective

Many computer vision tasks can be posed as either a graph partitioning (or coloring) problem, such as image segmentation [1], [30] and scene labeling or a graph matching (or correspondence) problem, such as wide baseline stereo [4], [19], [10], large motion [19], [6], [35], object tracking, and shape recognition [15], [5], [29], [37], [38] tasks.

This paper is aimed at an integrated solution for simultaneous graph partition and matching with graph editing. Our objective is to find and match an unknown number of common graph structures (objects) in two images. Fig. 1 shows an example in our experiments. The two input images (the top row) include three pairs of objects: person, cars, and meters with similar structure. This problem arises in recent object recognition research, such as object categorization and unsupervised learning.

Given two images, we first extract their primal sketches [11] as attributed graphs which represent both textures (by key points) and structures (by line segments and primitives). The points, line segments, and primitives are vertices of the graphs. The primal sketch graphs may be imperfect and should be fixed in the matching process through graph editing operators. Both graphs are partitioned into $K+1$ layers of subgraphs so that $K$ pairs of subgraphs are matched and the remaining layer contains unmatched background. Each pair of matched subgraphs represent a common object under geometric deformations and structural editing with graph operators to account for effects such as occlusion.

Unlike previous segmentation and correspondence problems motivated by large motion or wide baseline stereo, the scenes and objects in the two images may have quite different appearance and undergo large motion or non-rigid deformations, but they share similar graphic structures.
As some objects may have less texture, we have to relay on their contour information for the
matching, rather than independent interesting points, such as SIFT [16] or corners [23].

B. Related Works

Graph matching has been extensively studied in the literature for numerous vision tasks. We
divide the existing methods into three categories.

Category 1: single layer and point based. These algorithms match local independent features
without explicit graph structures, such as Harris corners, KLT features [23], scale invariant
features [16], local edge features [4] and geometric blur descriptors [5], [26], [8]. The two sets
of feature points are matched under a rigid affine transform plus non-rigid and locally smooth
distortions accounted by an TPS (thin plate spline) model. These point features are often robust
against certain geometric distortions and illumination changes, but carry little information about
larger object structures. Current state of the art algorithms include the iterative closest point (ICP) algorithm [26] and the soft-assignment algorithm [8].

**Category 2: single layer graph based.** Algorithms in this category match explicit graph structures with graph editing, such as skeleton (medial axis) graphs [37] and shock graphs [21], [25]. For recognizing flexible and articulated objects, the distance should be calculated based on not only the similarity of parts but also their connections and relative positions. Thus graph operators are introduced with cost to edit the graphs to achieve perfect structural match (isomorphism). In this paper, we use the terms *graph structure* or *graph topology* for connectivity between vertices. Recently it is also applied to unsupervised learning of object categories through matching parse trees across multiple images [27]. Some recent work on shape recognition represents the graph structures implicitly for computational efficiency, for example shape context [5] and shape matching [29]. These methods can be used for initialization in a bottom-up phase.

**Category 3: multi-layer point based.** This category includes layered motion by EM clustering [33], [24], [7] for small motion, and RANSAC based methods [18], [28], [19] for large rigid motion. A state of the art algorithm in this category is by **Wills and Belongie algorithm** [35] which computes large motion segmentation using RANSAC iteratively based on local texture features.

Our method belongs to **category 4: multi-layer graph based matching with explicit graph editing.** It is aimed at more general cases arising in recent object categorization and learning tasks where many of the traditional assumptions no longer hold, for example, the slow and smooth motion in layered motion [34], rigid transform and static objects in wide baseline stereo, foreground and background segmentation in the medial axis based shape recognition. To work
on real images, we adopt a primal sketch representation which represent both texture rich (non-sketchable) areas and textureless (sketchable) areas.

Fig. 2. Illustration of the layered graph matching algorithm. (a) Two graphs are partitioned into an unknown number of $K + 1$ layers of subgraphs $\{g_1, g_2, \ldots, g_k\}$ and $K$ pairs of subgraphs are matched and the remaining layer contains unmatched backgrounds $g_0$. (b) To prune the search space, some bottom-up steps compute some strongly coupled vertices as connected components and “freeze” them as composite node. Each node has two sets of candidates: the dots on the red line represents matching candidates and the dots on the blue lines represent label candidates. The sizes of these dots represent their weights. A stochastic algorithm is presented to sample effectively in the joint partition and match space.

C. Overview of the algorithm

Following the example in Fig. 1, Fig. 2 illustrates the computational diagram of the algorithm.
From two input images, we first extract two attributed graphs: one is called the source graph $G$ and the other the target graph $G'$. The vertices of these graphs are keypoints and image primitive. The keypoints are often isolated while the image primitive, as shown in Fig. 3.(d), have various degrees ($0 \sim 4$) of connections with other vertices. The graphs $G$ and $G'$ may not be perfect and should be edited in the partition and matching process. Both graphs are partitioned into $K + 1$ layers of subgraphs so that $K$ pairs of subgraphs are matched and the remaining layer contains unmatched background. Each pair of matched subgraphs represent a common object under geometric deformations and structural editing with graph operators to account for effects such as occlusion. Thus the distance measurement between two matched subgraphs includes geometric deformations, appearance dissimilarities, and the cost of graph editing operators.

Fig. 2(b) illustrates, for a solution, each vertex in the source graph has two set of candidate labels: the dots on the red line represents matching candidates and the dots on the blue lines represent label candidates. For example, consider a source graph $G$ of $N$ nodes and a target graph $G'$ of $M$ nodes, for a $K$-layer partitioning. The solution space has in the order of $O((KM)^N)$ states. This joint solution space for partitioning and matching is so large that it is prohibitive to search it exhaustively. The main contribution of the paper is to design a Markov chain Monte Carlo algorithm that can explore this solution space effectively. The algorithm is, in a spirit, similar to the data-driven Markov chain Monte Carlo (DDMCMC) used for image segmentation [30] and recognition [38].

