Fast Multi-Image Matching via Density-Based Clustering

Roberto Tron
Boston University
tron@bu.edu

Xiaowei Zhou, Carlos Esteves, Kostas Daniilidis
University of Pennsylvania
{xiaowz,machc,kostas}@seas.upenn.edu

Abstract

We consider the problem of finding consistent matches across multiple images. Previous state-of-the-art solutions use constraints on cycles of matches together with convex optimization, leading to computationally intensive iterative algorithms. In this paper, we propose a clustering-based formulation. We first rigorously show its equivalence with the previous one, and then propose QuickMatch, a novel algorithm that identifies multi-image matches from a density function in feature space. We use the density to order the points in a tree, and then extract the matches by breaking this tree using feature distances and measures of distinctiveness. Our algorithm outperforms previous state-of-the-art methods (such as MatchALS) in accuracy, and it is significantly faster (up to 62 times faster on some bechmarks), and can scale to large datasets (with more than twenty thousands features).

1. Introduction

Finding correspondences between points in two or more images is a fundamental step in many computer vision applications, such as matching of object instances [2, 30, 40], shape matching [10, 13], Structure from Motion [9, 18] and homography estimation [27]. Matching is typically performed by first determining repeatable *feature points* (e.g., corner-like patches), then extracting *feature descriptors* (such as SIFT [17], SURF [1], learned deep features [31] or others), and finally associating them across images using relative distances in feature space. Traditionally, the problem has been considered only in its *pairwise* version, i.e., with pairs of images. However, the applications above imply a *multi-image* version, where matches need to be found across several images, and need to map consistently to an underlying set of physical entities (e.g., 3-D points). Researchers have started to rigorously consider the multi-image version only recently. Early practical solutions used simple greedy strategies to add and remove pairwise matches [10]. More recent strategies formulate multi-image matching as a version of an optimal assignment problem [10, 23, 40]. In this paper, we use a *clustering* formulation of the problem, where each multi-image match is represented by a cluster in feature space. We first rigorously show that this formulation is consistent with previous ones that enforce *cycle consistency* in the graph of pairwise matches (§2). We then review existing practical approaches (§3) and propose QuickMatch (§4), an algorithm derived from QuickShift [29] that performs clustering by seeking the *modes* of an empirical *density estimate*. This strategy deals well with the typical conditions found in matching (large and unknown number of clusters whose size is limited by the total number of images). QuickMatch organizes all the datapoints (all the feature descriptors from all the images) in a tree (based on the density estimate, as in QuickShift) and then breaks it to find the clustering (i.e., matches). However, with respect to QuickShift, QuickMatch takes into account two characteristics of the matching problem: the *distinctiveness* of the descriptors (that is, the fact that it is more difficult to correctly match a point if there are other, distinct points in the scene with similar appearance) and the *exclusion constraints* given by the implicit mapping to distinct physical entities (e.g. 3-D points) in an underlying *universe*. We test QuickMatch against previous approaches on standard benchmarks, small scale Structure from Motion problems, and large scale object matching (§5).

Paper contributions Our first contribution is a rigorous analysis of the relation between the multi-image matching problem and graphs of pairwise matches. This analysis is based on graph theory, is more general than previous results that use factorizations of permutations, and it provides the theoretical justification for our clustering-based method, QuickMatch, which is also our most significant contribution. With respect to previous work, QuickMatch *1)* represents a novel application of density-based clustering; *2)* directly outputs consistent multi-image matches without explicit pre-processing (e.g., initial pairwise decisions) or post-processing (e.g., thresholding of a matrix); *3)* is non-iterative, deterministic, and initialization-free; *4)* produces better results in a small fraction of the time; *5)* can scale to large datasets that previous methods cannot handle; *6)* takes advantage of the distinctiveness of the descriptor as done in traditional matching [17] to counteract the problem of repeated structures; *7)* does not assume a one-to-one correspondence of features between images; *8)* does not require knowing the number of entities (i.e., clusters) in the *universe*.

In addition, our experiments (§5) uncover the fact that multi-matching methods yield more significative improvements when, assuming a bounded feature space, the ratio between features per cluster and total number of features is higher (in hindsight, this is intuitive, since it implies more easily recognizable clusters). This was not noticed before due to the poor scaling of existing algorithms.

2. Results on Multi-Image Matching and Graphs of Pairwise Matches

In this section we introduce different equivalent definitions of the multi-image matching problem, and use graph theory to analyze their relation with pairwise matches. These definitions motivate the two main practical approaches based on cycle consistency and on clustering (see also §3).

We consider a problem with N images, each one identified with a label i in the set $\mathcal{I} = \{1, \dots, N\}$. A set of K_i features, denoted as $\{x_{ik}\}_{k=1}^{K_i}$, is extracted from each image $i \in \mathcal{I}$. Each feature x_{ik} represents an entity in the universe (as seen in image i), and contains a descriptor (a vector encoding appearance information) lying in a *descriptor space*. We assume that we have available either a *distance* $d(x_{i_1k_1}, x_{i_2k_2}) \geq 0$ or a *similarity* $h(x_{i_1k_1}, x_{i_2k_2}) \in [0, 1]$ between any two features $x_{i_1k_1}$ and $x_{i_2k_2}$. We denote the set of *all* extracted features as $\mathcal{X} = \{x_{ik}\}_{k\in 1,\dots,K_i}^{i\in\mathcal{I}}$.

Let $x_{i_1k_1} \to x_{i_2k_2}$, $i_1 \neq i_2$ denote a *pairwise match*, that is, our belief that a feature $x_{i_1k_1}$ "maps to" another feature $x_{i_2k_2}$ representing the same entity. With practical algorithms, matches are not necessarily symmetric (i.e., $x_{i_1k_1} \to x_{i_2k_2}$ might not imply $x_{i_2k_2} \to x_{i_1k_1}$, see §3.1).

