CS 5114: THEORY OF ALGORITHMS

PROJECT REPORT

TEAM MEMBERS:

PRANAVI RAMBHAKTA

SUDHA RAVALI YELLAPANTULA

SHRUTI SHETTY

SOURABH SHETTY

Date: 12/12/2017

ABSTRACT:

The Earth Mover's Distance(EMD) is often used in the field of computer vision to compare the distributions of images based on content. It makes use of variable-size descriptions of distributions called signatures to improve expressiveness and efficiency over former histogram matching techniques. Signatures represent the compressed or approximate, dominant features of the original distribution. EMD is based on a solution to the transportation problem from linear optimization where difference between content like colour and texture are measured in terms of distances. [1] The problem is to minimise the cost required for transforming one distribution to another.

In this paper [2], an N log O(1) N-time randomized O(1)- approximation algorithm is used for computing the cost of minimum bichromatic matching between two planar point sets of size N. EMD(A,B) is defined to be the minimum cost of a perfect matching with edges between A and B. In order to produce an efficient algorithm for computing EMD(A, B), The main result of this paper is a constant factor approximation algorithm with running time N log O(1). The technique adopted in this method involves showing that the matching between the whole sets A and B can be decomposed into several matchings between subsets of those sets. It is observed that the cost of the total matching is well-approximated by the sum of the costs of (sub)-matchings computed for the subsets. Nonetheless, there might occur a case where the cost of one sub-matching dominates the whole sum. Hence, the sampling of matching uses random distribution where the probability of choosing a sub-matching is roughly proportional to its cost. This distribution can be computed by computing rough (logarithmic) approximation of the cost of each sub-matching which can be done very quickly. In this way, a small random sample of sub-matchings suffices to estimate the total cost.

The bipartite matching problem is a special case of the transportation problem. [3] In the Euclidean bipartite matching problem, two sets R and B where |R| = |B| = n are mapped such that no two edges share an endpoint and the sum of distances between the paired points is minimized. In the Euclidean non-bipartite matching, the two sets have different number of points. When the near-linear approximation algorithm uses a randomly shifted quadtree which is divided into small number of cells, the number of edges of even an approximate matching that cross the subdividing lines may be much larger than a constant or log n. This makes the number of subproblems in the natural dynamic programming approach too large. But in a near-linear time algorithm that gives at least a constant-factor approximation must exist, and that the subdivision due to a randomly shifted quadtree should be a useful tool. To ensure that the overall increase in cost is not too much we make sure that the number of levels in the quadtree is O(log(1/ε)). To do this we allow a cell of the quadtree to be partitioned into many sub cells. The size of the subproblems in the "merge" step may be quite large but we reduce this problem to a small-sized transportation problem. An algorithm that runs in O(n^(1+ε)) expected time and returns a matching whose expected cost is within O(1/ε) of the optimal is improved to that returns a matching whose expected cost is within O(log 1/ ε) of the optimal.

For point sets $A,B \subset Rd$, $|A| = |B| = n$, and for a parameter $\varepsilon > 0$, an algorithm is that computes, in $O(npoly(\log n, 1/\varepsilon))$ time, an ε-approximate perfect matching of A and B with high probability is presented [4]. Lp norm is approximated using a distance function, $d(\cdot, \cdot)$ based on a randomly shifted quad-tree. The algorithm iteratively generates an approximate minimum-cost augmenting path under the distance function in time proportional to the length of the path.The total length of the augmenting paths generated by the algorithm is $O((n/\varepsilon) \log n)$, ensuring that the algorithm runs in $O(npoly(\log n, 1/\varepsilon))$ time.

Optimal shape matching for a single shape in a database has superpolynomial time complexity. Even though hierarchical search methods, pruning, or triangle inequality may be used to improve search times, overall query times still have high complexity. To improve this complexity, Grauman and Darrell proposed a contour matching algorithm [4], which took into account recent developments in approximation techniques, and enabled fast shape-based similarity retrieval from the databases. They did this by treating the contour matching problem as a graph matching problem instead, and making use of the Earth Mover's. Distance. They also study how much the retrieval quality for the approximation method differs from that of the optimal graph matching method.

The Earth Mover's Distance

as a Metric for Image Retrieval[1]

The Earth Mover's Distance is introduced as a metric for content-based retrieval of images. Dissimilarity of images is used as a measure to find the difference in content of images. This paper discusses the different methods that have been used to calculate dissimilarity between two images based on histograms and proposes a new metric to overcome their limitations using a new data structure - signature.

The content of an image is represented as multidimensional distributions of its features like image intensities and texture information. For instance, the contents of a gray-scale image is represented as a one-dimensional distribution of image intensities, whereas, a color image is represented as a three-dimensional distribution of Red, Green and Blue(RGB) intensities. In order to save storage and processing time the images are compressed and saved in the form of histograms. The features can be represented in the form of integer vectors.

A histogram $\{h_i\}$ is defined as a mapping from a set of d-dimensional integer vectors **i** to the set of non-negative reals. Previous efforts were based on defining dissimilarity measures between histograms in order to find similar images. The image databases were indexed using histograms. Two types of dissimilarity measures discussed were:

1) Bin-by-Bin Dissimilarity Measures, and

2) Cross-Bin Dissimilarity Measures

Consider two images represented by their respective histograms $H = \{h_i\}$ and $K = \{k_i\}$. The *bin-by-bin* dissimilarity measures only compare contents of corresponding histogram bins, that is, they compare h_i and k_i for all i,where as, the *cross-bin* measures contains terms that compare non-corresponding bins, that is, h_i and k_j for $i \neq j$. These methods compared poorly to the perceptual dissimilarity of images.

The concept of variable-size descriptions of distributions called signatures was proposed to overcome the limitations of histograms (which consist of fixed number of bins). A clustering algorithm such as vector quantization is performed on the original distribution of an image and dominant clusters are extracted. A signature is used to represent these feature clusters. Therefore, a signature $\{ s_j = (\mathbf{m}_j, w_{\mathbf{m}j}) \}$represents a set of feature clusters. Each cluster is represented by its mean (or mode) $\mathbf{m}_j$, and by the fraction $w_{\mathbf{m}j}$ of pixels that belong to that cluster. The integer subscript j ranges from one to a value that varies with the complexity of the particular image. The ground distance for any two elements in signature set is defined as the distance between the centres of the respective clusters.

