République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Développement d'une application de traitement d'images

Réalisé par :

- **Sabri akram**
- **Benali Moustafa**

Présenté le 27 Juin 2013 devant la commission d'examination composée de MM.

- *Mr Chouiti Sidi Mohamed.* *(Examinateur)*
- *Mr Benazzouz Mertada .* *(Examinateur)*
- *Mme Chaouache Lamia.* *(Examinateur)*

Année universitaire : 2013-2014

Remerciements

On dit souvent que le trajet est aussi important que la destination. Les trois années de maîtrise nous ont permis de bien comprendre la signification de cette phrase toute simple. Ce parcours, en effet, ne s'est pas réalisé sans défis et sans soulever de nombreuses questions pour lesquelles les réponses nécessitent de longues heures de travail.

Nous tenons à la fin de ce travail à remercier ALLAH le tout puissant de nous avoir donné la foi et de nous avoir permis d'en arriver là.

En premier lieu nous tenons à remercier Monsieur CHOUITI, notre encadreur, qui nous a initié au traitement d'images et nous a fait découvrir le monde passionnant de la recherche et avec qui nous avons eu le plaisir de mener tous nos travaux de mémoire. Son entière disponibilité et ses merveilleuses explications ont été précieuses.

Nous souhaitons également faire part de nos reconnaissances à tous les enseignants qui nous ont éclairés la voie du savoir durant notre cycle.

Un grand merci également à l'ensemble du personnel pédagogique, technique et administratif du département, principalement mes condisciples lors de la formation.

Nous tenons à exprimer notre profonde gratitude à toutes celles et ceux qui nous ont apporté leur soutien, leur amitié ou leur expérience tout au long de ce travail de mémoire.

Dédicace

La page de la dédicace est généralement celle qui est écrite en dernier et lue en premier, aussi je doute que ce mémoire puisse faire exception à la règle.

Je dédie ce mémoire à toute ma famille, mon père Sabri Abdelkader, ma mère Saidouni Fatma, mes frères et mes sœurs,et ma fiançaille et mes-belles sœurs, et mes amis

Je le dédie aussi à mon encadreur Mr CHouiti qui m'a aidé à réaliser ce travail.

Je dédie ce travail ✍

Dédicace

Ce que personne ne peut compenser les sacrifices qu'ils ont consentis pour mon éducation et pour mon bien être, qui n'ont jamais cessé de me soutenir moralement et matériellement pour que je puisse finir mes études et avoir une bonne formation et surtout être le meilleur :

A mes très chers parents Benali Ahmed et Bourouaha Saliha

A mes frères Mohamed, Abdelkader, Ismail et mes sœurs, et ma belle-sœur.

A mon encadreur Monsieur CHouiti qui m'a facilité le chemin de mes études par ses précieux conseils et orientations bénéfiques.

A tous mes amis.

A ceux qui sont la source de mon inspiration et mon courage.

Je dédie ce travail ✍

Sommaire

Chapitre II:Traitement d'image en java

Chapitre III : Implémentation

Introduction générale

Notre travail consiste à étudier les techniques de traitements d'images et en particulier celles liées à la restauration et l'amélioration des images. Puis d'implémenter ces techniques en java, un langage évolué et orienté objet. Pour cela nous avons d'abord examiné les classes et packages concernant la manipulation des images en java. Puis nous avons utilisé les classes BufferedImage, ImageIO, etc. pour instancier, modifier et manipuler des objets images. Ces objets contiennent des données telles que la taille de l'image, les couleurs des pixels de l'image ou le modèle de couleur utilisé.
Le mémoire est organisé comme suit :

Chapitre I : nous présentons dans ce chapitre les caractéristiques d'une image, quelques formats et les différentes techniques de traitement d'une image.

Chapitre II : nous étudions les package et les classes permettant la manipulation d'objets image.

Chapitre III : nous présentons l'environnement de développement utilisé et notre logiciel dans lequel nous avons présenté Implémentation de quelques algorithmes de traitement d'image (compression, restauration, élaboration de l'histogramme).

Introduction générale

Chapitre I :

Caractéristiques et formats d'images

Chapitre I : Caractéristiques et formats d'images

Dans ce chapitre, nous allons présenter les différentes phases de la formation d'image ainsi que sa définition, de son interprétation par la machine, de ces différents types, de ces caractéristiques et de ces formats.

I.1 Définition de l'image : [1]

L'image est définie comme étant une fonction $f(x, y)$ à deux dimensions, où x et y sont les coordonnées spatiales, et f l'amplitude à tous points (x, y) correspondant à l'intensité ou au niveau de gris. Lorsque les points (x, y) et l'amplitude sont discrétisés, on parle d'image numérique ou digitale. Dans ce dernier cas la fonction f est remplacée par la lettre I et le couple (x, y) par le couple (i, j).

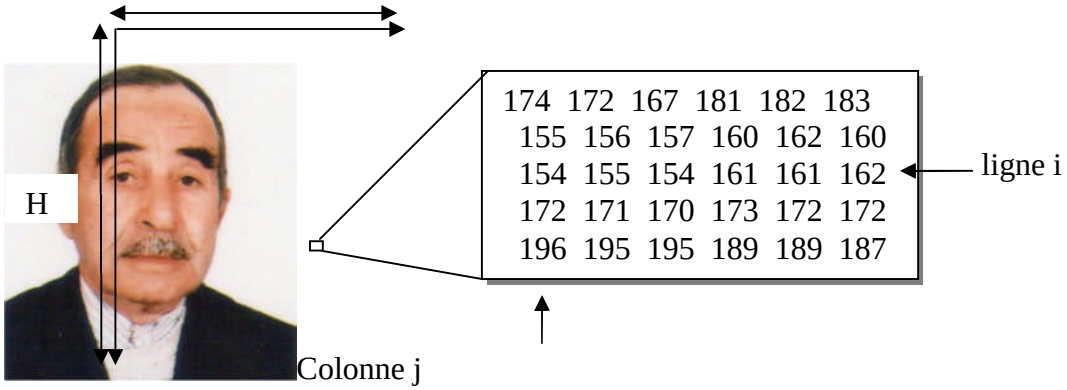


Figure I.1 – Image couleur, son repère et un extrait de pixels

L'image numérique, désignée aussi par le terme scène, possède un repère comme indiqué en figure I.1, il est différent de celui d'une fonction mathématique. Elle a une hauteur (H) et une largeur (W).

