# Game-Theoretic Hypergraph Matching with Density Enhancement

Jian Hou[a], Huaqiang Yuan[a], Marcello Pelillo[b,c]

[a]*School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China*
[b]*DAIS, Ca' Foscari University, Venice 30172, Italy*
[c]*European Centre for Living Technology, Ca' Foscari University, Venice 30123, Italy*

**Abstract**

Feature matching plays a fundamental role in computer vision and pattern recognition. As straightforward comparison of feature descriptors is not enough to provide reliable matching results in many situations, graph matching makes use of the pairwise relationship between features to improve matching accuracy. Hypergraph matching further employs the relationship among multiple features to provide more invariance between feature correspondences. Existing hypergraph matching algorithms usually solve an assignment problem, where outliers may result in a large number of false matches. In this paper we cast the hypergraph matching problem as a non-cooperative multi-player game, and obtain the matches by extracting the evolutionary stable strategies. Our algorithm exerts a strong constraint on the consistency of obtained matches, and false matches are excluded effectively. In order to increase the number of matches without increasing the computation load evidently, we present a density enhancement method to improve the matching results. We further propose two methods to enforce the one-to-one constraint, thereby removing false matches and maintaining a high matching accuracy. Experiments with both synthetic and real datasets validate the effectiveness of our algorithm.

*Keywords:* feature matching, hypergraph matching, game-theoretic, density enhancement

## 1. Introduction

Feature matching is used to build the correspondences between features in the model and test images [1, 2]. It plays an important role in various computer vision and pattern recognition tasks, including stereo vision and object tracking [3]. In industrial applications, e.g., automatic driving, three-dimensional scene/object modeling requires a large number of matches with a high accuracy. This presents two challenges to feature matching algorithms. First, due to the differences in viewpoints and lighting conditions, etc, the model and test images and corresponding features may have evident difference in appearance. In this case, straightforward comparison of feature descriptors, e.g., SIFT [4], may not be able to generate reliable matching results. Second, as the model and test images may have evident appearance difference, and model and test features are detected from two images separately, there may not be a one-to-one correspondence between two sets of features. In other words, some features in one set may have no correspondences in the other set. We call these features as outliers. Obviously, matching outliers to features results in false matches. If the number of outliers are large and the false matches from outliers cannot be removed, the matching accuracy will be low.

Since straightforward comparison of feature descriptors may not be enough for reliable feature matching, graph matching [5, 6] is proposed to encode the geometric relationship between features in a graph, thereby providing more invariance and improving the matching accuracy [7]. Hypergraph matching further encodes the geometric relationship among multiple features in a hypergraph, making it possible to explore more invariance between the model and test features. Existing hypergraph matching algorithms, e.g., [8, 9], typically maximize the matching score between feature correspondences to obtain the optimal assignment of matches. One major problem of these algorithms is that they match all model features to test features, and cannot remove the false matches caused by outliers. In the case that the number of outliers is large, these algorithms generate a low matching accuracy.

In this work we do hypergraph matching from a totally different perspective. By treating the affinity between model and test hyperedges as the higher-order similarity of matches, we transform hypergraph matching to a non-cooperative multi-player game and obtain a group of consistent matches as the final matches. In this way we are able to remove the false matches associated with outliers and improve the matching accuracy substantially. However, we also find that this method removes not only many false matches, but also some true matches. As a result, the number of obtained matches is a little small and some features (not outliers) are left unmatched. We study the algorithm and find out the reasons behind this problem. Motivated by density-based clustering, we then propose an enhancement method to obtain more matches outside the group. Furthermore, two methods are presented to enforce the one-to-one constraint, based on which the matching accuracy is not degraded with the increase of matches. We conduct experiments with synthetic and real datasets and show that our algorithm increases matches efficiently and maintains a high matching accuracy. Some of these works appeared in a preliminary version in [10].

The contributions of this paper are as follows. First, we transform hypergraph matching to a non-cooperative multi-player game and obtain a group of consistent matches as the final matches, thereby removing false matches and improving the matching accuracy. Second, we discuss the properties of the obtained matches and propose a density enhancement method, based on which we obtain more matches efficiently. Third, we propose two methods to enforce the one-to-one constraint, thereby maintaining a high matching accuracy while obtaining more matches. Compared with the conference version in [10], the extensions are listed below. First, we investigate the influences of four parameters $n_t$, $n_n$, $MinPts$ and $\sigma$ on the matching results empirically (Figures 7, 8 and 9). As a result, we find that $n_t$ can be fixed to be 1 and discuss the reason, thereby removing the parameter $n_t$ actually. We then recommend the range of the remaining three parameters. This facilitates the application of our algorithm in real-world tasks. The introduction of these parameters and related experiments and discussions are presented in Section 3.2, 3.3 and 4.3. Second, we

show that the density enhancement method is effective in increasing the number of matches, and the enforcing of one-to-one constraint is effective in removing false matches. In other words, both the major components of our algorithm are effective. Third, we conduct experiments on both feature matching and semantic matching, and discuss the advantages and deficiencies of our algorithm and further research directions.

Noticing that the GTHM (game-theoretic hypergraph matching) algorithm proposed in [11] is also a game-theoretic approach, we discuss its difference with our algorithm as follows. First, the GTHM algorithm extracts initial matches from SIFT based matching, and then does hypergraph matching to exclude the false matches. In the case that SIFT based matching is not reliable due to large viewpoint changes or repetitive patterns, this algorithm may not be able to generate enough matches. In contrast, our algorithm compares hyperedges to obtain matches, and each model feature is associated with multiple or many candidate matches. This helps find out true matches for more features and increase the number of matches. Second, our enhancement method increases matches evidently, whereas the increase in running time is small. Third, we present two methods to enforce the one-to-one constraint, based on which the evident increase of matches does not degrade the matching accuracy. As a result, our algorithm generates evidently more matches than GTHM in experiments, while achieving comparable matching accuracy with the latter.

The MC-HDSet algorithm [12] also adopts a game-theoretic approach to hypergraph matching. In order to increase the number of matches, this algorithm proposes to extract multiple groups, instead of one single group, to obtain the final matches, and presents empirical rules to determine the number of groups. Comparatively, our algorithm expands one single group by borrowing ideas from density based clustering, and the expansion is terminated based on the density information in the original group. Furthermore, we show that the hyperedges used in computation can be reduced evidently with little influence on matching accuracy and number of matches. Consequently, the experiments show that our algorithm is able to reduce the computation load evidently while maintaining

the same high matching accuracy and number of matches as MC-HDSet.

In Section 2 we review some learning-free hypergraph matching algorithms and learning-based (hyper)graph matching approaches. Section 3 is devoted to the game-theoretic algorithm, the density-based enhancement and two methods of enforcing the one-to-one constraint. We verify the effectiveness of the proposed algorithm and make comparisons with other algorithms in Section 4. Section 5 concludes this paper.

## 2. Related works

In graph matching, one key problem is to calculate the similarities (affinities) between nodes and between edges of two graphs. Traditional learning-free graph matching algorithms usually adopt simple parametric methods, e.g., Gaussian kernel, to estimate these similarities. As our algorithm belongs to this type, in this part we mainly review some hypergraph matching algorithms of this type. In addition, there are increasing interests in learning based graph matching in recent years [13]. Therefore we also review some graph matching learning algorithms in brief.

### *2.1. Hypergraph matching without affinity learning*

Many of existing hypergraph matching algorithms maximize the matching scores between hyperedges to solve an assignment problem. The HGM algorithm [14] solves a convex optimization in a probabilistic setting, and along this line a spectral algorithm is proposed in [15]. The TM algorithm [16] represents the affinity between hyperedges with a tensor, which is also adopted in later works including the RRWHM [17] and BCA [18] algorithms. In [9], the authors present a robust algorithm to integrate geometric information based on sub-pattern structure. By decomposing hypergraph matching into bipartite matching and several belief propagation sub-problems, [8] proposes to solve the sub-problems with a dynamic programming procedure. While the majority of algorithms use only the geometric information, the appearance information is also included

in [19], where the weights of two kinds of information are learnt in a semi-supervised manner. Different from existing algorithms working in continuous space, discrete algorithms are also explored in [20, 21], which are shown to be promising in convergence and computation efficiency.

The algorithms reviewed above typically cast hypergraph matching as an assignment problem. Given the set of features $F_1 = \{\zeta_i\}, i = 1, 2, \cdots, n_1$, in the model image and $F_2 = \{\xi_j\}, j = 1, 2, \cdots, n_2$, in the test image, these algorithms yield a binary assignment matrix $X \in \{0, 1\}^{n_1 \times n_2}$, with

$$X_{ij} = \begin{cases} 1, & (\zeta_i, \xi_j) \ is \ a \ match, \\ 0, & otherwise. \end{cases} \tag{1}$$

On one hand, this mode of matching helps overcome the variance between two images and generate a large amount of true matches. On the other hand, by assigning a correspondence to each model feature, these algorithms may generate many false matches, resulting in a small matching accuracy. The majority of false matches are associated with outliers, which have no correspondences and therefore should not be matched to any features. However, the assignment-based matching forces each outlier to accept a test feature as the correspondence, resulting in false matches. Due to the differences in viewpoints and lighting conditions between two images, there may be a large amount of outliers in the model features, resulting in many false matches. To illustrate this problem, we show the matching result of the BCA algorithm on the graf image pair in Figure 1. In this example there are 57 model features extracted from the model (left) image, out of which 32 are outliers. While the BCA algorithm matches all the 25 non-outlier model features to their correspondences correctly, it also assigns correspondences to the 32 outliers, resulting in 32 false matches (yellow lines). Consequently, the matching accuracy is rather low.