Given two distributions of land, both occupying the same area, one which has a mass of earth evenly distributed and the other has a collection. Then EMD measures the least amount of work needed to fill the space with earth. This intuition is used to find the dissimilarity in images. The proposed Earth Mover's Distance provides a metric based on signatures of different distributions for image retrieval. It can also be used for partial matches of images. The use of signatures significantly increases the efficiency of image retrieval systems. This is because in case of signatures the cost of moving large chunks of mass, that is, the clusters present in an image is minimized, whereas, in case of histograms cost of moving pixels is minimized.

The EMD is computed based on the transportation problem. The transportation problem is to find the least-expensive flow of goods from suppliers to consumers, given a set of suppliers and consumers, where for each supplier-consumer pair the cost of transporting a single unit of goods is given. While comparing two images in an image retrieval system, the signature of one image is treated as one of the suppliers and the signature of another image is treated like an element of the consumer set. The ground distance between two signatures is set at the cost for a supplier-consumer pair. The EMD is computed by finding the minimum amount of "work" required to transform one signature to another.

This is formalized as the following linear programming problem:

Let P = {($\mathbf{p}_1$, $\mathbf{w}_{p1}$), …….($\mathbf{p}_m$, $\mathbf{w}_{pm}$)} be the signature of the first image containing m clusters, where $\mathbf{p}_i$ is the cluster center/representative and $w_{\mathbf{p}}$, is the weight of the cluster; Q = {($\mathbf{q}_1$, w_{q1}), …….($\mathbf{q}_n$, $\mathbf{w}_{qn}$)} be the signature of the second image containing n clusters, where $\mathbf{q}_i$ is the cluster center/representative and $w_{\mathbf{q}}$, is the weight of the cluster; and **D** = [d_{ij}] the ground distance matrix where d_{ij} is the ground distance between clusters $\mathbf{p}_i$ and $\mathbf{q}_i$.

To find a flow **F =** [f_{ij}], with f_{ij} the flow between $\mathbf{p}_i$ and $\mathbf{q}_j$, that minimizes the total cost.

$$\text{WORK}(P, Q, \mathbf{F}) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij},$$

subject to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m,\ 1 \leq j \leq n \quad (1)$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{\mathbf{p}_i} \qquad 1 \leq i \leq m \quad (2)$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{\mathbf{q}_j} \qquad 1 \leq j \leq n \quad (3)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{\mathbf{p}_i}, \sum_{j=1}^{n} w_{\mathbf{q}_j}\right), \quad (4)$$

Then the EMD is given by :

$$\mathrm{EMD}(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}$$

The transportation problem can be solved using other efficient techniques such as the unc-apacitated minimum cost network flow problem which can be solved using bipartite graphs in $O(n^3 \log n)$, where n is the number of cluster in the signatures.

A Near Linear Time Constant Factor Approximation for Euclidean Bichromatic Matching (Cost)[2]

Two multisets A,B of points are considered where |A| = |B| = N. EMD(A,B) is defined to be the minimum cost of a perfect matching with edges between A and B, ie,

$$EMD(A,B) = \min_{\pi:A\to B} \sum_{a\in A} \|a - \pi(a)\|$$

where π ranges over all one-to-one mappings.

In the proposed algorithm, the matching between the sets A and B are decomposed into several matchings of those subsets in such a way that the cost of the resultant matching is well-approximated by the sum of costs of the sub-matchings computed. Since the cost of sub matchings vary, it is possible for such a case to arise where the cost of one submatching is able to dominate the total sum computed.

Hence, random distribution is chosen for the sampling of the submatching which would lead us to a better scenario where the probability of choosing a submatching is approximately equal to its cost.

The author computes a quick, rough logarithmic approximation of the cost of each submatching. Due to this, we arrive at a situation where a small random sample of submatchings is enough to estimate the total cost of the matching. It can be observed from this approach is designed to give an approximate cost but not an optimal matching.

Setup :

Now, the author applies simple and standard transformations to the input to make them more manageable, ie, this will result in the change of objective function cost by a factor of 1 + epsilon, as it is assumed that A and B are integer coordinates.

When a grid of length n = 2Tλ is shifted at random on a plane, any pair of points a,b would be "cut" by the grid with probability $\|a - b\|_1/n$. it follows that the probability that any edge of the minimal matching between A and B is cut is at most

$$EMD(A,B)/n \leq \lambda T/n \leq 1/2$$

Here, the upper probability is upto ½, and this problem consists of several bi-chromatic matching sub problems with n square subsets each.

Algorithm:

The algorithm is started by extending Earth Movers Distance to sets of non-equal size.
It can be observed that 2n is equal to the diameter of n^2, this would imply that, for this case, the minimum is would always be computed for |S| = |S'| = min(|A|,|B|).

For the first lemma, in the multisets A,B $\subset [n]^2$, |A|,|B|≤N.

$$EEMD_n(A,B) = \min_{S\subset A, S'\subset B, |S|=|S'|} [EMD(S,S') + n(|A-S| + |B-S'|)]$$

In the next step, EEMDn is decomposed into a sum of "m" metrics in terms of EEMDm where m is really small when compared to n. This decomposition is responsible for inducing distortion that is constant and does not vary.

The algorithm involves partitioning that enables us to reduce the original problem of large grid to several subproblems over smaller grids. This partitioning of the plane is done using randomly shifted grids.

The subproblems are constructed in an independent manner from one another. The reason for this is that the final estimation is performed by a biased sampling of the subproblems as assignment of higher probability mass is given to the large elements.

This decomposition result can be constructed as a low-distortion probabilistic embedding of EEMDn into a weighted sum of EEMDm's.

The decomposition procedure can be explained as follows.

