Practical and Efficient Multi-View Matching

Eleonora Maset, Federica Arrigoni, Andrea Fusiello
DPIA - University of Udine
Via delle Scienze, 208 - Udine (Italy)
maset.eleonora@spes.uniud.it, arrigoni.federica@spes.uniud.it, andrea.fusiello@uniud.it

Abstract

In this paper we propose a novel solution to the multi-view matching problem that, given a set of noisy pairwise correspondences, jointly updates them so as to maximize their consistency. Our method is based on a spectral decomposition, resulting in a closed-form efficient algorithm, in contrast to other iterative techniques that can be found in the literature. Experiments on both synthetic and real datasets show that our method achieves comparable or superior accuracy to state-of-the-art algorithms in significantly less time. We also demonstrate that our solution can efficiently handle datasets of hundreds of images, which is unprecedented in the literature.

1. Introduction

Establishing correspondences between feature sets is a fundamental problem in computer vision, that lies at the basis of any geometric computation (e.g., structure from motion) and also object recognition and shape analysis. In this paper we consider the case in which features are extracted from a collection of images.

The majority of the works on this topic focus on finding correspondences between two feature sets [14, 11, 13, 12, 15, 20]. However, in many tasks it is often required to find matches across multiple views. Moreover, recent studies have suggested that jointly optimizing the correspondences across the whole dataset can lead to significant improvements when compared to computing matches between pairs of views in isolation [19, 31], since pairwise matching algorithms can generate noisy and unreliable results.

As a matter of fact, appearance and geometry alone cannot guarantee the correctness of the matches, hence all one can do is to resort to higher level constraints that arise from the closed-loop consistency of matching across multiple views. This is called *joint matching* or *multi-view matching* by some authors.

A natural approach to joint matching consists in operating directly in the feature space, namely optimizing a cost function which explicitly depends on the features extracted in all the images. Early solutions of this type include the methods presented in [21, 22, 16, 25]. More recently, the authors of [6] cast multi-view matching to an image indexing problem, while in [3] a game-theoretical approach is adopted and the matching problem is expressed as a non-cooperative game. Rank constraints are introduced in [18] for point matching across video frames, and this approach is extended in [30, 8], where the joint matching problem is robustly formulated as a low-rank and sparse matrix decomposition.

A different approach is adopted in [19, 7, 2, 31, 28] where multi-view matching is solved in two steps: first, matching between pairs of images is performed in isolation; then, such correspondences are improved by globally optimizing their internal coherence, without relying on the actual value of the features. In this paper we concentrate on these methods, for they are faster and less memory-demanding than feature-based ones.

A key concept in this context is that of *cycle consistency*, namely the composition of pairwise matches along any loop should give the identity. This property is exploited in several algorithms to remove outliers among pairwise correspondences [29, 17, 9]. However, in practice, a few number of consistent cycles may be found due to noise, and considering all the cycles is computationally intractable. It is shown in some recent works [19, 7, 31] that, if all the pairwise correspondences are collected in a block-matrix, then cycle consistency can be reduced to the requirement that such a matrix is positive semidefinite and low-rank. The authors of [19] express multi-view matching as a synchronization problem, which is approximately solved via spectral decomposition. In [7] a solution based on semidefinite programming is proposed, which, however, assume total feature correspondences between all images. Such technique is extended in [2] in order to handle partial correspondences, and theoretical guarantees for exact matching in the presence of corrupted input are provided, assuming a certain noise model. In [31] the joint matching problem is formulated as a low-rank matrix recovery task and the nuclear-norm relaxation for rank minimization is employed. The resulting cost function is optimized via the Alternating Direction Method of Multipliers (ADMM). Finally, in [28] a solution based on the proximal Gauss-Seidel method is provided, which, as [7], assumes total correspondences between all the images, thus limiting its applicability to real scenarios.

In this paper we propose a novel closed-form method for multi-view matching, called MATCHEIG, which proceeds as follows: first, the top eigenvectors of the block-matrix containing pairwise correspondences are computed; then, the eigenvectors are projected onto permutations to yield the output pairwise matches. The resulting method is extremely simple and it can be coded in a few lines of Matlab. Its accuracy is comparable or superior to the state of the art, as shown by synthetic and real experiments, and it is significantly faster than all the competing methods. Thanks to its computational efficiency it successfully handles sets of hundreds of images, where others fail to produce a solution.

The proposed method share the same framework as [19], since it is based on a spectral decomposition. Differences with respect to [19] are detailed in Sec. 3, and they imply a significant improvement in performance, as demonstrated by the experiments.

The paper is organized as follows. Sec. 2 formally defines the problem and in Sec. 3 the proposed method is described. Sec. 4 presents experiments on both synthetic and real data and analyzes the results. Finally, Sec. 5 draws the conclusions.

