

Measuring the Similarity of Geometric Graphs*

Otfried Cheong[†] Joachim Gudmundsson[‡] Hyo-Sil Kim[†] Daria Schymura[§] Fabian Stehn[§]

Abstract

What does it mean for two geometric graphs to be similar? We propose a distance for geometric graphs that we show to be a metric, and that can be computed by solving an integer linear program. We also present experiments using a heuristic distance function.

1 Introduction

Computational geometry has studied the matching and analysis of geometric shapes from a theoretical perspective, and developed efficient algorithms measuring the *similarity* of geometric objects. Two objects are similar if they do not differ much geometrically. A survey by Alt and Guibas [1] describes the significant body of results obtained by researchers in computational geometry in this area.

This paradigm fits a number of practical shape matching problems quite well, such as the recognition of symmetries in molecules, or the self-alignment of a satellite based on star patterns. Other pattern recognition problems, however, seem to require a different definition of “matching.” For instance, recognizing logos, Egyptian hieroglyphics, Chinese characters, or electronic components in a circuit diagram are typical examples where this is the case. The same “pattern” can appear in a variety of shapes that differ geometrically. What remains invariant, however, is the “combinatorial” structure of the pattern.

We propose to consider such patterns as geometric graphs, that is, planar graphs embedded into the plane with straight edges. Two geometric graphs can be considered similar if both the underlying graph and the geometry of the planar embedding are “similar.” The distance measures considered in computational geometry, such as the Hausdorff distance, Fréchet distance, or the symmetric difference, do not seem to apply to geometric graphs.

Pattern recognition systems that combine a combinatorial component with a geometric component are already used in practice—in fact, *syntactic* or *structural* pattern recognition is based on exactly this idea:

A syntactic recognizer decomposes the pattern into geometric primitives and makes conclusions based on the appearance and relative position of these primitives [2, 6]. While attractive from a theoretical point of view, syntactic recognizers have not been able to compete with numerical or AI techniques for character recognition [5]. In general, the pattern recognition community may be said to consider graph representations as expressive, but too time-consuming, as subgraph isomorphism in general is intractable.

An established measure of similarity between (labeled) graphs is the *edit distance*. The idea of an edit distance is very intuitive: To measure the difference between two objects, measure how much one object has to be changed to be transformed into the other object. To define an edit distance, one therefore defines a set of allowed operations, each associated with a cost. An edit sequence from object A to object B is a finite sequence of allowed operations that transforms A into B . The distance between A and B is the minimal cost of an edit sequence from A to B .

The edit distance originally stems from string matching where the allowed operations are insertion, deletion and substitution of characters. The edit distance of strings can be computed efficiently, and the string edit distance is used widely, for instance in computational biology.

Justice and Hero [4] give a definition of an edit distance for vertex-labeled graphs that additionally allows relabeling of vertices, and give an integer linear programming formulation. The edit operations are insertion and deletion of vertices, insertion and deletion of edges, and a change of a vertex label.

It is natural to try to define an edit distance for geometric graphs as well. Simply considering a geometric graph as a graph whose vertices are labeled with their coordinates is not sufficient, as the cost of inserting and deleting an edge should also be dependent on the length of the edge. This leads to the following operations: Insertions and deletions of vertices, translations of vertices, and insertions and deletions of edges. However, it is difficult to give bounds on the length of an edit sequence: vertices can move several times to make insertions and deletions cheaper.

This leads us to define another graph distance function, given in the next section. It is not an edit distance, and so we need to prove explicitly that it is a metric. We also give an integer linear programming formulation that allows us to compute our distance for

*This work was supported by Korea Science & Engineering Foundation through the Joint Research Program (F01-2006-000-10257-0).

[†]KAIST, Korea. {otfried,hyosil}@tclab.kaist.ac.kr

[‡]NICTA, Australia. joachim.gudmundsson@nicta.com.au

[§]FU Berlin, Germany. {schymura,stehn}@inf.fu-berlin.de

small graphs with an ILP solver. Unfortunately, we do not know how to compute or even approximate our graph distance for larger graphs. In fact, we give two reductions from NP-hard problems, but both result in non-“practical” instances of the problem.

We therefore turn our attention to a heuristic. We define the *landmark distance* of two geometric graphs, and present pattern retrieval experiments on a database of 25056 graphs created from glyphs of Chinese characters. The idea of the landmark distance is to represent a geometric graph on n vertices as a set of n points in \mathbb{R}^6 . The landmark distance between two geometric graphs is then the Earth Mover’s Distance between the point sets representing the graphs.