In the first stage, An arbitrarily shifted grid G of cell length m, imposed over $[n]^2$. Initially, when G is interpreted as a set of cells, it is naturally associated with $[k]^2$ for k ≤ ⌈n/m⌉ + 1 ≤ 2n/m.

For any cell c ∈ G and multiset A $\subset [n]^2$, we define Ac = {a ∈ A : G(a) = c}, (which composes of subsets of $[m]^2$. Naturally, this grid decomposes to EEMDn in the following manner :

$$EEMD_n(A,B) \leq \sum_{c\in G} EEMD_m(A_c,B_c) + mEEMD_k(G(A),G(B))$$

The above inequality is true for any placement of the grid G on $[n]^2$. In the scenario where the grid G is shifted at random, an approximate version of the reverse inequality holds true as well.

The proof lies within the two statements (lemma 2 and 3) which are defined as :

(a) E[Z] ≤ 2·EEMDm(A,B) ; where Z = Pc∈G EEMDm (Ac,Bc)

(b) E [mEEMD(G(A),G(B))] ≤ EEMD(A,B)

Assume that, without loss of generality that |A|≤|B|. The probability that the edge (a,b) of the matching between A and S ⊂ B, |A| = |S| is cut by a randomly shifted grid is at most p(a,b) = ka−bk1 m. When an edge is cut we add at most of m to Z, otherwise we add ka−bk1 to Z. In the scenario that the edge is cut,

$$\begin{aligned} E[Z] &\leq m|A-B| + \sum_{(a,b)} (p(a,b)m + \|a-b\|_1) \\ &= m|A-B| + \sum_{(a,b)} (\|a-b\|_1 + \|a-b\|_1) \\ &\leq m|A-B| + 2EMD(A,S) \end{aligned}$$

In the next step, we apply the above lemmas in a recursive manner. Which would result in the imposition of grid G1 on $[n]^2$, then a grid G2 on G1, so on till Gt. Every grid has a cell length of m. This would imply that Gt has dimensions $M_t \times M_t$ where $M_t \leq n2^t/m^t$.

Consider for any i = 1...t, by applying second lemma i-1 times and third lemma once, we get E[Xi] ≤ 2·EEMD(A,B). Similarly, by applying second lemma "t" times we get E[Y] ≤ EEMD(A,B).

$$E[\sum_{i=1}^{t} X_i + Y] \leq (2t+1)EEMD(A,B)$$

At the same time, from the first lemma, we can draw a similarity to get EEMD(A,B) ≤ $\sum_{i=1}^{t} X_i + Y$ This is would lead us to the next step of them algorithm, where we can say that, for any δ > 0, EEMDn can be probabilistically embedded into a weighted sum of metrics EEMDm, m = n to the power δ (with non-negative weights), with distortion O(1/δ).

$$\begin{aligned} &EEMD_n(A,B) \leq \textstyle\sum_i w_i EEMD_m(f_i(A), f_i(B)) \\ &with\ probability\ 1 \\ &E[\textstyle\sum_i w_i EEMD_m(f_i(A), f_i(B))] \leq O(1/\delta) \cdot \\ &EEMD_n(A,B) \end{aligned}$$

In probability theory, Markov's inequality gives an upper bound for the probability that a non-negative function of a random variable is greater than or equal to some positive constant. Hence it can be extended in this case, to get Pr[S ≥ 4·O(1/δ) · EEMD(A,B)] ≤ 1/4. Also, S ≥ EEMD(A,B). Thus, it suffices to estimate S. For simplicity of notation, we (conceptually) replicate each EEMD metric several times, so that we have $w_i = 1$ for all i. Let Zi = EEMD(Ai,Bi), then $S = P_i Z_l$.

The estimation algorithm uses importance sampling where estimations Ei of Zi are computed such that Ei ≤ Zi ≤ λEi, which takes O ($N \log^{O(1)} N$) time.

Since each value Zi can be evaluated exactly in time $O(m^6)$ (using the Hungarian algorithm), it follows that the estimation of Zi can be done in time $O(\lambda m^6)$. We can choose the constant δ so that $\lambda m^6 = o(N)$. Thus, the total running time is $N \log^{O(1)} N$.

Conclusion:

In this paper, a constant-factor approximation has been presented for computing the cost of minimum bi-chromatic matching in the plane. The algorithm uses a combination of two ideas, mainly, a decomposition of EMD metric into several metrics over smaller domains, and calculating the total cost by sampling the metrics, using probabilities which approximate the costs of individual metrics.

A Near-Linear Constant-Factor Approximation for Euclidean Bipartite Matching[3]

Euclidean bipartite matching is pairing up a red point from a set R with a distinct blue point from a set B such that the sum of distances between the paired points in minimized. The red points and blue points belong to a set R^d where |R|=|B|=n. It is a special case of classical bipartite matching in a graph. Only two-dimensional Euclidean bipartite matching is considered. Straightforward translation of the two-dimensional algorithms is employed to get the best algorithms in any fixed dimension. The first polynomial time algorithm in the graph setting is the Hungarian algorithm. The running times were improved by using the concepts of scaling, geometric exploitation, divide and conquer and Monte Carlo randomized algorithm.

A generalization of the Euclidean bipartite matching (all demands are 1) is the transportation problem, in which we are given two sets of points U and V in R^2 and a positive integral demand λ(p) for each p ∈ U ∪ V and d(u, v) is the Euclidean distance between u and v.

$$\sum_{u \in U} \lambda(u) = \sum_{v \in V} \lambda(v)$$

A feasible solution to this problem is a subset M ⊆ U × V of edges and positive integral weights w (u, v) for each (u, v) ∈ M. The goal is to find a feasible solution M, w that minimizes

$$\sum_{(u,v) \in M} w(u,v) d(u,v)$$

$$\lambda(p) = \sum_{(p,q) \in M} w(p,q) \quad \forall p \in U \qquad \lambda(q) = \sum_{(p,q) \in M} w(p,q) \quad \forall q \in V$$

The non-bipartite and bipartite cases have similar running times for the implementation and scaling algorithms. The bipartite case can be more non-local than the non-bipartite case which results in the fact that bipartite matching is harder in the geometric setting. However, the near-approximation algorithm based on hierarchical decomposition of a point set by a randomly shifted quad-tree of the non-bipartite matching does not extend to the bipartite case. This is because of the large number of sub-problems in the dynamic programming approach generated by the large number of edges that cross the subdividing lines when a cell of the quad-tree is divided into four cells, even in an approximate matching. In spite of this problem, researchers felt that a linear time algorithm gives at least a constant-factor approximation using a randomly shifted quad-tree.

