Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

# Finding Median Point-Set Using Earth Mover's Distance*

**Hu Ding** and **Jinhui Xu**
Computer Science and Engineering, State University of New York at Buffalo
{huding, jinhui}@buffalo.edu

**Abstract**

In this paper, we study a prototype learning problem, called *Median Point-Set*, whose objective is to construct a prototype for a set of given point-sets so as to minimize the total Earth Mover's Distances (EMD) between the prototype and the point-sets, where EMD between two point-sets is measured under affine transformation. For this problem, we present the first purely geometric approach. Comparing to existing graph-based approaches (*e.g.,* median graph, shock graph), our approach has several unique advantages: (1) No encoding and decoding procedures are needed to map between objects and graphs, and therefore avoid errors caused by information losing during the mappings; (2) Staying only in the geometric domain makes our approach computationally more efficient and robust to noise. We evaluate the performance of our technique for prototype reconstruction on a random dataset and a benchmark dataset, handwriting Chinese characters. Experiments suggest that our technique considerably outperforms the existing graph-based methods.

## 1 Introduction

Finding the *prototype* of a set of examples (or observations) is an important problem frequently encountered in pattern recognition and computer vision. A commonly used approach for this problem is to first encode each example into a graph, and then compute the median of the graphs as the prototype (Jiang, Münger, and Bunke 2001; Sebastian, Klein, and Kimia 2001; Macrini, Siddiqi, and Dickinson 2008; Demirci, Shokoufandeh, and Dickinson 2009; Trinh and Kimia 2010; Ferrer et al. 2011; Ding et al. 2013). Such an approach although is quite general, it also suffers from several key limitations. Firstly, it could introduce a considerable amount of error due to information losing during the encoding and decoding processes. Secondly, since graph median often needs to solve some high complexity graph (or subgraph) matching problem, its computation could be rather inefficient. Thirdly, for complicated examples, a graph representation may not always be sufficient to capture all necessary features.

*The research of this work was supported in part by NSF under grant IIS-1115220.


To provide a remedy to the above issues, in this paper, we consider a family of prototype learning problems in some geometric (*e.g.,* Euclidean) space, and propose a purely geometric approach. In our problem, each example is represented as a point-set in Euclidean space, and the output is a new point-set, called median point-set (*i.e.,* prototype), in the Euclidean space with minimum total *Earth Mover's Distance* (EMD) (Cohen and Guibas 1999) to all input examples, where EMD is measured under affine transformations.

Our problem is motivated by a number of applications. One type of applications (*e.g.,* information fusion) is to accurately reconstruct an object from a set of its observations. Since each observation may only have partial information of the object and could contain various types of noises/errors, it thus can be modeled as an unsupervised prototype learning problem. Another type of applications is for object recognition and classification. In such applications, we are given a set of structured data items (or objects) in advance, and need to find the best match for each query object. Since the number of data items might be very large, a straightforward way of search could be quite costly. A more efficient way is to first partition the data items into clusters, build a prototype for each cluster, and then find the best match by comparing only to the prototypes.

To validate our approach, we test it using a random dataset and a benchmark dataset, handwriting Chinese characters, and compare its performance with some existing graph-based methods. Experimental results suggest that our technique considerably outperforms existing ones on both accuracy and robustness.

### 1.1 Related Works

Finding median graph from a set of given graphs is a problem with a long and rich history in pattern recognition. (Jiang, Münger, and Bunke 2001) first introduced the median graph concept, and presented a genetic algorithm to solve it. Later, (Ferrer et al. 2011) proposed an embedding algorithm. Their idea is to first map the graphs into some metric space, then find the median point in the metric space, and finally map the median point back to some graph. Their main drawback is the high computational complexity caused by computing the best graph matching.

When the input is a set of shapes, a frequently used strategy is to represent each shape as a graph abstraction. For

examples, (Demirci, Shokoufandeh, and Dickinson 2009) and (Macrini, Siddiqi, and Dickinson 2008) used the medial axis graph as the graph abstraction; (Sebastian, Klein, and Kimia 2001) and (Trinh and Kimia 2010) adopted a similar approach called shock graph to represent the graph abstraction. As pointed out in article (Demirci, Shokoufandeh, and Dickinson 2009), the main drawback of these approaches is the requirement of accurate segmentation of the raw images, since otherwise, the shape abstraction would be difficult to obtain. Another drawback is that a graph abstraction of a complicated shape (*e.g.,* handwriting characters) may lose a significant amount of information.

