HAL open science

Map Edit Distance vs Graph Edit Distance for Matching Images

Camille Combier, Guillaume Damiand, Christine Solnon

To cite this version:

Camille Combier, Guillaume Damiand, Christine Solnon. Map Edit Distance vs Graph Edit Distance for Matching Images. Graph-Based Representation in Pattern Recognition, May 2013, Vienna, Austria. pp.152-161, 10.1007/978-3-642-38221-5_16 . hal-00825788

HAL Id: hal-00825788

https://hal.science/hal-00825788v1

Submitted on 24 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Map Edit Distance vs Graph Edit Distance for Matching Images*

Camille Combier[1,2], Guillaume Damiand[3,2], and Christine Solnon[3,2]

[1] Université Lyon 1, LIRIS, UMR 5205 CNRS, 69622 Villeurbanne, France
[2] Université de Lyon, France
[3] INSA de Lyon, LIRIS, UMR 5205 CNRS, 69621 Villeurbanne, France
{guillaume.damiand,christine.solnon}@liris.cnrs.fr

Abstract. Generalized maps are widely used to model the topology of nD objects (such as 2D or 3D images) by means of incidence and adjacency relationships between cells (0D vertices, 1D edges, 2D faces, 3D volumes, ...). We have introduced in [1] a map edit distance. This distance compares maps by means of a minimum cost sequence of edit operations that should be performed to transform a map into another map. In this paper, we introduce labelled maps and we show how the map edit distance may be extended to compare labeled maps. We experimentally compare our map edit distance to the graph edit distance for matching regions of different segmentations of a same image.

1 Motivations

In many computer vision applications we have to match interest points or regions extracted from different images in order to, *e.g.*, recognize objects or reconstitute 3D models from 2D images. When looking for such matchings, graph-based approaches offer a good compromise between local approaches, which match each point with the most similar point of the other image independently from its relationships with other points, and global approaches such as RANSAC, which consider rigid transformations. Indeed, graph-based approaches are able to exploit local relationships while being more tolerant to deformations than global approaches such as RANSAC.

There exist different kinds of graph matchings [2], ranging from subgraph isomorphism to more error-tolerant matchings such as the graph edit distance. The graph edit distance is a generic measure, which is parametrized by edition costs, and it is widely used to match graphs. It defines the distance between two graphs G_1 and G_2 as the minimum cost sequence of edit operations for transforming G_1 into G_2. Edit operations are vertex and edge deletion, insertion and substitution. A vertex matching may be derived from the sequence of edit operations in a

* Paper published in Proceedings of 9th Workshop on Graph-Based Representation in Pattern Recognition, LNCS 7877, pp. 152-161, May 2013. Thanks to Springer Berlin Heidelberg. The original publication is available at http://dx.doi.org/10.1007/978-3-642-38221-5_16

straightforward way: Any vertex v_1 of G_1 which is substituted to a vertex v_2 of G_2 is actually matched with v_2.

Graphs are well suited to model binary relationships such as point proximity or region adjacency. However, graphs are less well suited to model the topology of the subdivision of a plane in faces, edges, and vertices. Combinatorial maps are very nice data structures to model this kind of topological information: they model the topology of nD objects subdivided into cells (*e.g.*, vertices, edges, faces, volumes, ...) by means of incidence and adjacency relationships between these cells. Combinatorial maps have been extended to generalized maps in [3], which are fully homogeneous in any dimension, thus simplifying algorithms and the development of computer libraries. In 2D, combinatorial and generalized maps may be used to model the topology of an embedding of a planar graph in a plane. In particular, these models are very well suited for scene modeling [4], for 2D and 3D image segmentation [5], and there exist efficient algorithms to extract maps from images [6].

We have defined a map edit distance in [1]. This map edit distance is a straightforward extension of the graph edit distance: it defines the distance between two maps as a minimum cost sequence of edit operations, and a matching may be derived from this edit operation sequence. However, this map edit distance has been defined for non labelled maps. In this paper, we introduce labelled maps, such that cells may be associated with labels which describe their properties, and we extend our map edit distance to labelled maps. Another goal of this paper is to compare our map edit distance with the graph edit distance for matching regions of different segmentations of a same image, and therefore answer the following question: Does the topology of the subdivision of the image in regions (besides region adjacency relationships) help to match image regions?

