Measuring the Similarity of Geometric Graphs

Otfried Cheong[1⋆], Joachim Gudmundsson[2], Hyo-Sil Kim[1∗], Daria Schymura[3⋆⋆], and Fabian Stehn[3∗∗]

[1] KAIST, Korea. {otfried,hyosil}@tclab.kaist.ac.kr
[2] NICTA,[⋆⋆⋆] Australia. joachim.gudmundsson@nicta.com.au
[3] FU Berlin, Germany. {schymura,stehn}@inf.fu-berlin.de

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

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

⋆⋆ This research was partially funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures", the DFG-projects AL253/6-1, KN591/2-2 and through the Joint Research Project 446 KOR 113/211/0-1.

⋆⋆⋆ NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

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, 7]. While attractive from a theoretical point of view, syntactic recognizers have not been able to compete with numerical or AI techniques for character recognition [6]. 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 known to be 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 [5] defined an edit distance for vertex-labeled graphs that additionally allows relabeling of vertices, and give an integer linear programming formulation of the edit distance. 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. We give some examples in the following section.

This leads us to define another graph distance function in Section 3. 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 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

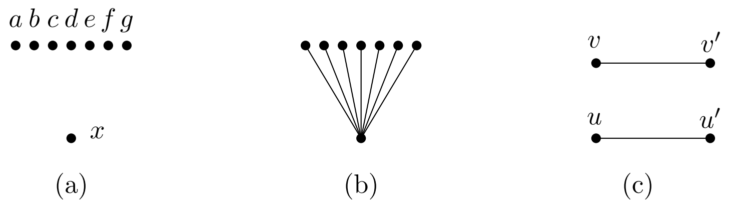

Fig. 1. Bad examples for edit distances.

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 Why not an edit distance?

A graph edit distance for geometric graphs needs to support at least the following primitive operations: insertion and deletion of vertices and edges, and translation of vertices.

Throughout this paper, we assume that geometric graphs are given with absolute coordinates in the plane. In other words, a translated, rotated, or scaled copy of a graph drawing is not necessarily similar to the original graph drawing. If a similarity measure that is invariant under some motions is needed, this can always be achieved by minimizing over all motions of interest.

Let us first assume that insertions and deletions of edges have cost identical to the length of the edge, and the cost of a vertex translation is the distance by which the vertex was translated. This leads to very artificial edit sequences, where vertices "hop around" several times. For instance, in the example shown in Figure 1, the cheapest edit sequence transforming the graph (a) into graph (b) (where both graphs are meant to be at the same location in the plane) is to translate x first close to a, to insert edge xa, then to translate x close to b, to insert edge xb and so on. The costs are roughly $d(x, a) + d(a, g) + d(x, g)$, which is much less than the sum of the lengths of the edges. In addition there is actually no optimal edit sequence—to reach the optimum, one would have to insert a vertex *on top* of an existing vertex—so the graph distance would have to be defined as the infimum over all edit sequences.

To fix this problem, we can change the cost of a vertex translation to account for the change in length of all the incident edges. Unfortunately, there is then no bound on the length of an optimal edit sequence, and we again have to define the graph distance as the infimum over all edit sequences. For instance, in the simple example of Figure 1 (c), where we want to measure the distance between the graph consisting of the single edge uu' and the graph consisting of the single edge vv', an optimal edit sequence would be to move both vertices alternatingly by an infinitesimal amount so as to minimize the change in edge length incurred.

3 Geometric graph distance

Our distance is inspired by the graph edit distance in that it is based on the primitive operations above. However, we do not allow arbitrary sequences of the 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

Only isolated vertices can be inserted or deleted, and this operation is free. Insertion or deletion of an edge e of length $|e|$ has cost $C_e|e|$. Translating a vertex has cost C_v times the distance of the translation plus, for each incident edge, C_e times the *change in the length* of the edge.

Note that *we measure the change in edge length between the two graphs*, and not for individual operations!

The ordering of the five types of operations is really the only ordering suitable for this definition: It has to be symmetric and so vertex translations have to appear in the middle; since only isolated vertices can be deleted, edges have to be deleted before vertices; and allowing deletions after insertions is never useful.

