RestorDoc

—

Logiciel de restauration de documents

Plate-forme de test d'algorithmes de traitement d'image en Common Lisp

—

Joachim Pouderoux - Jérôme Prieto
Hugh Putoa - Matthias Robine

8 Mars 2002

Client : Robert Strandh
Chargé de TD : Olivier Navas

Université Bordeaux 1 - Maîtrise Informatique

Résumé

Ce document présente le logiciel RestorDoc et ses motivations. Nous exposons d'abord les principales techniques de restauration de documents. Un aperçu des produits et logiciels existants montre ensuite qu'il n'existe pas d'outil adapté à la recherche sur des algorithmes de traitement d'image en Common Lisp. Nous présentons alors notre logiciel Restordoc, qui répond à cette attente. Nous détaillons ses fonctionnalités puis son interface. Nous expliquons les risques inhérents au choix du langage et à la bibliothèque graphique McCLIM qui est encore en cours d'implémentation. Enfin, un planning donne l'organisation choisie pour développer Restordoc, et un test est proposé pour montrer l'efficacité du logiciel.

Table des matières

1 Introduction au domaine d'application

Le sentiment que le livre est le moyen le plus sûr de transmettre un héritage, un savoir, remonte à des époques lointaines. Les bibliothèques abritent de nombreux ouvrages anciens menacés de disparaître, victimes de l'acidité du papier et de l'humidité. La sauvegarde de ce patrimoine passe par la restauration de documents.
Cette restauration consiste à retrouver la lisibilité d'origine d'un document, à corriger la courbure des pages, à effacer les taches et les déchirures ou à rehausser les caractères estompés par le temps. Ces opérations de restauration s'appliquent sur des documents de natures diverses comme des sources manuscrites ou imprimées, des gravures, des partitions de musique etc... - d'une façon générale, un document peut être ici défini comme une image en niveaux de gris dont l'origine est un document imprimé avec de l'encre noire sur papier blanc.
Toutes ces tâches font appel à l'outil informatique. En effet, la restauration de documents repose sur un processus de traitement d'image. La première étape consiste à numériser le document. La restauration s'effectue par transformation de l'image en lui appliquant une suite d'algorithmes. Présentons quelques types de traitement d'image :

1.1 Rehaussement

L'information contenue dans un document trop sombre ou trop clair peut sembler perdue. Il est cependant possible d'appliquer à l'image des techniques de rehaussement. L'histogramme de l'image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris dans l'image. Le rehaussement consiste alors à modifier l'histogramme de manière à utiliser tous les niveaux de gris au lieu d'une partie seulement. Par exemple, une image sombre concentrée dans les premiers niveaux de gris peut, après rehaussement, s'étaler sur tous les niveaux. L'information de l'image devient visible. Cette technique permet d'améliorer les contrastes et d'augmenter ainsi la qualité de l'image.

1.2 Binarisation

La binarisation permet, pour un document écrit, de séparer le texte du fond. Il s'agit de l'algorithme le plus simple de requantification, chaque pixel de l'image est transformé en un pixel de couleur noire ou blanche en fonction de sa valeur de départ. Considérons par exemple, une binarisation simple avec un seuil de 127 : chaque pixel dont la valeur est supérieure à 127 prend la valeur 255 (blanc), dans le cas contraire on lui associe la valeur 0 (noir).

1.3 Redressement

Un document n'est pas toujours correctement positionné dans le scanner, ce qui produit une image dont le contenu n'est pas horizontal. Le contenu d'un document peut aussi avoir été incliné à l'origine. Il est alors nécessaire de redresser l'image car cette inclinaison peut, par exemple, perturber l'analyse des symboles. Il existe des algorithmes qui détectent (*skew detection*) et calculent l'angle d'inclinaison d'un document. Cet angle est ensuite utilisé par un algorithme de correction (*skew correction*) pour redresser l'image par rotation. [1] décrit un algorithme qui réalise cette opération.

1.4 Segmentation

Ce traitement consiste à repérer dans une page des zones de natures différentes. On peut ainsi identifier les zones de texte, les images, les droites horizontales et verticales. Une méthode est détaillée dans [6].

