Détection de motifs graphiques dans des images de documents anciens

Sovann EN, Caroline PETITJEAN, Stéphane NICOLAS, Laurent HEUTTE

Université de Rouen, Laboratoire LITIS EA 4108
76821 Saint-Etienne-du-Rouvray, France
{sovann.en2,caroline.petitjean,stephane.nicolas,laurent.heutte}@univ-rouen.fr

Résumé – La détection de motifs graphiques consiste à rechercher dans une collection d'images de documents, les occurences les plus similaires à une requête image. Dans cet article, nous proposons un système non supervisé pour la détection de motifs, sans besoin de segmentation préalable, en nous inspirant de techniques récentes en vision par ordinateur. Notre approche s'appuie sur une décomposition des documents en fenêtres de tailles variées et une description de ces fenêtres par sac de mots visuels, le tout hors-ligne afin de diminuer le temps de calcul. Une technique de compression des données, proposée tout récemment en recherche d'images, permet de maintenir une quantité de mémoire raisonnable, mais nécessite d'approximer le calcul de distance à la requête. De premiers résultats encourageants sont obtenus sur la base de documents DocExplore, une base de documents médiévaux.

Abstract – Pattern spotting consists of retrieving the most similar graphical patterns from a collection of document images. Inspired by the recent advances in computer vision and word spotting techniques, we propose in this paper an unsupervised, segmentation-free pattern spotting system. Overall, the system includes a powerful patch-based framework, the bag of visual word model with an offline sliding window mechanism to avoid heavy computational burden during the retrieval process. Our system takes advantage of the most recent powerful compression and distance approximation techniques (product quantization and asymmetric distance computation) to efficiently index the great number of sub-windows produced by sliding windows and allows to retrieve small sized queries in a large indexed corpus.

1 Introduction

Avec la numérisation des documents historiques, le besoin d'outils d'indexation et de recherche par le contenu se fait plus pressant. De premiers outils de recherche de mots (*word spotting*) dans des images de documents manuscrits ont déjà été développés [1], d'abord avec des approches utilisant une segmentation explicite en mots au préalable [2, 3], puis plus récemment sans segmentation [4, 5, 6]. Un manuscrit recèle cependant d'autres éléments graphiques, qu'il peut être intéressant de détecter [4, 7, 8]. La détection de motifs graphiques (*pattern spotting*) consiste à rechercher dans une image de document, à partir d'une requête image, les occurences d'un objet graphique, i.e. un motif plus ou moins complexe tel qu'un symbole, une lettrine, un blason [8, 4] (voir la Fig. 1 pour des exemples de requêtes). La difficulté réside dans les possibles différences entre la requête et les occurences des objets dans le document, en termes de couleur, taille, contexte ou position dans le document. Contrairement à la détection d'objets, dans laquelle de multiples vues de l'objet sont disponibles pour l'apprentissage, aucune information sur le motif à détecter n'est connue a priori. L'utilisation de méthodes discriminantes pour la détection de motifs est donc limitée, d'autant plus que la sélection des exemples négatifs pour l'apprentissage n'est pas clairement définie. Nous allons privilégier une approche alternative au plus proche voisin, opérant en général par fenêtrage glissant, utilisée en recherche d'images par le contenu (CBIR).

Notons également que le motif étant en général de taille réduite par rapport à l'image, et pouvant se trouver n'importe où dans le manuscrit, la recherche doit être exhaustive, sur des fenêtres de tailles variées. Cependant l'utilisation d'une analyse par fenêtre est coûteuse en temps de calcul, même dans des cadres optimisés [9, 6]. Grâce aux techniques puissantes de compression récemment proposées dans [10], le grand nombre de fenêtres obtenues peut être réduit et stocké dans une quantité de RAM raisonnable.

Dans cet article, nous proposons une approche non supervisée sans segmentation, conçue pour la détection de motifs graphiques, prenant en compte les requêtes de petite taille. Notre méthode est une approche au plus proche voisin, avec en une procédure hors-ligne, qui prétraite les images pour supprimer le fond, extrait des fenêtres, les représente par des sacs de mots et compresse les données. La procédure en ligne calcule une distance approximative de la requête aux fenêtres candidates.

2 Etat de l'art

Il existe très peu de travaux en détection de motifs graphiques. Notre état de l'art s'inspire donc du word spotting et du CBIR. En word spotting, de nombreuses approches ont consisté à décrire la requête comme une séquence de caractéristiques de longueur variable et à utiliser le contexte environnant. Les méthodes utilisées, comme le Dynamic Time Warping (DTW) ou les modèles de Markov cachés (HMM) [11, 12, 8], sont limitées par leur coût computationnel et leur besoin d'apprentissage.