2 Geometric graph distance

Our distance is inspired by the graph edit distance: it is based on primitive operations, namely insertion and deletion of vertices and edges, and the translation of vertices. However, we do not allow arbitrary sequences of these operations. Instead the edit operations must be performed in this order:

1. Edge deletions
2. Vertex deletions
3. Vertex translations
4. Vertex insertions
5. Edge insertions

Vertex insertions and deletions are free, insertion or deletion of an edge e of length $\ell(e)$ have cost $C_E \cdot \ell(e)$, and translating a vertex has cost C_V times the distance of the translation plus C_E times the change in the length of the incident edges. Note that we measure the change in edge length from graph A to graph B , and not for the individual operations.

Taking into account the change in edge length is necessary, as otherwise the distance would not satisfy the triangle inequality, see Figure 1.

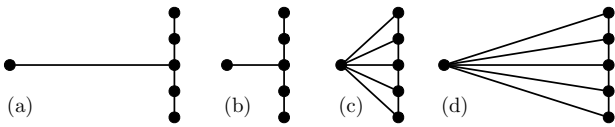


Figure 1: If the change in edge lengths are not taken into consideration then it is cheaper to go from graph (a) to graph (d) via graphs (b) and (c).

We can now show:

Theorem 1 *The geometric graph distance defined above is a metric on the set of geometric graphs without isolated vertices.*

The geometric graph distance can be formulated as an ILP as follows. Let $A = (V(A), E(A))$ and $B = (V(B), E(B))$ be the two geometric graphs. For each vertex v_i of $V(A)$ and v_j in $V(B)$, we have a

binary variable V_{ij} which is 1 if v_i is moved to v_j , otherwise 0. Similarly, for each edge e_i of $E(A)$ and e_j in $E(B)$, we have a binary variable E_{ij} which is 1 if the two endpoints of e_i are moved to the endpoints of e_j , otherwise 0. Furthermore, each vertex can only be moved once, hence, $\forall i \sum_j V_{ij} \leq 1$ and $\forall j \sum_i V_{ij} \leq 1$.

An edge can only be matched if the endpoints are the same, i.e., if $e_i = (v_a, v_b)$ and $e_j = (v_p, v_q)$ then $E_{ij} \leq \frac{1}{2} \cdot (V_{ap} + V_{bq} + V_{aq} + V_{bp})$.

These are all the constraints. The distance is the minimum of the function:

$$C_V \cdot \sum_{i,j} (|v_i v_j| \cdot V_{ij}) + C_E \cdot \left(\sum_i \ell(e_i) + \sum_j \ell(e_j) \right) - C_E \cdot \sum_{i,j} \left(\ell(e_i) + \ell(e_j) - |\ell(e_i) - \ell(e_j)| \right).$$

The first term is the cost of moving all vertices (the remaining vertices are deleted and inserted, which is for free). The second and third terms are the total cost of first deleting all edges of A and then inserting all edges of B . The only way to avoid deleting and inserting an edge is by moving it from A to B . In that case, E_{ij} is 1, and the cost is the difference in edge length, which is modeled by the fourth term.

Theorem 2 *The problem of computing the geometric graph distance as defined above is NP-hard.*

The theorem can be proven by using a reduction from the 3D-matching problem or the Hamiltonian path problem. However, we can only prove the theorem in the case when the two graphs are highly non-planar or for planar graphs when $C_V \ll C_E$. Thus, both results are for non-“practical” instances of the problem

3 Landmark Distance

The idea of the landmark distance is to designate a few vertices as *landmarks* and to represent the vertices of the graph by their distances to the landmarks.

Formally, let $G = (V, E)$ be a geometric graph, and let $C(G)$ be the complete graph on the set V . An edge (u, v) of $C(G)$ is given a weight as follows: if $(u, v) \in E$, then its weight is $|uv|$, otherwise its weight is $p \cdot |uv|$, where $p > 1$ is a fixed *penalty* value. The distance $d_G(u, w)$ between a vertex $u \in V$ and a landmark $w \in V$ is then defined as the length of the shortest path between u and w in $C(G)$. After testing different values for p we chose $p = 1.6$ for our experiments.

Let $L = w_1, \dots, w_k$ be k vertices of G called *landmarks*. For a vertex $v \in V$ with coordinates (x, y) , we define the *L-vector* L_v of v as the $(k+2)$ -dimensional vector containing the distances of v to the k landmarks as well as the coordinates of v :

$$L_v = (d_G(v, w_1), \dots, d_G(v, w_k), x, y).$$

The *landmark representation* $R(G)$ of a graph G is now simply the set of L -vectors of all vertices: $R(G) := \{L_v \mid v \in V(G)\}$.

