# A Tensor-Based Algorithm for High-Order Graph Matching

Olivier Duchenne, Francis Bach, In-So Kweon, and Jean Ponce

**Abstract**—This paper addresses the problem of establishing correspondences between two sets of visual features using higher-order constraints instead of the unary or pairwise ones used in classical methods. Concretely, the corresponding hypergraph matching problem is formulated as the maximization of a multilinear objective function over all permutations of the features. This function is defined by a tensor representing the affinity between feature tuples. It is maximized using a generalization of spectral techniques where a relaxed problem is first solved by a multi-dimensional power method, and the solution is then projected onto the closest assignment matrix. The proposed approach has been implemented, and it is compared to state-of-the-art algorithms on both synthetic and real data.

**Index Terms**—Hypergraphs, Graph Matching, Image Feature Matching.

## 1 Introduction

Establishing correspondences between two sets of visual features is a key problem in computer vision tasks as diverse as feature tracking [6], image classification [18] or retrieval [29], object detection [5], shape matching [19], [36], or wide-baseline stereo fusion [27]. Different image cues may lead to very different matching strategies. At one end of the spectrum, geometric matching techniques such as RANSAC [10], interpretation trees [14], or alignment [15] can be used to efficiently explore consistent correspondence hypotheses when the mapping between image features is assumed to have some parametric form (e.g., a planar affine transformation), or obey some parametric constraints (e.g., epipolar ones). At the other end of the spectrum, visual appearance alone can be used to find matching features when such an assumption does not hold: For example, bag-of-features methods that discard *all* spatial information to build some invariance to intra-class variations and viewpoint changes have been applied quite successfully in image classification tasks [34], [35]. Modern methods for image matching now tend to mix both geometric and appearance cues to guide the search for correspondences (see, for example, [18], [21]).

Many matching algorithms proposed in the 80s and 90s have an iterative form but are not explicitly aimed as optimizing a well-defined objective function (this is the case for RANSAC and alignment methods for example). The situation has changed in the past few years, with the advent of combinatorial or mixed continuous/combinatorial optimization approaches to feature matching (see, for example [5], [19], [23], [25], [36])[1]. This paper builds on this work in a framework that can accommodate both (mostly local) geometric invariants and image descriptors. Concretely, the search for correspondences is cast as a hypergraph matching problem using higher-order constraints instead of the unary or pairwise ones used by previous methods: First-order methods based (for example) on local image descriptions are susceptible to image ambiguities due to repeated patterns, textures or non-discriminative local appearance for example. Geometric consistency is normally enforced using pairwise relationships between image features. In contrast, we propose in this paper to use higher-order (mostly third-order) constraints to enforce feature matching consistency (Figure 2). This work generalizes the spectral matching method of [19] to higher-order potentials: The corresponding hypergraph matching problem is formulated as the maximization of a multilinear objective function over all permutations of the features. This function is defined by a tensor representing the affinity between feature tuples. It is maximized by first using a multi-dimensional power method to solve a relaxed version of the problem, whose solution is then projected onto the closest assignment matrix.

The three main contributions of this article are (1) the application of the tensor power iteration method to the high-order matching task, combined with a relaxation based on constraints on the row norms of assignment matrices, which is tighter than previous methods (Section 3), (2) an $\ell^1$-norm instead of the classical $\ell^2$-norm relaxation, that provides solutions that are more interpretable but still allows an efficient power iteration algorithm (Section 4), and (3) the design of appropriate similarity measures that can be chosen either

- *Olivier Duchenne and Jean Ponce are with the department of Computer Science at École Normale Supérieure de Paris and the Willow project team (CNRS/ENS/INRIA UMR 8548)*
  *E-mail: olivier.duchenne at ens.fr and jean.ponce at ens.fr*
- *Francis Bach is with INRIA and the Sierra team, Laboratoire d'Informatique de École Normale Supérieure de Paris (CNRS/ENS/INRIA UMR 8548)*
  *E-mail: francis.bach at ens.fr*
- *In-So Kweon is with RCVlab in the department of electrical engineering at the university of KAIST, South Korea.*
  *E-mail: iskweon at ee.kaist.ac.kr*

1. To be fair, it should be noted that optimization-based approaches to graph matching were considered a key component of object recognition and scene analysis strategies in the 70s and 80s, see for example the classical text by Ballard and Brown [4, Ch. 11].

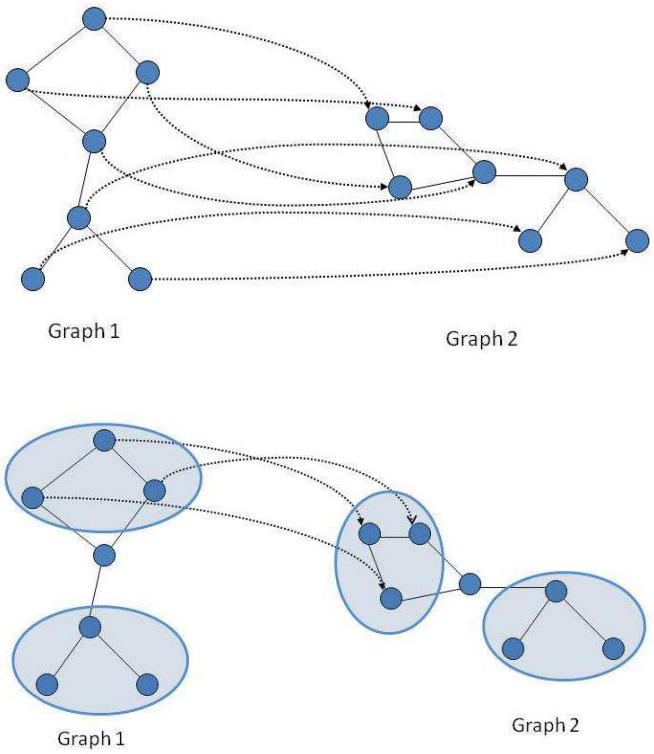


Fig. 1. Top: Given two graphs, the matching problem is to find node correspondences which preserve their topology. Bottom: In a hypergraph, one hyperedge can link more than 2 nodes. As an example, in this figure four hyper-edges are represented by circles. Each of them regroups 3 nodes that they link together. The three matches drawn on the bottom figures induce a matching between two hyperedges.

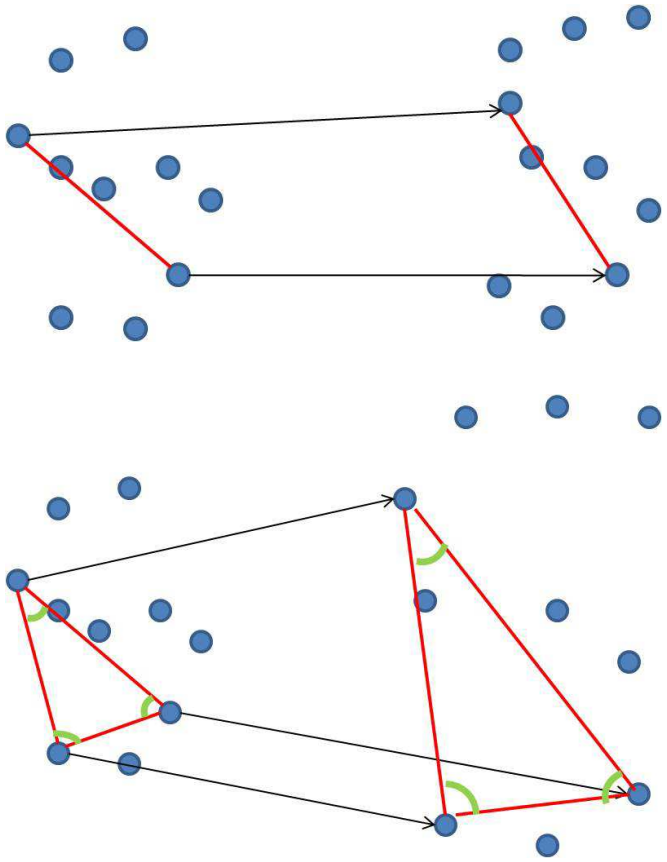

Fig. 2. Top: second-order potentials can be made rotation-invariant by comparing distances between matched points. Down: Third-order potentials can be made similarity-invariant by comparing the angles of triangles.

to improve the invariance of matching, or to improve the expressivity of the model (Section 5). The proposed approach has been implemented, and it is compared to state-of-the-art algorithms on both synthetic and real data. As shown by our experiments (Section 7), our implementation is, overall, as fast as these methods in spite of the higher complexity of the underlying model, with better accuracy on standard databases.

A preliminary version of this work appears in [9]. The source code of our software is available on line at http://www.di.ens.fr/~duchenne.

## 2 Graph Matching for Computer Vision

### 2.1 Previous work

As noted earlier, finding correspondences between visual features (such as interest points, edges, or even raw pixels) is a key problem in many computer vision tasks. The simplest approach to this problem is to define some measure of similarity between two features (e.g., the Euclidean distance between SIFT descriptors of small image patches [21]), and match each feature in the first image to its nearest neighbor in the second one. This naive approach will fail in the presence of ambiguities such as repeated patterns, textures or non-discriminative local appearance. To handle this difficulty, some methods try to enforce geometric consistency between pairs of feature correspondences. The basic idea is that if the points $p_1$ and $p'_1$ of image 1 are matched to points $p_2$ and $p'_2$ of image 2, then the geometric relation between $p_1$ and $p'_1$, and the one between $p_2$ and $p'_2$ should be similar.