A different way of looking at the distance is the following: Let $A = (V_A, E_A)$ and $B = (V_B, E_B)$ be the two graphs. We chose a subset $V^* \subseteq V_A$ and an injection $\sigma : V^* \to V_B$, and associate with them the following cost:

(i) we pay $C_e|uv|$ for any edge $(u, v) \in E_A$ such that not both $u, v \in V^*$ or $\sigma u \sigma v \notin E_B$;
(ii) we pay $C_e|uv|$ for any edge $(u, v) \in E_B$ such that not both $u, v \in \sigma V^*$ or $\sigma^{-1}u\sigma^{-1}v \notin E_A$;
(iii) we pay $C_e\big||uv| - |\sigma u \sigma v|\big|$ for each edge $(u, v) \in E_A$ with both $u, v \in V^*$ and $\sigma u \sigma v \in E_B$;
(iv) we pay $C_v|u\sigma u|$ for each $u \in V^*$.

The geometric graph distance $ggd(A, B)$ is the minimum of this cost over all choices of V^* and σ.

The reader may wonder if it is necessary to include the change of edge length, that is, the term (iii). Indeed, without this term, the distance would not satisfy the triangle inequality. An example of this is shown in Figure 2, where the distance between the graphs in (a) and (d) would be larger than the sum of the distances between the graphs in (a) and (b), (b) and (c), and (c) and (d).

We can now show:

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

Proof. Let A, B and C be geometric graphs without isolated vertices. It follows from the definition that $ggd(A, B) \geq 0$ and that $ggd(A, B) = ggd(B, A)$. We

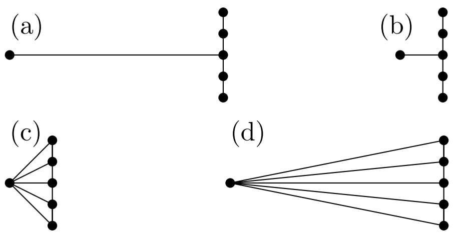

Fig. 2. The triangle inequality does not hold without accounting for the change in edge length.

clearly have $ggd(A, A) = 0$. Assume now that $ggd(A, B) = 0$. Then we must have $V^* = V_A$ and $\sigma V^* = V_B$, and for each $u \in V_A$ we must have $u = \sigma u$, and so $A = B$.

It remains to show the triangle inequality. Let $V_1^* \subset V_A$ and $\sigma_1 : V_1^* \to V_B$ be as in the definition of $ggd(A, B)$, and let $V_2^* \subset V_B$ and $\sigma_2 : V_2^* \to V_C$ be as in the definition of $ggd(B, C)$.

Let $V^* := \sigma_1^{-1}(\sigma_1 V_1^* \cap V_2^*)$, and let $\sigma : V^* \to V_C$ be defined as $\sigma u = \sigma_2 \sigma_1 u$. We evaluate the cost of V^* and σ. The cost of terms (iii) and (iv) is bounded by the sum of the corresponding terms in $ggd(A, B)$ and $ggd(B, C)$. Edges $(u, v) \in E_A$ such that not both of $u, v \in V_1^*$ are accounted for in $ggd(A, B)$, and edges $(u, v) \in E_C$ such that not both of $u, v \in \sigma_2 V_2^*$ are accounted for in $ggd(B, C)$.

For an edge $(u, v) \in E_A$ with $u \in V_1^* \backslash V^*$ and $v \in V_1^*$, we have that $\sigma_1 u \notin V_2^*$. It follows that $(\sigma_1 u, \sigma_1 v) \in E_B$ falls into case (i) for $ggd(B, C)$, and so the cost of deleting this edge is bounded by the change in edge length in $ggd(A, B)$ and the deletion cost in $ggd(B, C)$.

A symmetric argument holds for edges $(u, v) \in E_C$ with $u \in \sigma_2 V_2^* \setminus \sigma V^*$ and $v \in \sigma_2 V_2^*$. □

The geometric graph distance can be formulated as an ILP as follows: For each pair of vertices $u \in V_A$ and $v \in V_B$, we introduce a binary variable V_{uv}. We will have $V_{uv} = 1$ if $u \in V^*$ and $\sigma u = v$, and 0 otherwise. Similarly, for each pair of edges $e \in E_A$ and $e' \in E_B$, we introduce a binary variable $E_{ee'}$. This variable will be 1 if and only if both endpoints u, v of e are in V^*, and $\sigma u, \sigma v$ are the endpoints of e'.