I.2 Codage : [2]

L'information est codée en binaire. Le support évolue mais le principe est toujours le même un même élément peut se trouver dans 2 états différents stables. Il constitue une mémoire élémentaire ou bit

I.3Types d'images :[2]

En traitement d'images nous aurons à faire à quatre types d'images :

I.3.1 Image binaire :

Chaque pixel est soit blanc soit noir. Puisqu'il y a uniquement deux valeurs pour chaque pixel, un seul bit est utilisé pour le coder.

Codage d'une image en noir et blanc

Pour ce type de codage, chaque pixel est soit noir, soit blanc. Il faut un bit pour coder un pixel (0 pour noir, 1 pour blanc). L'image de 10000 pixels codée occupe donc 10000 bits en mémoire.

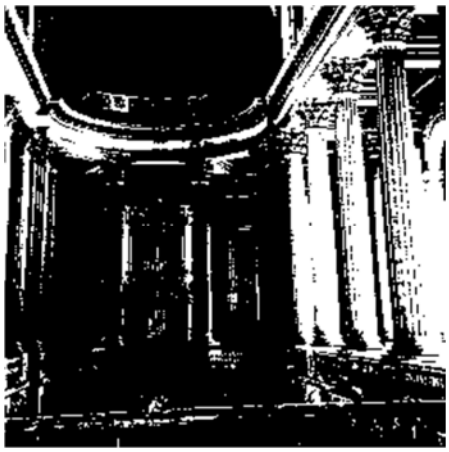


Figure I.2 – Image en noir et blanc

Ce type de codage peut convenir pour un plan ou un texte mais on voit ses limites lorsqu'il s'agit d'une photographie.

I.3.2 Image en niveau de gris :

Chaque pixel est un niveau de gris, allant de 0 (noir) à 255 (blanc). Cet intervalle de valeur signifie que chaque pixel est codé sur huit bits (un octet). 256 niveaux de gris suffisent pour la reconnaissance de la plus part des objets d'une scène.

Codage d'une image en niveaux de gris

Si on code chaque pixel sur 2 bits on aura 4 possibilités (noir, gris foncé, gris clair, blanc). L'image codée sera très peu nuancée.

En général on code chaque pixel sur 8 bits = 1 octet. On a alors 256 possibilités (on dit 256 niveaux de gris).

L'image de 10 000 pixels codée occupe alors 10 000 octets en mémoire.

Figure I.3 – Image en niveaux de gris

Exemple d'image en 72 pixels par pouce (environ 30 pixels par cm), codée en 256 niveaux de gris.

Cette image de 303 x 303 pixels occupe 303 x 303 = 91809 octets puisque chaque pixel occupe 1 octet en mémoire.

I.3.3 Image couleur (RGB) :

Chaque pixel possède une couleur décrite par la quantité de rouge (R), vert (G) et bleu (B). Chacune de ces trois composantes est codée sur l'intervalle [0, 255], ce qui donne 255^3 = 16 777 216 couleurs possibles. Il faut 24 bits pour coder un pixel.

Codage d'une image en couleurs 8 bits

Dans ce cas on attache une palette de 256 couleurs à l'image.

Ces 256 couleurs sont choisies parmi les 16 millions de couleurs de la palette RVB. Pour chaque image le programme recherche les 256 couleurs les plus pertinentes.

Chaque code (de 0 à 255) désigne une couleur.

L'image occupe 3 fois moins de place en mémoire qu'avec un codage 24 bits. L'image est moins nuancée : sa qualité est bonne mais moindre.

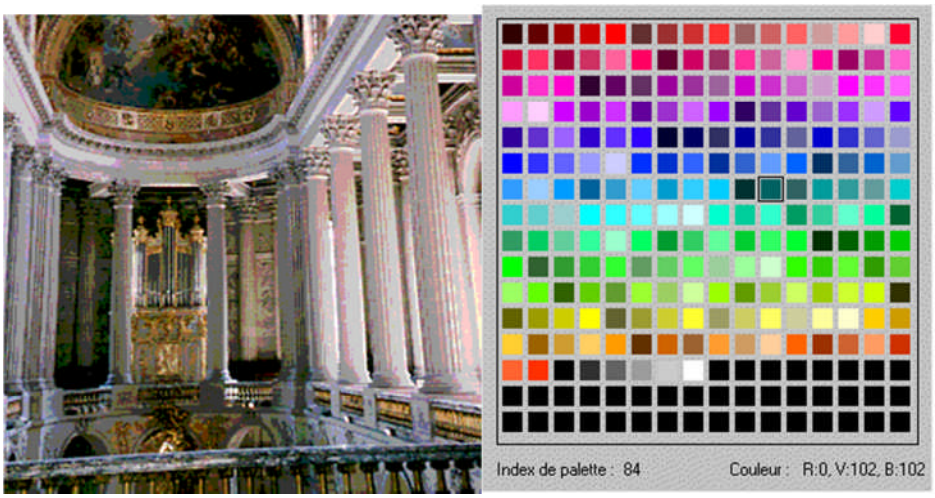


Figure I.4– Image en couleurs 8 bits**Figure I.5** – Palette 8 bits

Image couleur 8 bits et sa palette. Remarquer la couleur 84 et sa correspondance en RVB.

Codage d'une image en couleurs 24 bits

Il existe plusieurs modes de codage de la couleur. Le plus utilisé est le codage Rouge, Vert, Bleu (RVB). Chaque couleur est codée sur 1 octet = 8 bits. Chaque pixel sur 3 octets c'est à dire 24 bits : le rouge de 0 à 255, le vert de 0 à 255, le Bleu de 0 à 255.

Le principe repose sur la synthèse additive des couleurs : on peut obtenir une couleur quelconque par addition de ces 3 couleurs primaires en proportions convenables.

On obtient ainsi 256 x 256 x 256 = 16777216 (plus de 16 millions de couleurs différentes).