The algorithm consists of two iterative components, and both components are driven by bottom-up computations.

**Component 1: graph matching.** As each pair of subgraphs are matched by a TPS (thin-plate spline) transform and graph editing operators. This involves more parameters than the rigid
affine transformation where traditional sampling algorithms, like RANSAC [18] and MLESAC [28], are applied. To narrow the combinatorial number of possible matches, we start from some strong seed nodes in the source graph that have fewer possible match candidates in target graph. The seed nodes then grow to form what we call “seed graphs” using a two layers branch-and-bound algorithm. The nodes in a seed graph are “frozen” into a composite vertex and must match together. A seed graph in source graph and its match in target graph propose the strong candidates for growing into a pair of matched objects. As the seed graphs largely prune the matching candidates, we use Gibbs sampling method to flip the labels (the blue lines) for the seed graphs and expand to the surrounding nodes. Some similar strategy was used in the multi-level Swendsen-Wang cut [2] and local feature selection[31].

Component 2: graph partition. We adopt the Swendsen-Wang cuts algorithm [1] for sampling the partition space, i.e. flipping the candidates in the red lines (Fig. 2 (b)). Unlike the single site Gibbs sampler, the SW-cut algorithm groups adjacent nodes into connected components and flips the color of a large subgraph in each step. The connected components are drawn through probabilistic sampling by turning on/off edges in the graph according to the local probabilities. These local probabilities define how likely two adjacent nodes in a graph belong to the same color, based on their similarity in appearance and goodness of alignment.

Both components account for the strong coupling of some vertices in the graphs and make the computation feasible. We demonstrate this algorithm to a number of vision tasks: (i) large motion with both opaque and transparent occlusions, and both rigid and non-ridge transforms; (ii) object detection and matching with similar structure and different appearance for both cartoon and real images, and (iii) articulate motion. We compare our algorithm with some of the state of the art methods in each tasks respectively.
The paper is organized as follows. We first present the problem formulation in the Bayesian framework in Section II, and discuss the distance measures in Section III. Then Section IV presents the algorithm, and Section V discusses a set of experiments with comparisons. Finally the paper is concludes with discussions in Section VI.

II. PROBLEM FORMULATION

A. Background on primal sketch

![Primal Sketch Example](image)

Fig. 3. An example of primal sketch. (a) An input image. (b) A sketch graph which divides the image into two parts: the sketchable part in (e) and non-sketchable part in (c). (c), (d) Some examples in the image primitive dictionary. (f). The synthesized image. Results from [11], [12].

We start with a brief review of the primal sketch representation on which the graph partition and matching is based. Proposed in Marr’s book, primal sketch is supposed to be a symbolic and perceptually lossless representation of the input image. A mathematical model for primal sketch was proposed in [11], [12]. As Fig. 3 shows, given an input image I, we compute a
sketch graph $G$ whose vertices are image primitives shown in Fig.3.(d). These primitives are blobs, terminators, edges, bars (ridges), junctions, and crosses which have $0 \sim 4$ degrees of connections (“arms”) with other primitives. Each arm of the primitive has some intensity profile with a width of $5 \sim 7$ pixels. We denote the sketch graph $G$ by a 3-tuple $G = (V, E, A)$ with $V$ being a set of vertices (nodes) for the image primitives, $E$ the set of links for connections, and $A$ the attributes of the primitives including position, orientation, intensity profile etc. It is worth clarifying that the edges and bars in the image are primitives represented by the vertices, in $V$, they should not be confused with the links (often called edges) of the graph in $E$. The latter represent the connections of primitives and they are not seen in the image.

This sketch graph divide the image lattice into a sketchable parts in Fig.3.(e) and the non-sketchable parts in Fig.3.(c). The non-sketchable parts are stochastic texture modeled by Markov random field models, for example, each texture is represented by some histograms of Gabor filter responses. The pixels on the sketchable part are reconstructed by the primitives under some transformations, while the pixels on the non-sketchable part are synthesized from the markov random field models using the sketchable part as boundary condition. Fig. 3.(f) is a synthesized image which is almost perceptually equivalent to the input image.

In this paper, we compute the two sketch graphs $G$ and $G'$ from a pair of input images $I$ and $I'$ independently by a primal sketch approach.

\[
G = \arg \max_G p(I|G)p(G) \tag{1}
\]

\[
G' = \arg \max_{G'} p(I'|G')p(G'). \tag{2}
\]

$G$ is used as the source graph and $G'$ the target graph.

Fig. 4 shows an example for the sketch graphs for the matching example used in Fig. 1. More examples are shown in the experiment section.
$G$ and $G'$ are imperfect due to inference uncertainties and errors, and some part may be missing due to occlusion. Therefore we shall adopt graph operators to edit the graphs in the matching and partitioning process to correct the various errors. In the rest of the paper, we shall focus the discussion on the partition and matching algorithm, assuming that $G$ and $G'$ are given.

There are two exceptions. (i) In some tasks, such as object detection, the source graph $G$ is given as the template and thus we only need to compute $G'$. (ii) For certain structured texture areas, the primitives and sketch graphs may not be inferred stably, we may compute some keypoints as isolated vertices and add them to the sketch graphs.