We now consider two constructs for discussing *multi-image matches* (matches spanning multiple images): the *directed graph of pairwise matches* $\mathcal{G}$ and the *set of multi-image matches* $\mathcal{M}$. The main result of this section is then to provide different equivalent definitions of a set of multi-image matches in terms of directed graphs, laying the theoretical foundations for the algorithms discussed in §3 and §4.

We first need a few definitions adapted from graph theory.

Definition 1. *The* directed graph of pairwise matches *is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the set of vertices $\mathcal{V}$ corresponds to the set of features $\{x_{ik}\}$, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of pairwise matches of the form $x_{i_1k_1} \to x_{i_2k_2} \in \mathcal{E}$.*

Given a subset of features $\mathcal{C} \subset \mathcal{V}$, we define the *subgraph of $\mathcal{G}$ restricted to $\mathcal{C}$* as the directed graph $\mathcal{G}|_{\mathcal{C}} = (\mathcal{C}, \mathcal{E}')$ where $\mathcal{E}' = \{x_{i_1k_1} \to x_{i_2k_2} \in \mathcal{E} : x_{i_1k_1}, x_{i_2k_2} \in \mathcal{C}\}$. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a *clique* if $x_{i_1k_1} \to x_{i_2k_2} \in \mathcal{E}$ for all $x_{i_1k_1}, x_{i_2k_2} \in \mathcal{V}$ (i.e., all possible matches are present). A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a *union of cliques* if there exist partitions $\{\mathcal{C}_c\}$ and $\{\mathcal{E}_c\}$ of $\mathcal{V}$ and $\mathcal{E}$ of the same cardinality such that $\mathcal{G}|_{\mathcal{C}_c}$ is a clique with edges $\mathcal{E}_c$ for all c, where c is an index over the elements of the partitions. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be *connected* if for any $x_{i_1k_1}, x_{i_nk_n} \in \mathcal{V}$ there exist a sequence $x_{i_1k_1}, x_{i_2k_2}, \dots, x_{i_nk_n}$, called a *path*, such that $x_{i_jk_j} \to x_{i_{j+1}k_{j+1}} \in \mathcal{E}$ for all $j \in \{1, \dots, n-1\}$. A *cycle* is a path where start and end coincide, $x_{i_1k_1} = x_{i_nk_n}$.

We capture the intuitive concept of multi-image matching with the following definition:

Definition 2. *A* set of multi-image matches *is a set $\mathcal{M} = \{\mathcal{C}_c\}$ of* clusters *$\mathcal{C}_c = \{x_{i_1k_1}, x_{i_2k_2}, \dots\}$, each one corresponding to a single entity in the universe.*

It is immediate to state an equivalent definition in terms of pairwise matchings and cliques:

Definition 3. *A set of multi-image matches is a set $\mathcal{M} = \{\mathcal{C}\}$ of clusters $\mathcal{C}_c = \{x_{i_1k_1}, x_{i_2k_2}, \dots\}$ such that*

(C1) $\mathcal{M}$ is a partition of $\mathcal{X}$ (i.e., each feature x_{ik} appears exactly in one set $\mathcal{C}_c$);

(C2) Each set $\mathcal{C}_c$ has at most one feature per image;

(C3) There is an induced directed graph *$\mathcal{G}_{\mathcal{M}} = (\mathcal{X}, \mathcal{E}_{\mathcal{M}})$ of pairwise matches such that, for any $\mathcal{C}_c \in \mathcal{M}$, the subgraph $\mathcal{G}_{\mathcal{M}}|_{\mathcal{C}_c}$ is a clique (i.e., $\mathcal{G}_{\mathcal{M}}$ is a union of cliques).*

Conditions (C1) and (C2) prescribe that the same entity cannot appear in multiple clusters, and that a single entity cannot appear more than once per image. Condition (C3) requires that features from the same cluster always match.

Proposition 1. *Definition 3 implies the following properties on the induced graph $\mathcal{G}_{\mathcal{M}} = (\mathcal{X}, \mathcal{E}_{\mathcal{M}})$:*

(P1) Symmetry: If $x_{i_1k_1} \to x_{i_2k_2} \in \mathcal{E}_{\mathcal{M}}$, then $x_{i_2k_2} \to x_{i_1k_1} \in \mathcal{E}_{\mathcal{M}}$.

(P2) Cycle constraint: given a path $x_{i_1k_1}, x_{i_2k_2}, \dots, x_{i_nk_n}$ in $\mathcal{G}_{\mathcal{M}}$, having $i_1 = i_n$ (start and end images coincide) implies $k_1 = k_n$ (the path is a cycle).

(P3) Single match: If $x_{i_1k_1} \to x_{i_2k_2}$ and $x_{i_1k_1} \to x_{i_2k'_2}$ belong to $\mathcal{G}_{\mathcal{M}}$, then $k_2 = k'_2$ (a feature cannot correspond to two different features in another image).

Proof. Property (P1) is implied by the definition of clique in (C3). Property (P2) can be proved by way of contradiction: assume that $k_1 \neq k_n$; using the sequence of matches and the cycle constraint, we can deduce that $x_{i_1k_1} \to x_{i_nk_n}$; however this is in contradiction with (C2), unless $k_1 = k_n$. Property (P3) can be shown by using property (P1) to build the path $x_{i_2k_2}, x_{i_1k_1}, x_{i_2k'_2}$, and then using (P2). □

Given Definition 2, the *multi-image matching* problem can be formalized as follows.

Problem 1. *Given the set of features $\mathcal{X}$ and a distance $d(\cdot, \cdot)$ or a similarity $h(\cdot, \cdot)$, build a set of multi-image matches $\mathcal{M}$.*

Pairwise matching alone (see §3.1) cannot be used to solve Problem 1, since, in general, it does not produce graphs that are union of cliques. Instead, as we discussed in §3, practical algorithms aim to either building the induced graph $\mathcal{G}_{\mathcal{M}}$ (see §3.2) or the clustering $\mathcal{M} = \{\mathcal{C}_c\}$ (see §3.3 and §4). From a computational standpoint, directly enforcing the constraint imposed by (C3) (while pursuing some kind of optimality) is hard, while it is much easier to deal with the constraints given by properties (P1)–(P3). In particular, (P2) has inspired the use of the concept of *cycle consistency* in the literature. The definition we give below is the one of [10] paraphrased with our graph-theoretic terminology.