### *2.2. Learning based (hyper)graph matching*

Traditional learning-free (hyper)graph matching algorithms usually estimate similarities with simple parametric methods, which may not adapt to various

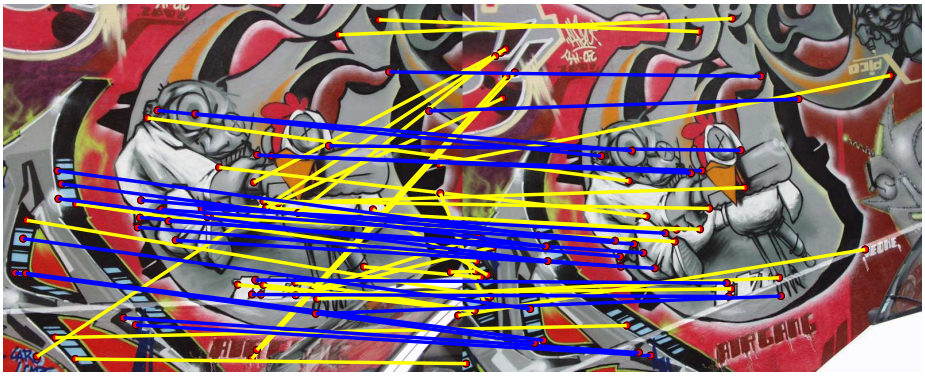


Figure 1: Demonstration of false matches caused by outliers. Blue and yellow lines denote true and false matches, respectively.

conditions in real-world tasks from the perspective of machine learning. Some works have been proposed to learn the affinity function in recent years. In [22], an unsupervised method is proposed to perform parameter learning without the assistance of correspondences between graphs. By making use of the ground truth correspondence information, a supervised graph matching learning algorithm is presented in [23].

With the popularity of deep learning in various graph related domains [24], some researchers also explored the application of deep learning in graph matching learning. A seminal work is presented in [25], which learns the affinities for graph matching with deep neural networks. Specifically, an end-to-end model is presented and represented as deep feature extraction hierarchies, and this model is used to learn all the parameters in the graph matching process. Noticing the deficiencies of the work in [25], the authors of [26] proposed a supervised permutation loss to capture the combinatorial nature of graph matching, and adopted deep graph embedding models to learn intra-graph and cross-graph affinity functions. While this embedding is designed for graph matching, it is also able to deal with higher-order structures in hypergraph matching. This algorithm is shown to outperform state-of-the-art graph matching learning methods in experiments. Graph neural network [27] has become an important approach in

graph matching recently. It is shown in [28] that graph neural networks can be trained to embed graphs into vector spaces for efficient similarity measuring. Then a graph neural network model is proposed to compute the similarity between graphs by fusing information across graphs through an attention-based matching mechanism. Graph neural network embedding is used in [29] to generate initial soft correspondences between nodes of two graphs, which are then refined iteratively to achieve local consensus between graphs with synchronous message passing networks. This algorithm is able to recover global correspondences while scaling to large datasets.

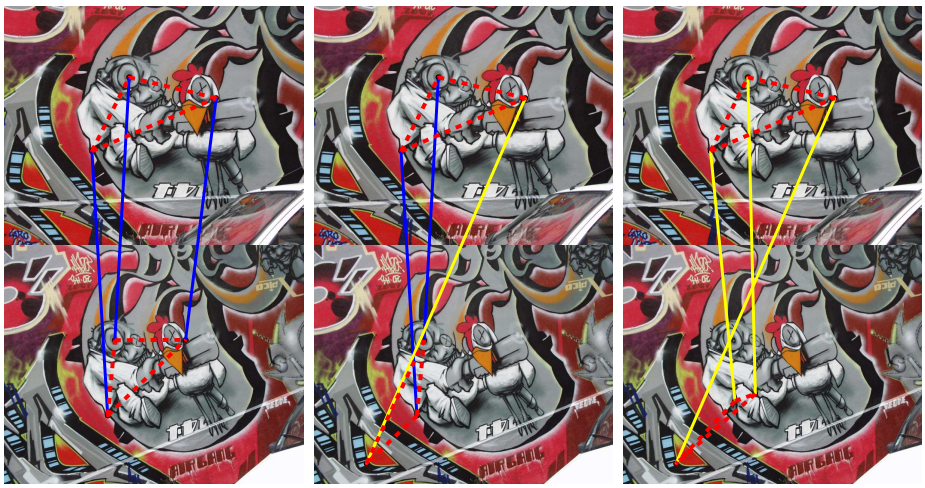
## 3. Density based Enhancement

Different from previous (hyper)graph matching algorithms, in this paper we cast cast hypergraph matching of features as a non-cooperative multi-player game and obtain a consistent group of matches as the final matches. In this part we introduce the game-theoretic algorithm and discuss its problem and the reason. Then we present a density enhancement method to solve the problem.

### *3.1. Game-theoretic approach*

In order to obtain a high matching accuracy, we investigate the different behaviors of true and false matches. Firstly, we denote the hypergraph of model features by $H_1 = (V_1, E_1)$, where $E = \{e_{1i}\}$ stands for the set of model hyperedges. Similarly, $H_2 = (V_2, E_2)$ with $E_2 = \{e_{2j}\}$ is used to represent the hypergraph of test features. Following previous works, in this paper we use the third-order hypergraph to accommodate both computation efficiency and representation ability. In the third-order hypergraph, one hyperedge is composed of three vertices. Therefore two hyperedges $e_{1i} \in E_1$ and $e_{2j} \in E_2$ are related by three matches, as illustrated in Figure 2, where red dashed triangles stand for hyperedges.

In Figure 2(a) the two hyperedges are correspondences, and they are related by three true matches. Evidently, the two hyperedges are similar in geometric



(a) True matches (b) True and false matches (c) False matches

Figure 2: Demonstration of hyperedges and associated matches. Hyperedges are denoted by red dashed triangles. True and false matches are denoted by blue and yellow lines, respectively. In this paper, the similarity between two hyperedges is treated as the higher-order similarity of the three associated matches.

structure and also in appearance. By treating the similarity between hyperedges as the higher-order similarity among three matches, we observe that three true matches have a large higher-order similarity. In contrast, Figure 2(b) shows that two true matches and one false match are associated with a small higher-order similarity, and Figure 2(c) indicates that three false matches are also associated with a small higher-order similarity. Based on these observations, we arrive at the conclusion that true matches are similar and they constitute a consistent group. Meanwhile, false matches are not consistent with true matches, and false matches are not consistent in themselves. To sum up, true matches form a consistent group in all the candidate matches. The hypergraph matching problem can then be translated to finding out the consistent group in the hypergraph of candidate matches.

To obtain the consistent group of candidate matches, we consider a non-cooperative three-player game played in the evolutionary scenario [30]. Three players are randomly drawn from a large population to play the game, where

each player in the population selects a pre-defined strategy. This game is repeated in the evolution, and players with small payoffs are driven to extinct gradually under the evolutionary pressure. Finally, the survived players form a group where all the inside players have large payoffs.

We then cast the hypergraph matching of features as such a game, and the candidate matches are the strategies selected by players. When three players are selected to play the game, they receive a payoff which is measured by the higher-order similarity among their selected strategies (candidate matches), namely the similarity between two hyperedges (Figure 2). In this formulation, the survived players form a large-payoff group, and their selected strategies (candidate matches) have large similarities. In other words, the candidate matches selected by survived players form a consistent group with large internal similarities. In evolutionary game theory, this consistent group of candidate matches can be obtained by extracting the evolutionary stable strategies (ESS's) [30]. Therefore the problem is transformed to extracting the ESS group.

