Visual Graph Analysis for Quality Assessment of Manually Labelled Documents Image Database

Romain Giot, Romain Bourqui, Nicholas Journet, Anne Vialard

Visual Graph Analysis for Quality Assessment of Manually Labelled Documents Image Database

Romain Giot, Romain Bourqui, Nicholas Journet and Anne Vialard
Laboratoire Bordelais de Recherche en Informatique UMR 5800
Domaine universitaire, 351, cours de la Libération 33405 Talence - France
Email: {romain.giot, bourqui, journet, vialard}@labri.fr

***Abstract*—The context of this paper is the labelling of a document image database in an industrial process. Our work focuses on the quality assessment of a given labelled database. In most practical cases, a database is manually labelled by an operator who has to browse sequentially the images (presented as thumbnails) until the whole database is labelled. This task is very repetitive; moreover the filing plan defining the names and number of classes is often incomplete, which leads to many labelling errors. The question is then to certify if the quality of a labelled batch is good enough to globally accept it. Our objective is to ease and speed up that evaluation that needs up to 1.5 more times than the labelling work itself. We propose an interactive tool for visualizing the data as a graph. That graph enhances similarities between documents as well as the labelling quality. We define criteria on the graph that characterize the three types of errors an operator can do: an image is mislabelled, one class should be split in more pertinent subclasses, several classes should be merged in another. This allows us to focus the operator attention on potential errors. He can then count the errors encountered while auditing the database and assess (or not) the global labelling quality.**

I. Introduction

This work aims at helping companies that provide digitizing solutions. One of their tasks is to label a huge amount of document images (ID cards, forms, tickets, receipts, ...). A typical digitizing process consists in scanning physical documents and manually labelling each image in accordance with a filing plan. According to our industrial partner[1], a human operator can manually label between 300 to 500 documents in one hour. On average, a typical customer order represents between 5 and 6 weeks of manual labelling for one operator. There are three main difficulties related to this task. Firstly, this work is extremely repetitive and requires a high level of concentration to avoid mislabelling. Secondly, some images are hard to label. Two images that belong to two different classes can look very similar and have only few differences (a different name in a form, the size of a ticket, ...). The last difficulty is linked to the filing plan which is a document dedicated to the human operator where the number and the name of classes are defined. As a labelling campaign lasts several weeks, the filing plan can change (classification rules, creation or suppression of classes). All these difficulties result in mislabelling errors. There are three different kinds of possible errors. The first one corresponds to a "misclassification error": a document image is classified in class A instead of class B. The two other errors are linked to changes in the filing plan: the "merge error" which corresponds to the addition of new classes in the filing plan (some images that were classified in class A need now to be classified in the classes A_1, A_2 and A_3) and the "split error" happens when several classes are removed from the filing plan (the classes A_1, A_2 and A_3 need to be merged in a unique class A). As the filing plan can change many times in a labelling campaign, split and merge errors can occur very often.

Digitizing companies guarantee a minimum amount of labelling errors to their customers. Quality assessment of labelled document images is provided by doing a manual audit. Generally, during the 2 or 3 first weeks of the labelling campaign, 100% of the already classified database is audited. In the next weeks, the percentage of audited images usually drops to 10% of the database. Statistics made on many productions show that an audit of 100% of a database needs 1.5 more times than the labelling process itself. This audit process is currently done through a very basic human-machine interface: for each class, images are sequentially presented to the operator who points out the labelling errors. We have previously tested this kind of visual interface in a document image classification process. Experiments we carried out in [1] clearly highlighted that dealing with a huge amount of documents and classes is really hard to handle in that way. That's why we propose to use information visualization techniques in order to help the operator.