Definition 4. *A directed graph of pairwise matches $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is* cycle consistent *if, for any $x_{i_1k_1}, x_{i_2k_2}, x_{i_3k_3} \in \mathcal{V}$,*

(L1) $x_{i_1k_1} \rightarrow x_{i_2,k_2} \in \mathcal{E}$ implies $x_{i_2k_2} \rightarrow x_{i_1,k_1} \in \mathcal{E}$ (2-cycle consistency, see also (P1));

(L2) $x_{i_1k_1} \rightarrow x_{i_2k_2}, x_{i_2k_2} \rightarrow x_{i_3k_3} \in \mathcal{G}$ implies $x_{i_3k_3} \rightarrow x_{i_1k_1}$ (3-cycle consistency).

The following result shows the equivalence between Definition 4 and (C3).

Proposition 2. *A directed graph of pairwise matches is cycle consistent if and only if it is a union of cliques.*

Proof. Cycle consistency implies that all matches are symmetric. We can therefore partition the graph $\mathcal{G}$ into *connected components* (maximally connected subgraphs). We prove the claim by showing that for any two $x_{i_1k_1}, x_{i_n,k_n}$ in the same component, $x_{i_1k_1} \rightarrow x_{i_n,k_n} \in \mathcal{E}$. By the definition of connected graph there exist a path $x_{i_1k_1}, x_{i_2k_2}, x_{i_3k_3} \ldots, x_{i_n,k_n}$. Using the definition of cycle consistency and the first two correspondences in the path, we deduce that $x_{i_1k_1} \rightarrow x_{i_3k_3}$. We can therefore build a shorter path $x_{i_1k_1}, x_{i_3k_3} \ldots, x_{i_n,k_n}$. By repeating the same argument we can build a sequence of paths $x_{i_1k_1}, x_{i_jk_j} \ldots, x_{i_n,k_n}, j \in \{2, \ldots, n\}$ until we arrive to $x_{i_1k_1} \rightarrow x_{i_n,k_n}$, thus showing the claim. □

As a consequence of Proposition 2, we can provide another equivalent definition of multi-image matches.

Theorem 1. *In Definition 3, condition (C3) can be equivalently substituted with the following:*

(C4) There is an induced directed graph $\mathcal{G}_{\mathcal{M}} = (\mathcal{X}, \mathcal{E}_{\mathcal{M}})$ of pairwise matches such that, for any $\mathcal{C}_c \in \mathcal{M}$, the subgraph $\mathcal{G}_{\mathcal{M}}|_{\mathcal{C}_c}$ is cycle consistent.

Overall, the goal of this section is to show that cycle-consistency (see Theorem 1) and clustering (see Definition 2) are both valid ways to approach Problem 1. In particular, this justifies the use of a clustering algorithm (QuickMatch) instead of relying on cycle consistency constraints (as previously done). Note that our results are more general than previous ones based on factorizations of permutation mappings [10, 23, 40], which assume a one-to-one correspondence between *universe* and features.

3. Review of Multi-Matching Algorithms

Section 2 provides the relation between feature clusters pairwise matches. However, practical algorithms need to solve Problem 1 and decide what matches are valid (while respecting the multi-image matching constraints above) by using actual distances or similarities. In this section we review existing algorithms, starting with basic pairwise matching.

3.1. Pairwise Matching

In an ideal world, descriptors for the same entity should be identical across images, and thus have zero distance or maximum similarity. Hence, the most straightforward criterion to use is to declare a pairwise match $x_{i_1k_1} \rightarrow x_{i_2k_2}$ whenever the distance $d(x_{i_1k_1}, x_{i_2k_2})$ (or the corresponding similarity) is lower (higher) than a fixed threshold. However, this might produce incorrect matches when different entities have similar appeareances in the same image (e.g., due to repeated structures). A way to improve the results is to adapt the threshold based on the *distinctiveness* of a point, quantified by the distance to the closest descriptor in the same image. This method, first proposed in [17], declares a match when

$$d(x_{i_1k_1}, x_{i_2k_2}) < \rho d_{i_1k_1}, \tag{1}$$

where ρ is a fixed constant (typically in the range $[0.33, 1]$) and $d_{i_1k_1} = d(x_{i_1k_1}, x_{i_1k^*})$ is the distance between the descriptor k_1 in image i_1 and the closest descriptor k^* in the same image. In practice this simple criterion is very effective in preventing a large number of incorrect matches. Note that an analogous criterion (with a reversed inequality) can also be formulated with similarities instead of distances.

3.2. Cycle Consistency and Optimization

Algorithms for multi-image matching solve Problem 1 by trying to find "tight" clusters inside which features have low distances or high similarities among them. Early work aimed to simply identify and remove bad matches by using the cycle exclusion property (P2) alone; these approaches, however, require relatively high percentage of good matches to work [10]. The majority of later approaches, instead, attack the problem by considering the full characterization given by Theorem 1, and use cycle consistency constraints.