Couleur	Noir	bleu pâle	vert pâle	Rose	Bleu	vert	rouge	Blanc
R	0	120	120	150	120	120	255	255
V	0	120	150	120	120	255	120	255
B	0	150	120	120	255	120	120	255

Table I.1 – Tableau du codage d'une image en couleurs 24 bits

L'image de 10000 pixels ainsi codée occupe 10000 x 3 = 30000 octets.

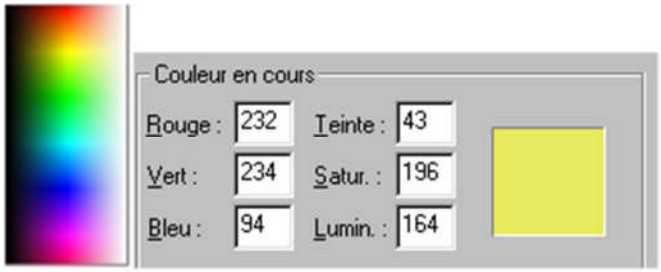


Figure I.6 – Palette 24 bits Exemple de couleur en 24 bits

Figure I.7 – Image en couleurs 24 bits

Dans le cas d'une image de 10 cm x 10 cm avec une résolution convenable de 100 pixels par cm (un pixel mesure 0,1 mm), elle est codée sur 1000 x 1000 = 1 000 000 pixels

Elle occupe : en noir et blanc : 1 000 000 bits = 125 000 octets

En 256 niveaux de gris : 1 million d'octets

En couleurs (24bits) : 3 millions d'octets.

C'est codage de la couleur qui est utilisé par la plupart des écrans d'ordinateurs actuellement.

On constate qu'il est très gourmand en mémoire. Pour faciliter le stockage des images en mémoire on utilise d'autres formes de codage.

I.4 Résolution :

Une image est divisée en points ou pixels. Considérons une image de 10 cm sur 10 cm avec une résolution très faible de 10 pixels par cm.

Elle est codée sur 100 x 100 = 10000 pixels.

Avec une résolution convenable de 100 pixels par cm (un pixel mesure 0,1 mm), elle serait codée sur 1000 x 1000 = 1000000 pixels = 1 M pixels. Le symbole M signifiant million.

I.5 Caractéristiques de l'image : [3]

I.5.1 Pixel :

Une image est constituée d'un ensemble de points appelés pixels (voir l'extrait de la figureI.1). Le pixel (Picture élément) représente ainsi le plus petit élément constitutif d'une image numérique. La quantité d'information que véhicule chaque pixel donne des nuances entre images monochromes et images couleurs. Pour les images 3D le «pixel» est alors appelé un voxel, et représente un volume élémentaire. Des exemples d'images de ce type se rencontrent dans les images médicales. Les images tomographiques axiales sont ainsi des images construites à partir de plusieurs radiographies faites sous des angles de vue différents.

I.5.2 Poids de l'image :

C'est la taille de l'image. Etant donné que cette dernière est représentée sous forme d'une matrice dont les valeurs représentent l'intensité (pixels), le nombre de colonne (W) multiplié par le nombre de ligne (H) donne le nombre total de pixels dans l'image.

Pour une image de 640x480 en couleur :

- ✓ Nombre de pixel = 640x480 = 307200
- ✓ Poids de chaque pixel = 3 octets
- ✓ Le poids de l'image = 307200x3 = 921600 octets = 900 Ko

I.5.3 Transparence : [4]

La transparence est une caractéristique définissant le niveau d'opacité des éléments de l'image, c'est la possibilité de voir à travers l'image des éléments graphiques situés derrière celle-ci.

I.5.4 Luminance : [4]

C'est le degré de luminosité des points de l'image. Elle est définie ainsi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface.

I.5.5 Contraste :

Est une propriété intrinsèque d'une image qui désigne et quantifie la différence entre les parties claires et foncées d'une image (elle différencie les couleurs claires des couleurs foncées).

En photographie on le définit le contraste comme la différence entre la densité la plus forte et la plus faible d'une image. Le contrôle du contraste est un élément important de la pratique photographique. Le contraste final de l'image dépend à la fois du sujet, de la nature et du traitement du négatif et du positif.

I.5.6 Histogramme : [4]

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image. Il fournit diverses informations comme les statistiques d'ordre (moyenne, variance,…), l'entropie, et peut permettre d'isoler des objets.

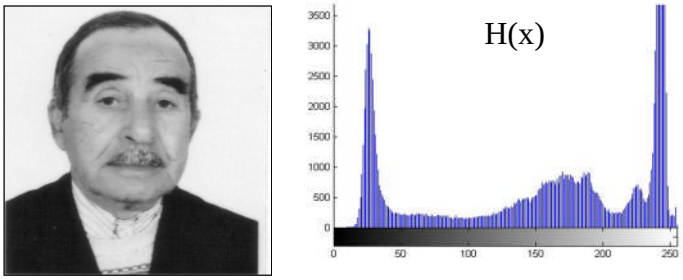


Figure I.8 – Image en niveau de gris et son histogramme

L'histogramme de la figure I.12 noté ici H(x), est le nombre de pixels dont le niveau de gris est égal à x. l'histogramme cumulatif normalisé est calculé comme suit :

$$HC(x) = \frac{\sum_{i=0}^{x} H(x)}{W * H} \quad \text{(I.1)}$$

HC(x) est le taux de pixels dont le niveau de gris est inférieur à x.

Nous présentons dans la suite quelques traitements d'analyse effectués uniquement à partir de l'histogramme. Retenons que certains de ces traitements sont souvent calculés au niveau des capteurs, et qu'en général leur pertinence est très intimement liée aux conditions d'acquisition.

(1) Normalisation : exploiter toute la dynamique de codage.

(2) Egalisation : équilibrer la dynamique de codage et augmenter le contraste.

(3) Segmentation : simplifier l'image en regroupant les pixels selon leurs valeurs.