Several pairwise geometric relations have been used. Leordanu and Hebert [19] use only the distance between two points, leading to a matching criterion which is invariant to rotation. In their objective function, Berg et al. [5] use a combination of potentials based on distances (rotation-invariant) and angles (scale-invariant), to find a trade-off between rotation and scale invariance. Some other methods (e.g., [29], [36]) use proximity, only assuming that two adjacent points should be matched in the other image to two points which are also close to each other. One difficulty here is to define an appropriate notion of neighborhood.

Recently, the computer vision community has put much effort in increasing the order of complexity of the models used: For example, Kohli et al. [16] introduce a high-order clique potential for segmentation, but the type of energy is limited to specific types of functions, using the alpha-expansion framework. Zass and Sashua [33] formulate the search for higher-order feature correspondences as a hypergraph matching problem. We will use the same formulation but a different optimization setup. In addition, unlike these authors, we will refrain from using independence assumptions (that may or may not be justified depending on the situation) to factor our

model into first-order interactions. As will be shown in the comparative experiments of Section 7, explicitly maintaining higher-order interactions in the optimization process leads to superior performance.

First, in Section 2.2 and 2.3, we discuss about classical graph matching. Then, in Section 3, we introduce our method.

### 2.2 Problem statement

We consider two images, and assume that we have extracted $N_1$ points from image 1, and $N_2$ from image 2. We do not assume that $N_1 = N_2$, i.e., there may be different numbers of points in the two images to be matched. Moreover, instead of points, we could use any other type of visual features such as edges, raw pixels, etc. Throughout this paper, for $s = 1, 2$, all indices $i_s, j_s, k_s$ will be assumed to vary from 1 to $N_s$. We will also note $i = (i_1, i_2)$, $j = (j_1, j_2)$, $k = (k_1, k_2)$ pairs of potentially matched points.

Let $P_n^s$ be the $n^{th}$ point of image $s$. The problem of matching points from image 1 to points from image 2 is equivalent to looking for an $N_1 \times N_2$ *assignment matrix* $X$ such that $X_{i_1,i_2}$ a.k.a. $X_i$ is equal to 1 when $P_{i_1}^1$ is matched to $P_{i_2}^2$, and to 0 otherwise. In this paper, we assume that a point in the first image is matched to *exactly one* point in the second image, but that one point in the second image may be matched to an arbitrary number of points in the first image, i.e., we assume that the sums of each row of $X$ is equal to one, but put no constraints on the column sums[2]. Thus, we consider the set $\mathcal{X}$ of assignment matrices:

$$\mathcal{X} = \{X \in \{0,1\}^{N_1 \times N_2},\ \forall i_1, \textstyle\sum_{i_2} X_{i_1,i_2} = 1\}.$$

Note that our definition is not symmetric (i.e., if we switch the two images, we obtain different correspondences). It can be made symmetric by considering the two possible matchings (image 1 to image 2 and image 2 to image 1) and combining them (in a mostly application-dependent way), e.g., by taking the union or intersection of matchings.

In [5], [8], [19], the matching problem is formulated as the maximization of the following score over $\mathcal{X}$:

$$\text{score}(X) = \sum_{i_1,i_2,j_1,j_2} H_{i_1,i_2,j_1,j_2} X_{i_1,i_2} X_{j_1,j_2},$$

where $H_{i_1,i_2,j_1,j_2}$ (which is equal to $H_{i,j}$ with our notations for pairs) is a binary potential corresponding to the pairs of feature nodes $(P_{i_1}, P_{j_1})$ of image 1, and $(P_{i_2}, P_{j_2})$ of image 2. H is a *positive* similarity measure, such that high values of $H$ correspond to similar pairs.

As described in Section 5, in this paper, we compute for each pair of nodes from the same image a feature vector $f$, and we compute $H$ as follow:

$$\forall i_1, i_2, j_1, j_2, H_{i_1,i_2,j_1,j_2} = \exp(-\gamma \|f_{i_1,j_1} - f_{i_2,j_2}\|^2).$$

Many other similarity measures are of course possible.

2. This framework can easily be extended to allow matching points from the first image to no point in the second image adding a dummy node to the second image as in [5] (if a point of the first image is matched to this dummy node, it means that it is matched to no point).

This graph matching problem is actually an integer quadratic programming problem, with no known polynomial-time algorithm for solving it. Approximate methods may be divided into two groups. The first one is composed of methods that use spectral representations of adjacency matrices (e.g., [19], [30]). The second group is composed of algorithms that work directly with the graph adjacency matrices, and typically involve a relaxation of the discrete optimization problem (e.g., [3], [32]). In this paper, we focus on improvements of the second group of methods.

In [8], [19], the set of binary matrices over which the optimization is performed is thus relaxed to the set of real matrices with Frobenius norm equal to $\sqrt{N_1}$, leading to the simpler problem:

$$\max_{\|X\|_F = \sqrt{N_1}} \sum_{i_1,i_2,j_1,j_2} H_{i_1,i_2,j_1,j_2} X_{i_1,i_2} X_{j_1,j_2}. \qquad (1)$$

Note that all the matrices in $\mathcal{X}$ have only $N_1$ non-zeros coefficients, which are equal to one, therefore they indeed all have their Frobenius norm equal to $\sqrt{N_1}$. In turn, Eq. (1) can be rewritten as $\max_{\|\tilde{X}\|_2 = \sqrt{N_1}} \tilde{X}^T \tilde{H} \tilde{X}$, where $\tilde{X}$ denotes the vector in $\mathbb{R}^{N_1 N_2}$ obtained by concatenating the columns of $X$ and, likewise, $\tilde{H}$ the $N_1 N_2 \times N_1 N_2$ symmetric matrix obtained by unfolding the tensor $H$. This is a classical Rayleigh quotient problem, whose solution $\tilde{X}^*$ is equal to $\sqrt{N_1}$ times the eigenvector associated with the largest eigenvalue (which we refer to as the *main eigenvector* $V$) of the matrix $\tilde{H}$ [12], and can be computed efficiently by the power iteration method described in the next section.

An important constraint that $H$ must satisfy is that it is pointwise non-negative. This is the main hypothesis of the Perron-Frobenius theorem [11] that ensures that $\tilde{X}^*$ only has non-negative coefficients, which simplifies the interpretation of the result (see [19]).

In order to obtain an assignment matrix in $\mathcal{X}$, i.e., a matrix with elements in $\{0, 1\}$ and proper row sums, the authors of [19] discretize the eigenvector $\tilde{X}^*$ using a greedy algorithm (see [19] for more details). One could also use the Hungarian algorithm with cost matrix $\tilde{X}^*$ to obtain a permutation matrix.

### 2.3 Power iterations for eigenvalue problems

The power iteration method is a very simple algorithm for computing the main eigenvector of a matrix, which is needed for matching.

**Input**: matrix $\tilde{H}$
**Output**: V main eigenvector of $\tilde{H}$
initialize $V$ randomly ;
**repeat**
  $V \leftarrow \tilde{H}V$ ;
  $V \leftarrow \frac{1}{\|V\|_2} V$ ;
**until** convergence;

**Algorithm 1:** Power iterations for eigenvalue problems.

This algorithm is guaranteed to converge geometrically to the main eigenvector of the input matrix [12]. As explained in

Section 6, in our situation, $H$ is very sparse and we want to take advantage of this. Indeed, each step of the power iteration algorithm requires only $O(m)$ operations, where $m$ is the number of non-zero elements of $H$. Also, typically, in our situation, the algorithm converges in a few dozen steps.

Thanks to this algorithm description, it becomes easy to see one reason why the output $V$ will have only non-negative values as described in the Perron-Frobenius theorem [11]. Let us assume we initialize $V$ with only non-negative values (since the algorithm converges to a global optimum, this will not change the output of the algorithm). In our case $H$ is also point-wise non-negative. Therefore at each iteration, each coordinate of $V$ will be replaced by a sum of products of non-negative values, which is also non-negative. So this property remains true until convergence. We will see that this nice property will be conserved in our higher-order algorithm.

## 3 Tensor formulation of hypergraph matching

We propose to use tensors to solve the high-order feature matching problem. Indeed, using tensors is quite natural to generalize the spectral matching [19] introduced in the previous section which deal with a matrix. Previous work except [33] only uses one-to-one and pair-to-pair comparisons for matching. In this paper, we want to compare tuples of points. We denote by $d$ the number of points per tuple, and add higher-order terms to the score function defined in Eq. (1). For simplicity, we will focus from now on third-order interactions ($d = 3$). Generalizations to higher-order potentials are (in theory at least) straightforward. However, in practice, it could lead to an exponential growth of the computational complexity.