As a similarity measurement for images, Earth Mover's Distance has been widely used in computer vision (Cohen and Guibas 1999; Rubner, Tomasi, and Guibas 2000; Giannopoulos and Veltkamp 2002; Grauman and Darrell 2004; Pele and Werman 2009). EMD can be viewed as a generalization of the bipartite matching, *i.e.,* from one-to-one matching to many-to-many matching, and has applications in many other areas. As observed in (Giannopoulos and Veltkamp 2002), a major advantage of EMD (over bipartite matching) is its robustness to noise. Such a generalization makes it more challenging to find the optimal solution, especially when the images can perform certain transformations. Article (Cohen and Guibas 1999) provided an iterative algorithm to find a local optimum for EMD under rigid transformation. Later, (Klein and Veltkamp 2005; Cabello et al. 2008; Ding and Xu 2013) presented approximate solutions, which are mainly of theoretical importance.

It is worth pointing out that (Ricci et al. 2013; Zen and Ricci 2011) also used EMD to compute a smooth representation of each short video clip and called it "prototype." Their problem has several significant differences from ours. (1) In their problem, a sequence of "prototypes" is computed for detecting activity, while in our problem only one prototype is computed from a set of exemplars. (2) In their problem, EMD is computed by treating objects in each frame as static data, while in our problem, EMD is computed under affine transformation (*i.e.,* point-sets can perform some affine transformations).

### 1.2 Our Contributions

One of our main contributions for this unsupervised prototype learning problem is the first purely geometric algorithm. Comparing to existing graph-based approaches, our approach has two major advantages. (1) Our approach does not need the encoding and decoding procedures mapping between the objects and their corresponding graphs. Consequently, it avoids errors caused by information losing during the mappings. (2) Since our approach always stays in the continuous Euclidean space and does not rely on graph matching, our algorithm is more efficient and robust.

## 2 Preliminaries

In this section, we introduce some definitions which will be used throughout the paper.

**Definition 1 (Earth Mover's Distance (EMD)).** *Let $A = \{p_1, \cdots, p_n\}$ and $B = \{q_1, \cdots, q_m\}$ be two sets of weighted points in $\mathbb{R}^d$ with nonnegative weights $\alpha_i$ and $\beta_j$ for each $p_i \in A$ and $q_j \in B$ respectively, and $W^A$ and $W^B$ be their respective total weights. The earth mover's distance between $A$ and $B$ is*

$$EMD(A,B) = \frac{\min_F \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}||p_i - q_j||}{\min\{W^A, W^B\}}, \quad (1)$$

*where $F = \{f_{ij}\}$ is a feasible flow satisfying the following conditions. (1) $f_{ij} \geq 0$, $\sum_{j=1}^{m} f_{ij} \leq \alpha_i$, and $\sum_{i=1}^{n} f_{ij} \leq \beta_j$ for any $1 \leq i \leq n$ and $1 \leq j \leq m$; (2) $\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij} = \min\{W^A, W^B\}$.*

**Definition 2 (Affine Transformation).** *In $2D$, affine transformation is the set of transformations including linear transformation, translation, and their combinations, where linear transformation is determined by a $2 \times 2$ matrix, and translation is determined by a 2D vector.*

**Note that** from a geometric point of view, linear transformation can be decomposed into rotation, scaling, reflection, shear mapping, and squeeze mapping (see Fig. 1 a).

Now, we discuss the main problem below.

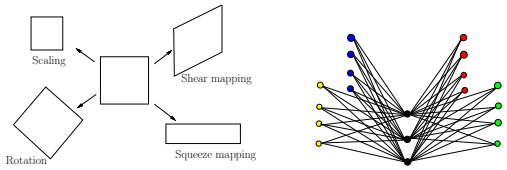
**Definition 3 (Median Point-Set Using Earth Mover's Distance ($\mathbb{M}_{emd}$)).** *Given a set of weighted point-sets $\{P_1, \cdots, P_n\}$ and an integer $m > 0$, the median point-set problem under EMD is to determine a new weighted point-set $Q$ containing no more than $m$ points such that the following objective function is minimized*

$$\sum_{i=1}^{n} \min_{\mathcal{T}} EMD(Q, \mathcal{T}(P_i)), \quad (2)$$

*where $\mathcal{T}(P_i)$ is the affine transformation of $P_i$. The point-set $Q$ achieving the optimal objective value is denoted as $\mathbb{M}_{emd}$. The weights of each $P_i$ and $Q$ are assumed to be normalized to the same value.*

**Note:** In Definition 3, some reasonable constraints may be imposed to avoid meaningless result. For example, if the coefficient of a scaling transformation were 0, every $P_i$ would be mapped into a single point, and the objective value (2) would become 0 after translating all these single points to make them overlap with each other. Thus, in order to avoid this, we first normalize the diameter of all the initial input point-sets to be the same, and select one point-sets as the initial estimation of $\mathbb{M}_{emd}$ (see the details in Section 4.2). Consequently, the point-sets would not be transformed into single points during the algorithm.

