Probabilistic Graph and Hypergraph Matching

Ron Zass and Amnon Shashua
School of Computer Science and Engineering
The Hebrew University of Jerusalem
zass,shashua@cs.huji.ac.il

Abstract

We consider the problem of finding a matching between two sets of features, given complex relations among them, going beyond pairwise. Each feature set is modeled by a hypergraph where the complex relations are represented by hyper-edges. A match between the feature sets is then modeled as a hypergraph matching problem. We derive the hyper-graph matching problem in a probabilistic setting represented by a convex optimization. First, we formalize a soft matching criterion that emerges from a probabilistic interpretation of the problem input and output, as opposed to previous methods that treat soft matching as a mere relaxation of the hard matching problem. Second, the model induces an algebraic relation between the hyper-edge weight matrix and the desired vertex-to-vertex probabilistic matching. Third, the model explains some of the graph matching normalization proposed in the past on a heuristic basis such as doubly stochastic normalizations of the edge weights. A key benefit of the model is that the global optimum of the matching criteria can be found via an iterative successive projection algorithm. The algorithm reduces to the well known Sinkhorn [15] row/column matrix normalization procedure in the special case when the two graphs have the same number of vertices and a complete matching is desired. Another benefit of our model is the straightforward scalability from graphs to hyper-graphs.

1. Introduction

Object representations through graph and hypergraph is very useful and popular for applications of tracking, query to image database matching and visual recognition in general. For an overview of recent literature, we refer the reader to a number of special issues that appeared on the subject [1, 2, 3] and the survey article [7].

A hypergraph representation allows us to exploit model relationships between different objects, or between object features under consideration. Normally nodes (vertices) represent features, object parts or elementary units of consideration for the task at hand, and hyper-edges represent a relationship among a tuple of nodes. In a graph, the relationship is pairwise in nature (as edges are defined by pairs of nodes), whereas in a hypergraph the relationships are multi-faceted involving multiple features at a time.

The hypergraph matching problem seeks a vertex-to-vertex mapping between two hypergraphs such that the overall discrepancy between the corresponding matching hyper-edges is minimized. Since the number of nodes in the two hypergraphs are not necessarily equal, the matching problem includes finding an optimal matching sub-graph under the considerations above. For example, in an image tracking problem, we would like to find the largest subset of pixels in the tracked area of consideration which preserves as much as possible the inter-pixel relations. Likewise, in an object recognition task, objects can undergo certain global transformations (such as affine) followed by some non-rigid deformations, the valence $d = n + 1$ (number of vertices in a hyper-edge) of the hyper-edge commensurate with the minimal number n of points required for resolving the global transformation. A hyper-graph matching then seeks an alignment between the query and an image in which a large proportion of hyper-edges agree on the same or similar global transformation.

On the other hand, there exist a number of problems with the use of graph matching. First, we notice the high computational complexity of many operations on graphs. For example, computing the similarity of two graphs is typically exponential in the number of nodes of the two graphs involved. Secondly, the repository of algorithmic procedures in the graph domain is quite limited when compared to the tools available for alternative representations such as vector, bag-of-features, and so forth.

In this paper we derive the hyper-graph matching problem in a probabilistic setting represented by a convex optimization. First, we formalize a soft matching criterion that emerges from a probabilistic interpretation of the problem input and output, as opposed to previous methods that treat soft matching as a mere relaxation of the hard matching problem. Second, the model induces an algebraic rela-

tion between the hyper-edge weight matrix and the desired vertex-to-vertex probabilistic matching. Third, the model explains some of the graph matching normalization proposed in the past on a heuristic basis such as doubly stochastic normalizations of the edge weights. A key benefit of the model is that the *global* optimum of the matching criteria can be found via an iterative successive projection algorithm. The algorithm reduces to the well known Sinkhorn [15] row/column matrix normalization procedure in the special case when the two graphs have the same number of vertices and a complete matching is desired. Another benefit of our model is the straightforward scalability from graphs to hyper-graphs.

1.1. Related Work

Matching algorithms can be categorized into *exact* and *inexact* methods, where in the former one seeks a matching in which all matched (hyper)edges agree, and the later allows some inconsistency in matched edges. In other words, exact matching is the problem of finding the largest sub-isomorphism (or monomorphism) between graphs. Therefore, the exact algorithms focus on heuristics that may accelerate the isomorphism search. While all hypergraph matching algorithms to date are of the exact type, there is high interest in inexact graph matching. In the inexact version, the complete consistency criterion is replaced by a matching cost to be minimized. The inexact matching problem minimizes non-convex cost functions over discrete constraints. A popular line of work is the one of continuous relaxation of the discrete constraints.

