

# GriMa: a Grid Mining Algorithm for Bag-of-Grid-Based Classification

Romain Deville, Elisa Fromont, Baptiste Jeudy, Christine Solnon

► **To cite this version:**

Romain Deville, Elisa Fromont, Baptiste Jeudy, Christine Solnon. GriMa: a Grid Mining Algorithm for Bag-of-Grid-Based Classification. The joint IAPR International Workshops on Structural and Syntactic Pattern Recognition (SSPR 2016) and Statistical Techniques in Pattern Recognition (SPR 2016), Nov 2016, Merida, Mexico. pp.132-142. hal-01376019

**HAL Id: hal-01376019**

**<https://hal.archives-ouvertes.fr/hal-01376019>**

Submitted on 5 Oct 2016

**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.

# GriMa: a Grid Mining Algorithm for Bag-of-Grid-Based Classification

Romain Deville<sup>1,2</sup>, Elisa Fromont<sup>1</sup>, Baptiste Jeudy<sup>1</sup>, and Christine Solnon<sup>2</sup>

<sup>1</sup> UJM, CNRS, LaHC UMR 5516, F-42000, SAINT-ETIENNE, France

<sup>2</sup> Université de Lyon, INSA-Lyon, LIRIS, UMR5205, F-69621, France

**Abstract.** General-purpose exhaustive graph mining algorithms have seldom been used in real life contexts due to the high complexity of the process that is mostly based on costly isomorphism tests and countless expansion possibilities. In this paper, we explain how to exploit grid-based representations of problems to efficiently extract frequent grid subgraphs and create Bag-of-Grids which can be used as new features for classification purposes. We provide an efficient grid mining algorithm called GRiMA which is designed to scale to large amount of data. We apply our algorithm on image classification problems where typical Bag-of-Visual-Words-based techniques are used. However, those techniques make use of limited spatial information in the image which could be beneficial to obtain more discriminative features. Experiments on different datasets show that our algorithm is efficient and that adding the structure may greatly help the image classification process.

## 1 Introduction

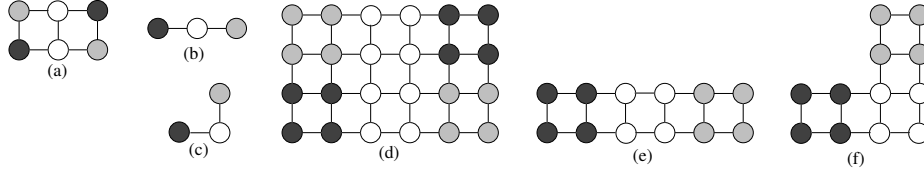
General-purpose exhaustive graph mining algorithms are seldom used in real-world applications due to the high complexity of the mining process mostly based on isomorphism tests and countless expansion possibilities during the search [6]. In this paper we tackle the problem of exhaustive graph mining for grid graphs which are graphs with fixed topologies and such that each vertex degree is constant, *e.g.*, 4 in a 2D square grid. This grid structure is naturally present in many boardgames (Checkers, Chess, Go, etc.) or to model ecosystems using cellular automata [10], for example. In addition, this grid structure may be useful to capture low-level topological relationships whenever a high-level graph structure is not obvious to design. In computer vision in particular, it is now widely acknowledged that high-level graph-based image representations (such as region adjacency graphs or interest point triangulations, for example) are sensitive to noise so that slightly different images may result in very different graphs [20,19]. However, at a low-level, images basically are grids: 2D grids of pixels for images, and 3D grids of voxels for tomographic or MRI images modelling 3D objects. When considering videos, we may add a temporal dimension to obtain 2D+t grids. We propose to exploit this grid structure and to characterize images by histograms of frequent subgrid patterns.

Section 2 describes existing works that use pattern mining approaches for image classification and existing graph mining algorithms related to our proposed approach. Section 3 introduces grid graphs and our sub-grid mining algorithm GRIMA. Section 4 experimentally compares these algorithms and shows the relevance of GRIMA for image classification.