Information Visualization exploits human visual capabilities to support visual exploration and analysis [2] and tackles the problem posed by the abundance of information [3]. Schneiderman [4] provides some recommendations for the visual exploration of data, which are now known as the Visual Information-Seeking Mantra: "overview first, zoom and filter, then details-on-demand". Providing an overview enables the operator to identify the main trends in the data and therefore to guide his exploration and to focus his attention on interesting parts. Zooming and filtering are basic interaction techniques in information visualization allowing to reduce the amount of displayed elements and thereby to reduce the operator's cognitive load. Last but not least, details-on-demand stands for techniques providing detailed information about few elements of the data when and only when requested by the operator.

In this article we propose to use the graph based information visualization framework Tulip [5] to create a visual interactive tool. The originality of this work stands in the visual representation of the document database as a multilevel graph visualisation: a first level represents the similarities between classes while a second one represents the similarities between documents. By adding a business logic linked to the audit process we provide a tool that allows a smart data browsing

[1]We would like to thanks www.gestform.com for supporting this work

to quickly detect labelling errors. An operator needs to have visual feedback about similarities between documents or classes and about interclass distances or intraclass distances to audit a document images database. Moreover, by focusing his attention on the part of the graph corresponding to the three kinds of typical errors (misclassification, split errors, merge errors), we assume that the whole auditing process will be easier and quicker.

Section II presents previous work about image database browsing. Section III details how we use the visual algorithms available in Tulip [5] to create a visual interface dedicated to auditing process. Section IV details 3 measures specially created to identify, in a graph of documents, the tree main errors that a human operator can do during a labelling campaign. Section V evaluates the proposal and section VI concludes this paper.

II. Previous Work

The research area closest to our work is the browsing of image databases. As far as we know, no specific proposition has been published on the problem of document image database visualization and browsing. In most cases image browsing solutions have been proposed for content-based image retrieval purposes.

The authors of [6] list three main classes of visualization methods in a survey on this subject. The first class corresponds to mapping-based visualizations: similarities between images are computed in a high-dimensional feature space and are preserved as well as possible in a 2D projection where images are displayed to the operator. The second class groups clustering-based visualizations methods. Clustering becomes necessary when the size of the database increases. Clusters of images are defined according to image feature vectors or according to image metadata and only one representative image of each cluster is displayed to the operator. The last class is composed of graph-based visualization methods. Generally, the nodes of the graph are images and edges between images are created according to their similarities. Among the ways to generate the final visualization from such a graph, one can find various mass spring algorithms. Standard browsing tools are then provided to explore the image database visualization: panning, zooming, vertical exploration in hierarchical visualizations,.. A specific way of browsing can be defined according to the application requirement. For example, in image retrieval applications, the browsing can be driven by relevance feedback criteria. As an example of a generic tool for interactive visualization and analysis of image databases, we can cite PEx-Image [7]. It integrates complementary functionalities: various features computations, feature selection, various 2D projections including distance-based projection and similarity trees. Several views of the same data can be coordinated. Many use-cases are described in the article: comparison between the relevance of two features sets for a given labelled dataset, classification task guided by image similarities, integration of textual information associated with images. In terms of data size, PEx-Image can handle up to 9000 images.

The authors of [8] present a framework combining multimedia analysis and advanced visualization to facilitate image retrieval in the domain of digital forensics. We can place our work in the same research topic, which is called *Multimedia Analytics*. We generate a visualization of a document image database which combines a clustering based on metadata (the image labels) and a graph based visualization. The proposed framework allows the operator to efficiently detect labelling errors in the database.

III. Graph Construction and visualization Tools

We want to abstract our image dataset as a graph and display it. The advantage of using a graph instead of a sequential images visual interface is the ability to show the proximity relation between documents through edges. In this section, we detail how the graph is constructed.

a) Features Extraction: For each image, we compute two kinds of features. Firstly, we apply an OCR on the whole database. It allows the computation of a global histogram of the occurrences of the 500 most frequent words extracted from the database (from one digitization campaign to another, the content of the histogram is different). A basic text mining algorithm (stop word and lemmatization) is used as preprocessing for listing only pertinent words. Secondly, we compute image features. We decided to characterize an image with the same features that we used in [1]: an image is divided into 12 areas of equivalent size; for each part, the average grayscale of the pixels is computed; the height and width of the image are also kept as features. The choice of using OCR results and very basic image features is motivated by the fact that we try to reproduce the cognitive mechanism of an operator. Most of the time, he labels an image from its layout after viewing a thumbnail or by identifying 2 or 3 words in the document.