Outline of the paper. In Section 2, we recall definitions related to generalized maps and to the map edit distance. In Section 3, we introduce labelled maps and show how the map edit distance may be extended to handle labels. In Section 4, we experimentally compare our map edit distance to the graph edit distance for matching regions of segmented images. In Section 5, we discuss further work.

2 Recalls on generalized maps and the map edit distance

In this work we consider generalized maps, and we refer the reader to [3] for more details.

Definition 1 (nG-map). *Let $n \geq 0$. An n-dimensional generalized map (or nG-map) is defined by a tuple $G = (D, \alpha_0, \ldots, \alpha_n)$ such that*

1. *D is a finite set of darts;*
2. *$\forall i \in \{0, \ldots, n\}$, α_i is an involution on D (*i.e.*, it is a bijection such that $\forall d \in D, \alpha_i(\alpha_i(d)) = d$);*
3. *$\forall i, j \in \{0, \ldots, n\}$ such that $i + 2 \leq j$, $\alpha_i \circ \alpha_j$ is an involution.*

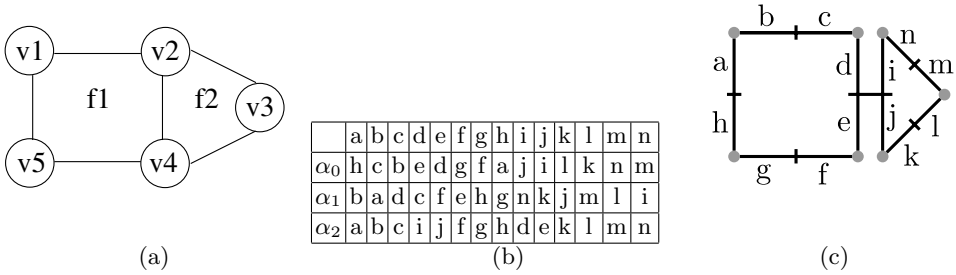


Fig. 1. (a) A plane graph. (b) The corresponding 2G-map. (c) Its graphical representation: darts are represented by segments labeled with letters, consecutive darts separated with a little segment are 0-sewn (*e.g.*, $\alpha_0(b) = c$ and $\alpha_0(c) = b$), consecutive darts separated with a dot are 1-sewn (*e.g.*, $\alpha_1(a) = b$ and $\alpha_1(b) = a$), parallel darts are 2-sewn (*e.g.*, $\alpha_2(d) = i$ and $\alpha_2(i) = d$).

We say that a dart d is i-sewn with a dart d' whenever $d' = \alpha_i(d)$ and $d \neq d'$, whereas it is i-free whenever $d = \alpha_i(d)$. We say that a dart d is free if it is i-free for every dimension.

2G-maps may be used to model the embedding of a planar graph into a plane. For example, Fig. 1 displays a plane graph, composed of 5 vertices, 6 edges and 2 faces, and the corresponding 2G-map, composed of 14 darts.

Cells are implicitly defined by sets of darts corresponding to orbits: the i-cell incident to a dart d is defined by $cell_i(d) =< \{\alpha_0, \ldots, \alpha_n\} \setminus \{\alpha_i\} > (d)$. Let us consider, for example, the 2G-map of Fig. 1. The cells incident to dart e are:

- $cell_0(e) =< \{\alpha_1, \alpha_2\} > (e) = \{e, f, j, k\}$, corresponding to vertex v_4;
- $cell_1(e) =< \{\alpha_0, \alpha_2\} > (e) = \{d, e, i, j\}$, corresponding to edge (v_2, v_4);
- $cell_2(e) < \{\alpha_0, \alpha_1\} > (e) = \{a, b, c, d, e, f, g, h\}$, corresponding to face f_1.

The map edit distance is based on edit operations which are used to transform maps. These edit operations allow one to add/delete free darts, and to sew/unsew darts. More precisely, let $G = (D, \alpha_0, \ldots, \alpha_n)$ be an nG-map.