2. Problem formulation

The goal of multi-view matching is to establish feature correspondences between all pairs of images. Let n denote the number of images and let m_i denote the number of features in image i. The correspondences between the features in image j and those in image i can be represented as a *partial permutation matrix* $P_{ij} \in \{0,1\}^{m_i \times m_j}$ constructed as follows: $[P_{ij}]_{h,k} = 1$ if feature k in image j is matched with feature h in image i; $[P_{ij}]_{h,k} = 0$ otherwise. If row $[P_{ij}]_{h,\cdot}$ is a row of zeros, then feature h in image i does not have a matching feature in image j. If column $[P_{ij}]_{\cdot,k}$ is a column of zeros, then feature k in image j does not have a matching feature in image i. A partial permutation matrix has at most one nonzero entry in each row and column, and these nonzero entries are all 1. If exactly one entry in each row and column is equal to 1 (and all other entries are 0), then the permutation is *total*. Partial permutations are suitable to model matches in practical scenarios, since they can represent missing correspondences, whereas the usage of total permutations requires that the same set of features is present in all the images, which is an unrealistic assumption.

Let us assume that all the features belong to a *universe* set. Let $P_i \in \{0,1\}^{m_i \times d}$ denote the partial permutation matrix representing the correspondences between the features in image i and those in the universe, where d denotes the size of the universe. In the absence of noise, the correspondences between image j and image i can be equivalently represented by first computing the matches between image j and the universe, and then from the universe to image i, namely

$$P_{ij} = P_i P_j^{\mathsf{T}}. \tag{1}$$

Equation (1) is called the *consistency constraint*. The matrix P_{ij} is referred to as the *relative* permutation of the pair (i,j), and the matrix P_i (resp. P_j) is referred to as the *absolute* permutation of image i (resp. j). According to Eq. 1, the solution to the multi-view matching problem can also be achieved by first computing n absolute permutations $P_1, \dots, P_n$ and then setting $P_{ij} = P_i P_j^{\mathsf{T}}$.

2.1. Spectral properties

As observed in [19, 7, 31], the consistency constraint can be expressed in a compact matrix form if all the absolute and relative permutations are collected in two block-matrices $X \in \{0,1\}^{m \times d}$ and $Z \in \{0,1\}^{m \times m}$ respectively, where $m = \sum_{i=1}^{n} m_i$, namely

$$X = \begin{bmatrix} P_1 \\ P_2 \\ \dots \\ P_n \end{bmatrix}, \quad Z = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \dots & & & \dots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{bmatrix}. \tag{2}$$

Note that Z may contain zero blocks: if all the features in image i do not match with any feature in image j, then $P_{ij} = 0$. Using this notation, Eq. (1) becomes

$$Z = XX^{\mathsf{T}} \tag{3}$$

which implies that Z is symmetric positive semidefinite and has rank d.

Proposition 1. *[19, 26] The columns of X are d (orthogonal) eigenvectors of Z corresponding to the eigenvalue n, assuming that all the permutations are total.*

Proof. In the case of total permutations we have $P_i^{\mathsf{T}} P_i = I_d$ and hence $X^{\mathsf{T}} X = nI_d$, where I_d denotes the $d \times d$ identity matrix. Thus $ZX = XX^{\mathsf{T}}X = nX$, which means that the columns of X are d eigenvectors of Z corresponding to the eigenvalue n. □

The following new result generalizes Prop. 1 to the case of partial permutations.

Proposition 2. *The columns of X are d (orthogonal) eigenvectors of Z and the corresponding eigenvalues are given by the diagonal of $V := \sum_{i=1}^{n} P_i^{\mathsf{T}} P_i$.*

Proof. In the case of partial permutations, P_i does not have an inverse, thus the $d \times d$ (diagonal) matrix $P_i^{\mathsf{T}} P_i$ is not equal to the identity: $[P_i^{\mathsf{T}} P_i]_{k,k} = 1$ if the k-th feature in the universe is present in image i; $[P_i^{\mathsf{T}} P_i]_{k,k} = 0$ otherwise. Define the $d \times d$ diagonal matrix $V := X^{\mathsf{T}} X = \sum_{i=1}^{n} P_i^{\mathsf{T}} P_i$. Then

$$ZX = XV \tag{4}$$

which is a spectral decomposition, i.e., the columns of X are d eigenvectors of Z and the corresponding eigenvalues are given by the diagonal of V. Specifically, the k-th eigenvalue is an integer which counts how many images match the k-th feature in the universe. $\square$

Note that in the case of total permutations all the features are present in all the images, therefore $V = nI_d$ and all the eigenvalues are equal, hence we get Prop. 1. Since Z has rank d, the matrix V contains the *largest* eigenvalues of Z and all the other eigenvalues are zero. Thus, in the presence of noise, we can take the eigenvectors of Z corresponding to the d largest eigenvalues as an estimate of X. In Sec. 2.2 we describe the meaning of this procedure in terms of an optimization problem, and in Sec. 3 we show how it can be exploited to derive an efficient method for multi-view matching.

2.2. Optimization problem

In practice, pairwise correspondences contain errors, hence what we measure is an estimate $\widehat{P}_{ij}$ of the relative permutation between image i and image j (in this paper we use the hat accent to denote approximate quantities). The goal is to compute a set of partial permutation matrices $\{P_{ij}\}_{i,j=1}^{n}$ such that the consistency constraint is satisfied and P_{ij} is as close as possible to its measure $\widehat{P}_{ij}$, namely $P_{ij} \approx \widehat{P}_{ij}$ for all $i, j \in \{1, \ldots, n\}$. A possible approach consists in considering the following optimization problem

$$\max_{\{P_{ij}\}_{i,j=1}^{n}} \sum_{i,j=1}^{n} \langle \widehat{P}_{ij}, P_{ij} \rangle \quad \text{s.t. } P_{ij} = P_i P_j^{\mathsf{T}} \tag{5}$$

where each optimization variable is constrained to be a partial permutation matrix. Here $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, i.e. $\langle A, B \rangle = \text{trace}(AB^{\mathsf{T}})$. The cost function in (5) counts, for each image pair (i, j), the number of features equally matched by permutations P_{ij} and $\widehat{P}_{ij}$.