B. Graph partition and coloring

The goal of graph partition is to divide the set of nodes $V$ in $G$ into a $K + 1$ subgraphs with the number of objects $K$ unknown. We denote a graph partition of $G$ by

$$\Pi_K(G) = \{g_0, g_1, \cdots, g_K\}$$

(3)

Each subgraph $g_i = (V_i, E_i, A_i)$ is a separate layer and is itself an attribute graph, with

$$\bigcup_{i=0}^{K} V_i = V, \quad V_i \cap V_j = \emptyset,$$

(4)
and

\[ E_i = \{ <s, t>: s, t \in V_i, <s, t> \in E \}. \] (5)

The vertices in \( V_i \) receive an unique color (label) \( l(v) \in \{0, 1, ..., K\} \). When the graph \( G \) is edited on-line, the partition above will be updated as well.

We assume \( K \) should be small and each subgraph is preferably contiguous. Thus we introduce a prior model for the partition \( \Pi_K \)

\[ p(\Pi_K | G) \propto \exp\{-\alpha K - \beta E(\Pi_K | G)\}, \quad \text{with} \quad E(\Pi_K | G) = \sum_{<s, t> \in E} 1(l(s) = l(t)). \] (6)

\( 1(x) \in \{-1, +1\} \) is an indicator function for a Boolean variable \( x \).

C. Integrating partition with matching

Given a source graph \( G = (V, E, A) \) and a target graph \( G' = (V', E', A') \), we can denote the graph matching function by

\[ \Psi: V \mapsto V' \cup \{\emptyset\} \] (7)

For each vertex (or node) \( v \in V \), \( \Psi(v) \in V' \) or it has no match in \( V' \) with \( \Psi(v) = \emptyset \). The vertices should not matched as independent points, and we shall impose the graph structure on the match later.

To couple with the graph partition formulation \( \Pi_G \), we rewrite \( \Psi \) in \( K \) matching functions,

\[ \Psi_i: V_i \mapsto V'_i \cup \{\emptyset\}, \quad i = 1, 2, ..., K. \] (8)

As the result of matching, the target graph \( G' \) is also partitioned into \( K+1 \) subgraphs \( \{g'_0, g'_1, \cdots, g'_K\} \), where subgraph \( g'_0 \) has no correspondence in the source graph. The prior over \( \Psi_i \) is defined so that two adjacent nodes in a subgraph \( g_i \) should either be both matched or both unmatched to
account for the occlusion effects.

\[
p(\Psi_i|g_i) \propto \exp\{-E(\Psi_i|g_i)\} \quad \text{with} \quad E(\Psi_i|g_i) = \sum_{<s,t> \in E_i} 1(\Psi_i(s) = \emptyset)1(\Psi_i(t) = \emptyset).
\]  

(9)

Now we have \( K \)-pairs of matched attribute graphs,

\[
(g_i, g'_i), i = 1, 2, ..., K.
\]  

(10)

where \( g_i \) is transformed into \( g'_i \) by a set of geometric transforms, photometric (appearance) transforms, and topological graph editing operators, denoted by

\[
\Phi_i = (\Phi^\text{geo}_i, \Phi^\text{pho}_i, \Phi^\text{top}_i)
\]  

(11)

We shall define the matching distances between \( g_i \) and \( g'_i \) based on the three aspects. From \( g_i, \Phi_i, \Psi_i \), one can predict \( g'_i \) in the target graph with a prediction probability \( p(g'_i|\Phi_i, \Psi_i, g_i) \). It shall be defined as a product of the probabilities for the photometric, geometric, and topological transforms in the next section.

By combining graph partitioning and graph matching, our objective is to compute the following solution configuration \( W \) from two graphs \( G, G' \),

\[
W = (\Pi_K, \{\Phi_i, \Psi_i\}^K_{i=0})
\]  

(12)

III. DISTANCE MEASURES

For a pair of graphs \( g_i, g'_i \) matched through a transform \( \Phi_i = (\Phi^\text{geo}_i, \Phi^\text{pho}_i, \Phi^\text{top}_i) \), we define the distance or equivalently energy and probability over the three type of attributes,

\[
E(g_i, g'_i) = E^\text{geo}(g_i, g'_i) + E^\text{pho}(g_i, g'_i) + E^\text{top}(g_i, g'_i).
\]  

(13)

In the following, we define the 3 types of distances. Note that each unmatched vertex receives penalty by the topological distance with graph operators.
A. Geometric distance

The geometric transform $\Phi_{gi}^{\text{geo}}$ from $g_i$ to $g'_i$ includes a global affine transformation $S_i$ and a TPS warping for local deformation $F_i(x, y)$ in the 2D domain $\Lambda_i$ covered by $g_i$.

$$\Phi_{gi}^{\text{geo}} = (S_i, F_i)$$  \hspace{1cm} (14)

The affine transformation $S_i$ is a matrix

$$S_i = \begin{pmatrix} S_{ix} & 0 \\ 0 & S_{iy} \end{pmatrix} \begin{pmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{pmatrix} \begin{pmatrix} 1 & \gamma_i \\ 0 & 1 \end{pmatrix}$$  \hspace{1cm} (15)

where $\theta$ is the rotation angle, $S_{ix}$ and $S_{iy}$ denote scaling, and $\gamma_i$ is shearing. The energy on $S_i$ is defined as

$$E_{\text{Aff}}^{\text{geo}}(S_i) = E_{\text{rot}}(\theta_i) + E_{\text{scl}}(S_{ix}, S_{iy}) + E_{\text{shear}}(\gamma_i).$$  \hspace{1cm} (16)

Another matching cost is the residue between the mapped position of each vertex $v \in V_i$ and the position of $v' = \Psi_i(v)$.

$$E_{\text{res}}^{\text{geo}}(V_i, V'_i) = \sum_{v \in V_i, \Psi_i(v) \neq \emptyset} \lambda_{\text{geo}}^{\text{geo}}((x_v - x_{v'})^2 + (y_v - y_{v'})^2).$$  \hspace{1cm} (17)

