Computer Vision and Image Understanding 178 (2019) 16–29

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Visual graph mining for graph matching

Quanshi Zhang [a,*], Xuan Song [b], Yu Yang [c,1], Haotian Ma [d,1], Ryosuke Shibasaki [b]

[a] Shanghai Jiao Tong University, China
[b] University of Tokyo, Japan
[c] University of California, Los Angeles, USA
[d] Southern University of Science and Technology, China

ARTICLE INFO

Communicated by O. Veksler

MSC:
41A05
41A10
65D05
65D17

ABSTRACT

In this study, we formulate the concept of "mining maximal-size frequent subgraphs" in the challenging domain of visual data (images and videos). In general, visual knowledge can usually be modeled as attributed relational graphs (ARGs) with local attributes representing local parts and pairwise attributes describing the spatial relationship between parts. Thus, from a practical perspective, such mining of maximal-size subgraphs can be regarded as the discovery of common objects from visual data without given annotations of object bounding boxes. From a theoretical perspective, in this study, we propose a generic definition of common subgraphs among ARGs. Many previous studies can be roughly considered as special cases of the definition. In our definition, we consider 1) variations of unary/pairwise attributes among different ARGs, 2) linkage conditions of different nodes, and 3) the learning of similarity metrics for each node. The generality of our subgraph pattern proposes great challenges to the graph-mining algorithm. We propose an approximate but efficient solution to the mining problem. We conduct five experiments to evaluate our method with different kinds of visual data, including videos and RGB/RGB-D images. These experiments demonstrate the generality of the proposed method.

1. Introduction

Graph mining is a classical field in data mining, which focuses on either mining common subgraphs from multiple graphs or mining frequent subgraphs from a single large graph. Pioneering techniques mainly mined subgraphs from graphs of tabular data, which contain distinct node and edge labels. However, for visual data collected from real-world situations, such as images and videos, we need to consider the fuzziness of the data. Given two visual subgraphs that share the same subgraph pattern, two corresponding nodes (or edges) in the two subgraphs may have a considerable variation in their labels or attributes. The attribute variation in a node (or an edge) corresponds to changes of local appearance (or global shape deformation). We define the data fuzziness as the variation of node/edge attributes, and the data fuzziness undermines the basis of conventional graph-mining approaches. Therefore, in this study, we comprehensively re-define subgraph patterns in a unified paradigm to overcome the data fuzziness. The new definition greatly enhances the flexibility and generality of the subgraph pattern, which presents new challenges to graph-mining methods.

Concept of subgraph patterns: Before the introduction of the theory, let us first consider a typical application, *i.e.* mining common objects from images without annotations of object bounding boxes. An image can be modeled as an attributed relational graph (ARG), as shown in Fig. 1. Each node in the ARG contains a number of high-dimensional unary attributes to describe different local features. Pairwise attributes on the edges measure different spatial relationships between object parts. In this case, the model for the common objects among the images corresponds to the subgraph pattern among the ARGs. In other words, "discovering frequent subgraph patterns among visual ARGs" can be regarded as an elegant solution to "mining and modeling objects with similar appearances and structures from visual data".

As shown in Fig. 2, unlike tabular data, visual data presents an intuitive problem, *i.e.* we should simultaneously consider object occlusions and intra-category variations in texture, rotation, scale, and pose among different objects. Such variations are formulated as attribute variations among the ARGs.

Compared to previous patterns in Zhang et al. (2014, 2016), we further consider the uncertainty of pattern similarity metrics, which is the biggest challenge in real-world situations. Different subgraph patterns usually have their own metrics to evaluate the similarity between subgraphs. First, we should discover the hidden dependency/linkage relations between nodes. The selective use of strong part dependencies (*e.g.* strong spatial relationships/edges between the head node and the body node) and neglect of weak linkages (*e.g.* weak spatial relationships/edges between forefeet and hind feet) would produce stable

* Correspondence to: John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China.
E-mail address: zqs1022@sjtu.edu.cn (Q. Zhang).

[1] Partially contributed to this paper, when they worked at the Shanghai Jiao Tong University as internship students.

https://doi.org/10.1016/j.cviu.2018.11.002
Received 16 August 2017; Received in revised form 27 October 2018; Accepted 2 November 2018
Available online 24 November 2018

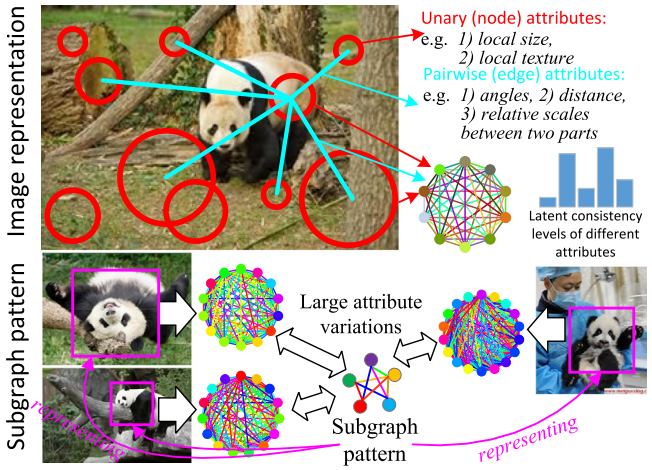

Fig. 1. Understanding the mVAP among ARGs. Node/edge colors denote different unary/pairwise attributes.

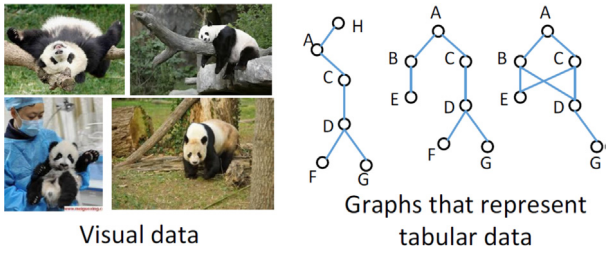

Fig. 2. Difference between tabular data and visual data.

mining performance. Second, we should incrementally discover latently effective attributes during the mining process. For example, the spatial relationship between parts may be the key factor to identify patterns of rigid objects, but such spatial relationships are not so significant when we measure the pattern similarity between dynamic animals.

Visual graph mining: The key to this research is the generality of our method, as this ensures its broad applicability. Therefore, in this study, we define a general visual attributed pattern (VAP) to comprehensively encode all the above visual challenges that are ubiquitous in different visual data.

In addition, we develop a generic method to efficiently mine such patterns. Given an initial graph template, we gradually modify this template to the maximal-size VAP (mVAP) by discovering new nodes, eliminating redundant nodes, adjusting node linkages, and training attributes and matching parameters.

The mVAP's comprehensive modeling of visual challenges raises graph-mining difficulties to a new level. First, as shown in Fig. 2, graph-mining approaches oriented to tabular data usually require the nodes or edges to have distinct labels. These methods simplify the matching between two graph nodes (or edges) as binary classification problems, *i.e.* considering the match between two nodes (or edges) is correct if their labels are the same; vice versa. Thus, these methods use such labels to enumerate new nodes for the common subgraph pattern. However, this node enumeration strategy is hampered by data fuzziness,[2] because people cannot identify correct matches based on the similarity of local

[2] This strategy uses local labels on nodes or edges to search new nodes for the pattern. However, in each specific visual ARG, both unary (node) attributes and pairwise (edge) attributes may be heavily biased. In addition, due to the existence of part (node) occlusion in visual ARGs, it is difficult to reduce the computational load by limiting the node enumeration within any single ARG.

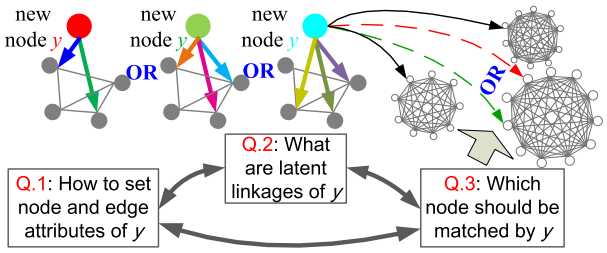

Fig. 3. Three-term chicken-and-egg problem in node discovery that raises the graph-mining challenge to a new level. We omit colors in unimportant nodes/edges for clarity.

attributes. Instead, people usually formulate the matching between visual graphs as a quadratic assignment problem (QAP), which needs a global optimization.

Second, simultaneously dealing with all visual variations is much more difficult than previous methods for mining knowledge from visual data, which selectively modeled certain variations in a specific kind of visual data and neglected other variations (see Section 2 for further discussion). For example, visual patterns in some pioneering studies, such as Zhang et al. (2014, 2016, 2013b), can be approximately understood as a special case of the mVAP.

The generality of the subgraph pattern increases difficulties of mining common subgraphs. The mVAP makes the discovery of a new pattern node y a **three-term** chicken-and-egg problem.[3] As shown in Fig. 3, the three interdependent terms include (1) the learning of the unary and pairwise attributes of y, (2) the discovery of y's latent linkages, and (3) the determination of the matching assignments mapping y to different ARGs. Moreover, these three mutually influenced terms are hidden in great visual variations. Thus, we must simultaneously estimate them, which is NP hard. Fortunately, we have demonstrated an approximate but efficient solution to this problem, which does not exhaustively enumerate pattern nodes and linkages. Our method iteratively grows and refines the subgraph pattern by minimizing energies of a sequence of Markov Random Fields (MRF). Our method is just an approximate solution to graph mining, because as proved in Szeliski et al. (2006), our optimization method (Kolmogorov, 2006) only provides an approximate solution to the energy minimization of MRFs.

Summary: The contributions of this study can be summarized as follows.

(1) We formulate a novel subgraph pattern, *i.e.* mVAP, in a generic form that has sufficient expressive power to comprehensively model object knowledge in fuzzy visual data.

(2) The mVAP is a generic definition of subgraph patterns among visual data. Some pioneering studies (Zhang et al., 2016, 2014, 2013b) can be regarded as special cases of the mVAP. The generality of the proposed mVAP is tested by different visual ARGs in different experiments.

(3) The pattern generality makes the mining of mVAPs a three-term chicken-and-egg problem. We propose an approximate yet efficient solution to the NP-hard mining problem.

The remainder of this paper is organized as follows. The next section discusses some related work. Section 3 defines the subgraph pattern for visual data, and Section 4 presents the graph mining algorithm. We design five experiments to evaluate the proposed method in Section 5. Finally, the overall study is summarized in Section 6.

[3] (1) Given pattern attributes and linkages, we can use graph matching techniques to compute the optimal matching assignments; (2) Given the matching assignments, the pattern attributes can be modified to better represent corresponding subgraphs in the ARGs; (3) Given the pattern attributes and matching assignments, strong linkages between pattern nodes can be determined. Please see the supplementary materials for details.

2. Related work

Graph mining: Ideas of graph mining were first proposed for tabular data, which are reviewed in Jiang et al. (2012). Conventional ideas of mining maximal subgraph patterns have been realized as maximal frequent subgraph (MFS) extraction (Thomas, 2010; Huan et al., 2004; Yan et al., 2005; Koyut urk et al., 2004) and the mining of maximal cliques (Wang et al., 2006; Zeng et al., 2006). MFS extraction approaches (Thomas, 2010; Huan et al., 2004; Jiang and Ngo, 2003) usually require the graphs to contain distinct node/edge labels or use local consistency to determine a set of node correspondence candidates between different graphs. Moreover, as shown in Fig. 2, the graphs must have distinguishing structures. Thus, these methods define MFSs using graph isomorphisms. They mainly enumerate nodes from different graphs to search common subgraphs with isomorphic structures and labels. The distinct labels are used to prune the search range and avoid the NP-hard computation in the worst case. Similarly, the mining of maximal cliques (Wang et al., 2006; Zeng et al., 2006; Xie et al., 2012; Liu and Yan, 2010) mainly extracts dense cliques that maintain geometric consistency. Subgraph patterns for ARGs have been defined (Tong et al., 2007; Silva et al., 2012), and the softness of clique patterns has been formulated (Quadrianto et al., 2012; Brunato et al., 2008).

However, the above methods are oriented to tabular data, and cannot be applied to fuzzy visual data[4] because of the requirement for node/edge labels or potential node correspondences. Visual ARGs may have considerable attribute variations, and cannot provide node correspondences in a local manner. In particular, recent studies (Zhang et al., 2018, 2017) further used graphs to represent middle-layer features in convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012; He et al., 2016).