1.5 Analyse des symboles, indexage et compression

Le but est de reconnaître des caractères ou des signes. L'exploitation de la redondance des symboles dans un texte permet de générer un représentant idéal pour chacun d'eux. Les symboles sont alors remplacés par leur représentant. Cette analyse permet de compresser le document : c'est la compression symbolique. Au lieu de stocker l'image, il suffit de stocker chaque représentant et ses positions dans le document[10].

Tous ces traitements d'image précèdent une éventuelle analyse OCR (*Optical Characters Recognition*). Celle-ci consiste à identifier un caractère parmi des échantillons modèles[2].

2 Analyse de l'existant

La restauration de documents est un domaine vaste qui suscite de nombreuses recherches.
Plusieurs organismes se sont intéressés à la restauration, d'où la multitude d'algorithmes et logiciels existants à ce jour.
Nous citons ici quelques logiciels et projets qui ont retenu notre attention.

2.1 Produits existants

Présentons la société I2S, située à Pessac, qui est l'auteur de nombreux projets.
I2S était à l'origine composée d'ingénieurs en développement, spécialistes de la vision de haute précision. Elle fut ensuite renforcée par des éléments aux compétences nouvelles (réimpression numérique d'ouvrages anciens ou épuisés, base de données de bibliothèques et archives publiques, registres anciens ou précieux, manuscrits, ...).

- *Digibook*© est un scanner automatisé pour ouvrages anciens. Il assure une correction d'éclairage en temps réel et possède des fonctions de correction de courbure, de détourage automatique et de redressement de texte, associées à la retouche d'image (tramage, modification des contrastes, gomme).
- *Book Restorer*©[1] est un logiciel qui intervient dans la vérification des documents numérisés, dans leur restauration et dans leur édition. Il permet par exemple de consulter, restaurer, sauvegarder et publier des images de livres numérisés par Digibook©. Book Restorer© est un logiciel payant et il n'existe pas encore de version d'évaluation (cf. Fig.1). Notre logiciel peut être une étape dans l'élaboration d'un logiciel similaire mais libre.

Les sociétés Xerox (en asssociation avec I2S Bordeaux) et SGBI Entreprise (Transvision) sont à l'origine de *DEBORA*© (*Digital accEs to Books of the RenAissance*).
DEBORA© est un projet européen qui développe des outils permettant l'accès, à partir de postes de consultation distants, à des collections de documents du XVIème siècle. Ce projet regroupe des logiciels d'amélioration et de restauration d'images, d'analyse des formes et d'images de mots, et de comparaison des composantes non textuelles (lettrines, bandeaux, ornements, ...).

La société MATHWORKS commercialise *MATLAB*© (cf. Fig.2). Ce logiciel est une plate-forme intégrant le calcul mathématique, la visualisation