- Let $d \in D$ be a free dart of G (*i.e.*, $\forall i \in \{0, \ldots, n\}, \alpha_i(d) = d$). The $del_d(G)$ operation removes d from D.
- Let $d \notin D$ be a dart. The $add_d(G)$ operation adds d to D so that d becomes a free dart of G, *i.e.*, $\forall i \in \{0, \ldots, n\}, \alpha_i(d) = d$.
- Let S be a set of triples (d, i, d') such that $d \in D$ and $d' \in D$ are i-free (*i.e.*, $d \neq d', \alpha_i(d) = d$, and $\alpha_i(d') = d'$). The $sew_S(G)$ operation i-sews d to d' for every triple $(d, i, d') \in S$, *i.e.*, it sets $\alpha_i(d)$ to d' and $\alpha_i(d')$ to d.
- Let S be a set of triples (d, i, d') such that $d \in D$ and $d' \in D$ are i-sewn darts (*i.e.*, $d \neq d', \alpha_i(d) = d'$, and $\alpha_i(d') = d$). The $unsew_S(G)$ operation i-unsews d to d' for every triple $(d, i, d') \in S$, *i.e.*, it sets $\alpha_i(d)$ to d and $\alpha_i(d')$ to d'.

When comparing these edit operations to classical graph edit operations, the *del* and *add* operations are related to vertex deletion and addition operations, whereas the *sew* and *unsew* operations are related to edge deletion and addition

operations. A main difference is that sew/unsew operations operate on sets of darts instead of sewing/unsewing darts one by one. Indeed, sewing/unsewing a single dart may lead to a non valid nG-map. Let us consider for example the nG-map of Fig. 1. We cannot 2-unsew darts d and i without also 2-unsewing darts e and j (otherwise $\alpha_0 \circ \alpha_2$ no longer is an involution so that Property 3 of definition 1 no longer is satisfied). The map edit distance is then defined as the cost of the minimal cost edit path.

Definition 2 (edit path). *Let G be an nG-map and $\Delta =< \delta_1, \ldots, \delta_k >$ be a sequence of k edit operations. Δ is an edit path for G if $\delta_k(\delta_{k-1}(\ldots(\delta_1(G))))$, denoted $\Delta(G)$, is an nG-map (according to definition 1).*

Definition 3 (map edit distance). *Let c be a function which associates a cost $c(\delta) \in \mathbb{R}^+$ with every operation δ. The edit distance between the two nG-maps G and G' is $d_c(G, G') = \sum_{\delta_i \in \Delta}(c(\delta_i))$ where Δ is an edit path such that $\Delta(G) = G'$ and $\sum_{\delta_i \in \Delta}(c(\delta_i))$ is minimal.*

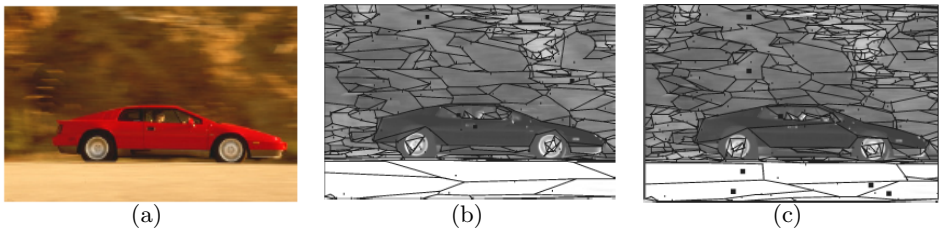
3 Extension of the map edit distance to labelled maps

Generalized maps describe the topology of the subdivision of a space into cells. However, they do not express other information such as, for example, geometry, texture or colour information. This kind of information may be added by means of labels associated with cells. In generalized maps, cells are implicitly defined by sets of darts and correspond to orbits. Therefore, to associate a label with an i-cell c, we propose to label every dart of c, *i.e.*, every dart d such that $cell_i(d) = c$. Note that, a dart belongs to exactly one cell for every dimension $i \in \{0, \ldots, n\}$. For consistency reasons, we impose that all darts of a same i-cell have the same label for dimension i.