We define a new high-order score:

$$\text{score}(X) = \sum_{i_1,i_2,j_1,j_2,k_1,k_2} H_{i_1,i_2,j_1,j_2,k_1,k_2} X_{i_1,i_2} X_{j_1,j_2} X_{k_1,k_2}, \quad (2)$$

where we assume that $H$ is a 6-dimensional super-symmetric tensor, i.e., invariant under permutations of indices in $\{i_1, j_1, k_1\}$ or $\{i_2, j_2, k_2\}$.

Here, the product $X_{i_1,i_2} X_{j_1,j_2} X_{k_1,k_2}$ will be equal to 1 if and only if the points $\{i_1, j_1, k_1\}$ are respectively matched to the points $\{i_2, j_2, k_2\}$. In this case, it will add $H_{i_1,i_2,j_1,j_2,k_1,k_2}$ to the total score function and 0 otherwise.

As described in Section 5, $H$ represents a similarity measure, which will be high if the set of features $\{i_1, j_1, k_1\}$ is similar to the set $\{i_2, j_2, k_2\}$. In our experiments, we compute for each triplet of nodes in the same image a feature vector $f$, and we compute $H$ as follow:

$$\forall i, j, k, H_{i_1,i_2,j_1,j_2,k_1,k_2} = \exp(-\gamma \| f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2} \|^2).$$

More details are provided in latter sections.

As explained in the next section, we can rewrite the score compactly using tensor notation as:

$$\text{score}(\tilde{X}) = \tilde{H} \otimes_3 \tilde{X} \otimes_2 \tilde{X} \otimes_1 \tilde{X}, \quad (3)$$

with the same notation as in the matrix case: $\tilde{X} = vec(X)$ and $H$ is rewritten as a tensor $\tilde{H}$ of size $(N_1 N_2)^d$.

This score can be interpreted as a hypergraph matching score. In a hypergraph, an edge can link more than two vertices together (Figure 1). In this framework, any element of $H$ is a matching score between two hyper-edges.

In Section 5, we will explain how higher-orders potentials can be used to have more invariant or more expressive features.

### 3.1 A short introduction to tensors

A tensor is the $n$-dimensional generalization of a matrix: a matrix can be represented as 2-D rectangular table, and tensors can be viewed as $n$-dimensional hyper-rectangular tables. Each of the elements of such a tensor is indexed by $n$ numbers: $H = \{H_{i_1,i_2,\ldots,i_n}\}$.

A tensor and a vector can be multiplied in different ways. In this article, we use the following notation:

$$B = A \otimes_k V,$$
$$B_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n} = \sum_{i_k} A_{i_1,\ldots,i_k,\ldots,i_n} V_{i_k},$$

where $V$ is a $n$-dimensional vector and $A$ a $n$-dimensional tensor. Like a matrix multiplied by a vector produces a vector, an $n$-dimensional tensor multiplied by a vector is $(n-1)$-dimensional. Also, like the matrix-vector multiplication that can be done in two ways (on the left or on the right), the tensor-vector multiplication can be done in $n$ different ways. The index $k$ in the notation $\otimes_k$ indicates that we multiply on the $k^{th}$ dimension.

In Eq. (3), we use the following calculus:

$$\begin{aligned} \text{score}(\tilde{X}) &= \tilde{H} \otimes_1 \tilde{X} \otimes_2 \tilde{X} \otimes_3 \tilde{X} \\ &= (((\tilde{H} \otimes_1 \tilde{X}) \otimes_2 \tilde{X}) \otimes_3 \tilde{X}) \\ &= (((\sum_k H_{i,j,k} X_k)_{i,j} \otimes_2 \tilde{X}) \otimes_3 \tilde{X}) \\ &= \sum_{i,j,k} H_{i,j,k} X_i X_j X_k. \end{aligned}$$

So the two expressions of the score in Eq. (2) and (3) are equivalent.

### 3.2 Tensor power iterations

To find the optimum of the high-order score of Eq. (2), we use a generalization of the previously mentioned power iterations, as proposed in [17]. The algorithm presented below extends Algorithm 1.

This method is not guaranteed to reach a global optimum. However, it converges to a stationary point for tensors that lead to convex functions of $X$ [28]. In our experiments, it converges almost always to a very satisfactory solution. Also, the authors of [28] propose a smart way to initialize it, to lead to a quantifiable proximity to the optimal solution.

We can see that, as in the matrix case, if we initialize $V$ with only non-negative values, the resulting vector will have only non-negative values. This is required to have a meaningful result. Indeed, if negative values of $X$ in the score in Eq. (2) were allowed, some product of negative values could have a positive value. Therefore even coordinates of $X$ with a

**Input**: supersymmetric tensor $\tilde{H}$
**Output**: V main eigenvector of $\tilde{H}$
initialize $V$ randomly ;
**repeat**
  $V \leftarrow \tilde{H} \otimes_1 V \otimes_2 V$ ;
  ( i.e. $\forall i,\ V_i \leftarrow \sum_{j,k} H_{i,j,k} V_j V_k$ )
  $V \leftarrow \frac{1}{\|V\|_2} V$ ;
**until** convergence;

**Algorithm 2:** Supersymmetric tensor power iteration (third order).

low value could increase the final score, preventing us from interpreting the coordinates of $H$ as a similarity potential activated only when all corresponding pairs are matched.

### 3.3 Tensor power iterations for unit-norm rows

In our context, we want to constrain the norm of each row of $X$ to 1, which is a tighter relaxation of $\mathcal{X}$ than matrices of fixed Frobenius norm. In addition, this corresponds to a many-to-one matching setting: all nodes of the first images are matched to exactly one node in the second image, but several nodes in the first images can be matched to the same one in the second image. We denote by $\mathcal{C}_2$ the set of matrices such that all theirs rows have unit Euclidean norm.

We can extend the previous algorithm to this new set of matrices (Algorithm 3).

**Input**: supersymmetric tensor $\tilde{H}$
**Output**: $V = [v_{1,1}, \ldots, v_{N_1,N_2}]^T$ stationary point
initialize $V$ randomly ;
**repeat**
  $V \leftarrow \tilde{H} \otimes_1 V \otimes_2 V$ ;
  ( i.e. $\forall i,\ V_i \leftarrow \sum_{j,k} H_{i,j,k} V_j V_k$ )
  $\forall i_1,\ V(i_1,:) \leftarrow \frac{1}{\|V(i_1,:)\|_2} V(i_1,:)$ ;
**until** convergence;

**Algorithm 3:** Supersymmetric tensor power iteration (third order) with unit norm constraints. $V(i,:)$ denotes the vector $(V_{i,1}, V_{i,2}, \ldots, V_{i,N_2})^T$ .

As shown in the appendix, we have extended the proof of [28] to handle those new constraints. In particular, we have shown that this algorithm has the same nice properties of the previous one: if the score is a convex function of $X$, then Algorithm 3 converges to a stationary point $V$. Note that we can always make the score convex by adding to it a multiple of the function $\tilde{X}^\top \tilde{X}$. Since the $\tilde{X}$ vectors in $\mathcal{C}_2$ all have the same norm, this change the value of the score function only by a constant and thus does not change its optima on $\mathcal{C}_2$ (the set of matrices whose Euclidean norms of each of the $N_1$ rows are equal to one).

Finally, we want to obtain correspondences and need to compute a binary matrix $X$ from $V$. We obtain a natural projection step here on the set $\mathcal{X}$: For each row, the coordinate with maximum value in $V$ is set to 1 in $X$, and the other coordinates of $X$ are set to 0.

### 3.4 Merging potentials of different orders

It could be interesting to include in the matching process, at the same time, information about different potential orders (e.g., considering at the same time pair similarities and triplet similarities). To do this, a first solution is to include the low-order information into the tensor of the highest-order potential $H$. Cour and Shi [8] present a method to do this, combining second and first-order potential. The generalization to our setting is straightforward. However, in our power iteration framework, it is equivalent to use the simple following algorithm (which could also be extended to constrain rows to have unit norms):

**Input**: several supersymmetric tensors $\tilde{H}^d$ of order $d$
**Output**: V main eigenvector of H
initialize $V$ randomly ;
**repeat**
  $V \leftarrow \tilde{H}^4 \otimes_1 V \otimes_2 V \otimes_3 V +$
  $\tilde{H}^3 \otimes_1 V \otimes_2 V + \tilde{H}^2 \otimes_1 V + \tilde{H}^1$;
  ( i.e. $\forall i, V_i \leftarrow \sum_{j,k,l} H^4_{i,j,k,l} V_j V_k V_l +$
  $\sum_{j,k} H^3_{i,j,k} V_j V_k + \sum_j H^2_{i,j} V_j + H^1_i$ )
  $V \leftarrow \frac{1}{\|V\|_2} V$ ;
**until** convergence;

**Algorithm 4:** Multiple order supersymmetric tensor power iteration (fourth order).

## 4 $\ell^1$-NORM CONSTRAINT FOR ROWS

One of the main problems of spectral relaxations is that the solution is often nearly uniform, which means that it is hard to extract from it an assignment matrix with values in $\{0, 1\}$. This is due in part to the relaxation of the set $\mathcal{X}$ of assignment matrices to matrices in $\mathcal{C}_2$ with unit $\ell^2$-norm rows, which does not lead to sparsity. In fact, we can also relax the set $\mathcal{X}$ to the matrices in $\mathcal{C}_1$ with rows having unit $\ell^1$-norm (i.e., sum of absolute values). As shown in Figure 3, this leads to results that are more easily interpretable.