[1]http ://www.i2s-bookscanner.com/fr

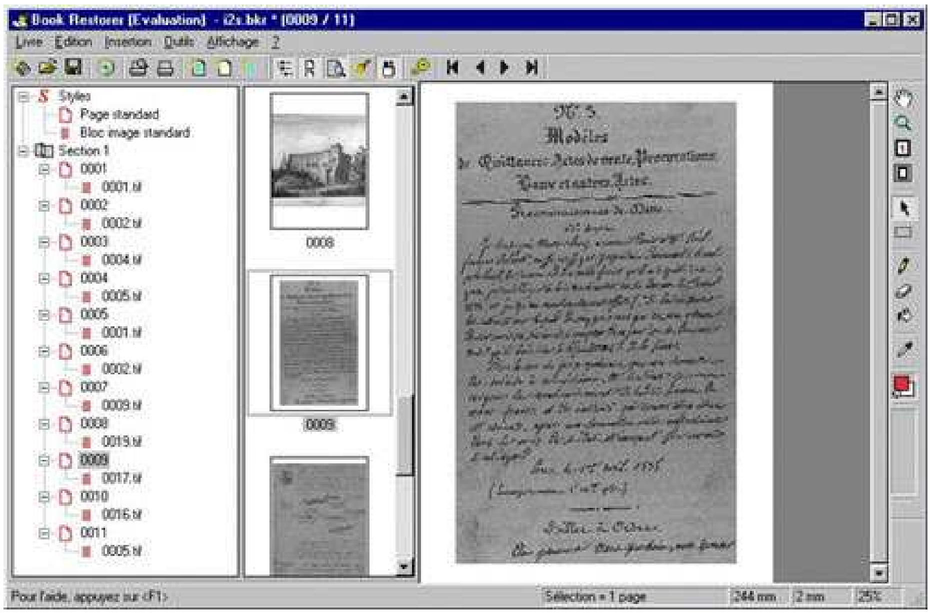


FIG. 1 – Interface du logiciel Book Restorer©

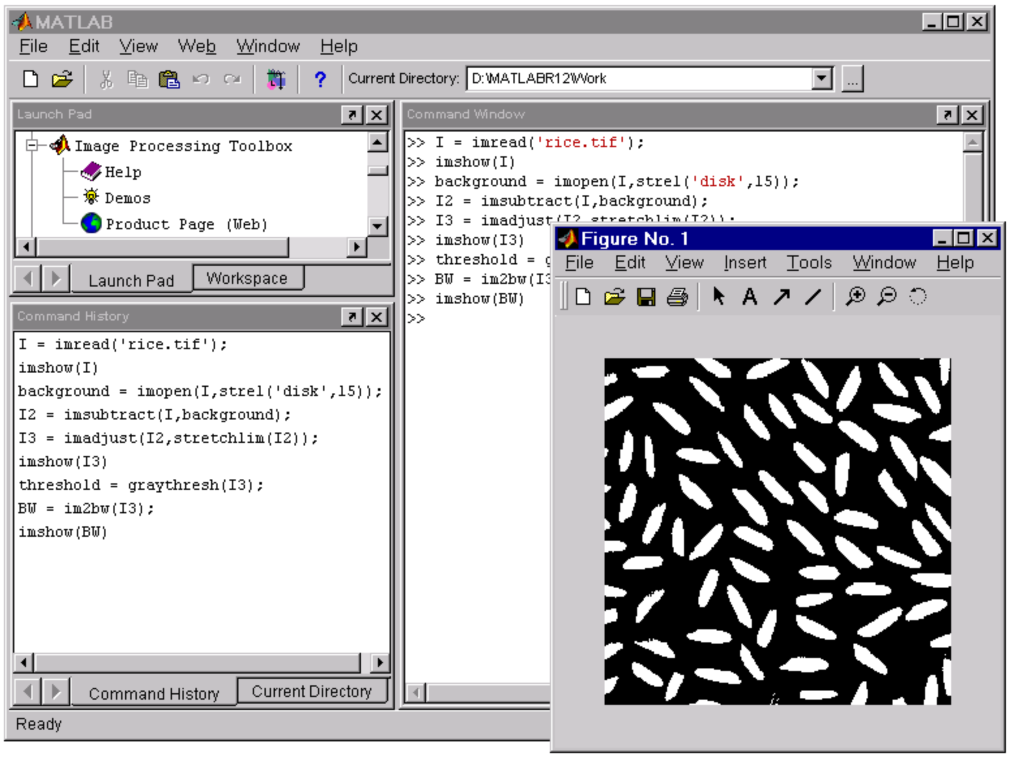


FIG. 2 – Interface du logiciel MATLAB© avec l'outil *Image Processing*

de graphiques et un puissant langage technique appelé MATLAB©. Des boîtes à outils étendent l'environnement de calcul de MATLAB© pour offrir de nouvelles fonctions et outils interactifs. L'*Image Processing Toolbox* est destinée à améliorer et à analyser les images numériques, ainsi qu'à développer des algorithmes de traitement d'images. Celle-ci facilite, en outre, l'apprentissage et la formation aux techniques de traitement d'images, aussi bien pour l'éducation que pour la recherche. La plupart des algorithmes sont mis en œuvre dans le langage MATLAB© qui permet d'explorer et de personnaliser les algorithmes existants de la boîte à outils ou de développer ses propres algorithmes. Cependant le langage MATLAB© n'est pas ouvert - le code développé n'est pas portable - et le logiciel est onéreux.

Peter Mattis et Spencer Kimball ont développé *The GIMP*[2] (*GNU Image Manipulation Program*) en langage C. C'est un logiciel libre de retouche et de création d'images numériques. Ses nombreuses fonctionnalités produisent une interface chargée. The GIMP est extensible, il permet l'adjonction de plug-ins. De nombreux développeurs y ont contribué. Cela permet de programmer de complexes tâches de traitement d'image. Cependant, The GIMP est plus orienté utilisateur que programmeur : il n'a, par exemple, pas de fenêtre d'interaction avec un langage informatique.