In this paper, a Monte Carlo algorithm for the two-dimensional bipartite matching problem is given that, for any 0 <ε < 1, runs in O(n1+ε) expected time and returns a matching whose expected cost is within O(log(1/ε)) of the optimal. A perfect bipartite matching is when each point in P is exactly present in exactly one pair.

Perfect bipartite matching P = R ∪ B is a subset M ⊆ R × B of red-blue pairs.

The cost of P is $\mu(P,M) = \sum_{(u,v)\in M} d(u,v)$ and the cost of the min-cost P $\mu(P) = \min_{M} \mu(P,M)$

M∗(P) be a min-cost matching of P. P′ be the point set obtained by "moving" each point p ∈ P to a point p′ and Δ = $\sum_{p\in P} d(p,p')$.

M is any perfect matching of P while M′ is the corresponding perfect matching of P′.

$$\mu(M) \leq \mu(P) + 2\Delta$$

A rough approximation of μ(P) is determined by computing in O(n log n) time a number α.

$$\alpha \leq \mu(P) \leq 2n^2\alpha.$$

The minimum spanning tree T of P can be computed in O(n log n) time. Let e1, . . . , e2n−1 be the edges of T in increasing order of their lengths. For 0 ≤ i ≤ 2n−1,let Gi denote the subgraph induced by the edges e1, . . . , ei, and let i^* be the smallest integer for which each component of G_{i*} has equal number of red and blue points. Given the ordering of the edges, i∗ can be computed in O(n) time. The length of ei∗ is the desired value of α. By a well known property of MSTs, $||e||_\infty \geq ||e||_{i*\infty}$ and $||e||_2 \geq ||e||_\infty$, we conclude that μ(P) ≥ α. Note that for each edge (u, v) ∈ M′ there is a path between u and v in Gi∗ . Since each edge of Gi∗ has length at most α, we conclude from the triangle inequality that $||uv||_\infty \leq n\alpha$. Thus d(u, v) ≤ 2nα and μ(M′) ≤ 2n2α.

Algorithm: Input: Point set P, D is the bounding box of P and S is any subset of the P and m= |S|/2.

In the algorithm, hierarchical subdivision is used by incorporation the idea of a randomly shifted quad-tree. A cell of the quad-tree is subdivided into subcells and compute a matching such that the number of edges that "cross" a subcell is the minimum number that needs to be in any matching. This is due to an imbalance between the number of red and blue points in the subcell. To ensure that the overall increase in cost is not too much we make sure that the number of levels in the quadtree is O(log(1/ε)). To do this we allow a cell of the quadtree to be partitioned into a large number of subcells. The size of the subproblems in the "merge" step may be quite large but we reduce this problem to a small-sized transportation problem.

Every node of the quad-tree is associated with a point set and bounding box D of the point set. The nodes are visited in a top down approach. The root node of this subdivision is associated with the input point set P and its bounding square. For a leaf, matching of the point set associated with that node is computed directly. For an internal node of the subdivision, the algorithm uses a randomly shifted grid of an appropriate size to break up D into a set of cells C.

C denotes the set of grid cells that intersect D. For each grid cell C ∈ C, S_C = S ∩ C while χ(C) = ||SC ∩ R| − |SC ∩ B||. If S_c has an imbalance in the number of red and blue points, a set

Q_C formed by arbitrarily picking χ(C) number of the red or blue points depending on which color points are higher in the S_c. Now, Q is defined as the union of Q_C over all grid cells C and it is obvious that Q has equal number of red and blue points. Each of the points in Q_C can be made equivalent to the same points when shifted to the center of the grid cell C.

A procedure called ***Match*** is used in which an optimal matching M of S is computed using Hungarian algorithm if m is smaller than some constant i.e node is a leaf. α is computed in O(mlogm) time and if $2m^5$ α is greater than 1/8 times the side-length of D, matching of S is computed by making a call to the subroutine ***Sub-Match*** with parameters S, D, α, m which itself is a recursive procedure. As |Q| is bounded by the number of edges of M that cross the grid lines, Probability of number of edges n of M that cross the grid lines being greater than 0 is at most $1/m^3$ because of the large grid size, and |Q| is bounded by n, the running time is $O(m^3)$ (for running the Hungarian algorithm) with probability at most $1/m^3$ and is O(m) otherwise.

When the above condition is not present, random shift of the grid G $2m^5$ α is taken. An optimal matching M' is computed for these moved points. For each cell for which point set $S_C - Q_C$ is nonempty, M_C of this point set is computed by calling the subroutine Sub-match. Each such node becomes a child of the current node. The resulting matching is $M' \cup (S_C \in C)\ M_C'$.

When the subroutine is called, either if L ≤ α/m2 (L is the side length of D) or if $|S|/2 \le n^{6\delta}$ where when δ = ε/12, perfect matching, perfect matching of S is computed and returned. The cost associated is $O(n^{1+12\delta})$ as for every $n_{i=}|S|$, the running times is $O(n_i^3)$. Otherwise, random shift of the grid G L/max{8,mδ} is taken and the following steps are similar to those in the match procedure. Similar to procedure match, the running time is O(m). The recursive subroutine when first invoked has a bounding box size of $16m^5$ α and falls by a factor of at least at least mδ at each level but not less than α/m2. The number of levels of recursion is therefore O(1/δ)

Overall running time= $(m\log m + m + m/\delta + n^{1+12\delta})$ and when δ = ε/12, it is approximately $O(n^{1+\varepsilon})$.