**Main Challenges:** There are two main difficulties in finding $\mathbb{M}_{emd}$. The first one is due to the intrinsic difficulty of determining the median of a set of geometric structures. Note that if each $P_i$ is only a single point, the problem is the median point (*i.e.,* Fermat-Weber point) problem and can be relatively easily solved. However, when each $P_i$ becomes a much more complicated geometric structure, there is no existing solution, to the best of our knowledge. The second one comes from the need of determining a proper affine transformation for each $P_i$. If $Q$ is already known, finding the transformation is equivalent to the minimum cost matching between $Q$ and each $P_i$. However, since $Q$ is unknown, the problem becomes much harder.




(a) (b)

Figure 1: (a) shows an example of rotation, scaling, shear mapping, and squeeze mapping respectively; (b) shows the LP-model (7)-(11), where the black nodes denote the points of the current estimation $E_t$, the nodes with the same (other) color denote a point-set $P_i$, and the edges denote the flows between each $P_i$ and $E_t$. The LP-model is to minimize the total cost of all flows.

## 3 Minimizing EMD under Affine Transformation

In this section, we consider the problem of minimizing EMD between two point-sets $A$ and $B$ under affine transformation. Clearly, this is a basic step for computing the function (2).

Note that if the two point-sets $A$ and $B$ are fixed (*i.e.,* no transformation is allowed), their EMD can be optimally computed by using linear programming (Cohen and Guibas 1999) or other more efficient algorithms mentioned in Section 1.1. However, if $A$ or $B$ can be transformed, the problem becomes much harder and available techniques are still quite limited (Cohen and Guibas 1999; Cabello et al. 2008; Klein and Veltkamp 2005; Ding and Xu 2013; Ding, Berezney, and Xu 2013) (*e.g.,* mainly considering rigid transformation). To the best of our knowledge, there is no efficient algorithm considering affine transformation.

**Flow-Transformation Iterative Algorithm.** (Cohen and Guibas 1999) introduced an iterative algorithm for minimizing EMD of two point-sets under rigid transformation (including translation and rotation). Their idea is as follows. Initially, $A$ and $B$ are fixed in the space, and the algorithm computes $EMD(A, B)$; then based on the obtained flows from $A$ to $B$, it computes an optimal transformation for $B$, such that the cost is minimized; the algorithm iteratively run this flow-transformation iteration until the cost converges to some local optimum. This idea is similar in spirit to a number of existing algorithms for some other optimization problems, such as *Iterative Closest Point* (ICP) (Besl and McKay 1992) and *Expectation-maximization* (EM) (McLachlan and Basford 1988) algorithms.

For our problem of minimizing EMD under affine transformation, the flow-transformation iterative algorithm is applicable after some modification. In each iteration, the flow step is still the same as in (Cohen and Guibas 1999), and the transformation step needs to compute an optimal affine transformation rather than an optimal rigid transformation. Let $\mathcal{T}$ be any affine transformation containing a linear transformation (*i.e.,* a $2 \times 2$ matrix) $\mathcal{M}$ and a translation (*i.e.,* a $2 \times 1$ matrix) $\mathcal{S}$. Then, for any point $q_i \in B$, $\mathcal{T}(q_i) = \mathcal{M}q_i + \mathcal{S}$, and the cost function from Definition 1 becomes

$$Cost(A, \mathcal{T}(B)) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}||p_i - (\mathcal{M}q_i + \mathcal{S})||}{\min\{W^A, W^B\}}. \quad (3)$$

Note that the flows $\{f_{ij}\}$ are obtained in the transformation step. From the cost function (3), it is easy to see that it is convex on the 6 variables (4 from $\mathcal{M}$ and 2 from $\mathcal{S}$). To minimize the cost function (3), we can relax it to $L_2^2$-norm (*i.e.,* replace the Euclidean distance $||p_i - (\mathcal{M}q_i + \mathcal{S})||$ by the squared Euclidean distance $||p_i - (\mathcal{M}q_i + \mathcal{S})||^2$). Then the problem of miniming (3) becomes a standard $L_2^2$-norm convex optimization problem with respect to the 6 variables, which can be easily solved in linear time.