2.2 Travaux de recherche

Dans le cadre de leur TER de l'année 2000 [10], Samuel ALVES-ROSA, Emmanuel BARADAT, Sylvain COLLOT et Christophe MESPLEDE ont effectués leurs recherches sur l'aide à la restauration de vieux livres. Ce qui consiste en :
- la restauration de pages de livres usagés,
- la reconnaissance de symboles,
- l'exploitation de la redondance de symboles afin de les remplacer par le meilleur candidat,
- la compression des images restaurées pour un stockage plus efficace.

Leur programme AdVitam implémente ces techniques en C mais sans interface graphique adaptée.

Jacques-Olivier LAPEYRE a étudié des algorithmes de binarisation et de compression symbolique de documents en noir et blanc dans le cadre de son mémoire de DEA 2001[5]. Il détaille différents algorithmes qui traitent de la binarisation et évalue leur degré d'automatisation en vue de l'augmenter.
Jacques-Olivier LAPEYRE souligne ([5] p.76) le manque d'une architecture

[2] http ://www.gimp.org

adaptée au test de ses algorithmes.

Les logiciels présentés dans la section précédente sont : soit payants (et leurs sources ne sont pas librement distribuées), soit généraux et non adaptés à la recherche en restauration de documents. Il est fastidieux et coûteux en temps pour un chercheur de devoir tester ses algorithmes sans un outil adapté. L'exemple de The GIMP est caractéristique : il fournit tous les outils pour la restauration de documents mais il ne permet pas à un programmeur d'algorithmes de traitement d'image de les tester efficacement (l'insertion d'un plug-in nécessite une connaissance approfondie du logiciel, l'accès au zoom n'est pas direct...). Il manque un logiciel adapté, RestorDoc doit répondre à cette attente.

3 Introduction au projet

Notre projet consiste en la réalisation d'une interface graphique qui répond à des besoins scientifiques. Le développement d'algorithmes de traitement d'image en Common Lisp a créé le besoin d'une interface adaptée.
C'est pourquoi notre plate-forme est programmée en utilisant McClim, une bibliothèque Lisp et libre mais qui est encore en cours d'implémentation (cf. p.15). Notre logiciel intégre des algorithmes de traitement d'image, et gère - c'est à dire charge, affiche et sauvegarde - des images au format PGM (*Portable Gray Map*) en 256 niveaux de gris.
Notre programme est cloisonné, ainsi le programmeur d'un algorithme peut utiliser le logiciel pour tester son efficacité sans se soucier de la gestion graphique.
L'interface est efficace : toute transformation à effectuer ne prend pas plus d'un clic de souris. Des raccourcis clavier sont possibles pour un accès rapide aux fonctions. L'interface est alors adaptée aux utilisateurs d'Emacs.
Le logiciel permet le multi-fenêtrage afin de comparer l'efficacité des algorithmes. Chaque transformation ouvre une nouvelle fenêtre d'affichage. Il est ainsi possible de visualiser 2 transformations (et plus) et comparer leur temps d'exécution. De plus, cela permet d'appliquer de nouveau à une transformation à une image résultat. Il est facile d'effectuer les opérations de zoom et de déplacement de l'image :

- le zoom s'effectue avec un clic droit de la souris et un déplacement avant ou arrière
- et le déplacement de l'image avec un clic gauche de souris et un déplacement.

On peut maintenant décrire l'interface (cf. prototype papier p.14) :

- en haut, on trouve une barre de menus, avec un menu "File" pour ouvrir, sauvegarder l'image et quitter le logiciel, et un menu "Modules" pour appliquer les différentes transformations ;
- sur la gauche de l'interface on trouve une barre d'outils qui permet des retouches élémentaires sur l'image (gomme, crayon, pipette...) ;
- au centre se trouve la fenêtre d'affichage de l'image : le canevas ;
- en bas, il y a une fenêtre d'interaction Common Lisp. Elle permet, entre autres, l'intégration des algorithmes dans le logiciel tels des plug-ins. Cela crée une entrée-menu dans l'interface et associe éventuellement un raccourci clavier pour effectuer la transformation ;
- enfin, il y a une barre d'état où sont affichées des informations telles que le temps d'exécution ou l'historique des transformations effectuées depuis l'image source.