Assume that there are $n$ strategies, which correspond to the $n$ candidate matches. With $s_i$, $i = 1, 2, 3$, denoting the strategy selected by the $i$-th player in each game, the payoff $\pi(s_1, s_2, s_3)$ of three players is equal to the higher-order similarity of the three associated candidate matches. The ESS $x = (x_1, x_2, \cdots, x_n)$ denotes the weights of the $n$ strategies, and can be obtained with the Baum-Eagon inequality [31] as

$$x_j(t+1) = x_j(t)\frac{u(e^j, x(t), x(t))}{u(x(t), x(t), x(t))}, \quad j = 1, \cdots, n, \tag{2}$$

where $e^j$ is a $n$-dimensional vector $z$ with $z_j = 1$ and $z_l = 0$ for $l \neq j$, and

$$u(y^{(1)}, y^{(2)}, y^{(3)}) = \sum_{(s_1,s_2,s_3)\in S^3} \pi(s_1, s_2, s_3) \prod_{i=1}^{3} y_{s_i}^{(i)}. \tag{3}$$

In Eq. (3), $\pi(s_1, s_2, s_3)$ is the payoff of three players when they select strategies $s_1$, $s_2$ and $s_3$, $y_{s_i}^{(i)}$ is the probability of the $i$-th player selecting $s_i$, and $S^3$ denotes the set of all combinations of strategies selected by three players. Therefore in Eq. (2) the denominator $u(x(t), x(t), x(t))$ denotes the expected payoff of the

entire population, and the numerator $u(e^j, x(t), x(t))$ represents the expected payoff of players selecting the $j$-th strategy. As the higher-order similarity is supersymmetric, $u(e^j, x(t), x(t))$ can be used without loss of generality. Due to limited space, we refer the reader to [31] for more details on the derivation of Eq. (2) and Eq. (3).

In order to obtain the ESS $x$, we need to calculate the payoff $\pi(s_1, s_2, s_3)$, which corresponds to the higher-order similarity of three candidate matches. As illustrated in Figure 2, the higher-order similarity of three matches is equal to the similarity between the corresponding model and test hyperedges. Denoting the two hyperedges (triangles) by $e_{1i}$ and $e_{2j}$, and the three angles by $\alpha_{1k}$ and $\alpha_{2k}$, $k = 1, 2, 3$, the similarity of two hyperedges can be calculated as

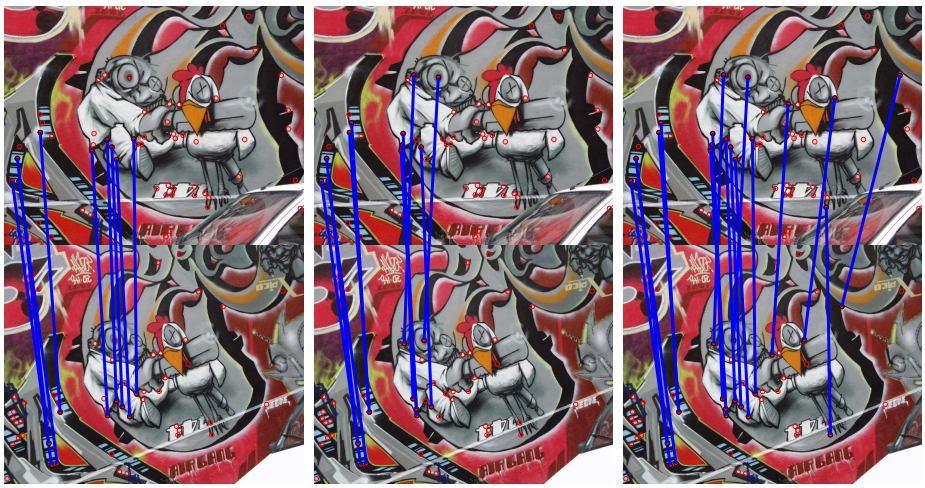
$$\chi(e_{1i}, e_{2j}) = exp\left(-\frac{1}{\sigma}\sum_{k=1}^{3}|(sin(\alpha_{1k}) - sin(\alpha_{2k}))|\right), \tag{4}$$

where $\sigma$ is a scaling parameter. This similarity can then be used as the payoff $\pi(s_1, s_2, s_3)$ in Eq. (3).

After we obtain the weights of strategies with Eq. (2), the strategies whose weights are greater than a threshold form an ESS group. As one strategy corresponds to one candidate match in our algorithm, the matches whose weights are greater than a threshold form an ESS group. Accordingly, $x_j$ is the weight of the $j-th$ match, which reflects the consistency of $j-th$ match with other matches. To some extent, an ESS group can be regarded as the counterpart in a hypergraph of a dominant set [32] in a graph. The dominant set concept has some interesting properties, which have been used in various applications [33, 34] and will be used in our approach.

### *3.2. Problem*

With the game-theoretic hypergraph matching algorithm presented above, we show the matching result on the graf image pair in Figure 3(a). It can be observed that all the 9 matches are true matches, indicating that our algorithm is effective in removing false matches. However, we also observe that there are only 9 matches, much smaller than the 25 true matches of the BCA algorithm in



(a) 0.5-500, 9/9, 5.50s (b) 0.5-1000, 11/11, 11.30s (c) 1-1000, 18/18, 29.65s

Figure 3: Matching results with different parameters. Blue lines denote true matches. In the captions, the first items denote "$n_t$-$n_n$", the second ones denote "number of true matches/number of all matches", and third ones denote running time. The parameters $n_t$ and $n_n$ are explained in the following text.

Figure 1. This means that some true matches are misclassified as false matches and rejected, thereby reducing the number of matches. In many applications, it may be necessary to obtain more matches.

In order to find out why the number of matches from this algorithm is small, we investigate the whole algorithm in depth. In our opinion, the reasons behind this problem are as follows. First, the consistency among candidate matches in the ESS group is very high. As aforementioned, an ESS group can be regarded as the extension of a dominant set to the hypergraph. In a dominant set, each pair of data items have a large similarity. In other words, one data is similar to not only its nearest neighbors, but also all the members in the same dominant set. Equivalently, in an ESS group each triplet of candidate matches have a large consistency (similarity). This extremely strict constraint on internal consistency limits the number of matches in the ESS group inevitably.

The second reason relates to the parameters in the matching process. The transition from graphs to hypergraphs brings a significant increase in the amount

of (hyper)edges. In order to reduce the computation load, a common practice is to sample the test hyperedges in the matching process. Specifically, each hyperedge in $E_1$ and its $n_n$ nearest neighbors in $E_2$ are used in computation with Eq. (2). In addition, we also sample the model hyperedges to save computation further. For each model feature $\zeta_i$, we find its $n_t(n_1-1)$ nearest neighbors, with $n_t$ denoting the percentage of nearest neighbors used to build model hyperedges. Then each pair of these nearest neighbors are used to form a hyperedge with $\zeta_i$. By introducing the parameters $n_t$ and $n_n$, we reduce the computation load of the matching process. The cost is that some model hyperedges have no chance to be matched to their correspondences, and therefore their associated true matches are discarded as false matches. This also explains why some true matches are removed by our algorithm.

Based on the second reason, one straightforward approach to obtain more matches is to use larger $n_t$ and $n_n$. In fact, Figure 3(b) and Figure 3(c) show that using larger $n_t$ and $n_n$ does increase the number of matches evidently. However, we also observe that this increase is obtained at the cost of a significant increase in running time. More generally, we illustrate this problem with three datasets, namely VGG, Middlebury and KITTI datasets, which will be introduced in Section 4. The matching results and running time on these datasets are reported in Figure 4. It can observed from Figure 4 that while the increase in $n_t$ and $n_n$ does increase the number of matches, it also results in a large increase in the running time.

### *3.3. Enhancement*

Starting from the high internal consistency in the ESS group, we present a density based enhancement to increase the number of matches while avoiding the significant increase of computation load. Our approach is motivated by the density based clustering algorithm DBSCAN [35], which requires a relatively low consistency in the cluster. As the DBSCAN algorithm is based on the pairwise similarity/distance, we firstly define density-related concepts in the third-order hypergraph. Given $\mathbb{H} = (\mathbb{V}, \mathbb{E})$ denoting a third-order hypergraph

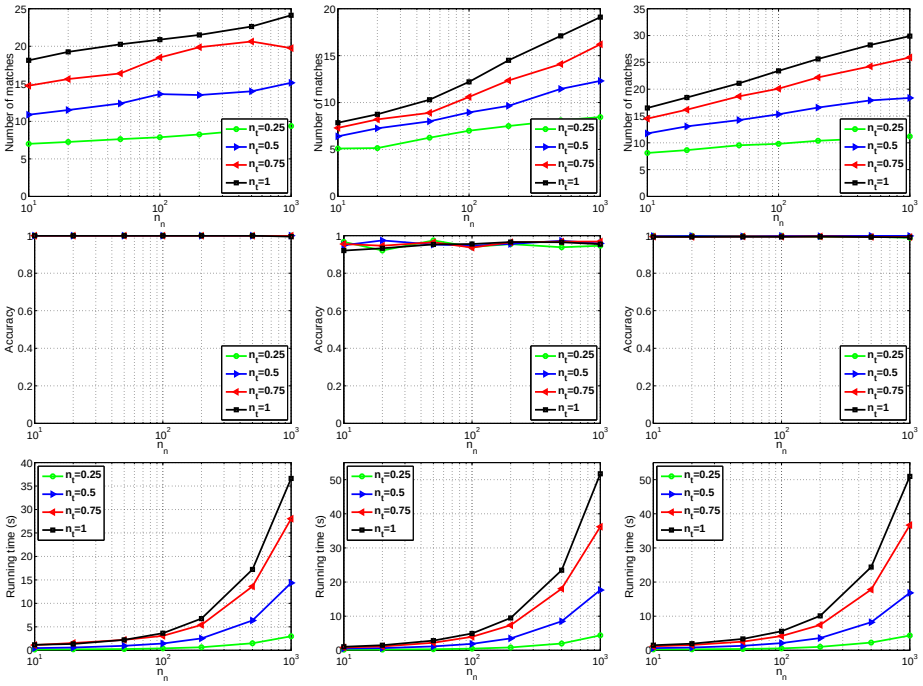



Figure 4: Matching results with different parameters on three datasets. From left to right, the three columns belong to VGG, Middlebury and KITTI datasets, respectively.