After features extraction, each document can be described either by a vector of size 500 (text features), 14 (image features), or 514 (fusion of the 2 vectors).

b) Graph Construction: Let us denote by $G^f = (V, E^f)$ the graph representing a dataset depending on the extracted features f. Each element of the node set $V = (v_i)_{i=1...n_v}$ corresponds to a document. $E^f = (e_i)_{i=1...n_e}$ is the set of edges and corresponds to an oriented similarity between two documents. There is an edge $e_i^f = (v_m, v_n)$ if the document v_n is within the k closest neighbors of v_m according to the features f. Let us denote by $\mathcal{L} = (l_i)_{i=1...n_l}$ the set of classes labels (*e.g.*, the type of document). The application $C : V \to \mathcal{L}$ provides the label of each document. A metagraph $G_M(G^f, C) = (V_M, E_M)$ is constructed from the graph G^f. Each node v_{M_i} of the metagraph represents a class of documents: $v_{M_i} = \{v_j / C(v_j) = l_i\}$. There is an oriented edge (metaedge) between two metanodes (classes) if at least one node (document) of the source class has one of its k-nearest neighbor in the target class: $E_M = \{(v_{M_i}, v_{M_j}) / \exists (v_n, v_m) \in E^f, C(v_n) = l_i, C(v_m) = l_j\}$.

c) Graph Quality: We want to assess the quality of the labelling. It can be done with measures able to quantify the quality of a graph partitioning by analysing the meta-graph. Theses quality measures are adapted from [9] because of the k-nn structure of the graph. We evaluate the internal cohesion of a metanode (document class) by comparing its edge density to the maximal edge density it could have. Let us denote by $E^f(v_{M_i}, v_{M_j}) \subset E^f$ the set of edges linking one node of class l_i to one node of class l_j, and by $V(v_{M_i}) \subset V$ the set of nodes of class l_i (*i.e.* the nodes represented by v_{M_i}). The internal

cohesion of a metanode (*i.e.*, a class) is then defined as: $IC(v_{M_i}) = \frac{|E^f(v_{M_i},v_{M_i})|}{|V(v_{M_i})|*min(k,|V(v_{M_i})|-1)}$. Note that the minimum stands for cases where a metanode contains less than k nodes. The same principle is used to evaluate the cohesion between two classes of documents. The external cohesion is defined on a meta edge as: $EC(e_{M_{ij}}) = \frac{|E^f(v_{M_i},v_{M_j})|}{|V(v_{M_i})|*min(k,|V(v_{M_j})|)}$ with $e_{M_{ij}} = (v_{M_i}, v_{M_j})$. In the best theoretical case, all the IC values are equal to 1 and all the EC values are equal to 0 meaning that all the k nearest neighbors of a document are in the same class. In other words, it means that the manual annotation is consistent with the computed features.

From these local quality metrics, we are able to compute a global quality metric (higher is better):

$$Q^f = \frac{\sum_{v_{M_i}} IC^f(v_{M_i})}{|\mathcal{L}|} - \frac{\sum_{e_{M_{ij}}} EC^f(e_{M_{ij}})}{\sum_{v_{M_i}} min\big(|\mathcal{L}|-1,|V(v_{M_i})|*k\big)}$$

d) Interaction for Database Understanding: This section describes the exploration tool that we have designed to help experts to identify labelling errors (Figure 1 shows a screenshot of that software).