We define the *landmark distance* $d_L(G_0, G_1)$ between two geometric graphs G_0 and G_1 (both with

given landmarks) as the *normalized Earth Mover’s Distance* between the point sets $R(G_0)$ and $R(G_1)$.

The normalized Earth Mover’s Distance (nEMD) is a distance measure defined on weighted point sets [7]. Let P and Q be two weighted point sets with $\sum_{p \in P} w(p) = \sum_{q \in Q} w(q) = 1$, where $w(u) \geq 0$ is the weight of a point u .

Intuitively, the set P can be seen as a set of earth piles and the set Q can be seen as holes in the ground, where the weight of each point stands for the amount of earth at that position or the size of the hole, respectively. The Earth Mover’s Distance is then defined as the *cheapest* way to move the earth into the holes, where piles can be split and the cost of transporting s units of earth from a pile to a hole is equal to s times the distance between the pile and the hole.

The nEMD can be formalized by the following linear program. Let d_{pq} denote the Euclidean distance of $p \in P$ to $q \in Q$ and let f_{pq} denote the flow from p to q . Then the nEMD(P, Q) is defined as the minimal total cost that establish a maximum flow of 1:

$$\text{nEMD}(P, Q) = \min \sum_{p \in P} \sum_{q \in Q} f_{pq} d_{pq}$$

subject to the following constraints

$$\begin{aligned} \forall p \in P \quad \forall q \in Q \quad f_{pq} &\geq 0 \\ \forall p \in P \quad \sum_{q \in Q} f_{pq} &= w(p) \\ \forall q \in Q \quad \sum_{p \in P} f_{pq} &= w(q) \end{aligned}$$

In the definition of the landmark distance, we give all vertices equal weight and use the ℓ_1 -metric in \mathbb{R}^{k+2} as the underlying distance for the nEMD.

Theorem 3 *The landmark distance on graphs with given landmarks has the following properties:*

- (i) $d_L(G_0, G_1) \geq 0$,
- (ii) $d_L(G_0, G_1) = d_L(G_1, G_0)$,
- (iii) $d_L(G_0, G_2) \leq d_L(G_0, G_1) + d_L(G_1, G_2)$.

All the properties in the theorem follow directly from the fact that the normalized EMD is a metric [7]. Note that $d_L(G_0, G_1) = 0$ does not imply $G_0 = G_1$.

4 Experimental results

We performed pattern retrieval experiments on a database of graphs generated from Chinese character glyphs using the landmark distance. The motivation here was not to build a Chinese character recognition system—a lot of research has been done in this area, and it is not our intention to compete with these finely tuned results of years of research. We turned to Chinese characters because we found them to be a source of large number of graphs with known semantics.

We selected six different fonts, including two Korean, two Japanese, and two Chinese fonts. We picked

a set of 4176 Chinese characters (or, more precisely, Unicode code points) that exist in all six fonts, and generated graphs for each of these $6 \times 4176 = 25056$ glyphs as follows: We draw the glyph and compute its medial axis. We prune away small features, and then simplify each chain of degree-two vertices using the Imai-Iri algorithm [3]. Figure 2 shows three examples of glyphs and the corresponding graphs. For the distance computations, all graphs were then linearly scaled to fill a unit square (not shown in the figure).

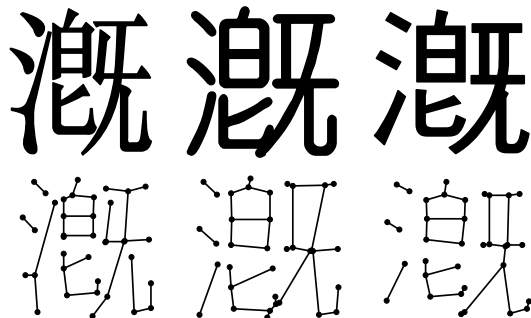


Figure 2: Three glyphs and generated graphs for U+6f11 “slowly flowing water”.

As explained, our database consists of 4176 sets of six graphs that represent the same abstract Chinese character. In principle, the graphs representing the same character should be similar, and we have assumed this as the ground truth of our database. In reality, there can be considerable variation between graphs with the same semantics, due to a different glyph style, or actual variations in character shape.

The experiment We selected one of the six fonts (the Korean “dotum” font) as our reference font, and built a database of 4176 “model” graphs from this.