4 Besoins fonctionnels

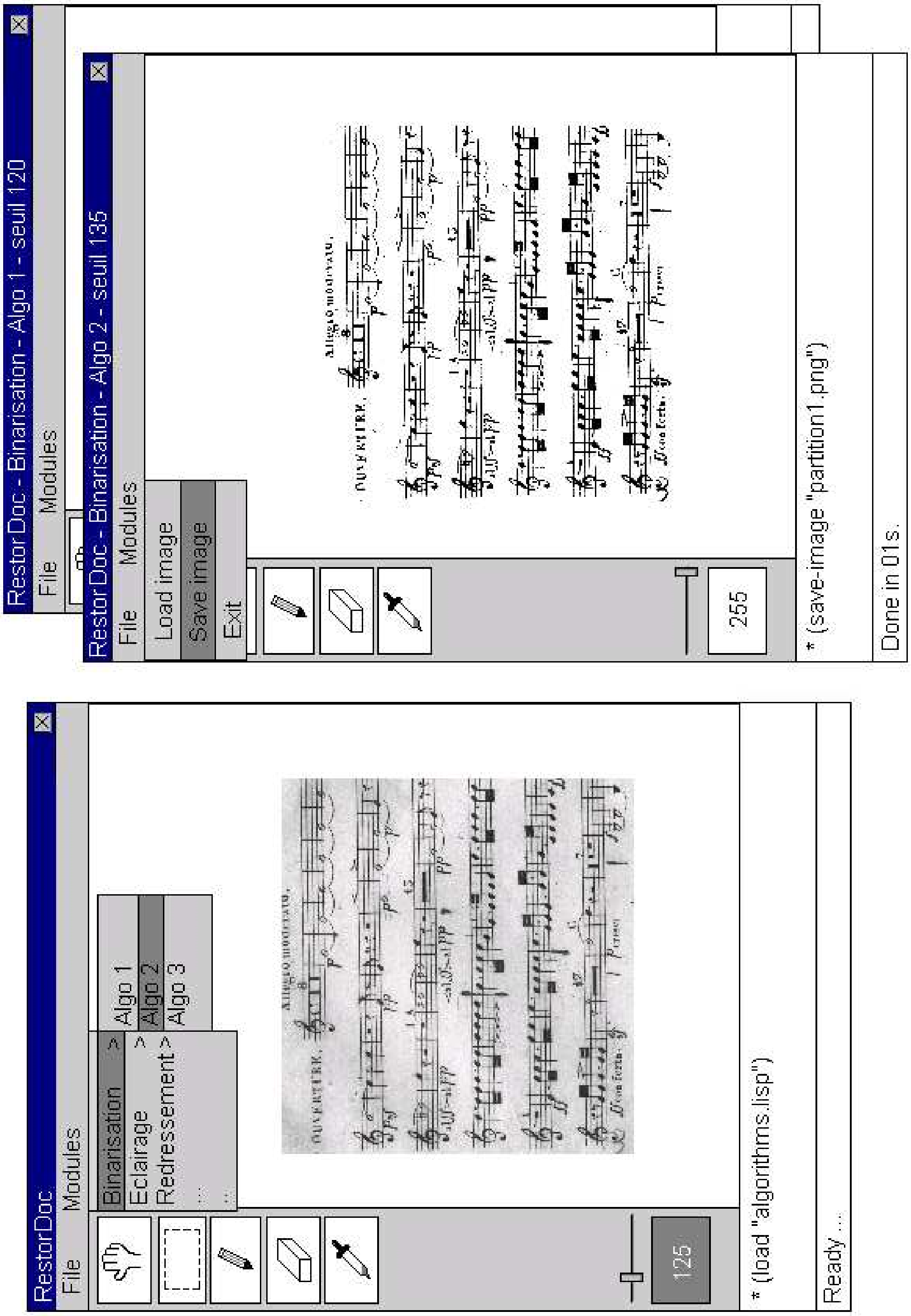
4.1 Utilisateur interface graphique

	Description	Action(s) à réaliser/Résultat(s)
Charger-Image	Charge un fichier image au format PGM et la substitue à l'image associée au canevas de la fenêtre.	Soit clic sur "Load image" dans le menu "File" puis choix du fichier dans le sélecteur de fichier, soit via un raccourci clavier puis en tapant le chemin du fichier dans l'invité.
Sauvegarder-Image	Sauvegarde l'image associée au canevas de la fenêtre courante.	Soit clic sur "Save image" dans le menu "File" puis choix du chemin grâce au sélecteur de fichier et entrée du nom, soit via un raccourci clavier puis en tapant le chemin du fichier dans l'invité.
Appliquer-Module	Lance l'algorithme sélectionné sur l'image affichée dans le canevas de la fenêtre.	Clic sur le nom de l'algorithme dans le menu "Modules".
Obtenir-couleur-pixel	La couleur de la zone de couleur prend la couleur du pixel sélectionné.	Clic sur la pipette dans la barre d'outils puis choix du pixel dans le canevas par clic gauche. La couleur du pixel s'affiche dans la zone de couleur de la barre d'outils.
Changer-couleur-par-défaut	Modifie la couleur de la zone de couleur	Clic sur la glissière dans la barre d'outils et maintenir en la déplaçant jusqu'à l'obtention de la couleur voulue dans la zone de couleur de la barre d'outils.

Modifier-couleur-pixel		Clic sur le crayon puis sur le pixel de l'image à modifier dans le canevas.
Effacer-pixel		Clic sur la gomme dans la barre d'outils puis choix du pixel dans le canevas par clic gauche.
Zoomer-image		Cliquer sur le centre du zoom avec le bouton droit dans le canevas et maintenir le clic en avançant la souris pour zoom avant et en reculant pour zoom arrière.
Déplacer-image		Clic sur la main dans la barre d'outils puis maintenir un clic gauche sur le canevas et déplacer avec la souris.
Déterminer-région	Détermine les coordonnées d'une région rectangulaire d'application définie par l'utilisateur à l'aide de la souris.	Clic sur le cadre de sélection dans la barre d'outils puis maintenir un clic gauche sur un point du canevas vers un autre point.
Afficher-temps-d'execution	Affiche dans la barre d'état le temps d'exécution de l'algorithme qui a engendré l'image.	