Definition 4 (Labelled nG-maps). *Let $n \geq 1$. A labelled nG-map is a tuple $G = (D, \alpha_0, \ldots, \alpha_n, L, l)$ such that $(D, \alpha_0, \ldots, \alpha_n)$ is an nG-map, L is a set of labels, and $l : D \times \{0, \ldots, n\} \rightarrow L$ is a labelling function such that $\forall d, d' \in D, \forall i \in \{0, \ldots, n\}, cell_i(d) = cell_i(d') \Rightarrow l(d, i) = l(d', i)$.*

In other words, $l(d, i)$ is the label associated with the i-cell incident to d. Let us consider, for example, the 2G-map of Fig. 1. To associate the label x with vertex v_4, we define $l(e, 0) = l(f, 0) = l(j, 0) = l(k, 0) = x$; to associate the label y with edge (v_2, v_4), we define $l(d, 1) = l(e, 1) = l(i, 1) = l(j, 1) = y$; to associate the label z with face f_1, we define $l(a, 2) = l(b, 2) = l(c, 2) = l(d, 2) = l(e, 2) = l(f, 2) = l(g, 2) = l(h, 2) = z$.

The map edit distance of [1] has been defined for non labelled maps. To extend it to labelled maps, we introduce a new edit operation that substitutes dart labels. Let $G = (D, \alpha_0, \ldots, \alpha_n, L, l)$ be a labelled nG-map, $d \in D$ be a dart, $i \in \{0, ..., n\}$, and $l' \in L$ be a label. The $subs_{(d,l',i)}(G)$ operation substitutes the label $l(d, i)$ of dart d with the new label l'. The cost function c must also be extended so that the cost of an edit operation ($subs, del, add, sew$, or $unsew$) depends on dart labels.



(a) (b) (c)

Fig. 2. (*a*) An image. (*b*) A segmentation of (a) in 389 regions. (*c*) A finer segmentation of (a) in 415 regions.

4 Experimental comparison

In this section, we compare the map edit distance with the graph edit distance for matching regions of different segmentations of a same image. Our goal is to evaluate the interest of using maps, which model the topology of the subdivision of the image in regions, instead of using graphs, which only model region adjacency relationships.

4.1 Test suite

For this very first experimental comparison, we compare different segmentations of a same image. This allows us to have a ground truth for evaluating matchings: we consider that two regions coming from two different segmentations of the same image are correctly matched if their intersection is not empty. We have considered 6 different images (2 cars, 2 cows and 2 motorbikes) extracted from the ETHZ benchmark[4]. For each image, we have generated different segmentations, using the algorithm of [7] with different threshold values, so that the number of regions varies from 240, for the coarser segmentations, to 460, for the finer ones. Fig. 2 gives an example of two segmentations of a same image. Note that all segmentations are recomputed from the same initial image so that regions of finer segmentations are not necessarily subdivisions of regions of coarser segmentations.

For each segmentation, we have built a graph and a 2G-map which represent it. The graph is a classical region adjacency graph (RAG), which associates a vertex with each region, and an edge with every pair of adjacent regions. The 2G-map associates a face with every region, the edges of the map describe the adjacency relations between the regions and the vertices of the map describe the adjacency relations between the edges.

The sizes of the resulting RAGs and 2G-maps are given in Table 1. Note that if the number of faces of the 2G-maps corresponds to the number of vertices of the RAGs, the 2G-maps have slightly more edges than RAGs as two regions having multiple adjacency relationships are linked by a single edge in RAGs (multiple adjacency occurs when two regions are adjacent several times).

[4] available at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html`

RAGs				2G-maps							
Nb of vertices		Nb of edges		Nb of darts		Nb of vertices		Nb of edges		Nb of faces	
Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
240	460	688	1217	2792	4948	463	816	698	1237	240	460

Table 1. Comparison of the sizes of RAGs and 2G-maps, for the coarser segmentations (Min) and the finer ones (Max).

Each region r of the segmented images is described by two basic descriptors: A color descriptor, $color(r)$, which is the average color of the pixels of r (a value ranging between 0 and 255 as we consider gray-level colours), and an area descriptor, $area(r)$, which is the number of pixels of r.

First experiments showed us that the graph edit distance can hardly correctly match regions when RAGs are not labelled. Actually, RAGs usually have many automorphisms (*i.e.*, symetries), so that there exist many different matchings between two isomorphic RAGs which preserve adjacency relationships (but of course, only one of these matchings correctly matches regions). In order to improve the matching process, we have added structural labels to RAGs. Therefore, for RAGs:

- each edge (u, v) is labelled with $adj(u, v)$, the number of adjacency relationships between the two regions associated with u and v;
- each vertex u corresponding to a region r_u is labelled with a triple $(totAdj(u), color(r_u), area(r_u))$, where $totAdj(u) = \sum_v (adj(u, v))$ is the total number of adjacency relationships of all edges (u, v) incident to u.