The constraints are as follows: For each $u \in V_A$, we have

$$\sum_{v \in V_B} V_{uv} \leq 1,$$

and for each $v \in V_B$ we have

$$\sum_{u \in V_A} V_{uv} \leq 1$$

Together, these constraints ensure that σ is an injective function. Furthermore, for each variable $E_{ee'}$ we introduce the following constraint, where $e = (u, v)$

and $e' = (u', v')$:

$$E_{ee'} \leq \frac{1}{2} \cdot (V_{uu'} + V_{vv'} + V_{uv'} + V_{vu'})$$

These are all the constraints. Then $ggd(A, B)$ is the minimum of the function:

$$C_v \sum_{u,v} |uv| \cdot V_{uv} \; + \; C_e \sum_{e \in E_A} |e| \; + \; C_e \sum_{e' \in E_B} |e'|$$
$$-C_e \sum_{e,e'} \Big(|e| + |e'| - \big| |e| - |e'| \big| \Big) \cdot E_{ee'}.$$

The first term is the cost of translating vertices (the remaining vertices are deleted and inserted, which is free). The second and third terms are the total cost of deleting all edges of A, and 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, $E_{ee'} = 1$, and the cost is the difference in edge length, which is modeled by the fourth term, that is, we subtract $|e|$ and $|e'|$ and add $\big||e| - |e'|\big|$ instead.

We were able to compute distances between some small graphs with less than ten vertices using the ILP solver of the Gnu linear programming toolkit (GLPK). Unfortunately, we do not know how to compute the distance for larger graphs, and indeed this is a hard problem.

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

We give two proofs. However, the first proof assumes that graphs can be non-planar, the second proof assumes that we can choose $C_v \ll C_e$. Thus, both results are for non-“practical” instances of the problem.

Proof (assuming graphs can be non-planar). We reduce 3DMatching to the problem. Remember that the input for 3DMatching consists of three disjoint copies X, Y, Z of $\{1, \ldots, n\}$, and a set T of m triples from $X \times Y \times Z$. The problem is to determine whether there is a subset S of T of exactly n triples that cover X, Y, Z completely.

We can reduce 3DMatching to graph matching as follows: Pick four points t_0, x_0, y_0, z_0 in the plane as the corners of a unit square. Pick n points $x_1, x_2, \ldots, x_n$ very close to x_0, and do the same for $y_1, \ldots, y_n$ and $z_1, \ldots, z_n$. Finally, pick m points $t_1, \ldots, t_m$ very close to t_0.

The graph G_0 consists of $4n$ vertices and edges, forming the n disjoint loops $t_i x_i y_i z_i t_i$.

The graph G_1 consists of $3n+m$ vertices as follows: Let triple i in T be (j, k, l). Then G_1 includes the loop $t_i x_j y_k z_l t_i$. Let $M \leq 4m$ be the number of edges of G_1.

Clearly we have to insert $M - 4n$ edges to go from G_0 to G_1. If there is a subset S of n triples covering X, Y, Z, then we can map the loops of G_0 to the loops corresponding to the triples in S, and the total edit cost is very close to $M - 4n$. On the other hand, if there is no such subset, then at least one edge has to be deleted from G_0, and the total cost is at least close to $M - 4n + 1$. And so 3DMatching can be decided by computing the geometric graph distance for a highly non-planar graph. $\square$

Proof (assuming C_e and C_v are part of the input and $C_v \ll C_e$ is allowed). Consider the decision version of our problem, that is, given two geometric graphs G_0 and G_1 and three positive real constants C_v, C_e and K, is the geometric graph edit distance between G_0 and G_1 less than K?

The reduction is done by using the well-known Hamiltonian path problem restricted to grid graphs, which is known to be NP-complete [4]. An instance of the Hamiltonian path problem is a grid graph G with n vertices and m edges.

The reduction is as follows. Set $C_e = 1$, $C_v = 1/(n \cdot \text{diam}(G_0 \cup G_1))$ and $K = m - n + 2$. Let G_0 be a grid path on n vertices, where each edge has length 1, and let G_1 be the grid graph defined by G on n vertices. We claim that G_1 contains a Hamiltonian path if and only if the geometric graph edit distance between G_0 and G_1 is at most K.