While some works pose the problem as a global non-convex optimization [13, 32, 33, 35, 36, 39], the most reliable solutions are based on convex relaxations [10, 23, 40]. These approaches keep track of the matches using an indicator matrix X. This matrix has dimensions $K \times K$, where $K = \sum_{i=1}^{N} K_i$ is the total number of points, and its stucture is

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1N} \\ X_{21} & X_{22} & \cdots & X_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ X_{N1} & \cdots & \cdots & X_{NN} \end{bmatrix}, \tag{2}$$

where each block X_{ij} has dimension $K_i \times K_j$. Ideally, each entry of X should be binary (where 1 indicates correspondences between two points); however, this constraint is typically relaxed to the unit interval $[0,1]$. Intuitively, each matrix X_{ij} contains the pairwise matching information from image j to i encoded as a permutation matrix. The self-permutation X_{ii} is canonically set to the identity matrix for all i. Cycle consistency (Definition 4) can be expressed with the constraints $X_{ij} = X_{ji}^{\mathrm{T}}$ (2-cycle consistency, (L1)) and $X_{i_1 i_3} = X_{i_1 i_2} X_{i_2 i_3}$ (3-cycle consistency, (L2)). The remaining properties required by the definition of multi-image matches (see Theorem 1 and Definition 3) can be imposed by enforcing $0 \leq X_{ij}\mathbf{1} \leq 1$ (the same inequalities with the transpose X_{ij} are ensured by the symmetry constraint). The papers [10, 23] show that, after assuming that $K_i = M$ for all $i \in \mathcal{I}$, and that the M features in any image are in one-to-one correspondence with those in any other image, the cycle consistency constraint implies the low-rank factorization $X = AA^{\mathrm{T}}$, where $A \in \mathbb{R}^{K \times M}$. Each M-by-M block in A can be interpreted as a permutation mapping the M entities of the universe to the features in each image (this is equivalent to the main result of §2, albeit in a less general setting). Using this fact, [10, 23, 40] optimize an objective function of the form $f(X) = \operatorname{tr}(S, X)$ subject to X satisfying all the constraints above, plus a semi-definite positive (SDP) constraint [10] or a nuclear norm low-rank regularizer [40]; or subject to $X = AA^{\mathrm{T}}$, where A is orthonormal [23]. The matrix $S \in \mathbb{R}^{K \times K}$ represents the matrix of distances or similarities between feature points computed from $d(\cdot,\cdot)$, $h(\cdot,\cdot)$, or other similar information (such as the result of pairwise graph matching [40]). The papers [10, 40] solve the optimization problem using the Augmented Lagrangian Method (ALM), while [23] reduces the problem to finding the leading singular vectors of the similarity matrix S[1]. This line of methods has a few limitations. While some of these methods take into consideration the distinctiveness of the features (1) while computing the input pairwise matches, this information is not used again during the multi-match problem. Moreover, since these approaches relax the constraint that the matrix A must have discrete entries in $\{0, 1\}$, an additional thresholding step might be required. Finally, the size of the universe M needs to be explicitly provided.

Additional methods We mention the works of [22, 37], which are also based on the use of permutation matrices from a universe to the features, but exploit the fact that these matrices, when applied to the feature descriptors, should ideally produce identical vectors that can then be assembled into a rank-1 matrix. These approaches share the same limitations as [10, 23, 40]. A parallel line of work [32, 33, 35, 36] considers the generalization of the multi-image matching problem given by the multi-image *graph* matching problem, where second order similarities (i.e., similarities between pairs of pairs of features) are taken into account. Directly incorporating these second order similarities is not considered in this paper, although this information can be indirectly used by treating them as first-order similarities (as done in [40]).

[1] Interestingly, the resulting algorithm is equivalent to spectral clustering (see §3.3), although the authors of [23] did not realize this connection.

3.3. Clustering

In this work we attack Problem 1 using the clustering perspective given by Definition 2. The basic premise is that points in the same cluster $\mathcal{C}_c$ should form a tight group in feature space (see Fig. 1b). In principle, we could apply popular clustering algorithms such as k-means [19] or spectral clustering [21], but these algorithms require specifying in advance the number of clusters M, which is generally unknown. More importantly, they do not exploit two specific characteristics of multi-image matching. First, the distinctiveness criterion (1) can be used to estimate the size and separations of the clusters. Second, the exclusion properties (C1) and (C2) from the definition impose restrictions on the possible clustering.

In [34], the clustering perspective is used in an algorithm that iterates between a matching step and the selection of *mean features* (similarly to k-means); this algorithm is sensitive to the initialization used, and multiple iterations are typically needed in order to achieve convergence.

In this paper, we will instead build upon density-based clustering algorithms. These algorithms originated with MeanShift [7], and aim to find clusters of points from the modes of a non-parametric estimate of the density distribution of the data [24, 25]. Among these algorithms, we choose QuickShift [29], which is efficient and can be easily modified to incorporate the descriptor distinctiveness and the exclusion property constraint. Additionally, this algorithm does not make any explicit assumption on the shape of the clusters, and allows, for instance, non-isotropic, elongated clusters (due, for instance, to points seen from different viewpoints). The details of the algorithm are presented next.

4. QuickMatch: Method Description

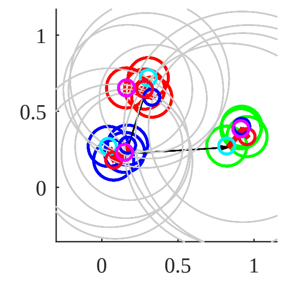
Our algorithm, which we call QuickMatch, first organizes all descriptors $\{x_{ik}\}$ in a tree (similarly to QuickShift). Then, this tree is broken into small connected components (i.e., a forest) representing the multi-image matches (clusters). The details are given below, and see Algorithm 1 for a summary.

4.1. Density Estimation

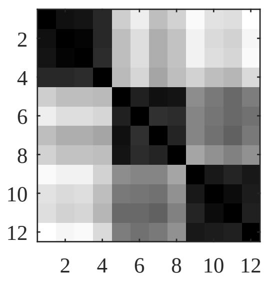
We start by defining a density function

$$D(x) = \sum_{i=1}^{N} \sum_{k=1}^{K_i} a(d_{ik}) \tilde{h}(x, x_{ik}; \rho_{\mathrm{den}} d_{ik}) \qquad (3)$$

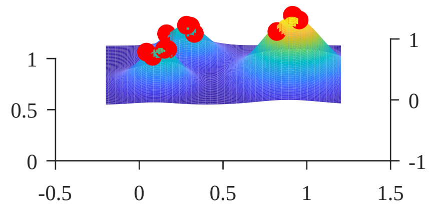
where $\tilde{h}$ is a density kernel function with bandwidth $\rho_{\mathrm{den}} d_{ik}$, $0 < \rho_{\mathrm{den}} < 1$ is a user-defined ratio, a is an arbitrary



(a) Descriptors and QuickMatch clusters. Colors of the small/large circles: index of the image/match for each descriptors. Red arrows: the tree after QuickMatch. Black arrows: edges discarded by QuickMatch.