It is necessary to display the computed meta-graph on screen. Each node is represented by a thumbnail of the document it represents and each edge is represented as a straight line. First step when building a visualization of a graph is to assign coordinates to its nodes and/or edges using a graph drawing algorithm. While many are dedicated to particular classes of graphs (*e.g.* planar graphs, trees, or hierarchical graphs), the most popular approach to draw general graphs is the force-directed one as it provides visually pleasant and structurally significant results. Such algorithms use a physical analogy to lay the nodes out by considering each node as a physical object and each edge as a spring. In the resulting layout, close nodes (in term of graph distance) should therefore be laid out close in the representation. A force model allows then, through several iterations, to obtain a local minimum energy level. We use the FM^3 algorithm [10] that provides a good compromise between computation time and aesthetics. A good visualization of graph should also avoid node-node overlaps. Even if FM^3 can take node sizes into account to prevent such overlaps, it does not provide any guarantee. To remove the remaining overlaps, we use as a post-process the algorithm *Fast Overlap Removal* (FOR) [11] that removes node-node overlaps while minimizing the total amount of displacements. In our case, we need to draw the meta-level as well as the classes of documents. In order to take into account the area needed to draw each class during the layout of the meta-level, we use a bottom-up approach. The induced graph (nodes of the same class) represented by each metanode is therefore drawn first using FM^3 and FOR. The layout bounding box of each class is then used to set the size of the corresponding metanode. Finally the meta-level is drawn with the same combination of algorithms.

To guide the operator in its exploration process, we also render the internal/external cohesions (see III.c) with colors. As these two measures are both bounded between 0 and 1, we use a linear color mapping from red to green where green is always associated to a good cohesion and red to a bad one. Concerning the metanodes, the higher is the internal cohesion, the higher the level of confidence in the labelling of the corresponding documents is. On the contrary, a high external cohesion indicates that the corresponding classes are linked by a large number of edges. The operator should therefore check whether the corresponding classes should be merged. To highlight even more high external cohesions, the width of the edges of G_M are also mapped to the external cohesion.

As it was previously mentioned, the graph can be generated according to several configurations of features. The operator interface provides a tool that allows to select manually the features to use and then change the visual representation of the database. Note that we can also generate a graph based on the fusion of the graph based on OCR features and the graph based on image features.

All these general interaction tools are useful for visualizing a big clustered image database. However, to facilitate the labelling audit, the operator needs to be further guided. We propose to focus his attention on potential labelling/split/merge errors.

Our software also provides a *zoom and pan* (in order to help the operator to understand the recommendations when clicking on them) interaction tool as well as an interaction tool to emphasize the neighborhood of a given node. When clicking on a node, we build a metagraph as defined in III.b by only considering the focus document and its neighborhood. Again, a color mapping (resp. color and width mappings) is applied on the metanodes (resp. the edges linking metanodes). The left panel of the software displays detailed information about the focused document: a bigger picture of it, a description of it, and a thumbnail of its neighbors ordered by their own class.

IV. Three measures for identifying potential labelling errors and improve the operator experience

In section I, we have listed three kinds of recurrent errors occurring when labelling a dataset. In order to correct an error, it is necessary to apply the inverse operation (*i.e.* a merge for a split error or a split for a merge error). The operator interface integrates an error suggestion module where potential errors and their proposed corrections are listed. It is up to the operator to accept or reject each proposition. If the manual labelling is effectively erroneous, the operator can validate the proposition and an error counter is incremented. As it would be confusing for the operator to change the topology of the graph during the audit process, we do not correct and modify the graph after each error assessment. With the global view of the errors encountered, the operator can finally accept or reject the labelling work whenever he wants.

a) Mis-labelling errors: For each class (metanode), we propose a list of documents (nodes) which could be moved in another class. The main idea is to identify nodes that are highly connected to a set of nodes belonging to another class. We verify all the possible moves of documents from the class of interest to the classes where it is linked and keep the modification which provides the best final quality value if we apply it. The complexity depends on the number of inter edges. The operator can assess or not the error and its proposition of correction. If he does, the number of errors is incremented. Then the second most isolated node of class l_n is processed in the same way, and so on until the operator considers there is no more labelling