Consider an optimal transformation of G_0. We will argue that the optimal solution will always translate a maximum number of edges in G_0, and that all the $n - 1$ edges in G_0 can only be used if there is a Hamiltonian path in G_1.

Since G_1 is a grid graph all edges have length 1, thus, translating an edge in G_0 does not change its length. Due to the choice of C_v and C_e this implies that moving an existing edge is always cheaper than inserting a new edge.

Consider the cost of an optimal solution. Since G_0 only contains $n - 1$ edges and G_1 contains m edges one has to insert at least $m - n + 1$ new edges, with a total cost of at least $m - n + 1$. The cost of translating the vertices of G_0 does not exceed 1 in total. The $n - 1$ edges of G_0 can only be reused if G_1 contains a Hamiltonian path. Otherwise, for each deleted and inserted edge the additional cost is 2. Thus, if all the edges in G_0 can be used then the total cost is at most $K = m - n + 2$ and there exists a Hamiltonian path in G, otherwise not. $\square$

4 Landmark Distance

Since we do not know how to compute our geometric graph distance efficiently, we turned to a heuristic distance measure, the *landmark distance*. The idea of the landmark distance is to designate a few vertices of the graph as its *landmarks* and to represent the vertices of the geometric graph by their distances to the landmarks.

Formally, let $G = (V, E)$ be a geometric graph, and let K_G be the complete graph on the vertex set V. An edge (u, v) of K_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 v in K_G. After testing different values for the penalty p we chose $p = 1.6$ for our experiments.

Let $L = w_1, \ldots, w_k$ be k vertices of G called *landmarks* (in our experiments $k = 4$). 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:

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

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

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 [8]. 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, a point $p \in P$ can be seen as a pile of earth of size $w(p)$, while a point $q \in Q$ can be seen as a hole in the ground of size $w(q)$. 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.

Formally, nEMD can be defined by the following linear program. Let f_{pq} denote the flow of earth from $p \in P$ to $q \in Q$. Then

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

where $d(p, q)$ is the distance between p and q in some underlying metric, subject to the constraints:

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

In our definition of the landmark distance, we give all points of $R(G)$ equal weight, and we use the ℓ_1-metric[4] in $\mathbb{R}^{k+2}$ as the underlying distance for the nEMD.

The landmark distance is "nearly" a metric:

Theorem 3. *The landmark distance on graphs with given landmarks has the following properties:*
(i) $d_L(G_0, G_0) = 0$ *and* $d_L(G_0, G_1) \geq 0$*,*
(ii) $d_L(G_0, G_1) = d_L(G_1, G_0)$*,*
(iii) $d_L(G_0, G_2) \leq d_L(G_0, G_1) + d_L(G_1, G_2)$*.*

Proof. All the properties in the theorem follow directly from the fact that the normalized EMD is a metric [8]. □

However, it is easy to see that $d_L(G_0, G_1) = 0$ does not imply $G_0 = G_1$.

5 Experimental results

We performed pattern retrieval experiments on a database of graphs generated from Chinese character glyphs using the landmark distance. The motivation here

[4] Whether the ℓ_1- or ℓ_2-metric is used makes little difference in the experiments. The advantage of the ℓ_1-metric is that it does not need floating point arithmetic.

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 a large number of geometric 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 3 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).

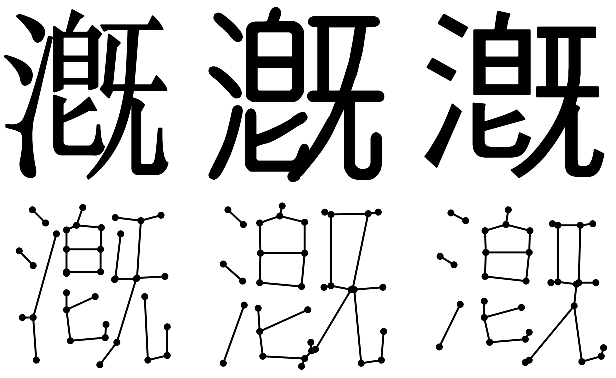

Fig. 3. Three glyphs and generated graphs for the Chinese character U+6f11 "slowly flowing water."

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 generated from glyphs 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 font.

We then considered each of the remaining 20880 graphs as a *pattern*, and computed its distance to each model, using the EMD implementation by Rubner et al. [8]. 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.