with $v_i, v_j, v_k \in \mathbb{V}$ and $\pi(v_i, v_j, v_k)$ representing the third-order similarity of of $v_i$, $v_j$ and $v_k$, the similarity between a vertex $v_i$ and a pair of vertices $(v_j, v_k)$ is defined as $\pi(v_i, v_j, v_k)$. With this definition of pair-vertex similarity, the nearest neighbors of a vertex pair $(v_i, v_j)$ are vertices, but not other vertex pairs. As aforementioned, each triplet of matches in an ESS group have a large similarity. Based on the definition of vertex-pair similarity, we see that in an ESS group, one vertex is similar to not only its nearest neighbors (pairs), but also all the other pairs in the group. As this constraint is too strict and reduces the number of matches, we relax this constraint and require one vertex to have large similarities with only the nearest pairs. Recall that in DBSCAN-like density based clustering [35], one data is similar to only the nearest neighbors. We are motivated by this observation to relax the constraint in a DBSCAN-like manner. For this purpose, we define the following concepts further.

**Definition 1.** *$\varepsilon$-neighborhood. Given $\mathbb{H} = (\mathbb{V}, \mathbb{E})$ denoting a third-order hypergraph with $v_i, v_j, v_k \in \mathbb{V}$ and $\pi(v_i, v_j, v_k)$ representing the third-order similarity of of $v_i$, $v_j$ and $v_k$, the $\varepsilon$-neighborhood of a vertex pair $(v_i, v_j)$ is a neighborhood $\mathbb{N}$ such that $\pi(v_i, v_j, v_k) \leq \varepsilon$ holds for all $v_k \in \mathbb{N}$, with $v_i, v_j, v_k \in \mathbb{V}$.*

**Definition 2.** *Core pair. Given $\mathbb{H} = (\mathbb{V}, \mathbb{E})$ denoting a third-order hypergraph with $v_i, v_j, v_k \in \mathbb{V}$, a vertex pair $(v_i, v_j)$ is called a core pair, if a minimum of $MinPts$ vertices are in its $\varepsilon$-neighborhood.*

The concept of *core pair* in a third-order hypergraph is equivalent to the *core point* in the DBSCAN algorithm, and the parameters $\varepsilon$ and $MinPts$ correspond to $Eps$ and $MinPts$ in the DBSCAN algorithm, respectively. Intuitively, core pairs have large similarities with neighboring vertices and correspond to matches with large confidences to be true matches. As mentioned above, the consistency among matches in an ESS group $D$ is very high and some true matches are also rejected by this strict constraint. This means that the included matches have a large confidence to be true matches. Therefore we have reason to treat each vertex pair in an ESS group as a core pair. In other words, each vertex pair $(v_i, v_j)$ with $v_i, v_j, \in D$ has a minimum of $MinPts$ vertices in the $\varepsilon$-neighborhood. This enables us to calculate $\varepsilon$ as the minimum similarity of a core pair to its $MinPts$-th nearest neighbor in the ESS group, i.e.,

$$\varepsilon = \min_{v_i, v_j \in D} \pi(v_i, v_j, v_{ij}^*), \tag{5}$$

with $v_{ij}^*$ representing the $MinPts$-th nearest vertex of $(v_i, v_j)$ in the ESS group. Then we can add new matches into the ESS group in an DBSCAN-like manner. Specifically, for each pair of vertices in the ESS group, we retrieve the neighboring vertices in its $\varepsilon$-neighborhood. If one of these neighboring vertices is outside the ESS group, it is included. In this way, the ESS group is expanded and the number of matches (vertices) is increased.

### *3.4. Enforcing one-to-one constraint*

The original ESS group exerts a strict constraint on the high similarity (consistency) of matches. While this strict constraint leads to a small number

of matches, the large internal consistency also means a large confidence of the inside matches being true matches. In other words, the outside matches have a smaller confidence to be true matches than the inside ones. By relaxing the constraint, we include some outside matches into the ESS group and obtain more matches. In this process, the matching accuracy may be degraded as outside matches are more likely to be false matches.

In order to obtain more matches without degrading the matching accuracy, we propose two methods to enforce the one-to-one constraint. Both methods are based on the special properties of our game-theoretic approach. The first method is called the internal method, which enforces the constraint between the matches in the original ESS group. As aforementioned, each candidate match in the original ESS group is assigned a weight, which reflects the consistency of the candidate match with others. Therefore a large weight means a large confidence of the candidate match being a true match. Based on this observation, if two or more matches in the ESS group are associated with the same feature, only the one with the largest weight is adopted, and all the others are discarded.

The second method is the external one, which is applied to the matches outside the ESS group. As we discuss above, compared with the matches outside the ESS group, the inside matches are more likely to be true matches. Therefore if a match outside the ESS group is associated with the same feature as one match in the ESS group, it will not be included into the ESS group. In this way, we reduce the probability of false matches being included into the ESS group.

### *3.5. Algorithm*

The whole clustering procedure of our hypergraph matching algorithm is described as follows. Given the set $F_1$ of model features and $F_2$ of test features, we build the set $E_1 = \{e_{1i}\}$ of model hyperedges and $E_2 = \{e_{2j}\}$ of test features. For each $e_{1i} \in E_1$, we find its $n_n$ most similar test hyperedges, obtaining the set $P = \{(e_{1i}, e_{2k})\}$ of all hyperedge pairs. From each hyperedge pair $(e_{1i}, e_{2k})$ we obtain three associated candidate matches and the corresponding higher-order

similarity, based on which the hypergraph of candidate matches $\mathbb{H} = (\mathbb{V}, \mathbb{E})$ is built. After obtaining the ESS group $D$, we enforce the one-to-one constraint with the internal method and calculate $\varepsilon$ with Eq. (5). Then for each pair of vertices in the ESS group, we find the vertices in the $\varepsilon$-neighborhood and outside the ESS group and include them into the ESS group. Then the one-to-one constraint is enforced between the original vertices and new members in the ESS group with the external method. Following the MC-HDSet algorithm, the ESS group in our algorithm can be regarded as a HDSet (Hypergraph Dominant Set). As the ESS group is expanded based on DBSCAN, we name our algorithm as HDSet-DBSCAN. The whole process of our HDSet-DBSCAN algorithm is described in Algorithm 1.

Based on the computation steps in Algorithm 1, we discuss the computation complexity in the following. The building of $E_1$ and $E_2$ in Line 1 results in $n_1 \frac{(n_t(n_1-1))(n_t(n_1-1)-1)}{2}$ model hyperedges and $n_2(n_2-1)(n_2-2)$ test hyperedges, indicating a complexity $O(n_t^2 n_1^3 + n_2^3)$. In Line 2, we find the $n_n$ most similar test hyperedges for each each model hyperedge, and the computation complexity is $O(n_t^2 n_n n_1^3 n_2^3)$. The hypergraph $\mathbb{H}$ includes at most $n_1 n_2$ vertices, and the computation complexity of Line 3 is therefore $O(n_1 n_2 n_i)$, with $n_i$ denoting the number of interations in generating $D$. In our experiments, $n_i$ is limited to no more than 500. The calculation of $\varepsilon$ in Line 4 involves the traverse of vertex pairs in $D$ and the sorting of similarities between a pair and neighboring vertices in $D$, resulting in a computation complexity $O(|D|^4)$. Here $|D|$ denotes the number of matches in $D$ and it is therefore close to $\min(n_1, n_2)$. In Line 5, the enforcing of one-to-one constraint needs to compare each match in $D$ with all the other matches in $D$, indicating a computation complexity $O(|D|^2)$. From Line 6 to Line 15, three consecutive operations include traverse of vertex pairs in $D$ (Line 6), traverse of vertices in $\mathbb{V}$ (Line 8), and traverse of vertices in $D$ (Line 12). These three operations together correspond to a computation complexity $O(|D|^3 n_1 n_2)$. In summary, the computation complexity of the whole algorithm is $O(n_t^2 n_n n_1^3 n_2^3)$.