Improved Algorithm: Input point set P = RUB of 2n points and a parameter 0 < ε < 1

Ignoring the step where, if $|S|/2 \le n^{6\delta}$ where when δ = ε/12, perfect matching, perfect matching of S is computed in subroutine Sub-match, the algorithms runs in (mlogm+ m/δ) expected time and has 1/δ levels the sub-problem is reduced to at most $n^{6\delta}$. The only modification is to set δ = 1/12 instead of δ = ε/12. This algorithm runs in O(n log n) expected time, has a constant number of levels, and reduces the problem to sub-problems of size at most n. We do this till the size of the sub-problems is at most $n^{1/4}$ and further till the size of at most $n^{\varepsilon/2}$. This can be done by replacing step2 of the subroutine by: If $|S|/2 \le n\varepsilon/2$ we compute an optimal matching of S using the cubic algorithm and return this matching and if |S|/2 ≤ m1/2, return the matching of S computed by Match(S,D). After this point, the matching can be obtained using Hungarian algorithm.

Conclusion: The number of levels is O(log1/ε), and the analysis of cost gives an approximation of O(log 1/ε) times the optimal. The running time is $O(n \log n \log 1/\varepsilon + n^{1+\varepsilon})$ which is $O(n^{1+\varepsilon})$.

A Near-Linear Time ε-Approximation Algorithm for Geometric Bipartite Matching[4]

The graph G = (V,E) where V = A ∪ B and E ⊆ A × B is a weighted bipartite graph with |A|=|B|=n. Let d(a, b) be the cost of an edge (a, b) ∈ E. A matching M in G is a set of vertex-disjoint edges. The cost of M is w(M). M is a perfect matching if |M| = n. An optimal matching is a perfect matching with minimum cost. An ε-approximate matching is a perfect matching whose cost is within (1+ε) times the cost of an optimal matching. Optimal matching on weighted bipartite graphs with n vertices and m edges can be computed using the Hungarian algorithm in O(mn).

For any Lp-norm, an ε-approximate matching of A and B can be computed with high probability in O(npoly(log n, 1/ε)) time by the algorithm suggested in this paper which iteratively generates an approximate minimum-cost augmenting path under the distance function (quad-tree based) which approximate the Lp-norm. A transformation like {(A'$_1$,B'$_1$), . . . , (A'$_k$,B'$_k$)} decomposes with high probability, computing ε-approximate matching of A,B. For any Lp-norm, an ε-approximate matching of A and B can be computed with high probability in O(npoly(log$^{\Delta}$, 1/ε)) time as that the total length of the augmenting paths generated by the algorithm is O((n/ε) log n).

Algorithm:

A randomly shifted quad-tree Q is based on a distance function that ε-approximates the Lp-norm. For A,B ⊆ [$^{\Delta}$]^d, integers are randomly chosen from i_1 to i_d uniformly and G is randomly-shifted hypercube that contains both A and B. Let □ be the least common ancestor of a and b, they lie in two different children of □. From previous papers, $w(M^*) \leq (1+\varepsilon/2)w_{OPT}$ where w_{opt} be the cost of the optimal matching under any L_p norm.

For any matching M, the edges are partitioned or clustered into specific classes. A vertex is called free if it is incident on no edge, else it is called matched. Two edges (a_l, b_l),(a_m, b_m)∈M can belong to the same class if they have the same common ancestor in Q and a_l, a_m are in the same subcell of G[□] and b_l,b_m are also in the same subcell of G[□] while A_F and B_F are sets of free vertices. K_M is the resulting sets of M_1…M_r into equivalence classes. For an edge (a, b) ∈ A × B, if there is an i ≤ r such that (a, b) ∈ Ai × Bi, then we call (a, b) a local edge. All other edges are called non-local

Adjusted cost: $$\Phi_M(a,b) = \begin{cases} d(a,b) & \text{if } (a,b) \text{ is a local edge,} \\ d(a,b)+\theta & \text{otherwise.} \end{cases}$$

Based on the classes formed, some edges are called special and the cost of such an edge is given by d(a,b)+ θ .The value of theta should satisfy the given inequality for the algorithm to compute ε-approximate matching in O(npoly(log$^{\Delta}$, 1/ε)) time. The θ is guessed and the value which computes the perfect matching with the least cost is used to obtain the ε-approximate matching.

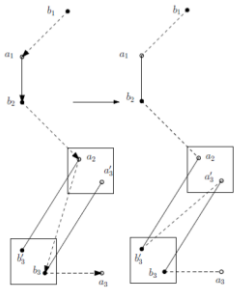
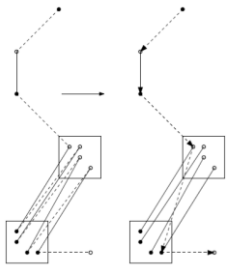
$$\varepsilon w(M^*)/6n \leq \theta \leq \varepsilon w(M^*)/3n$$

The algorithm employs a data structure that stores A, B and a matching M which is updated at every iteration after updating the augmenting path. There are methods FINDAP() which returns an augmenting path of the smallest net cost and another method AUGMENT() which updates M. The nodes are visited in bottom-up manner. An alternating path in a matching M is a simple path in which only the alternate edges are present in M while an augmenting path is an alternating path between two free vertices. M is augmented along the alternating path by changing M to $M \oplus \Pi$ i.e remove the edges present in the matching and replace them with the edges that are not in the matching. If all edges in an alternating path are local edges, they belong to the same class. An alternating path is called compact if the number of non-local edges is at least equal to a quarter of the length of the path. There is always an augmenting path of minimum net cost that is also a compact path. The algorithm maintains the following invariant, which we call the alternating cycle invariant.

The ***data structure*** employs a weighted directed graph $\vec{G}_M(A, B)$is considered to modify the problem of finding a compact augmenting path of minimum net cost into finding an “optimal" path between free vertices in this directed graph.
In the graph: Each free vertex of A is a sink and each free vertex of B is a source,
For a local edge (a,b), it is directed from a to b and $\mu_M(a, b) = -\Phi_M(a, b)$,
For a non-local edge (a,b), it is directed from b to a and $\mu_M(a, b) = \Phi_M(a, b)$
A path in the graph alternates between local and non-local edges but no local edge is allowed to be a matching edge.