In the context of second-order interactions, solving the $\ell^1$-norm problem cannot be done by power iterations. However, in our higher-order context, this can be done seamlessly. Indeed, solving the following problem on $\mathcal{C}_1$:

$$\max_{X \in \mathcal{C}_1, X \geqslant 0} \sum_{i,j} H_{i,j} X_i X_j$$

is equivalent to solving (on $\mathcal{C}_2$):

$$\max_{Y \in \mathcal{C}_2} \sum_{i,j} H_{i,j} Y_i^2 Y_j^2$$

with the change of variable: $Y_i^2 = X_i$. The order of this new problem is 4 when using the tensor power iteration algorithm, but the complexity is still as low as second-order problem (see Algorithm 5). Using this algorithm we usually obtain in practice an almost completely binary solution, as shown on a particular example in Figure 3. This method is easily extended to solve any high-order matching problem.

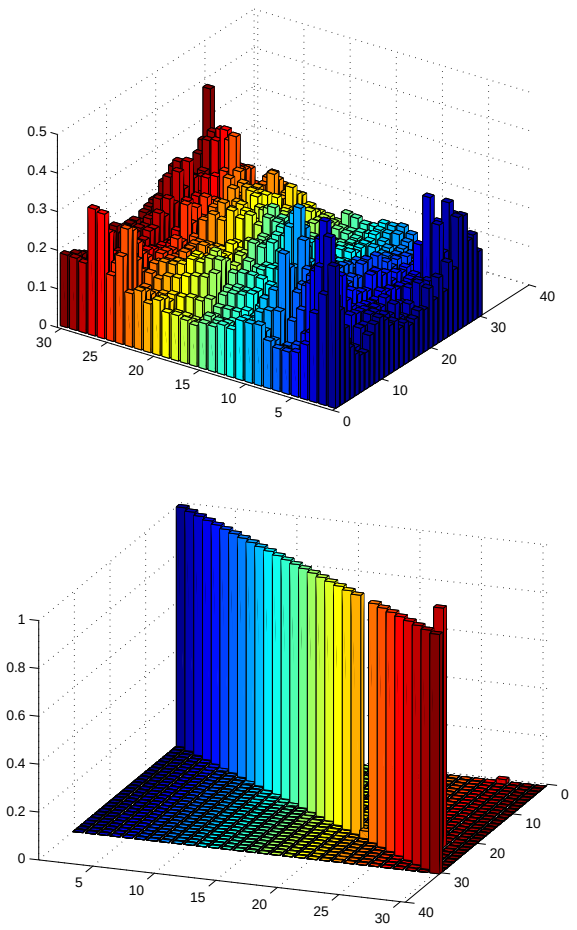

Fig. 3. We run the spectral algorithm to match a random point cloud to a randomly perturbated copy of itself. We show here the resulting assignment matrices by using $\ell^2$ or $\ell^1$-norm constraints (respectively Algorithm 1 and 5). The indexing of the points is made such that for all $i$, the point $i$ of the first cloud corresponds to the point $i$ of the second cloud. So the perfect assignment matrix should be the identity matrix. The horizontal axes correspond to the coordinates of the assignment matrix and the vertical one to its values. Top: Values of the assignment matrix when the $\ell^2$-norm is used. They are hard to project to a matrix in $\mathcal{X}$, i.e., with values in $\{0,1\}$. Bottom: When using the $\ell^1$-norm, we obtain directly a very clear assignment matrix with minor adjustments. Indeed, its values are nearly boolean.

**Input**: matrix $\tilde{H}$
**Output**: $V$ stationary point
initialize $V$ randomly ;
**repeat**
  $V \leftarrow (\tilde{H}(V \circ V)) \circ V$ ;
  ( i.e. $\forall i,\ V_i \leftarrow V_i \sum_j H_{i,j} V_j^2$ )
  $\forall i_1,\ V(i_1,:) \leftarrow \frac{1}{\|V(i_1,:)\|_2} V(i_1,:)$ ;
**until** convergence;

**Algorithm 5:** Tensor power iteration for the $\ell^1$-norm relaxation. Here, $\circ$ represents the Hadamard product (or pointwise product). $V(i,:)$ denotes the vector $(V_{i,1}, V_{i,2}, \ldots, V_{i,N_2})^T$ .

## 5 Building Tensors for Computer Vision

We can use higher-order potentials to increase either the geometric invariance of image features, or the expressivity of the models (see Figure 4). We describe here a few possible potentials. They are all based on computing a Gaussian kernel between appropriate invariant features. Clearly, many other potentials are possible.

In this section we will only consider third-order potentials. As illustrated by Figure 2, classical methods try to remove ambiguities by looking for matches that preserve some properties of point pairs. Here, we will try to preserve properties of point triplets. In particular, in most of the cases, we will use the properties of the triangle formed by three points. Basically, if the points $(P_1^1, P_2^1, P_3^1)$ are matched to the points $(P_1^2, P_2^2, P_3^2)$, the corresponding triangles should be similar.

In [19], rotation and translation-invariant potentials based on edge lengths and angles are used since it is impossible to build invariants to larger classes of transformations from feature pairs alone. Here, we propose using potentials based on triplets of points, which can be made invariant to richer classes of transformations, including (planar) similarities, affine transformations, and projective ones.

### 5.1 Similarity-invariant potentials

The angles of a triangle are invariant under similarities. Thus we can describe each triangle by its three angles (Figure 2). However, in our implementation, we rather use the sines of the angles to speed-up the computation.

### 5.2 Affine-invariant potentials

When the camera is moving, in the general case, the transformation of the image is perspective. However, this transformation can also be affine when the seen object is planar or when looking locally at the image. Therefore affine invariance can be a good approximation of perspective invariance.

Concretely, we normalize each triangle into an equilateral one, and then compare the intensity patterns of normalized triangles by normalized cross correlation. This description of the triangle is of course invariant under affine transformation.

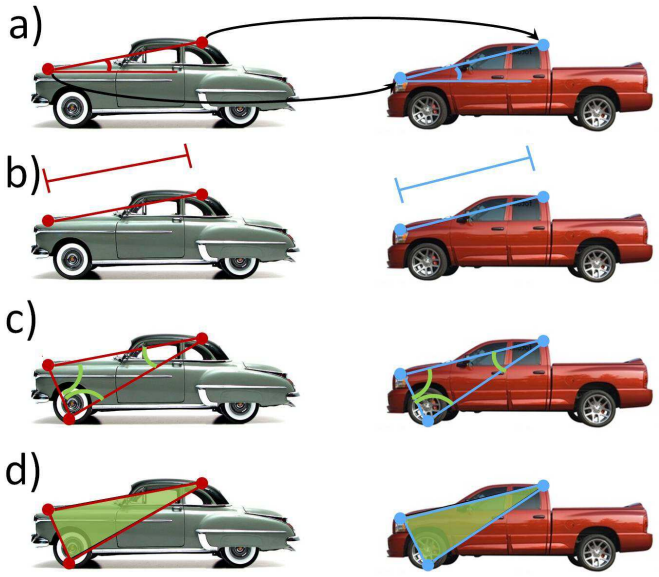


Fig. 4. (a) a scale-invariant pairwise potential, the angle with respect to the horizontal axis. (b) a rotation-invariant pairwise potential, the distance between the two points. (c) a scale and rotation invariant triplet potential, the 3 angles of the triangle. (d) triplets allow the description of the interior of the triangle, which is a much richer description, and can be affine invariant.

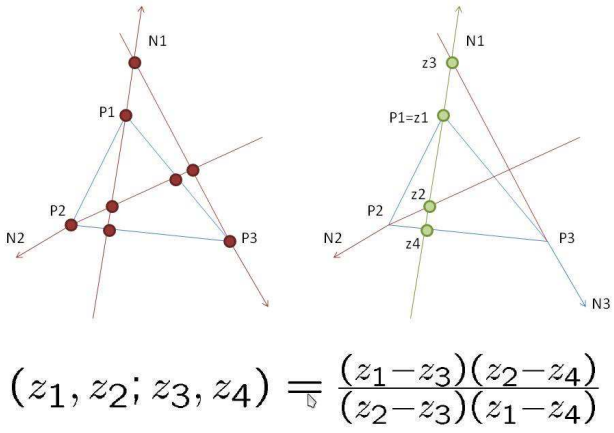


Fig. 5. Left: diagram illustrating the features used in the proposed projective-invariant potential. In order to describe the blue triangle, we build three red lines with four points on each. Right: in order to describe one of these lines (the green one), we denote the four points by $z_1$ to $z_4$ (complex numbers). Bottom: we use the cross ratio formula to compute one descriptor for each of the three lines.