Given the vertices matches, there is one more term on the TPS deformation $F$

$$E_{\text{TPS}}^{\text{geo}}(F_i) = \lambda \int \int_{\Lambda_i} (F_{xx}^2 + 2F_{xy}^2 + F_{yy}^2)dxdy.$$  \hspace{1cm} (18)

Then the overall energy for geometrical transform is

$$E^{\text{geo}}(g_i, g'_i) = E_{\text{Aff}}^{\text{geo}}(S_i) + E_{\text{res}}^{\text{geo}}(V_i, V'_i) + E_{\text{TPS}}^{\text{geo}}(F_i).$$  \hspace{1cm} (19)

We may drop the affine term if we allow free rigid affine transforms of the objects between two images.
B. Photometric distance

As Fig. 3.(d) shows, each vertex (primitive) has a center landmark and 0 ~ 4 axes (arms) for connecting with other vertices. For two matched vertices \( \{v \in V_i, v' \in V'_i\} \), \( v' = \Phi^\text{pho}_i(v) \) in \( g_i \) and \( g'_i \), the photometric distance between them is defined on their intensity profiles of their matched arms. The penalty for unmatched arms is included in the graph editing cost.

Suppose there are \( c \) matched arms between \( v \) and \( v' \), we denote by \( \mu^\text{arm}_j(v) \), \( j = 1, ..., c \) the intensity profile perpendicular to each arm of \( v \). For primitives with no arms, such as blobs, we set \( c = 1 \) and calculate the profile along its radius. After geometric alignment, the the photometric energy is measured by

\[
E^\text{pho}(g_i, g'_i) = \sum_{(v, v')} \sum_j \lambda^\text{pho} \| \mu^\text{pho}_j(v) - \mu^\text{pho}_j(v') \|^2
\]

The photometric distance often provides matching cues for motion or stereo tasks, but for some applications, \( E^\text{pho}(V_i, V'_i) \) can be dropped if we only care about the common shapes regardless of their appearance differences.

C. Topological distance

Preserving graph connectivity structure is an important aspect in graph matching. Especially for object recognition under occlusions and learning common templates from multiple images. Editing the graphs is also needed for fixing the errors in primal sketch and distinguishes our approach from many previous works that match merely a set of points [33], [16], [23], [8].

For a node \( v \in V_i \) in \( g_i \), we denote its neighbors by

\[
\partial v = \{ u : <v, u> \in E_i \}
\]

Suppose \( v' = \Psi_i(v) \in V' \) is a matched node. The match \( \Psi_i \) is said to be isomorphic between \( v \)
Fig. 5. Two basic graph operators for editing graphs. $\delta^A$: adding/deleting a vertex, and $\delta^B$: adding/deleting a link or arm. Some typical examples of editing is shown under each operator.

and $v'$, if

$$u \in \partial v \iff u' \in \partial v', \quad u' = \Psi_i(u), u = \Psi_i^{-1}(u').$$

(22)

If the match is not isomorphic, a number of operators have to be applied, such as adding/deleting vertices and links. Previous graph editing work involved the medial axis and shock graphs[37], [21].

Considering computing efficiency, we define only two basic graph operators $\delta^A$ and $\delta^B$ in Fig. 5, and they are associated with costs $\text{cost}(\delta^A)$ and $\text{cost}(\delta^B)$. They are capable of correcting mostly topological transition. In the literature, large and composite graph operators are learned and used in [32].

Suppose the two operators $\delta^A$ and $\delta^B$ are used $N^A$ and $N^B$ times respectively between a pair
of matched subgraphs \((g_i, g'_i)\). More precisely, we have

\[
M = \sum_{v \in V_i} 1(\Psi_i(v) = \emptyset) + \sum_{v' \in V'_i} 1(\Psi^{-1}_i(v') = \emptyset) \tag{23}
\]

\[
N = \sum_{\langle v_m, v_n \rangle \in E_i} 1(\Psi(v_m), \Psi(v_n) \not\in E'_i) + \sum_{\langle v'_m, v'_n \rangle \in E'_i} 1(\langle v'_m, v'_n \rangle \not\in E_i). \tag{24}
\]

Thus the topological distance for matching \(g\) to \(g'\) is,

\[
E_{\text{top}}(g_i, g'_i) = M \text{cost}(\delta^A) - N \text{cost}(\delta^B), \quad i = 1, 2, \ldots, K. \tag{25}
\]

**D. Distance for the unmatched background**

The three distances defined so far are applicable to the matched subgraphs \((g_i, g'_i), i = 1, 2, \ldots, K\). For the unmatched background subgraphs \((g_0, g'_0)\), only the topological editing distance is applicable,

\[
E(g_0, g'_0) = E_{\text{top}}(g_i, g'_i) = (|V_0| + |V'_0|)\text{cost}(\delta^A) + (|E_0| + |E'_0|)\text{cost}(\delta^A). \tag{26}
\]

Thus the choice of background really depends on our penalty term \(\alpha K\) for the number of common objects. For a small \(\alpha\), we may allow some individual vertex being matched as common graphs. Then the ratio \(\frac{\alpha}{\text{cost}(\delta^A) + \text{cost}(\delta^B)}\) decides the threshold for how large a component that we prefer.

**IV. The Inference Algorithm for Partition and Matching**

**A. The Bayesian formulation**

Based on the definition of \(W = (\Pi_K, \{\Phi_i, \Psi_i\}_{i=0}^K)\) in Section II and the distance \(E(g_i, g'_i)\) in Section III, we can formulate the inference problem in a Bayesian framework.