The structural labels $adj(u, v)$ and $totAdj(u)$ greatly improve results for RAGs. We do not add these structural labels to 2G-maps as this information is already available in 2G-maps. Therefore, for every dart d of 2G-maps, we define $l(d, 2) = (color(r_d), area(r_d))$, where r_d is the region associated with the dart d. As no information is associated with vertices and edges of the 2G-maps, we define $l(d, 0) = l(d, 1) = \epsilon$.

4.2 Cost functions

For RAGs, we define the cost of substituting a vertex u whose label is $(totAdj(u), color(r_u), area(r_u))$ with a vertex v whose label is $(totAdj(v), color(r_v), area(r_v))$ by

$$\begin{aligned} c(subs(u, v)) = \omega_{struct} &\cdot |totAdj(u) - totAdj(v)| \\ &+ \omega_{color} \cdot \frac{|color(r_u) - color(r_v)|}{255} \\ &+ \omega_{area} \cdot (1 - \frac{min(area(r_u), area(r_v))}{max(area(r_u), area(r_v))}) \end{aligned}$$

where ω_{struct}, ω_{color}, and ω_{area} are 3 parameters which determine the relative weights of structural, color and area information. The cost of adding or deleting a vertex or an edge is set to 1.

For 2G-maps, we define the cost of substituting a dart u whose label is $l(u,2) = (color(r_u), area(r_u))$ with a dart v whose label is $l(v,2) = (color(r_v),\ area(r_v))$ by

$$c(subs(u,v)) = \omega_{color} \cdot \frac{|color(r_u) - color(r_v)|}{255} + \omega_{area} \cdot (1 - \frac{min(area(r_u), area(r_v))}{max(area(r_u), area(r_v))})$$

where ω_{color} and ω_{area} are 2 parameters which determine the relative weights of color and area information. The cost of adding or deleting a dart is set to 1. The cost of sewing/unsewing operations is equal to the number of triples (d, i, d') added/removed (as sew/unsew operations add/remove sets of seams for consistency reasons).

4.3 Matching algorithms

Computing edit distances is a NP-hard problem, both for graphs and maps. Exact algorithms do not scale well and cannot compute edit distances within a reasonable amount of time for the graphs and maps considered here. Therefore, we use heuristic algorithms, which compute approximate solutions.

For the graph edit distance, we use the algorithm proposed in [8]: The graph matching problem is approximated by an assignment problem which is solved by the Munkres algorithm [9].

For the map edit distance, we use an extension to labelled maps of the greedy algorithm described in [10]: Starting from an empty matching, this algorithm iteratively matches darts until no more darts can be matched; at each iteration the pair of darts to be matched is chosen in order to minimize the corresponding edit costs.

Both algorithms have polynomial time complexities: $\mathcal{O}(v^3)$ for the graph edit distance, where v is the number of vertices of the largest graph, and $\mathcal{O}(d^2 \cdot log(d))$ for the map edit distance, where d is the number of darts of the largest map. However, as d is more than ten times larger than v on our benchmark, the matching process is faster for graphs than for maps: for the coarsest segmentations, having 240 regions, graphs (having 240 vertices) are matched in 2 seconds or so whereas maps (having 2792 darts) are matched in 8 seconds or so; for the finest segmentations, having 460 regions, graphs (having 460 vertices) are matched in 5 seconds or so whereas maps (having 4948 darts) are matched in 25 seconds or so.

4.4 Experimental results

Let us now compare graph and map edit distances for matching regions of two different segmentations of a same image. Comparing different segmentations of a same image allows us to have a ground truth: we consider that two regions coming from two different segmentations of the same image are correctly matched if their intersection is not empty. When images are modelled with RAGs, we measure the percentage of vertices which are correctly matched, *i.e.*, whose associated regions are correctly matched. When images are modelled with 2G-maps, we measure the

