Multi-Image Matching via Fast Alternating Minimization

Xiaowei Zhou, Menglong Zhu, Kostas Daniilidis
GRASP Laboratory, University of Pennsylvania
{xiaowz, menglong, kostas}@seas.upenn.edu

Abstract

In this paper we propose a global optimization-based approach to jointly matching a set of images. The estimated correspondences simultaneously maximize pairwise feature affinities and cycle consistency across multiple images. Unlike previous convex methods relying on semidefinite programming, we formulate the problem as a low-rank matrix recovery problem and show that the desired semidefiniteness of a solution can be spontaneously fulfilled. The low-rank formulation enables us to derive a fast alternating minimization algorithm in order to handle practical problems with thousands of features. Both simulation and real experiments demonstrate that the proposed algorithm can achieve a competitive performance with an order of magnitude speedup compared to the state-of-the-art algorithm. In the end, we demonstrate the applicability of the proposed method to match the images of different object instances and as a result the potential to reconstruct category-specific object models from those images.

1. Introduction

Finding feature correspondences between two images is a fundamental problem in computer vision with various applications such as structure from motion, image registration, shape analysis, to name a few. While previous efforts were mostly focused on matching a pair of images, many tasks require to find correspondences across multiple images. A typical example is nonrigid structure from motion [3, 12], where one can hardly reconstruct a nonrigid shape from two frames. Furthermore, recent work has shown that leveraging multi-way information can dramatically improve matching results compared to pairwise matching [29, 16].

The most important constraint for joint matching is the cycle consistency, i.e., the composition of matches along a loop of images should be identity, as illustrated in Figure 1. Given pairwise matches, one can possibly identify true or false matches by checking all cycles in the image collection. But there are many difficulties for this approach [10]. For example, the input pairwise matches are often very noisy with many false matches and missing matches, and the features detected from different images may only have a partial overlap even if the same feature detector is applied [27]. Therefore, it is likely that very few consistent cycles can be found. Moreover, how to sample cycles is not straightforward due to the huge number of possibilities [16]. Recent work on joint matching has shown that, if all feature correspondences within multiple images are denoted by a large binary matrix, the cycle consistency can be translated into the fact that such a matrix should be positive semidefinite and low-rank [18, 29, 16]. Based on this observation, convex optimization-based algorithms were proposed, which achieved the state-of-the-art performances with theoretical guarantees [16, 10]. But these algorithms rely on semidefinite programming (SDP), which is not computationally efficient to handle image matching problems in practice.

In this paper, we propose a novel algorithm for multi-image matching. The inputs to our algorithm are original similarities between feature descriptors such as SIFT descriptors [25] and deep features [35], or optimized affinities provided by existing graph matching solvers [22]. The outputs are feature correspondences between all pairs of images. Unlike many previous methods starting from quantized pairwise matches [29, 10], we postpone the decision until we optimize for both pairwise affinities and multi-image consistency. Instead of using SDP relaxation, we formulate the problem as a low-rank matrix recovery problem and employ the nuclear-norm relaxation for rank minimization (Section 4.1). We show that the positive semidefiniteness of a desired solution can be spontaneously fulfilled (Section 4.2). Moreover, we derive a fast alternat-
ing minimization algorithm to globally solve the problem
in the low-dimensional variable space (Section 5). Besides
validating our method on both simulated and real bench-
mark datasets, we also demonstrate the applicability of the
proposed method combined with deep learning and graph
matching to match images with different objects and recon-
struct category-specific object models (Section 6).

2. Related work

The early work on joint matching aimed to select cycle-
consistent matches and identify incorrect matches from bad
cycles [41, 28]. The assumption for this family of meth-
ods is that correct matches are dominant in the raw in-
put. Otherwise, it is difficult to find a sufficient number
of closed cycles [16]. Some works proposed to use the cy-
cle consistency as an explicit constraint for sparse feature
matching [38, 37, 39, 36] or pixel-wise flow computation
[42], but the resulting optimization problems are noncon-
vex and can hardly be solved globally. Recent results in
[18, 16, 29] showed that the consistent matches could be ex-
tracted from the spectrum (top eigenvectors) of the matrix
composed of all pairwise matches. The rationale behind this
spectral technique is that the problem can be formulated as
a quadratic integer program and relaxed into a generalized
Rayleigh problem. But the relaxation assumes full feature
 correspondences (bijection) between images [29]. Recently,
Huang and Guibas [16] proposed an elegant solution based
on convex relaxation and derived the theoretical conditions
for exact recovery. The result is further improved in [10]
by assuming that the underlying rank of the variable matrix can
be reliably estimated. In these works, the problem is formu-
lated as SDP, which has a limited computational efficiency
in real applications.