### 5.3 Projective-invariant potentials

Inspired by [20], we can also develop higher-order potentials invariant to planar projective transforms with no parametric form when the scene shape is unknown. If we sample only feature points on the contours in the image, we can use the edge direction as an additional feature, and focus on properties of three points and three directions that are conserved under projective transforms. The main property conserved by a projective transform is the cross-ratio. So if we suppose that the object surface in the triangle we are looking at is flat, we can build three lines with four points on each. We compute the descriptor of the three points $P_1, P_2, P_3$ shown in Figure 5. We also show the three vectors $N_1, N_2, N_3$ which are orthogonal to the image gradient at each point. We draw a line from the point $P_1$ in the direction of vector $N_1$ which intersects the other lines $(P_2, N_2)$, $(P_3, N_3)$, $(P_2, P_3)$ in the points $z_2, z_3, z_4$. And we add $z_1 = P_1$. If the $z_i$ are written with complex numbers, their cross ratio is:

$$(z_1, z_2; z_3, z_4) = \frac{(z_1 - z_3)(z_2 - z_4)}{(z_2 - z_3)(z_1 - z_4)},$$

and this formula is computed for each of the three points. We will use the three cross-ratios defined by those points to make a perspective-invariant descriptor.

### 5.4 More expressive potentials

Most previous approaches focus on using rather simple geometric relationships between points. Typically the descriptor of their pairwise relationship is a scalar or a low-dimensional vector. As a consequence, such descriptors have low discriminative power, and many different pairs of points have similar descriptors. Therefore matching becomes ambiguous. We believe that our higher-dimensional framework makes it possible to build more expressive features. Triangles, unlike line segments, have an interior. So, it should be possible to have image-based features to describe this interior. To do this, we can use the simple method explained in section 5.2, or, for instance, a histogram of gradient features. Obviously, many other types of features are possible (e.g., bags of words.). These new features would be more specific, and would have higher discriminative power. Therefore a triplet in one image would have fewer similar triplets in the other. In this situation, the matching would become less ambiguous and easier to compute.

### 5.5 3D potential

We now present a high-order potential to match 3D point clouds. We assume that the vertical axis is known and design a potential which is invariant to scale and rotation about this axis. Clearly, many other potentials are possible depending on the assumptions made (e.g., scale/vertical axis known or unknown). We use a 6-dimensional feature to describe the 3D point triplets (Figure 6): The first three features are the angles between the three edges of the triangle and the vertical. For the three other features we use the angles of the triangle (as in the 2D case).

## 6 Implementation

In the case of $d$-th order potentials, the brute force algorithm (see Algorithm 2) has a complexity $O(n^{2d})$ per iteration, where $n = \max\{N_1, N_2\}$. However, Leordeanu and Hebert [19] argue that, in their case, the matrix $\tilde{H}$ has approximately $O(n^3)$ non-zero values. Therefore, the complexity of their

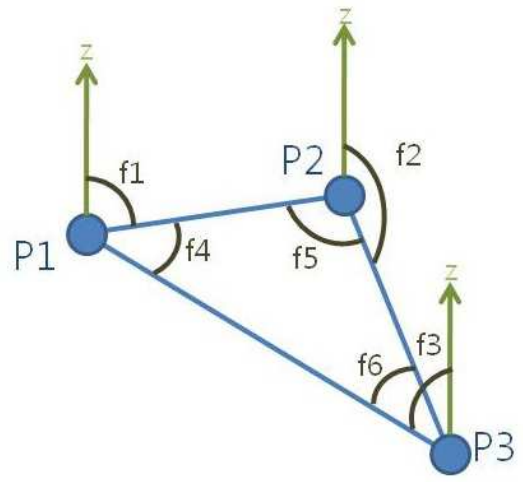


Fig. 6. Diagram explaining the 3D invariant construction. The 3 points described ($P_1$ to $P_3$) and their triangle are in blue. $f_1$ to $f_6$ are the angles used as features. The 3 vertical green arrows represent the vertical axis.

**Input**: images $I_1$, $I_2$ and point sets $\mathcal{P}_1$,$\mathcal{P}_2$
**Output**: tensor H
$H \leftarrow$ empty tensor ;
**foreach** $t \in$ *set of all tuples in* $\mathcal{P}_2$ **do**
  $f \leftarrow$computetupleFeature$(t, I_2)$ ;
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$ ;
**end**
$\mathcal{T} \leftarrow$computeANNtree$(\mathcal{F})$ ;
$\mathcal{S} \leftarrow$select some tuples in $I_1$ ;
**foreach** $s \in \mathcal{S}$ **do**
  $\mathcal{N} \leftarrow$search for $k$ nearest-neighbors$(\mathcal{T}, s)$ ;
  **foreach** $n \in \mathcal{N}$ **do**
    $H(\text{index}(s), \text{index}(n)) \leftarrow$
    similarity$(\text{descriptor}(s), \text{descriptor}(n))$ ;
  **end**
**end**

**Algorithm 6:** Efficient ANN-based algorithm for computing the tensor.

algorithm is around $O(n^3)$ operations per iteration for second-order potentials.

As we increase the order of the potential, the number of elements of the tensor $H$ increases exponentially, and the computation time of the standard algorithm would be the same. Therefore an efficient algorithm is required.

The first step of our algorithm is to build the tensor. This step is often neglected in the literature, but actually requires as much computation as the matching itself, even in the case of conventional graph matching. The tensor size $(N_1 N_2)^d$ is huge . If we computed it completely, it would require $O((N_1 N_2)^d n_f)$ operations, where $n_f$ is the size of the descriptor of a tuple. However, if we use a truncated similarity measure (i.e., with a compact support), the tensor $H$ can be very sparse. Moreover, in practice, it is not necessary to compute $H$ completely. Therefore in our algorithm, we compute only a small part of $H$.

In our experiment, we use a truncated Gaussian kernel: $H_{i,j,k} = \exp(-\gamma \|f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2}\|^2)$ if $\|f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2}\| \leq \sigma$ otherwise 0, where $f_{i_1,j_1,k_1}$ is the feature vector describing the tuple $(i_1, j_1, k_1)$.

So, for each tuple $i$ of the first image, we need to find the features of image 2 in a neighborhood of size $\sigma$. In practice, we only take the $k$ nearest neighbors with $k$ fixed. This allows us to use a standard implementation of approximate nearest neighbors [24], which in practice is very efficient. This ANN search implementation is based on kd-trees. In our experiments using this approximate algorithm does not bring any significant change to the results.

However, doing this for all tuples in image 1 would be very time consuming, and forcing all the tuples to be matched correctly is very redundant. So, as in [33], we only sample $tN_1$ triangles in image 1, with fixed $t$.

Then, we sample all the possible triangles of image 2, and compute their descriptors. We store them in a kd-tree to allow an efficiently search. For each of the selected triangles of image 1, we find the $k$ approximate nearest neighbors of image 2. Then we compute the tensor values: $H_{i_1,j_1,k_1,i_2,j_2,k_2} = \exp(-\gamma \|\text{triangle}(i_1, j_1, k_1) - \text{triangle}(i_2, j_2, k_2)\|^2)$ if $\text{triangle}(i_2, j_2, k_2)$ is among the $k$ nearest neighbors of $\text{triangle}(i_1, j_1, k_1)$, and 0 otherwise. Then we start the power iteration.

The total complexity of the algorithm is $O(n^d \log(n) + ntk \log(n))$ per iteration. The final algorithm typically takes one second for 80 points, $t = 20$ and $k = 500$. The complete setup is summarized in Algorithm 6.

### *Smart selections of triangles*

There are several strategies for selecting triangles depending of the final goal. If one wants to match and allow deformations, the triangle should be selected at small scales. On the other hand, if one wants to capture the global property of a shape, one should select big triangles.

## 6.1 Separable Similarity Measure

One important problem with spectral methods (high-order or not) is that $H$ can be huge. But in some cases, it is possible to avoid computing it. $H_{i,j,k}$ should be a similarity measure between the tuples $(i_1, j_1, k_1)$ and $(i_2, j_2, k_2)$. In this section, we explain that if we can decompose this measure as the inner product of two descriptors $< f_{i_1,j_1,k_1}, f_{i_2,j_2,k_2} >$, we do not have to compute the whole $H$. When the similarity function is a positive definite kernel, it is always possible to write it as an inner product. However, we also need to have a finite representation of $f$.

In this situation, we can write (for the second order case):

$$\tilde{H} = \sum_d F_1^d \otimes_{kro} F_2^d,$$

where $\otimes_{kro}$ is the Kronecker product, and $F^m_{I,i,j}$ is the $m^{th}$ descriptor of the pair $(i, j)$ of image $I$. These two feature matrices can also be very sparse if one considers only the relationship between certain pairs (e.g., pairs of close points).

In this situation we can simplify the computation of the power iteration step:

$$\begin{aligned}
\tilde{X} &\leftarrow \tilde{H}\tilde{X} \\
\tilde{H}\tilde{X} &= (\sum_m F_1^m \otimes_{kro} F_2^m)\tilde{X} \\
&= \sum_m vec(F_2^m X (F_1^m)^T) \\
\forall i_1, i_2, X_{i_1,i_2} &\leftarrow \sum_{j_1,m} F^m_{1,i_1,j_1} \sum_{j_2} X_{j_1,j_2} F^m_{2,i_2,j_2} \\
&(= \sum_{j_2,m} F^m_{2,i_2,j_2} \sum_{j1} X_{j_1,j_2} F^m_{1,i_1,j_1}),
\end{aligned}$$

by using the formula $(B^T \otimes_{kro} A)vec(X) = vec(AXB)$.