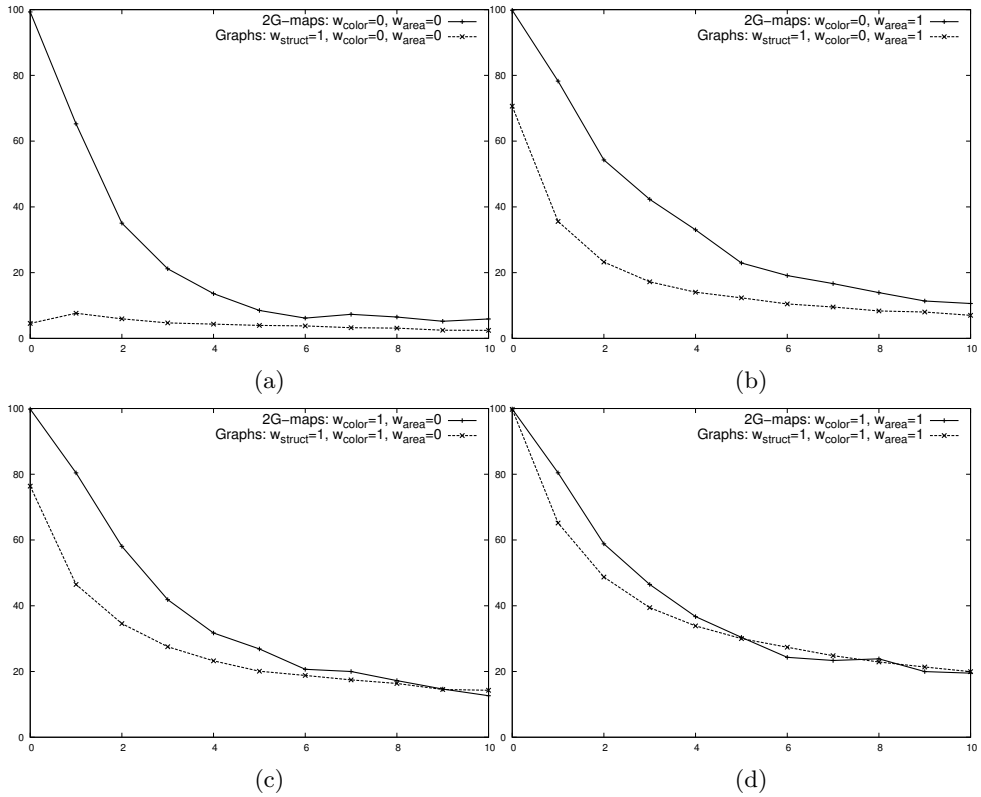


Fig. 3. Average percentage of correctly matched darts/vertices (on the y-axis) with respect to the difference k of segmentation levels (on the x-axis). (a) Using structural information only, *i.e.*, $\omega_{colour} = 0$ and $\omega_{area} = 0$. (b) Using structural and area information, *i.e.*, $\omega_{colour} = 0$ and $\omega_{area} = 1$. (c) Using structural and colour information, *i.e.*, $\omega_{colour} = 1$ and $\omega_{area} = 0$. (d) Using structural, colour and area information, *i.e.*, $\omega_{colour} = 1$ and $\omega_{area} = 1$.

percentage of darts which are correctly matched, *i.e.*, whose associated 2-cells are correctly matched.

For each image, we sort its different segmentations from the coarsest one (which has 240 regions) to the finest one (which has 460 regions). For each segmentation i, we measure the percentage of correctly matched vertices/darts when comparing it with segmentation $i+k$ of the same image, where the difference k of segmentation levels ranges from 0 up to 10 (it may be less than 10 if segmentation $i+10$ does not exist): when $k = 0$, we actually compare a segmentation with itself; the larger k, the more different the two segmentations.

Fig. 3 displays the evolution of the percentage of correctly matched vertices/darts with respect to the difference k of segmentation levels (on average

for the 6 images and each segmentation level i). For the graph edit distance, we always set ω_{struct} to 1 as this always improves results. We consider different combinations of the two other weight parameters ω_{colour} and ω_{weight}. Let us first compare graphs and 2G-maps when colour and area information is ignored, *i.e.*, $\omega_{colour} = 0$ and $\omega_{area} = 0$. For 2G-maps, we are able to correctly match all darts when $k = 0$ (*i.e.*, when we compare isomorphic 2G-maps), but this rate quickly decreases when increasing k: it is equal to 20% or so when $k = 3$, and smaller than 10% when $k \geq 5$. For graphs, we are never able to match more than 10% of the vertices, even when $k = 0$. Actually, RAGs have many automorphisms (*i.e.*, symetries). Some of these symetries are broken by adding the structural labels (number of adjacency relationships on edges, and total number of adjacency relationships on vertices). However, even with structural labels, RAGs still have many automorphisms so that a vertex may be matched with several vertices.