Generally, node matching between ARGs is formulated as a quadratic assignment problem (QAP), which should be solved via a global optimization (as in (3), (8), and (9)). Thus, we must reformulate the whole theory on the basis of graph matching. In addition, as in Zhang et al. (2016), the visual graph mining usually requires a rough graph template to start the whole mining process.

Learning graph matching: Graph matching between a graph template and a number of ARGs is a typical problem in computer vision (GunPark et al., 2003; Park et al., 2005; Cho and Lee, 2012a; Cho et al., 2010a; Kim et al., 2012; Duchenne et al., 2011; Cho et al., 2014; Wang et al., 2014). Yan et al. (2013, 2014) proposes to simultaneously match multiple graphs. Given a graph template and a number of ARGs, methods for learning graph matching have been proposed to train parameters or refine the graph template for better matching performance. Most techniques (Cho et al., 2013; Caetano et al., 2009; Leordeanu et al., 2012; Torresani et al., 2008; Leordeanu and Hebert, 2008) take a supervised approach, *i.e.* they require the manual labeling of node correspondences between different ARGs. Leordeanu et al. (2012) proposed the first unsupervised method of learning graph matching, and Zhang et al. (2013b) further refined the template structure in an unsupervised fashion. Cho and Lee (2012b) proposed a similar idea that matched two ARGs and simultaneously extracted the most reliable edges between the two matched subgraphs. Essentially, these methods are not comparable with graph mining. They mainly train parameters or delete "bad" nodes from the graph template, rather than discovering new pattern nodes and recovering the prototype graphical patterns. Zhang et al. (2015a) used graph-matching results on RGB-D images to supervise the matching model for RGB images.

Visual mining: From the perspective of applications, there are numerous ways of mining objects from visual data, such as object discovery (Tuytelaars et al., 2010; Joulin et al., 2014; Cho et al., 2015; Kwak et al., 2015; Wei et al., 2017; Kantorov et al., 2016), co-segmentation (Joulin et al., 2012; Kim and Xing, 2012), edge model extraction (Lee and Grauman, 2009), the learning of structural patterns (Leordeanu et al., 2007; Brendel and Todorovic, 2011), and a number of techniques (Kim et al., 2008; Tan and Ngo, 2009; Yuan et al., 2012; Zhao and Yuan, 2010; Cho et al., 2010b; Parikh et al., 2009) related to maximal clique mining. However, these studies were mainly designed with some specific techniques oriented to their own applications. They have not summarized all the detailed problems in their applications into a unified paradigm for the general problem of visual graph mining, which consider both texture variations and structure deformation.

In contrast, we aim to build a generic unified paradigm for the subgraph pattern oriented to visual data, and we explore a theoretical solution to graph mining. We expect our mVAP to be a generalization of previous visual subgraph patterns and have a clear expressive power in describing objects, which elegantly encodes all variations in texture, rotation, scale, and pose. Pioneering studies on graph mining (Zhang et al., 2016, 2014, 2015b) simply assumed that common objects had no significant deformation between any pair of parts. Theoretically, these subgraph patterns can be approximately understood as a special case of the mVAP, *i.e.* the complete-graph pattern described by Definition 2(a). More specifically, the quadratic assignment problem (QAP) in Definition 2(a) can describe both the paradigm of minimizing matching energies in Zhang et al. (2014, 2016) and the paradigm of maximizing matching compatibilities in Zhang et al. (2013b). In this study, we propose a method to extract the mVAP as a more generic form of subgraph patterns that follow the paradigm of minimizing matching energies, which extends previous complete subgraph patterns to incomplete subgraph patterns. Thus, the definition of the mVAP ensures a broad applicability. For example, mVAPs are more suitable to describe non-rigid objects than previous subgraph patterns. Furthermore, the mVAP can encode different kinds of visual fuzziness in a generic form. We develop a generic method to mine such mVAPs. The generality of our method is demonstrated in five experiments.

3. Maximal-size subgraph pattern

Definition 1 (*ARG*)**.** An ARG $\mathcal{G}$ is a three-element tuple $\mathcal{G} = (\mathcal{V}, \mathbf{F}_{\mathcal{V}}, \mathbf{F}_{\mathcal{E}})$, where $\mathcal{V}$ is the node set. Undirected edges connect each pair of nodes to form a complete graph. $\mathcal{G}$ contains N_P types of local attributes for each node and N_Q types of pairwise attributes for each edge. $\mathbf{F}_{\mathcal{V}} = \{\mathcal{F}_i^s | s \in \mathcal{V}, i = 1, 2, \dots, N_P\}$ and $\mathbf{F}_{\mathcal{E}} = \{\mathcal{F}_j^{st} | s, t \in \mathcal{V}, s \neq t, j = 1, 2, \dots, N_Q\}$ denote the local and pairwise attribute sets, respectively. Each attribute corresponds to a feature vector.

Notation: Similar to the above notation, the maximal frequent subgraph pattern among positive ARGs $\mathbf{G}^+ = \{\mathcal{G}_k^+ | k = 1, 2, \dots, N^+\}$ can be represented as a five-element tuple $G = (V, E, \mathbf{F}_V, \mathbf{F}_E, \mathbf{W})$, and allows this to be an incomplete graph. Edges $(s,t) \in E$ and $(t,s) \in E$ are regarded as two different directed edges in G. $\mathbf{W} = \{w_i^P | i = 1, 2, \dots, N_P\} \cup \{w_j^Q | j = 1, 2, \dots, N_Q\} \cup \{P_{none}, Q_{none}\}$ is the parameter set, where w_i^P and w_j^Q denote positive weights for local attributes $\{F_i^s | s \in V\}$ and pairwise attributes $\{F_i^{st} | (s,t) \in E\}$.

Theoretically, we use $x_s^{\mathcal{G}_k^+}$ to denote the node in a positive ARG $\mathcal{G}_k^+$ that is matched to an mVAP node $s \in V$, when we match the mVAP to $\mathcal{G}_k^+$. Similarly, for negative ARGs $\mathbf{G}^- = \{\mathcal{G}_l^- | l = 1, 2, \dots, N^-\}$ (used later to define the mVAP), $x_s^{\mathcal{G}_l^-}$ denotes s's corresponding node in a negative ARG $\mathcal{G}_l^-$ when we match the mVAP to $\mathcal{G}_l^-$.

To simplify the notation, we always use k and l to denote the index for a positive ARG and the index for a negative ARG. Therefore, we can simply use x_s^k and x_s^l to represent $x_s^{\mathcal{G}_k^+}$ and $x_s^{\mathcal{G}_l^-}$ in the following paragraphs without causing ambiguity. Similarly, variables ξ_k^+ and ξ_l^- in the definition of the mVAP also correspond to $\mathcal{G}_k^+$ and $\mathcal{G}_l^-$.

Thus, we can regard $\{x_s^k | s \in V\}$ as a set of labels that represent the node correspondences between pattern G and a positive ARG $\mathcal{G}_k^+$. Note

[4] In this study, we focus on general visual data with great fuzziness, rather than the simplest visual data used in Jiang and Ngo (2003) and Hong and Huang (2004).

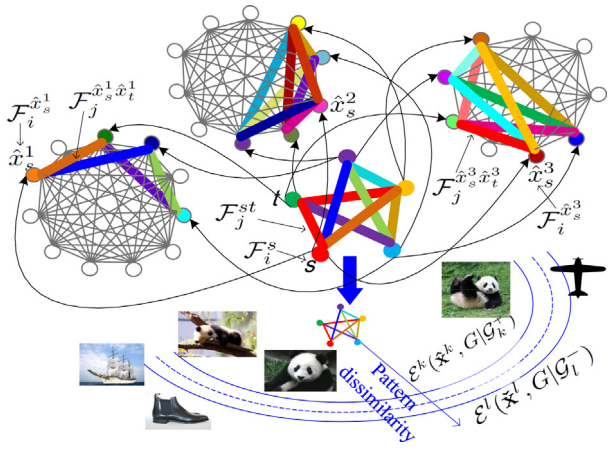

Fig. 4. Notation of the mVAP defined in Definition 2.

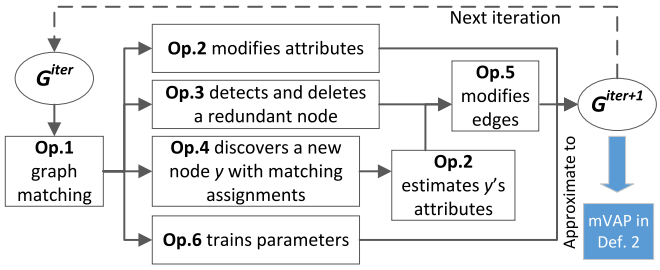

Fig. 5. Flowchart of an approximate solution to the NP-hard mining problem.

that **occlusion** in ARGs should be also considered. Thus, in addition to mapping s in G to $x_s^k \in \mathcal{V}_k^+$, s can also be mapped to a dummy node *none* (*i.e.* $x_s^k \in \mathcal{V}_k^+ \cup \{none\}$), when its corresponding node in $\mathcal{G}_k^+$ is **occluded**. Parameters $P_{none}, Q_{none} \in \mathbf{W}$ are constant penalties for mapping s to *none*.

Graph matching: we use square differences to define the attribute dissimilarity (or fuzziness) of matching s to x_s^k in ARG $\mathcal{G}_k^+$.

$$\mathcal{E}_s^k(x_s^k, G|\mathcal{G}_k^+) = P_s(x_s^k, G|\mathcal{G}_k^+) + \sum_{(s,t)\in E_s} Q_{st}(x_s^k, x_t^k, G|\mathcal{G}_k^+)$$

$$P_s(x_s^k, G|\mathcal{G}_k^+) = \begin{cases} \sum_{i=1}^{N_P} w_i^P \|\mathcal{F}_i^s - \mathcal{F}_i^{x_s^k}\|^2, & x_s^k \in \mathcal{V}_k^+ \\ P_{none}, & x_s^k = none \end{cases}$$

$$Q_{st}(x_s^k, x_t^k, G|\mathcal{G}_k^+) = \begin{cases} \sum_{j=1}^{N_Q} w_j^Q \|\mathcal{F}_j^{st} - \mathcal{F}_j^{x_s^k x_t^k}\|^2/|E_s|, & \\ & x_s^k \neq x_t^k \in \mathcal{V}_k^+ \\ +\infty, & x_s^k = x_t^k \in \mathcal{V}_k^+ \\ Q_{none}/|E_s|, & x_s^k \text{ or } x_t^k = none \end{cases} \quad (1)$$

where functions P_s and Q_{st} measure the difference in local and pairwise attributes. Infinite penalties are used to avoid many-to-one node assignments. $E_s = \{(s,t)|t \in V\} \subset E$ denotes the set of outgoing edges of s. $\mathbf{x}_s = \{x_s^k|k = 1, 2, \ldots, N^+\}$ denotes matching assignments of s. The above equation is a prototype formulation of graph matching for various subgraph patterns.

Definition 2 (*mVAP*). As shown in Fig. 4, given a set of positive ARGs $\mathbf{G}^+ = \{\mathcal{G}_k^+|k = 1, 2, \ldots, N^+\}$, a set of negative ARGs $\mathbf{G}^- = \{\mathcal{G}_l^-|l = 1, 2, \ldots, N^-\}$, the minimum degree d, and a threshold τ, $G = (V, E, \mathbf{F}_V, \mathbf{F}_E, \mathbf{W})$ is a mVAP, if and only if:
(a) $(V, \{x_s^k\}, \mathbf{F}_V, \mathbf{F}_E) = \text{argmin}_{V,\{x_s^k\},\mathbf{F}_V,\mathbf{F}_E} \sum_{s\in V}[\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+) - \tau]$, where $\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+) = \text{mean}_k \mathcal{E}_s^k(x_s^k, G|\mathcal{G}_k^+)$;
(b) $\forall s \in V, E_s = \text{argmax}_{E_s:|E_s|\geq\min\{d,|V|-1\},\mathcal{E}_s(\mathbf{x}_s,G|\mathbf{G}^+)<\tau}|E_s|$;
(c) $\mathbf{W} = \underset{\mathbf{W}}{\text{argmin}}\|\mathbf{w}\|^2 + \frac{C}{N^+}\sum_k \xi_k^+ + \frac{C}{N^-}\sum_l \xi_l^-$,

$$\forall k, -[\mathcal{E}^k(\mathbf{x}^k, G|\mathcal{G}_k^+) + b] \geq 1 - \xi_k^+,$$
$$\forall l, \mathcal{E}^l(\mathbf{x}^l, G|\mathcal{G}_l^-) + b \geq 1 - \xi_l^-,$$

where $\mathbf{x}^l \leftarrow \text{argmin}_{\mathbf{x}^l}\mathcal{E}^l(\mathbf{x}^l, G|\mathcal{G}_l^-)$, $\mathcal{E}^k(\mathbf{x}^k, G|\mathcal{G}_k^+) = \sum_{s\in V}\mathcal{E}_s^k(x_s^k, G|\mathcal{G}_k^+)$.