**Algorithm 1** Median point-set (outline)

**Input:** $\{P_1, \cdots, P_n\}$ and $m \in Z^+$.
**repeat**
1. Dynamic step.
2. Static step, with the following inner loop:
   (a) Location updating.
   (b) Weight updating.

**until** The objective value becomes stable.

## 4 Algorithm for Median Point-Set

### 4.1 Overview

Before discussing the details of the algorithm, we first outline the main steps in Algorithm 1. The terms "dynamic" and "static" are used to indicate the status of the $n$ input point-sets. "Dynamic" means that the position of each $P_i$ is changed using some transformation, and "static" means that the position of each $P_i$ remains unchanged.

In each iteration, we maintain an estimation for $\mathbb{M}_{emd}$ (which is the current best solution of $\mathbb{M}_{emd}$). In the dynamic step, we fix this estimation, and find a good transformation for each $P_i$ using the algorithm in Section 3. In the static step, we update the estimation based on the $n$ transformed point-sets from the dynamic step. The static step is a loop consisting of two sub-steps in its body, location updating and weight updating. Location updating modifies the location of each point in the current estimation, and weight updating changes the distribution of the weight. We will unfold our ideas for each step in the following sections.

### 4.2 Initial Estimation for $\mathbb{M}_{emd}$

As mentioned in the overview, the algorithm computes an estimation for $\mathbb{M}_{emd}$ in each iteration. We first show that a good initial estimation leads to a good approximation.

**Theorem 1.** *Let $P_l$ be a point-set randomly selected from the input (*i.e., *$P_l$ is one of $n$ point-sets in the input). Then with probability $1/2$, there exists an affine transformation $\tilde{\mathcal{T}}$ such that $\tilde{\mathcal{T}}(P_l)$ yields a 3-approximation for $\mathbb{M}_{emd}$ (with respect to the objective function (2)).*

*Proof.* Let $\mathcal{T}_i$ be the affine transformation for $P_i$ in the optimal solution. By Markov inequality, we have

$$EMD(\mathbb{M}_{emd}, \mathcal{T}_l(P_l)) \leq 2 \times \frac{1}{n}\sum_{i=1}^{n} EMD(\mathbb{M}_{emd}, \mathcal{T}_i(P_i)) \quad (4)$$

with probability $1/2$. In Definition 3, we have assumed that the weights of each $P_i$ and $\mathbb{M}_{emd}$ are normalized to be the same. This means that $EMD$ satisfies the triangle inequality (Giannopoulos and Veltkamp 2002). Thus, we have

$$\begin{aligned} & \frac{1}{n}\sum_{i=1}^{n} EMD(\mathcal{T}_l(P_l), \mathcal{T}_i(P_i)) \\ \leq & \frac{1}{n}\sum_{i=1}^{n}(EMD(\mathcal{T}_l(P_l), \mathbb{M}_{emd}) + EMD(\mathbb{M}_{emd}, \mathcal{T}_i(P_i))) \\ \leq & 3 \times \frac{1}{n}\sum_{i=1}^{n} EMD(\mathbb{M}_{emd}, \mathcal{T}_i(P_i)) \end{aligned} \quad (5)$$

with probability $1/2$, where the first inequality follows from triangle inequality and the second inequality follows from (4). This means that we can choose $\mathcal{T}_l$ as $\tilde{\mathcal{T}}$, and the theorem follows from inequality (5). □

After fixing $P_l$, we still need to resolve one issue in order to obtain an initial estimation for $\mathbb{M}_{emd}$: $P_l$ may have more than $m$ points. Thus, we perform $k$-medians clustering (let $k = m$) on $P_l$ and use the $m$ median points $\{c_1, \cdots, c_m\}$ as the initial estimation, where each $c_j$ is associated with a weight equal to the total weight of the corresponding cluster.

### 4.3 Dynamic Step

The dynamic step is relatively simple. In this step, we fix the position of the current estimation for $\mathbb{M}_{emd}$, and determine an appropriate transformation (using the algorithm in Section 3) for each input point-set $P_i$ so as to minimize the EMD from the estimation to $P_i$. Clearly, the value of the objective function (2) can only be improved in this step.

### 4.4 Static Step

In this step, we fix the position of each input point-set after performing its corresponding transformation determined in the dynamic step, and update the position and weight of the current estimation. Let $E_t = \{e_1^t, \cdots, e_m^t\}$ be the estimation in the $t$-th iteration of the inner loop in the static step, where $e_j^t$ is associated with a weight $w_j^t$. The updating is done iteratively on the location of each $e_j^t$ and the weight distribution among $E_t$.