## 2 Related Work

*Mining for image classification.* Pattern mining techniques have recently been very successfully used in image classification [9,22] as a mean to obtain more discriminative mid-level features. However, those approaches consider the extracted features used to describe images (*e.g.*, Bags-of-Visual-Words / BoWs) as spatially independent from each other. The problem of using bag-of-graphs instead of BoWs has already been mentioned in [2,18,21] for satellite image classification and biological applications. However, none of these papers provide a general graph representation nor a graph mining algorithm to extract the patterns. In [17], authors have already shown that, by combining graph mining and boosting, they can obtain classification rules based on subgraph features that contain more information than sets of features. The GSPAN algorithm [23] is then used to compute the subgraph patterns but a limited number of features per image is used to be able to scale on real-life datasets.

*Graph Mining.* GSPAN [23] and all similar general exhaustive graph mining algorithms [11] extract frequent subgraphs from a base of graphs. During the mining process, GSPAN does not consider edge angles so that it considers as isomorphic two subgraphs that are different in a grid point of view as shown in Fig. 1 (b and c). Because of this, GSPAN may consider as frequent a huge number of patterns and does not scale well. On the other hand, PLAGRAM [19] has been developed to mine plane graphs and thus to scale efficiently on these graphs. However, in PLAGRAM, the extension strategy (which is a necessary step for all exhaustive graph mining algorithms) is based on the faces of the graph which induces an important restriction: All patterns should be composed of faces and the smallest possible subgraph pattern is a single face, *i.e.*, a cycle with 3 nodes. Using PLAGRAM to mine grids is possible but the problem needs to be transformed such that each grid node becomes a face. This transformation is illustrated in Fig. 1: The two isomorphic graphs (b) and (c) (which are both subgraphs of (a)) are transformed into the two non-isomorphic graphs (e) and (f) (so that only (e) is a subgraph of (d)). However, this artificially increases the number of nodes and edges which may again cause scalability issues. Grid graphs have already been introduced in [14], but authors did not take into account the rigid topology of the grid and, in particular, the angle information that we use to define our grid mining problem. Finally, grid graphs are special cases of geometric graphs, for which mining algorithms have been proposed in [1]. These algorithms (FREQGEO and MAXGEO) may be seen as a generalization of our new algorithm GRIMA but, as such, they are not optimized for cases where the graph is known to be a grid



**Fig. 1.** Examples. (b) and (c) are isomorphic graphs and are both subgraphs of (a). However, (b) and (c) are not grid isomorphic, and (b) is a subgrid of (a) whereas (c) is not a subgrid of (a). (d) (resp. (e) and (f)) is obtained from (a) (resp. (b) and (c)) by replacing each node by a 4-node face (with the same label on the 4 nodes). (e) and (f) are not isomorphic, and (e) is a subgraph of (d) whereas (f) is not a subgraph of (d).

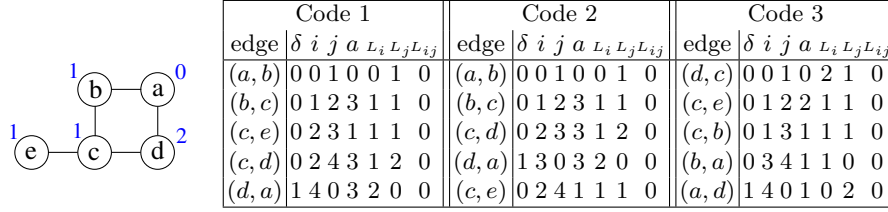
and we show in Sec. 3 that their complexity is higher than the complexity of our algorithm. Besides, the authors do not provide any implementation (thus, also no experiment) of their proposed algorithm which could allow us a comparison with our method.