I.5.7 Homogénéité :

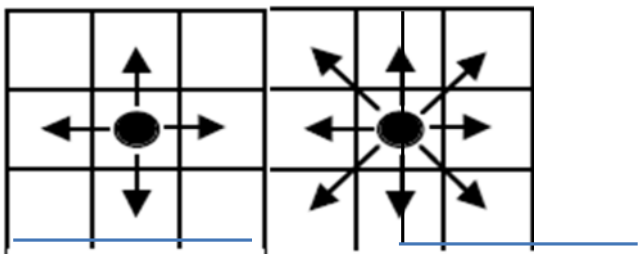
L'homogénéité est une information locale et correspond au caractère uniforme d'une région. Une région dans une image est dite homogène si elle regroupe un ensemble de pixels qui possèdent des caractéristiques similaires ou uniformes. Ces caractéristiques peuvent être par exemple la variance du niveau de gris, la couleur,…

I.5.8 Connexité :

En traitant une image, on est souvent amener à se déplacer dans celle-ci. Un déplacement doit souvent obéir à des règles de voisinage, on utilise généralement deux types de voisinage : le voisinage à 4-connexité (4 pixels voisins, fig. II.13(a)) et le voisinage à 8-connexité (8 pixels voisins, fig. II.135(b)). Deux pixels seront considérés comme connexes (appartenant au même objet donc) s'ils satisfont deux critères:

- d'une part un critère de similarité (par exemple même niveau de gris)

- s'ils sont adjacents (voisins)



(a) 4-connexité (b) 8-connexité

Figure I.9 – Voisinages

I.5.11 Région :

Une région est un ensemble de pixels connexes et homogènes. Un pixel n'appartient à une région donnée que s'il vérifie les caractéristiques de celle-ci (intensité moyenne, centre de gravité,…). Une région est toujours limitée par un contour.

I.6 Les différents formats d'images :

On peut classer les images en deux formats :

I.6.1 Format vectorielle : [2]

Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique.

Par exemple, un cercle est décrit par une information du type (cercle, position du centre, rayon). Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans.

I.6.2 Format matricielle : [2]

Une image matricielle est formée d'un tableau de points ou pixels. Plus la densité des points sont élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée.

Corrélativement la place occupée en mémoire et la durée de traitement seront d'autant plus grandes.

Les images vues sur un écran de télévision ou une photographie sont des images matricielles.

On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanner.

Parmi ces formats on peut citer :

BMP (BitMap) : Le format BMP est le format par défaut du logiciel Windows. C'est un format matriciel. Les images ne sont pas compressées. Son logiciel d'origine.

Le format EPS : matriciel n'est pas très différent du EPS vectoriel. En fait seules les données contenues dans le fichier sont différentes. Ainsi un logiciel de retouche de photos tel que Photoshop permet l'importation, la modification et l'exportation de fichiers en format EPS.

GIF (GraphicalInterchange Format) : Le format GIF est un format qui a ouvert la voie à l'image sur le World Wide Web. C'est un format de compression qui n'accepte que les

images en couleurs indexés codé sur 8 bits, C'est un format qui perd beaucoup de son marché suite à une bataille juridique concernant les droits d'utilisation sur Internet.

JPEG (Joint Photographique Experts Group) : Les images JPEG sont des images de 24 bits. C'est-à dire qu'elles peuvent afficher un spectre de 16 millions de couleurs. C'est la meilleure qualité d'images disponible.

PCX : Le format PCX est utilisé par le logiciel Paintbrush sous Windows. C'est un format matriciel.

TIFF (Tagged Image File Format): Le format TIFF, conçu à l'origine par la compagnie Aldus est un format matriciel. Conçu au départ pour n'accepter que les images en RGB, ce format permet de coder des images CYMK.

PBM (Potable BitMap) : Commençons par le plus simple : le format pbm. Ce format permet de stocker des images en noir et blanc

PGM (Potable GrayMap) : Le format pgm permet de représenter des images en niveaux de gris dont les pixels ont des valeurs entières comprises entre 0 (noir) et 255 (blanc). La valeur de chacun des pixels est enregistrée dans le fichier au format ASCII.

PPM (Potable PixMap): Le format ppm concerne les images couleurs. Chaque pixel a pour valeur un triple (R, G, B) composé d'une composante rouge, verte et bleue. Chaque composante est représentée par un entier pouvant prendre ses valeurs entre 0

I.7 les techniques fondamentales en traitement d'images :

Une panoplie de traitements peut être appliquée à l'image numérique, la figure suivante résume l'ensemble de ces traitements.