We assume that the matching for each pair of subgraphs is independent. So we have,

\[
W^* = \arg \max_W p(\Pi_K, \{\Phi_i, \Psi_i\}_{i=0}^K | G, G')
\]

\[
= \arg \max_W p(\Pi_K | G) p(\Psi | \Pi_K, G) p(\Phi | \Psi, \Pi_K, G) p(G' | \Phi, \Psi, \Pi_K, G)
\]

\[
= \arg \max_W p(\Pi_K | G) p(\Psi | \Pi_K, G) \prod_{i=0}^K p(\Phi_i | \Psi_i, g_i) p(g'_i | \Phi_i, \Psi_i, g_i) \tag{27}
\]
$G'$ is deterministically decided from $G, \Pi_K, \Psi, \Phi$, thus the likelihood probability $p(G'|\Phi, \Psi, \Pi_K, G)$ is a delta function and can be dropped. It is more intuitive to rewrite it in an energy minimization problem,

$$W^* = \arg\min_W \{ \alpha K + \beta E(\Pi_K|G) + \gamma \sum_{i=1}^{K} E(\Phi_i|g_i) + \sum_{i=0}^{K} E(g_i, g'_i) \}.$$  \hspace{1cm} (28)

The first three terms are prior energy functions for the number of layers, the piecewise continuity in the partition labels, and the piecewise continuity in matching. The fourth term is the matching energy including the geometric transforms, photometric distance, and the costs of topological editing operators.

In the computation of $W$, the complexity is mostly with the partition and matching, once $\Pi_K$ and $\{\Psi_i\}$ are known, computing $\{\Phi_i\}$ is relatively easy and can be done deterministically by

$$\Phi_i = \arg\min E(g_i, g'_i), \ i = 1, 2, \ldots, K.$$  \hspace{1cm} (29)

Thus our discussion shall only be focused on $\Pi_K$ and the matching $\Psi$.

Each vertex $v$ in $G$ has two set of label candidates as Fig. 2.(b) illustrates. One set $L_v$ is for the labeling

$$l_v \in L_v \subset \{0, 1, \cdots, K\},$$  \hspace{1cm} (30)

and the other set $M_v$ is for matching the vertices in $G'$,

$$m_v \in M_v \subset V' \cup \{\emptyset\}.$$  \hspace{1cm} (31)

The two candidate sets are shown by the dots on the red line and blue line respectively.

The joint state solution space $\Omega_{\Pi,\Psi}$ is combinatorial. Consider a source graph of $N$ nodes and a target graph of $M$ nodes with $K$ objects. The solution space would have $O((KM)^N)$ possible solutions. To sample this space effectively, we need to narrow the combinations using bottom-up information for both $l_v$ and $m_v$. 
In the following, we shall present a Markov chain Monte Carlo algorithm to explore the joint partition and matching space, using the bottom-up computation in a strategy of DDMCMC [38], [30].

B. Bottom-up computation I: growing seed graph to narrow the match space

The general idea is similar in spirit to RANSAC and its variations[28], [31]. In our setting, a pair of matched objects may undergo non-rigid TPS (thin-plate spline) transform and graph editing operators, or they are not the same object at all. Thus the match space is much more complex than the low dimensional parameters spaces where RANSAC enjoys. Therefore, instead of drawing points randomly, we start with some “strong vertices” as “seeds” and grow them into seed graphs in a two-layers computing strategy, as shown in Fig. 2. The vertices in a seed graph are locked and have to match together. Thus a seed graph is a composite vertex and to be sampled as a whole, as the “seed 1”, “seed 2”, “seed 3” in Fig. 2 (b).

A seed graph should be maximally discriminative and robust. For example in regions with structured texture, some affine invariant key feature points will be helpful. In sketchable (textureless) parts, some structural elements are more informative, for example, the long curves in Fig. 10 and junctions in Fig. 14. In practice, we prefer to using long curves or composite curve groups as seed graphs, since junctions are often caused by motion and occlusion.

We define the long curves as initial seed graphs and curve groups as composite seed graphs, and the branch-and-bound algorithm is adopted for seed graph growing. This algorithm was used in matching skeleton (medial axis) graphs for object recognition [37], and can quickly prune bad branches and keep good candidates in the target graph $G'$, with the specific measurement distance. The growing process can be summarized with two steps.

Step 1: Initial seed graph growing. Each single vertex $v \in G$ is first initially matched to
all the vertices $v' \in G'$, and a matching probability is calculated based on the informative descriptors. We use SIFT descriptor [16] for those texture parts and shape context descriptor [5] for those sketchable parts. If this matching probability has a low entropy, then the vertex is considered “strong” as it has less ambiguity. We thus start with a random “strong” vertex $v$ with its candidates $M_v$, and then grow a seed into a small (initial) seed graph by adding neighboring vertices and re-calculate the matching probability using the branch-and-bound algorithm. Here we use the Hausdorff distance [22] to measure grown curves with candidates in this algorithm.

**Step II: Composite seed graph growing.** Given a set of initial seed graphs $\tilde{g} = (\tilde{V}, \tilde{E}, \tilde{A})$ identified in $G$, then for each candidate subgraphs in $G'$, we calculate the distance $E(\tilde{g}, \tilde{g}')$ as before to account spatial and geometric constraints, except that we ignore the graph editing distance to save computation. With this measurement, those initial seed graphs can be merged as composite seed graphs, based on second layer branch-and-bound algorithm.

The seed graph growing process is illustrated in Fig. 6. In this example, our goal is to search for a bear head template (shown on the right) as a target graph $G'$ in a cluttered cartoon image. The top raw shows some initial seed graphs, the middle row shows the candidate set $M_v$ for each of the seed graphs on top. last row in Fig. 6 shows the best candidate found for the bear head template. It is worthy mentioning that, for most real images, we can’t prune all ambiguous matches directly and need to keep a small set of candidates for next sampling step.