### 3 Grid Mining

*Grid Graphs.* A grid is a particular case of graph such that edges only connect nodes which are neighbors on a grid. More formally, a grid is defined by  $G = (N, E, L, \eta)$  such that  $N$  is a set of nodes,  $E \subseteq N \times N$  is a set of edges,  $L : N \cup E \rightarrow \mathbb{N}$  is a labelling function,  $\eta : N \rightarrow \mathbb{Z}^2$  maps nodes to 2D coordinates, and  $\forall (u, v) \in E$ , the coordinates of  $u$  and  $v$  are neighbors, *i.e.*,  $|x_u - x_v| + |y_u - y_v| = 1$  where  $\eta(u) = (x_u, y_u)$  and  $\eta(v) = (x_v, y_v)$ . Looking for patterns in a grid amounts to searching for subgrid isomorphisms. In an image analysis context, patterns should be invariant to translations and rotations so that grid patterns which are equivalent up to a translation and/or a rotation should be isomorphic. For example, graphs (b) and (c) of Fig. 1 are isomorphic. However, (c) cannot be obtained by translating and/or rotating (b), because the angle between edges of (c) is different from the angle between edges of (b). Therefore, (b) and (c) are not grid-isomorphic. Finally, a grid  $G_1 = (N_1, E_1, L_1, \eta_1)$  is subgrid-isomorphic to a grid  $G_2 = (N_2, E_2, L_2, \eta_2)$  if there exists  $N'_2 \subseteq N_2$  and  $E'_2 \subseteq E_2 \cap N'_2 \times N'_2$  such that  $G_1$  is grid isomorphic to  $G'_2 = (N'_2, E'_2, L_2, \eta_2)$ .

*Frequent Grid Mining Problem.* Given a database  $D$  of grid graphs and a frequency threshold  $\sigma$ , the goal is to output all frequent subgrid patterns in  $D$ , *i.e.*, all grids  $G$  such that there exist at least  $\sigma$  grids in  $D$  to which  $G$  is subgrid isomorphic.

*Grid Codes.* In graph mining algorithms, an important problem is to avoid generating the same pattern multiple times. One successful way to cope with this is to use canonical codes [23] to represent graphs and thus explore a canonical code



**Fig. 2.** Examples of grid codes (node labels are displayed in blue; edges all have the same label 0). Codes 1 and 2 correspond to traversals started from edge  $(a, b)$ , and differ on the third edge. Code 3 corresponds to a traversal started from edge  $(d, c)$  and it is canonical.

search space instead of a graph one. In this paragraph, we define the canonical code used in our algorithm. A **code**  $C(G)$  of a grid  $G$  is a sequence of  $n$  edge codes ( $C(G) = \langle ec_0, \dots, ec_{n-1} \rangle$ ) which is associated with a depth-first traversal of  $G$  starting from a given initial node. During this traversal, each edge is traversed once, and nodes are numbered: The initial node has number 0; Each time a new node is discovered, it is numbered with the smallest integer not already used in the traversal. Each edge code corresponds to a different edge of  $G$  and the order of edge codes in  $C(G)$  corresponds to the order edges are traversed. Hence,  $ec_k$  is the code associated with the  $k^{\text{th}}$  traversed edge. This edge code  $ec_k$  is the tuple  $(\delta, i, j, a, L_i, L_j, L_{(i,j)})$  where

- $i$  and  $j$  are the numbers associated with the nodes of the  $k^{\text{th}}$  traversed edge.
- $\delta \in \{0, 1\}$  is the direction of the  $k^{\text{th}}$  traversed edge:
  - $\delta = 0$  if it is forward, *i.e.*,  $i$  already appears in the prefix  $\langle ec_0, \dots, ec_{k-1} \rangle$  of the code and  $j$  is a new node which is reached for the first time;
  - $\delta = 1$  if it is backward, *i.e.*, both  $i$  and  $j$  already appear in the prefix.
- $a \in \{0, 1, 2, 3\}$  is the angle of the  $k^{\text{th}}$  traversed edge:
  - $a = 0$  if  $k = 0$  (first edge);
  - Otherwise,  $a = 2A/\pi$  where  $A \in \{\pi/2, \pi, 3\pi/2\}$  is the angle between the edge which has been used to reach  $i$  for the first time and  $(i, j)$ .
- $L_i, L_j, L_{(i,j)}$  are node and edge labels.