(b) Pairwise Euclidean distances between descriptors (darker colors mean lower values), ordered according to the ground-truth matches. Notice the block-diagonal structure.



(c) The density used by QuickMatch to order the points in a tree. Notice how the descriptors on the right are more discriminative (farther away with respect to other descriptors) and hence they produce a stronger peak).

Figure 1: A toy multi-matching problem with 4 "images" and 3 two-dimensional descriptors per image, its intepretation as a clustering problem, and the results of QuickMatch.

adaptive amplification factor that depends on d_{ik} and, again, d_{ik} is the distance between x_{ik} and the closest descriptor from the same image. In our experiments we will use a non-normalized Gaussian kernel with $\sigma = \rho_{\text{den}} d_{12}$,

$$\tilde{h}(x_1, x_2; \sigma) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right), \tag{4}$$

although other kernels (e.g., parabolic or triangular) could be used. Empirically, the value of $\rho_{\text{den}} = 0.25$ gives a good separation between two samples that are d_{ik} apart, while avoiding the fragmentation of clusters. The function $a(d_{ik})$ can be used to boost the density for descriptors that are easier to match (i.e., where d_{ik} is large), and effectively controls the relative importance of each match (given by the heights of its mode in the density). We use $a(d_{ik}) = \log(1 + d_{ik})$, which helps with the ordering of "relevance" of the clusters, but does not appreciably influence the clustering.

Use of similarities instead of distances If instead of distances $d(\cdot,\cdot)$ we have similarities $h(\cdot,\cdot)$, we redefine (3) as $\tilde{h}(x_1, x_2; \sigma) = h(x_1, x_2)^{\frac{1}{\sigma^2}}$. This choice reduces to (4) when the similarity h is computed from d using a unit-bandwidth Gaussian. To produce sparser similarities, the elevation to power operation can also be approximated with the piecewise linear function $\tilde{h}(x_1, x_2; \rho) = \max(0, \frac{h(x_1,x_2)-\rho}{1-\rho})$ (this is used in the experiments in §5.3).

4.2. Construction of the Tree

Given a descriptor x_{ik}, the parent in the tree is given by the closest point with higher density from another image:

$$\operatorname{parent}(x_{ik}) = \operatorname*{argmin}_{i'k' \in J} d(x_{ik}, x_{i'k'}),$$
$$\text{where } J = \{i'k' : k \neq k', D(x_{i'k'}) > D(x_{ik})\}, \tag{5}$$

and $d(\cdot,\cdot)$ is the distance in the descriptor space (see Fig. 1a). If x_{ik} is the root of the tree (highest mode), then $\operatorname{parent}(x_{ik}) = \emptyset$. Equation (5) is the same as in QuickShift, except that the set J incorporates constraint (C2). If similarities are used instead of distances, then $\operatorname{argmin}_{\jmath \in J} d(x_\imath, x_\jmath)$ should be substituted with $\operatorname{argmax}_{\jmath \in J} h(x_\imath, x_\jmath)$.

Intuitively, with this procedure, a point tends to connect through a "short" edge to another point with a higher density in the same cluster, or, if it has already the highest value of the cluster (i.e., it represents a mode of D), then through a "long" edge to a node in another cluster (see Fig. 1a). Here, "short" and "long" refer to other edges in the same cluster.

4.3. Breaking the Tree into Clusters

Given a tree, we would like to break it into clusters (connected components) that correspond to valid multi-image matches, following the exclusion property (C2) and the distinctiveness criterion (1). Our algorithm works by tracking the collection of multi-image matches sets $\{\mathcal{C}_c\}$. We define two functions: $\texttt{matchImgs}(\mathcal{C}_c)$, which returns the indices of the images $\{i_1, i_2, \ldots\} \subset \mathcal{I}$ to which the features in $\mathcal{C}_c$ belong, and $\texttt{matchDis}(\mathcal{C}_c)$, which returns the tightest distinctiveness criterion, $\texttt{matchDis}(\mathcal{C}_c) = \min_{ik:x_{ik} \in \mathcal{C}_c} d_{ik}$ (if similarities are used, $\texttt{matchDis}(\mathcal{C}_c) = \max_{ik:x_{ik} \in \mathcal{C}_c} h_{ik}$). We use a bottom-up procedure where we initialize one cluster for each point, and then consider each edge in the tree in ascending distance order (or descending similarity), from the shortest to the longest. Assume that the edge under consideration connects two datapoints x_{ik} and $x_{i'k'}$ that belong to two clusters $\mathcal{C}_c$ and $\mathcal{C}_{c'}$, respectively. We consider an edge valid, and merge $\mathcal{C}_c$ and $\mathcal{C}_{c'}$, when:

(EC1) The edge is short with respect to the distinctiveness criterion of the two clusters, that is $d(x_{ik}, x_{i'k'}) \leq \rho_{\text{edge}} \min\big(\texttt{matchDis}(\mathcal{C}_c), \texttt{matchDis}(\mathcal{C}_{c'})\big)$ (swap h with d and reverse the inequality if similarities are used).

(EC2) The sets of images covered by the two clusters are disjoint, $\texttt{matchImgs}(\mathcal{C}_c) \cap \texttt{matchImgs}(\mathcal{C}_{c'}) = \emptyset$;

If either condition is not satisfied, we discard the edge. Since the only operation we are performing on the clusters is merging, the sets $\{\mathcal{C}_c\}$ are always disjoint (condition (C1)). Moreover, condition (EC2) implies that a single cluster cannot contain two points from the same image (condition (C2)). Finally, we canonically consider every feature in a cluster $\mathcal{C}_c$ as matching with all the other features in the same cluster (condition (C3)). The user-defined constant ρ_{edge} can be used to tune the balance between false positives and false negatives in the final matches. The algorithm terminates after considering all the edges (see Fig. 1a).