The inexact algorithms differ in the way they minimize the non-convex cost over the relaxed constraints. [10] approximate the non-convex cost by a series of linear approximations using Taylor expansions of the cost function about the current approximated solution: [14] approximate the continuous rank-1 matching problem by relaxing the rank-1 constraint into a positive semidefinite constraint, ending with an SDP optimization problem. The matching is then obtained from the leading eigenvector of the SDP solution; [11] approximate the non-convex quadratic cost as an eigenvector problem, by further relaxations of the constraints; [8] extend this spectral method by incorporating more of the original constraints while staying in the eigenvector type of problems, and therefore obtaining better approximation of the original problem. In addition, [8] also suggest a preprocessing on the input, normalizing it into a doubly stochastic matrix, improving any continuous matching algorithm.

We note that some authors have dealt with finding probabilistic matching between graphs, either by applying a post-processing procedure on the results to obtain probabilities (cf. [8]), by directly searching for a doubly stochastic matching matrix (cf. [14, 18]) or by seeking a non-negative matching matrix that is interpreted as soft matching (cf. [11]). Our work differs substantially in that manner, as we assume a probabilistic interpretation of the problem input and exploit connection between input and output probabilities to extract probabilistic matching, rather than enforcing probabilistic structure on the output as a relaxation of the hard matching problem.

From an algorithmic point of view, our method has some resemblance to the work of [18] and to Softassign [10]. In [18], iterative projections onto convex domain is used to solve an approximate relaxed problem. The use of iterative projections is common to our work. However, by using the Relative Entropy error we avoid the non-convex criteria function and the following convex approximation, thereby finding the global optimal solution. Softassign [10] uses the Sinkhorn algorithm on a series of matrices indexed by a control variable to make the soft matching matrix discrete in a deterministic annealing scheme. The parametric input to the Sinkhorn algorithm is obtained through a Taylor expansion approximation of the graph matching problem, which come down to a marginalization of the edge-to-edge matching score matrix weighted by the current matching matrix. Our method differs in two ways. First, we use marginalization without weights, showing that this gives the optimal soft matching solution without need of for the repeated approximations that ends in non-optimal solution. Second, we extend the Sinkhorn algorithm to deal with inequalities in an optimal way, while Softassign deal with inequalities by adding slack nodes which again lead to non-optimal solution.

2. A Bottom-up Probabilistic View of Hypergraph Matching

Let $G = (V, E)$ and $G' = (V', E')$ be two hypergraphs where hyper-edges correspond to a d-tuple of vertices. A directed hypergraph is when the order of vertices in the d-tuple matters unlike an undirected hypergraph when the order does not matter, i.e., there is one hyper-edge per d-tuple. Our derivations and discussion apply to both cases.

The set of vertices of the hypergraph G is represented by V and its hyper-edges by $E \subseteq V^d$. A hyper-edge $e \in E$ is denoted by $e = (v_{i_1}, ..., v_{i_d})$ where $v_{i_j} \in V$. A matching between G and G' is a vertex to vertex mapping $m : V \rightarrow V'$. The vertex matching induces an edge matching, $m : E \rightarrow E'$, as $m(e) = (m(v_{i_1}), ..., m(v_{i_d})) \in E'$.

The input to the hypergraph matching problem are weights among pairs of hyper-edges from the two hypergraphs representing a probability that the two match. These are represented by a $|V|^d \times |V'|^d$ matrix S with entries:

$$S_{e,e'} = Pr(m(e) = e' \mid G, G').$$

The output of the (soft) hypergraph matching problem is the probability that two vertices match, represented by a $|V| \times$

$|V'|$ matrix X whose entries are:

$$X_{v,v'} = Pr(m(v) = v' \mid G, G').$$

For X to be a valid soft matching it has to be *Doubly Semi Stochastic (d.s.s.)*, i.e., $X \geq 0,\ X\mathbf{1} \leq \mathbf{1},\ X^\top \mathbf{1} \leq \mathbf{1}$, where $\mathbf{1}$ is a vector of ones. The later two inequalities become equalities, and X becomes doubly stochastic, when $|V| = |V'|$ and all vertices are matched. *Our objective is to derive an algebraic relation between the input S and the output X.* We will use the algebraic relation to setup an optimization problem for solving for the optimal X given S.

Note that given we have found the optimal X, if one desires a hard matching result $\bar{X}$ whose entries are in $\{0, 1\}$, then the optimal approach involves a solution to a *linear assignment problem* (cf. [5]). The hard matching result is not optimal overall as there could be a different route to a better result that does not go through an optimal soft assignment (after all, the hard matching problem is NP hard).

Note that by setting up the probabilistic framework as we have done above, we have already departed from the standard approach, which starts from the definition of a *hard* matching and via relaxation of the optimization constraints obtain some form of soft matching result. We, on the other hand, took the other way around: we start with a direct definition of a soft matching problem and (as we shall see later) obtain a framework through which we could achieve an optimal probabilistic matching. Then, if a hard matching is desired we solve a linear assignment problem to find the closest permutation matrix.

The advantage of setting up the soft matching problem as the first step is that in many applications of interest (like in tracking) we would like a probabilistic matching to begin with, i.e., to keep holding the matching ambiguities until a later stage where more data comes in to disambiguate the matching. Moreover, as we shall see, we can set up the framework such that we would be guaranteed to obtain the optimal soft matching solution.