For example, let us consider code 1 in Fig. 2. The fourth traversed edge is  $(c, d)$ . It is a forward edge (because  $d$  has not been reached before) so that  $\delta = 0$ . The angle between  $(b, c)$  and  $(c, d)$  is  $3\pi/2$  so that  $a = 3$ . The fifth traversed edge is  $(d, a)$  which is a backward edge (as  $a$  has been reached before) so that  $\delta = 1$ .

Given a code, we can reconstruct the corresponding grid since edges are listed in the code together with angles and labels. However, there exist different possible codes for a given grid, as illustrated in Fig. 2: Each code corresponds to a different traversal (starting from a different initial node and choosing edges in a different order). We define a total order on the set of all possible codes that may be associated with a given grid by considering a lexicographic order (all edge code components have integer values). Among all the possible codes for a grid, the largest one according to this order is called **the canonical code** of this grid and it is unique.

*Description of GRIMA.* Our algorithm, GRIMA, follows a standard frequent subgraph mining procedure as described, *e.g.*, in [19]. It explores the search space of all canonical codes in a depth-first recursive way. The algorithm first computes all frequent edges and then calls an **Extend** function for each of these frequent extensions. **Extend** has one input parameter: A pattern code  $P$  which is frequent and canonical. It outputs all frequent canonical codes  $P'$  such that  $P$  is a prefix of  $P'$ . To this aim, it first computes the set  $E$  of all possible valid extensions of all occurrences of  $P$  in the database  $D$  of grids: A valid extension is the code  $e$  of an edge such that  $P.e$  occurs in  $D$ . Finally, **Extend** is recursively called for each extension  $e$  such that  $P.e$  is frequent and canonical. Hence, at each recursive call, the pattern grows.

*Node-induced GRIMA.* In our application, nodes are labelled but not edges (all edges have the same label). Thus, we designed a variant of GRIMA, called node-induced-GRIMA, which computes node-induced grids, *i.e.*, grids induced by their node sets. This corresponds to a “node-induced” closure operator on graphs where, given a pattern  $P$ , we add all possible edges to  $P$  without adding new nodes. In the experiments, we show that this optimization decreases the number of extracted patterns and the extraction time.

*Properties of GRIMA and Complexity.* We can prove (not detailed here for lack of space) that GRIMA is both *correct*, which means that it only outputs frequent subgrids and *complete*, which means that it cannot miss any frequent subgrid. Let  $k$  be the number of grids in the set  $D$  of input grids,  $n$  the number of edges in the largest grid  $G_i \in D$  and  $|P|$  the number of edges in a pattern  $P$ . GRIMA enumerates all frequent patterns in  $\mathcal{O}(kn^2 \cdot |P|^2) = \mathcal{O}(kn^4)$  time per pattern  $P$ . This is a significant improvement over FREQGEO and MAXGEO [1] which have a time complexity of  $\mathcal{O}(k^2n^4 \cdot \ln n)$  per pattern.

## 4 Experiments

To assess the relevance of using a grid structure for image classification, we propose to model images by means of grids of visual words and to extract frequent subgrids to obtain Bags-of-Grids (BoGs). We compare these BoGs with a standard classification method which uses simple unstructured Bags-of-Visual-Words (BoWs). Note that neither BoG nor BoW-based image classification give state-of-the-art results for these datasets. In particular, [5] reported that the features discovered using deep learning techniques give much better accuracy results than BoWs and all their extensions on classification problems. However, our aim is to compare an unstructured set of descriptors and a set of descriptors structured by the grid topology. The method presented in this paper is generic and may be used with any low-level features (*e.g.* deep-learned features) as labels.