4.4. Computational Considerations

As attested by the experiments in §5.1, QuickMatch is several times faster and more scalable than previous solutions. Nevertheless, there are a few aspects that can greatly speed up an implementation with relatively modest effort. *1)* We can store the tree generated in §4.2 using two 1-D vectors (one for the indeces of the parent of each point, the other for the corresponding distance). *2)* The functions `matchImgs` and `matchDis` do not need to be explicitly implemented. Their outputs can instead be efficiently obtained using data structures that are updated when an edge is considered valid. In particular, the intersection operation in condition (EC2) can be efficiently implemented using *bit arrays*. *3)* The tree breaking procedure of §4.3 needs to handle sets of at most $2N$ datapoints, since each cluster, by construction, cannot contain more than N elements (N being the number of original images). *4)* The most computationally expensive steps of the algorithms are the computation of all the pairwise point distances and the corresponding kernel values in §4.1; however, these operations are embarrassingly parallel. *5)* For large datasets, we can significantly reduce the memory and computation requirements by considering only a subset of nearest neighbors for each point. If a sufficient number of neighbors is used, the approximation introduced is practically acceptable (this technique is used in §5.2 and §5.3). An optimized Matlab implementation is available on the first author's website (`http://sites.bu.edu/tron/software/`).

Algorithm 1 The QuickMatch algorithm

```
Require: Descriptors $\{x_{ik}\}$, parameters $\rho_{\text{den}}$, $\rho_{\text{edge}}$.
Ensure: Clusters $\{\mathcal{C}_c\}$ with multi-image matches.
  for all pairs $x_{ik}$, $x_{i'k'}$ do                ▷ Estimating the density
    Compute the pairwise density $\tilde{h}$ (e.g., using (4)).
  end for
  for all $x_{ik}$ do
    Compute density $D(x_{ik})$.
  end for
  for all $x_{ik}$ do                                  ▷ Building the tree
    Compute $\text{parent}(x_{ik})$.
  end for
  Initialize one cluster $\mathcal{C}_c$ per datapoint.  ▷ Breaking the tree
  Order edges in the tree from shortest to longest.
  for all edges (in order) do
    Let $x_{ik} \in \mathcal{C}_c$, $x_{i'k'} \in \mathcal{C}_{c'}$ be the endpoint of the edge
    if $d(x_{ik}, x_{i'k'}) \leq \rho_{\text{edge}} \min\big(\texttt{matchDis}(\mathcal{C}_c, \mathcal{C}_{c'})\big)$
  and $\texttt{matchImgs}(\mathcal{C}_c) \cap \texttt{matchImgs}(\mathcal{C}_{c'}) = \emptyset$ then
      Merge $\mathcal{C}_c$ and $\mathcal{C}_{c'}$.
    else
      Discard the edge.
    end if
  end for
```

5. Experimental Evaluation

5.1. Graffiti Dataset

In this section we use the Graffiti dataset[2] to evaluate QuickMatch. The main goal of these experiments is to use the *same evaluation protocol* as previous work [40] to compare our proposed solution with other state-of-the-art [23,40] and baseline algorithms (pairwise matching).

The dataset contains images from six planar scenes with six views each, and a manual ground-truth annotation of around 1000 keypoints. Using the evaluation protocol introduced in [20, 40], we extract 1000 SIFT features from each image using VLFeat [28]. The multi-image matching problem is solved using traditional pairwise matching, the spectral method of [23], the MatchALS method of [40] and QuickMatch (with $\rho_{\text{edge}} = 0.7$). We evaluate each result by transferring the annotated keypoint from each view into the other views using Matlab's `scatteredInterpolant` function to interpolate from the matched features, and then calculate the distance between the estimated and true annotated keypoint positions. We compute the percentage of matches that have errors less than a thresholds ranging from zero to 0.1 pixels, and plot the resulting curves, shown in Fig. 2. Tables 1 contain the summary of the area-under-the-curve (AUC) metric and computation times (all recorded on the same machine). This protocol with annotated keypoints and interpolation allows to indirectly evaluate the quality of the matches independently from their number (thus avoiding the need for precision/recall curves). Overall, QuickMatch improves the AUC metric by 3.2 to 17.9 percentage points with respect to the Spectral method and by 1.5 to 5.5 with respect to MatchALS (which represents the previous best method in this area). Moreover, QuickMatch is 8.8 to 32.1 times faster than the Spectral method and 34.9 to 62.2 times faster than MatchALS.

5.2. Structure-from-Motion Datasets

In this section we compare QuickMatch against the pairwise matching baseline on small Structure from Motion

[2]http://www.robots.ox.ac.uk/˜vgg/data/data-aff.html

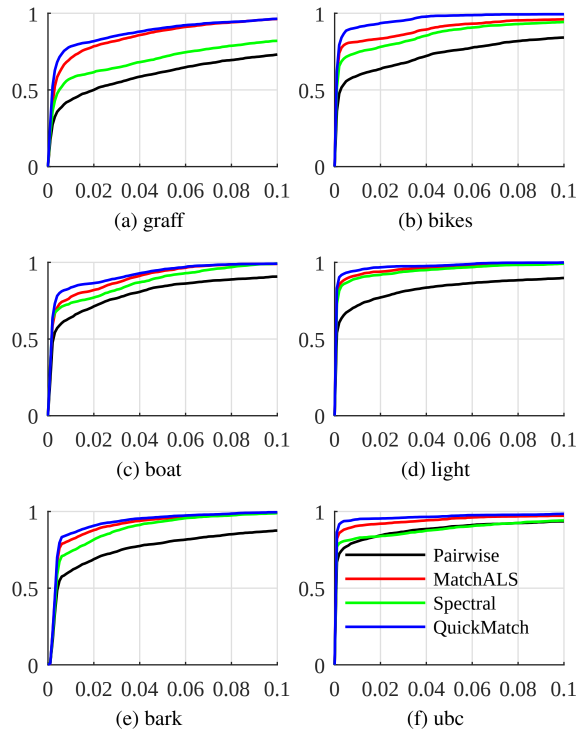


Figure 2: Graffiti dataset: Curves for percentage of annotated points (y-axis) versus distance in pixels (x-axis).