This decomposition of the similarity measure decreases the amount of memory required by the program. Its time complexity is only $O(nnz(F_1)n + nnz(F_2)n)$ per iteration and, unlike the method described in the previous section, it is an exact algorithm.

In the higher-order case, we can also use this decomposition and the new power iteration step can be written as follows (for the third order case) :

$$\begin{aligned}
&\forall i_1, i_2, \\
&X_{i_1,i_2} \leftarrow \sum_{j_1,k_1,j_2,k_2} H_{i_1,j_1,k_1,i_2,j_2,k_2} X_{j_1,j_2} X_{k_1,k_2} \\
&= \sum_{j_1,k_1,j_2,k_2,m} F^m_{1,i_1,j_1,k_1} F^m_{2,i_2,j_2,k_2} X_{j_1,j_2} X_{k_1,k_2} \\
&= \sum_{j_1,k1,m} F^m_{1,i_1,j_1,k_1} \sum_{j_2} X_{j_1,j_2} \sum_{k_2} F^m_{2,i_2,j_2,k_2} X_{k_1,k_2}.
\end{aligned}$$

Here, the complexity is $O(nnz(F_1) \cdot n + \|F_1\|_{\inf,0,0} \cdot \|F_2\|_{0,0,\inf} \cdot d + nnz(F_2) \cdot n)$ per iteration, where $\|F_1\|_{\inf,0,0}$ is the number of doublets $(j_1, k_1)$ such that $F_{1,(\cdot),j_1,k_1}$ is not null, and $\|F_2\|_{0,0,\inf}$ is the number of doublets $(i_2, j_2)$ such that $F_{2,i_2,j_2,(\cdot)}$ is not null.

# 7 Experiments

In the experiments presented here (with some exceptions detailed in the subsections), we use Algorithm 6 to compute the tensor. Then, we use the tensor power iteration with unit-norm row constraints described in Algorithm 3, and the $\ell^1$-variant of Algorithm 5. The three first experiments use the simple similarity-invariant potential presented in section 5.1. The smart selection of triangles is not used in those experiments. As explained in section 6, we compute $H$ using the following formula: $H_{i,j,k} = \exp(-\gamma \| f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2} \|^2)$. We set the parameter $\gamma$ as follows: we compute all the $\ell^2$ distances between the tuples of image 1 and their nearest neighbors in image 2 as described in Algorithm 6, then we set $\gamma$ to the inverse of the average of all the squares of the computed distances. More details are given in the following subsections.

*Running time*

Even though our tensor power iteration is slower than the probabilistic hypergraph matching method proposed in [33], since both methods have to first compute $H$, their total running times are similar.

## 7.1 Synthetic data

Following [19], [33], we first use synthetic data in order to quantitatively compare our algorithm to the state of the art. We sample randomly and uniformly $n = 25$ points in the 2D plane. We create a second set of points by perturbing the first one with Gaussian noise on their positions. Then, we compare different algorithms to match those two sets. The algorithm provides $n$ matches. The accuracy of the algorithm is computed as the number of good matches divided by $n$.

In order to have a fair comparison between our method and probabilistic hypergraph matching [33], we first compute the tensor as described earlier. Then, we marginalize it as explained in [33], and we use the resulting vector with the algorithm they provide on line. We also compare our result and [33] to spectral matching [19] to show the improvement of using higher-order potentials.

First, we add Gaussian noise to the position of the second point set, apply a global rotation, and add outliers. The results are shown in Figure 7 (top). We can see that our method outperforms the other two. Our interpretation is that when many outliers are added, the ambiguity of pairwise methods [19] increases, because many pairs become similar, whereas triplets are less likely to become similar. Moreover, probabilistic hypergraph matching [33] reduces the high-order problem to a first-order one, so that it is likely to match points which have the same neighborhoods. Such a method thus becomes ambiguous when there are many outliers.

Second, we add Gaussian noise, rotation, *and* rescaling. Indeed, low-order matching techniques, such as the spectral method, cannot handle those transformations (rotation and rescaling at the same time). In Figure 7 (bottom), we can see that our method and the one of [33] are indifferent to those transformations, but the performance of [19] drops to $50\%$ after a scaling of only $1.1$ or $0.9$, and quickly reaches chance level at $1.2$ or $0.8$.

## 7.2 House Dataset

The House dataset [2] is commonly used to test the performance of matching algorithms. Some objects are taken from different viewpoints and $n = 30$ key points, which are present in every frame, are labeled. The scale is always roughly the same, but the transformation is now perspective (although the experiment has been designed such that it is almost orthographic). Since the ground truth is provided, it is also easy to compute the accuracy of the algorithm. The algorithm provides $n$ matches, and the accuracy is the number of good matches among them divided by $n$. In Figure 8, we can see that the low-order algorithms cannot handle the fact that in perspective transforms, the relative positions of points change in a complex way.

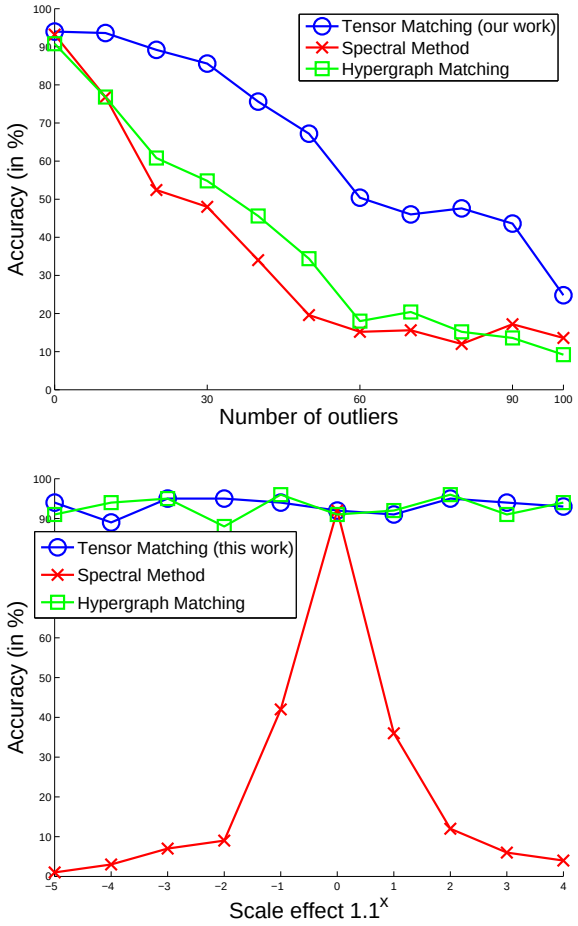


Fig. 7. Top: Accuracy as a function of the number of added outliers. Bottom: Accuracy, as a function of the rescaling. (e.g., $x = 2$, correspond to a scaling of $1.1^2$).

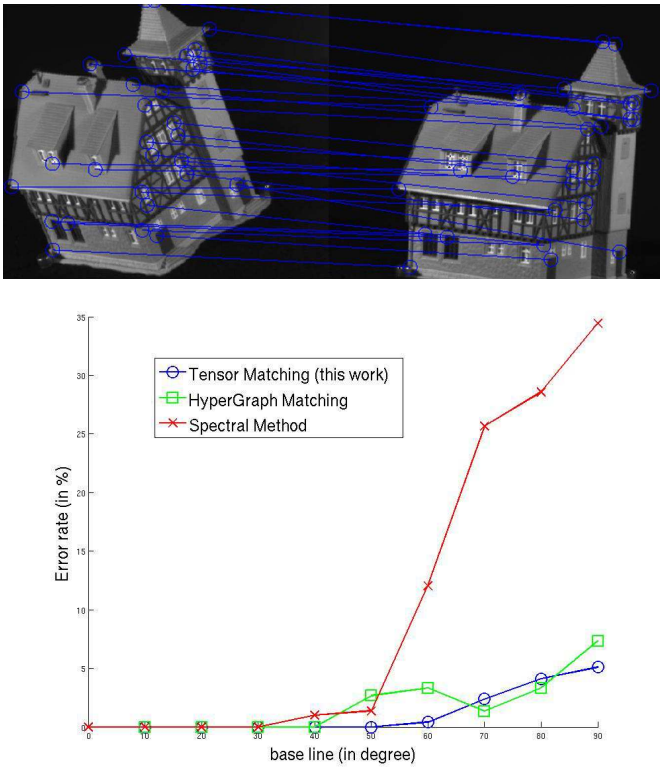


Fig. 8. House data. Top: Correspondences found by the proposed method in the house dataset. Bottom: Error Rate on this dataset depending on the base line angle, for different methods.