For ease of discussion, we let $\{p_1^i, \cdots, p_{N_i}^i\}$ be the point-set $P_i$, $\alpha_s^i \geq 0$ be the weight of $p_s^i$, and $W$ be the normalized weight of each $P_i$. With a slight abuse of notation, we still use $P_i$ and $p_s^i$ to denote the corresponding point-set and point after performing the transformation obtained in the dynamic step. Let $f_{s,j}^i$ be the weight flow from $p_s^i$ to $e_j^t$ under the current matching between $E_t$ and $P_i$ (see Definition 1).

**Location updating:** From the functions (1) & (2), we know that the contribution of $e_j^t$ to the objective function (2) is

$$\frac{1}{W}\sum_{i=1}^{n}\sum_{s=1}^{N_i} f_{s,j}^i ||e_j^t - p_s^i||. \quad (6)$$

To determine the new location of $e_j^t$, consider the set of input points $\{p_s^i \mid 1 \leq i \leq n, 1 \leq s \leq N_i\}$. We can imagine that each input point $p_s^i$ is associated with a "new weight" $f_{s,j}^i$. Then, the value of (6) is minimized when the location of $e_j^t$ is at the geometric median of the set of "weighted" points. This means that we can use the algorithm (Weiszfeld 1937) to calculate the new location of $e_j^t$. Obviously, after performing this location updating for each $e_j^t$, the value of the objective function (2) can only be improved.

**Weight updating:** In this step, we fix the position of each $e_j^t$, and re-calculate the weight distribution among these $m$ points in the estimation. Below we formulate the problem as a linear programming problem. We treat $w_j^t$ and $f_{s,j}^i$ as variables, then we have the following LP-model for minimizing the objective function (2).

$$\min \ g = \frac{1}{W}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{s=1}^{N_i} f_{s,j}^i ||e_j^t - p_s^i|| \quad (7)$$

$$\sum_{s=1}^{N_i} f_{s,j}^i \leq w_j^t, \forall 1 \leq i \leq n, 1 \leq j \leq m \quad (8)$$

$$\sum_{j=1}^{m} f_{s,j}^i \leq \alpha_s^i, \forall 1 \leq i \leq n, 1 \leq s \leq N_i \quad (9)$$

$$\sum_{j=1}^{m}\sum_{s=1}^{N_i} f_{s,j}^i = W, \forall 1 \leq i \leq n \quad (10)$$

$$\sum_{j=1}^{m} w_j^t = W \quad (11)$$

In the above linear programming, $g$ in (7) is equivalent to the total EMD from $E_t$ to the $n$ input point-sets, and (8)-(11) are constraints for the flows based on Definition 1. Note that $||e_j^t - p_s^i||$ in (7) is a constant since the positions of both $e_j^t$ and $p_s^i$ are fixed. Figure 1b gives an illustration for the LP-model. From the above LP formulation, we immediately have the following theorem.

**Theorem 2.** *Finding the optimal weight distribution on $E_t$ in the weight updating step is equivalent to solving the LP-model (7)-(11).*

The static step repeatedly conduct the location updating and weight updating until the solution becomes stable (note that the objective value decreases in both location and weight updatings).

### 4.5 Summary of the Algorithm

From Sections 4.3 and 4.4, we know that the objective value of (2) keeps improving in both dynamic and static steps. Thus we have the following theorem.

**Theorem 3.** *Algorithm 1 for determining the median point-set converges on its objective value defined by (2).*

## 5 Further Improvements

In this section, we introduce several improvements for the above algorithm, which could significantly reduce the running time and improve the performance of our approach.

### 5.1 Improving the LP-Model Using Geometry

In the LP-model (7)-(11), we have a total of $(m\sum_{i=1}^{n} N_i + m)$ variables (all $f_{s,j}^i$ and $w_j^t$). Although efficient algorithms

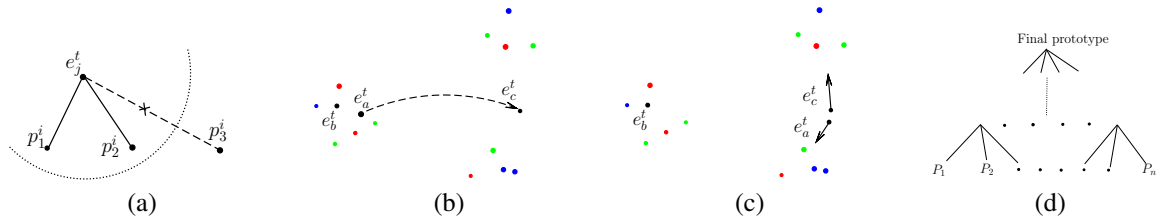



Figure 2: (a) is an example for the idea in Section 5.1: the matching between $e_j^t$ and $p_3^i$ is deleted since their distance is larger than some threshold; (b) and (c) illustrate the greedy jump procedure; (d) shows the divide-merge approach.

exist for solving LP (*e.g.,* simplex algorithm), it could still be slow if the dataset becomes very large.