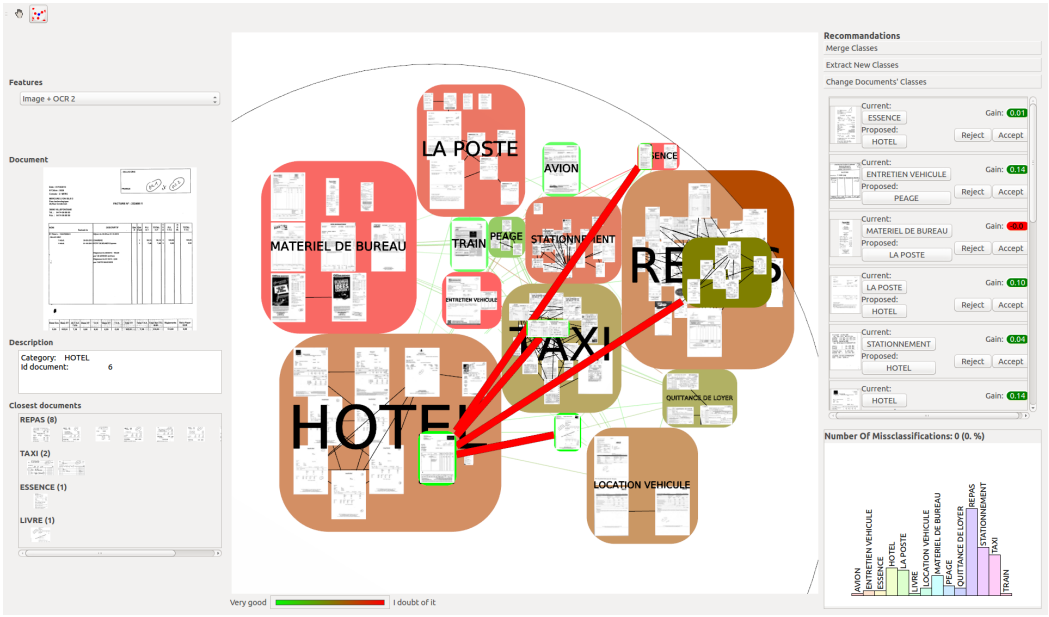


Fig. 1: Screenshot of our software. The middle panel shows the metagraph as defined in III.b. While nodes correspond to documents and edges link similar documents, metanodes contain all nodes with identical labels and metaedges link metanodes containing linked documents. Metanode color (resp. metaedge color) indicates its internal cohesion (resp. external cohesion). When pointing a document, the tool also displays the metagraph corresponding to the direct neighborhood of that document.

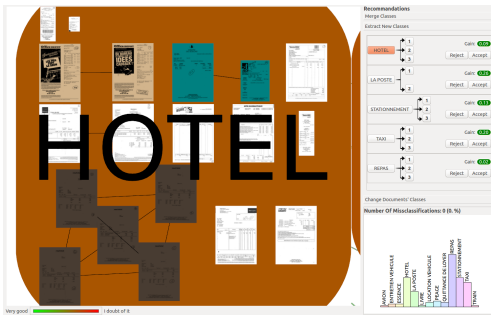
errors in the class l_n. The number of counted errors is the total number of errors assessed by the operator.

b) Merge Error: In case of a merge error, the correction must propose to extract from a metanode some nodes in other new classes (*ie*: split a metanode). Given an existing class l_n, we can define SG_n^f the induced subgraph of G^f where all its nodes are of class l_n. We compute a partitioning of this subgraph using the Markov clustering algorithm MCL [12]. MCL algorithm is able to extract compact clusters in a graph. Let's denote by MCL_n^f the list of clusters of nodes computed by the MCL algorithm on the graph SG_n^f.We can consider that each cluster corresponds to a new class. We exclude clusters composed of only one node (considered as noise). Considering that MCL often split a metanode into a lot of clusters of different sizes, we propose to keep only the three biggest of them. We assume that proposing no more than three splits is enough for confirming (or not) the merge error that we have detected.The operator can then take the decision to create three new classes from this clustering result. If the operator accept the split suggestion of a class n, the number of errors associated to this kind of error is incremented by the number of nodes removed from the original class.