The connection between the input matrix S and the desired output X can be described compactly if we make the following assumption: $m(v_i) \perp m(v_j) \mid G, G'$, i.e., that the matches are pairwise conditionally independent. As we shall see next, this assumption not only leads to a simple and elegant connection between S and X, but also explains normalization heuristics on the input S proposed in the past.

Before we state our result, we need one more preparation which is the definition of the Kronecker product between matrices resulting in a block matrix. If A is an $p \times q$ matrix and B is a $r \times s$ matrix, the Kronecker product $A \otimes B$ is the $pr \times qs$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1q}B \\ \vdots & \ddots & \vdots \\ a_{p1}B & \dots & a_{pq}B \end{bmatrix}$$

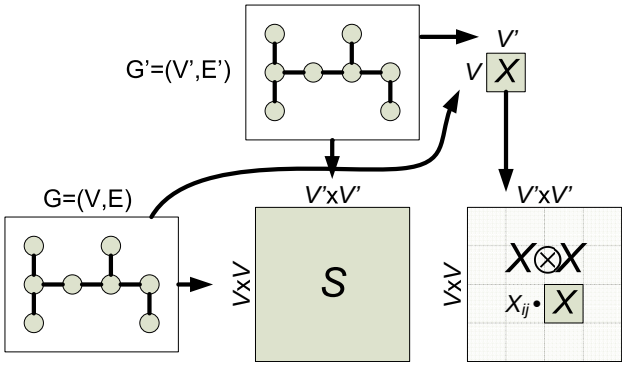


Figure 1. A visualization of eqn. 1 for the case of graphs ($d = 2$). The matrix S holds the probability for any possible match between the edges of G and the edges of G'. The matrix X holds the unknown probability of matching between vertex of G and vertex of G'. The matrix $X \otimes X$ holds the probability of matching two vertices in G to two vertices in G'. Under conditional independence assumption, this is the probability of matching the corresponding edges, which is exactly the probability given in S.

We also define the following shorthand notations: $\otimes_{i=1}^{d} A_i = A_1 \otimes ... \otimes A_d$ and $\otimes^d A = \otimes_{i=1}^{d} A$. We are now ready to state the connection between S and X:

Proposition 1 *Under the conditional independence assumption of the matching process, $m(v_i) \perp m(v_j) \mid G, G'$, the following result holds:*

$$S = \otimes^d X. \tag{1}$$

Proof:

$$\begin{aligned} S_{e,e'} &= Pr\left(m(e) = e' | G, G'\right) = \\ &= \prod_{i=1}^{d} Pr\left(m(v_i) = v_i' | G, G'\right) = \\ &= \prod_{i=1}^{d} X_{v_i, v_i'}, \end{aligned}$$

where $e = (v_1, ..., v_d)$ and $e' = (v_1', ..., v_d')$. By definition of the Kronecker product of matrices, we have $S = \otimes^d X$. □

Fig. 1 visualizes this connection for the case of graphs ($d = 2$). Taken together, we define an optimization problem for recovering X from S by minimizing the distance (to be defined) between S and $\otimes^d X$:

$$\begin{aligned} \min_X \quad & dist\left(S, \otimes^d X\right) \\ s.t. \quad & X\mathbf{1} \leq \mathbf{1},\ X^T\mathbf{1} \leq \mathbf{1},\ X \geq 0, \end{aligned} \tag{2}$$

where the distance function would be discussed in the sequel. It is worthwhile noting that the d.s.s property on X occurs in tandem with $\otimes^d X$, as stated below:

Proposition 2 *Let $X \geq 0$ be a non-negative matrix, then*

- *X satisfies $X\mathbf{1} \leq \mathbf{1},\ X^T\mathbf{1} \leq \mathbf{1}$ iff the matrix $\otimes^d X$ satisfies $(\otimes^d X)\mathbf{1} \leq \mathbf{1},\ (\otimes^d X)^T\mathbf{1} \leq \mathbf{1}$.*
- *$X\mathbf{1} = \mathbf{1}$ iff $(\otimes^d X)\mathbf{1} = \mathbf{1}$.*
- *$X^T\mathbf{1} = \mathbf{1}$ iff $(\otimes^d X)^T\mathbf{1} = \mathbf{1}$.*

The proof is straightforward and thus omitted. The proposition indicates that if S happens to be d.s.s and we find $X \geq 0$ such that $S = \otimes^d X$, then X is d.s.s as well. This gives a theoretical justification to the heuristic used by [8], proposing to replace the matrix S with the nearest (in Relative Entropy) stochastic matrix to S before performing graph matching, showing improved results for many graph matching algorithms.