To reduce the number of variables, we observe that the median point-set problem lies in Euclidean space. Thus it is natural to consider using the geometric information of the space to improve the efficiency of the LP-model. In formula (7), each variable $f_{s,j}^i$ has a coefficient $||e_j^t - p_s^i||$. From the definition of EMD, we know that if the distance between $e_j^t$ and $p_s^i$ is very large, it is unlikely that there will be a flow $f_{s,j}^i$ from $p_s^i$ to $e_j^t$, and the flow will be small even if there exists one. This means that in implementation, we can choose a threshold and ignore the the variable $f_{s,j}^i$ (*i.e.,* $f_{s,j}^i = 0$) if $||e_j^t - p_s^i||$ is larger than the threshold. Figure 2a shows an example. For a properly chosen threshold, this can significantly reduce the number of variables, and improve the efficiency of the algorithm.

### 5.2 Avoiding Local Optimum Trap: Greedy Jump

From Theorem 3, we know that the objective value will eventually converge. A possible problem is that it may converge to some local optimum. To have a better understanding, consider the example in Figure 2b. In this example, the black points denote the current estimation $E_t$, and the point with the same color denote one point-set $P_i$. All input points form three clusters. It is easy to see that an optimal solution should have one estimation point in each cluster. But in the example, $e_a^t$ and $e_b^t$ are trapped into the same cluster and $e_c^t$ is at the median point of the other two clusters. In such a scenario, a good strategy is to let $e_a^t$ jump to a new position near $e_c^t$, and re-run the algorithm such that $e_a^t$ and $e_c^t$ can move to two different clusters (see Figure 2c).

**Greedy jump:** To avoid the local optimum trap, we first check whether there exist any pair of points from $E_t$ which are very close. If there exist a close pair $e_a^t$ and $e_b^t$, we find the point from $E_t$ which has the largest contribution to the objective function (2), say $e_c^t$, *i.e.,*

$$c = \arg\max_{1\le j\le k} \frac{1}{W}\sum_{i=1}^{n}\sum_{s=1}^{N_i} f_{s,j}^i ||e_j^t - p_s^i||. \tag{12}$$

Then we shift the weight $w_a^t$ from $e_a^t$ to $e_b^t$, *i.e.,* the weight of $e_a^t$ becomes 0, and the weight of $e_b^t$ becomes $w_a^t + w_b^t$. Finally, we move the point $e_a^t$ to $e_c^t$, *i.e.,* $e_a^t$ become a new point coincident with $e_c^t$, and re-run the static step on the new estimation $E_t$. See Figure 2b and 2c for an illustration.

### 5.3 A Divide-Merge Approach for Speedup

To further improve our algorithm for large datasets, we consider the case that there are a large number of point-sets in the input. In this case, the algorithm could be slow if each point-set has a large number of points. For example, if there are 1000 point-sets with each containing 30 points and $m$ is 20, the total number of variables in the LP-model will be more than $20 \times 30 \times 1000 = 6 \times 10^5$. Even though we may reduce this number by using the idea presented in Section 5.1, it could still be quite challenging in practice.

**Main idea:** When the number of point-sets is large, our idea is to first divide the large dataset into a number of small groups, then compute the prototype for each small group, and finally compute the prototype for the prototypes from these small groups. This strategy can be easily extended to multi-levels. In each level, we divide the dataset into multiple small groups, solve them separately, and return the resulting prototypes to upper level; the bottom level is the original dataset, and the top level is the final output prototype. Figure 2d gives the hierarchical structure of this approach.