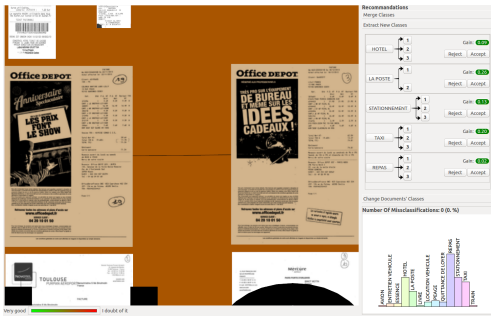
c) Split Error: We consider that the correction of a split error consists in merging two existing classes into another one. If there is an important number of edges in G^f linking nodes of one class to another, the two classes are likely to be merged. Thus, for each metanode, we propose another metanode with which it could be merged. Given a source metanode v_{M_m}, the target metanode is found as follows: $\arg\max_n |\{(v_i, v_j) \in E^f / C(v_i) = l_m, C(v_j) = l_n, m \neq n\}|$. In practice, if the proposed merging is pertinent, the operator validates it. The number of counted errors is incremented by the number of nodes of v_{M_m}.

V. Evaluation

a) Visual results: As mentioned in [8] about multimedia analytics, it is hard to evaluate such kind of software because "there are so many factors influencing the results". The tests we proceeded on documents coming from industrial digitization campaigns show that the database total size is not the only issue we have to deal with for providing an easy visualization. Classical problems of document image database visualization are partially handled in our software. The visualization is correct up to 300 documents in a class (no image overlapping, fast processing-time for displaying). However, some specific factors can limit the operator interactions during the audit process. Obviously, the operator efficiency depends on how well he knows the database content and on the size of each document class. Thanks to our proposition we can overcome this last issue for identifying mislabelling, merge and split errors. Counting a mislabelling error is made easy by presenting to the operator, at the same time, only one document image and its supposed correct label (cf figure 1, right part). In the same way, split error identification is made easy by avoiding a sequential browsing of each metanode. By alternatively clicking on the two proposed label, the operator can visualize the two classes easily and decide if he has to merge them or not. At least, the features for identifying merge errors really help for quickly identifying if a metanodes has to be split in three new metanodes. As it is illustrated on figure 2.a-b by successively clicking on buttons "1/2/3" near the class label a camera movement is carried out (zoom and pan). It allows a very simple comparison of each subgraph extracted after the application of MCL algorithm.



(a)



(b)

Fig. 2: An example of merge error. In that case, MCL algorithm proposed to split the "Hotel" class (a) into 3 main sub-classes (colored in the picture); one of these sub-classes corresponds to "Office Furniture" documents (b).

TABLE I: Percentage (mean/standard deviation) of wrong suggestions before a good re-labelling proposition. Feature used: f_1=ocr+image, f_2=ocr , f_3=image

		$k=10$			$k=20$		
		f_1	f_2	f_3	f_1	f_2	f_3
DB 100	μ	0.24	0.20	0.28	0.22	0.23	0.23
	σ	0.87	0.24	0.94	0.84	0.42	0.94
DB 3000	μ	0.30	0.24	0.34	0.28	0.26	0.47
	σ	0.23	0.17	0.27	0.28	0.27	0.28

TABLE II: Computational performance (in seconds) of the applications for three different databases