Regarding methodology, our work is inspired by the re-
cent advances on low-rank matrix recovery which make use
of convex relaxation [8, 7] and explore the underlying low-
rank structure to accelerate computation [5, 15]. Our work
is also related to some other problems that aim to find global
estimates from pairwise estimates such as rotation averaging
[14, 34] and model fusion [40].

3. Preliminaries and notation

Suppose we have \( n \) images and \( p_i \) features from each
image \( i \). The objective is to find feature correspondences
between all pairs of images. Before introducing the pro-
posed method, we first give a brief introduction to pairwise
matching techniques and the definition of cycle consistency.

3.1. Pairwise matching

To match an image pair \((i, j)\), one can compute similari-
ties for all pairs of feature points from two images and store
them in a matrix \( S_{ij} \in \mathbb{R}^{p_i \times p_j} \).

We represent the feature correspondences for image pair
\((i, j)\) by a partial permutation matrix \( X_{ij} \in \{0, 1\}^{p_i \times p_j} \),
which satisfies the doubly stochastic constraints:

\[
0 \leq X_{ij} 1 \leq 1, \quad 0 \leq X_{ij}^T 1 \leq 1.
\] (1)

To find \( X_{ij} \), we can maximize the inner product between
\( X_{ij} \) and \( S_{ij} \) subject to the constraints in (1) resulting in
a linear assignment problem, which has been well studied
and can be efficiently solved by the Hungarian algorithm.

In image matching, spatial rigidity is usually preferred,
i.e., the relative location between two features in an im-
age should be similar to that between their correspondences
in the other image. This problem is well known as graph
matching and formulated as a quadratic assignment prob-
lem (QAP). While QAP is NP-hard, many efficient algo-
rithms have been proposed to solve it approximately, e.g.,
[22, 1, 11]. Those solvers basically relax the binary con-
straint on the permutation matrix, solve the optimization,
and output the confidence of a candidate match being cor-
rect. We refer readers to the related literature for details.
Here we aim to emphasize that the outputs of graph match-
ing solvers are basically optimized affinity scores of can-
didate matches, which consider both feature similarity and
spatial rigidity. We will use these scores (saved in \( S_{ij} \)) as
our input in some cases.

3.2. Cycle consistency

Some recent work proposed to use the cycle consist-
cy as a constraint to match a bunch of images [29, 37, 10]. The cycle consistency can be described by

\[
X_{ij} = X_{iz} X_{zj}, \quad (2)
\]

for any three images \((i, j, z)\) and can be extended to the case
with more images.

The recent results in [16, 29] show that the cycle con-
sistency can be described more concisely by introducing a
virtual "universe" that is defined as the set of unique fea-
tures that appear in the image collection. Each point in the
universe may be observed by several images and the cor-
responding image points should be matched. In this way,
consistent matching should satisfy \( X_{ij} = A_i A_j^T \), where
\( A_i \in \{0, 1\}^{p_i \times k} \) denotes the map from Image \( i \) to the uni-
verse, \( k \) is the number of points in the universe, and \( k \geq p_i \)
for all \( i \).

Suppose the correspondences for all \( m = \sum_{i=1}^n p_i \) features in the image collection is denoted by \( X \in \{0, 1\}^{m \times m} \):

\[
X = \begin{pmatrix}
X_{11} & X_{12} & \cdots & X_{1n} \\
X_{21} & X_{22} & \cdots & X_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
X_{n1} & \cdots & \cdots & X_{nn}
\end{pmatrix}, \quad (3)
\]
and all $A_i$s are concatenated as rows in a matrix $A \in \{0,1\}^{m \times k}$. Then, one can write $X$ as

$$X = AA^T,$$  
(4)

From (4), it is clear to see that a desired $X$ should be both positive semidefinite and low-rank:

$$X \succeq 0, \quad \text{rank}(X) \leq k.$$  
(5)

Using (5) the cycle consistency can be effectively imposed without checking all cycles of pairwise matches. Moreover, partial matching is allowed, while bijection needs to be assumed in (2).