Item (a) presents a basic principle for a subgraph pattern among positive ARGs. It estimates the best node correspondences $\{\mathbf{x}_s\}$ and attributes $(\mathbf{F}_V, \mathbf{F}_E)$ for G that minimize the overall pattern fuzziness. In other words, this item requires the pattern and its corresponding subgraphs in different ARGs to have similar attributes, where the average attribute dissimilarity between each node s and its corresponding nodes is given by $\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+)$. The attributes and node correspondences are estimated via global optimization, considering large intra-category attribute variations. In addition, to ensure low fuzziness in pattern G, we use a threshold τ to limit $\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+)$. Note that this also maximizes the pattern size $|V|$, as long as the fuzziness $\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+)$ for each node s is less than τ.

Item (b) mines the node dependency in G. Given a certain fuzziness limit τ for G, this item encourages each pattern node s to have as many outgoing edges as possible. In addition, a minimum edge number d is used to ensure a certain pattern density. In general, for each node, this item learns a set of edges with most reliable dependency, *i.e.* the edges have similar corresponding pairwise attributes through different ARGs.

This item enables the mVAP to represent a non-rigid object more naturally, because a local part in a non-rigid object usually maintains stable spatial relationships with a few neighboring parts, rather than all parts.

Item (c) discovers latently effective metrics of pattern similarity for G. Different attributes in G correspond to different metrics of distance measurements, and the effective ones should have a good capacity for classifying positive and negative[5] subgraphs. $\mathcal{E}^k(\mathbf{x}^k, G|\mathcal{G}_k^+)$ measures the average attribute dissimilarity between G and the corresponding subgraph in $\mathcal{G}_k^+$, where $\mathbf{x}^k = \{x_s^k|s \in V\}$. Just like in the SVM, ξ_k^+ and ξ_l^- denote penalties for classification.

This item assigns higher weights for more reliable attributes, which boosts the stability of graph matching. Learning attribute weights are of great values in real applications, because not all attributes are equally reliable.

4. Graph mining

Given a set of positive ARGs $\mathbf{G}^+$, a set of negative ARGs $\mathbf{G}^-$, and an initial graph template G^0 that roughly[6] corresponds to a fragment of the target subgraph pattern, the goal of graph mining is to iteratively estimate all the parameters contained in the pattern and modify G^0 to the maximal-size VAP among these ARGs, $G^0 \rightarrow G^1 \rightarrow \ldots \rightarrow G^n =$ mVAP. We apply $n = 20$ iterations to mine mVAP in experiments. In this way, we gradually refine G^0 (describing both a fragment of the target object and some background) to a mVAP (a model for common objects among images).

Therefore, as shown in Fig. 5, we define six operations, which form an energy minimization framework, to modify the current pattern. We can demonstrate[7] that these operations present an approximate but efficient solution to graph mining.

Initialization: The initial graph template G^0 can be manually labeled as a complete graph. Even bad labeling is acceptable, *e.g.* using an object

[5] Negative ARGs do not contain pattern G and usually represent background.

[6] For example, we do not need to provide a rough localization of the target subgraph, but the initial graph template should contain at least two nodes and potentially correspond to at least a part of the target object. The mining process is sensitive to the initial template. If the initial template only describes noisy background, our method may not mine meaningful patterns.

[7] Please see the supplementary materials for proofs.

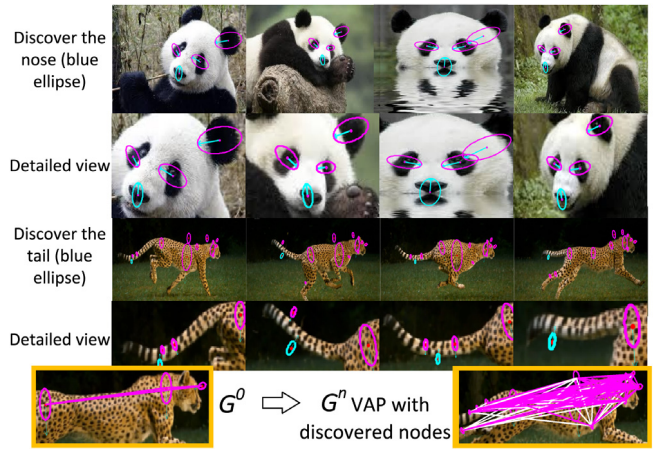

Fig. 6. Node discovery. The panda's nose and the cheetah's tail (cyan) are identified as the most probably missing nodes and added to G^{iter} (magenta) . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

fragment mixed with background to construct G^0. We then initialize matching parameters in $\mathbf{W}^0$ as $P^0_{none} = Q^0_{none} = +\infty$ and $w^{P,0}_{i=1,2,\ldots,N_P} = w^{Q,0}_{j=1,2,\ldots,N_Q} = 1/(N_P + N_Q)$.

Op. 1, graph matching: Node correspondences are estimated using Definition 2(a), in which the energy is written as

$$\text{Energy}^{(a)}(\mathbf{x}, G) = \sum_{s\in V}[\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+) - \tau], \text{where } \mathbf{x} = \bigcup_{s\in V}\mathbf{x}_s \tag{2}$$

We update node correspondences between G^{iter} and positive ARGs in a new iteration.

$$\begin{aligned}
&\forall k,\ \{x^k_s\}^{iter+1} \leftarrow \underset{\{x^k_s\}}{\text{argmin}}\ \text{Energy}^{(a)}(\mathbf{x}, G^{iter})\\
&= \underset{\{x^k_s\}}{\text{argmin}} \sum_{s\in V^{iter}} \mathcal{E}^k_s(x^k_s, G^{iter}|\mathcal{G}^+_k)\\
&= \underset{\{x^k_s\}}{\text{argmin}} \sum_{s\in V^{iter}} P_s(x^k_s, G^{iter}|\mathcal{G}^+_k)\\
&\quad + \sum_{(s,t)\in E^{iter}} Q_{st}(x^k_s, x^k_t, G^{iter}|\mathcal{G}^+_k)
\end{aligned} \tag{3}$$

This quadric assignment problem (QAP) is a typical case of graph matching, and can be solved by global optimization techniques, such as the TRW-S (Kolmogorov, 2006). Given $P_s(x^k_s, G^{iter}|\mathcal{G}^+_k)$ for all choices for (s, x^k_s) and $Q_{st}(x^k_s, x^k_t, G^{iter}|\mathcal{G}^+_k)$ for all choices for (s, t, x^k_s, x^k_t), the QAP problem can be automatically solved. In our experiments, the optimization for matches to most ARGs can be converged within 20 iterations with great stability, because graph matching on visual data are usually simple enough to reach a global minimum.

Op. 2, attribute estimation: Based on Definition 2(a), we update the attributes of G^{iter} in a new iteration.

$$(\mathbf{F}^{iter+1}_V, \mathbf{F}^{iter+1}_E) \leftarrow \underset{\mathbf{F}_V,\mathbf{F}_E}{\text{argmin}}\ \text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter}) \tag{4}$$

For clarity, we simply use x^k_s to denote the node correspondence in the $(iter+1)$ iteration. The above equation can be solved as

$$\mathcal{F}^{s,iter+1}_i = \underset{k:\delta(x^k_s)=1}{\text{mean}}\ \mathcal{F}^{x^k_s}_i, \quad \mathcal{F}^{st,iter+1}_j = \underset{k:\delta(x^k_s)\delta(x^k_t)=1}{\text{mean}}\ \mathcal{F}^{x^k_s x^k_t}_j \tag{5}$$

where $\delta(\cdot)$ indicates whether a node is matched to *none*. If $x^k_s = none$, then $\delta(x^k_s) = 0$; otherwise, 1.

The above equation computes the mean of attributes of all the matched graphs as attributes of the mVAP.

Op. 3, delete a redundant node: According to Definition 2(a), this operation selects and deletes the worst node $\hat{s}$ from G^{iter} to reduce the overall energy.

$$\begin{aligned}
&\Delta^{\text{del}}_s\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter}) = \tau - \mathcal{E}_s(\mathbf{x}^{iter+1}_s, G^{iter}|\mathbf{G}^+)\\
&\hat{s} \leftarrow \underset{s\in V^{iter}}{\text{argmin}} \Delta^{\text{del}}_s\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter})\\
&\text{if } \Delta^{\text{del}}_{\hat{s}}\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter}) < 0 \text{ then } V^{iter+1} \leftarrow V^{iter} \setminus \{\hat{s}\}
\end{aligned} \tag{6}$$

The energy change due to removing node s is denoted by $\Delta^{\text{del}}_s\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter})$, which is actually affected by the connection of s, E_s. Thus, we need to take linkage estimation in Definition 2(b) into account. Therefore, in this operation, we tentatively assign each node s with the linkages E_s that minimize $\mathcal{E}_s(\mathbf{x}^{iter+1}_s, G^{iter}|\mathbf{G}^+)$ to protect the good nodes from being deleted, *i.e.* we use a greedy strategy to select linkages with minimum energies $\text{argmin}_{E_s:|E_s|\geq d_1}\mathcal{E}_s(\mathbf{x}^{iter+1}_s, G^{iter}|\mathbf{G}^+) = \{(s,t) \in E^{iter} | 1 \leq \text{rank}_{t\in V^{iter}}\sum_k Q_{st}(x^k_s, x^k_t, G^{iter}|\mathcal{G}^+_k) \leq d_1\}$, where $d_1 = \min\{d, |V^{iter}| - 1\}$. Thus, based on $\{E_s\}$ and $\mathbf{x}$, we can directly compute $\hat{s}$ in the above equation.

Op. 4, node discovery: Node discovery is illustrated in Fig. 6. We formulate the energy change considering both Definition 2(a) and (b):

$$\begin{aligned}
&\Delta^{\text{add}}_y\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{iter}) = \min_{\{x^k_y\},\{\mathcal{F}^y_i\},\{\mathcal{F}^{yt}_j\},E_y} \mathcal{E}_y(\mathbf{x}_y, G^{new}|\mathbf{G}^+) - \tau\\
&\text{if } \Delta^{\text{add}}_y\text{Energy}^{(a)}(\mathbf{x}^{iter+1}, G^{new}) < 0 \text{ then } V^{iter+1} \leftarrow V^{iter} \bigcup \{y\}
\end{aligned} \tag{7}$$

where y denotes a missing node of G^{iter}, and G^{new} corresponds to a dummy enlarged pattern including y. In this operation, we need to simultaneously discover matching assignments $\{x^k_y\}$ and attributes $(\{\mathcal{F}^y_i\}, \{\mathcal{F}^{yt}_j\})$ of the new node y that represent the most reliable hidden pattern in ARGs. We have the following approximate solution to this problem. First, we use the following equation to estimate rough values of $\{x^k_y\}$, which can be regarded as a QAP of an MRF *w.r.t* $\{x^k_y\}$ and directly solved.

$$\begin{aligned}
&\underset{\{x^k_y\}}{\text{argmin}} \sum_{1\leq k,k'\leq N^+} \widetilde{M}_{kk'}(x^k_y, x^{k'}_y)\\
&\text{where } \widetilde{M}_{kk'}(x^k_y, x^{k'}_y) = \min_{|E_y|=d_2} \sum_{(y,t)\in E_y} m^{kk'}_t + \sum_{i=1}^{N_P} \frac{w^P_i \|\mathcal{F}^{x^k_y}_i - \mathcal{F}^{x^{k'}_y}_i\|^2}{2(N^+)^2}\\
&m^{kk'}_t = \frac{\delta(x^k_t)\delta(x^{k'}_t)\sum_{i=1}^{N_Q} w^Q_i \|\mathcal{F}^{x^k_y x^k_t}_i - \mathcal{F}^{x^{k'}_y x^{k'}_t}_i\|^2}{2|E_y|(N^+)\sum_j \delta(x^j_t)}\\
&\quad + \frac{(1 - \delta(x^k_t)\delta(x^{k'}_t))Q_{none}}{|E_y|(N^+)(N^+ + \sum_j \delta(x^j_t))}
\end{aligned} \tag{8}$$

Second, we use $\{x^k_y\}$ to estimate E_y as $\{(y,t)|t \in V^{iter}, 1 \leq \text{rank}_{t\in V^{iter}}\sum_k Q_{yt}(x^k_y, x^k_t, G^{iter}|\mathcal{G}^+_k) \leq d_2\}$, where $d_2 = \min\{d, |V|\}$. We avoid the uncertainty in node linkages of y, E_y, by tentatively applying the most conservative setting (accurate linkages will be estimated in **Op. 5**). Third, given E_y, we can further refine matching assignments $\{x^k_y\}$ as follows.

$$\begin{aligned}
&\underset{\{x^k_y\}}{\text{argmin}} \sum_{1\leq k,k'\leq N^+} M_{kk'}(x^k_y, x^{k'}_y)\\
&M_{kk'}(x^k_y, x^{k'}_y) = \frac{\sum_{i=1}^{N_P} w^P_i \|\mathcal{F}^{x^k_y}_i - \mathcal{F}^{x^{k'}_y}_i\|^2}{2(N^+)^2} + \sum_{t:(y,t)\in E_y} m^{kk'}_t
\end{aligned} \tag{9}$$

Finally, given the refined $\{x^k_y\}$, y's attributes $(\{\mathcal{F}^y_i\}, \{\mathcal{F}^{yt}_j\})$ can be further estimated using **Op. 2**.