We have established a statistically valid connection between input and output of the soft matching problem. We note that this connection is at the hurt of the traditional hard matching problem as well. Adding orthogonality to our constraints restricts the valid matching results to hard matching, $X_{v,v'} = 1$ if $m(v) = v'$ and 0 otherwise. Plugging this into the criterion in eq. 2 and using the Frobenius error norm as a distance measure, the problem is equivalent to maximizing $\sum_{e'=m(e)} S_{e,e'}$, which is the traditional inexact hard matching problem.

In the next section we show how to find X, in the context of a relative entropy error measure, as the global optimum of the optimization problem above.

3. Globally Optimal Soft Matching

In the section above we have ended with the conclusion that an optimal probabilistic vertex matching between two hypergraphs G, G' can be obtained by finding a d.s.s matrix X such that $\otimes^d X$ is nearest to S. In this section we derive the solution in the context of the relative entropy error measure (maximum likelihood). We setup the problem as a constrained convex optimization with a dual block update algorithm. The block update algorithm reduces to the well known Sinkhorn successive row/column iterative approximation when $|V| = |V'|$.

Eqn. 2 with a relative entropy error measure,

$$D(A||B) = \sum_{i,j} A_{i,j} log\left(\frac{A_{i,j}}{B_{i,j}}\right) - A_{i,j} + B_{i,j},$$

takes the following form:

$$\begin{aligned} X^* &= \underset{X\geq 0}{\operatorname{argmin}}\ D\left(S||\otimes^d X\right) \\ &\quad s.t.\quad X\mathbf{1} \leq \mathbf{1},\ X^T\mathbf{1} \leq \mathbf{1}. \end{aligned} \tag{3}$$

The criterion function can be reduced to a much simpler form by defining a marginalization $|V| \times |V'|$ matrix Y defined below:

$$Y_{v,v'} = \sum_{i=1}^{d} \sum_{\substack{e \mid e_i = v \\ e' \mid e'_i = v'}} S_{e,e'},$$

where e_i is the i'th vertex in the d-tuple associated with the hyper-edge e (and likewise for e'_i). It can be verified that Y contains all the information necessary to find X^* and that the optimization problem below is equivalent to eqn. 3:

$$\begin{aligned} X^* &= \underset{X\geq 0}{\operatorname{argmin}}\ D(Y\ ||\ X) + \left(\mathbf{1}^T X\mathbf{1}\right)^d - \mathbf{1}^T X\mathbf{1} \\ &\quad s.t.\quad X\mathbf{1} \leq \mathbf{1},\ X^T\mathbf{1} \leq \mathbf{1}. \end{aligned} \tag{4}$$

Although the problem above is convex, it can be simplified further by setting the value of $\mathbf{1}^T X\mathbf{1}$ to some fixed value k (which represents the total number of matches) and solve for $X^*(k)$:

$$\begin{aligned} X^*(k) &= \underset{X\geq 0}{\operatorname{argmin}}\ D(Y\ ||\ X) \\ &\quad s.t.\quad X\mathbf{1} \leq \mathbf{1},\ X^T\mathbf{1} \leq \mathbf{1}, \mathbf{1}^T X\mathbf{1} = k \end{aligned} \tag{5}$$

Since $X^*(k)$ as a function of $k = \mathbf{1}^T X\mathbf{1} \geq 0$ is convex in k, we can use this solution for minimizing over $0 \leq k \leq min(|V|, |V'|)$, a single variable convex problem that can be solved numerically.

It is worthwhile noting similarities to special cases that have been handled in the past. When the row and column sum of the matching matrix X are given in advance, meaning we know which of the vertices are being matched, we are left with minimizing $D(Y||X)$ under column and row sum *equality* constraints. This can be solved using the Menon extension [12] of the Sinkhorn algorithm [15] for finding the nearest doubly stochastic matrix in relative entropy. Softassign [10], for instance, uses the Sinkhorn algorithm on a weighted version of the Y matrix after extending it with slack variables, and then update the weights iteratively.

We turn our attention to the algorithm for recovering $X^*(k)$ for a fixed given value of k, i.e., the algorithm for finding the optimal solution for eqn. 5.

Eqn. 5 is a special case of the general convex problem $\min_X f(X)$ where $X \in C_1 \cap C_2 \cap C_3$. In our case, the convex criterion is $f(X) = D(Y\ ||\ X)$, and the convex sets C_j are defined by:

$$\begin{aligned} C_1 &= \{X \mid X \geq 0,\ X\mathbf{1} \leq \mathbf{1}\} \\ C_2 &= \{X \mid X \geq 0,\ X^\top\mathbf{1} \leq \mathbf{1}\} \\ C_3 &= \{X \mid X \geq 0,\ \mathbf{1}^\top X\mathbf{1} = k\} \end{aligned}$$

We define a sub-problem operator $P_j(H)$ for $j = 1, 2, 3$ as follows:

$$P_j(H) = \underset{X \in C_j}{\operatorname{argmin}} f(X) - \langle X, H \rangle ,$$

where $\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$ is the dot product between two matrices. By repeated applications of $P_j(H)$ in a cyclic manner we obtain a primal-dual block update algorithm, for finding the globally optimal solution X, defined below.