En CBIR, les approches au plus proche voisin classique-

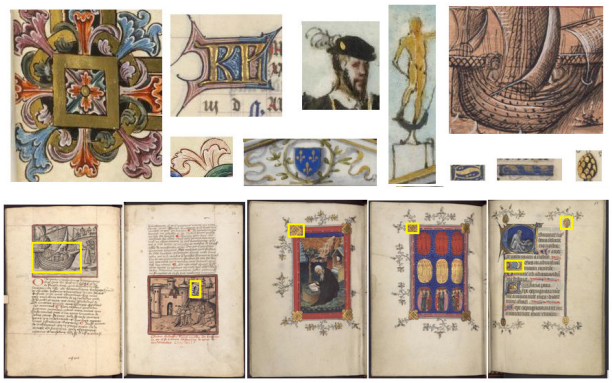

FIGURE 1 – Exemples de requêtes extraites de la base DocExplore (1ère ligne) et emplacement de requêtes dans des images de document (rectangle englobant jaune) (2ème ligne)

ment utilisées sont fondées sur le calcul de distance entre la requête et des fenêtres (obtenues par fenêtrage glissant), la distance pouvant être euclidienne ou le produit scalaire [5, 13, 6, 14]. Cependant, lorsque la base de documents est grande, la construction en ligne du vecteur de caractéristiques de chaque fenêtre devient infaisable. Certains auteurs ont tenté d'accélérer la recherche sur des fenêtres glissantes, en utilisant des critères permettant d'éliminer rapidement des fenêtres candidates ou en construisant des caractéristiques plus efficaces [4, 9, 6].

De récentes avancées en recherche d'images ont montré qu'une approche au plus proche voisin sur un très grand nombre de vecteurs (jusqu'à 100 milliards) pouvait être effectuée de manière efficace, i.e. en quelques secondes sur une machine standard avec un seul processeur [10]. Cela va permettre de remplacer le fenêtrage glissant en ligne classique, par un mécanisme hors-ligne, afin d'accélérer les temps de recherche et de faire une recherche plus efficace. Ces techniques sont utilisées dans [5], le travail récent le plus significatif en word spotting. Nous proposons ici l'adaptation de ces techniques d'indexation et de compression à la détection de motifs graphiques. Si ces deux tâches sont généralement considérées comme similaires, elles ont aussi des caractéristiques spécifiques qui doivent être prises en compte. En détection de motifs, la couleur et/ou l'échelle de niveaux de gris sont des caractéristiques importantes pour discriminer les objets, alors qu'en word spotting, les mots peuvent être distingués du fond par une simple distribution bimodale. Une autre caractéristique importante concerne la position, l'orientation et la taille des objets à détecter. En word spotting, ces informations sont connues a priori, alors qu'elles ne sont pas disponibles en détection de motifs graphiques. Par exemple, le même objet peut apparaître plus grand ou plus petit, et ce n'importe où dans la page. Les mots sont au contraire écrits horizontalement, avec une hauteur relativement constante, qui permet l'utilisation de fenêtres glissantes aux dimensions fixes. Les mots sont généralement faciles à isoler, alors que les motifs peuvent apparaître sur un fond texturé, ce qui les rend plus difficile à segmenter. Ainsi, le prétraitement et les caractéristiques utilisés en word spotting ne peuvent être appliqués directement à la détection de motifs graphiques.

Dans la suite nous présentons notre système de détection de motifs graphiques avec un prétraitement dédié (suppression du fond uniforme), extraction de fenêtres de tailles variées, représentation des fenêtres par une approche classique de sac de mots visuels, compression et codage de l'information hors-ligne par différentes techniques (LSI, PQ, IVF), et calcul de la distance entre requête et fenêtre par approximation. Dans la partie expérimentale, l'apport de chaque partie sera évalué.

3 Notre système de détection des motifs

Le système que nous proposons se présente comme illustré sur la Figure 2. Les différentes étapes de ce système sont détaillées ci-après :

Prétraitement : suppression du fond Idéalement, seules les zones graphiques (hors texte) devraient être retenues pour l'extraction des fenêtres, mais la séparation texte-graphique reste une tâche ardue [15]. Afin de diminuer quand même le nombre de fenêtres à traiter, nous avons opté pour une approche simple permettant d'éliminer rapidement le fond. Sur une image binaire seuillée adaptativement, une approche de type croissance de région aggrège les pixels par zone itérativement, en partant du centre, si la zone considérée a un nombre de pixels fond inférieur à 5%.