We can understand **Op. 4** as follows. In this operation, our method selects a new node y, which matches nodes with similar local attribute and pairwise attributes through different ARGs. We use an MRF to ensure each pair of the matched nodes should have similar attributes.

Op. 5, fill edges: Based on Definition 2(b), we design the following procedure to update the edge set of each node s.

Initialization $E^{iter+1}_s = \emptyset$.
for $i = 1$ **to** $|V^{iter}| - 1$ **do**
  If $\mathcal{E}_s(\mathbf{x}_s, G|\mathbf{G}^+)|_{E^{iter+1}_s \bigcup\{(s,t_i)\}} < \tau$ **then** $E^{iter+1}_s \leftarrow E^{iter+1}_s \bigcup\{(s,t_i)\}$,
  else break; where $t_i = \underset{t\in V\setminus\{s,t_1,\ldots,t_{i-1}\}}{\text{argmin}} \sum_k Q_{st}(x^k_s, x^k_t, G^{iter}|\mathcal{G}^+_k)$.

end for

In each step, we use a greedy strategy to fill an edge that minimum the energy.

Op. 6, train matching parameters: Parameter training involved in Definition 2(b) can be solved using a linear SVM.

$$\begin{aligned}&\min_{\mathbf{w},\xi,b} \|\mathbf{w}\|^2 + \frac{C}{N^+}\sum_k \xi_k^+ + \frac{C}{N^-}\sum_l \xi_l^-,\\ &\text{s.t. } \forall k, -(\mathbf{w}\cdot\mathbf{a}_k^+ - b) \geq 1-\xi_k^+,\ \xi_k^+ \geq 0;\\ &\forall l, (\mathbf{w}\cdot\mathbf{a}_l^- - b) \geq 1-\xi_l^-,\ \xi_l^- \geq 0\end{aligned} \tag{10}$$

where $\mathbf{w} = [w_1^P, \ldots, w_{N_P}^P, w_1^Q, \ldots, w_{N_Q}^Q]^T$, $\mathbf{a}_k^+ = [a_1^{k,P}, \ldots, a_{N_P}^{k,P}, a_1^{k,Q}, \ldots, a_{N_Q}^{k,Q}]^T$, $a_i^{k,P} = \text{mean}_{s\in V:\delta(x_s^k)=1} \|\mathcal{F}_i^s - \mathcal{F}_i^{x_s^k}\|^2$, $a_j^{k,Q} = \text{mean}_{s\in V:\delta(x_s^k)=1}$ $\text{mean}_{(s,t)\in E_s:\delta(x_t^k)=1} \|\mathcal{F}_j^{st} - \mathcal{F}_j^{x_s^k x_t^k}\|^2$. Given $\mathbf{x}^l$,[8] we can compute $\mathbf{a}_l^-$ in the same way as $\mathbf{a}_k^+$. ξ_k^+ and ξ_l^- denote penalties for classification.

Computing $\mathbf{w}$ as above, we remove all the negative elements in $\mathbf{w}$ ($w_i \leftarrow \max\{w_i, 0\}$) and normalize $\mathbf{w}$ ($\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|_1$). We gradually update the attribute weights in each iteration, *i.e.* $\mathbf{w}^{iter+1} \leftarrow \lambda\mathbf{w} + (1-\lambda)\mathbf{w}^{iter}$, where $\lambda = 0.5$. Finally, given $\mathbf{w}^{iter+1}$, we estimate P_{none}^{iter+1} and Q_{none}^{iter+1} as

$$\begin{aligned}&P_{none}^{iter+1} \leftarrow \bar{P}^+ + \alpha(\bar{P}^- - \bar{P}^+),\ Q_{none}^{iter+1} \leftarrow \bar{Q}^+ + \alpha(\bar{Q}^- - \bar{Q}^+);\\ &\bar{P}^+ = \underset{1\leq k\leq N^+, s\in V^{iter}}{\text{mean}} P_s(x_s^k, G^{iter}|\mathcal{G}_k^+),\\ &\bar{Q}^+ = \underset{1\leq k\leq N^+, s\in V^{iter}}{\text{mean}} \sum_{(s,t)\in E_s} Q_{st}(x_s^k, x_t^k, G^{iter}|\mathcal{G}_k^+)\end{aligned} \tag{11}$$

where w_j^Q, w_i^P are elements in $\mathbf{w}$; $\bar{P}^+$, $\bar{Q}^+$ denote the average unary and pairwise matching penalties in positive ARGs, respectively; $\bar{P}^-$, $\bar{Q}^-$ for negative ARGs are defined in the same way as $\bar{P}^+$, $\bar{Q}^+$; and $\alpha > 0$. We simply set $\alpha = 1.0$. To demonstrate the robustness to parameter settings, we applied $\alpha = 1.0$ uniformly to all experiments based on all types of ARGs.

5. Experiments

The proposed visual graph mining provides a generic solution to the discovery of common patterns from cluttered visual data, where the target objects are randomly placed. Therefore, we design five experiments and test the broad applicability of our method by applying it to four types of visual data, including RGB-D and RGB indoor scenes, web images, and videos.

In order to demonstrate the generality of our method, we design four kinds of ARGs to represent the four types of visual data. In the first two experiments, we apply our method to cluttered indoor RGB-D and RGB images, respectively. We use two types of ARGs, which use edge segments as graph nodes, to represent the indoor objects with clear edges. Then, in Experiments 3 and 4, we apply our method to more general images, *i.e.* ubiquitous web images that can be collected from the internet and those in the challenging Pascal VOC dataset, respectively. Consequently, we design another two types of ARGs for image representation. One type of ARGs use SIFT feature points as graph nodes, and the other type of ARGs choose the automatically extracted middle-level patches as nodes. Finally, we further test the performance of mining deformable models from videos in Experiment 5.

Note that the same visual data can be represented using different ARGs with different features/attributes, although in our experiments, we use different ARGs for different visual data. People can change the design of features/attributes, *e.g.* the ARG used in Experiments 3 and 5 can be also applied to object discovery in Experiment 2. People can design their own ARGs for data representation, *e.g.* extracting regions from feature maps of convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012; He et al., 2016) as local attributes.

[8] We tentatively set P_{none} and Q_{none} to be infinite to avoid matching to *none* in the calculation of $\mathbf{x}^l$.

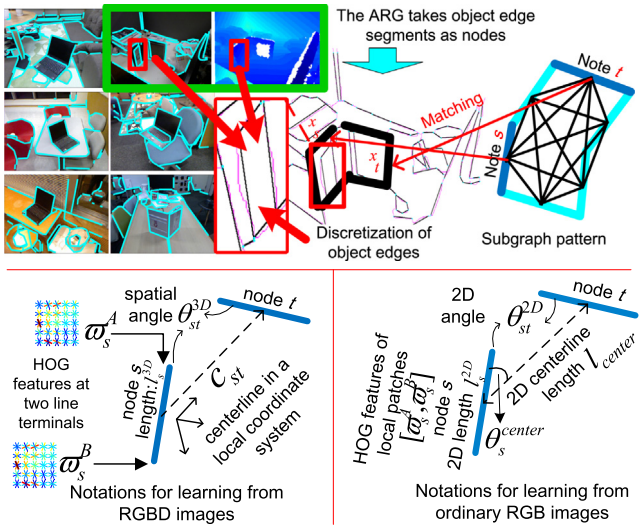

Fig. 7. Notation for edge-based ARGs that are used in Experiments 1 and 2.

We design a total of seven competing methods to enable a comprehensive comparison. They include typical image/graph matching approaches, unsupervised methods for learning graph matching, and the only pioneering method in the scope of visual graph mining (Zhang et al., 2016), which consider both the expressive power of graph mining and the challenges of visual mining.

5.1. Experiment 1: mining edge-based models from cluttered indoor RGB-D images

Dataset: The Kinect RGB-D image dataset proposed in Zhang (2016) is the benchmark RGB-D image dataset for testing graph matching and mining,[9] and has been used in Zhang et al. (2014, 2013b). This dataset consists of RGB-D images containing about 1200 objects. These RGB-D images are collected in different environments, indoors and outdoors. The RGB-D image is in the size of 640 × 480. Each RGB-D image produces two ARGs to describe its appearance on the RGB-D channels and the RGB channels. On average, each ARG contains 60.9 nodes. In this study, we use six categories, *i.e. notebook PC*, *drink box*, *basket*, *bucket*, *sprayer*, and *dustpan*, which contain sufficient objects for both training and testing. All the target objects are randomly placed in cluttered indoor scenes. Zhang (2016) provides a revised version of initial graph templates for matching.

Edge-based ARGs for RGB-D images: We use edge-based ARGs, which were widely applied in Zhang et al. (2013a,b, 2014), to represent RGB-D objects in an indoor environment. The ARGs are designed as follows. First, object edges are extracted from the images using the technique of Arbelaez et al. (2011), which uses only the RGB channels in the RGB-D images. Then, continuous edges are divided into line segments, and we take these line segments as graph nodes of the ARGs, as illustrated in Fig. 7. Please see Zhang et al. (2013a) for technical details of edge segmentation. Each pair of graph nodes is connected to construct a complete graph.

A total of two local attributes ($N_P = 2$) and three pairwise attributes ($N_Q = 3$) are designed to achieve robustness to rotation and scale variations in graph matching. As the first unary attribute, HoG features (Dalal and Triggs, 2005) are extracted from two patches at the line terminals of each node s. The HoG feature contains 5 × 5 cells, each covering half of

[9] This is one of the largest RGB-D object datasets. This dataset was designed for the evaluation of graph matching and graph mining, as it contains multiple kinds of intra-category variation, such as variations in texture, rotation, scale, and structure deformation.

its neighbors. In each cell, gradients are computed using four orientation bins from 0° to 180°. As the locally collected patch does not suffer much from illumination changes, all the cells are normalized within a single block. The second unary attribute describes the spatial length of the line segment of each node s (l_s^{3D}). The attribute is defined as the logarithm of the line length $\mathcal{F}_2^s = \log l_s^{3D}$. The first pairwise attribute, which is given by $\mathcal{F}_1^{st} = \theta_{st}^{3D}$, represents the spatial angle between each pair of line segments s and t. For each edge (s,t), we define the line connecting the centers of the line segments s and t as the *centerline* of (s,t). This centerline can be regarded as the relative spatial translation between two nodes s and t. To construct rotation-robust ARGs, Zhang et al. (2013a) defined a local 3D coordinate system for each centerline that is independent of the global object rotation to measure the translation. Let $\mathbf{c}_{st}$ denote the translation between nodes s and t. The second and third pairwise attributes represent the length and local orientation of the translation, and are written as $\mathcal{F}_2^{st} = \|\mathbf{c}_{st}\|$ and $\mathcal{F}_3^{st} = \mathbf{c}_{st}/\|\mathbf{c}_{st}\|$, respectively.

Experimental settings: We test the graph-mining performance of the proposed method under different parameter settings of d and τ. We follow the same experimental settings as in Zhang et al. (2014), including the same labeling of the initial graph templates G^0 and the same leave-one-out cross validation process.