We then considered each of the remaining 20880 “pattern” graphs, and computed its distance to each model, using the EMD implementation by Rubner et al. [7]. The models were then sorted in order of distance from the pattern. Ideally, the nearest model should be a graph for the same Chinese character, so we determined the index of the occurrence of the same Chinese character in the ranked list of models.

As a control experiment, we first tried this experiment using the Hausdorff distance of the graph vertices. The results are given in the first line of Table 1: For 31.8% of the 20880 patterns the best (most similar) model belonged to the same Chinese character, for 50.9% of the patterns, one of the first (most similar) ten models belongs to the same character.

It is clear that the Hausdorff distance does not capture the problem well enough (even though we ignore edge information here, we do not believe that using the edge information would actually help).

The Earth Mover’s Distance, however, does much

Graph distance		Index in ranked model list							
		1	2	3	4	5	6–10	11–20	21–200
Hausdorff distance of vertices	# patterns	6647	1272	705	496	387	1115	1257	4283
	accum %	31.8	37.9	41.3	43.7	45.5	50.9	56.9	77.4
EMD of vertices	# patterns	16844	1646	519	258	184	286	193	200
	accum %	80.7	88.6	91.0	92.3	93.2	94.5	95.4	96.4
Landmark distance	# patterns	17814	1298	454	260	152	317	195	221
	accum %	85.3	91.5	93.7	95.0	95.7	97.2	98.1	99.2

Table 1: A summary of our experimental studies of Chinese character retrieval.

better, even when we ignore all information about the edges of the graph. The second line of our table shows this experiment, where we ranked the models by nEMD of the graph vertices only. The results are surprisingly good considering that the edge information is not used at all: for 94.5% of the patterns, the correct Chinese character is found in the top ten.

Landmark selection Selecting the right landmarks in each graph is critical to the success of the landmark distance. We fixed four landmarks in each model graph, by choosing the four vertices that are extreme in the four diagonal directions. For a pattern, we actually try all plausible choices of landmarks, and use the landmarks that result in the smallest distance to a given model (so the choice of landmarks could be different for each model).

Speeding up the computation It turned out that computing the EMD is rather expensive, and computing 4176×20880 EMD distances in every experiment is very time-consuming.

We therefore used a heuristic to speed up the computation: Instead of using the EMD, we compute a simplified landmark distance $d'_L(G_0, G_1)$ which is defined as follows: For each point u in $R(G_0)$, find the nearest point $nn(u) \in R(G_1)$. Then

$$d'_L(G_0, G_1) := \sum_{u \in R(G_0)} \|u - nn(u)\|_1.$$

We first rank all models using this simplified landmark distance. We then look at the nearest 200 models, and recompute the distance from the pattern to these 200 models using the landmark distance.

The heuristic greatly speeds up the computation while having little effect on the quality of the recognition. As an example of the speed-up we compared one character (6f01s) against the entire database which required 31 seconds using the EMD while the above heuristic only required 1.8 seconds, which is a speed-up of approximately 17. However, it should be noted that the exact timings depend very much on the character.

For 85.3% of the patterns this approach finds the same Chinese character and in 97.2% of the cases it

is in the top ten.

5 Conclusions and open problems

We believe that we have only scratched the surface of this problem. We gave some evidence that our geometric graph distance is hard to compute, but we lack a formal proof that it is NP-hard for planar graphs with realistic values of C_E and C_V .

Is there a PTAS for our distance, or at least a constant-factor approximation?

If we do not want insertions and deletions of vertices to be free, can we incorporate that into our distance?

Finally, a major problem of our metric is that it does not allow us to cheaply “bend” an edge, that is, to insert a degree-two vertex into an edge and then to move that vertex slightly. Can we define a metric that allows this while still being computable at least through integer linear programming?

References

- [1] H. Alt and L.J. Guibas. *Discrete Geometric Shapes: Matching, Interpolation, and Approximation*, pages 121–153. Elsevier B.V., 2000.
- [2] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [3] H. Imai and M. Iri. Polygonal approximations of a curve - formulations and algorithms. In *Computational Morphology*, pages 71–86. Elsevier B.V., 1988.
- [4] D. Justice and A. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214, 2006.
- [5] S. Lucas, E. Vidal, A. Amiri, S. Hanlon, and J. C. Amengual. A comparison of syntactic and statistical techniques for off-line OCR. In *2nd International Colloquium Grammatical Inference and Applications*, pages 168–179. Springer-Verlag, 1994.
- [6] T. Pavlidis. *Structural Pattern Recognition*. Springer-Verlag, New York, 1977.
- [7] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth movers distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), 2000. <http://robotics.stanford.edu/~rubner>.