Extraction des fenêtres Le motif pouvant se trouver n'importe où dans l'image, l'extraction des fenêtres ne peut s'appuyer sur une segmentation ligne-texte, comme dans [5]. Les fenêtres sont donc extraites sur toutes les parties graphiques fournies par l'étape précédente et sont de tailles variables, afin de pouvoir convenir à des requêtes de tailles différentes.

Représentation des fenêtres et LSI La description par sacs de mots nécessite l'extraction dense de patches dans les fenêtres. Celles-ci sont divisées en une grille de patches 3×3 pixels, chaque patch étant décrit avec SIFT en 128 dimensions. Cette dimension est réduite à 64 avec une ACP, comme dans [10]. Un dictionnaire de K mots visuels est construit avec un K-moyennes. Un histogramme de dimension K indiquant les occurences des mots visuels les plus proches permet de caractériser la fenêtre. Nous utilisons ensuite un schéma de pondération Term Frequency-Inverse Document Frequency (Tf.idf) pour donner plus d'importance aux mots visuels fréquents. Enfin, nous employons une technique (LSI pour Latent semantic indexing) qui permet de trouver une structure sémantique aux mots visuels, en s'appuyant sur une décomposition en valeur singulière de la matrice contenant tous les descripteurs, et une troncature de cette décomposition, pour arriver à une description en dimension réduite (notée D dans la suite) des diférentes fenêtres. Pour plus de détails, nous référons à [5].

Compression et calcul de distance (PQ, ADC, IVF). Les deux techniques de compression (PQ pour Product Quantization) et de calcul de distance approximé (ADC pour Asymmetric Distance Computation) ont montré leur efficacité en recherche d'images [10]. Nous les utilisons ici pour la détection

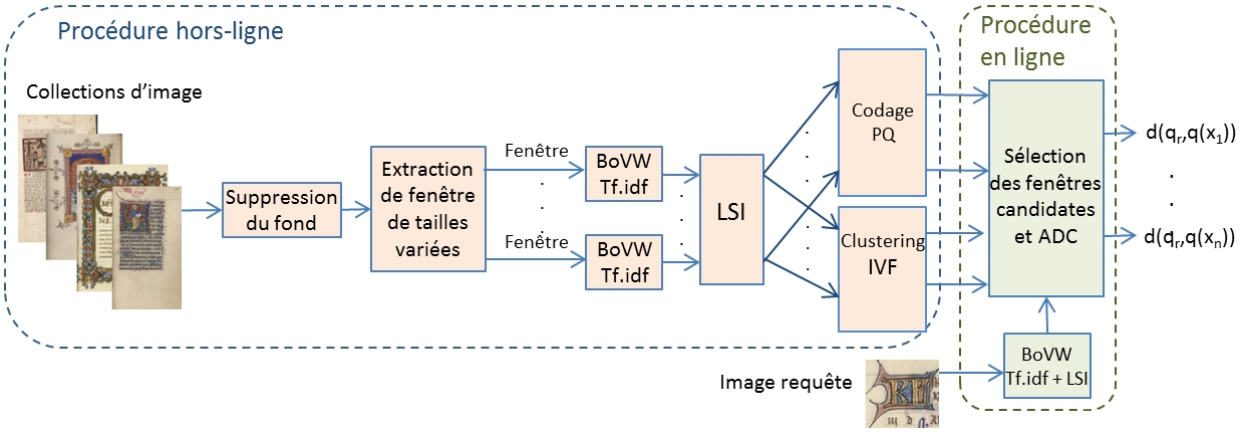

FIGURE 2 – Notre système de détection de motifs graphiques. BoVW : sac de mots visuels, PQ : product quantization, IVF : inverted file structure, ADC : asymmetric distance computation, LSI : latent semanting indexing.

de motifs. Soit $q_r \in \mathbb{R}^D$ le vecteur représentant la requête et $X = \{x_1, x_2, ...x_n\}$, $x_j \in \mathbb{R}^D, \forall j \in [1 \cdots n]$, l'ensemble des vecteurs représentant les fenêtres. La distance entre la requête q_r et $x_j \in X$ est calculée par ADC selon :

$$d(q_r, x_j) = d(q_r, q(x_j)) \quad (1)$$

où $q(x_j)$ est la version quantifiée de x_j, calculée hors ligne, et $d(\cdot)$ la distance euclidienne. La taille D du vecteur devrait être la plus grande possible, mais les vecteurs de grande dimension étant difficiles à indexer, les auteurs de [10] ont proposé une manière plus efficace de coder l'image. L'idée est de diviser le vecteur en m segments et pour chaque segment, un dictionnaire est construit. Ainsi, la distance approximée peut se réécrire :