Given each parameter setting, we perform a series of cross validations. For each RGB-D image in a category, we take the target object within it as the initial graph template and start an individual mining process. We randomly select 2/3 and 1/3 of the remaining images in this category to construct the positive ARGs for training and testing, respectively. We use images in the other categories to construct negative ARGs. We randomly select the same number of negative ARGs as positive ARGs in terms of both training and testing. In this way, given each specific parameter setting, we obtain a set of mVAPs for a category. In Section 5.6, we propose a number of metrics for evaluation, and the overall performance of graph mining with certain parameters is evaluated by computing the average performance among all the mined mVAPs in all the categories.

5.2. Experiment 2: mining edge-based models from cluttered indoor RGB images

Experimental settings: The design of Experiment 2 is similar to that of Experiment 1. We apply the same dataset of Kinect RGB-D images, but use only the RGB channels in the RGB-D images. Initial graph templates for matching are also provided by Zhang et al. (2014). We set different parameter values to mine mVAPs, and the evaluation method follows the same cross-validation procedure as in Experiment 1. In particular, a new type of ARGs are used to represent objects in cluttered indoor RGB images.

Edge-based ARGs for RGB images: The ARGs use edge segments as graph nodes, just like those for RGB-D images. A total of one type of unary attribute ($n^U = 1$) and three types of pairwise attributes ($n^P = 3$) are designed. Please see Fig. 7 for notation.

The only unary attribute is defined as the HoG features at the two line terminals, as for RGB-D images. The first pairwise attribute between each pair of nodes s and t represents the 2D angle between their line segments, which is given by $\mathcal{F}_1^{st} = \theta_{st}^{2D}$. Then, the angles between the centerline of (s,t) and the two line segments are defined as the second pairwise attribute, denoted by $\mathcal{F}_2^{st} = [\theta_s^{center}, \theta_t^{center}]$, where θ_s^{center} represents the angle between the line of s and the centerline. The third pairwise attribute is given by $\mathcal{F}_3^{st} = \frac{1}{l_{center}}[l_s^{2D}, l_t^{2D}]$, where l_s^{2D} and l_{center} are the lengths of line segment s and the centerline, respectively.

5.3. Experiment 3: mining SIFT-based models from web images

Dataset: In this experiment, we apply our method to a more general visual data, *i.e.* web images. Ten keywords ("bag", "boot", "camera", "coca cola", "glasses", "hamster", "iphone", "panda", "sailboat", and "spider") are used to collect internet images for ten categories. For each category, we select the 200 top-ranked images to construct the ARGs. This dataset has been published in Zhang (2016) and used to test conventional graph mining (Zhang et al., 2014). In this dataset, a revised version of initial graph templates are also provided.

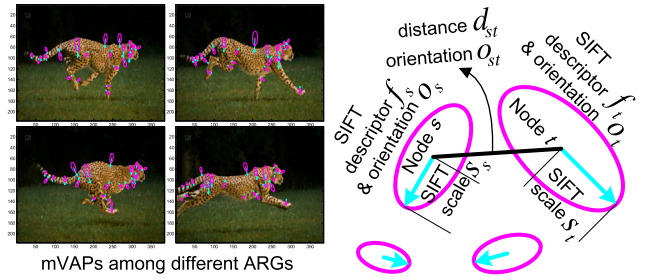

Fig. 8. Notation for the ARGs that take SIFT points as nodes and are used in Experiments 3 and 5.

Experimental settings: We apply different values of τ to test the mining performance. Note that, in ubiquitous web images, we cannot ensure there exists a single pattern that is able to describe target objects in all the images. Therefore, we set two criteria to control the image quality during the mining process. First, we require that during **Op. 1** in each iteration, more than 60% of the pattern nodes should be matched to the nodes in the positive ARGs, rather than *none*. In other words, if the matched subgraphs in positive ARGs represent less than 60% of the pattern size, these subgraphs will be considered invalid and removed. Second, in each iteration, no more than 20 web images should be used for model mining. We simply select the 20 top-ranked images from those collected, *i.e.* the 20 images (ARGs) with the lowest matching energies.

ARGs based on SIFT points: We use a new kind of ARG that selects points of interest from the SIFT features in each image as the graph nodes. The SIFT-based ARGs are designed for more general images, especially those without clear edges. We design two local features ($N_P = 2$) and five pairwise attributes ($N_Q = 5$) for this ARG type (please see Fig. 8 for notation). Let f_s, o_s, $\mathbf{p}_s$, and $\mathbf{s}_s$ be the 128-dimensional descriptor, orientation, position, and scale of node s's SIFT feature. We set the unary attributes for each node s as $\mathcal{F}_1^s = f_s$ and $\mathcal{F}_2^s = o_s$, and set six pairwise attributes for each edge (s,t) as $\mathcal{F}_1^{st} = angle(\mathbf{o}_s, \mathbf{o}_t)$, $\mathcal{F}_2^{st} = angle(\mathbf{o}_s, \mathbf{p}_s - \mathbf{p}_t)$, $\mathcal{F}_3^{st} = angle(\mathbf{o}_t, \mathbf{p}_s - \mathbf{p}_t)$, $\mathcal{F}_4^{st} = \ln(\mathbf{s}_s/\mathbf{s}_t)$, $\mathcal{F}_5^{st} = \ln(\sqrt{\mathbf{s}_s^2 + \mathbf{s}_t^2}/\|\mathbf{p}_s - \mathbf{p}_t\|)$, $\mathcal{F}_6^{st} = \mathbf{p}_s - \mathbf{p}_t/\|\mathbf{p}_s - \mathbf{p}_t\|$.

5.4. Experiment 4: mining models from Pascal VOC2007

Dataset: In this experiment, we mine the models from one of the most challenging visual datasets, Pascal VOC2007 (Everingham et al., 0000). Because the pattern mining process requires most positive images to contain the target subgraph pattern, we apply our method to the Pascal VOC2007 6 × 2 dataset (Deselaers et al., 2010), which contains *bus*, *motorbike*, *aeroplane*, *horse*, *boat*, and *bicycle* categories with *left* and *right* poses. It is because that we believe that a subgraph pattern usually describes objects with similar poses.

Experimental settings: Note that, because Pascal VOC images have significant intra-category appearance variations due to occlusions and texture changes, no specific pattern can successfully describe all training images. Therefore, we only mine the pattern corresponding to the top-10% of images, *i.e.* we determine the 10% of the training images with the lowest matching energies in **Op.** 1 in each iteration, and use these for the further mining process. Consequently, in Table 2, we evaluate the pattern's expressive power on the top-10%, 20%, … , 50%, and 100% of the Pascal VOC2007 testing images.

ARGs based on middle-level patches: As shown in Fig. 9, we construct ARGs using middle-level patch features (Singh et al., 2012) as nodes and taking their HoG features as the only unary attributes.

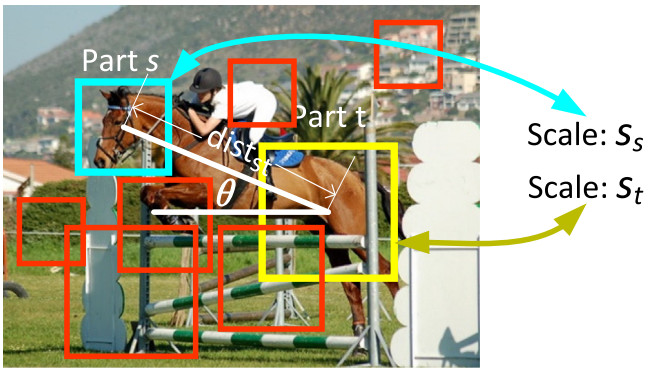

Fig. 9. Notation for the ARGs in Experiment 4 that take middle-level patches as nodes.

Let S_s and (u_s, v_s) denote the patch scale and 2D coordinates of node s. Three pairwise attributes ($N_Q = 3$) are used for each edge (s,t), *i.e.* $\mathcal{F}_1^{st} = \frac{1}{dist_{st}}[u_s - u_t, v_s - v_t]^T$, $\mathcal{F}_2^{st} = \log(S_s/S_t)$, $\mathcal{F}_3^{st} = [\log(S_s/dist_{st}), \log(S_t/dist_{st})]$, where $dist_{st} = \sqrt{(u_s-u_t)^2 + (v_s-v_t)^2}$.

5.5. Experiment 5: mining SIFT-based models from videos

In this experiment, we collect video sequences for a cheetah, swimming girls, and a frog from the internet. Each frame of these videos is formulated as an ARG. As in Experiment 3, the ARGs are constructed based on SIFT points. Thus, we use our method to mine mVAPs from these videos as the models for deformable objects. The initial graph templates only contain three nodes.

5.6. Competing methods

We comprehensively compare our method with seven competing methods, including three image/graph matching approaches, two methods of unsupervised learning for graph matching, one approach for refining the pattern structure, and one pioneering technique for visual graph mining. All the competing methods are designed considering the same scenario of "learning models from a number of ARGs and a single labeled object". To enable a fair comparison, all competing methods are provided with the same initial graph templates, as well as the same sets of training ARGs and testing ARGs.

First, image/graph matching approaches without any learning techniques are taken as the baseline. Generally speaking, there are two typical paradigms for image matching. One is the minimization of matching energy as in this study, and the other is the maximization of matching compatibility, where matching compatibility is usually defined as $\text{argmax}_{\mathbf{x}} C(\mathbf{x}) = \sum_{s,t} e^{-P_s(x_s) - P_t(x_t) - Q_{st}(x_s, x_t)}$. Thus, three competing methods, *i.e. MA*, *MS*, and *MT*, are designed to represent these two paradigms. *MA* method uses TRW-S (Kolmogorov, 2006) to minimize the matching energy in (1). *MS* and *MT* maximize the overall matching compatibility $C(\mathbf{x})$ (proposed in Leordeanu and Hebert, 2005; Leordeanu et al., 2012), in which $P_s(\cdot)$ and $Q_{st}(\cdot,\cdot)$ are defined using absolute attribute differences. *MS* and *MT* use spectral techniques (Leordeanu and Hebert, 2005) and TRW-S (Kolmogorov, 2006), respectively, to compute $\text{argmax}_{\mathbf{x}} C(\mathbf{x})$.

Second, we compare the proposed method with unsupervised approaches for learning graph matching, *i.e.* those that do not require people to label matching assignments. These techniques mainly learn matching parameters, but do not consider the learning of pattern structure, to achieve good matches between a graph template and a set of ARGs. Leordeanu et al. (2012) proposed the benchmark of unsupervised learning for graph matching. We design two competing methods to represent this technique, *i.e. LS* and *LT*. These two methods use (Leordeanu and Hebert, 2005; Kolmogorov, 2006), respectively, to solve the graph-matching optimization $\text{argmax}_{\mathbf{x}} C(\mathbf{x})$. They iteratively train the attribute weights, *i.e.* w_i^P and w_j^Q in the matching compatibility $C(\mathbf{x})$ mentioned above.

Third, we compare our method with the structural refinement method (Zhang et al., 2013b), denoted by *SR*. Actually, we can regard this method as the boundary between graph mining and unsupervised learning for graph matching. *SR* refines the pattern's structure by simultaneously deleting "bad" nodes, training matching parameters, and estimating attributes, but the key for graph mining, *i.e.* the discovery of new pattern nodes, is not involved. Note that when we apply different values of τ, our method will produce mVAPs with different graph sizes. Therefore, to enable a fair comparison, we require *SR* to modify the initial graph template to a pattern with the same size as the mVAP that is mined for a given τ.

Finally, we compare the proposed method with the only pioneering method that mines subgraph patterns, namely, maximal-size soft attributed patterns (*mSAP*), from visual data (Zhang et al., 2014). This method considers both the expressive power of graph mining and the challenges of visual mining. However, Zhang et al. (2014) requires the parameters $\mathbf{w}$, P_{none} and Q_{none} to be manually set. Thus, to enable a fair comparison, we apply the training of P_{none} and Q_{none} proposed in this study to Zhang et al. (2014), and initialize $\mathbf{w}$ as in our method.

5.7. Evaluation metrics, results, and analysis

5.7.1. Performance evaluation

Figs. 10, 11, 12, and 13 show the objects corresponding to the patterns mined in Experiments 1, 3, 4, and 5, respectively.

We evaluate our methods from two perspectives, *i.e.* the perspective of graph matching and that of graph mining, which are different from each other. In general, graph matching more focuses on the correctness of matches, which requires discriminative features for matching. Whereas, graph mining focuses on the correct growth of graph templates, which extracts the most common features among ARGs. We can regard the learning of graph matching as a discriminative learning problem, and parallel the graph mining to a generative learning problem. In addition, the matching accuracy is also sensitive to the size of a graph template. Small pattern fuzziness produces mVAPs with a small number of nodes, which lack enough information for reliable object detection, while large fuzziness makes the pattern contain unreliable nodes and decreases the matching performance. Without a proper template size, a correctly grown graph template may still exhibit a weak matching performance. Among the following metrics, the *average precision* and *average matching rate* measures the correctness of graph matching, and the *energy gap* and *pattern fuzziness* are suitable to evaluate the graph-mining quality.