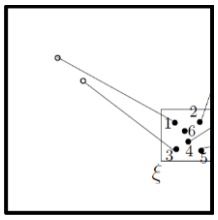
(i)A directed path in the graph from b to a can be transformed into a compact augmenting path from a to b such that $\mu(\vec{\Pi}) = \phi_M(\Pi)$.

(ii)An augmenting path can be transformed into directed path in the graph such that $\mu(\vec{\Pi}) \leq \phi_M(\Pi)$.

(i)  (ii) 

To efficiently compute a minimum cost directed path from the source to the sink, the vertices of the graph are clustered hierarchically. If two points p,q belong to the same cluster, then for any point r ∈ (A ∪ B), both edges (p,r) and (q,r) are either directed toward r or away from r. Let M = {(a_1,b_1),...,(a_k,b_k)} be the matching, and let K_M = {(A_1,B_1),...,(A_U,B_u)} be the classification of points in A and B induced by M.
For any ξ ∈ G[□], let Aξ = A∩ξ and Bξ = B ∩ξ, where $A_□$ = A∩□ , $B_□$= B ∩ □

- Free clusters: All free points of Aξ and Bξ belong to a single cluster ({6} is a free cluster)
  $A_\xi^F = A_F \cap \xi, \quad B_\xi^F = B_F \cap \xi.$
- Internal clusters: All points of Aξ (Bξ) whose partner point is also inside □belong to the same cluster ({1,3} is an internal cluster)
  $A_\xi^I = \{a_i \in A_\xi \mid (a_i, b_i) \in M, b_i \in B_\square\}$
- Boundary clusters: All points of Aξ whose partner points lie outside 2 are partitioned into various clusters.({2,4,5} are boundary clusters)



This is a subcell □

The cost of all edges in X × Y is the same in ~ G and all edges are oriented in the same direction- either all are oriented from B to A or all of them are oriented from A to B. Let $\overline{\pi}$ be a maximal connected sub-path of ~ Π that lies inside . Assume that it contains at least one edge. It is claimed that the entry point lies in an entry cluster while the exit points in exist cluster.

Indeed, if π is an initial portion of ~ Π, then the entry point is a source vertex, belonging to some entry cluster. If the first edge of π is a local edge or non-local edge, by construction, the entry point belongs to an internal cluster or boundary cluster respectively. Since all local edges are oriented from A to B, the entry point must belong to an entry cluster Similarly, all non-local edges are directed from B to A, the entry point must belong to some an entry cluster.

Information stored at each node: For a cell, let ~ G□ be the subgraph containing all directed paths of ~ G that lie completely within□ . For a pair of points a,b ∈ A2 ∪B2, ψ2(a,b) be the smallest mixed cost path from a to b in ~ G2, and we refer to such a path as an optimal path from a to b. Mixed cost ensures that an optimal path is always simple. At each cell in Q, for all X,Y ∈ X↓□ ×X↑□ , we store ψ □ (X,Y),i.e., the mixed cost of the optimal path between every entry cluster to every exit clusters. Similarly, the same proof applied for exit points.

Compressed graph: We show that if we have $\psi_\square(X,Y)$ at the four children of a cell, mixed path and optimal path can be computed for each such cell in poly(logΔ,1/ε) and O(kpoly(logΔ,1/ε)) time respectively where k is the length of the optimal path, such that $\psi_\square(a,b) = \psi_\square(X,Y)$ using procedures, ASCEND(2) and EXTRACTPATH(X,Y, 2). A weighted directed graph is constructed $H_\square = (V_\square, E_\square)$. For each pair of entry and exit clusters, an interior edge (X,Y) is added with $\psi_{\square^i}$ (X,Y) as its mixed cost. If there are edges between points of X and Y in ~ G, then the cost and direction of every edge in X ×Y is identical. Let this cost be β. We add an edge between X and Y with the direction identical to edges in X × Y and set its mixed cost to be (β,1).

Auxiliary procedures and Implementing FindAP and Augment Paths :

For any pair $X, Y \in \mathbb{X}^{\downarrow}_{\square} \times \mathbb{X}^{\uparrow}_{\square}$,

$$\min_{\substack{X' \in D(X) \\ Y' \in D(Y)}} \overline{\psi}_{\square}(X', Y') \leq \psi_{\square}(X, Y).$$

LEMMA 10. *For every node* $\square \in Q$ *and for any* $(X, Y) \in \mathbb{X}^{\downarrow}_{\square} \times \mathbb{X}^{\uparrow}_{\square}$,

$$\psi_{\square}(X, Y) = \min_{\substack{X' \in D(X) \\ Y' \in D(Y)}} \overline{\psi}_{\square}(X', Y').$$

EXTRACTPATH (X,Y,□)is implemented by starting with constructing the graph $H_{\square}$ and computing optimal paths for all pairs of vertices .Following steps include retrieving an augmenting path in the subgraph, where ψ(Π) = ψ(X',Y ') = ψ2(X,Y). The total time spent is O(kpoly(logΔ,1/ε)) where k is the number of edges in Π. As $H_{\square}$has no negative cycles and mixed costs have been minimized, the shortest minimum-cost path has been computed with Π being a simple path.

Implementing FINDAP and AUGMENT: When ∇ is the root of Q and S∇ ⊆ X↓ ∇ be the subset of free clusters at ∇, BF =SS∇ and AF =ST∇.. We compute an optimal path from a vertex b to a vertex a using EXTRACTPATH(S,T,∇) which can be converted into a compact augmenting path.

Hence, FINDAP takes O(|Π|poly(logΔ,1/ε)) time while AUGMENT(Π) updates M to M' = M ⨁Π and the information stored at Q. For a point p ∈ A ∪ B, let C(p) be the set of ancestors of the leaf of Q that contains p. Set C(Π) = Sp∈Π C(p). Hence, for a node 2 ∈ Q, ~ G2 does not change if 2 does not contain any vertex of Π. For each point p ∈ Π, we update the information at nodes of C(p) in a bottom-up manner. More precisely, suppose we have already updated ψ2 values for some cell 2 ∈ C(Π). Next, we call ASCEND(p(2)) and update the information at p(2). Since |C(Π)| = O(|Π|logΔ), the total time spent by AUGMENT(Π) is O(|Π|poly(logΔ,1/ε)).