### 7.3 Natural images

We take images from the Caltech-256 image database [13] which depict objects on a clear background. We extract their silhouettes and subsample points on them. We can then match images from the same class using our algorithm; results are presented in Figure 9. Our tensor-based algorithm is able to match objects with different visual appearances in the presence of strong deformations.

### 7.4 3D object matching

We have also experimented with the 3D point descriptor described in Section 5. We have downloaded some freely available 3D models [1], have manually extracted some points at key positions. After that, we execute our algorithm with only our third-order potential (no local description of the shape). Points are not always matched perfectly, but the results is almost always visually good. Some results are shown in Figure 10.

For more completeness, we also use some 3D models from SHREC 2009 dataset [26], randomly select 70 points on each of them, and match them. The results can be seen on Figure 11. The two sets of experiments have been performed with the same set of parameters.

### 7.5 Potentials of different orders

As explained in Section 3.4, we can also simultaneously use potentials of different orders. We have chosen an example (Figure 12) which is both hard for first-order matching based on SIFT-descriptors, and for tensor matching based on triplets only. We have taken two pictures of the same person with changes in both viewing angle and face expression. Since the face is deformable, local descriptors tend to be unreliable. In addition, algorithms based on the assumption that the transformation is parametric (such as RANSAC) are not applicable.

In both pictures, we automatically extract around 300 interest points, and for each of them compute its SIFT descriptor (using the implementation of [31]). Then we match each interest point to the one with the closest descriptor in the other image, when the match is unambiguous, as described in [22]. The result (Figure 12.a) is not satisfying. We believe that this is due to the non-parametric deformation of the face, occlusion, relatively textureless images, and ambiguities due to the symmetry.

We use also triplet information only, without SIFT descriptors. Here too, the graph nodes are the SIFT interest points, and are automatically extracted. This leads to ambiguity in the matching process: not all nodes in one image have similar nodes in the other image. Moreover, matching with only

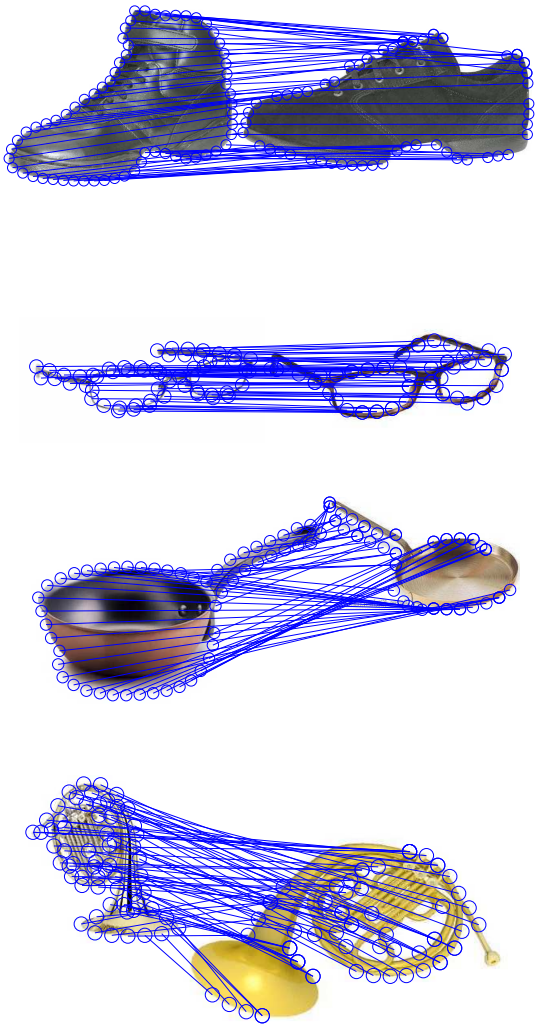

Fig. 9. Matching silhouettes from the Caltech-256 database.

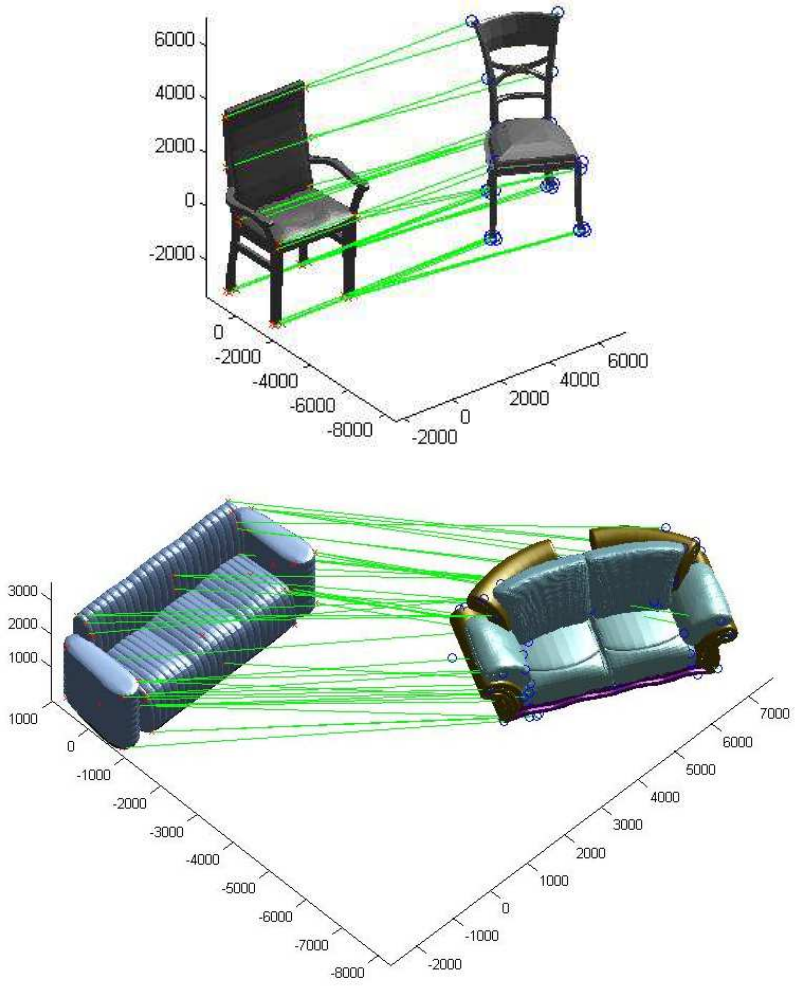


Fig. 10. Two pairs of 3D models matched by our algorithm.

scale- and rotation-invariant features (see Section 5.1) is too ambiguous. So we use as triplet descriptor the concatenation of the sines of the three angles and the image coordinates (x-y) of the three edges. This descriptor is only translation invariant but can be robust to small scale or rotation changes. In Figure 12.b, one can see that the resulting matching is globally good, but some details are wrong. Since the descriptors are only based on geometry (and not on the image), the algorithm matches the left (resp. right) part of face 1 to the left (resp. right) part of face 2. However, since the face has turned in the second image, the corresponding parts no longer keep their original geometric location, resulting in wrong matches.

As explained in section 3.4, we can also combine both cues. The algorithm can use both image-based cues from the first-order potential and geometry-based cues from the third-order potential. The result is very satisfying (Figure 12.c). We also show in figure 12.d a result with dummy nodes which is slightly improved.

## 8 Conclusion

In this paper, we have proposed a tensor-based algorithm for high-order graph matching in computer vision applications. We have reached state-of-the-art performance with simple potentials that are invariant to rigid, affine or projective image transformations. This work can be extended in a number of ways, for example by considering more complex features based on three, four or even more point or line features to be fully invariant to richer classes of transformations. It would also be natural to follow the approach of [7] and learn potentials automatically from labeled or partially labeled data.

## Acknowledgments

We would like to thank Choi Ouk for very helpful discussions, which has inspired a lot this article. This paper was supported

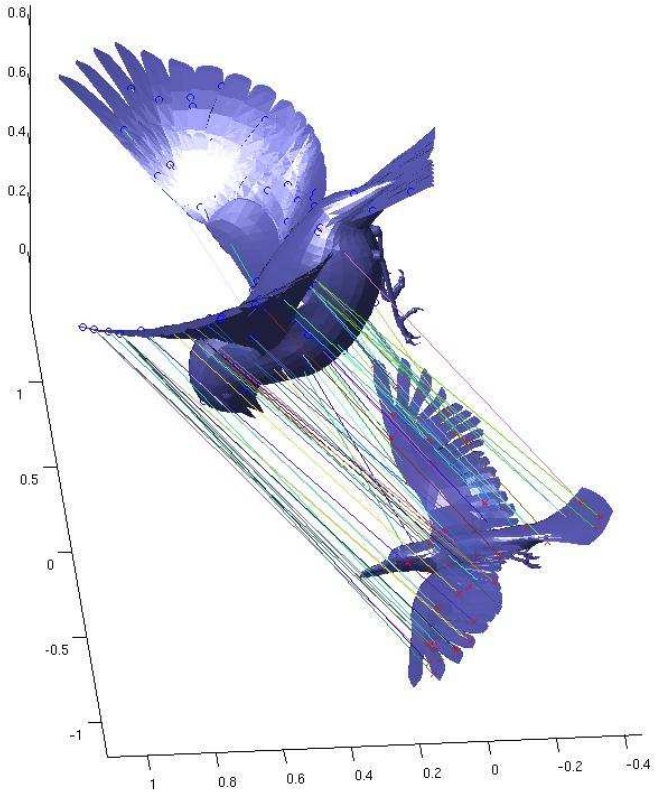

Fig. 11. Random points are selected on each of the two 3D models, and then matched.

in part by a grant from the Agence Nationale de la Recherche (MGA Project), by the ERC grants SIERRA and VideoWorld, by the Korea Association of Robot industry(KAR) grant funded by the Korea government(MKE) (No.10031903), and by the Defense Acquisition Program Administration and Agency for Defense Development, Korea, through the Image Information Research Center at KAIST.