$\|V\|$	$\|\mathcal{L}\|$	Drawing	Quality	Split sugg.	Move sugg.	Merge sugg.
115	14	0.032	0.006	0.006	0.021	0.001
3224	14	1.927	0.014	0.275	1.343	0.009
30394	210	8.463	0.047	11.954	75.937	2.634

b) Suggestion performance: Results of tests carried-out on two real databases (with more than 100 and 3000 documents and 14 classes) are presented in TABLE I. k parameter has been empirically selected. We want to verify if the system allows the operator not to browse all the documents in order to find errors. For each database, we generate 100 graphs where only one labelling error has been generated. For different values of k and different extracted features, we compute how many times (proportionally to the number of documents of the current class of the erroneous document) an operator has to reject our proposition before we correctly identify a labelling error. Globally, test shows that for a small or a big database, about 20% to 30% of a metanode need to be browsed before we identify a labelling error (instead of 100% without our tool). Huge values of standard deviation show that very often, first proposition is good (50% of the time), but sometimes more than 45% of document class have to be browsed. A usage evaluation and demonstration is available in http://njournet.com/files/DocClass.mp4. TABLE II presents the computational performance of the implemented methods on three different databases with $k = 10$ (application in C++, Intel® Core™ i7-3840QM CPU@2.80GHzx8, 32Gb of RAM).

VI. Conclusion and Perspectives

This article presents a new proposition for an efficient browsing of large databases of document images to identify classification errors. We focus the operator attention on potential errors through graph based quality measurements. We validated the computational efficiency of proposal with a database composed of 30000 document manually labelled. Besides testing new image feature, we will also investigate if it is relevant to use a learning distance method or non linear distance instead of using a simple k-nn algorithm. It is also interesting to take the advantage of the computational capabilities of the computer to use more complex features and to reach quickly a best labelling in the feature extraction step. The software should also allow the correction of the labelling errors instead of only finding them.

References

[1] O. Augereau, N. Journet, and J. P. Domenger, "Document Images Indexing with Relevance Feedback : an Application to Industrial Context," in *ICDAR*, 2011, pp. 1190–1194.

[2] C. Ware, *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[3] J. J. Thomas and K. A. Cook, Eds., *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2006.

[4] B. Schneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in *Proc. of the IEEE Symposium on Visual Languages*, 1996, pp. 336–343.

[5] D. Auber, P. Mary, M. Mathiaut, J. Dubois, A. Lambert, D. Archambault, R. Bourqui, B. Pinaud, M. Delest, G. Melançon *et al.*, "Tulip: a scalable graph visualization framework," in *Extraction et Gestion des Connaissances (EGC) 2010*, 2010, pp. 623–624.

[6] W. Plant and G. Schaefer, "Visualisation and browsing of image databases," in *Multimedia Analysis, Processing and Communications*. Springer, 2011, pp. 3–57.

[7] D. M. Eler, M. Y. Nakazaki, F. V. Paulovich, D. P. Santos, G. F. Andery, M. C. F. Oliveira, J. Batista Neto, and R. Minghim, "Visual analysis of image collections," *Visual Computer*, vol. 25, no. 10, pp. 923–937, 2009.

[8] M. Worring, A. Engl, and C. Smeria, "A multimedia analytics framework for browsing image collections in digital forensics," in *Proceedings of the 20th ACM International Conference on Multimedia*, 2012, pp. 289–298.

[9] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner, "Using Automatic Clustering to Produce High-Level System Organizations of Source Code," in *IEEE Proc. Int. Workshop on Program Understanding (IWPC'98)*, 1998, pp. 45–53.

[10] S. Hachul and M. Jünger, "Drawing large graphs with a potential-field-based multilevel algorithm," in *Graph Drawing*, 2005, pp. 285–295.

[11] T. Dwyer, K. Marriott, and P. Stuckey, "Fast node overlap removal," in *Proc. Graph Drawing 2005 (GD'05)*, 2005, pp. 153–164.

[12] O. C. Enright A.J., Van Dongen S., "An efficient algorithm for large-scale detection of protein families," *Nucleic Acids Research*, vol. 30, no. 7, 2002.