Adding colour or area information, *i.e.*, setting ω_{colour} or ω_{area} to 1, significantly improves results and, when $k \leq 5$, results obtained with 2G-maps are significantly better than those obtained with graphs. However, for higher values of k, the percentage of correctly matched darts/vertices hardly reaches 20%.

Finally, when combining colour and area information, graphs and 2G-maps obtain rather similar results, though 2G-maps are slightly better than graphs when $k \leq 4$.

This first experiment shows us that structural information is better modelled and exploited with 2G-maps than with graphs. This comes from the fact that 2G-maps do not only model region adjacency relationships, but also other topological information such as, for example, the order in which faces are encountered when turning around a vertex. As a matter of fact, if RAGs usually have many automorphisms, 2G-maps usually have no automorphism at all. However, when adding colour or area labels, the difference between 2G-maps and graphs becomes less significant as this information greatly improves the graph matching process.

Note that the differences observed between 2G-maps and graphs may also come from the matching algorithms we have considered: these matching algorithms are heuristic algorithms which compute approximate solutions. The graph matching algorithm of [8] considers only local, rather than global, edge structure during the optimization process. Also, the greedy map matching algorithm of [10] considers local seams to choose the next pair of darts to match. It is not possible to compute exact solutions within a reasonable amount of time, considering the fact that graphs (resp. maps) have hundreds (resp. thousands) of vertices (resp. darts). Therefore, we cannot assess the quality of the approximations computed by the heuristic algorithms.

5 Conclusion

In this paper, we have extended generalized maps to labelled nG-maps, thus allowing us to add information on cells in every dimension, and we have extended the map edit distance to handle these labels by adding a dart substitution operation. We have compared the map edit distance with the graph edit distance

for matching regions of different segmentations of a same image. We have shown that regions are better matched when we use 2G-maps for modelling the topology of the subdivision of the image in regions rather than when we use RAGs for modelling region adjacency relationships.

We have performed similar experiments on 2D meshes modelling 3D objects. We have generated different degradations of a same mesh (obtained by merging nearly co-planar adjacent faces), and compare the percentage of correctly matched faces when using 2G-maps and when using graphs. We observed similar results, *i.e.*, 2G-maps allow us to better match faces than graphs.

As future works, we plan to study different domains of application which use graphs and similarity measures in order to see if we can improve existing solutions by using nG-maps and the map edit distance. We also would like to improve our algorithm in order to speed-up the computation times and propose more heuristics to guide the choice of the best pair of darts to be matched. Lastly we plan to study other types of similarity measures. We think for example to extend graph kernels to generalized maps.

References

1. Combier, C., Damiand, G., Solnon, C.: From maximum common submaps to edit distances of generalized maps. Pattern Recognition Letters **33**(15) (2012) 2020–2028
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years Of Graph Matching In Pattern Recognition. International Journal of Pattern Recognition and Artificial Intelligence (2004)
3. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. Computational Geometry and Applications **4**(3) (1994) 275–324
4. Fradin, D., Meneveaux, D., Lienhardt, P.: A hierarchical topology-based model for handling complex indoor scenes. Computer Graphics Forum **25**(2) (June 2006) 149–162
5. Braquelaire, J.P., Brun, L.: Image segmentation with topological maps and inter-pixel representation. Visual Communication and Image representation **9**(1) (1998) 62–79
6. Damiand, G.: Topological model for 3d image representation: Definition and incremental extraction algorithm. Computer Vision and Image Understanding **109**(3) (2008) 260–289
7. Dupas, A., Damiand, G.: First results for 3d image segmentation with topological map. In Coeurjolly, D., Sivignon, I., Tougne, L., Dupont, F., eds.: DGCI. Volume 4992 of Lecture Notes in Computer Science., Springer (2008) 507–518
8. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vision Comput. **27** (June 2009) 950–959
9. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society of Industrial and Applied Mathematics **5**(1) (March 1957) 32–38
10. Combier, C., Damiand, G., Solnon, C.: Measuring the distance of generalized maps. In: GbR. LNCS, Springer (2011) 82–91