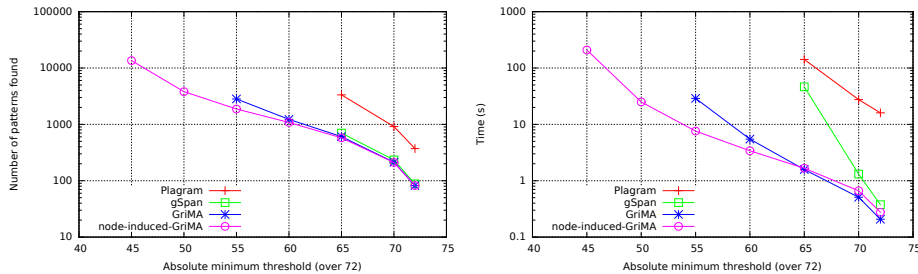
*Datasets.* We consider three datasets. *Flowers* [15] contains 17 flower classes where each class contains 80 images. *15-Scenes* [12] contains 4485 images from

15 scene classes (*e.g.*, kitchen, street, forest, etc.) with 210 to 410 images per class. *Caltech-101* [8] contains pictures of objects belonging to 101 classes. There are 40 to 800 images per class. As it is meaningless to mine frequent patterns in small datasets of images (when there are not enough images in a dataset, there may be a huge number of frequent patterns), we consider a subset of *Caltech-101* which is composed of the 26 classes of *Caltech-101* that contain at least 80 pictures per class. For each class, we randomly select 80 images. This subset will be referred to as *Caltech-26*

*BoW design.* Given a set of patches (small image regions) extracted from an image dataset, visual words are created by quantizing the values of patch descriptors [7,4]. The set of computed visual words is called the *visual vocabulary*. Each image is then encoded as an histogram of this visual vocabulary, called a bag-of-visual-words. A decade ago, patches used to create the visual vocabulary were selected by using interest point detectors or segmentation methods. However, [16] has shown that randomly sampling patches on grids (called dense sampling) gave as good (and often better) results for image classification than when using complex detectors. This made this problem a particularly suited use-case for our algorithm. In our experiments, visual words are based on 16x16 SIFT descriptors [13] which are 128-D vectors describing gradient information in a patch centered around a given pixel. 16x16 SIFT descriptors are extracted regularly on a grid and the center of each SIFT descriptor is separated by  $s = 8$  pixels (descriptors are thus overlapping). We use the K-means algorithm to create visual word dictionaries for each dataset. The optimal number of words  $K$  is different from one method (BoG) to the other (BoW) so this parameter is studied in the experiments.

*BoG design.* The first steps (computation of descriptors densely distributed in each image and creation of the visual vocabulary) are similar for BoW and BoG-based methods. For BoG, we create a square grid based on the grid of visual words by connecting each node to its 4 neighbors (except on the border of the image). In our experiments, grid nodes are labeled by visual words and edges remain unlabeled although the algorithm is generic and could include labels on edges. For efficiency reasons, we preprocess grids before the mining step to remove all nodes which are surrounded by nodes with identical labels (thus creating holes in place of regions with uniform labels). Once grids are created for all images, we mine frequent subgrid patterns with GRIMA class by class. Finally, we consider the union of all frequent patterns and all visual words, and represent each image by an histogram of frequency of patterns and words. These histograms are given as input to the classifier.

*Efficiency Results.* Figure 3 compares scale-up properties of GRIMA, node-induced-GRIMA, GSPAN and PLAGRAM for the *Flowers* dataset (average results on 10 folds for each class, where each fold contains 72 images). We have set the number  $K$  of words to 100, and we have increased the absolute frequency threshold from 45 (62.5%) to 72 (100%). Note that, as explained in Sect. 2, the



**Fig. 3.** Number of patterns (left) and time needed (right) with respect to different support thresholds  $\sigma$  to compute all patterns for a grid, on average for all classes of *Oxford-Flowers17* dataset for GSPAN, PLAGRAM, GRiMA and node-induced-GRiMA. Time is limited to 1 hour per class.