Multi-fenêtrage	Chaque application d'un algorithme entraîne la création d'une nouvelle fenêtre.	
Modifier-paramètres	Modifie les paramètres d'un algorithme.	Cliquer l'entrée-menu de la transformation à réaliser. Presser la touche SHIFT avant de relâcher le clic. Remplacer les paramètres dans la fenêtre d'interaction.

	Description	**Action(s) à réaliser/Résultat(s)**
Charger-fichier-initialisation	Au démarrage du logiciel, chargement d'un fichier Lisp qui déclare les modules à charger.	Les modules des fichiers sont intégrés au logiciel et les entrées-menu du menu “Modules” sont créées.
Intègrer-module	Intègre au logiciel une fonction Lisp qui prend au moins comme argument le buffer de l'image à transformer.	Appel de la fonction d'insertion de module. Il faut passer en paramètre : le type d'algorithme, son nom (qui formeront l'entrée dans le menu), l'adresse de la fonction de transformation, ses paramètres par défaut (hormis le premier paramètre obligatoire qui est le buffer contenant l'image), et éventuellement un raccourci clavier.
Créer-entrée-menu		L'intégration d'un module entraîne la création dans le menu “Modules” d'une entrée-menu. Cette entrée appelle la fonction du module.
Supprimer-entrée-menu		Appel dans la fenêtre d'interaction d'une fonction supprimer-entrée avec le nom de l'entrée-menu à supprimer.
Déclarer-raccourci-clavier	Associe un raccourci clavier à l'appel d'une fonction.	Signaler le raccourci à utiliser lors de l'insertion du module.

5 Prototype papier



6 Besoins non fonctionnels

6.1 Common Lisp

Common Lisp est un langage de très haut niveau issu de la normalisation d'un langage LISP (*List Processing Language*). Le développement de Lisp a commencé en 1956 au MIT, et Common Lisp a été normalisé par ANSI à partir de 1986. C'est un langage interactif et incrémental : un programme est élaboré à partir d'une boucle `lecture-évaluation-affichage` et chaque itération modifie l'environnement précédent. Ce fonctionnement induit un modèle de programmation différent [8] des langages tels que C ou Java. De plus, Common Lisp s'est enrichi d'une couche objet CLOS (*Common Lisp Object System*) et peut ainsi rivaliser avec les autres langages orientés objet.