Algorithm 1 (Successive Projections)

- *Define* $\boldsymbol{\lambda}_j^{(0)}, X_j^{(0)} \in R^{|V| \times |V'|}$, $j = 1, 2, 3$ *and set* $\boldsymbol{\lambda}_j^{(0)} = 0$ *and* $X_3^{(0)} = Y$. *Use the convention* $X_0^{(t)} \equiv X_3^{(t-1)}$.
- *Iterate on* $t = 1, 2, ...$ *till convergence:*
  - *For* $j = 1, 2, 3$*:*
    - $X_j^{(t)} = P_j \left(\boldsymbol{\lambda}_j^{(t-1)} + \nabla f(X_{j-1}^{(t)}) \right)$,
    - $\boldsymbol{\lambda}_j^{(t)} = \boldsymbol{\lambda}_j^{(t-1)} + \nabla f(X_{j-1}^{(t)}) - \nabla f(X_j^{(t)})$.

At convergence with T iterations, $X_j^{(T)} = X^*$, $j = 1, 2, 3$, is the optimal solution. The algorithm employs successive Bregman projections and is derived using the framework of Fenchel Duality. This algorithm is known under various names depending on the function $f(\cdot)$ and the convex sets C_j. For example, if C_j are linear subspaces and $f()$ is the L_2 distance then the matrices $\boldsymbol{\lambda}_j$ are redundant (also known as "deflections") and the scheme is attributed to Von-Neumann [13] and for general convex sets it is attributed to Dykstra [9, 4]. For general strictly convex functions and general convex sets, the algorithm above is essentially contained in [17, 6].

The key therefore is the operator $P_j(H)$ which needs to be solved at each step of the algorithm. It can be verified that for each of the sets C_j, finding the optimal X that minimizes $D(Y \mid\mid X) - trace(H^\top X)$ under each of the constraint sets C_j is a straightforward optimization problem with a closed form solution. It is also worthwhile noting that when $|V| = |V'|$, the constraints on X becomes $X\mathbf{1} = X^\top \mathbf{1} = \mathbf{1}$, the deflections drop out, and the algorithm reduces to the well known Sinkhorn [15] successive rows and columns approximation algorithm.

4. Sampling

The matching algorithm solves a matrix nearest problem with Y as an input. Since Y is in $R^{|V| \times |V'|}$, the time complexity of the algorithm depend on the number of vertices rather than the number of hyper-edges, and therefore independent of the hypergraph degree. However, in order to calculate Y we have to marginalize S, the hyper-edge to hyper-edge correlation matrix. For cases where a complete calculation of S is impractical or unwarranted, we suggest a smart sampling scheme of S for efficient approximation of Y. We sample z hyper-edges per vertex. When coming to calculate $Y_{v,v'}$, we use only the correlations between sampled hyper-edges. In the sampling process we have to ensure that for any two matching vertices, matching hyper-edges will be sampled, a non-trivial task for high degree hypergraphs. We therefore base the sampling on the heuristic that hyper-edges that involve vertices of close proximity in one object, are probable to translate to hyper-edges of close proximity in the second object as well. Therefore, for each vertex we sample the z closest hyper-edges (measured as the mean distance of the involved vertices). This sampling is also appealing when using hypergraph matching to match two non-rigid body, as it incorporate the assumption that small regions goes an approximated affine transformation. We note that other sampling heuristic may be considered, such as sampling based on vertex to vertex similarity (such as SIFT descriptors). Sampling only z hyper-edges per vertex, we calculate Y at $O\left(|V| \cdot |V'| \cdot z^2\right)$ steps.

5. Experiments

Although our main contribution is inexact hypergraph matching[1], we start our experiments with a graph matching ($d = 2$) experiment. This allows us to compare our method with other inexact graph matching algorithms, as there are no other inexact hypergraph matching algorithms in the literature. We compare our graph matching results with the spectral technique of [11], with and without the balancing normalization suggested by [8]. In order to have ground truth information to measure against, we use synthetic dataset. We generate a cloud of 25 points, uniformly distributed in 2D space, with mean distance between neighboring points normalized to 1. We produce a rotated duplicate of the point cloud, while perturbing the point locations with a random distortion. The two point sets translates into two graphs, where edges are labeled with the Euclidean distance between their two vertices. We use $S_{e,e'} = \exp^{-|d-d'|}$, where d is the distance associated with the edge e, as the edge-to-edge similarity measure. Our task is to recover the point-to-point matching. In fig. 2(a) we plot the number of correct matches as a function of the average point location perturbation. Although the Frobenius error norm that is used by spectral matching is known to better handle additive noise, our scheme achieves comparable results. Next, we add additional points that do not have a match, and measure the amount of correct matches as a function of the number of additional points (without perturbing the point locations). In fig. 2(b) we add points only

[1] Complete implementation can be found at `www.cs.huji.ac.il/~zass/gm/`

to the second graph, and in fig. 2(c) we add points without match to both graphs, obtaining significantly better matching when using our method.