(SfM) datasets. The goal of these experiments is not to fully evaluate the application of QuickMatch to SfM (which would require considering the effect of the entire SfM pipeline and is out of the scope of this paper), but rather to provide an analysis which is complementary to the one of §5.1 by allowing the explicit calculation of precision/recall curves (using the epipolar constraint as validation). More in detail, we use the SfM datasets from [26] which include ground-truth camera pose information. We extract around 500–600 SIFT feature points from each image (again using VLFeat) and perform both pairwise matching and multi-image matching using QuickMatch. We can validate matches across pairs of images by calculating the essential matrices from the provided ground truth camera poses, and then computing the epipolar-line-distance errors for the matches. For a given image pair, we compute the precision ($\frac{N_{\text{th}}}{N_{\text{matches}}}$, x-axis)-recall ($\frac{N_{\text{th}}}{N_{\text{features}}}$, y-axis) curves (where N_{matches} is number of matches returned, N_{features} is the lower number of features from either image, and N_{th} is number of matches with epipolar-line-distance error less than $5\,\text{px}$). The curves are generated by varying the parameter ρ_{edge} for QuickMatch in the range $[0.5, 1.5]$ and the analogous parameter for pairwise matching (namely, the threshold `THRESH` in VLFeat's `vl_ubcmatch`) in the range $[1, 10]$. The curves, shown in Fig.5, are averaged across all pairs of images. The precision values are relatively low because the actual number of true matches between features extracted in the different pairs of images is unknown, so we use the number of features in the images N_{matches} as a proxy. In all datasets, the use of QuickMatch consistently improves the precision of the matches. In general, QuickMatch takes around twice the time of computing pairwise matches. For the sake of transparency, we mention that if we repeat the same experiment with a significantly larger number of extracted features (in the order of tens of thousands instead of 500–600), the use of QuickMatch does not provide significant benefits vis-à-vis pairwise matching. We conjecture that this is due to the fact that in this regime feature vectors become quite dense in feature space (while the number of features per clusters remains about the same), and the descriptors are not informative enough. This conjecture is supported by the results of the next section, in which QuickMatch performs significantly better than pairwise matching.

	Pairwise	Spectral	MatchALS	QuickMatch
graff	58.8	68.8 (23.3 s)	84.3 (68.0 s)	**86.7** (**1.6 s**)
bikes	72.1	85.0 (53.1 s)	88.8 (94.9 s)	**95.3** (**1.7 s**)
boat	79.3	86.6 (13.2 s)	89.6 (61.7 s)	**91.5** (**1.5 s**)
light	81.9	94.0 (54.6 s)	95.3 (105.8 s)	**96.9** (**1.7 s**)
bark	75.1	87.6 (17.8 s)	90.3 (60.7 s)	**91.8** (**1.5 s**)
ubc	87.4	87.5 (16.7 s)	93.4(59.4 s)	**95.7** (**1.7 s**)

Table 1: Graffiti dataset: Area under the curve and computation times (in parentheses). Times for Spectral and MatchALS do not include pairwise matching.

5.3. Object Instance Matching Dataset

Finding semantic correspondences between images of different object instances is an open problem [4,11,38], with applications in reconstruction of object category models for object shape and pose estimation [12]. In this section, we demonstrate on the FG3DCar dataset [15] that QuickMatch can not only improve the matching accuracy (compared to pairwise matching), but also serve as an approach to discover discriminative parts shared by a class of objects.

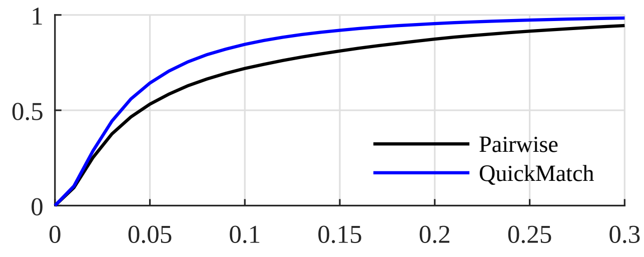


Figure 3: FG3DCar dataset: Curves for percentage of annotated points (y-axis) versus distance in pixels (x-axis).

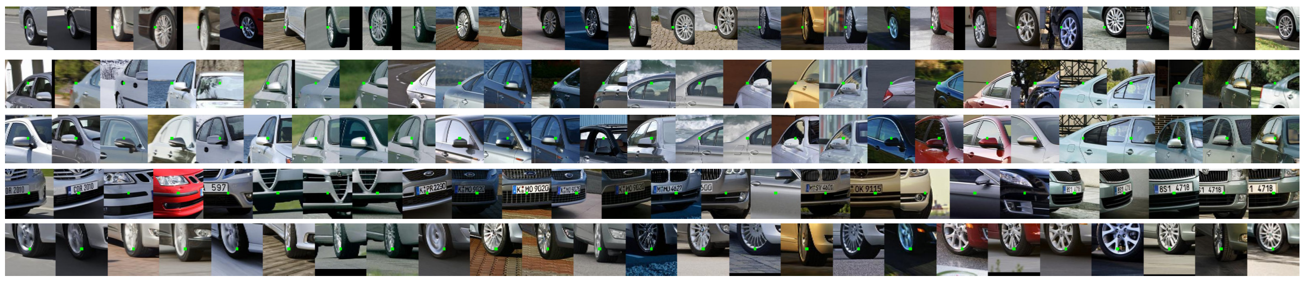


Figure 4: Discovered parts in sedan images.