If $\widehat{Z}$ denotes the block-matrix containing the measured relative permutations $\widehat{P}_{ij}$, then Eq. (5) rewrites

$$\max_{Z} \langle \widehat{Z}, Z \rangle = \max_{Z} \text{trace}(\widehat{Z} Z^{\mathsf{T}}) \quad \text{s.t. } Z = XX^{\mathsf{T}} \tag{6}$$

$$\iff \max_{X} \langle \widehat{Z}, XX^{\mathsf{T}} \rangle = \max_{X} \text{trace}(X^{\mathsf{T}} \widehat{Z} X) \tag{7}$$

where X is constrained to be composed of partial permutation matrices. Maximizing the objective function in Eq. (7) is a challenging task since the feasible set consists of binary variables which makes the problem combinatorially NP-hard. Moreover, optimizing with respect to multiple permutation matrices simultaneously increases the difficulty of the problem. For these reasons, it is common practice to relax some constraints on the optimization variables, thus providing tractable approaches that solve the multi-view matching problem approximately but efficiently. Some examples include the *semidefinite* relaxation [2], the *low-rank* relaxation [31] and the *spectral* relaxation [19]. The SPECTRAL method of [19] treats X as a real matrix instead of a binary matrix and enforces the columns of X to be orthogonal, resulting in the following optimization problem

$$\max_{U^{\mathsf{T}} U = I_d} \text{trace}(U^{\mathsf{T}} \widehat{Z} U) \tag{8}$$

where the notation U instead of X is used to underline that, due to the relaxation, the optimal U will not be composed of partial permutation matrices. Equation (8) is a generalized Rayleigh problem, whose solution is given by the d leading eigenvectors of $\widehat{Z}$. In order to obtain proper correspondences from U, each $m_i \times d$ block is projected onto the nearest permutation matrix via the Kuhn-Munkres algorithm [10], which solves a linear assignment problem, thus returning a set of estimated absolute permutations.

3. Our method

The SPECTRAL method is extremely fast, as multi-view matching is solved in one shot via spectral decomposition. However, since *absolute* permutations are computed, the knowledge of the size of the universe d is required, which is not available in practice. The importance of a correct estimate of d is also demonstrated experimentally in Sec. 4.1.1.

We introduce here a novel technique for multi-view matching, dubbed MATCHEIG, which inherits the positive aspects of the SPECTRAL method, namely efficiency and simplicity, and at the same time it overcomes its drawback, i.e., the need of the correct value of d as input. The key observation is that relative permutations are independent from d, thus a method that aims at producing *relative* permutations instead of absolute ones can get by without knowing precisely d. Specifically, our method proceeds as follows. First, the top d eigenvectors of $\widehat{Z}$ are computed and collected in a $m \times d$ matrix U, as done by SPECTRAL. Let D be the diagonal matrix containing the corresponding d eigenvalues $\lambda_1, \ldots \lambda_d$. The matrix

$$\widehat{Z}_d = UDU^{\mathsf{T}} \tag{9}$$

is the solution of (6) under the spectral relaxation. In this way we get an estimate of Z – which contains *relative* permutations, and this is a key difference with respect to the SPECTRAL method that provides an estimate of X – which contains *absolute* permutations.

Suppose that we are given an estimate $\widehat{d}$ of the size of the universe such that $\widehat{d} \geq d$, and we compute $\widehat{Z}_{\widehat{d}}$ accordingly. Since $\widehat{Z}$ has approximately rank d, we expect that the least $\widehat{d} - d$ eigenvalues $\lambda_{d+1}, \dots \lambda_{\widehat{d}}$ are smaller than the top d eigenvalues, thus the corresponding eigenvectors in U have a limited impact on $\widehat{Z}_{\widehat{d}}$, in particular: $\|\widehat{Z}_d - \widehat{Z}_{\widehat{d}}\|_2 = |\lambda_{d+1}|$.

Note that, due to the relaxation, the $m_i \times m_j$ blocks of $\widehat{Z}_d$ are not guaranteed to be partial permutation matrices. In order to enforce this constraint we analyze two different strategies. A first stage common to both consists in setting to zero all the entries smaller than a given threshold t. We set $t = 0.25$ in simulations and $t = 0.5$ in real experiments. A higher threshold allows for more missing matches, and this is useful in real datasets to model the presence of isolated features.

Then, a principled approach consists in projecting each block onto the closest partial permutation matrix via the Kuhn-Munkres algorithm. We call this method MATCHEIG-CP, where CP stands for "closest permutation". This projection, however, slows down the computing time, so in our MATCHEIG algorithm we use a greedy strategy that, if applied to each block, returns a valid permutation, although not the closest one. This strategy, implemented in the function `matrix2perm()`, is approximate but it produces no noticeable loss in accuracy, while greatly boosting the speed, as experiments will demonstrate.