To demonstrate the disadvantage of graph matching, we repeat the test of fig. 2(a), where this time the second cloud of points is the result of an affine transformation (that does not preserve distances) of the first cloud, followed by a perturbation of the point locations. We compare the three graph matching algorithms, and a hypergraph matching version of the problem using our scheme. Throughout our experiments, we use hypergraphs of degree $d = 4$. Each hyperedge, a 4-tuple of points, (v_1, v_2, v_3, v_4), is associated with an affine invariant measure. In particular, we use the ratio between the areas of the v_1, v_2, v_3 and the v_2, v_3, v_4 triangles. The correlations between two hyper-edges, $e \in E$ and $e' \in E'$ with an associated ratio of r and r', is set to $\exp^{-|r-r'|^2}$. Fig. 2(d) shows the improved results of our hypergraph matching algorithm, when compared with any of the graph matching algorithms used. In fig. 2(e-f) we repeat the experiments of fig. 2(b-c), that deal with additional points that have no match, for the case of affine translated point set, demonstrating the advantage of using hypergraphs matching. Next we test the effect of the number of sampled hyper-edges per vertex on the hypergraph matching results. We use the above settings, with average perturbation set to one in fig. 2(g), and with 50 additional points that have no match to one of the graphs in fig. 2(h), demonstrating that the hypergraph matching results can be further improved by increasing the number of samples (recall that we used only 60 samples per vertex).

Before going to image related experiments, we compare the run-time of our algorithm for the case of graphs with the run-time of spectral matching, that is considered a fast continuous graph matching algorithm [8]. Fig. 3(a) shows the net time (without counting time spend on calculating correspondences), demonstrating the efficiency of our scheme. In fig. 3(b) we study the gross run time (including correspondences calculations) of our scheme for two hypergraphs of degree 4 with 50 points each, when the number of samples per vertex vary.

We move next to image matching examples. Before going to non-rigid bodies, we demonstrate the inability of graph matching to cope with even simple and global affine transformation of an image, where hypergraph matching performs adequately. Fig. 4 shows the matching results between a pair of images[2], taken from different views of the same scene. Hypergraph matching restores all matches correctly, while spectral graph matching produces 10 wrong matches out of a total of 33.

Finally, we demonstrate our scheme on several challenging non-rigid point matching examples. We use the point tracking results of [16], that track non-rigid bodies over im-

[2]http://www.robots.ox.ac.uk/∼vgg/research/affine/index.html

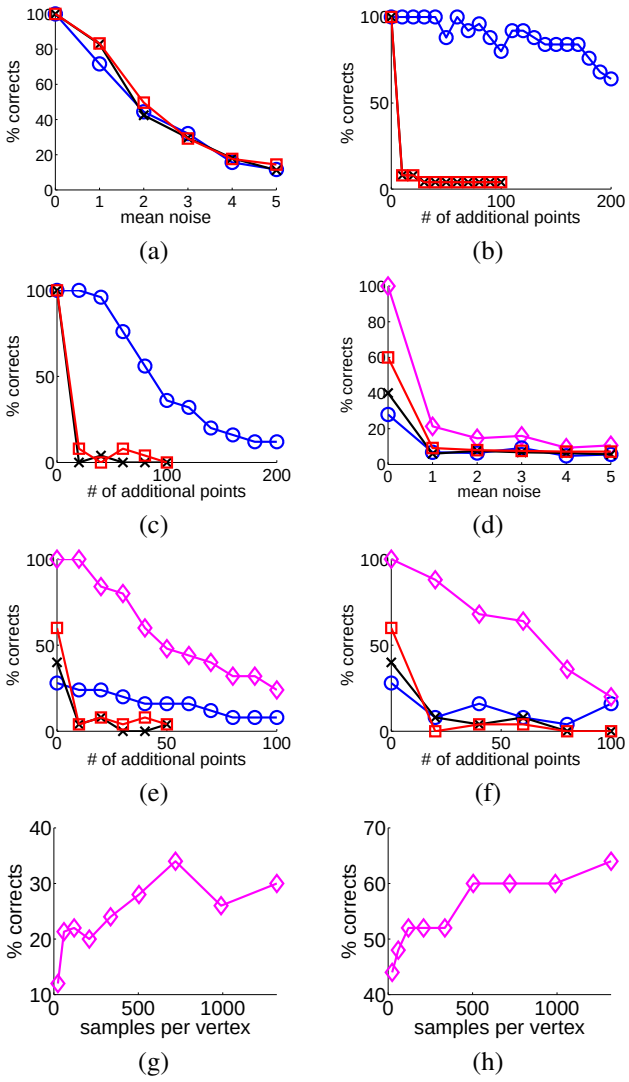
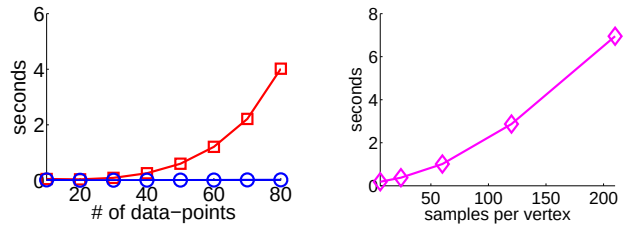