4. Joint matching via rank minimization

Given affinity scores $\{S_{ij} \mid 1 \leq i, j \leq n\}$, we aim to find globally consistent matches $X$. Note that $S_{ij}$ can be all-zero if matching is not performed for a pair $(i,j)$. Moreover, affinity scores can be computed from either feature similarities or graph matching solvers according to specific scenarios, as described in Section 3.1.

4.1. Formulation

We formulate the problem as a low-rank matrix recovery problem. We maximize the inner product between $X_{ij}$ and $S_{ij}$ for all $i$ and $j$ as multiple linear assignment problems. At the same time, we minimize the rank of $X$ to enforce the cycle consistency. We ignore the positive semidefinite constraint on $X$ and will explain the reasons later.

To make the optimization tractable, we make the following relaxations: (1) $X$ is treated as a real matrix $X \in [0, 1]^{m \times m}$ instead of a binary matrix, which is a general practice in solving matching problems. Experimentally, we found that the solution values were very close to 0 or 1 and could be stably quantized by a threshold of 0.5. This might be attributed to the existence of a linear term in the cost function [26], (2) Rank of $X$ is replaced by the nuclear norm $\|X\|_*$ (sum of singular values), which is a tight convex relaxation proven to be very effective in various low-rank problems such as matrix completion [8] and robust principal component analysis [7].

The estimated $X$ should be sparse since at most one value in each row of $X_{ij}$ can be nonzero. To induce sparsity, we minimize the sum of values in $X$. Combining all three terms, we obtain the following cost function:

$$f(X) = -\sum_{i=1}^{n} \sum_{j=1}^{n} \langle S_{ij}, X_{ij} \rangle + \alpha \langle 1, X \rangle + \lambda \|X\|_*,$$

$$= -\langle S - \alpha 1, X \rangle + \lambda \|X\|_*,$$  
(6)

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $S \in \mathbb{R}^{m \times m}$ is the matrix collecting all $S_{ij}s$. $\alpha$ is the weight of sparsity, which can be interpreted as a threshold to remove small scores in $S_{ij}s$. In our implementation, we normalize the scores to let them lie between 0 and 1 and empirically set $\alpha = 0.1$. $\lambda$ controls the weight of the nuclear norm. We will discuss $\lambda$ in Section 4.2 and Section 6.1.2.

Besides the doubly stochastic constraints in (1), additional constraints shall be imposed on $X$ after relaxation:

$$X_{ii} = I_{p_i}, \quad 1 \leq i \leq n,$$  
(7)

$$X_{ij} = X_{ji}^T, \quad 1 \leq i, j \leq n, i \neq j,$$  
(8)

$$0 \leq X \leq 1,$$  
(9)

where (7) constrains self-matching to be identity, (8) constrains $X$ to be symmetric, and (9) constrains the values in $X$ to lie in $[0,1]$.

Finally, we obtain the following optimization problem:

$$\min_{X} \langle W, X \rangle + \lambda \|X\|_*,$$

s.t. $X \in C,$  
(10)

where $W = \alpha 1 - S$ and $C$ denotes the set of matrices satisfying the constraints given in (1), (7), (8) and (9).

Upon our experimental observation, the result doesn’t degrade noticeably when removing the doubly stochastic constraints in (1). This might be attributed to the existence of the sparsity regularizer. Therefore, we remove (1) in implementation to accelerate the computation.

4.2. Positive semidefiniteness

We ignore the positive semidefinite constraint for two reasons: (1) solving SDP is generally unscalable; (2) with the constraints in (7) and (8), the solution to (10) turns out to be nearly positive semidefinite if $\lambda$ is sufficiently large.¹

Suppose $\sigma_1, \cdots, \sigma_m$ are eigenvalues of $X$. From (7), we have $X_{ii} = 1$ for all $i$, and $\sum_{i=1}^{m} \sigma_i = \text{trace}(X) = m$, which implies that the sum of $\sigma_i$s is fixed. From (8), we have $X$ is symmetric, and $\sigma_i$s are all real numbers. When we choose a large $\lambda$, $\|X\|_* = \sum_{i=1}^{m} |\sigma_i|$ dominates the cost function, and a solution with all nonnegative $\sigma_i$s will give the lowest cost, because $\sum_{i=1}^{m} |\sigma_i| \geq \sum_{i=1}^{m} \sigma_i = m$ and the equality holds iff. $\sigma_i \geq 0$ for all $i$.