## 6 Experiments

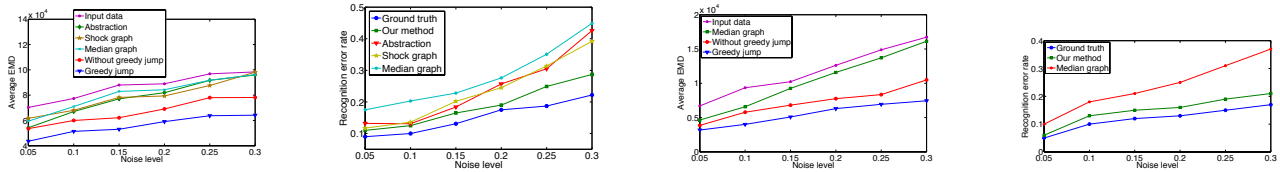
We test our prototype reconstruction algorithm (from Section 4) on two datasets: a random dataset and a benchmark dataset, handwriting Chinese characters (KAIST Hanja DB2 database). We consider two quality measurements: (1) the difference between the output prototype and the ground truth (*i.e.,* reconstruction error), and (2) the accuracy for query test (*i.e.,* recognition error).

For comparisons, we use the graph-based approaches in (Demirci, Shokoufandeh, and Dickinson 2009; Trinh and Kimia 2010; Ferrer et al. 2011). **Note** that for Chinese characters, we only use median graph method in (Ferrer et al. 2011). This is because the methods in (Demirci, Shokoufandeh, and Dickinson 2009; Trinh and Kimia 2010) require that each object be a connected region without any hole, and obviously this is not the case for Chinese characters.

**Random data:** We generate $6 \times 10$ random datasets with 6 different noise levels. For each noise level $t\%$, we randomly generate 10 weighted point-set $Q_t^1, \cdots, Q_t^{10}$ with the size varying from 30 to 60 in $2D$, and 10 corresponding datasets with each containing 100 weighted point-sets as follows: for each point $q \in Q_t^j$ associating a weight $w_q$, $1 \le j \le 10$, we replace it by multiple weighted points $\{q^1, \cdots, q^s\}$ around

(a) (b) (c) (d) (e) (f) (g)

Figure 3: (a) shows one case of the ground truth and the selected feature points; (b)-(g) depict the alignments of the ground truth and the reconstructed prototypes (represented by feature points) under 6 noise levels respectively.




(a) (b) (c) (d)

Figure 4: (a) and (b) depict the prototype construction errors and recognition errors for random data; (c) and (d) show the prototype construction errors and recognition errors for Chinese characters.

it ($s < 20$) with total weight equal to $w_q$, where the position of each $q^i$ is randomly selected following a Gaussian distribution with variance $t\%$ of the spread diameter of $Q_t^j$; then $\bigcup_{q \in Q_t^j} \{q^1, \cdots, q^s\}$ forms a weighted point-set; repeat this procedure 100 times to generate 100 point-sets. We set the noise level $t\%$ to be $\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$, and for each noise level, use the initial point-sets $Q_t^j$, $1 \leq j \leq 10$, as the ground truth of the prototype. We measure the performance by the EMD between our output prototype and $Q_t^j$ for each dataset. Figure 4a shows the average EMD for $j = 1, \cdots, 10$ under different noise levels. Further, for each noise level, we randomly generate another dataset containing $10 \times 100$ point-sets via the same generation method. Then we use the obtained prototypes to classify these 1000 point-sets (based on their EMD to prototypes). Figure 4b shows the recognition errors (*i.e.,* the number of wrong classifications over the total number of point-sets).

**Chinese characters:** We test our method on 50 characters. For each character, we build 6 datasets with different noise percentages $\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$, and each dataset contains 100 images of the character transformed by different affine transformations. Since each image contains thousands of pixels and it is too expensive to handle them directly, in the data representation step, we generate a set of *weighted feature points* to represent each image. More specifically, for an $N \times N$ gray level image, we consider the $N^2$ pixels as $N^2$ weighted points (with weights equal to their gray levels), then perform a $k$-means clustering (using Lloyd's algorithm) on them, and finally output the $k$ mean points as the feature points, with each associated with a weight equal to the total weight of the corresponding cluster (see Figure 3a for an example). We set the value of $k$ between 20 and 40. To measure the performance, we compute the EMD between the output prototype and the original character (ground truth). Figure 4c shows the results under different noise levels. Similar to random data, Figure 4d shows the recognition errors under different noise levels. In Figure 3, we give an example for the alignment of the computed prototypes (represented by feature points) and the ground truth under different noise levels.