face-based expansion strategy of PLAGRAM does not allow it to find patterns with no face (such as trees, for example). To allow PLAGRAM to mine the same patterns as its competitors, each node of the original grid graph is replaced by a four-node face (as illustrated in Fig. 1). This way, each frequent subgrid in the original grid is found by PLAGRAM in the expanded grid. However, some patterns found by PLAGRAM do not correspond to patterns in the original grid (e.g., faces in the expanded grid which correspond to edges without their end nodes in the original grid). For this reason, the number of patterns that are mined by PLAGRAM and its computation time are higher than the ones reported for its competitors. GSPAN does not consider edge angles when mining subgraphs, and two isomorphic subgraphs may have different edge angles so that they do not correspond to isomorphic subgrids, as illustrated in Fig. 1. Therefore, GSPAN and GRiMA compute different sets of frequent patterns. However, Fig. 3 shows us that the number of patterns is rather similar (slightly higher for GSPAN than for GRiMA). GRiMA and node-induced-GRiMA scale better than GSPAN, and are able to find all frequent patterns in less than 10s when  $\sigma = 60$  whereas GSPAN has not completed its execution after 3600s. The number of patterns found by both PLAGRAM and GSPAN for a support lower than 65 are thus not relevant and are not shown in the Figure. As explained in Sect. 3, node-induced-GRiMA mines less patterns and is more efficient than GRiMA, and much more efficient than GSPAN and PLAGRAM.

*N-ary classification results.* Let us now evaluate the interest of using grids for image classification on our three datasets. For each dataset we consider different values for the number  $K$  of visual words, with  $K \in \{100, 500, 1000, 2000, 4000\}$ . Note that this corresponds to changing the  $K$  of K-means.

GRiMA has one parameter,  $\sigma$ , and Fig. 3 shows us that the number of mined patterns grows exponentially when decreasing  $\sigma$ . CPU time is closely related to the number of mined patterns, and  $\sigma$  should be set so that we have a good tradeoff between the time needed to mine patterns and the gain in accuracy. The number of mined patterns (and the CPU time) depends on  $\sigma$ , but also on



K	15 Scenes				Flowers				Caltech-26			
	BoW	BoG	#pat.	$\sigma$	BoW	BoG	#pat.	$\sigma$	BoW	BoG	#pat.	$\sigma$
100	70.7%	70.0%	4,190	60%	48.0%	63.3%	5,284	80%	71.7%	73.7%	6,473	80%
500	73.7%	72.0%	4,027	40%	59.6%	63.9%	4,377	55%	76.3%	75.4%	4,240	60%
1000	73.8%	73.1%	4,877	30%	63.7%	64.6%	4,545	45%	77.3%	76.9%	6,044	45%
2000	73.2%	73.8%	6,128	20%	67.5%	66.9%	4,640	35%	77.0%	77.2%	4,502	40%
4000	74.2%	75.0%	4,218	20%	66.9%	67.0%	5,822	25%	77.2%	75.7%	16,345	35%

**Table 1.** Classification results according to the number of visual words  $K$  and the datasets. For BOG, the column #pat displays the number of frequent patterns extracted by node-induced-GRIMA and  $\sigma$  give the frequency freshold (in percentage of the number of images per class) reached to obtain at least 4000 patterns.

the size  $K$  of the vocabulary and on the dataset. Therefore, instead of fixing  $\sigma$ , we propose to fix the number  $p$  of patterns we want to mine: Starting with  $\sigma = 100\%$ , we iteratively decrease  $\sigma$  and run node-induced-GRIMA for each value of  $\sigma$  until the number of mined patterns reaches  $p$ . In our experiments, we have set  $p$  to 4,000 as this value corresponds to a good tradeoff between time and gain in accuracy.