The `matrix2perm()` function takes a matrix C as input and returns a (partial) permutation matrix P constructed as follows: search among the non-zero entries of C for the ones where the maximum over the corresponding row or column is achieved. These entries are then sorted by decreasing magnitude and examined sequentially starting from the largest element: let (i, j) be the index of the current entry, and let P be the output matrix, initialized to 0; then $[P]_{i,j}$ is set to 1 provided that P remains a partial permutation.

The idea behind this procedure is the following. For a given row i, which corresponds to a feature in one image, each entry $[C]_{i,j}$ represents the extent of pairing between feature i and feature j, and the greatest element in this row can be regarded as the most likely correspondence. The same holds for each column. To these putative matches we need to apply the principle of exclusion, and we do it in a greedy way [23]: the strongest match wins and inhibits other 1s to be placed in its row or column.

The Matlab code of MATCHEIG is reported in Algorithm 1. MATCHEIG-CP has a function implementing the Kuhn-Munkres projection instead of `matrix2perm()`. We exploited the sparse eigen-solver (`eigs`) since $\widehat{Z}$ is sparse and only the top $\widehat{d}$ eigenvectors are required.

Note that, because of noise, $\widehat{Z}_{\widehat{d}}$ is full, in general, and its size can become large in practical scenarios. However, this matrix needs not to be explicitly computed, for only one block is needed at a time. Specifically, when an image pair (i, j) is considered, the product $U_i D U_j^{\mathsf{T}}$ need to be computed, where U_i denotes the $m_i \times \widehat{d}$ block in U corresponding to image i and U_j denotes the $m_j \times \widehat{d}$ block in U corresponding to image j. Therefore we only need to store the matrix $UD^{\frac{1}{2}}$ instead of $\widehat{Z}_{\widehat{d}}$, and this observation considerably reduces the storage space necessary to run the algorithm.

Note also that the projection step (either via the Kuhn-Munkres algorithm or via the approximate strategy) can be performed in parallel, since each image pair is independent from the others, thus speeding up the process.

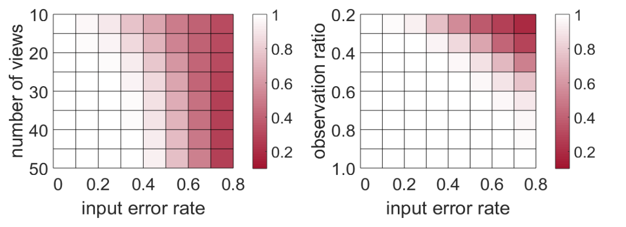
Algorithm 1 MATCHEIG

```
function Zout = MatchEIG(Z,d,n,dimP,t)
% Z: input matches
% d: estimate of universe size
% n: number of images
% dimP: size of each permutation
% t: threshold

cm = [0;cumsum(dimP(1:end-1))];
blk = @(k) 1+cm(k):cm(k)+dimP(k);

% spectral decomposition
[U,D] = eigs(Z,d,'lm');
U=U*sqrt(D);

for i=1:n
    for j=i:n
        Zb = U(blk(i),:)*U(blk(j),:)';
        Zb(Zb<t)=0; % thresholding
        % project onto permutations
        Zout(blk(i),blk(j))=matrix2perm(Zb);
    end
end
end
```

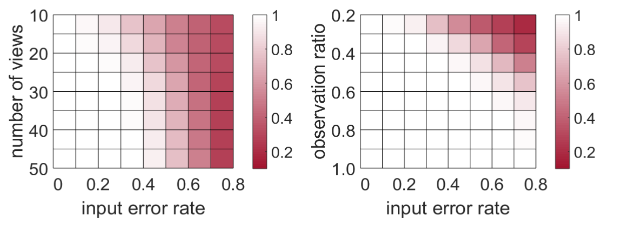
3.1. Computational complexity

The core of the algorithm is the eigenvalue decomposition of a sparse matrix, which is computed with a Lanczos method, implemented by the `eigs` function of Matlab. The method is iterative: every iteration is $O(m)$ [5], but the number of iterations cannot be bounded by a constant.

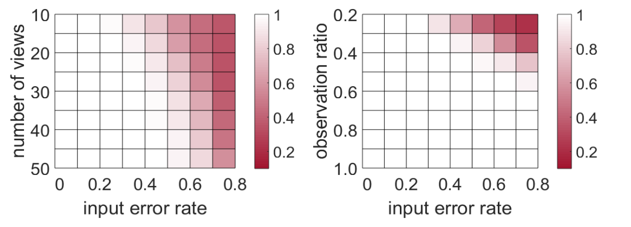
The second and final step of our algorithm is the projection onto permutation matrices. For a matrix of dimension r, computing the nearest permutation via the Kuhn-Munkres algorithm takes $O(r^3)$ time [1]. As for the approximate strategy, we have to sum the cost for computing the maximums over rows/columns, which is $O(r^2)$, and the cost for sorting such values, which is $O(r \log(r))$, resulting in $O(r^2)$. Since the average dimension of each block in $\widehat{Z}$ is m/n and the number of blocks is n^2, the total cost for the projection step is $O(m^3/n)$ for MATCHEIG-CP and $O(m^2)$ for MATCHEIG.