**Algorithm 1** The HDSet-DBSCAN algorithm.

**Input:**

model features $F_1 = \{\zeta_i\}$, $i = 1, \cdots, n_1$

test features $F_2 = \{\xi_i\}$, $i = 1, \cdots, n_2$

$n_t$, $n_n$, $MinPts$, $\sigma$ //find the setting of parameters in Section 4.3

**Output:**

a set of matches $D = \{(\zeta_k, \xi_{k'})\}$, $k = 1, \cdots, n_m$

Build the set $E_1$ of model hyperedges and $E_2$ of test hyperedges.
Build the hypergraph $\mathbb{H} = (\mathbb{V}, \mathbb{E})$.
Obtain the original ESS group $D$ by Eq. (2).
Calculate $\varepsilon$ with Eq. (5).
Enforce one-to-one constraint in $D$ with the internal method.
Build a set $\Gamma = \{(v_i, v_j)\}$, $v_i, v_j \in D$, of vertex pairs.
**for** $(v_i, v_j) \in \Gamma$ **do**
    Find $\Upsilon = \{v_k\}$ in the $\varepsilon$-neighborhood of $(v_i, v_j)$.
    **for** $v_k \in \Upsilon$ **do**
        **if** $v_k \notin D$ **then**
            Include $v_k$ into $D$.
            Enforce one-to-one constraint with the external method.
        **end if**
    **end for**
**end for**

## 4. Experiments

Our algorithm casts hypergraph matching as a non-cooperative multi-player game, and uses a density-based enhancement to obtain more matches while maintaining a high matching accuracy. In the experiments, we firstly validate the effectiveness of the two major components, i.e., density-based enhancement and enforcement of the one-to-one constraint, separately. Then experiments on some real image datasets are conducted and comparisons are made with

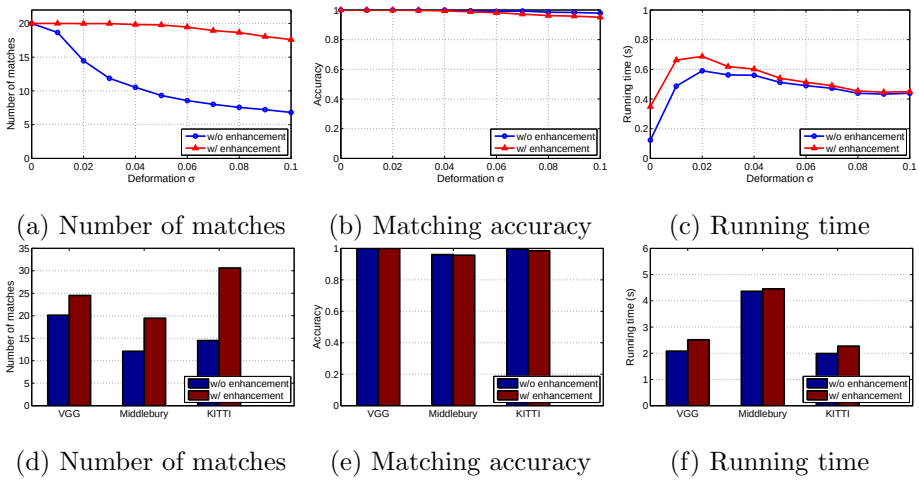
some other hypergraph and graph matching algorithms. The experiments are conducted with Matlab on a computer with Intel Core i7-10510U processor (1.8GHz) and 16 GB RAM.

### *4.1. Effect of enhancement*

The first part of experiments are conducted on synthetic datasets. We generate a set $P$ of 20 random 2D points following the normal distribution $N(0, 1)$, and then transform $P$ to obtain the other set $Q$. The transformation from $P$ to $Q$ includes a rotation of $\pi/3$, a scale change of factor 2, and Gaussian deformation $N(0, \sigma^2)$. As the deformation is likely to result in false matches, we set $\sigma$ as 0 to 0.1 with the step of 0.01 and show the matching results between $P$ and $Q$ in the top row of Figure 5.

Figure 5(a) indicates that with the increase of deformation $\sigma$, our algorithm without the enhancement generates less and less matches. This is not surprising, as the deformation reduces the consistency among true matches and results in small ESS groups. With the help of the enhancement, our algorithm is able to accommodate the relatively small consistency and generates more matches outside the original ESS group. This effect is especially evident with large deformation $\sigma$, where the enhancement can double the number of matches. Meanwhile, we observe from Figure 5(b) that the deformation has a small influence on the matching accuracy, and the enhancement only decreases the matching accuracy slightly. In other words, most of the matches from the enhancement are true matches, indicating the effectiveness of the enhancement. Finally, Figure 5(c) shows that the enhancement leads to a small increase in the computation load. It is interesting to notice that the increase of computation load is very small with large deformation $\sigma$. This observation implies that our algorithm is promising in dealing with real image datasets with large deformation.

We then conduct experiments on three real datasets to demonstrate the effect of the enhancement. The VGG dataset is from the University of Oxford, consisting of 8 image pairs, namely graf, boat, wall, bark, bikes, trees, leuven and ubc. The Middlebury dataset includes 20 image pairs of indoor scenes selected




(a) Number of matches (b) Matching accuracy (c) Running time

(d) Number of matches (e) Matching accuracy (f) Running time

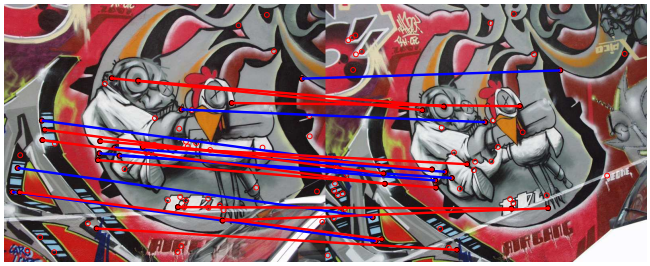
Figure 5: Comparison of the matching results before and after enhancement. The top row belongs to the synthetic point sets, and the bottom row belongs to real image datasets.

from the 2014 Middlebury stereo dataset [36]. The KITTI dataset is composed of 65 representative image pairs of outdoor scenes selected from the 200 image pairs of the KITTI stereo 2015 dataset [37]. These three real datasets include various images of indoor and outdoor scenes, making the matching results a convincing measure of the matching performance.

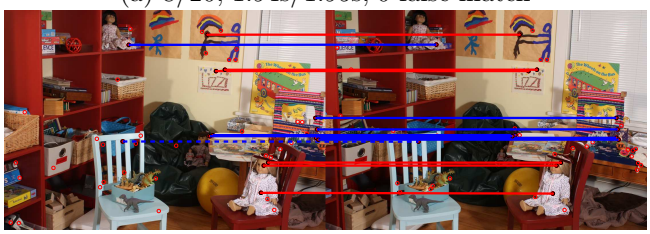
Following the experiments with synthetic datasets, we report the average matching results and running time with and without the enhancement in the bottom row of Figure 5. It can be observed that the enhancement generates evidently more matches on all the three datasets. Meanwhile, the matching accuracies only decrease slightly, and the increases in running time are small. In other words, the evident increase in the number of matches is obtained at a small cost in matching accuracy and computation load, validating the effectiveness of the enhancement. Some examples of the matching results on the three real datasets are shown in Figure 6.

### *4.2. Effect of one-to-one constraint*

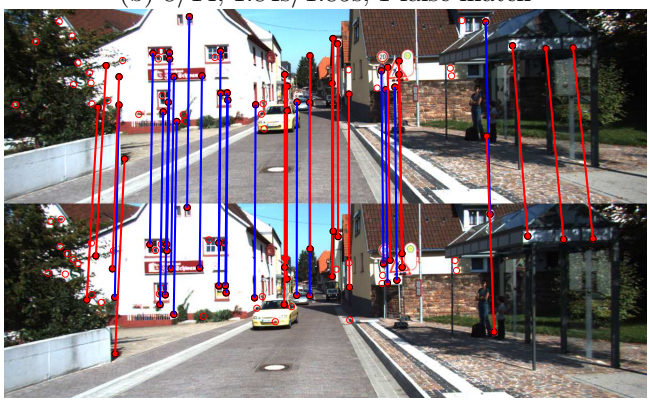
In this paper we propose a density enhancement to obtain more matches. In order not to degrade the matching accuracy with the increase of matches,



(a) 8/20, 1.94s/1.95s, 0 false match



(b) 8/14, 1.84s/1.85s, 1 false match



(c) 21/40, 2.73s/3.07s, 0 false match

Figure 6: Some matching results of our algorithm. Blue and red lines denote the matches in the ESS group and from the enhancement, respectively. Dashed lines denote false matches. The first and second items denote the number of matches and running time, respectively. The results in front of "/" are obtained without the enhancement, and those behind "/" are obtained with the enhancement.