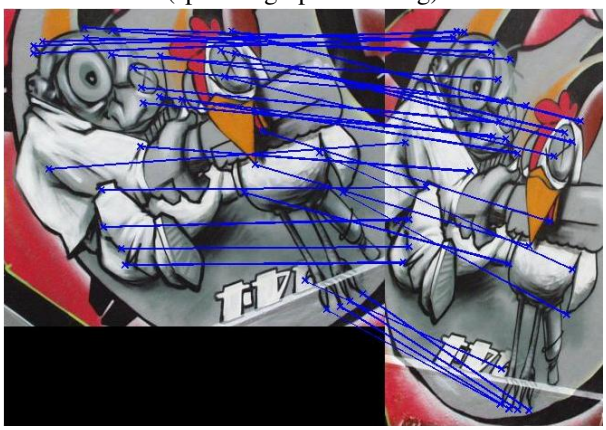
Figure 2. Synthetic experiment with ground truth. A cloud of 25 points is going through a transformation. We first deal with a transformation that preserves distances. (a) Our scheme (with $d = 2$, in blue o) compared with spectral graph matching with (red □) and without (black ×) prior normalization in the present of random perturbation of point locations (average distance between neighboring points is 1). Although the Frobenius norm used by spectral matching is known to function better under additive noise, the global solution of our relative entropy scheme achieves comparable results. (b) Our scheme give superior results when additional points that have no match are added to one of the points set, (c) or to both points sets. In (d-f) we repeat these experiments with a general affine transformation that does not preserve distances, demonstrating the power of hypergraph matching ($d = 4$, in magenta ◇) in such cases. Next, we demonstrate how results can be improved by using more samples per vertex: (g) with perturbation of point locations by an average of 1, and (h) when 50 additional points that have no match are added to one of the graphs.



(a) (b)

Figure 3. (a) Running time of our scheme (with $d = 2$, in blue o) and spectral graph matching (red □), when the number of points varies. Times do not include calculation of correlations. While spectral matching is considered one of the more efficient matching algorithms, our scheme highly outperforms it. (b) Running time of our scheme ($d = 4$) with sampling, when the number of samples per vertex varies, this time including the correlations time, as it changes with the number of samples.

(spectral graph matching)



(hypergraph matching)

Figure 4. Matching images that relate by an affine transformation. Hypergraph matching produces all matches correctly, while spectral graph matching produces 10 (out of 33) wrong matches (wrong matches are in red).

ages sequences[3]. We take only the first and last frame of each sequence, ending with a much more challenging task than the original tracking problem. While the non-rigid bodies do not fall into the affine transformation scheme, our sampling allows our scheme to exploit local transformations that are approximately affine in order to extract correct matches. Fig. 5 shows the results of matching points from the two end frames. While not all points are matched, the large majority of the matches are visually correct.

6. Summary

We have presented a probabilistic model for soft hypergraph matching between complex feature sets. The model induces an algebraic relation between the hyper-edge weight matrix and the desired vertex-to-vertex probabilistic matching which scales naturally from graphs to hypergrpahs. A key benefit of the model is that the *global* optimum of the matching criteria can be found via an iterative successive projection algorithm. The algorithm reduces to the well known Sinkhorn [15] row/column matrix normalization procedure in the special case when the two graphs have the same number of vertices and a complete matching is desired. Our experiments demonstrate the effectiveness of our algorithm both as an efficient solution to the special case of graph matching, and as a new tool for non-rigid matching.

An important combinatorial challenge in hypergraph matching is the need to calculate the hyper-edge to hyper-edge correlation, which grow exponentially with the hypergraph degree. We have presented a sampling scheme that samples $O\left(|V| \cdot |V'| \cdot z^2\right)$ correlations. The efficiency of this sampling scheme has yet to be formalized and studied.

We have limited our discussion to the case where all hyperedges are of the same degree. While treating hyperedges of different degrees is straightforward from a theoretical point of view, the issue of balancing between different type of measurements, produced by different number of vertices, should be further studied.