Identifying good seed graphs in $G$ is a matter of deciding the visiting/search order. In practice, the seed growing and merging process reduces the computation complexity. Since the nodes inside a seed graph are bonded and always receive the same color, they share two common vertical lines (red and blue) in Fig. 2 (b) with fewer candidates. To quantitatively illustrate the benefit of seed graph initialization in reducing computation complexity, we plotted the energy
Fig. 6. Computing seed subgraphs. On the right: a bear head as source graph \( G \). Top: some long curves selected as initial seeds. Middle: candidate sets \( M_v \) for each seed. "bottom": a best candidate by growing and merging the seed graphs.

convergence curves in the computational process in Fig. 7 for two examples.

Fig. 7.(a) plots two energy minimization curves for the image pair in Fig. 1), and Fig. 7.(b) shows the curves for searching the bear head template in Fig. 6. The two solid curves denote energy descending with seed graph initialization, while the dashed curves are results without using seed graph.

Fig. 7. Speed comparison of inference with and without seed graphs. (a) The energy curves on both multi-layer matching (the image pair in Fig. 1). (b) The energy plots for a single-layer matching (the image and template in Fig. 6). The solid curve are energy plots with seed graph initialization, and the dashed curve are without seed graphs.
C. Bottom-up computation II: connected components

The second type of bottom-up computation is to narrow the partition space by forming connected components (CCP) as in SW-cut[1]. Vertices in a CCP is like those in a seed graph and should receive the same label in partition. But it is different in two aspects: (i) a CCP is formed probabilistically based on their similarity in appearance within the source graph, and (ii) it is assembled and disassembled on-line.

For vertices $V$ in a source graph, we form an adjacency graph $G_+ = <V, E_+>$ by connecting adjacent vertices. Note that the links $E$ in $G$ is only for vertex (image primitive) connectivity while $E'$ is broader.

For each vertex $v \in V$, we collect some bottom-up discriminative features (like color, oriented gradient)

$$F(v) = (F_1(v), F_2(v), ..., F_N(v)). \tag{32}$$

In this paper, we use a 12-bin normalized gray histogram $h(v)$, so $F(v) = h(v)$. For each edge $e = <s, t> \in E_+$, a binary random “bond” variable $u_e = \{0, 1\}$ is defined. Then a discriminative probability on the edge is calculated in the bottom-up stage (computed once off-line) to indicate how likely the two vertices belong to the same object, i.e. receiving the same label.

$$q_e = q(u_e = 1|F(s), F(t)) \propto \exp \left(-\frac{KL(h(s)||h(t)) + KL(h(t)||h(s))}{4T}\right) \tag{33}$$

where $KL(\cdot)$ is the Kullback-Leibler divergence between any two histograms and $T$ is a temperature factor. Lower temperature leads to larger CCP.

During the MCMC inference process, for each edge $e = <s, t>$ connecting two vertices at the same layer $l(s) = l(t)$, it is turned off with probability $1 - q_e$. The edges remaining “on” form some CCPs, which will share the same label in partition during the MCMC computation. Usually vertices in a CCP have similar appearance and thus most likely belong to the same
object. By flipping the labels of these vertices together make the MCMC computation much faster than flipping the label of one vertex at each time as the Gibbs sampler does.

**D. MCMC algorithm for simultaneous partitioning and matching**

Given the bottom-up computation, we solve the double labeling problem as illustrated in Fig. 2(b) where the dots on the two vertical lines denote the partitioning and matching candidate sets $L_v$ and $M_v$ with weights.

In each computing step, the algorithm assigns a partition label or a matching label to each node or seed graph and it simulates a Markov chain which visits a sequence of states $\{\Pi_K(t), \Psi(t)\}$ in the joint space over time $t$. We compute $\Phi(t)$ deterministically given $\Pi_K(t), \Psi(t)$ for computational simplicity.

Thus our sampling process iterates between two types of Markov chain Monte Carlo dynamics and infers $\Pi_G$ and $\Psi$ respectively. The basic goal is to realize a reversible jump between any two successive graph partition states $\Pi_G$ and $\Pi'_G$, and then perform graph nodes matching to update matching state from $\Psi$ to $\Psi'$. For both partitioning and matching, whether a new state is accepted is decided based on a Metropolis-Hastings [17] decision to guarantee the convergence of the inference algorithm.

**Dynamics I: graph matching by Gibbs sampling**

Given a current partition state $\Pi_K$, the algorithm searches all nodes $v$(including the composite nodes or subgraphs) in $G$. We compute the probability for each candidate $v' \in M_v$ including $\emptyset$, we compute the conditional probability

$$\Psi(v) = v' \sim p(v'|\Pi_k, \Psi_{\partial v}), \quad \forall v' \in M_v.$$  

where $\Psi_{\partial v}$ means matching in the neighborhood of $v$. 
To improve efficiency, the algorithm visits stronger seed graphs which have less candidates, and then visit nearby nodes to propagate the information. Note the topological editing are applied here when evaluating the match.

**Dynamics II. graph partition by SW-cuts**

Given the current match $\Psi = \{\Psi_i, i = 1, 2, ..., K\}$ and partition $\Pi_K = \{g_i, i = 0, 1, 2, ..., K\}$, we sample the partition $\Pi_K$ from a conditional posterior probability by the Swendsen-Wang cuts [1]. Each SW-cut iteration consists of two steps.