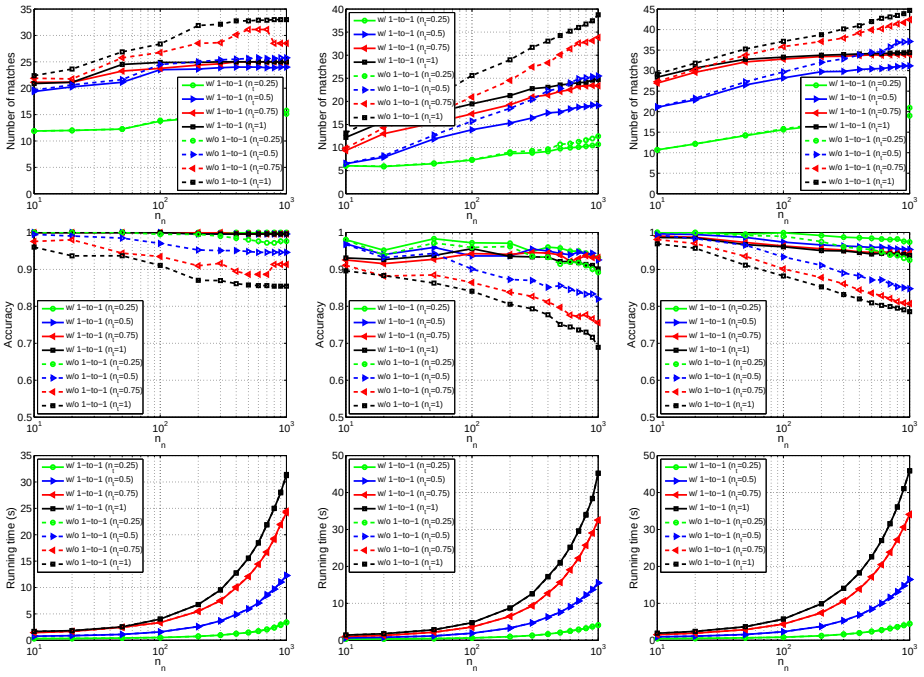



Figure 7: Matching results of our algorithm with and without enforcing the one-to-one constraint. From left to right, the three columns belong to VGG, Middlebury and KITTI datasets, respectively.

we propose two methods to enforce the one-to-one constraint. Here we test the effect of the two methods in obtaining a high matching accuracy. Specifically, we compare the matching results with and without enforcing the one-to-one constraint, and report the comparisons in Figure 7.

In Figure 7, the top row shows that with all the three datasets, enforcing the one-to-one constraint reduces the number of matches evidently. Meanwhile, the middle row indicates that by enforcing the one-to-one constraint, the matching accuracy is improved evidently. In other words, by enforcing the one-to-one constraint, we remove many false matches, thereby obtaining a high matching accuracy. In addition, we observe from the bottom row that the increase of running time from enforcing the one-to-one constraint is negligible with all the three datasets. These comparison results together validate the effectiveness of

our methods to enforce the one-to-one constraint.

### *4.3. Parameters*

Figure 7 shows that both increasing $n_t$ and increasing $n_n$ are able to increase the number of matches. This means that a certain number of matches can be obtained with the combination of a large $n_t$ and a small $n_n$, or the combination of a small $n_t$ and a large $n_n$. We find that it is more efficient to increase the number of matches by increasing $n_t$. Based on Figure 7, we show the relationship between $n_t$ and the corresponding number of matches and running time in Figure 8. It can be observed that with the same running time, a large $n_t$ is able to generate more matches than a small one, and a large $n_t$ consumes less running time in generating the same number of matches than a small one. In order to explain this observation, we have a look at the different influences of $n_t$ and $n_n$ on the matching results. Hypergraph matching matches one model hyperedge to the corresponding test hyperedge, and then obtains the matches of associated features. With the parameter $n_t$, one model feature and each pair in its $n_t(n_1 - 1)$ nearest neighbors form a model hyperedge (triangle). A small $n_t$ corresponds to small triangles, meaning that the matching process lays importance on the local consistency between the model and test images. With a large $n_t$, we have both small and large triangles (hyperedges), accommodating both local and global consistency between two images. Therefore increasing $n_t$ helps increase the number of matches effectively. With the parameter $n_n$, one model hyperedge selects its $n_n$ most similar test hyperedges as candidate correspondences, which are used in further processing. Generally, one model hyperedge and the corresponding test hyperedge are similar, and therefore a small $n_n$ is usually enough to include the corresponding test hyperedge into the candidates. In this case, while increasing $n_n$ results in an increase of the computation load, it is unlikely to increase the number of matches evidently. In other words, it does not pay off to increase the number of matches by increasing $n_n$. For this reason, we choose to fix $n_t$ at the maximum, i.e., $n_t = 1$, and actually remove this parameter in this paper. This enables us to use a small

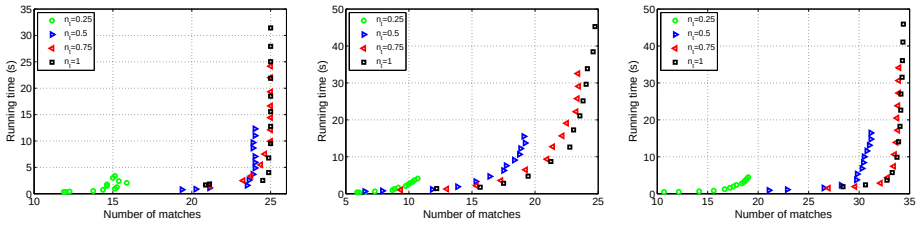



Figure 8: The relationship between $n_t$ and the corresponding number of matches and running time.

$n_n$ and therefore a small number of test hyperedges in computation, which further means a small computation load. Figure 4 shows that from $n_n = 100$ to $n_n = 1000$, the running time rises drastically, whereas the increase in the number of matches is small. Therefore it seems appropriate to set $n_n \leq 100$ as a compromise between computation load and number of matches.

In addition to $n_n$, our algorithm has two other parameters. One parameter is the DBSCAN parameter $MinPts$ introduced in Section 3.3, and the other is the scaling parameter $\sigma$ in Eq. (4). With $n_n = 100$, $MinPts = 2, 3, \cdots, 10$ and $\sigma = 1, 2, 5, 10, 15, 20$, we report the clustering results in Figure 9.

Figure 9 shows that with all the three datasets, the parameter $MinPts$ has a small influence on the number of matches and matching accuracy. Based on the running time, small $MinPts$, e.g., $MinPts \leq 4$ is preferred. Meanwhile, the parameter $\sigma$ has little influence on the number of matches and matching accuracy. Based also on the running time, we prefer to select the parameter $\sigma$ in the range [2,5].

### *4.4. Comparison*

As our algorithm is not learning-based, we make a comparison with some other learning-free matching algorithms, including second-order algorithms MPM (Max Pooling Matching) [5], RRWM (Reweighted Random Walks for Graph Matching) [38] and SM (spectral matching) [7], and higher-order ones HGM, TM, RRWHM, BCA, GTHM and MC-HDSet. The parameters of these algorithms have been selected to generate the best average matching accuracy on

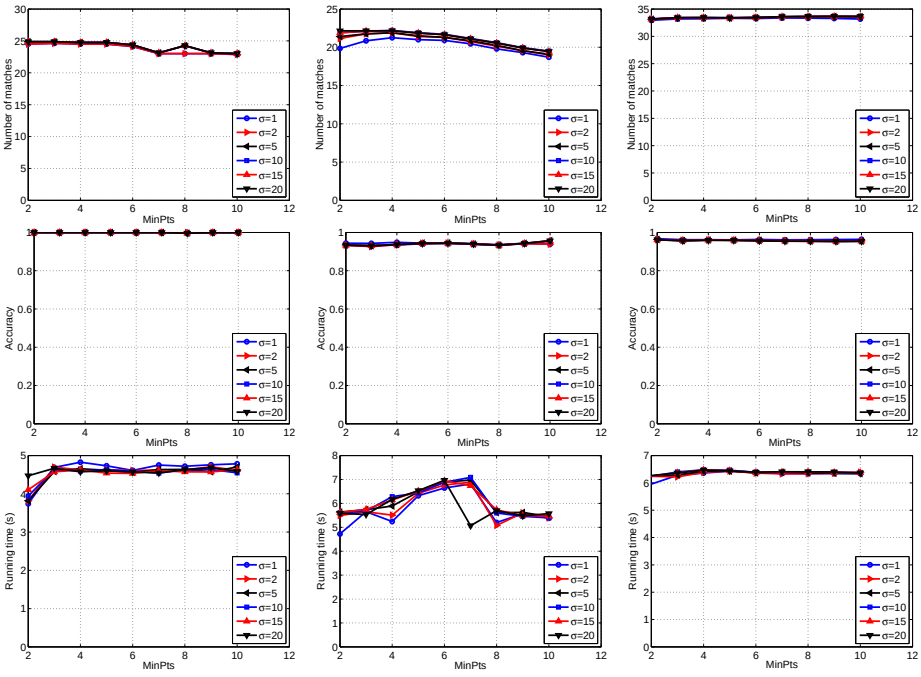



Figure 9: Demonstration of the influence of two parameters $\sigma$ and $MinPts$ on the clustering results of our algorithm. The left, middle and right columns belong to the VGG, Middlebury and KITTI datasets, respectively.

each dataset. In our algorithm, the parameters are selected to generate comparable number of matches as MC-HDSet, in order to compare their computation efficiency. The detailed comparison is reported in Table 1 to Table 3, where "# model", "# test", "# true" denote number of model features, number of test features and maximum number of true matches, whereas "# matches obtained" and "# true obtained" denote number of matches and number of true matches obtained by the algorithms, respectively.