The boundness $\|X\|_* \geq m$ also implies that the solution to (10) will be insensitive to $\lambda$ when $\lambda$ is sufficiently large, and then minimizing the nuclear norm is equivalent to adding a positive semidefinite constraint. The effect of $\lambda$ is experimentally illustrated in Section 6.1.2.

¹We use the term “nearly positive semidefinite” to refer to the property that the negative eigenvalues of a matrix, if there exist, are negligible compared to the norm of the matrix.
Moreover, with the following equation \[ \min_{A, B} \langle W, A B^T \rangle + \lambda \frac{1}{2} \| A \|_F^2 + \lambda \frac{1}{2} \| B \|_F^2, \]
\[ \text{s.t. } A B^T \in C. \] (11)

Moreover, with the following equation \[ \| X \|_* = \min_{A: A B^T = X} \frac{1}{2} \left( \| A \|_F^2 + \| B \|_F^2 \right), \] (12)
we finally obtain the following formulation:
\[ \min_{A, B} \langle W, A B^T \rangle + \lambda \frac{1}{2} \| A \|_F^2 + \lambda \frac{1}{2} \| B \|_F^2, \]
\[ \text{s.t. } A B^T \in C. \] (13)

The selection of matrix dimension \( k \) is critical to the success of change of variables, while it directly affects the computational complexity. We will first provide the algorithm, analyze its complexity and then discuss the selection of \( k \).

5.2. Algorithms

The problem in (13) is not straightforward to solve due to the constraint on the product of variables. Instead, we rewrite the problem as
\[ \min_{X : A B^T} \langle W, X \rangle + \lambda \frac{1}{2} \| A \|_F^2 + \lambda \frac{1}{2} \| B \|_F^2, \]
\[ \text{s.t. } X = A B^T, \ X \in C, \] (14)

and apply the ADMM [2] to solve (14).

The augmented Lagrangian of (14) reads:
\[ \mathcal{L}_{\mu} (X, A, B, Y) = \langle W, X \rangle + \lambda \frac{1}{2} \| A \|_F^2 + \lambda \frac{1}{2} \| B \|_F^2 - \frac{\mu}{2} \| X - A B^T \|_F^2 \]
\[ + \langle Y, X - A B^T \rangle + \frac{\mu}{2} \| Y \|_F^2 \] (15)

where \( Y \) is the dual variable and \( \mu \) is a parameter controlling the step size in optimization. We keep the constraint \( X \in C \) since it can be easily handled as we will show later.

Then, the ADMM alternately updates each primal variable by minimizing \( \mathcal{L}_{\mu} \) and updates the dual variable via gradient ascent while fixing all other variables. The overall algorithm is summarized in Algorithm 1.

\[ \text{Algorithm 1: Multi-Image Matching via Alternating Least Squares (MatchALS)} \]

Input: Pairwise affinity scores \( S \)
Output: Globally consistent matches \( X \)
1 randomly initialize \( A \) and \( B, Y = 0 \);
2 \( W = \alpha 1 - S \);
3 while not converged do
4 \( A \leftarrow \left( X + \frac{1}{\mu} Y \right) B \left( B^T B + \frac{1}{\mu} I \right)^{-1} \);,
5 \( B \leftarrow \left( X + \frac{1}{\mu} Y \right) A \left( A^T A + \frac{1}{\mu} I \right)^{-1} \);,
6 \( X \leftarrow \mathcal{P}_C \left( A B^T - \frac{1}{\mu} (W + Y) \right) \);
7 \( Y \leftarrow Y^{+} + \mu \left( X - A B^T \right) \);
8 end
9 quantize \( X \) with a threshold equal to 0.5.

Minimizing \( \mathcal{L}_{\mu} \) over \( A \) turns out to be a regularized least squares problem with a closed-form solution given in Step 4 in Algorithm 1. The update of \( B \) can be solved similarly. The update of \( X \) requires to solve:
\[ \min_{X \in C} \| X - A B^T + \frac{1}{\mu} (W + Y) \|_F^2, \] (16)
and the solution turns out to be a projection to $C$. Since the constraints in $C$ are all linear, the projection can be solved conveniently. We denote the solution by $P_C(\cdot)$ and leave the details to the supplementary material.