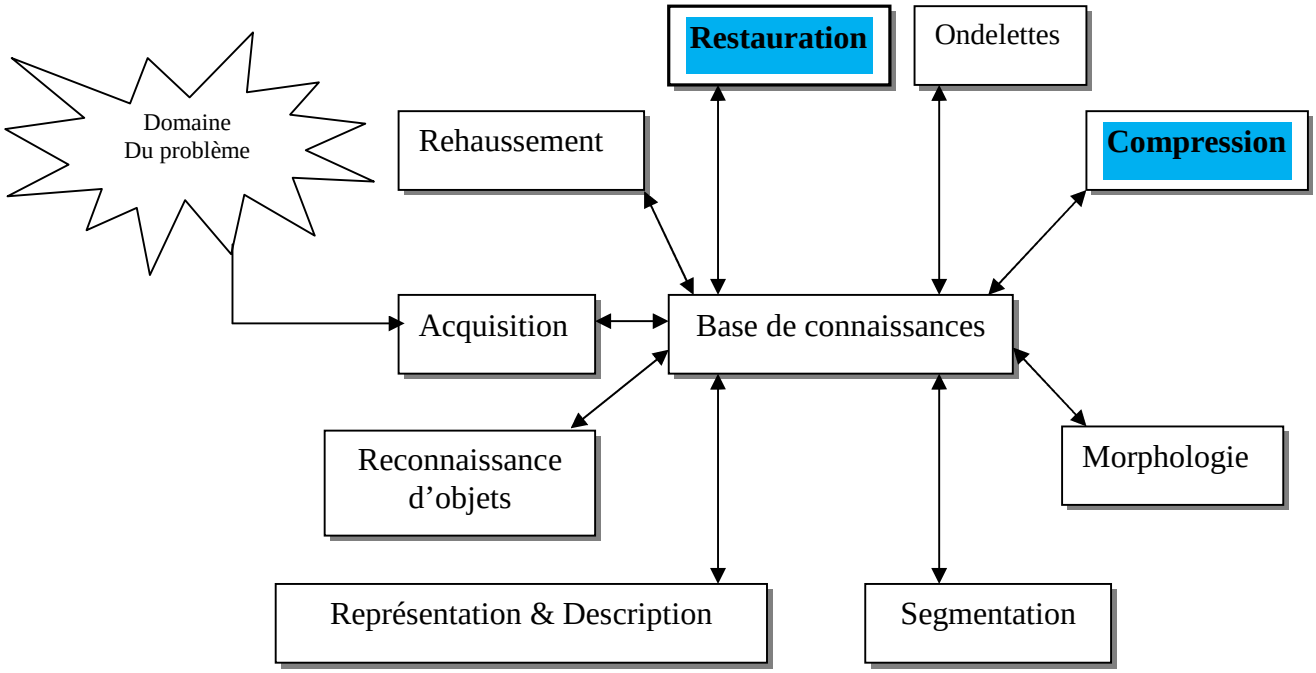


Figure I.10 – Traitements fondamentaux en traitement d'images

I.7.1 Acquisition de l'image :

L'acquisition d'images est une mesure spatiale d'une interaction entre une onde et de la matière.

L'onde est émise par une source et reçu par un capteur.

La matière occupe de l'espace et possède une masse.

Elle a pour objet de passer de la scène physique à une forme numérique observée.

I.7.2 Le rehaussement de l'image :

Le mécanisme de formation des images est loin d'être parfait donc présence de différentes formes de bruit, améliorer le contraste d'où l'objectif est de :

- Rehausser le niveau de gris
- Accentuer les caractéristiques

I.7.3 Les ondelettes :

Elles permettent de représenter l'image en différentes résolutions. Elles sont utilisées aussi en compression.

I.7.4 La morphologie :

C'est outil permettant d'extraire des composantes d'une image pour décrire et représenter différentes formes.

I.7.5 La segmentation :

C'est une procédure permettant de partitionner l'image en ses constituants ou objets. La segmentation automatique est la tâche la plus difficile en traitement d'images. Plus la segmentation est meilleur plus l'étape de reconnaissance d'objets est réussite.

I.7.6La représentation et description :

La représentation est l'étape qui vient juste après la segmentation. Le résultat de la segmentation est un ensemble de pixels relatifs à une région. Ces données doivent être converties en une forme traitable par un ordinateur. On peut soit représenter la région ou sa frontière. La description est l'extraction d'attribut permettant de distinguer une classe d'objets d'une autre classe.

I.7.7 La reconnaissance : [1]

Est le traitement qui affecte une étiquette (exemple : route, voiture,..) à un objet en se basant sur ses descripteurs.

La base de connaissance contient la connaissance du domaine du problème en cours du traitement. Dans son aspect le plus simple, elle peut consister en coordonnées de l'objet à traiter, ceci permet de réduire l'espace de recherche. Comme elle peut être complexe contenant toutes les défaillances que peut présenter un produit manufacturé.

I.7.8 La restauration d'image :

Les images subissent des dégradations, ces dégradations sont dues, d'une part au Appareilles d'acquisition et d'autre part aux conditions de prise de vue...

Son but est d'améliorer la qualité d'une image, atténuer, supprimer les dégradations

La restauration se fait par des technique comme noir et blanc, négative, rotation, etc.

Noir et blanc : L'image noir et blanc, est une image dont les couleurs ont été remplacés par le gris sauf le noir et le blanc de l'image.

Négative : estune image dont les couleurs ont été inversées par rapport à l'originale par exemple le noir devient blancet inversement.

Comprissions d'image : [6]

La compression d'image est une application de la compression de données sur des images numériques. Cette compression a pour utilité de réduire la redondance des données d'une image afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou la transmettre rapidement.

La compression d'image peut être effectuée avec perte de données ou sans perte

Compression sans perte :

Appelée aussi compression non destructrice, la qualité de l'image après décompression est la même que celle de l'image originale, le taux de compression de ce type est limité.
Ce type de compression on le trouve beaucoup dans le domaine où la précision est majeure commel'image médicale (IRM par ex.) ou la télédétection (imagerie satellite par ex.).

Compression avec perte :

C'est une compression destructrice, elle permet de sacrifier certains détails de l'image non récupérable en décompression au profit de réduction de poids. Cette dégradation peut être contrôlée selon la qualité qu'on veut obtenir en fonction du taux de compression choisie.

Ce type de compression on le trouve généralement dans le domaine normal pratique là oùla réduction du poids de l'images est très important, comme le domaine multimédia par exemple (web, photographie) où la fidélité envers l'image original n'est pas très importante et le taux de compression sera plus grand que celui d'une compression sans perte dufait qu'on est juste limité par la qualité qu'on souhaite obtenir.

Conclusion

Dans ce chapitre, nous avons cité les différents traitements que peut subir une image,, Dans la suite nous présentons le langage java et l'utilisation des package et des classes concernons l'implémentation de ces traitement.

Chapitre II :

Traitement d’image en java

Chapitre II : Traitement d'image en java

Java est un langage orienté objet ou son constructeur on a prévu des package et des classes permettant la manipulation d'objets image. C'est grâce à son API Java 2D, qu'on peut écrire des logiciels de traitement et d'analyse d'images.

II.1 Fonctionnalités de Java2D : [7]

Java 2D est une API composée par un ensemble de classes destinées à l'imagerie et à la création de dessin 2D.
Elle permet de :

- dessiner des lignes, des rectangles et toute autre forme géométrique ;
- replisser les formes par des couleurs unies ou en dégradés et lui ajouter des textures ;
- ajouter du texte, lui attribuer différentes polices et contrôler son rendu ;
- dessiner des images, éventuellement en effectuant des opérations de filtrage.

Dans cet article on va traiter la partie imagerie de l'API comme on va voir les différentes fonctions fournies pour faire le traitement d'images.