Thus, we compare our mVAP with graph matching techniques using the evaluation metric for graph matching and compare with the mSAP (Zhang et al., 2014) using the evaluation metric for graph mining for a fair comparison.

Energy gap between positive and negative matches: We can use the mined patterns to match the target objects in previously unseen positive ARGs and negative ARGs. Therefore, we can regard the ratio of the average energy of the positive matches to that of the negative matches as a metric to evaluate the distinguishing ability of the pattern. This is similar to the "eigengap" for evaluating the spectral graph matching (Leordeanu and Hebert, 2005; Leordeanu et al., 2012). Patterns with low positive–negative energy ratios usually have a stable matching performance.

In Experiment 1, we produce different sets of PC patterns with different sizes by applying different values of τ via a series of cross-validations. Fig. 14(top) shows how the average energy ratio changes with the average pattern size among different sets of *notebook PC* patterns. Fig. 14(bottom) illustrates the performance using different values of τ. The mVAPs exhibit lower energy ratios than the other competing methods. We compare the mSAPs produced by Zhang et al.

Fig. 10. Object matching using mVAPs mined from RGB-D images (*i.e.* ARGs taking edges as nodes) in Experiment 1. Magenta/white edges denote directed edges with single/double orientation(s).

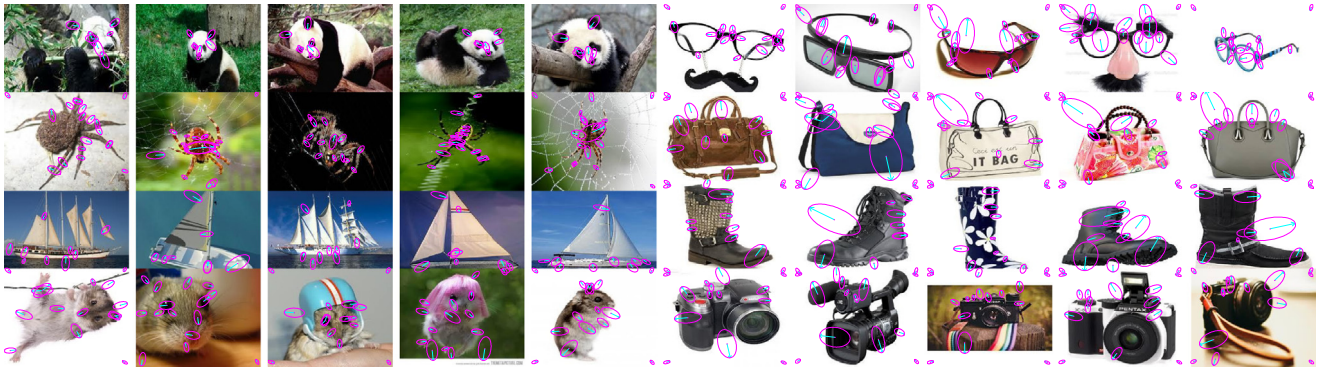

Fig. 11. Object matching using mVAPs mined from web images (*i.e.* ARGs taking SIFT points as nodes) in Experiment 3. Considering copyright reasons, results for the "iphone" and "coca cola" categories are not shown.

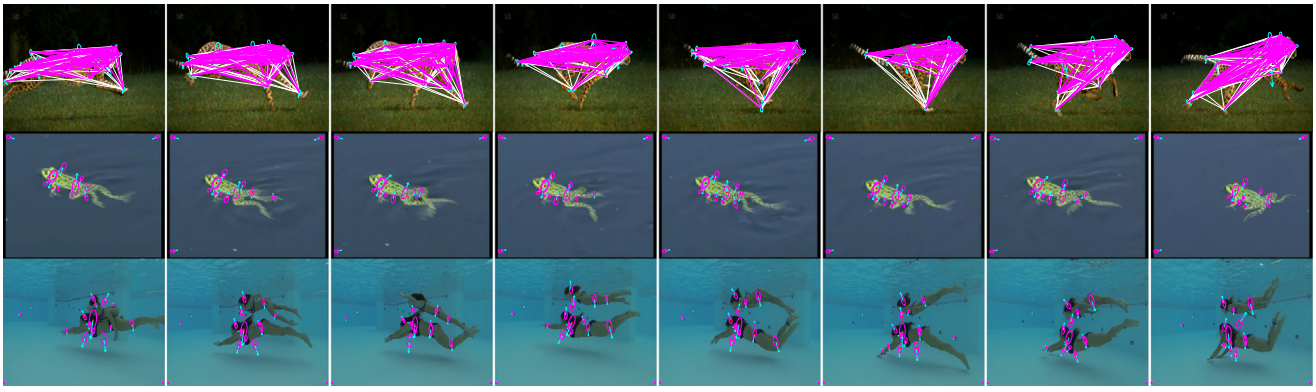

Fig. 12. Object matching using mVAPs mined from video frames (*i.e.* ARGs taking SIFT points as nodes) in Experiment 5. In the first row, magenta/white edges denote directed edges with single/double orientation(s). mVAPs fail to encode nodes on legs of the cheetah and the frog, because shape deformation of cheetah legs and frog legs is extremely large, which makes the fuzziness of leg nodes exceed the threshold τ.

(2014) (blue dashed lines) with the mVAPs mined with $d = +\infty$ (blue solid lines), because both of these patterns are complete graphs. Our mVAPs outperform the mSAPs.

Average precision (AP): We now test the object detection performance of the mined pattern. As mentioned above, we match the patterns to a set of previously unseen positive and negative ARGs.

Table 1
Comparison of the APs of patterns mined from web images. The mVAPs are mined by setting $d = 2$.

τ	AP										
	Bag mSAP, Our	Boot mSAP, Our	Camera mSAP, Our	Coca cola mSAP, Our	Glasses mSAP, Our	Hamster mSAP, Our	iphone mSAP, Our	Panda mSAP, Our	Sailboat mSAP, Our	Spider mSAP, Our	Average mSAP, Our
1/32	72.4 **84.8**	71.4 **93.7**	79.0 **90.1**	65.2 **66.9**	**73.0** 69.2	50.7 **84.8**	81.9 **92.9**	90.5 **93.8**	**69.4** 68.3	80.1 **94.5**	73.3 **83.9**
2/32	**90.9** 84.9	**76.4** 68.0	**85.5** 79.1	80.2 **88.0**	**69.6** 69.3	55.5 **82.4**	**76.7** 68.4	**94.8** 94.1	**67.6** 66.6	57.2 **74.3**	75.4 **77.5**
3/32	89.6 **92.7**	76.9 **80.0**	83.9 **84.2**	67.6 **82.2**	77.9 **83.9**	45.8 **82.4**	**78.0** 68.4	**94.9** 94.3	**67.9** 66.6	82.2 **86.0**	76.5 **82.0**
4/32	90.5 **92.5**	80.9 **81.1**	86.9 **88.3**	**73.6** 66.9	84.6 **86.1**	**55.5** 54.2	**78.3** 67.6	**95.8** 95.7	**63.4** 49.1	82.1 **87.7**	**79.2** 76.9
5/32	90.5 **92.4**	76.8 **82.2**	87.0 **92.0**	78.1 **81.4**	82.0 **95.2**	**53.9** 52.5	73.7 **79.0**	94.5 **97.7**	62.5 **68.3**	82.7 **88.4**	78.2 **82.9**
6/32	88.0 **91.6**	78.5 **79.0**	88.6 **92.3**	73.5 **82.7**	82.3 **88.1**	54.6 **56.8**	76.4 **82.9**	93.5 **96.7**	63.5 **69.8**	83.1 **90.6**	78.2 **83.0**

Fig. 13. Object matching using mVAPs mined from Pascal VOC2007 images (*i.e.* ARGs taking middle-level patches as nodes) in Experiment 4.

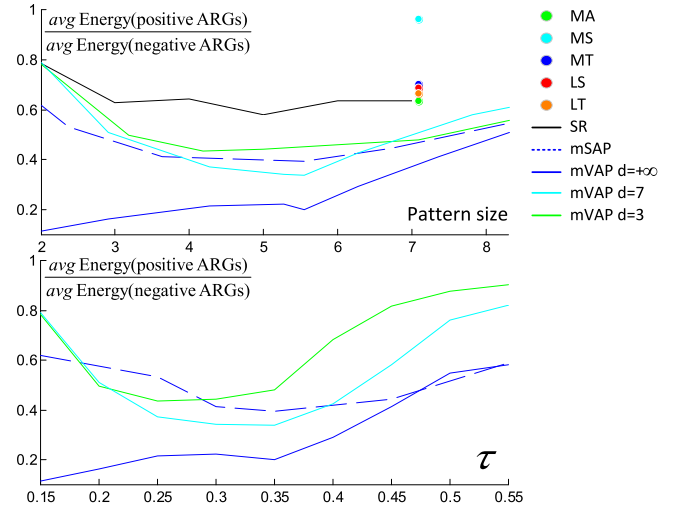

Fig. 14. Ratio of the energies of positive matches to those of negative matches. (top) We compare the energy ratios between different competing methods. (bottom) We show the ratio changes along with τ . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We use the simplest way of identifying the true and false detections: Given the matching assignments $\mathbf{x}^k$ for each ARG k and a threshold, if the value of $[-\mathcal{E}^k - \zeta \sum_s \delta(x_s^k)/|V|]$ (here, $\zeta = 10$) is greater than the threshold, we consider this to be a true detection; otherwise, it is a false detection. Hence, we can draw a precision–recall curve of object detection by choosing different thresholds. The average precision (AP) of the precision–recall curve is used as a metric to evaluate the graph matching performance. Note that because the mSAP is also defined based on its matching energy, the AP of an mSAP uses the same criterion to identify true detections.

Tables 1 and 2 shows that our mVAPs are more distinguishing than mSAPs. In Experiment 1, we use different values of τ to produce different sets of patterns with different average sizes. Fig. 16(top) shows how the average AP of the patterns changes with their size for different pattern sets. In Fig. 16(bottom), we plot the curves that illustrate the relationship between the average AP of the mVAPs and the value of τ. In Fig. 16(top), we compare our method with a total of six competing methods. It can be seen that our method exhibits superior performance to the other approaches. Fig. 16 gives a clear comparison between the mVAPs mined with different values of d. Compared to mining complete subgraph patterns (*i.e.* setting $d = +\infty$), the combination of mining pattern linkages increases the detection accuracy. This figure also demonstrates that the mining process may drift if the pattern is modified to contain too few or too many nodes. Small pattern fuzziness (τ) produces mVAPs with a small number of nodes, which lack enough

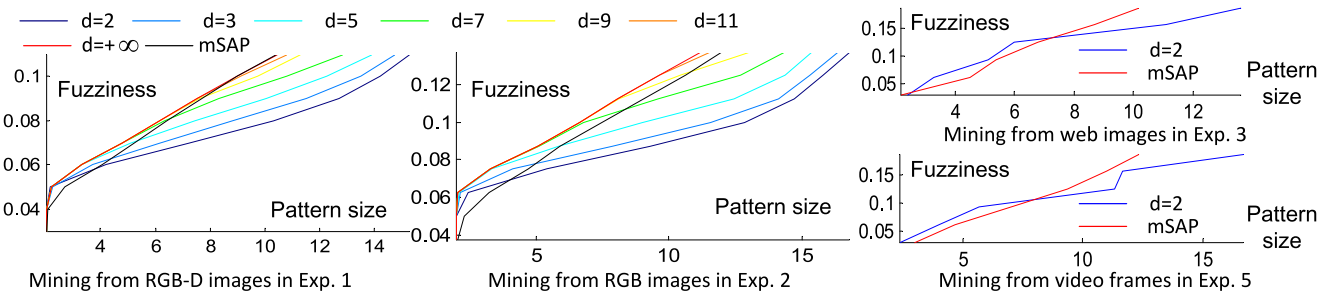

Fig. 15. Comparison of pattern fuzziness.