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

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 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 (that is, a graph drawing is interpreted as a set of points in the plane). The results are surprisingly good considering that the edge information of the graphs is not used at all: for 94.5% of the patterns, the correct Chinese character is found in the top ten.

Landmark selection. We fixed four landmarks in each model graph, by choosing the four vertices that are extreme in the four diagonal directions, that is, the vertices extreme in the directions $(1, 1)$, $(1, -1)$, $(-1, -1)$, and $(-1, 1)$.

Ideally, we would apply the same approach to the pattern graph, but very often this does not find the "right" vertex, as for example in Figure 4. In such a case, the distance between the graphs is far larger than it should be—selecting the right landmarks in each graph is critical to the success of the landmark distance.

We actually try all plausible choices of landmarks for a given pattern, and then use the landmarks that result in the smallest distance to a given model (so we could select different landmarks for different models). We consider a vertex to be a plausible landmark if it is the corner of a quadrant that contains no vertices.

Speeding up the computation. It turns out that computing the EMD is a rather expensive operation, and computing 4176×20880 EMD distances in every experiment is very time-consuming. For instance, for U+6f01, whose graph has

33 vertices, computing the EMD for each model in the entire model database takes 31 seconds.

We therefore used a heuristic to speed up the computation: Instead of 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

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

Note that this "distance" is not symmetric—we compute it with G_0 being the pattern, G_1 being one of the models. It is also by itself not a very good distance measure, and ranks the models much more poorly than the EMD.

So what we do instead is to first rank all models using the simplified landmark distance. We then look at the top 200 models (that is, the 200 models with the smallest simplified landmark distance), and recompute the distance from the pattern to these 200 models using the landmark distance (that is, using the EMD computation).

The heuristic greatly speeds up the computation: Comparing the character U+610f mentioned above against the entire model database now takes only 1.8 seconds, a speed-up of around 17. This speed-up comes at nearly no cost—the quality of the recognition is nearly identical to using the full EMD computation.

The result. The result of the our landmark distance experiment is given in the last line of Table 1. For 85.3% of the patterns our approach finds the correct Chinese character in the top position, and in 97.2% of the cases it is in the top ten. This is a small but real improvement over the EMD for this dataset.

6 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.

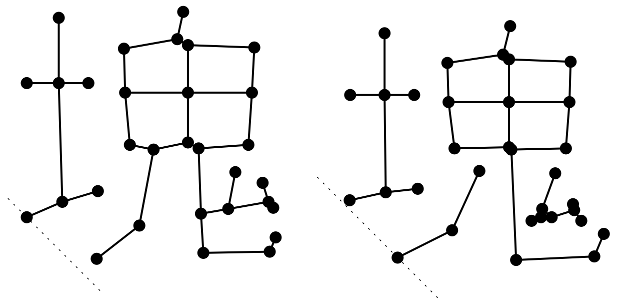

Fig. 4. Extremal vertices are not robust.

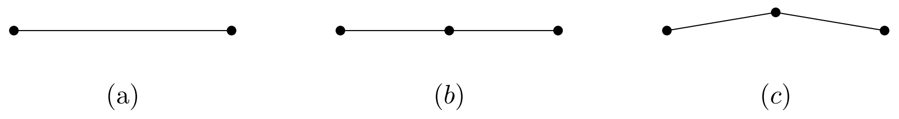

Fig. 5. Our distance does not handle bending edges well.

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. The three graphs shown in Figure 5 are all quite similar, but our distance is large between (a) and (b) and small between (b) and (c). Can we define a metric that allows this operation while still being computable at least through integer linear programming?[5]

As for our experiments, perhaps the graphs in our database are "too easy" in the sense that even ignoring the edge information, the EMD does pretty well. It would be interesting to try harder graph sets, or to consider arbitrary rigid motions.

References

1. H. Alt and L.J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, 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 G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. Elsevier B.V., 1988.
4. A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.
5. 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.
6. 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.
7. T. Pavlidis. *Structural Pattern Recognition.* Springer-Verlag, New York, 1977.
8. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), 2000. `http://robotics.stanford.edu/~rubner`.

[5] A natural approach would be to allow for free insertion of vertices on edges before all other operations, and deletion of degree-two vertices where the incident edges form an angle of 180° after all other operations. This, however, defines a measure that is not a metric.