Step 1 is to generate a set of connected components (CCPs) within each of the current subgraphs $g_i$. This is done by turning off the adjacency edge $e \in E_+$ between vertices in each $g_i$ with probability $1 - q_e$. The vertices within each $g_i$ remain connected (through the adjacency edges in $E_+$, not the edges in $E_i$).

Step 2 is to choose a CCP probabilistically and flip its label. The vertices in the CCP have the same label in the current state, and the flipping step assigns the same label to them in the next state. This may have three possible consequences:

1) Splitting a CCP from one subgraph $g_i$, and merging it to another subgraph $g_j$ of different label. Thus the number of objects $K$ remains unchanged.

2) Merging a CCP to an existing object and thus reducing the number of layers. This occurs when the CCP is equivalent to an existing subgraph $g_i$.

3) Splitting a CCP to a new object $g_{K+1}$ and thus increasing the number of layers.

Fig. 8 shows an example of the SW-cut iteration for the running example in Fig. 1. For space limitation, we only demonstrate the portion of images and corresponding graphs in Fig. 8.(a). Fig. 8 (b) is the current state $A$, where a CCP (the head and shoulder of a person) is selected by “cutting” some adjacency edges, and is highlighted by the dashed window in (c). (d-f) are
Fig. 8. An illustration of SW-cut iteration. (a) A pair of images cropped from Fig. 1. (b) The current partition state $A$, where the CCP is the head and shoulder highlighted by a dashed frame in (c). (d) and (e) are the possible cases for the next state $B$, through split-and-merge and split operations respectively, and note that (f) is considered as the possible consequence of merge operation, only if the CCP is is equivalent to an existing subgraph.

three possible new states through split-and-merge, split, and merge respectively.

The move between two states $A$ and $B$ is a reversible jump following the Metropolis-Hastings method[17].

Let $Q(A \rightarrow B)$ and $Q(B \rightarrow A)$ are the proposal probability from $A$ to $B$ and from $B$ to $A$ respectively. The chosen $CCP$ is labeled as the $i$-th layer $CCP \subset g_i$ in state $A$ and the $j$-th
layer in state $B$.

\[ Q(B \rightarrow A) = q(CC\mathbb{P}|B)q(l(CC\mathbb{P}) = i|CC\mathbb{P}, B) \]  \hspace{1cm} (34)  

\[ Q(A \rightarrow B) = q(CC\mathbb{P}|A)q(l(CC\mathbb{P}) = j|CC\mathbb{P}, A) \]  \hspace{1cm} (35)  

$q(CC\mathbb{P}|B)$ and $q(CC\mathbb{P}|A)$ are the probabilities of choosing a CCP at states $B$ and $A$ respectively. $q(l(CC\mathbb{P}) = i|CC\mathbb{P}, B)$ and $q(l(CC\mathbb{P}) = j|CC\mathbb{P}, A)$ are the probabilities for assigning a new label to the CCP.

The acceptance probability for the move from $A$ to $B$ is

\[ \alpha(A \rightarrow B) = \min \left( 1, \frac{Q(B \rightarrow A)p(B|G, G')}{Q(A \rightarrow B)p(A|G, G')} \right) \]  \hspace{1cm} (36)  

where $p(A|G, G')$ and $p(B|G, G')$ are the posterior probability defined in Eqn. ?? For clarity, we do not unfold them in details. Following proof in [1], the probability ratio of generating connected component can be computed on “cuts”, as

\[ \frac{q(CC\mathbb{P}|B)}{q(CC\mathbb{P}|A)} = \frac{\sum_{e \in \text{Cut}(CC\mathbb{P}, V_i \setminus CC\mathbb{P})} (1 - q_e)}{\sum_{e \in \text{Cut}(CC\mathbb{P}, V_j \setminus CC\mathbb{P})} (1 - q_e)}. \]  \hspace{1cm} (37)  

The overall algorithm for layered graph matching framework is summarized in Fig. 9.

V. Experiments

We apply the algorithm to three tasks: (i) object detection and recognition in cartoon and real images, (ii) large motion (with rigid and non-rigid transforms) and object matching in both artificial and real images, and (iii) human articulate motion analysis between images from a video sequence. We compare the performance to the state-of-the-art methods. The data were provided by the Lotus Hill Annotation project [36].

**Example I: Object detection using templates.** The cartoon object detection experiments are shown in Fig. 6, Fig. 10, and the real target detection are shown in Fig. 11. The task is to find a
1) Given input images $I, I'$, compute the sketch graphs $G$ and $G'$.

2) Bottom-up computation I: selecting and growing seed graphs in $G$.

3) Bottom-up computation II: computing the edge probability $q_e$ on the adjacency graph $G_+$. 

4) Initialize the partition $\Pi_K$ and matching $\Psi$ for $G$.

5) Iteratively partitioning and matching
   a) Fixing the matching state $\Psi$, sample the partition $\Pi_K$.
      i) Generate a connected component through turning on/off the edges probabilistically.
      ii) Assign a partition label to the selected subgraph with an acceptance probability.
   b) Fixing the partition state $\Pi_K$, sample $\Psi$.
      i) Visit seed graphs first and sample corresponding candidates using Gibbs sampler.
      ii) Visit single nodes around seed graphs and sample their candidates.

6) Output matching and partitioning configuration $W$.