5.3. Computational complexity

The time complexity of an iteration in Algorithm 1 is dominated by matrix multiplication that requires $O(m^2k)$ flops\(^2\). We compare it to the state-of-the-art algorithm MatchLift\(^1\), which is based on SDP. The time complexity of an iteration in MatchLift is dominated by the eigenvalue decomposition that requires $O(m^3)$ flops. As $m$ is much larger than $k$, MatchALS has a lower complexity compared to MatchLift. Moreover, matrix multiplication is parallelizable and has been inherently multithreaded in Matlab, while the parallelization of eigenvalue decomposition is an open problem. Both MatchALS and MatchLift are based on ADMM and require similar numbers of iterations to converge upon our observation.

The CPU time for some problem sizes is shown in Table 1. The algorithms are implemented in Matlab and tested on a PC with an Intel i7 3.4GHz CPU and 8G RAM. We also compare the time cost of partial SVD using PROPACK\(^2\), a toolbox widely used to solve large-scale matrix completion problems\(^4\). In partial SVD, only $r$ leading singular vectors are computed, which is much faster than full SVD when $r/m$ is extremely small. But it is not efficient for a relatively large $r$. In our problem, $r$ should be larger than the true rank and we test partial SVD with $r = p$ in Table 1.

5.4. Selection of $k$ and rank reduction

From the previous subsections we see that $k$ determines the complexity of MatchALS and $k$ should be larger than the rank of true solution, i.e. the size of universe. While some spectral techniques have been proposed in previous work for rank estimation\(^5\), we found that the estimation was inaccurate when the input was noisy and incomplete. Fortunately, our solution doesn’t depend on $k$ if $k$ is larger than the underlying true rank (demonstrated later in Figure 3). A heuristic choice is to set $k = 2\hat{r}$, where $\hat{r}$ is a rough estimate of the size of universe.

In real applications, there are likely to be many isolated features in each image which don’t have any correspondence in other images. However, the constraint in (7) implies that every image feature must be matched to a point in the universe. To see this, recall that we hope $X = AA^T$ in (4). If diagonal values of $X$ are all ones, every row of $A$ has a unit norm, which indicates a match to the universe. Therefore, the size of universe is dramatically increased by those isolated features, and consequently a very large $k$ needs to be selected, which severely increases the computation. To address this issue, we loose the constraint in (7) to be

$$\text{trace}(X) = m', \quad \text{off-diagonal values}\{X_{ii}\} = 0, \quad 1 \leq i \leq n, \quad (17)$$

where $m' \leq m$ is a predefined constant. When $m' = m$, (17) is reduced to (7). When $m' < m$, we allow some rows and columns in $X$ to be null, which is most likely to happen for the rows and columns corresponding to the isolated features, since “switching” them off will not lose many affinity scores but be able to reduce the nuclear norm immediately. By using such a “rank reduction” strategy, the algorithm can automatically prune the isolated features and reduce the size of universe, which enables us to select a smaller $k$ for better computational efficiency. We set $m' = m$ in simulation since there is no isolated feature and $m' = 0.7m$ in real experiments.

6. Experiments

6.1. Simulation

We evaluate the performance of the proposed method using synthesized data. Given a permutation matrix $X$ and the ground truth $X^*$, we measure the error rate by intersection over union:

$$1 - \frac{\left|\tau(X) \cap \tau(X^*)\right|}{\left|\tau(X) \cup \tau(X^*)\right|}, \quad (18)$$

where $\tau$ denotes the matches defined by a permutation matrix and $|\cdot|$ means the size of a set.

6.1.1 Matching errors

We follow the settings in\(^6\) to evaluate the performance of MatchALS and compare it to alternative methods. The size of universe is fixed as 20 points and in each image a random sample of the points are observed with a probability denoted by $\rho_o$. The number of images is denoted by $n$. Then, the ground-truth pairwise matches are established, and random corruptions are simulated by removing some true matches and adding some false matches to achieve an error rate of $\rho_c$. Finally, the corrupted permutation matrix is fed into Algorithm 1 as the input affinity scores.

We evaluate the performance of MatchALS under various $\rho_o$, $\rho_c$ and $n$. We compare MatchALS to two related methods: MatchLift\(^1\) and the spectral method\(^5\). Both of the alternative methods require to know the size of universe and we provide the true value $r^* = 20$. For MatchALS parameters, we set $k = 2r^*$ and $\lambda = 50$.