II.2 Architecture de l'API :

Pour travailler avec les images sous Java 2D, il faut connaitre deux classes importantes de l'API :

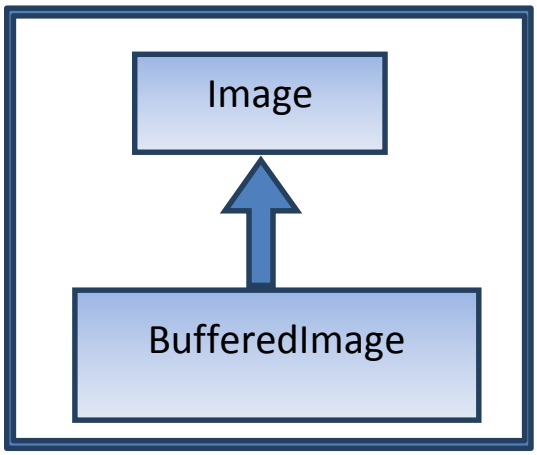


Figure II.1- deux classe improtantes de l'API

II.2.1 java.awt.Image :

C'est la super classe fournie dans la JDK depuis sa version 1.0. C'est une classe abstraite qui permet de représenter les images sous forme d'un rectangle de pixels.
La restriction de cette classe est qu'elle ne permet pas d'accéder aux pixels. Elle est donc inadaptée pour le traitement d'images.

II.2.2 java.awt.image.BufferedImage :

Ajoutée à la JDK depuis sa version 2. Elle hérite de la classe Image et implémente ses interfaces pour permettre d'examiner l'intérieur des images chargées et travailler directement sur ses données. On peut donc à partir d'un objet BufferedImage, récupérer les couleurs des pixels et changer ces couleurs. Comme son nom l'indique, BufferedImage est une image tampon. Cette classe gère l'image dans la mémoire et fournit des méthodes pour l'interprétation des pixels. Si on veut faire du traitement d'images, il faut donc travailler avec des objets BufferedImage. Comme le montre la figure suivante, un objet BufferedImage est composé de deux parties :

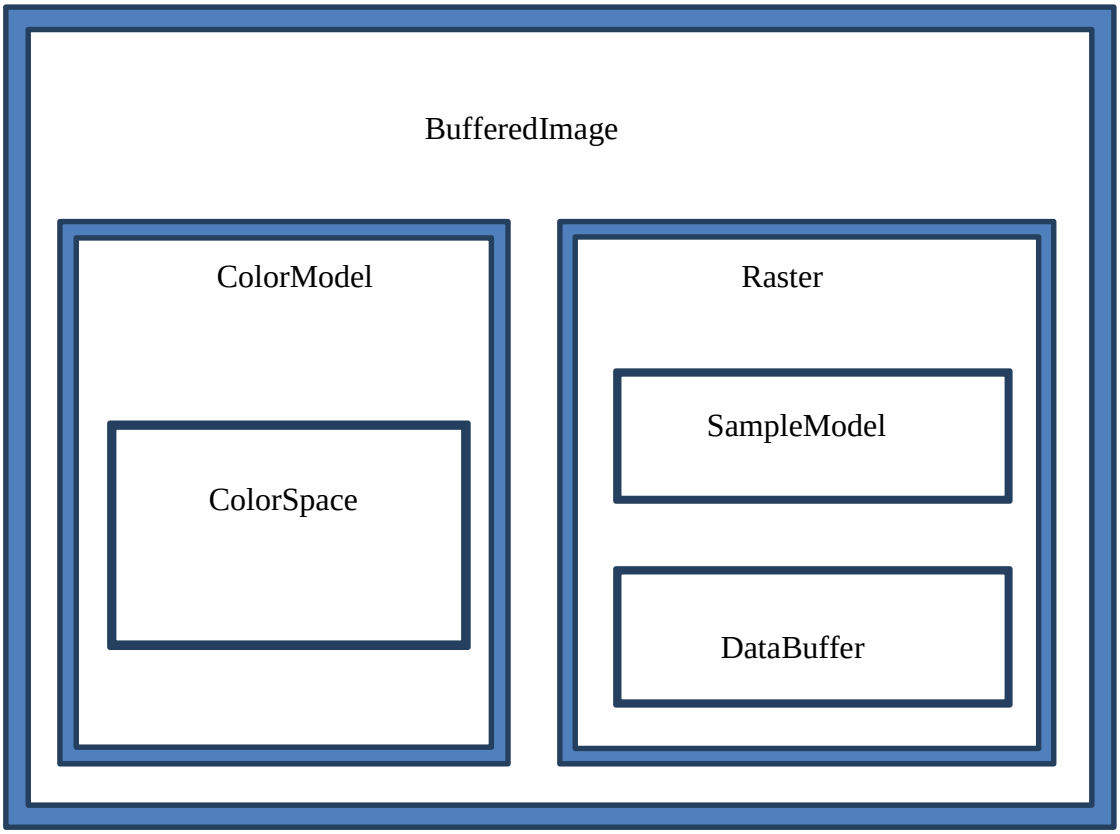


Figure II.2- les différents classes de BufferedImage

II.2.2.1 java.awt.image.ColorModel:

Cette classe définit la façon d'interpréter les couleurs. Le ColorModel est capable de traduire les valeurs des données provenant du Raster en objet **java.awt.Color**.

II.2.2.2 java.awt.image.Raster :

Elle contient les données de l'image et peut les représenter comme un tableau de valeurs de pixels. Aussi, elle maintient les données de l'image dans la mémoire et fournit des méthodes pour accéder à des pixels spécifiques au sein de l'image. Modifier les données d'une image est en créer d'autres instances.

II.3Les différentes implantations d'image :

- Une image est un objet de la classe abstraite

  java.awt.Image

- Il existe plusieurs implantations :

  –java.awt.image.BufferedImage qui correspond à tableau De Pixel en mémoire.

  – java.awt.image.VolatileImage qui correspond à une Image Stockée dans la carte vidéo.

  – sun.awt.image.ToolkitImagequi correspond à une Image Chargée de façon paresseuse par rapport à une Source de Donnée.