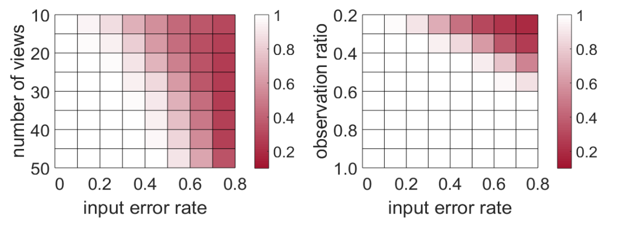
(a) MATCHEIG



(b) MATCHEIG-CP



(c) MATCHALS [31]



(d) SPECTRAL [19]

Figure 1: F-score for the competing methods (the higher the better). In the left column, the number of views n and the input error rate r_e are varying, while the observation ratio r_o is constant and equal to 0.4. In the right column, r_o and r_e are varying, while $n = 30$. In all the experiments, the size of the universe d is set to 100.

4. Experiments

In order to evaluate the performance of the proposed methods, we ran experiments on both synthetic and real datasets. In the synthetic experiments, performances have been measured in terms of *precision* (number of correct matches returned divided by the number of matches returned) and *recall* (number of correct matches returned divided by the number of correct matches that should have been returned). In order to provide a single figure of merit we computed the *F-score* (twice the product of precision and recall divided by their sum), which is a measure of accuracy and reaches its best value at 1 and worst at 0. In the real experiments the number of matches that should have been returned is not known, hence only the precision can be computed.

Results in terms of accuracy (or precision) and computing time are compared with methods which, as ours, first compute pairwise matches and then jointly update them without involving the features, namely MATCHALS [31] and SPECTRAL [19]. The MATCHALS code is available online [1] and the SPECTRAL code is courtesy of the authors of [31]. The method described in [2] has already been shown to have accuracy comparable with [31] with a much higher computing time. For this reason we did not consider it in the experiments. All the algorithms are implemented in Matlab and tested on a PC with an Intel Core i5-4200M CPU @ 2.50GHz and 8GB RAM. Our implementation of MATCHEIG is available on the web[2].

4.1. Synthetic experiments

For the synthetic case, the size of the universe was set to $d = 100$, while the number of views varied from $n = 10$ to $n = 50$. The observation ratio r_o, i.e., the probability that a feature is seen in a view, decreased from 1 (that corresponds to total permutations) to 0.2. After generating ground-truth absolute permutations, pairwise matches were computed from Eq. (3), and random errors were added to relative permutations by switching two matches, removing true matches or adding false ones. In the experiments the input error rate r_e, i.e., the ratio of mismatches, varied from 0 to 0.8. For each configuration the test was run 10 times and the average F-score was evaluated.

Synthetic experiments were run providing the true rank (equal to the size of the universe d) to all the methods (MATCHALS uses it to compute the parameter $k = 2d$, as suggested in [31]). Moreover, for MATCHALS we fixed the number of possible iterations to 100 and we chose the values proposed by the authors for all the other parameters.

Results in terms of accuracy are illustrated in Fig. 1. The analyzed methods show similar behaviors, achieving high accuracy rates even in the presence of high noise contamination, especially when the number of views is large. Please note that there is no loss of accuracy when using the approximated projection onto permutations (MATCHEIG) with respect to the exact closest permutation, as done by MATCHEIG-CP.

We also evaluated the computing time, varying the size

[1] `https://fling.seas.upenn.edu/~xiaowz/dynamic/wordpress/matching/`
[2] `http://www.diegm.uniud.it/fusiello/demo/mvm/`

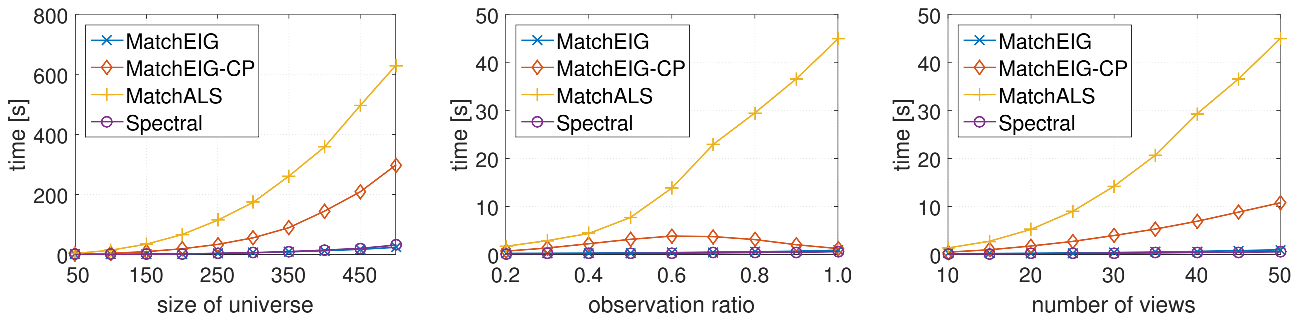


Figure 2: Average computing time against size of the universe, observation ratio and number of views.

of the universe, the observation ratio and the number of views. Figure 2 shows that MATCHEIG and SPECTRAL are the fastest algorithms, while MATCHALS is on average an order of magnitude slower. Comparing MATCHEIG and MATCHEIG-CP, it is clear that the Kuhn-Munkres algorithm is computationally much more expensive than the approximate procedure. Since the accuracy provided by the two projection algorithms is the same, as demonstrated by Fig. 1, only the fastest version (MATCHEIG) will be used in real experiments.