$$d(q_r, q(x_i)) = \sqrt{\sum_j d(u_j(q_r), q_j(u_j(x_i)))^2} \quad (2)$$

où $j = 1..m$ et $u_j(q_r) \in \mathbb{R}^{D/m}$ correspond au j-ème segment du vecteur q_r ($m = 8$ ici comme recommandé dans [10]). Lors de la phase en ligne, le système calcule d'abord la distance entre les segments constituant la requête, et tous les clusters possibles des segments, et stocke toutes les distances dans une table de correspondance. Le calcul de distance entre la requête et les fenêtres candidates utilise ensuite cette table et somme les distances correspondant à chaque segment des candidats.

Nous employons également une technique (IVF pour Inverted file structure) qui permet de réduire le nombre de fenêtres candidates [10]. L'idée est d'appliquer un clustering préalable sur les données, et de réduire les données examinées à celles appartenant aux clusters de la requête. Notons que la requête pouvant apparaître en bordure de clusters, plusieurs clusters sont examinés.

4 Résultats expérimentaux

Notre système est testé sur la base de documents DocExplore [1] qui consiste en 1597 images de documents de la période médiévale. Nous avons annoté 1097 motifs graphiques, sur les conseils d'un historien spécialisé en manuscrits médiévaux, un motif pouvant apparaître 0, 1 ou plusieurs fois dans une image. Ces motifs correspondent à 32 objets graphiques de formes variées tels que des drapeaux, des lettrines, des séparateurs de texte (voir Fig. 1). Chacun des 1097 motifs est utilisé comme image requête. Les requêtes sont de petites tailles ; elles représentent en moyenne 2% de la taille de l'image. Leurs dimensions variant entre 20 et 160 pixels, la largeur et la hauteur des fenêtres extraites prennent les valeurs suivantes : {20, 40, 80, 160} pixels, avec un pas de 20 pixels.

La performance du système implémenté en Python est évaluée par la mesure Average Precision (mAP), qui est la surface sous la courbe rappel-précision. Sur notre base de 1597 images de documents, le système fournit 14.5M fenêtres. Dans la suite, nous testons différentes configurations du système, en faisant varier la taille du dictionnaire (K ={500, 1000, 5000, 10000}) et de l'espace LSI (D ={128, 256, 512}), comme le montre le Tableau 1.

La première ligne du tableau ($D = K$) correspond au système sans technique de compression (pas de LSI, PQ ni IVF). Le meilleur résultat est obtenu pour un dictionnaire de taille 1000. Notons que les performances diminuent si le dictionnaire est trop grand : il est possible que la petite taille des requêtes ne permette pas de retrouver des mots visuels significatifs.

Comme on pouvait s'y attendre, lorsque la dimension du sac de mots visuels est réduite grâce au LSI ($D = \{512, 256, 128\}$), les performances diminuent. La perte en précision augmente d'autant plus que l'espace original est de grande dimension ; comparons par exemple la perte quand $K = 500$ et $K = 10000$ sur une dimension $D = 128$: le mAP évolue de 0.189 à 0.174 dans le premier cas, alors qu'il diminue sévèrement de 0.171 à 0.124 dans le second cas.

Lorsque PQ et ADC sont introduits (lignes désignées par "PQ" dans le Tab. 1), le système voit ses performances diminuer, ce qui s'explique par l'utilisation de la compression, un processus avec perte. Remarquons la diminution des besoins

1. Les manuscrits proviennent du projet DocExplore : `http://www.docexplore.eu/`.

TABLE 1 – mAP obtenus avec différentes configurations : pas de compression ($D = K$), LSI seul (lignes avec D =XXX), LSI/PQ (lignes avec PQ), LSI/PQ/IVF (lignes avec IVFPQ). T (s) : temps en sec pour une requête. Mém : mémoire requise en Go. * : temps et mémoire donnés pour $K = 500$.

K	500	1000	5000	10000	T (s)	Mém
$D = K$	0.189	0.190	0.186	0.171	26.8*	29*
D=512	-	0.187	0.146	0.139	33.2	29.2
PQ	-	0.135	0.138	0.122	30.1	0.93
IVFPQ	-	0.137	0.140	0.125	3.4	0.93
D=256	0.188	0.180	0.147	0.138	14.3	14.6
PQ	0.140	0.143	0.135	0.129	13.6	0.46
IVFPQ	0.140	0.143	0.135	0.129	1.6	0.46
D=128	0.174	0.164	0.130	0.124	7.2	7.3
PQ	0.130	0.128	0.115	0.109	5.3	0.23
IVFPQ	0.130	0.127	0.115	0.109	0.8	0.23

en mémoire pour le stockage des 14.5M fenêtres : la quantité nécessaire est divisée par un facteur d'environ 30.