II.4 Chargement d'une image depuis un fichier : [8]

Java permet de charger des fichiers GIF, JPEG ou PNG depuis une unité de stockage (disque ou réseau). Le fichier est chargé par la classe Applet ou Toolkit de**java.awt**,nous sommes ici dans une application nous utilisons. Par exemple :

Avec Toolkit (limité à GIF, JPEG, PNG) :

```
/** Accès au toolkit : */
java.awt.Toolkit toolkit=java.awt. Toolkit.getDefaultToolkit ();
/** lecture de l'image : */
Image image= toolkit.getImage ("fichier");
Image image= toolkit.getImage (url);
```

Figure II.3- Code source pour chargement d'image

Avec Toolkit (supporte GIF, JPEG, PNG, BMP et WBMP) :

```
BufferedImage image = ImageIO.read (/* File, URL ou InputStream */);
```

Figure II.4- Code source pour lire une image

A noter qu'avec **ImageIO**, les images sont complètement chargées lors du retour de la méthode **read()**, alors que les images renvoyées par le **Toolkit** sont chargées en arrière-plan.

II.5 Affichage d'une image : [8]

L'affichage d'une image est une chose relativement simple à comprendre lorsque l'on saitcomment les images sont chargées.

Tout Component peut afficher une image au moyen d'un objet Graphiques, donné en paramètre de la méthode paint du composant. La méthode à utiliser est ImageIO.write :

```
BufferedImage image;
...
File f = new File("...");
try {
if(!ImageIO.write(image, "png", f))
JOptionPane.showMessageDialog(null, "Ecriture impossible : "+format);
}catch (IOException e) {
   ...
}
```

Figure II.5- Code source pour afficher l'image

Afin de réafficher le composant une fois que l'image a été chargée, il est nécessaire de choisirl'ImageObserver à fournir en paramètre, le plus simple étant de donner le Component lui-même (this).

Bien que tout composant puisse afficher une image, il est conseillé pour commencer de choisir une Applet ou une Frame.

II.6 Comment sauver une image au format JPEG ou PNG : [8]

Pour sauver une Image, vous devez la convertir en BufferedImage, puis la sauver avec ImageIO. Pour simplifier, vous pouvez utiliser la méthode suivante **:**

```
importjavax.imageio.*;
import java.io.*;
importjavax.swing.*;
import java.awt.image.*;
/**
Enregistre l'image sur le disque. Le format est défini par
l'extension du fichier. Un message d'erreur est affiché si
le format est inconnu.
@paramimage_name nom de fichier à écrire. Doit se terminer par .JPEG ou .PNG
@param img l'image à sauvegarder
*/
public void save(String image_name, Image img) {
BufferedImage bi = new BufferedImage (width, height, BufferedImage
.TYPE_INT_ARGB);
Graphics2D g = bi.createGraphics();
g.drawImage(img, 0, 0, width, height, null);
String file_format = image_name.substring(image_name.lastIndexOf('.')+1);
try { booleansuccess = ImageIO . write(bi, file_format , new File (image_name));
if (!success) { JOptionPane.showMessageDialog(new JFrame(), "Ecriture
impossible:"+file_format);
} catch (Exception e) { e.printStackTrace();
}//end try
}//end Save
```

Figure II.6- Code source pour sauvegarde l'image

Conclusion :

Après l'étude de l'implémentation de techniquede techniques de traitement d'image en java nous allons présenter dans le chapitre III d'abord l'environnement et les outils utilisés puis la différente fonctionnalité et les interfaces de notre application.

Chapitre III :

Implémentation

Chapitre III : Implémentation

Dans ce chapitre nous allons présenter notre implémentation logicielle ainsi que les résultats,Des quelques traitements qu'on a fait.

III.1Langage de programmation utilisé :

La programmation sous Windows a la réputation, en partie injustifiée, d'être difficile et de demander un important investissement personnel de la part du développeur. Or, aujourd'hui de nombreux logiciels de programmation permettent un développement aisé sous Windows.

III.2 NetBeans : [9]

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'EDI NetBeans, jusqu'à ce qu'il soit acheté par Sun Microsystems en 1999. Sun place le projet sous double licence CDDL et GPL v2 en juin de l'année suivante

NetBeans Est un environnement de développement intégré (EDI), placéen open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Développement and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML.

III.3 Présentation de l'application :

Cette application est un logiciel écrit en Java permettant d'appliquer certaines opérations sur les images, Elle permet de :

- lire des fichiers images et les afficher sur l'écran ;
- enregistrer des images après modification sur le disque ;
- appliquer un ensemble de traitement comme :
    - rendre l'image négative, ou noire et blanc,
    - les rotations (gauche / droite) de l'image,

- retour au l'image original,
- retailler l'image,
- sauvegarder l'image,
- compression d'image,
- afficher l'histogramme de l'image,
- réglage des niveaux,
- atténuation des extrêmes,
- accentuations.

III.4 Interface graphique :

L'application est composée de deux classes : classe cadre, et classe PanDessin, comme il est présenté dans la figure suivante :