**Analysis:** Comparing to the average EMD between the input datasets and the ground truth, our method can reduce the EMD obviously under each noise level in Figure 4a & c. As a comparison, we also run the graph-based approaches in (Demirci, Shokoufandeh, and Dickinson 2009; Trinh and Kimia 2010; Ferrer et al. 2011) on the same datasets. Results suggest that our method significantly outperforms the graph-based methods under all noise levels, and the advantage (for both of prototype construction error and recognition error) become even more obvious when the noise level increases, which indicates that our method is robust to noise. We attribute the improved performance to the following two facts: (1) Our method is purely geometric and therefore does not need to map the objects between Euclidean space and graph domain; (2) Our method adopts affine transformation to calculate EMD. These together help to improve both efficiency and robustness of our approach.

## 7 Conclusions

In this paper, we present a purely geometric approach for the prototype learning problem. Different from the existing graph-based methods, our approach does not need to map the object between Euclidean space and graph domain, and consequently, avoids the problem of information losing in existing methods. Experiments on random data and the benchmark data (handwriting Chinese characters) suggest that our approach significantly outperforms the existing graph-based methods, and is more robust to noise.

## References


Besl, P. J., and McKay, N. D. 1992. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14(2):239–256.

Cabello, S.; Giannopoulos, P.; Knauer, C.; and Rote, G. 2008. Matching point sets with respect to the earth mover's distance. *Comput. Geom.* 39(2):118–133.

Cohen, S. D., and Guibas, L. J. 1999. The earth mover's distance under transformation sets. In *ICCV*, 1076–1083.

Demirci, M. F.; Shokoufandeh, A.; and Dickinson, S. J. 2009. Skeletal shape abstraction from examples. *IEEE Trans. Pattern Anal. Mach. Intell.* 31(5):944–952.

Ding, H., and Xu, J. 2013. Fptas for minimizing earth mover's distance under rigid transformations. In Bodlaender, H. L., and Italiano, G. F., eds., *ESA*, volume 8125 of *Lecture Notes in Computer Science*, 397–408. Springer.

Ding, H.; Berezney, R.; and Xu, J. 2013. k-prototype learning for 3d rigid structures. In Burges, C. J. C.; Bottou, L.; Ghahramani, Z.; and Weinberger, K. Q., eds., *NIPS*, 2589–2597.

Ding, H.; Stojkovic, B.; Berezney, R.; and Xu, J. 2013. Gauging association patterns of chromosome territories via chromatic median. In *CVPR*, 1296–1303. IEEE.

Ferrer, M.; Karatzas, D.; Valveny, E.; Bardají, I.; and Bunke, H. 2011. A generic framework for median graph computation based on a recursive embedding approach. *Computer Vision and Image Understanding* 115(7):919–928.

Giannopoulos, P., and Veltkamp, R. C. 2002. A pseudometric for weighted point sets. In Heyden, A.; Sparr, G.; Nielsen, M.; and Johansen, P., eds., *ECCV (3)*, volume 2352 of *Lecture Notes in Computer Science*, 715–730. Springer.

Grauman, K., and Darrell, T. 2004. Fast contour matching using approximate earth mover's distance. In *CVPR (1)*, 220–227.

Jiang, X.; Münger, A.; and Bunke, H. 2001. On median graphs: Properties, algorithms, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.* 23(10):1144–1151.

Klein, O., and Veltkamp, R. C. 2005. Approximation algorithms for computing the earth mover's distance under transformations. In Deng, X., and Du, D.-Z., eds., *ISAAC*, volume 3827 of *Lecture Notes in Computer Science*, 1019–1028. Springer.

Macrini, D.; Siddiqi, K.; and Dickinson, S. J. 2008. From skeletons to bone graphs: Medial abstraction for object recognition. In *CVPR*. IEEE Computer Society.

McLachlan, G., and Basford, K. 1988. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.

Pele, O., and Werman, M. 2009. Fast and robust earth mover's distances. In *ICCV*, 460–467. IEEE.

Ricci, E.; Zen, G.; Sebe, N.; and Messelodi, S. 2013. A prototype learning framework using emd: Application to complex scenes analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(3):513–526.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision* 40(2):99–121.

Sebastian, T. B.; Klein, P. N.; and Kimia, B. B. 2001. Recognition of shapes by editing shock graphs. In *ICCV*, 755–762.

Trinh, N. H., and Kimia, B. B. 2010. Learning prototypical shapes for object categories. In *Proceedings of CVPR Workshop on Structured Models in Computer Vision (SMiCV'10)*. IEEE.

Weiszfeld, E. 1937. On the point for which the sum of the distances to n given points is minimum. *Tohoku. Math. Journal.* 1(1):355–386.

Zen, G., and Ricci, E. 2011. Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes. In *CVPR*, 3225–3232. IEEE.