We arrive at the following observations from Table 1 to Table 3. First, except for GTHM, MC-HDSet and our algorithm, all the other second-order and higher-order matching algorithms generate low matching accuracies. The reason is that these algorithms are assignment based, and the outliers in model features are also matched to test features. In fact, the outliers contribute to the

Table 1: Average matching results on the VGG dataset.

| | # model | # test | # true | # matches obtained | accuracy | # true obtained | runtime (s) |
|---|---|---|---|---|---|---|---|
| MPM | 54.63 | 55.00 | 29.00 | 54.63 | 0.33 | 17.63 | 4.51 |
| RRWM | | | | 54.63 | 0.32 | 17.25 | 1.03 |
| SM | | | | 54.63 | 0.24 | 12.38 | 0.69 |
| HGM | | | | 54.63 | 0.37 | 19.38 | 0.81 |
| TM | | | | 54.63 | 0.37 | 19.13 | 0.81 |
| RRWHM | | | | 54.63 | 0.51 | 27.25 | 1.07 |
| BCA | | | | 54.63 | 0.50 | 26.50 | 1.33 |
| GTHM | | | | 19.75 | 1.00 | 19.75 | 0.49 |
| MC-HDSet | | | | 23.38 | 1.00 | 23.38 | 6.98 |
| Ours | | | | 24.50 | 1.00 | 24.50 | 2.41 |

majority of false matches in these algorithms. Comparatively, the RRWHM and BCA algorithms perform better than the other graph and hypergraph matching algorithms in matching accuracy. Second, while the GTHM, MC-HDSet and our algorithm generate similar and high matching accuracies, their performance differences lie in the number of matches and running time. Compared with GTHM, our algorithm generates evidently more matches, from 21.64% of KITTI dataset to 53.15% of Middlebury dataset. This means that our algorithm increases the number of matches without degrading the matching accuracy. This is attributed to the density based enhancement and the enforcing of the one-to-one constraint. Compared with MC-HDSet, our algorithm generates similar number of matches and reduces the running time significantly. The reason lies in that in our algorithm $n_n$ can be set to be smaller than 100, which is much smaller than $n_n = 500$ in MC-HDSet. The small $n_n$ reduces the search space and therefore reduces the computation load. Third, our algorithm involves a larger computation load than other hypergraph matching algorithms except for MC-HDSet. On one hand, the extraction of the ESS group is a little time consuming. On the other hand, in order to obtain more matches, our algorithm involves a large number of candidate hyperedges, thereby increasing the com-

Table 2: Average matching results on the Middlebury dataset.

| | # model | # test | # true | # matches obtained | accuracy | # true obtained | runtime (s) |
|---|---|---|---|---|---|---|---|
| MPM | 58.45 | 54.90 | 29.40 | 58.45 | 0.37 | 20.50 | 6.68 |
| RRWM | | | | 58.45 | 0.35 | 19.55 | 1.29 |
| SM | | | | 58.45 | 0.30 | 16.15 | 0.90 |
| HGM | | | | 58.45 | 0.33 | 18.45 | 1.07 |
| TM | | | | 58.45 | 0.32 | 17.80 | 1.05 |
| RRWHM | | | | 58.45 | 0.41 | 23.60 | 1.51 |
| BCA | | | | 58.45 | 0.42 | 23.95 | 1.76 |
| GTHM | | | | 12.70 | 0.96 | 12.45 | 0.43 |
| MC-HDSet | | | | 21.45 | 0.97 | 20.85 | 8.71 |
| Ours | | | | 19.45 | 0.96 | 18.55 | 4.43 |

putation load. This is one major problem of our algorithm and will be studied in more depth in future works.

In real-world feature matching tasks, the outliers in model and test features are usually inevitable and may have a large portion in all the features. If not dealt with appropriately, these outliers may result in a large amount of false matches and a low matching accuracy. The matching score based hypergraph matching algorithms solve an assignment problem by maximizing the matching scores between model and test features. While our experiments show that these algorithms are able to find out most of the true matches, they cannot identify the false matches caused by outliers. As a result, their matching accuracies are typically rather low. Compared with these matching score based algorithms, the game-theoretic algorithm extracts a group of consistent matches, thereby removing the false matches caused by outliers. However, this algorithm has a somewhat too high requirement on the consistency of matches, and some true matches are also removed. Consequently, the number of matches of this algorithm is quite small. The MC-HDSet algorithm extends the game-theoretic algorithm by extracting multiple groups of consistent matches to increase the number of matches, at the cost of a large computation load. In this paper,

Table 3: Average matching results on the KITTI dataset.

| | # model | # test | # true | # matches obtained | accuracy | # true obtained | runtime (s) |
|---|---|---|---|---|---|---|---|
| MPM | 60.35 | 61.00 | 37.78 | 60.35 | 0.45 | 27.02 | 5.53 |
| RRWM | | | | 60.35 | 0.43 | 25.46 | 1.40 |
| SM | | | | 60.35 | 0.33 | 19.83 | 1.01 |
| HGM | | | | 60.35 | 0.57 | 34.42 | 1.13 |
| TM | | | | 60.35 | 0.58 | 34.38 | 1.12 |
| RRWHM | | | | 60.35 | 0.60 | 35.97 | 1.42 |
| BCA | | | | 60.35 | 0.60 | 36.03 | 1.81 |
| GTHM | | | | 25.18 | 0.99 | 25.05 | 1.00 |
| MC-HDSet | | | | 31.02 | 0.99 | 30.92 | 8.61 |
| Ours | | | | 30.63 | 0.99 | 30.20 | 2.26 |

we propose to expand the group of consistent matches to obtain more matches and enforce the one-to-one constraint to improve the matching accuracy. By investigating the influence of parameters on matching results, we find out the appropriate parameters. As a result, our algorithm obtains comparable number of matches and matching accuracy to MC-HDSet, whereas the computation load is reduced significantly in comparison with the latter. In this sense, we believe our algorithm goes a step forward towards efficient and accurate hypergraph matching in the presence of outliers.

### *4.5. Deficiencies*

In the above-mentioned experiments, the VGG, Middlebury and KITTI datasets are of stereo datasets, and each pair of images in these datasets are taken of the same object/scene from different viewpoints. In the literature of (hyper)graph matching, there are another type of matching, namely semantic matching [39], where a pair of images are taken of the same type of objects but not the same object. We conduct semantic keypoint matching experiments with the Pascal VOC dataset with Berkeley annotations [40]. This dataset contains 2819 training images and 2996 validation images from the Pascal 2009 person

Table 4: Average matching results on the Pascal VOC dataset.

| | # model | # test | # true | # matches obtained | accuracy | # true obtained | runtime (s) |
|---|---|---|---|---|---|---|---|
| MPM | 12.24 | 12.52 | 8.83 | 12.24 | 0.08 | 1.00 | 0.15 |
| RRWM | | | | 12.24 | 0.11 | 1.32 | 0.16 |
| SM | | | | 12.24 | 0.08 | 0.91 | 0.15 |
| HGM | | | | 12.24 | 0.13 | 1.55 | 0.38 |
| TM | | | | 12.24 | 0.16 | 1.87 | 0.41 |
| RRWHM | | | | 12.24 | 0.18 | 2.22 | 0.18 |
| BCA | | | | 12.24 | 0.20 | 2.43 | 0.16 |
| MC-HDSet | | | | 6.29 | 0.09 | 0.59 | 36.60 |
| Ours | | | | 8.96 | 0.20 | 1.80 | 0.78 |

category, and each image is annotated with at most 20 keypoints which correspond to 20 human parts. We match the 2819 training images to the first 2819 validation images and obtain 2819 image pairs. Then 2 image pairs are removed as the number of keypoints in either the training or validation image is less than 3 and not applicable to hypergraph matching. The average matching results on the 2817 image pairs are reported in Table 4. Note that the GTHM algorithm relies on SIFT matching of feature points and is therefore not applicable in this set of experiments.

We observe from Table 4 that all the nine algorithms generate rather low matching accuracies on the Pascal VOC dataset. Comparatively, the hypergraph matching algorithms perform better than graph matching algorithms in matching accuracy. The matching accuracy of our algorithm is also not satisfactory, but it is still comparable to the best result (0.20) of the BCA algorithm. As all these algorithms generate low matching accuracies on this dataset, the reason of the low matching accuracy does not lie in our algorithm. In our opinion, the low matching accuracies are mainly caused by the learning-free affinity between (hyper)edges of these algorithms. In these algorithms, the second-order affinity is usually calculated by comparing the edge lengths, and third-order similarity

by comparing the three angles or sins of the three angles. These (hyper)edge affinities work well for the stereo datasets in our experiments, where the viewpoint change is usually small and visual information is available. However, in semantic matching, the corresponding (hyper)edges may experience significant changes in shapes. In other words, corresponding edges may not be similar in edge length and corresponding hyperedges may not be similar in three angles. As a result, the learning-free affinities based on edge lengths or triangle angles are not accurate enough for semantic matching.