For the classification task, we use  $N(N - 1)/2$  binary Support Vector Machines (SVM), where  $N$  is the number of classes. Each SVM is trained to discriminate one class against another. The final classification is obtained using a vote among the classes. SVMs are trained using Libsvm [3] with the intersection kernel presented in [?]. Inputs of SVMs are either BoW histograms (for the unstructured visual word-based method) or  $\text{BoW} \cup \text{BoG}$  histograms (for the grid-based method). The parameter  $C$  of SVM has been optimized by cross validation on all datasets. We consider 10-fold cross-validation for *Flowers* and *Caltech-26* since the number of images is the same for each class. For *15-Scenes*, we have created 10 different folds: For each fold, we randomly select 100 images per class for training and 50 images per class for testing. We provide a Student two-tailed t-test to statistically compare both accuracies (BoW and  $\text{BoW} \cup \text{BoG}$ ) for each dataset (the results in green are statistically better, in red statistically worse at  $p = 0.05$  level).

We report in Table 1 accuracy results obtained for all datasets. As expected, we need to lower the support threshold  $\sigma$  to obtain at least 4000 patterns when the number of visual words increases (since there are more different labels in the grid, the patterns are less frequent). On *15-Scenes*, the use of structural information becomes important only for  $K = 4000$  (but this also gives the best accuracy results). On *Flowers* on the contrary, BOG greatly improves the results for low number of visual words ([100,1000]) but it is not statistically better (nor worse) for higher numbers ([2000,4000]) where the accuracy is the best ( $K = 2000$ ). For *Caltech-26*, BOG slightly improves the results when  $K = 100$ , and it is not significantly different for higher values of  $K$ . Overall, those results show that adding structural information does not harm and sometimes improve the representation of images and that our grid mining algorithm can be used in a real-life context.

	BOW	BOG											
		75%	70%	65%	60%	55%	50%	45%	40%	35%	30%	25%	20%
suburb	99.1	99.1	99.1	99.1	99.1	<b>99.2</b>	<b>99.2</b>	<b>99.2</b>	<b>99.2</b>	<b>99.3</b>	<b>99.3</b>	<b>99.3</b>	<b>99.3</b>
coast	97.4	97.4	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	<b>97.5</b>	97.2	97.1	97.2
forest	98.8	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	<b>98.9</b>	98.8	98.7
highway	97.5	<b>97.7</b>	<b>97.7</b>	<b>97.6</b>	<b>97.7</b>	<b>97.6</b>	97.5	97.5	97.4	97.3	97.3	<b>97.2</b>	<b>97.1</b>
moutain	97.7	97.6	97.6	97.6	97.7	97.6	97.6	97.5	97.5	<b>97.5</b>	<b>97.5</b>	97.4	<b>97.3</b>
street	97.8	<b>97.9</b>	97.8	97.8	97.7	97.7	97.8	97.7	97.7	<b>97.5</b>	<b>97.3</b>	<b>97.1</b>	<b>96.8</b>
industrial	94.6	<b>94.7</b>	<b>94.9</b>	<b>94.8</b>	<b>95.0</b>	<b>95.1</b>	<b>95.1</b>	<b>95.1</b>	<b>95.2</b>	<b>95.3</b>	<b>95.6</b>	<b>95.4</b>	<b>95.4</b>

**Table 2.** Accuracy of each binary SVM for 7 classes of the *15-scenes* dataset with a vocabulary of  $K = 1000$  visual words and a threshold  $\sigma \in [75\%, 20\%]$ . For each class, we highlight in bold BOG results that are better than BOW, and in green (resp. red) BOG results that are significantly better (resp. worse) than BOW. For the other 8 classes, BOW and BOG are never significantly different.