Fig. 9. Algorithm summary for layered graph matching framework

Fig. 10. Examples of object detection in cartoon images: finding a bear head template in a series of cartoon images. Each result shows: (left) an input image, (middle) a source graph $G$, and (right) the detected object (head at various pose) with graph editing.
Fig. 11. Examples of object detection in real images: given templates, finding the bicycle and car in 4 cluttered images (the left column). The templates and seed graphs matching are shown in middle column and the targets are localized by transformed templates in the right column.

given template $G'$ – the head of a bear in a series of cartoon images, or the bicycle and car instance of the real images. The images are transferred into a sketch graph $G$ with color and intensity discarded. The matching is based on the graphs alone. The results show successful matches with the unmatched part being backgrounds. This method tolerates geometric deformations and some structural differences. The matching time for each image is about 30-50 seconds on a PC. Note we also show the seed graphs matching results in the middle column in Fig. 11. In addition, our layered graph matching method was used as an independent module for top-down verification in
the object recognition framework [15], where the possible candidates and objects’ rough scale and location can be provided by some bottom-up methods.

---

**Example II: Layered matching in artificial images.** To evaluate the performance we test the algorithm on some artificially constructed image pairs. Fig. 12.(a) and (b) shows three pairs of images for large motion (opaque and transparent) with occlusion. Note that no photometric information is used here and the connected components is selected randomly when partition sampling. The results are shown in Fig. 12.(c), where the occluded line segments are recovered. For comparison, Fig. 12).(d) shows the single layer matching results produced by the state-of-the-art graph matching algorithm [8], and Fig. 12).(e) displays the shape context match [5] results. As one may expected, these one layer methods cannot handled such examples.

Fig. 13 shows another example with non-rigid motion. Due to the relative low complexity of
testing images, the computing cost is around around 30 40s for each image pair. The conclusion can be drawn that the proposed layered matching framework is capable of dealing with the multi-layer matching with topology changing, and even under the complex rigid transformation, which is difficult to handle by the traditional sampling scheme, like RANSAC or MLESAC.

**Example III: Layered matching in real images.** Figs. 1, 16, and 14 show three experiment results for simultaneous matching and partitioning in real images.

In Fig. 1, the cars have different appearance and are in slightly different poses occluded by a person. They are matched, and the occluded segments are recovered. The matching energy is plot against the iterations in Fig. 7(b).

In Fig.14, the people inside the picture switched positions with large camera motion, and the sketch graphs and layered matching result are presented respectively. No camera information is given in the experiment. For comparison, we show the SIFT match results in Fig. 15 (a). The algorithm extracts many interest points on textured surface (picture frame in the background)
Fig. 14. An example of layered graph matching and the comparison is presented in Fig. 15. The sketch graphs for the input image pair is presented in (a) and the layered matching result in (b). There are three matched layers \((g_1, g_2, g_3)\) between the images, which are denoted by three colors (blue, green, and yellow) and the unmatched sketches \((g_0)\) are highlighted in red. For clarity, only some of the matched contours between images were labeled with different colors to show layered match.

and matches them quite well. However, on the non-texture surface, it gives very few matches. We show corner detection with RANSAC match results in Fig. 15 (b). Most of the corners are matched correctly with motion estimation, but it fails in matching the cartoon areas (background and person).

**Example IV: Layered matching for articulate motion analysis.** We demonstrate another
Fig. 15. For comparison. (a) Matching results by SIFT descriptor for comparison. Dense matches are found around texture regions, but very few matches can be found around textureless areas. (b) Harris corner detection and robust estimation with RANSAC. Most of the corners are matched correctly with motion estimation, but curves and lines in the textureless areas are not matched well.

Fig. 16. Another example for simultaneous graph matching and partition. Two input images $I, I'$ (top) are automatically transformed into attribute graphs $G, G'$. Both $G$ and $G'$ are partitioned into 4 layers of subgraphs with layers 1-2-3 being the common objects matched between $I, I'$, and layer 0 the unmatched backgrounds. The dark line segments are edited portions.
Fig. 17. Articulate motion analysis according to layered matching two far frames in a video sequence. Two far frames extracted in a video sequence and corresponding attribute graphs are shown in (a) and (b) respectively, and 4 body parts are segmented with different colors (red, yellow, blue and green), due to different motion captured by layered matching, as illustrated in (c).

Fig. 18. One contour motion analysis example by C. Liu [13]. Two testing images (with sketch graphs) and motion estimation result are shown in (a) and (b) respectively.

application of articulate motion analysis in video sequence. Two far frames extracted in a video sequence and corresponding attribute graphs are shown in Fig. 17 (a) and Fig. 17 (b) respectively, and 4 body parts are segmented with different colors (red, yellow, blue and green), due to different motion captured by layered matching, as illustrated in Fig. 17 (c). For comparison, one example of contour motion analysis in [13] is also presented in Fig. 18.
VI. DISCUSSION

In this paper, we present an integrated algorithm for simultaneous graph partition and matching with structural editing. It is aimed at general situations: (1) rigid and non-rigid motion of objects in a pair of images, (2) Camera motion, (3) series of occlusion. We compare it to a number of graph matching algorithms in the literature. Despite the bottom-up pruning of candidate labels in both the partition and matching spaces, the solution space is still quite large and thus demands long computing time. We propose to study more efficient methods in two directions: (i) pursue hierarchic recognition and grouping and thus match a tree structured representation rather than flat subgraphs. Of course, the inferences of the hierarchic parse tree (or graph) takes extra time to infer and brings more uncertainty. (ii) Although we use the SW-cut which is known to run fast on graph partition alone. But for graph matching, multiple subgraphs in the source graph $G$ compete for subgraphs in $G'$. Thus we encounter stronger coupling unseen in the graph partition (segmentation) problem previously. Thus we need to design bigger composite Markov chain moves. That is, a single Markov chain step should integrate both partition and matching to resolve the coupling and competition between subgraphs.

ACKNOWLEDGEMENT

The authors would like to thank Ziqiang Liu for his previous work on the graph matching project at UCLA. The data used in this paper were provided by the Lotus Hill Annotation project [36].

REFERENCES