The output error rates under various settings are shown in Figure 2. When the number of images is sufficiently large, all methods can achieve nearly exact recovery even
In each dataset, to obtain the optimized joint matching, MatchALS and MatchLift perform similarly and outperform the spectral method especially when the observation ratio is small. Compared to MatchLift, the proposed method obtains a competitive performance without exactly knowing the true rank and requires much less computation time.

### 6.1.2 Sensitivity to parameters

The sensitivity of MatchALS to the parameters in (13) is illustrated in Figure 3. The figure shows that MatchALS is insensitive to the predefined dimension of factor matrices $k$ when $k$ is larger than the true rank $r^*$, as we explained in Section 5.1. When $k < r^*$, the problem in (13) is no longer equivalent to the original convex problem in (10), and consequently the alternating minimization fails. In practice, we choose $k = 2 \hat{r}$ as a compromise between safety and efficiency. The right panel in Figure 3 illustrates that the algorithm is insensitive to $\lambda$ when $\lambda$ is sufficiently large as we explained in Section 4.2. In all our experiments, we set $\lambda = 50$.

### 6.2. Real experiments

#### 6.2.1 Graffiti datasets

We evaluate the performance of our algorithm on six benchmark datasets from the Graffiti datasets\(^3\). In each dataset, there are six images of a scene with various image transformations such as viewpoint change, blurring, illumination variation, etc.

We detect 1000 affine covariant features [27] with SIFT [25] descriptors from each image using the VLFeat library [32]. For each image pair $(i, j)$, we compute the inner products between feature descriptors as affinity scores and only keep the scores larger than 0.7 and collect them in $S_{ij}$. If the ratio between the first and the second largest scores in a row/column is smaller than 1.1, we set all scores in this row/column to be zero in order to remove indistinctive features. After computing all $S_{ij}$, we remove the features that have candidate matches in less than two images since they have no contribution to joint matching. Finally, we input the affinity scores to Algorithm 1 to obtain the optimized joint matches.

For evaluation, we adopt the metric used in [10]: for a testing point in an image, we calculate the distance between its estimated correspondence and the true correspondence in another image. If the distance is smaller than a threshold, we regard that a correct match is found for this testing point. Then, we plot the percentages of testing points with correct matches versus the threshold values and obtain a curve analogous to a precision-recall curve. If a testing point is not aligned with any detected point, its estimated correspondence and the true correspondence are varying, while the observation ratio is small. Compared to MatchLift, the proposed method obtains a competitive performance without exactly knowing the true rank and requires much less computation time.

\(^3\)http://www.robots.ox.ac.uk/~vgg/data/data-aff.html
Figure 4. The performance curves on the Graff, Bikes and Light datasets. The y-axis shows the percentages of correct matches. The x-axis shows the distance threshold over the image width. Please see Section 6.2.1 for details. Four methods are compared: MatchALS, MatchLift \([10]\), the spectral method \([29]\), and the original pairwise matching. The areas under curves for all six datasets are given in Table 2.

![Performance curves](image)

Figure 5. The matches between the 1st and the 4th images on the Graff, Bikes and Light datasets. Best viewed in color. The true matches and false matches are shown in yellow and blue, respectively. The top and bottom rows correspond to the results of pairwise matching and joint matching by MatchALS, respectively.

![Match examples](image)

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>MatchALS</th>
<th>Spectral</th>
<th>MatchLift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graffiti</td>
<td>60.2%</td>
<td>87.3%</td>
<td>75.6%</td>
<td>80.6%</td>
</tr>
<tr>
<td>Bikes</td>
<td>76.8%</td>
<td>94.3%</td>
<td>86.7%</td>
<td>92.5%</td>
</tr>
<tr>
<td>Boat</td>
<td>86.2%</td>
<td>93.9%</td>
<td>87.7%</td>
<td>91.7%</td>
</tr>
<tr>
<td>Light</td>
<td>76.0%</td>
<td>93.9%</td>
<td>90.0%</td>
<td>94.0%</td>
</tr>
<tr>
<td>Bark</td>
<td>71.7%</td>
<td>92.2%</td>
<td>91.2%</td>
<td>90.0%</td>
</tr>
<tr>
<td>UBC</td>
<td>88.0%</td>
<td>97.0%</td>
<td>92.9%</td>
<td>96.8%</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
<td>85.8</td>
<td>86.4</td>
<td>2518.4</td>
</tr>
</tbody>
</table>

Table 2. The matching scores and the average computation time (seconds) on the Graffiti datasets. The score is calculated as the area under the curve shown in Figure 4.

a remarkable speedup (∼30 times on average) compared to MatchLift.