L'addition de l'étape d'IVF permet de diminuer le nombre de fenêtres à examiner. Nous avons testé différents nombres de clusters pour IVF (1024, 4096, 8192) ; nous reportons ici seulement les meilleurs résultats, obtenus avec 1024 clusters. Dans ce cas, le système doit examiner environ 70 000 fenêtres au lieu de 14.5M. Les résultats restent stables malgré l'introduction de l'IVF, tout en diminuant drastiquement le temps de détection.

5 Conclusion

Nous avons proposé une méthode non supervisée pour la détection de motifs graphiques dans des images de documents anciens, à la croisée de la recherche d'image par le contenu et de la détection de mots, et en utilisant des techniques de compression récentes. Les fenêtres sont décrites par sac de mots, mais des travaux sont actuellement en cours pour rechercher des descripteurs plus pertinents [16]. Bien que la phase de prétraitement ne soit pas le point principal de la méthode, le développement d'un outil de séparation texte/graphique pourra être d'une grande aide, afin de réduire le nombre de fenêtres à traiter et de supprimer des faux positifs.

Remerciements

Les auteurs remercient le Conseil Régional de Haute-Normandie (projet PlaIR2.0) pour son soutien financier, la Librairie Municipale de Rouen pour la mise à disposition des documents anciens, Pr. Elisabeth Lalou pour son expertise sur la recherche de motifs graphiques dans les manuscrits médiévaux, et Pr. Frédéric Jurie pour ses conseils.

Références

[1] R. Manmatha, C. Han, and E. M. Riseman, "Word Spotting : A New Approach to Indexing Handwriting," in *ICCVPR*, 1996, pp. 631–637.

[2] J. L. Rothfeder, S. Feng, and T. M. Rath, "Using corner feature correspondences to rank word images by similarity," in *CVPR Workshop*, vol. 3, 2003, pp. 30–38.

[3] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz, "Towards an omnilingual word retrieval system for ancient manuscripts," *Pattern Recognition*, vol. 42, no. 9, pp. 2089–2105, 2009.

[4] V. Dovgalecs, A. Burnett, P. Tranouez, S. Nicolas, and L. Heutte, "Spot it ! finding words and patterns in historical documents," in *ICDAR*, 2013, pp. 1039–1043.

[5] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, "Efficient segmentation-free keyword spotting in historical document collections," *Pattern Recognition*, pp. 545–555, 2015.

[6] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Efficient exemplar word spotting," *British Machine Vision Conference*, pp. 67.1–67.11, 2012.

[7] T. Rakthanmanon, Q. Zhu, and E. J. Keogh, "Mining historical documents for near-duplicate figures," in *International Conference on Data Mining*, 2011, pp. 557–566.

[8] Q. Zhu and E. Keogh, "Mother fugger : mining historical manuscripts with local color patches," in *International Conference on Data Mining*, 2010, pp. 699–708.

[9] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows : Object localization by efficient subwindow search," in *IEEE CVPR*, 2008, pp. 1–8.

[10] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans on PAMI*, vol. 34, no. 9, pp. 1704–1716, 2012.

[11] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *IEEE Trans on PAMI*, vol. 34, no. 2, pp. 211–224, 2012.

[12] J. A. Rodríguez and F. Perronnin, "A model-based sequence similarity with application to handwritten word spotting," *IEEE Trans on PAMI*, vol. 34, no. 11, pp. 2108–2120, 2012.

[13] P. Yarlagadda, A. Monroy, B. Carque, and B. Ommer, "Recognition and analysis of objects in medieval images," in *ICCV*, 2010, pp. 296–305.

[14] S. En, F. Jurie, S. Nicolas, C. Petitjean, and L. Heutte, "Linear discriminant analysis for zero-shot learning image retrieval," in *VISAPP*, 2015.

[15] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "Competition on Historical Newspaper Layout Analysis," in *ICDAR*, 2013, pp. 1454–1458.

[16] O. Kihl, D. Picard, and P.-H. Gosselin, "A unified framework for local visual descriptors evaluation," *Pattern Recognition*, vol. 48, no. 4, pp. 1174 – 1184, 2015.