*Binary classification results.* Finally, we evaluate the interest of using grids for a binary classification task on the *15-scenes* dataset (similar results were observed for the two other datasets). The goal is to provide an insight into each class separately, to see if all classes benefit from the use of grids in a same way. Table 2 reports binary classification results: For each class  $C$ , we train a binary SVM to recognize images of  $C$ , against images of all other classes. We consider the same parameter settings as for the  $N$ -ary classification task described previously, except that we train  $N$  binary SVMs (instead of  $N(N - 1)/2$ ). Only the 7 classes with statistically significant differences are shown in the table. We can see that some classes really benefit from the use of structured patterns for almost all frequency tresholds (*e.g.*, *industrial*, *coast*, *suburb*, *forest*) whereas for some classes, using unstructured information gives better results (*e.g.*, *street* or *moutain*). This is due to the fact that for some classes, the structure is too similar from this class to another to use it to discriminate classes.

## 5 Conclusion

We have presented GRIMA, a grid mining algorithm for Bag-of-Grid-based classification. GRIMA takes into account relative angle information between nodes in the grid graph (as well as possible labels on the nodes and the edges) and is designed to scale to large amount of data. We believe that the grid structure can not only be useful for directly modeling some real life problems but can also be used when an underlying high level structure is not obvious to define as it is the case for image classification problems. Experiments show that our algorithm is necessary (compared to existing ones) to efficiently tackle real life problems. They also show on three image datasets that patterns extracted on this grid structure can improve the classification accuracies. However, to further increase the discriminative power of the grid-patterns for image classification we would need to combine at the same time state-of-the-art deep-learned descriptors and design smart post-processing steps as the ones developed in [9] for unstructured

models. Besides, we plan to upgrade the GRIMA algorithm to mine 2D+t grids which is necessary to tackle different real-life applications such as the analysis of ecosystems.

## Acknowledgement

This work has been supported by the ANR project SoLStiCe (ANR-13-BS02-0002-01).

## References

1. Hiroki Arimura, Takeaki Uno, and Shinichi Shimozone. Time and space efficient discovery of maximal geometric graphs. In *International Conference on Discovery Science*, pages 42–55, 2007.
2. F. Brandao Da Silva, S. Goldenstein, S. Tabbone, and R. Da Silva Torres. Image classification based on bag of visual graphs. In *IEEE SPS*, pages 4312–4316, 2013.
3. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
4. K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, pages 76.1–76.12, 2011.
5. K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.
6. Diane Cook and Lawrence Holder. *Mining Graph Data*. J. Wiley & Sons, 2006.
7. G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV*, pages 1–22, 2004.
8. L. Fei-Fei, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*, 2004.
9. B. Fernando, É. Fromont, and T. Tuytelaars. Mining mid-level features for image classification. *IJCV*, 108(3):186–203, 2014.
10. P. Hogeweg. Cellular automata as a paradigm for ecological modelling. *Applied Mathematics and Computation*, 27:81–100, 1988.
11. C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *KER*, 28:75–105, 2013.
12. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.
13. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
14. R. Marinescu-Ghemeci. Maximum induced matchings in grids. In *Optimization Theory, Decision Making, and Operations Research Applications*, pages 177–187. Springer, 2013.
15. M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729, 2008.
16. E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, volume 3954, pages 490–503, 2006.

17. S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. Bakir. Weighted substructure mining for image analysis. In *IEEE CVPR*, pages 1–8, 2007.
18. F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing*, 14(2):169–180, 2005.
19. B. Ozdemir and S. Aksoy. Image classification using subgraph histogram representation. In *ICPR*, pages 1112–1115, 2010.
20. A. Prado, B. Jeudy, E. Fromont, and F. Diot. Mining spatiotemporal patterns in dynamic plane graphs. *IDA*, 17:71–92, 2013.
21. É. Samuel, C. de la Higuera, and J.-C. Janodet. Extracting plane graphs from images. In *SSPR*, pages 233–243, 2010.
22. F. Brandão Silva, S. Tabbone, and R. Torres. Bog: A new approach for graph matching. In *ICPR*, pages 82–87, 2014.
23. W. Voravuthikunchai, B. Crémilleux, and F. Jurie. Histograms of Pattern Sets for Image Classification and Object Recognition. In *CVPR*, pages 1–8, 2014.
24. X. Yan and J. Han. gSpan: graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.