6.2 CLIM - Common Lisp Interface Manager

CLIM[3] est un ensemble de spécifications décrivant une API (*Application Programmer Interface*). Le but est de pouvoir écrire rapidement des interfaces graphiques portables en Common Lisp. CLIM se charge d'adapter l'application au système sur lequel elle s'exécute. Elle paraît alors spécifique au système.
CLIM repose sur CLOS, la couche objet de Lisp, et décrit ainsi un ensemble de protocoles et de fonctions nécessaires à son fonctionnement.
L'intérêt, mais aussi la difficulté, de CLIM est qu'il s'agit d'une interface de très haut niveau. De nombreux concepts originaux sont introduits par CLIM : les transformations, les présentations... Autre originalité : les fenêtres d'une application sont décrites (ou déclarées), et non pas créees par des appels successifs à une fonction de création de fenêtre ou widget.

6.3 McCLIM

McCLIM [4] est un projet visant à développer une implémentation libre des spécifications CLIM. Initié par Mike McDonald en 1998, McCLIM a fusionné avec le projet FreeCLIM, et a évolué grâce à la participation de Robert Strandh et Gilbert Baumann.
McCLIM n'implémente pas encore la totalité de CLIM et de nombreuses fonctionnalités sont à améliorer ou à utiliser avec précaution. De plus, McCLIM ne fonctionne pour l'instant que sous X Window, par le biais de la bibliothèque CLX.
Cependant, le projet est diffusé sous licence GNU LGPL et il est entretenu quotidiennement grâce a un serveur CVS. Il est donc probable que dans un avenir proche, McCLIM sera suffisamment stable et fonctionnel pour réaliser de grosses applications.

[3] CLIM specifications 2.0
[4] http ://www.mikemac.com/mikemac/McCLIM

7 Risques et parades

Le principal risque auquel RestorDoc peut être confronté est lié à McCLIM. En effet, comme précisé précédemment, McCLIM est toujours au stade du développement. De nombreux bugs subsistent et toutes les spécifications de CLIM ne sont pas encore supportées. RestorDoc peut souffrir de cette instabilité, d'autant qu'il est possible que certaines fonctions soient modifiées, renommés, voire supprimées au cours des prochaines mises à jour. Il sera en outre difficile de différencier les bugs provoqués par McCLIM de ceux liés au développement de RestorDoc.
La parade réside dans le fait que McCLIM évolue et que ses sources sont disponibles. Deux attitudes pourront alors être adoptées : soit attendre que le bug soit corrigé par un tiers, soit entreprendre nous-même le debugage. Quant à l'évolution de McCLIM, elle pourra nécessiter la mise à jour de RestorDoc.

8 Test de validation

Nous allons tester le fonctionnement de notre interface avec un algorithme de binarisation.

Ecrivons tout d'abord un algorithme de binarisation par seuillage global dans un fichier `binarise.lisp` :

```
(defun binarisation-seuillage-global (image seuil)
  (let ((hauteur (car (array-dimensions image)))
        (largeur (cadr (array-dimensions image))))
    (loop for r from 0 below hauteur
          do (loop for c from 0 below largeur
                   do (setf (aref image r c)
                            (if (> (aref image r c) seuil) 255 0))))
```

Il faut ensuite ajouter une ligne pour l'intégration de l'algorithme dans le logiciel :

```
(integrer-module ``Binarisation'' ``Seuil global''
                 #'binarisation-seuillage-global ``128''
                 :shortkey ``C-1'')
```

On peut alors tester l'algorithme au sein de RestorDoc.