Based on the experimental results in Tables 1 to 4, our algorithm has two major deficiencies. First, the learning-free affinity based on comparison of triangles does not work in the case of large viewpoint change, especially in semantic matching. Second, our algorithm is still computationally expensive in comparison with other algorithms. Detailed inspection of the matching process shows that the major computation load of our algorithm comes from the building of the affinity tensor, i.e., for each model hyperedge finding the $n_n$ most similar test hyperedges. These two problems call for further works to improve our algorithm.

## 5. Conclusions

We present a game-theoretic hypergraph matching algorithm to obtain a large number of matches efficiently with a high accuracy. First, our algorithm casts hypergraph matching as a non-cooperative multi-player game, and obtains the final matches by extracting a group of consistent candidate matches. Compared with existing assignment based algorithms, our approach rejects the false matches from outliers in features and generates a high matching accuracy. Second, we discuss the reason behind the small number of matches, and then propose a density enhancement method to obtain more matches efficiently. Third, two methods are proposed to enforce the one-to-one constraint, thereby maintaining a high matching accuracy with evidently increased matches. Compared with two other game-theoretic algorithms GTHM and MC-HDSet,

our algorithm generates more matches or reduces the computation load, while achieving the same high matching accuracy. Finally, we investigate the influence of parameters on matching results, based on which we recommend the appropriate parameters involved in our algorithm. Experiments on synthetic point pattern matching and real feature matching demonstrate the performance of our algorithm.

One major problem of our algorithm is that the learning-free affinity between hyperedges does not work for semantic keypoint matching with large viewpoint changes between a pair of images. Possible approaches to solve this problem include exploring more generic affinity models and combining with existing learning-based affinity algorithms. Another problem lies in the large computation load, which requires further works on efficient building of affinity tensors. We plan to reduce the computation load from two aspects, i.e., reducing the number of model hyperedges and finding the most similar test hyperedges more efficiently. In reducing the number of model hyperedges, a possible approach is to sample the model hyperedges in some way to accommodate both local and global consistency among matches with less model hyperedges. In finding the most similar test hyperedges, we will explore more efficient (approximate) nearest neighbor methods and adaptive methods to determine the number of most similar test hyperedges.

**Acknowledgement**

This work is supported in part by the National Natural Science Foundation of China under Grant No. 62176057.

**References**

[1] J. Ma, X. Jiang, A. Fan, J. Jiang, J. Yan, Image matching from handcrafted to deep features: A survey, International Journal of Computer Vision 129 (1) (2021) 23–79.

[2] B. Jiang, P. Sun, B. Luo, GLMNet: Graph learning-matching convolutional networks for feature matching, Pattern Recognition 121 (2022) 1–7.

[3] C. Wang, X. Wang, J. Zhang, L. Zhang, X. Bai, X. Ning, J. Zhou, E. Hancock, Uncertainty estimation for stereo matching based on evidential deep learning, Pattern Recognition 124 (2022) 1–13.

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

[5] M. Cho, J. Sun, O. Duchenne, J. Ponce, Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2014, pp. 2091–2098.

[6] L. Bai, L. Rossi, Z. Zhang, E. R. Hancock, An aligned subtree kernel for weighted graphs, in: International Conference on Machine Learning, 2015, pp. 30–39.

[7] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: IEEE International Conference on Computer Vision, 2005, pp. 1482–1489.

[8] Z. Zhang, J. McAuley, Y. Li, W. Wei, Y. Zhang, Q. Shi, Dynamic programming bipartite belief propagation for hyper graph matching, in: International Joint Conference on Artificial Intelligence, 2017, pp. 4662–4668.

[9] R. Zhang, W. Wang, Second- and high-order graph matching for correspondence problems, IEEE Transactions on Circuits and Systems for Video Technology 28 (10) (2018) 2978–2992.

[10] J. Hou, H. Yuan, Efficicent and accurate hypergraph matching, in: International Conference on Multimedia and Expo, 2021, pp. 1–6.

[11] H. Zhang, P. Ren, Game theoretic hypergraph matching for multi-source image correspondences, Pattern Recognition Letters 87 (2017) 87–95.

[12] J. Hou, M. Pelillo, H. Yuan, Hypergraph matching via game-theoretic hypergraph clustering, Pattern Recognition 125 (2022) 1–12.

[13] J. Yan, S. Yang, E. Hancock, Learning for graph matching and related combinatorial optimization problems, in: International Joint Conference on Artificial Intelligence, 2020, pp. 4988–4996.

[14] R. Zass, A. Shashua, Probabilistic graph and hypergraph matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[15] M. Chertok, Y. Keller, Efficient high order matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (12) (2010) 2205–2215.

[16] O. Duchenne, F. Bach, I. Kweon, J. Ponce, A tensor-based algorithm for high-order graph matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2009, pp. 1980–1987.

[17] J. Lee, M. Cho, K. M. Lee, Hyper-graph matching via reweighted random walks, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1633–1640.

[18] Q. Nguyen, A. Gautier, M. Hein, A flexible tensor block coordinate ascent scheme for hypergraph matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2015, pp. 5270–5278.

[19] M. Leordeanu, A. Zanfir, C. Sminchisescu, Semi-supervised learning and optimization for hypergraph matching, in: IEEE International Conference on Computer Vision, 2011, pp. 2274–2281.

[20] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, S. M. Chu, Discrete hypergraph matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2015, pp. 1520–1528.

[21] J. Yan, C. Li, Y. Li, G. Cao, Adaptive discrete hypergraph matching, IEEE Transactions on Cybernetics 48 (2) (2018) 765–779.

[22] M. Leordeanu, R. Sukthankar, M. Hebert, Unsupervised learning for graph matching, International Journal of Computer Vision 96 (2012) 28–45.

[23] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, in: IEEE International Conference on Computer Vision, 2013, pp. 25–32.

[24] L. Bai, L. Cui, Y. Jiao, L. Rossi, E. R. Hancock, Learning backtrackless aligned-spatial graph convolutional networks for graph classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2) (2022) 783 – 798.

[25] A. Zanfir, C. Sminchisescu, Deep learning of graph matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2018, pp. 2684 – 2693.

[26] R. Wang, J. Yan, X. Yang, Learning combinatorial embedding networks for deep graph matching, in: IEEE International Conference on Computer Vision, 2019, pp. 3056–3065.

[27] L. Bai, Y. Jiao, L. Cui, L. Rossi, Y. Wang, P. Yu, E. Hancock, Learning graph convolutional networks based on quantum vertex information propagation, IEEE Transactions on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2021.3106804.

[28] Y. Li, C. Gu, T. Dullien, O. Vinyals, P. Kohli, Graph matching networks for learning the similarity of graph structured objects, in: International Conference on Machine Learning, 2019, pp. 3835–3845.

[29] M. Fey, J. E. Lenssen, Deep graph matching consensus, in: International Conference on Learning Representations, 2020, pp. 1–23.

[30] J. W. Weibull, Evolutionary Game Theory, Cambridge University Press, 1995.

[31] S. Rota Bulò, M. Pelillo, A game-theoretic approach to hypergraph clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (6) (2013) 1312–1327.

[32] M. Pavan, M. Pelillo, Dominant sets and pairwise clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 167–172.

[33] S. Rota Bulò, M. Pelillo, Dominant-set clustering: A review, European Journal of Operational Research 262 (1) (2017) 1–13.

[34] X. Tao, T. Cui, A. Plaza, P. Ren, Simultaneously counting and extracting endmembers in a hyperspectral image based on divergent subsets, IEEE Transactions on Geoscience and Remote Sensing 58 (12) (2020) 8952–8966.

[35] M. Ester, H. P. Kriegel, J. Sander, X. W. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[36] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, P. Westling, High-resolution stereo datasets with subpixel-accurate ground truth, in: German Conference on Pattern Recognition, 2014, pp. 31–42.

[37] M. Menze, C. Heipke, A. Geiger, Object scene flow, ISPRS Journal of Photogrammetry and Remote Sensing (JPRS) 140 (2018) 60–76.

[38] M. Cho, J. Lee, K. M. Lee, Reweighted random walks for graph matching, in: European Conference on Computer Vision, 2010, pp. 492–505.

[39] R. Wang, J. Yan, X. Yang, Neural graph matching network: Learning lawlers quadratic assignment problem with extension to hypergraph and multiple-graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (9) (2022) 5261 – 5279.

[40] L. Bourdev, S. Maji, T. Brox, J. Malik, Detecting people using mutually consistent poselet activations, in: European Conference on Computer Vision, 2010, pp. 168–181.