ACI: For any alternating cycle C, $\phi_M(C) \geq 0$ which can be proved by induction.

Conclusion:

- Bounding the cost of M assuming ACI: M∗ being an optimal matching of A,B under d(·, ·) w(M) ≤ (1 + ε/3)w(M∗)
- Bounding the running time of the algorithm:
  $M_0 = \emptyset$, $M_{i+1} = M_i \oplus \Pi_i$ and $M_n = M$
  FINDAP and AUGMENT in step i take $O(|\Pi_i| \text{poly}(\log \Delta, 1/\epsilon))$ time
  Also, $\sum_{i=1}^{n} |\Pi_i| = O((n/\varepsilon) \log n)$

  This shows that the algorithm runs in $O(n \text{poly}(\log \Delta, 1/\varepsilon))$ time.

Fast Contour Matching
Using Approximate Earth Mover's Distance[5]

Weighted graph matching can be exploited as it is a good wat to align a pair of shapes represented by a set of descriptive local features. Similarity a shape to another shape can often be determined by finding the minimum cost of matching their features. This cost can be determined by measuring how dissimilar they are in terms of their spatial locations, appearance, curvature, or orientation.

There are quite a few shape matching algorithms that are even successful in finding minimum costs and matchings. However, they necessitate the computation of minimum cost correspondences between the sets of features of each shape. This computation has a super-polynomial time complexity based on the number of features, which gets even worse when looking up similar shapes in a large database. Optimized search and filtering methods are often used to reduce the size of the database wherever possible, but in the worst case it still reaches unacceptably high levels of complexity since it's the individual comparisons themselves that are slow.

To address this, Grauman and Darrell propose a new contour matching algorithm that makes use of approximation techniques. This consequently enables faster retrieval of similar shapes from a database, since each comparison now costs less time. The problem is treated like a graph matching problem, and the metric of similarity used is the Earth Mover's Distance, i.e, analogically, the movement of dirt it would take to transform one location to match another in the space. To use this for our problem of least cost matching, we can contextualize it as looking for the movement of dirt that requires the least amount of work.

For a metric space (X,D) and two n element sets A,B $\subset$ X, the distance is the minimum cost of a perfect matching between A and B:

$$EMD(A,B) = \min_{\pi:\mathbf{A}\to\mathbf{B}} \sum_{a\in\mathbf{A}} D(a,\pi(a))$$

Algorithm:

If the total weight of the two sets of points is equal, we can do a complete one-to-one correspondence to find the solution, which makes it similar to the bipartite graph matching problem. If the total weights of the two are unequal, however, a partial matching is done, and the distance is calculated by calculating the work required to move from the set with the higher weight to the one with the lower weight. We also make use of Locality-Sensitive Hashing to index a database of examples by using hash tables whose hash function ensures a high collision rate for similar examples, and a low one for dissimilar ones.

The approach to match contours to with approximate EMD is:

1. Extract features from a database of shape images, treating each image's features as a uniformly weighted point set.
2. For each input shape, produce a sparse vector by using the EMD embedding across the point sets.
3. Generate random LSH hash functions, and use it to enter all the embedded database vectors into the hash tables.
4. Now, given a new query shape, we compute the embedding for it using the same random functions we used to compute the embedding for the database.
5. Now we can simply match the embedding of the new query shape against just the shapes belonging to the hash buckets indexed by the same embedding.

The time required for this to run is found to be O(*nd* log(Δ))

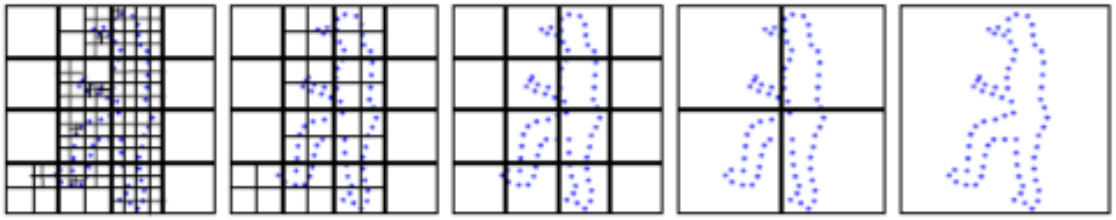


Figure above: Imposing a hierarchy of grids on a set of contour points to get its embedding. Embedding shape features residing in higher dimensions is an analogous process of imposing hyperplane grids on the space.

Algorithm 1 The procedure for embedding sets of local features

Given: N weighted point sets $\{\mathbf{A}_1, \ldots, \mathbf{A}_N\}$, where $\mathbf{A}_i = \{(\mathbf{f}_1, w_1) \ldots, (\mathbf{f}_{m_i}, w_{m_i})\}$ is a weighted point set composed of m_i d-dimensional features $\mathbf{F}^i$ with scalar weights $\mathbf{W}^i = [w_1, \ldots, w_{m_i}]$, where $t(\mathbf{A}_i) = \sum_{j=1}^{m_i} w_j$, and Δ is the diameter of $\cup_{i=1}^{N} \mathbf{F}^i$.