Pas	Action à effectuer	Résultat attendu
1	Lancement du programme.	Ouverture d'une fenêtre A.
2	Ouverture d'une image PGM à l'aide de l'entrée "Load image" du menu "File".	Affichage de l'image dans le canevas.
3	Zoom de l'image avec clic droit maintenu.	Affichage de l'image zoomée dans la fenêtre A.
4	Sélection de la couleur d'un pixel avec la pipette.	La couleur du pixel apparaît dans la zone de couleur.
5	Déplacement de l'image dans le canevas à l'aide de la main.	
6	Changement de la couleur sélectionnée dans la zone de couleur à l'aide de la glissière.	
7	Chargement du fichier Lisp en entrant la commande d'insertion d'un module dans la fenêtre d'interaction.	Chargement du code de l'algorithme dans l'environnement Lisp. Une nouvelle entrée apparaît dans le menu "Modules"
8	Sélection de l'algorithme à lancer par un raccourci clavier.	Ouverture d'une fenêtre B et affichage de l'image transformée dans le nouveau canevas. Le temps de calcul est visible dans la barre d'état de la fenêtre B.
9	Sélection de l'algorithme dans le menu de la fenêtre A avec la souris. Par pression simultanée d'une touche du clavier, on modifie les paramètres par défaut dans la fenêtre d'interaction.	Ouverture d'une fenêtre C et affichage de l'image transformée dans le nouveau canevas. Le temps de calcul est visible dans la barre d'état de la fenêtre C.
10	Sélection de l'option "Save image" dans le menu "File" de la fenêtre C.	Sauvegarde de l'image de la fenêtre C.
11	Fermeture des fenêtres A, B et C grâce au menu "File".	Fermeture de l'application.

9 Planning

Semaine	**Prévisions**
11/03 $\rightarrow$ 17/03	Lecture du code de McCLIM, des exemples et de la documentation. Elaboration de l'architecture du logiciel. Mise en place d'un serveur CVS pour maintenir le logiciel.
18/03 $\rightarrow$ 24/03	*Groupe 1 :* Elaboration de la fenêtre principale *Groupe 2 :* Détermination de la méthode d'intégration d'un module et mise à jour de la gestion des images dans l'extension de McCLIM (création de la fonction `save-image`)
25/03 $\rightarrow$ 31/03	Programmation des fonctions relatives à l'intégration des modules et gestion des processus.
01/04 $\rightarrow$ 07/04	*Groupe 1 :* Programmation des fonctions associées à la boîte à outils et à l'affichage dans la barre d'état (temps d'exécution, historique...) *Groupe 2 :* Gestion du zoom et de la sélection d'une région.
08/04 $\rightarrow$ 14/04	Améliorations : raccourcis claviers, mini-buffer, optimisations du code.
15/04 $\rightarrow$ 21/04	*Groupe 1 :* Validation du logiciel *Groupe 2 :* Rédaction du manuel technique.
22/04 $\rightarrow$ 28/04	Rédaction du manuel utilisateur et suite du manuel technique.
29/04 $\rightarrow$ 05/05	Remise des manuels et du code.
06/05 $\rightarrow$ 12/05	Préparation de la soutenance.
13/05 $\rightarrow$ 19/05	Soutenance.

Références

[1] T. PARKINSON R. SHIU A. AMIN, S. FISCHER. Fast algorithm for skew detection. San Jose, USA, 1996. In the Symposium on Electronic Imaging. School of Computer Science and Engineering, University of New Soth Wales, Sydney NSW, 2052, Australia.

[2] L. NAJMAN D. REYREN A. BELAID, L. PIERRON. *Numérisation de Documents : Principes et Evaluation des Performances.* ADBS, La Bresse, 2000. Cours INRIA sur les Bibliothèques numériques.

[3] I. STEPHAN B. COUASNON, P. BRISSET. Using logic programming languages for optical music recognition. Paris, France, 1995. In the Third International Conference on the Practical Application of Prolog. IRISA, Insa-Département Informatique.

[4] S. FEINER J. HUGUES J.FOLEY, A VAN DAM. *Computer Graphics : Principles and Practice.* Addison-Wesley, Reading, second edition, 1990.

[5] J.O. LAPEYRE. *Binarisation et Compression Symbolique de Documents noir et blanc.* Mémoire de dea informatique, Labri, Université Bordeaux I, Juin 2001.

[6] R. DERICHE N. PARAGIOS. Coupled geodesic active regions for image segmentation. Rapport de Recherche 3783, INRIA, Institut National de Recherche en Informatique et en Automatique, 1999.

[7] I. PITAS. *Digital Image Processing Algorithms and Applications.* Wiley, 2000.

[8] I. DURAND R. STRANDH. *Traité de Programmation en Common Lisp.* Méta Modulaire, 2001.

[9] J.C. RUSS. *The Image Processing Handbook.* CRC Press, third edition, 1999.

[10] S. COLLOT C. MESPLEDE S. ALVES-ROSA, E. BARADAT. *Aide à la Restauration de Vieux Livres.* Mémoire de ter, Université des Sciences Bordeaux I, 2000.