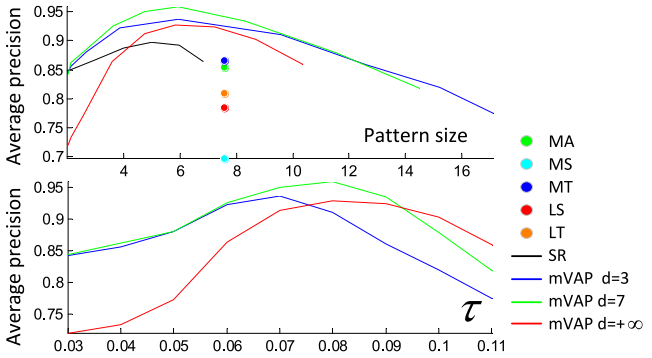

Fig. 16. APs of object detection using the mined patterns. (top) We compare the APs between different competing methods. (bottom) We show the AP changes along with τ.

Table 2
Comparison of average APs of the twelve patterns mined from the Pascal VOC2007 6×2 dataset. The mVAPs are extracted by setting $d = 10, \tau = 1.0$.

Testing images	Top 10%	Top 20%	Top 30%	Top 40%	Top 50%	Top 100%
mSAP	68.9	46.0	33.3	26.0	21.5	11.8
Our	**80.9**	**50.5**	**37.7**	**30.0**	**24.7**	**13.5**

information for reliable object detection, while large fuzziness makes the pattern contain unreliable nodes and decreases the performance.

Pattern fuzziness: In the experiments, we control the pattern fuzziness τ to obtain patterns with different sizes. In Fig. 15, we show the relationship between the pattern fuzziness and the average size of the patterns mined under this fuzziness. Because (Zhang et al., 2014) defined the fuzziness of the mSAPs in a similar way, we can compare the pattern fuzziness between the mSAPs and the mVAPs mined using different values of d. As in our method, Zhang et al. (2014) can mine mSAPs with different sizes given different fuzziness settings. Only patterns of similar size can be fairly compared.

Fig. 15 shows that in most cases, the mVAPs produced by our method are less fuzzy than the mSAPs mined by Zhang et al. (2014). More sparse mVAPs (with smaller values of d) usually have less fuzziness. Theoretically, if we ignore the training of matching parameters, the mining of mSAPs is equivalent to our mining of mVAPs when $d = +\infty$. Thus, the curves of mSAPs and mVAPs with $d = +\infty$ exhibit some similarities. Note that when the pattern size is very small (*e.g.* less than four nodes), the fuzziness of mVAPs is a bit higher than the fuzziness of mSAPs. It is because that the mSAP only uses positive ARGs for mining, but the mining of mVAPs requires both positive and negative ARGs. A small mVAP is usually not discriminative in classification of positive and negative ARGs, so in this case, the discriminative learning of mVAPs faces an over-fitting problem when we mine a small mVAP.

Average matching rate: We use the average matching rate (AMR) to evaluate the matching performance. AMR is widely used in the evaluation of learning graph matching (Zhang et al., 2013b; Leordeanu et al., 2012; Leordeanu and Hebert, 2008). The AMR is measured across all matching results produced by the extracted maximal SAP in the cross validation. Fig. 17 quantitatively compares average matching rates of mSAPs and mVAPs that are learned at different fuzziness levels τ in Experiment 2. Our mVAPs outperform mSAPs. Our method learns weights for attributes in mVAPs, so mVAPs usually exhibit higher matching rates. In addition, when we set a high fuzziness threshold τ, both mVAPs and mSAPs will contain nodes for unreliable contexts, which decreases the matching rate. In most cases, subgraph patterns with more than 10 nodes may contain unreliable nodes.

Discussions: In general, mVAPs exhibit higher APs than mSAPs and graph-matching baselines. mVAPs exhibit lower fuzziness than mSAPs with the same pattern size. mVAPs ($d = +\infty$) also exhibit larger energy gaps than mSAP with the same pattern size or with the same fuzziness. Usually, baselines of learning graph matching can only train matching weights of different attributes. However, in addition to the learning of attribute weights, graph-mining methods can also revise the structure of graph template by removing bad nodes and adding good nodes, thereby exhibiting better performance. In particular, compared to the mSAP, our method further learns linkages of the mVAP, which boosts the flexibility of the subgraph pattern. Thus, our mVAP usually outperform the mSAP in experiments.

5.7.2. Other analysis

Fig. 18 compares the attribute weights that are trained for a static iPhone in Experiment 3 and a running cheetah in Experiment 5. Static objects usually have higher weights on the pairwise relationship between object parts, whereas patterns for dynamic animals focus more on local attributes.

Latent pattern density: Parameter d in Definition 2 describes the minimum number of outgoing edges and is not a strong control of pattern density. Given a pattern fuzziness τ, the latent pattern density is automatically mined. Fig. 20 shows the average number of outgoing edges for each node mined using different parameters in Experiments 1 and 2. It demonstrates that the number of outgoing edges for each node is mainly controlled by the pattern's fuzziness level τ. If we set a large value for d, *e.g.* $d = +\infty$, then Fig. 15 indicates that we would obtain small patterns, which in return limits the pattern's edge number.

Computation time of learning/mining: Fig. 19 shows the average computational cost of mining each category model in Experiment 1. We implemented the algorithm using Matlab, and computed the running time using eight hyper threads on a machine with an Intel Xeon CPU X5560 @2.80 GHz. Larger values of τ usually produce larger patterns, and require more computational time.

We now briefly analyze the computational cost of the competing methods. The main computational task of these approaches is the energy minimization (or compatibility maximization) of graph matching during the learning/mining process. First, image/graph matching approaches, such as *MA*, *MS*, and *MT*, do not apply any learning techniques.

Second, unsupervised methods for learning graph matching (*LS* and *LT*) use M iterations to learn the matching parameters without changing the size of the graph template. In each iteration, they match the graph template to all N^+ positive ARGs. We use n^0 and n_k^+ to denote the node number of the initial graph template and the node number of the kth

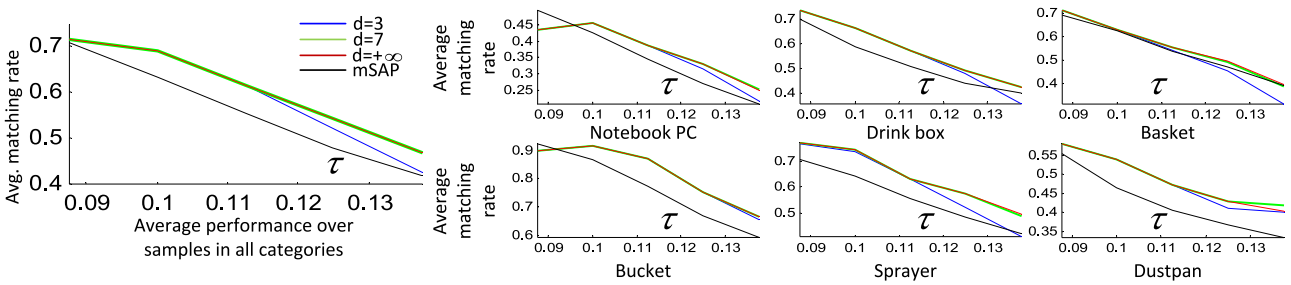

Fig. 17. Comparison of average matching rates of patterns mined in Experiment 2.

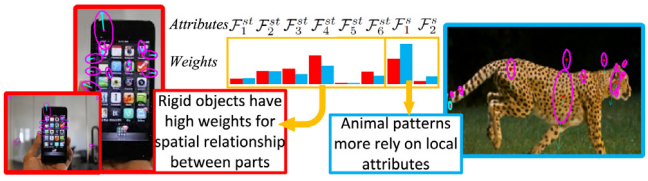

Fig. 18. Attribute weights mined for a rigid pattern in Experiment 3 and a animal pattern in Experiment 5.

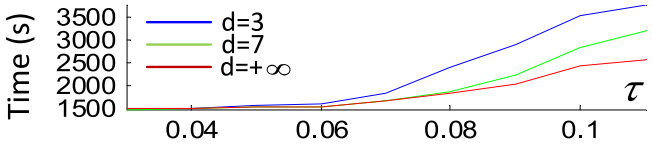

Fig. 19. Time costs in Experiment 1.

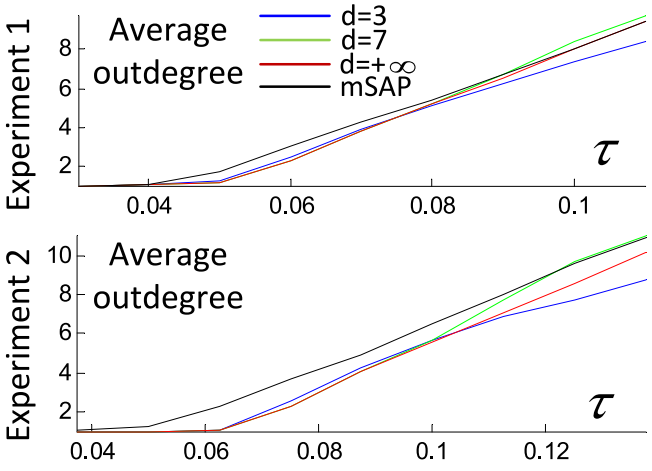

Fig. 20. Average number of latent linkages mined for each node. The linkage mining process is insensitive to d.

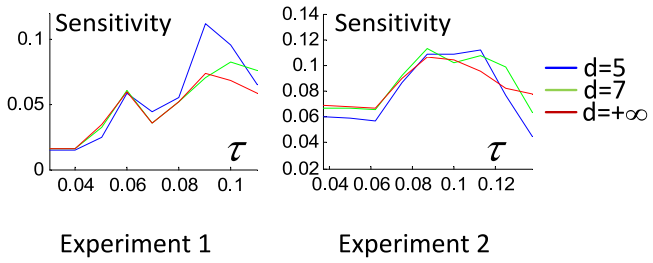

Fig. 21. mVAPs' sensitivity to initial graph templates.

positive ARG $\mathcal{G}_k^+$, respectively. The matching to $\mathcal{G}_k^+$ can be computed as a QAP that assigns each of the n^0 template nodes to one of n_k^+ labels.[10] Note that many techniques can be applied to the QAP of matching optimization, and each of them has its own accuracy and computational cost (please refer to Szeliski et al., 2006 for details). Thus, we simply use $c(n^0 \to n_k^+)$ to denote the computational cost of this QAP. Larger values of n^0 and n_k^+ will result in higher cost. Therefore, the computational cost of unsupervised learning can be formulated as $M \sum_k c(n^0 \to n_k^+)$.

Third, for the mining of mSAPs (Zhang et al., 2014, 2016), let $n^1, n^2, \ldots, n^M$ denote the node number of the pattern after $1, 2, \ldots, M$ iterations. Because this method only uses positive ARGs for training, the computational cost of graph matching is $\sum_{m=0}^{M-1} \sum_k c(n^m \to n_k^+)$. In addition, the computational cost of node discovery in each iteration m can be formulated as a QAP that assigns each of the N^+ positive ARGs with one of $\max_k\{n_k^+ - n^m\}$ labels (*i.e.* determining the node corresponding to the new node y in each positive ARG). Hence, its computational cost is $c(N^+ \to \max_k\{n_k^+ - n^m\})$. Thus, we can summarize the overall computational cost as $\sum_{m=0}^{M-1}[c(N^+ \to \max_k\{n_k^+ - n^m\}) + \sum_k c(n^m \to n_k^+)]$.

Fourth, let us focus on *SR*. Because *SR* only deletes bad nodes without adding new nodes, we assume that the pattern size is $\min\{n^1, n^0\}, \ldots, \min\{n^M, n^0\}$ after $1, \ldots, M$ iterations to enable a fair comparison. Consequently, its time cost can be written as $\sum_{m=0}^{M-1} \sum_k c(\min\{n^m, n^0\} \to n_k^+)$.

Finally, let us analyze the proposed method. In addition to the positive matches, our method matches the current pattern to all N^- negative ARGs in each iteration. Moreover, unlike the mining of mSAPs (Zhang et al., 2014), the QAP for node discovery must be solved twice to detect each new pattern node. Therefore, the overall computational cost of graph mining is $\sum_{m=0}^{M-1}[2c(N^+ \to \max_k\{n_k^+ - n^m\}) + \sum_k c(n^m \to n_k^+) + \sum_l c(n^m \to n_l^-)]$, where n_l^- denotes the node number of the negative ARG $\mathcal{G}_l^-$. In summary, compared to other methods of pattern learning/mining, our method requires additional computation for the matching to negative ARGs. The additional computation cost is linear with respect to the number of negative ARGs. Besides, the mining of node linkages doubles the computational cost of node discovery. Nevertheless, the proposed method has the same order of time complexity as Zhang et al. (2014).