Let $t_{max} = \max\limits_{i} t(\mathbf{A}_i)$
for all $i = 1, \ldots, N$ **do**
  if $t(\mathbf{A}_i) < t_{max}$ **then**
    $\mathbf{A}_i \leftarrow \{(\mathbf{f}_1, w_1), \ldots, (\mathbf{f}_{m_i}, w_{m_i}), (\mathbf{d}_1, u), \ldots, (\mathbf{d}_q, u)\}$, where u is unit weight, $q = t_{max} - t(\mathbf{A}_i)$, and $\mathbf{d}_z$ is a random selection from $\{\mathbf{f}_1, \ldots, \mathbf{f}_{m_i}\}$.
  end if
end for
Let $L = \lceil \log \Delta / \log 2 \rceil + 1$.
For $1 \le l \le L$, let each $\mathbf{s}^l = [s_1^l, \ldots, s_d^l]$ be a random vector from $[0, 2^l]^d$.
for all $\mathbf{A}_i, 1 \le i \le N$ **do**
  for all $(\mathbf{f}_j = [f_1^j, \ldots, f_d^j], w_j) \in \mathbf{A}_i$ **do**
    for all $\mathbf{s}^l, 1 \le l \le L$ **do**
      $\mathbf{x}_l^j = [c(s_1^l, f_1^j), \ldots, c(s_d^l, f_d^j)]$, where $c(s_k^l, f) = trunc((f - s_k^l)/2^l)$
      $\mathbf{v}_l^j = w_j \times 2^l$
    end for
    $\mathbf{p}_j^i = [(\mathbf{x}_1^j, v_1^j), \ldots, (\mathbf{x}_L^j, v_L^j)]$
  end for
  $embed(\mathbf{A}_i) = tally(sort([\mathbf{p}_1^i, \ldots, \mathbf{p}_{m_i}^i]))$, where pairs $(\mathbf{x}, v)$ represent sparse vector entries with index $\mathbf{x}$ and value v, $sort()$ returns $[(\mathbf{x}_{s_1}, v_{s_1}), \ldots, (\mathbf{x}_{s_n}, v_{s_n})]$ such that $\mathbf{x}_{s_t} \le_{LEX} \mathbf{x}_{s_{t+1}}$ (lexicographic ordering of concatenated vector elements) and $tally()$ sums values of sparse vector entries with equal indices.
end for

There are however, some drawbacks to using this. While simple to implement, it requires preprocessing and advance knowledge of shapes. Thus, Grauman and Darrell attempted to use richer shape descriptors to gain more robust matchings. The shape context feature at a single contour point is a log-polar histogram of the coordinates of the rest of the point set, measured using the reference point as the origin, which makes it inherently translation invariant. While this method allows matching with the full shape descriptor, it is still preferable to use a low-dimension feature descriptor as any constant change in point dimensions changes the constant distortion factor *C* in the embedding. A low dimensional feature subspace is found by sampling a large number of features from the database. It was found that using a low dimensional subspace was still able to capture much of the local contour variation in the database.

Conclusion:

Thus, Grauman and Darrell have presented a new contour matching algorithm for shape matching that uses approximation to EMD to judge the similarity between sets of local shape descriptors. It allows much faster shape-based similarity retrieval from a database, and its runtime is only linearly dependent on the number of feature points

CONCLUSION:

The Earth Mover's Distance (EMD) is a dissimilarity measure between signatures used for image retrieval in different feature spaces. It is a transportation problem where the minimal cost to transform one distribution to another is calculated. This problem can be dealt as the un-capacitated minimum cost network flow problem and it can be solved using bipartite graphs in $O(N^3 \log n)$. The EMD can be applied to variable-length representations of distributions(signatures) and allows partial matches between images. The correspondences between bins of two histograms has led us towards a definition of the distances between sets of features. This implies that distance measures based on bipartite graph matching can be defined as the minimum cost of matching elements between two sets of elements.

Euclidean Bichromatic Matching, EMD (A,B) is defined to be the minimum cost of a perfect matching with edges between A and B. For this, a constant factor approximation algorithm with running time NlogO(1) N is proposed. After decomposing the matching is several subsets, random distribution is employed for the sampling which proved to be a very fast and efficient way of estimating the cost of the matching.

In the Euclidean bipartite matching problem, two sets R and B where $|R| = |B| = n$ are mapped such that no two edges share an endpoint and the sum of distances between the paired points is minimized. An near-linear time constant factor approximation algorithm that uses a randomly shifted quadtree is given such that for any parameter $0 < \varepsilon < 1$, the algorithm runs in $O(n1+\varepsilon)$ expected time and returns a matching whose expected cost is within a multiplicative factor $O(\log(1/\varepsilon))$ of the optimal. The dimension d is considered to be a fixed constant.

Given two point sets $A,B \subset Rd$, $|A| = |B| = n$, and for a parameter $\varepsilon > 0$, a near-linear time ε-approximation algorithm is that computes an ε-approximate perfect matching of A and B is presented [4]. The algorithm runs in in $O(n\text{poly}(\log n, 1/\varepsilon))$ time with high probability. A distance function which approximates Lp norm and a randomly shifted quad-tree together are used for the algorithm which iteratively generates an approximate minimum-cost augmenting path in time proportional to the length of the path. The total length of the augmenting paths generated by the algorithm is $O((n/\varepsilon) \log n)$, ensuring that the algorithm runs in $O(n\text{poly}(\log n, 1/\varepsilon))$ time.

As an application of EMD, a fast contour matching algorithm is proposed that was quickly able to compute the minimum weight matching between sets of descriptive local features which used approximation techniques and low-distortion embedding of the Earth Mover's Distance (EMD) into a normed space. It allowed much faster shape-based similarity retrieval from a database, and was shown to be a much faster algorithm since its runtime was only linearly dependent on the number of feature points present.

References:

[1] Yossi Rubner, Carlo Tomasi, Leonidas J. Guibas, The Earth Mover's Distance as a Metric for Image Retrieval, International Journal of Computer Vision, v.40 n.2, pages 99-121, Nov. 2000.

[2] Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In Proc. of the 18th Ann. ACM-SIAM Symp. on Discrete Algo., pages 39–42, 2007.

[3] P. K. Agarwal and K. R. Varadarajan, A near-linear constant-factor approximation for Euclidean bipartite matching, In Proc. of 12th Annual Sympos. on Comput. Geom., pages 247–252, 2004.

[4] R. Sharathkumar, Pankaj K. Agarwal, A near-linear time ε-approximation algorithm for geometric bipartite matching, Proceedings of the forty-fourth annual ACM symposium on Theory of computing, May 19-22, 2012.

[5] Kristen Grauman, Trevor Darrell, Fast contour matching using approximate earth mover's distance, In Proc. Of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.