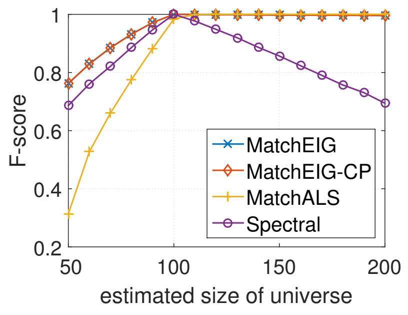


Figure 3: F-score versus estimated size of the universe $\widehat{d}$. The true size of universe is $d = 100$, $n = 30$, $r_o = 0.6$, and $r_e = 0.1$.

4.1.1 Sensitivity to rank estimate

All the evaluated methods require as input an estimate of the size of the universe, which corresponds to the rank of the ground-truth matrix Z. However, when the input matrix $\widehat{Z}$ is noisy, estimating this rank can be difficult [31], hence the sensitivity of a method to the estimated rank $\widehat{d}$ becomes crucial. As demonstrated in Fig. 3, MATCHEIG and MATCHALS give good results whenever $\widehat{d} \geq d$. The SPECTRAL method, instead, is extremely sensitive to this parameter and performs well only if $\widehat{d} = d$.

4.2. Real experiments

For evaluating the applicability of the proposed method in real scenarios, we tested MATCHEIG, MATCHALS and SPECTRAL on popular benchmark datasets (Graffiti, EPFL [27] and Middlebury [24] datasets) with up to 363 images. To generate the input to the algorithms, a set of features was first extracted in each image with SIFT [14] using the VLFeat library[3]. Subsequently, correspondences between pairs of images were established using nearest neighbor and ratio test as in [14] and refined using RANSAC [4]. Finally, features with matches in less than two images were removed, since they are not significant in joint matching.

In these experiments the universe set is not known, so we estimated its dimension as twice the average number of features present in each image, and provided all the methods with this estimate. For the same reason only the precision value is reported, together with the absolute number of correct matches returned, that can give some relative indication on the recall. For MATCHALS, we set the parameter m'/m to 0.7 (see [31]) to take into consideration the presence of isolated features in real images, as suggested by the authors. The maximum number of iterations was fixed to 100 and the default values were used for all the other parameters.

Matches are considered correct if the corresponding point is located within a given distance threshold from what is predicted. In the case of the Graffiti datasets the ground-truth homographies allow to predict the position of the point, whereas for EPFL and Middlebury datasets the ground-truth cameras allow to predict the epipolar line where the corresponding point should lie. In both cases the threshold (in pixels) has been set equal to 0.01 times the image diagonal.

4.2.1 Graffiti dataset

The Graffiti datasets[4] consist of eight sequences with six images each, showing different structured and textured planar scenes. Each dataset is characterized by different image transformations, e.g., change of viewpoint, zoom, blur, illumination and rotation.

Table 1 shows the performances of joint matching on

[3]`http://www.vlfeat.org/`

[4]`http://www.robots.ox.ac.uk/~vgg/data/data-aff.html`

			Input	MATCHEIG			MATCHALS [31]			SPECTRAL [19]		
Dataset	n	$\widehat{d}$	PR [%]	PR [%]	CM	time [s]	PR [%]	CM	time [s]	PR [%]	CM	time [s]
Graffiti	6	382	93.85	95.63	678	12	96.02	747	91	**96.24**	614	24
Boat	6	578	98.75	**99.03**	1731	35	98.63	1154	209	98.66	1544	45
Bark	6	684	99.71	99.77	1323	58	**100.00**	914	307	99.67	1225	77
Ubc	6	891	99.36	99.66	3828	139	**99.67**	3030	681	99.63	3533	207
Trees	6	1015	98.48	98.46	2885	184	**98.73**	2484	971	98.51	2719	255
Light	6	1113	98.67	99.39	4105	336	**99.43**	3287	1258	98.95	2253	416
Wall	6	1236	99.40	99.45	3253	341	99.38	2871	1644	**99.57**	2760	456
Bikes	6	1759	99.12	99.39	4866	954	**99.53**	4228	3828	99.26	4149	1298

Table 1: Results on the Graffiti dataset. n is the number of images, PR is the precision and CM is the number of correct matches returned. Time is in seconds.