We select three image pairs to visually demonstrate the effect of joint matching in Figure 5. A match with a deviation less than five pixels from the ground truth is declared as true. Clearly, the joint matching can prune the false matches (fewer blue lines), complete some missing matches (denser yellow lines), and achieve almost correct matching for these image pairs with large disparities in viewpoints, blurring and illumination changes.

6.2.2 Matching different objects

Recent years have witnessed growing interest in reconstructing category-specific object models from single images, which is still an open problem \([33, 9]\). Among a series of challenges, feature matching for different object instances is the foremost and previous work usually assumed that correspondences of some keypoints were given \([33, 9]\). In this section, we demonstrate the applicability of joint matching to solve this problem.

We use the FG3DCar datasets \([23]\) and try to match the images of different car models in the same category (e.g., sedan or SUV). Following the general practice in object reconstruction \([33, 9]\), we assume segmentation is provided such that background can be ignored, and we only match images with similar views. We select nine sedans and eight SUVs and match two sets of images separately. Note that the car models are all different from each other. See Figure 6 for examples.

To extract descriptive features we first detect image edges by the structured forests \([13]\) and sample a number of points on the edges with constant spacing. On average, we obtain ∼600 feature points for each image. Since the object appearance is changed from image to image and the features are automatically extracted, the matching is extremely difficult. Inspired by recent works \([35, 9]\), we adopt...
Figure 6. Matching different cars. Best viewed in color. Left: the correspondences of sedan images and the reconstruction. Right: the correspondences of SUV images and the reconstruction. Only four selected images are shown for each image set. Note that the cars in images are all different and the feature points are automatically detected. The markers with the same color indicate the matched feature points. The 3D reconstruction is rendered with the colors in the first image and visualized in two viewpoints.

Figure 7. The performance curve of car image matching. “Deep” represents deep features. “GM” denotes graph matching. “Joint” means joint matching using the proposed method.

deep features, i.e., middle-layer responses of convolutional neural nets (CNN), as descriptors for feature matching. More specifically, we use the publicly available deep learning toolbox Caffe [17] and the pre-trained CNN Alexnet [19]. We feed a $192 \times 192$ patch around each feature point forward through the Alexnet. The center columns of conv4 and conv5 layers are concatenated and normalized to form a 640 dimensional feature vector. To leverage the prior on object rigidity, we use pairwise graph matching solved by the Reweighted Random Walk algorithm [11] and collect the output scores of candidate matches as affinity scores. Then, we delete the points with candidate matches in less than two images and run MatchALS.

We adopt the same metric introduced in Section 6.2.1 for quantitative evaluation and use the manually-annotated landmarks provided in the datasets as ground truth. The result on the sedan images is shown in Figure 7. Matching with SIFT features fails since local image patterns are different for two cars. Graph matching with deep features obtains a much better performance, which is further improved by the proposed joint matching algorithm. We obtain a very similar result on the SUV images, which is not plotted.

The results are visualized in Figure 6. The corresponding parts of cars are basically matched in spite of the large differences in appearances and viewpoints. Note that the features are automatically detected and therefore not fully overlapped for two images. For a simple demonstration, we run rigid reconstruction from the estimated feature correspondences by triangulation with an orthographic camera model and the viewpoints provided in the dataset. Despite some noises and missing points, we can clearly see the 3D structures of a sedan and a SUV. We believe that more appealing reconstructions can be obtained by using sophisticated reconstruction techniques and more information such as object silhouette and surface smoothness, while they are out of the scope of this paper.

7. Conclusion

In this paper, we proposed a practical solution to multi-image matching. We use pairwise feature similarities or graph matching scores as input and obtain accurate matches by an efficient algorithm that globally optimizes for both feature affinities and cycle consistency of matches. The experiments not only validate the effectiveness of the proposed method but also demonstrate that joint matching is a promising approach to matching images with different object instances as the first step towards reconstructing object models from crowd-sourced image collections. As future work, we would like to explore more applications and incremental algorithms for joint matching.

Acknowledgments: Grateful for support through the following grants: NSF-DGE-0966142, NSF-IIS-1317788, NSF-IIP-1439681, NSF-IIS-1426840, ARL RCTA W911NF-10-2-0016, and ONR N000141310778.
References