## REFERENCES

[1] http://archive3d.net/.
[2] http://vasc.ri.cmu.edu//idb/html/motion/house/index.html.
[3] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE PAMI*, 15, 1993.
[4] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
[5] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, volume II, pages 435–439, 2005.
[6] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, 1998.
[7] T. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *ICCV*, 2007.
[8] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS 19*, 2007.
[9] Olivier Duchenne, Francis Bach, InSo Kweon, , and Jean Ponce. A tensor-based algorithm for high-order graph matching. In *CVPR*, 2009.

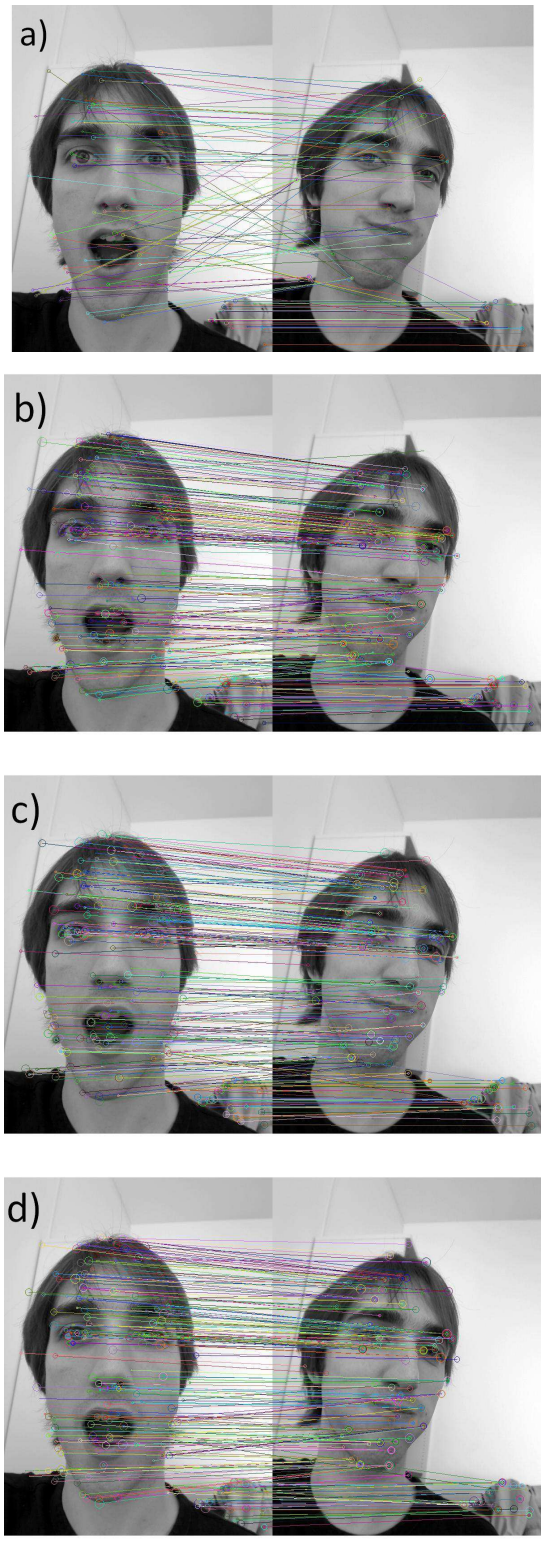


Fig. 12. (Best seen in color.) Matching of two different pictures with a) SIFT only, b) tensor matching only, c) combined, and d) combined with dummy nodes

[10] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, 1981.

[11] G. Frobenius. Ueber matrizen aus nicht negativen elementen. *Sitzungsber. Knigl. Preuss. Akad. Wiss.*, page 456477, 1912.

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

[13] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[14] W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *PAMI*, 9(4):469–482, 1987.

[15] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *ICCV*, 1987.

[16] P. Kohli, P. Mudigonda, and P. Torr. $p^3$ and beyond: Solving energies with higher order cliques. *CVPR*, 2007.

[17] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[19] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[20] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

[21] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(4):91–110, 2004.

[22] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[23] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *PAMI*, 25(2), 2003.

[24] David M. Mount and Sunil Arya. http://www.cs.umd.edu/ mount/ann/.

[25] R. Oliveira, R. Ferreira, and J. Costeira. Optimal multi-frame correspondence with assignment tensors. In *Proc. European Conf. Comp. Vision*, 2006.

[26] I. Pratikakis, M. Spagnuolo, T. Theoharis, R. Veltkamp (editors, A. Godil, H. Dutagaci, C. Akgl, A. Axenopoulos, B. Bustos, M. Chaouch, P. Daras, T. Furuya, S. Kreft, Z. Lian, T. Napolon, A. Mademlis, R. Ohbuchi, P. L. Rosin, B. Sankur, T. Schreck, X. Sun, M. Tezuka, A. Verroust-blondet, M. Walter, and Y. Yemez. Shrec09 track: Generic shape retrieval.

[27] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *ICCV*, 1998.

[28] P. A. Regalia and E. Kofidis. The higher-order power method revisited: convergence proofs and effective initialization. *ICASSP*, 2000.

[29] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–535, 1997.

[30] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *PAMI*, 10(5):695–703, 1988.

[31] A. Vedaldi. A matlab implementation of sift, 2008.

[32] M. Zaslavskiy, F. Bach, and J.P. Vert. A path following algorithm for the graph matching problem. 31(12):2227–2242, December 2009.

[33] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *CVPR*, 2008.

[34] H. Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

[35] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: An in-depth study. *Int. J. of Comp. Vision*, 73(2):213–238, 2007.

[36] Y. Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *PAMI*, 28(4):643, 2006.


**Olivier Duchenne** received the M.S. degree in Computer Science and Applied Mathematics in École Normale Supérieure (ENS), in Paris in 2008. He then joined the research team, WILLOW in the same university under the supervision of professor Jean Ponce. He received the best student paper, honorable mention, at the conference ieee CVPR 2009, in Miami. His research interests include computer vision, graph matching and video action recognition.

**Francis Bach** received the graduate degree from the Ecole Polytechnique, Palaiseau, France, in 1997, and the PhD degree from the Computer Science Division at the University of California, Berkeley, in 2005. He leads the Sierra INRIA Project Team at the Computer Science Department of the École Normale Supérieure, Paris, France. His research interests include machine learning, optimization, graphical models, kernel methods, sparse methods, and statistical signal processing.

**Inso Kweon** received the B.S. and the M.S. degrees in Mechanical Design and Production Engineering from Seoul National University, Seoul, Korea, in 1981 and 1983, respectively, and the M.S. and Ph.D. degree in Robotics from the Robotics Institute at Carnegie Mellon University, Pittsburgh, U.S.A, in 1986 and 1990,respectively. He worked for Toshiba R&D Center, Japan, and joined the Department of Automation and Design Engineering at KAIST in 1992. He is now a Professor in the Department of Electrical Engineering at KAIST. His research interests are in computer vision, robotics, pattern recognition, and automation. Specific research topics include invariant based visions for recognition and assembly, 3D sensors and range data analysis, color modeling and analysis, robust edge detection, and moving object segmentation and tracking. He is a member of ICASE, IEEE, and ACM.

**Jean Ponce** is a Professor at École Normale Supérieure (ENS) in Paris, France, where he leads a joint ENS/INRIA/CNRS research team, WILLOW, that focuses on computer vision and machine learning. Prior to this, he served for over 15 years on the faculty of the Department of Computer Science and the Beckman Institute at the University of Illinois at Urbana-Champaign. Dr. Ponce is the author of over 150 technical publications, including the textbook “Computer Vision: A Modern Approach”, in collaboration with David Forsyth. He is a member of the Editorial Boards of Foundations and Trends in Computer Graphics and Vision, the International Journal of Computer Vision, and the SIAM Journal on Imaging Sciences. He was also editor-in-chief of the International Journal on Computer Vision (2003-2008), an Associate Editor of the IEEE Transactions on Robotics and Automation (1996-2001), and an Area Editor of Computer Vision and Image Understanding (1994-2000). Dr. Ponce was Program Chair of the 1997 IEEE Conference on Computer Vision and Pattern Recognition and served as General Chair of the year 2000 edition of this conference. In 2003, he was named an IEEE Fellow for his contributions to Computer Vision, and he received a US patent for the development of a robotic parts feeder. In 2008, he served as General Chair for the European Conference on Computer Vision.