			Input	MATCHEIG			MATCHALS [31]			SPECTRAL [19]		
Dataset	n	$\widehat{d}$	PR [%]	PR [%]	CM	time [m]	PR [%]	CM	time [m]	PR [%]	CM	time [m]
Herz-Jesu-P8	8	386	94.40	**95.08**	4545	< 1	94.87	4047	2	94.41	3987	< 1
Entry-P10	10	432	75.11	**79.24**	5978	5	74.17	5726	4	76.10	6236	4
Fountain-P11	11	374	94.35	**94.70**	6988	3	94.15	6717	3	91.92	7849	3
Castle-P19	19	314	70.29	**75.21**	5109	3	66.22	7014	9	34.41	7605	3
Herz-Jesu-P25	25	517	90.20	**93.45**	25120	7	89.23	32528	41	47.86	32876	8
Castle-P30	30	445	72.32	**81.01**	16754	8	68.92	24844	57	34.67	25884	10
Temple Ring	47	396	73.72	**88.25**	18426	6	55.91	40096	260	28.99	46432	7
Dino Ring	48	340	75.37	**92.11**	23406	2	66.66	44215	94	34.49	48979	3
Temple	312	689	55.50	**89.06**	379545	153	–	–	–	14.56	1571708	228
Dino	363	493	63.48	**95.66**	862221	88	–	–	–	18.97	2231890	111

Table 2: Results on the EPFL [27] and Middlebury datasets [24]. n is the number of images, PR is the precision and CM is the number of correct matches returned. Time is in minutes.

these datasets. The input error is already small, and all the methods achieve a precision higher than 95%, with a comparable number of correct matches returned, confirming the results of the synthetic experiments. This dataset is not particularly challenging, as it consists of few images with little differences in visual content among them; however it is widely used for testing multi-view matching algorithms, therefore it has been included here. A consideration can be made regarding the computing time, with MATCHEIG being on average 5 times and 1.5 times faster than MATCHALS and SPECTRAL, respectively.

4.2.2 EPFL dense multi-view stereo test images

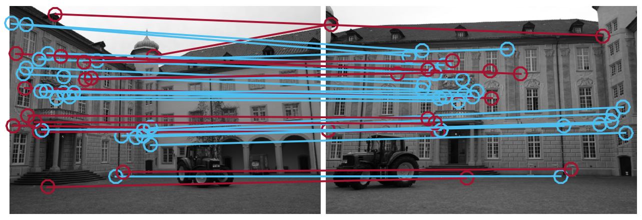
A more challenging benchmark are the EPFL dense multi-view stereo test images [27]. These are six image sets representing outdoor scenes, composed of a number of images that varies from 8 to 30. Performing multi-view matching on the *Entry-P10* and *Castle-P** sequences is particularly difficult due to the presence of repetitive structures. For practical reasons, we rescaled images to 20% of the original size.

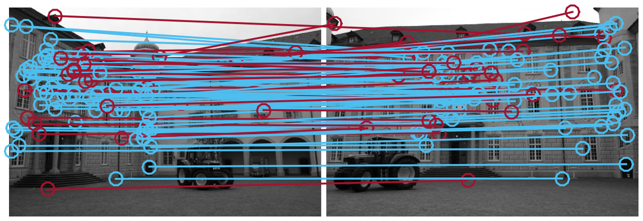
As can be seen in the upper part of Tab. 2, the quality of the input matches is lower than that on the Graffiti datasets and it can be significantly improved by joint matching. On these datasets, MATCHEIG outperforms the other algorithms, both in terms of precision and computing time.

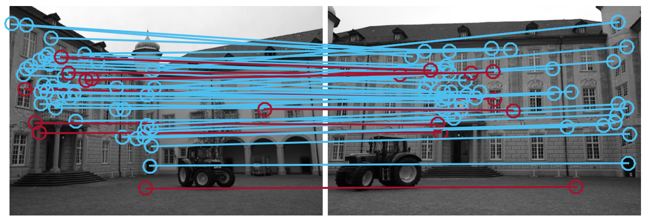
Note that on some sequences SPECTRAL achieves a very low precision, probably due to the fact that the chosen value of $\widehat{d}$ is not an accurate estimate of d. Fig. 4 shows a representative example of the results obtained by the competing methods. With respect to pairwise matching, joint matching reduces the number of false matches and complete the matches with new ones retrieved indirectly via loop closure.

4.2.3 Middlebury multi-view stereo dataset

Multi-view matching often needs to be applied on sets of hundreds of images. Therefore we selected the two largest sets among the Middlebury multi-view stereo datasets [24] to evaluate the practical applicability of MATCHEIG, MATCHALS and SPECTRAL. The *Temple* and the *Dino* sets


(a) Input


(b) MATCHALS


(c) MATCHEIG

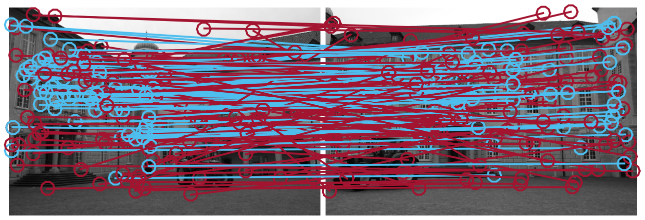

(d) SPECTRAL

Figure 4: (best viewed in color) Representation of the matches between two images of the *Castle-P19* set [27]. Wrong matches are drawn in red.

consist of 312 and 363 views respectively, sampled on a hemisphere. We also considered the smaller *Dino Ring* and *Temple Ring* sets, that contain approximately 50 views each, sampled on a ring around the object.

Results are reported at the bottom of Tab. 2. In these cases, the initial pairwise matching provides a noisy input, upon which only MATCHEIG is able to improve. In fact, on the smaller datasets, MATCHALS and SPECTRAL produced results worse than the input (MATCHALS taking a very long time), while on larger sets MATCHALS did not achieve a solution: due to a memory request that exceeded the available space it aborted prematurely.