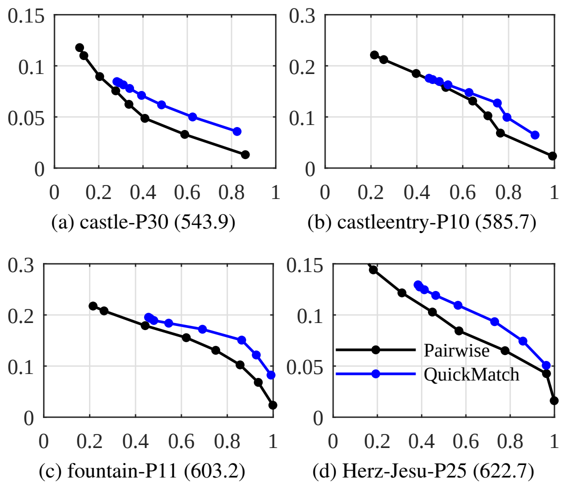


Figure 5: Results on the SfM datasets from [26], curves for recall (y-axis) versus precision (x-axis). Captions show dataset name, number of images and average number of features/image.

The FG3DCar dataset consists of high-resolution car images with keypoint and viewpoint annotations. In our experiment, all left-view sedan images (37 images in total) are used for matching. As the cars in two images might be of different models with very different local features, traditional geometric features like SIFT can hardly work. Instead, we adopt the deep features extracted from pre-trained convolutional neaural networks (CNNs) for matching, which have been proven to be effective in previous work [16]. Consistent with [40], we assume that the background segmentation is given, and we sample feature points along the edges on objects. The edges are detected by the structured forests [6]. Then, we forward each image through the AlexNet [14] trained on ImageNet [5] and stack the feature map responses from Conv1 to Conv5 corresponding to each feature point as its descriptor (a.k.a. hypercolumns [8]). In order to leverage the prior of object rigidity and avoid forming a dense distance matrix as the input to QuickMatch, we first run graph matching for all pairs of images using the Reweighted Random Walk algorithm [3] and collect the output scores of graph matching to form a sparse affinity matrix for all feature points. This affinity matrix is used to provide the similarities $h(\cdot,\cdot)$ for QuickMatch. Graph matching took around 5 s per image pair, while QuickMatch (with the computed pairwise similarities), took around 108 s to run on the entire dataset. Note that the other optimization-based method like MatchALS cannot be used here due to the large size of the problem (which contains more than twenty thousands features).

We evaluate the accuracy using the same protocol used for the Graffiti dataset in §5.1. The matching accuracy is shown in Figure 3, which clearly shows the advantage of joint matching. We also visualize the top-5 largest clusters by showing the image patches centered at the feature points in each cluster, as shown in Figure 4. Despite of few outliers, all patches in the same cluster correspond to a semantically meaningful part, such as wheel, window and light. This illustrates the possibility of using the proposed approach for automatic discovery of semantic parts in a large collection of images, which could be potentially used for other high-level tasks, such as fine-grained classification and pose estimation. Note that the only training data used here is ImageNet with class labels. No correspondence or part annotation is used.

6. Conclusion and Future Work

We presented QuickMatch, a novel method that solves the multi-image matching problem as a clustering problem using density-based techniques, distinctiveness of features, and exclusion constraints. Our method significantly improves the accuracy of state-of-the-art multi-image methods while being at least one order of magnitude faster and more scalable. In the future we will investigate iterative post-processing strategies for improving the performance of QuickMatch, especially in the challenging regime with a large number of distinct features but a low number of samples for each feature. We will also study the integration of QuickMatch with deep learning frameworks for improving semantic matching.

We gratefully acknowledge the support of ONR grant N00014-17-1-2093.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] J. Carreira, A. Kar, S. Tulsiani, and J. Malik. Virtual view networks for object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2937–2946, 2015.

[3] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*. Springer, 2010.

[4] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *NIPS*, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision*, 2013.

[7] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[10] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. *Computer Graphics Forum*, 32(5):177–186, 2013.

[11] A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *CVPR*, 2016.

[12] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *CVPR*, pages 1966–1974, 2015.

[13] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. A. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Transactions on Graphics*, 31(4):54, 2012.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[15] Y.-L. Lin, V. I. Morariu, W. Hsu, and L. S. Davis. Jointly optimizing 3d model fitting and fine-grained classification. In *European Conference on Computer Vision*, 2014.

[16] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NIPS*, 2014.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[18] Y. Ma. *An invitation to 3-D vision: from images to geometric models*. Springer, 2004.

[19] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[20] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Neural Information Processing Systems*, 2:849–856, 2002.

[22] R. Oliveira, J. Costeira, and J. Xavier. Optimal point correspondence through the use of rank constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1016–1021, 2005.

[23] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems*, 2013.

[24] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[25] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[26] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[27] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[28] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[29] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *IEEE European Conference on Computer Vision*, pages 705–718. Springer, 2008.

[30] S. Vicente, J. Carreira, L. Agapito, and J. Batista. Reconstructing Pascal VOC. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2014.

[31] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *International Conference on Computer Vision*, 2013.

[32] J. Yan, M. Cho, H. Zha, X. Yang, and S. Chu. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[33] J. Yan, Y. Li, W. Liu, H. Zha, X. Yang, and S. M. Chu. Graduated consistency-regularized optimization for multi-graph matching. In *European Conference on Computer Vision*, 2014.

[34] J. Yan, Z. Ren, H. Zha, and S. Chu. A constrained clustering based approach for matching a collection of feature sets. In *International Conference on Pattern Recognition*, 2016.

[35] J. Yan, Y. Tian, H. Zha, X. Yang, Y. Zhang, and S. Chu. Joint optimization for consistent multiple graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1649–1656, 2013.

[36] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu. Consistency-driven alternating optimization for multigraph matching: a unified approach. *Image Processing, IEEE Transactions on*, 24(3):994–1009, 2015.

[37] Z. Zeng, T.-H. Chan, K. Jia, and D. Xu. Finding correspondence from multiple images via sparse and low-rank decomposition. In *IEEE European Conference on Computer Vision*, pages 325–339. Springer, 2012.

[38] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016.

[39] T. Zhou, Y. J. Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1191–1200. IEEE, 2015.

[40] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.