Sensitivity to initial templates: We compute mVAPs' sensitivity to initial templates by measuring changes in the matching performance when we use initial templates of different sizes to learn mVAPs. As mentioned in Experiments 1 and 2, we use different initial graph templates G^0 of each category to learn different mVAPs for this category. Now, given certain parameters of τ and d, let us consider all mVAPs that are learned using all initial graph templates of all categories. We divide all mVAPs into seven groups, *i.e.* mVAPs that are learned using initial templates with five nodes, those corresponding to initial templates with six nodes, ..., those corresponding to initial templates with ten nodes, and other mVAPs. We compute average matching rates for the first six groups of mVAPs as $r_5, r_6, \ldots, r_10$, which correspond to initial graph templates with 5–10 nodes, respectively. Then, the sensitivity is reported as the standard deviation of $\{r_5, r_6, \ldots, r_10\}$. Fig. 21 shows the sensitivity of mVAPs that are learned with different values of τ and d.

[10] Here, matching choices of *none* are ignored.

When we set a low or a high value of τ, the mSAP is usually less sensitive to the size of the initial graph template.

6. Conclusions and discussions

In this paper, we have designed a new subgraph pattern for visual data with great generality, which presents great challenges to both fields of graph mining and computer vision. The high generality of subgraph patterns proposes great challenges for graph-mining algorithms. We have proposed an approximate but practical method to overcome great data fuzziness and comprehensively discover pattern nodes, mine node connections, identify significant attributes, and optimize attribute values and parameters. Our method has been tested on four kinds of ARGs in five experiments. Compared to the competing methods, subgraph patterns mined by our method exhibit better performance.

The core of this study is to propose a unified definition for visual subgraph patterns that encodes different kinds of visual variations, rather than a sophisticated approach for a specific visual task. Compared to previous subgraph patterns, the new subgraph pattern has more generality and flexibility, but the new definition makes graph mining a three-term chicken-and-egg problem. We propose an approximate but efficient solution to the graph-mining problem. In real applications, we can achieve a non-parametric mining process by selecting τ and d that maximize the AP of pattern-based detections on training images.

In addition, we can add some task-specific techniques to improve the performance. For example, we may use features extracted from feature maps of convolutional neural networks as local features/attributes to boost the performance. We can also learn root templates, extract multiple model components for a single category, and apply the non-linear SVM, in order to compete with supervised deformable part models (Girshick et al., 2011) and And-Or graph (Park et al., 2017). We could use segmentation techniques to produce G^0, thus achieving a fully unsupervised system, or design a hierarchical And-Or structure for the pattern to achieve more robustness to object detection. However, in this paper, we simply apply our method to ARGs with simple attributes to clarify the story of graph mining.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cviu.2018.11.002.

References

Arbelaez, P., Maire, M., Fowlkes, C., Malik, J., 2011. Contour detection and hierarchical image segmentation. In: PAMI 33. pp. 898–916.

Brendel, W., Todorovic, S., 2011. Learning spatiotemporal graphs of human activities. In: ICCV.

Brunato, M., Hoos, H.H., Battiti, R., 2008. On effectively finding maximal quasi-cliques in graphs. In: n the Learning and Intelligent Optimization Conference(LION), Vol. 5313. pp. 41–55.

Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J., 2009. Learning graph matching. In: PAMI.

Cho, M., Alahari, K., Ponce, J., 2013. Learning graphs to match. In: ICCV.

Cho, M., Kwak, S., Schmid, C., Ponce, J., 2015. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In: CVPR.

Cho, M., Lee, K.M., 2012a. Progressive graph matching: Making a move of graphs via probabilistic voting. In: CVPR.

Cho, M., Lee, K.M., 2012b. Progressive graph matching: Making a move of graphs via probabilistic voting. In: CVPR.

Cho, M., Shin, Y.M., Lee, K.M., 2010a. Unsupervised detection and segmentation of identical objects. In: CVPR.

Cho, M., Shin, Y.M., Lee, K.M., 2010b. Unsupervised detection and segmentation of identical objects. In: CVPR.

Cho, M., Sun, J., Duchenne, O., Ponce, J., 2014. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In: CVPR.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: CVPR.

Deselaers, T., Alexe, B., Ferrari, V., 2010. Localizing objects while learning their appearance. In: ECCV.

Duchenne, O., Joulin, A., Ponce, J., 2011. A graph-matching kernel for object categorization. In: ICCV.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 0000. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Girshick, R., Felzenszwalb, P., McAllester, D., 2011. Object detection with grammar models. In: NIPS.

GunPark, B., Lee, K.M., Lee, S.U., Lee, J.H., 2003. Recognition of partially occluded objects using probabilistic arg (attributed relational graph)-based matching. In: Computer Vision and Image Understanding (CVIU), Vol. 90. pp. 217–241.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: CVPR.

Hong, P., Huang, T., 2004. Spatial pattern discovery by learning a probabilistic parametric model from multiple attributed relational graphs. In: Discrete Applied Mathematics, Vol. 139. pp. 113–135.

Huan, J., Wang, W., Prins, J., Yang, J., 2004. Spin: Mining maximal frequent subgraphs from graph databases. In: ACM SIGKDD.

Jiang, C., Coenen, F., Zito, M., 2012. A survey of frequent subgraph mining algorithms. In: The Knowledge Engineering Review. pp. 1–31.

Jiang, H., Ngo, C.W., 2003. Image mining using inexact maximal common subgraph of multiple args. In: International conference on visual information system.

Joulin, A., Bach, F., Ponce, J., 2012. Multi-class cosegmentation. In: CVPR.

Joulin, A., Tang, K., Fei-Fei, L., 2014. Efficient image and video co-localization with frank-wolfe algorithm. In: ECCV.

Kantorov, V., Oquab, M., Cho, M., Laptev, I., 2016. Contextlocnet: Context-aware deep network models for weakly supervised localization. In: ECCV.

Kim, G., Faloutsos, C., Hebert, M., 2008. Unsupervised modeling of object categories using link analysis techniques. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–8.

Kim, K.I., Tompkin, J., Theobald, M., Kautz, J., Theobalt, C., 2012. Match graph construction for large image databases. In: ECCV.

Kim, G., Xing, E., 2012. On multiple foreground cosegmentation. In: CVPR.

Kolmogorov, V., 2006. Convergent tree-reweighted message passing for energy minimization. In: IEEE PAMI, Vol. 28. pp. 1568–1583.

Koyut urk, M., Grama, A., Szpankowski, W., 2004. An efficient algorithm for detecting frequent subgraphs in biological networks. J. Bioinform. 20, 200–207.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: NIPS.

Kwak, S., Cho, M., Laptev, I., Ponce, J., Schmid, C., 2015. Unsupervised object discovery and tracking in video collections. In: ICCV.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. In: Proceedings of the IEEE.

Lee, Y.J., Grauman, K., 2009. Shape discovery from unlabeled image collections. In: CVPR.

Leordeanu, M., Hebert, M., 2005. A spectral technique for correspondence problems using pairwise constraints. In: ICCV.

Leordeanu, M., Hebert, M., 2008. Smoothing-based optimization. In: CVPR.

Leordeanu, M., Hebert, M., Sukthankar, R., 2007. Beyond local appearance: category recognition from pairwise interactions of simple features. In: CVPR.

Leordeanu, M., Sukthankar, R., Hebert, M., 2012. Unsupervised learning for graph matching. In: IJCV 96.

Liu, H., Yan, S., 2010. Common visual pattern discovery via spatially coherent correspondences. In: CVPR.

Parikh, D., Zitnick, C., Chen, T., 2009. Unsupervised learning of hierarchical spatial structures in images. In: CVPR.

Park, B.G., Lee, K.M., Lee, S.U., 2005. Face recognition using face-arg matching. IEEE Trans. Pattern Anal. Mach. Intell. 27, 1982–1988.

Park, S., Nie, X., Zhu, S.C., 2017. Attributed and-or grammar for joint parsing of human pose, parts and attributes. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol. PP. p. 1.

Quadrianto, N., Chen, C., Lampert, C.H., 2012. The most persistent soft-clique in a set of sampled graphs. In: ICML.

Silva, A., Jr., W.M., Zaki, M.J., 2012. Mining attribute-structure correlated patterns in large attributed graphs. In: VLDB.

Singh, S., Gupta, A., Efros, A.A., 2012. Unsupervised discovery of mid-level discriminative patches. In: ECCV.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C., 2006. A comparative study of energy minimization methods for markov random fields. In: ECCV.

Tan, H.K., Ngo, C.W., 2009. Localized matching using earth mover's distance towards discovery of common patterns from small image samples. In: Image and Vision Computing, Vol. 27. pp. 1470–1483.

Thomas, L., 2010. Maximal Frequent Subgraph Mining. International Institute of Information Technology, Hyderabad, India.

Tong, H., Gallagher, B., Faloutsos, C., Eliassi-Rad, T., 2007. Fast best-effort pattern matching in large attributed graphs. In: KDD.

Torresani, L., Kolmogorov, V., Rother, C., 2008. Feature correspondence via graph matching: Models and global optimization. In: ECCV.

Tuytelaars, T., Lampert, C.H., Blaschko, M.B., Buntine, W., 2010. Unsupervised object discovery: A comparison. In: IJCV, Vol. 88. pp. 284–302.

Wang, C., Wang, L., Liu, L., 2014. Progressive mode-seeking on graphs for sparse feature matching. In: ECCV.

Wang, J., Zeng, Z., Zhou, L., 2006. Clan: An algorithm for mining closed cliques from large dense graph databases. In: ACM SIGKDD.

Wei, X.S., Zhang, C.L., Wu, J., Shen, C., Zhou, Z.H., 2017. Unsupervised object discovery and co-localization by deep descriptor transforming. In: IJCAI.

Xie, H., Gao, K., Zhang, Y., Li, J., Ren, H., 2012. Common visual pattern discovery via graph matching. In: ACM MM.

Yan, J., Li, Y., Liu, W., Zha, H., Yang, X., Chu, S.M., 2014. Graduated consistency-regularized optimization for multi-graph matching. In.

Yan, J., Tian, Y., Zha, H., Yang, X., Zhang, Y., Chu, S.M., 2013. Joint optimization for consistent multiple graph matching. In: ICCV.

Yan, X., Zhou, X., Han, J., 2005. Mining closed relational graphs with connectivity constraints. In: ACM SIGKDD.

Yuan, J., Zhao, G., Fu, Y., Li, Z., Katsaggelos, A., Wu, Y., 2012. Discovering thematic objects in image collections and videos. In: PAMI, Vol. 21. pp. 2207–2219.

Zeng, Z., Wang, J., Zhou, L., Karypis, G., 2006. Coherent closed quasi-clique discovery from large dense graph databases. In: ACM SIGKDD.

Zhang, Q., 2016. Kinect RGB-D Image Dataset https://sites.google.com/site/quanshizhang/dataset.

Zhang, Q., Cao, R., Shi, F., Wu, Y.N., Zhu, S.C., 2018. Interpreting cnn Knowledge via an Explanatory Graph. AAAI.

Zhang, Q., Cao, R., Wu, Y.N., Zhu, S.C., 2017. Growing Interpretable Part Graphs on Convnets via Multi-Shot Learning. AAAI.

Zhang, Q., Song, X., Shao, X., Shibasaki, R., Zhao, H., 2015a. From rgb-d images to rgb images: Single labeling for mining visual models. In: ACM Transactions on Intelligent Systems and Technology (ACM-TIST), Vol. 6. p. 16.

Zhang, Q., Song, X., Shao, X., Zhao, H., Shibasaki, R., 2013a. Category modeling from just a single labeling: Use depth information to guide the learning of 2d models. In: CVPR.

Zhang, Q., Song, X., Shao, X., Zhao, H., Shibasaki, R., 2013b. Learning graph matching for category modeling from cluttered scenes. In: ICCV.

Zhang, Q., Song, X., Shao, X., Zhao, H., Shibasaki, R., 2014. Attributed graph mining and matching: An attempt to define and extract soft attributed patterns. In: CVPR.

Zhang, Q., Song, X., Shao, X., Zhao, H., Shibasaki, R., 2016. Object discovery: soft attributed graph mining. In: PAMI 38.

Zhang, Q., Wu, Y.N., Zhu, S.C., 2015b. Mining and-or graphs for graph matching and object discovery. In: ICCV.

Zhao, G., Yuan, J., 2010. Mining and cropping common objects from images. In: ACM MM.