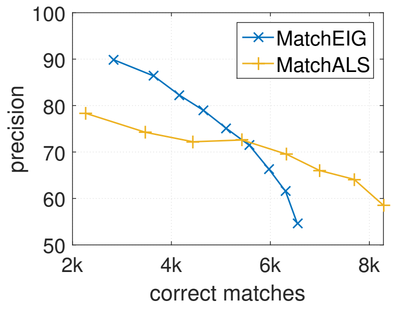


Figure 5: Precision vs CM curve on the *Castle-P19* set [27].

The precision/recall tradeoff in MATCHEIG is affected by the threshold t, which controls the degree of "partiality" of the permutations. A small t yields more matches, but with low precision. On the other hand, a high t produces few but extremely reliable matches. Fig. 5 shows the precision vs CM curve (recall is not available) on a typical dataset: MATCHALS has an almost constant precision for a wide range of the controlling parameter (m'/m), whereas in the left part of the curve MATCHEIG has a superior precision, and this is where we suggest our algorithm should be used. We tuned the value of t empirically so as to be in this part of the curve in all the considered datasets.

This choice is motivated by applications: with respect to the null hypothesis that a match is an inlier, a Type I error (rejecting an inlier) is less serious than a Type II error (not rejecting an outlier) when using matches to compute geometrical models.

5. Conclusions

We presented a closed-form solution to (joint) multi-view matching, based on a spectral decomposition. MATCHEIG handles realistic situations, such as partial permutations and image sets of unprecedented size in the literature. It also enjoys a very compact Matlab implementation.

While experiments on simulated data – and one easy real dataset – only highlight the superior computational efficiency of the method, with the accuracy being on a par with the others, on challenging real datasets MATCHEIG outperforms the competing methods both in speed and precision.

Applications for a practical multi-view matching are countless, and surely include structure from motion, which motivates this study as well as future developments.

Acknowledgments. This work has been supported by GNCS project "DECORE".

References

[1] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802 – 804, 1971.

[2] Y. Chen, L. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. In *International Conference on Machine Learning*, pages 100–108, 2014.

[3] L. Cosmo, A. Albarelli, F. Bergamasco, A. Torsello, E. Rodolá, and D. Cremers. A game-theoretical approach for joint matching of multiple feature throughout unordered images. In *International Conference on Pattern Recognition*, pages 57–60, 2016.

[4] M. A. Fischler and R. C. Bolles. Random Sample Consensus: a paradigm model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[5] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, 1996.

[6] M. Havlena and K. Schindler. VocMatch: Efficient multi-view correspondence for structure from motion. In *European Conference on Computer Vision*, pages 46 – 60, 2014.

[7] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Eurographics Symposium on Geometry Processing*, volume 32, pages 177–186, 2013.

[8] K. Jia, T.-H. Chan, Z. Zeng, S. Gao, G. Wang, T. Zhang, and Y. Ma. ROML: A robust feature correspondence approach for matching objects in a set of images. *International Journal of Computer Vision*, 117(2):173–197, 2016.

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

[10] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 2:83 – 97, 1955.

[11] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision*, pages 1482 – 1489, 2005.

[12] W.-Y. D. Lin, M.-M. Cheng, J. Lu, H. Yang, M. N. Do, and P. Torr. Bilateral functions for global motion modeling. In *European Conference on Computer Vision*, pages 341–356, 2014.

[13] Y. Lipman, S. Yagev, R. Poranne, D. W. Jacobs, and R. Basri. Feature matching with bounded distortion. *ACM Transactions on Graphics*, 33(3):26:1–26:14, 2014.

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

[15] J. Ma, W. Qiu, J. Zhao, Y. Ma, A. L. Yuille, and Z. Tu. Robust L_2E estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing*, 63(5):1115 – 1129, 2015.

[16] J. Maciel and J. P. Costeira. A global solution to sparse correspondence problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):187 – 199, 2003.

[17] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas. An optimization approach to improving collections of shape maps. In *Eurographics Symposium on Geometry Processing*, volume 30, pages 1481–1491, 2011.

[18] R. Oliveira, R. Ferreira, and J. P. Costeira. Optimal multi-frame correspondence with assignment tensors. In *European Conference on Computer Vision*, pages 490–501, 2006.

[19] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems 26*, pages 1860–1868. 2013.

[20] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint probabilistic matching using m-best solutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–145, 2016.

[21] V. Salari and I. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):87 – 91, 1990.

[22] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or “how do I organize my holiday snaps?”. In *European Conference on Computer Vision*, pages 414–431, 2002.

[23] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of London B*, volume 244, pages 21–26, 1991.

[24] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006.

[25] K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51 – 65, 2005.

[26] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30(1):20 – 36, 2011.

[27] C. Strecha, L. J. V. von Hansen, W.and 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*, 2008.

[28] J.-G. Yu, G.-S. Xia, A. Samal, and J. Tian. Globally consistent correspondence of multiple feature sets using proximal Gauss–Seidel relaxation. *Pattern Recognition*, 51:255 – 267, 2016.

[29] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1426 – 1433, 2010.

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

[31] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *International Conference on Computer Vision*, pages 4032 – 4040, 2015.