References

[1] IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI): Special section on graph algorithms and computer vision. 23 (2001) 10401151. 1

[2] Pattern Recognition Letters: Special issue on graph based representations. 24 (2003) 10331122. 1

[3] Int. Journal of Pattern Recognition and Art. Intelligence: Special issue on graph matching in pattern recognition and computer vision. 18 (2004) 261517. 1

[4] J. Boyle and R. Dykstra. A method for finding projections onto the intersections of convex sets in Hilbert

[3] http://movement.stanford.edu/nonrig/

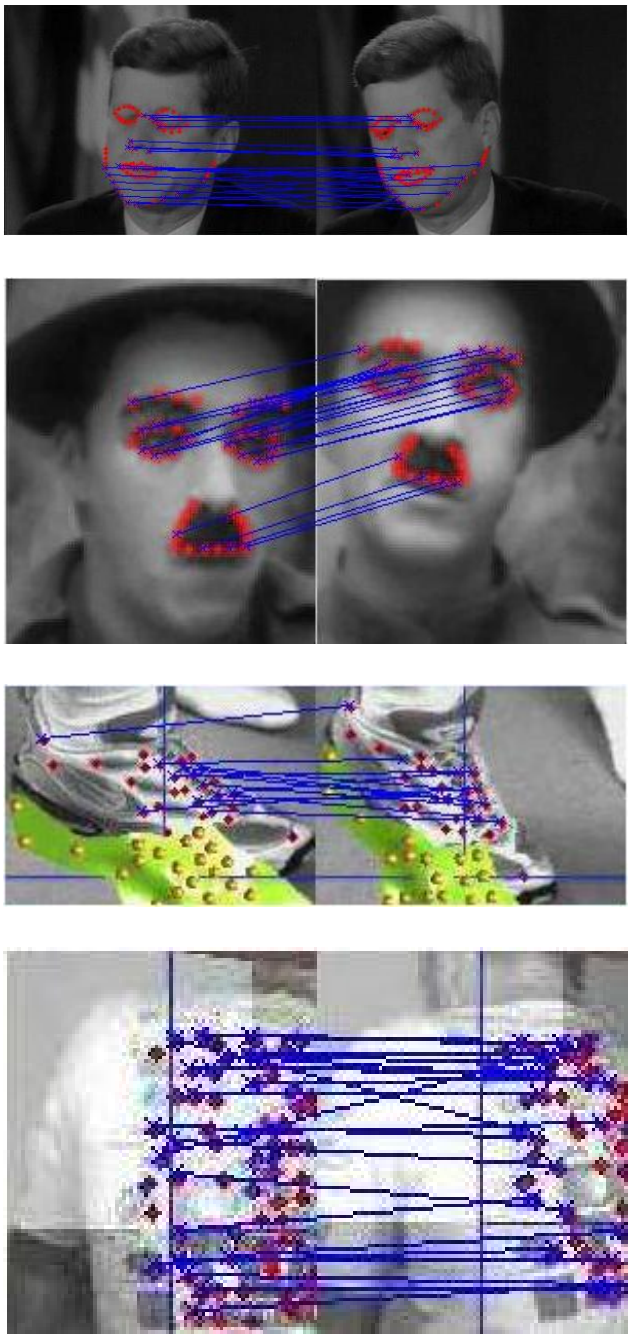


Figure 5. Non-rigid body matching. Points (in red) are tracked along an image sequence [16]. We then use only the first and last frames of the sequence, and match the two point sets. We use an affine invariant similarity, and our sampling scheme allows the matching to exploit local transformations which are approximately affine. While not all points are matched, resulting matches are visually correct.

spaces. *Advances in order restricted statistical inference, Proc. Symp., Iowa City/Iowa*, 1985. 5

[5] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM Monographs on Discrete Mathematics and Applications, in preparation. 3

[6] Y. Censor and S. Reich. The Dykstra algorithm with Bregman projections. *Communications in Applied Analysis*, 2(3):407–419, 1998. 5

[7] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *Int. Journal of Pattern Recognition and Articial Intelligence*, 18:265–298, 2004. 1

[8] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Neural Information Processing Systems (NIPS)*, pages 313–320, 2007. 2, 4, 5, 6

[9] R. Dykstra. An iterative procedure for obtaining i-projections onto the intersection of convex sets. *The Annals of Probability*, 13:975–984, 1985. 5

[10] S. Gold and A. Rangarajan. Softmax to softassign: neural network algorithms for combinatorial optimization. *J. Artif. Neural Netw.*, 2(4):381–399, 1995. 2, 4

[11] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482–1489, October 2005. 2, 5

[12] M. V. Menon. Matrix links, an extremization problem, and the reduction of a non-negative matrix to one with prescribed row and column sums. *Canadian Journal of Mathematics*, 20:225–232, 1968. 4

[13] J. V. Neumann. *Functional Operators Vol. II*. Princeton University Press, 1950. 5

[14] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *Computer Vision and Pattern Recognition (CVPR)*, 2005. 2

[15] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343348, 1967. 1, 2, 4, 5, 7

[16] L. Torresani and C. Bregler. Space-time tracking. In *ECCV*, pages 801–812, 2002. 6, 8

[17] P. Tseng. Dual coordinate ascent methods for non-strictly convex minimization. *Mathematical Programming*, 59(1):231–247, 1993. 5

[18] B. J. van Wyk and M. A. van Wyk. A pocs-based graph matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 26(11):1526–1530, 2004. 2