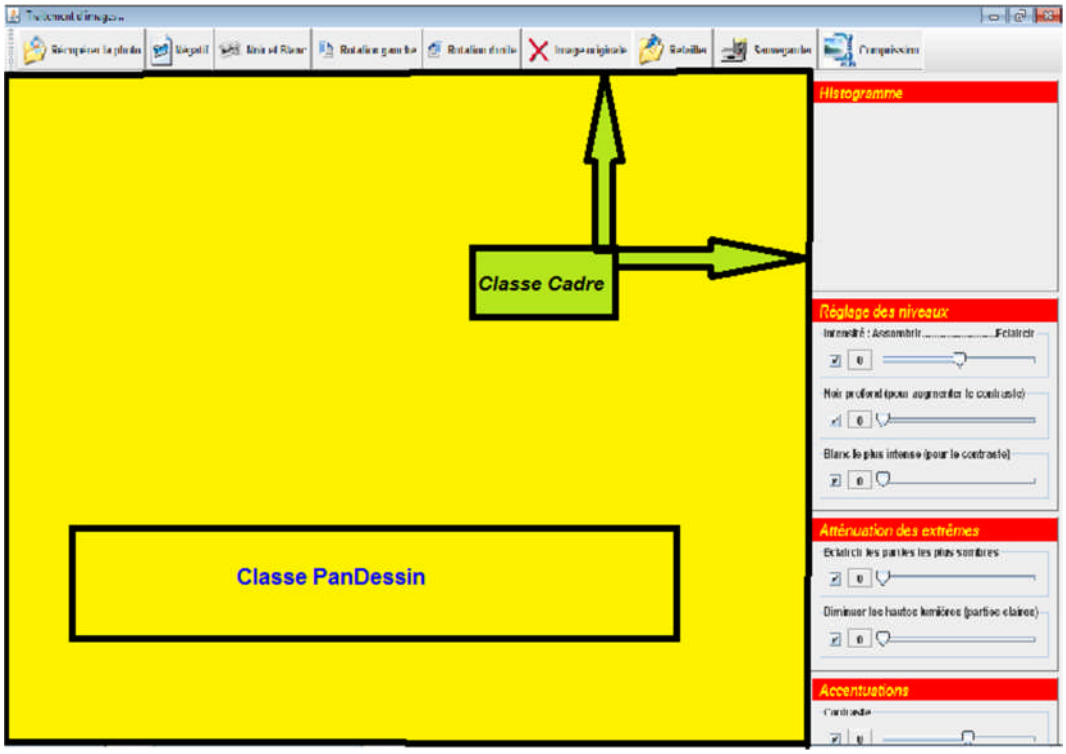


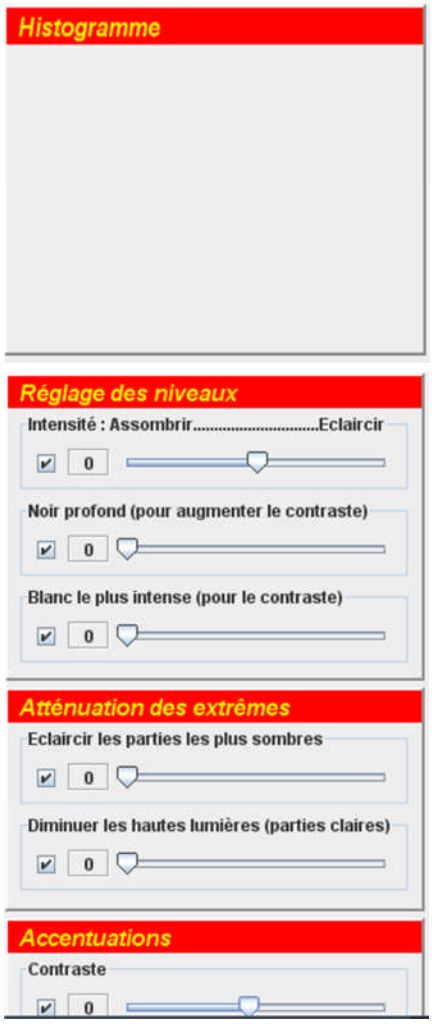
Figure III.1- interface de l'application

III.4.1 Classe Cadre :

Cette classe représente les fenêtres principales de l'application, qui contient des Bouton,qui permet de charger et traiter l'image, comme il est présenté dans(figure III .2).

Figure III.2- la barre de classe cadre qui contient les Bouton

- Le Bouton <<**Récupérer la photo** >> qui permet d'ouvrir un fichier image stocké sur le disque, <<**Sauvegarder**>> qui permet d'enregistrer une image sur le disque après modification.



Ainsi le traitement d'image mentionné dans la barre présenté dans la figure : (figure III.3), qui contient réglage des niveaux, atténuation des extrêmes, accentuations et l'histogrammequi définit l'intensité de chaque de ces derniers.

Figure III.3- la barre du classe cadre qui contient autretraitement

III.4.2 Classe PanDessin :

Cette classe dérive de JPanel. Elle présente le panneau où seront affichées les images chargées par l'application.

Elle admet un attribut unique monImage de type BufferedImage qui présente à tout moment l'image chargée affichée dans le panneau.

Cette classe présente le cœur de l'application. Ces méthodes permettent de modifier les images.

```
Public Fenêtre () {
        this.setTitle("Traitement d'images...");
        this.setBounds(50, 50, 600, 400);
        this.setDefaultCloseOperation(this.EXIT_ON_CLOSE);
        this.setExtendedState(this.MAXIMIZED_BOTH);
        this.getContentPane().add(barreBoutons,
        BorderLayout.NORTH);
        boutonNouvellePhoto.setFocusPainted(false);
        barreBoutons.add(boutonNouvellePhoto);
        barreBoutons.add(boutonNégatif);
        barreBoutons.add(boutonNoirBlanc);
        barreBoutons.add(boutonRotationGauche);
        barreBoutons.add(boutonRotationDroite);
        barreBoutons.add(boutonReinitialiser);
        barreBoutons.add(boutonRetailler);
        barreBoutons.add(boutonSauvegarder);
        barreBoutons.add(boutonCompression);
        this.getContentPane().add(image, BorderLayout.CENTER);
        this.getContentPane().add(palettes, BorderLayout.EAST);
        palettes.setPreferredSize(new Dimension(310, this.getHeight()));
```

```
                palettes.add(niveaux);

                palettes.add(atténuation);

                palettes.add(accentuation);

}
```

Figure III.4- La classe principale de l'application

III.5Etude de différent traitement :

III.5.1 image négative :

Une image négative (figure III .5 .2), est une image dont les couleurs ont été inversées par rapport à l'originale(figure III.5.1), par exemple le noir devient blancet inversement, et le bleu devient marron.

Ainsi l'Histogramme de l'image négative montré dans la figure :(figure III.5.3).

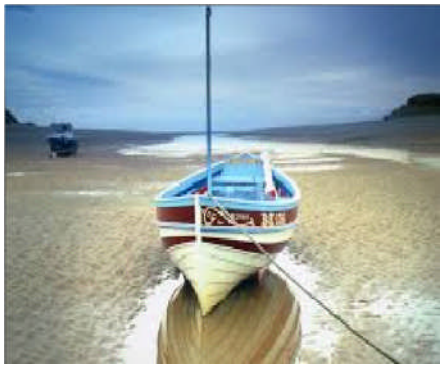


Figure III.5.1- image original

Figure III.5.2- image négative

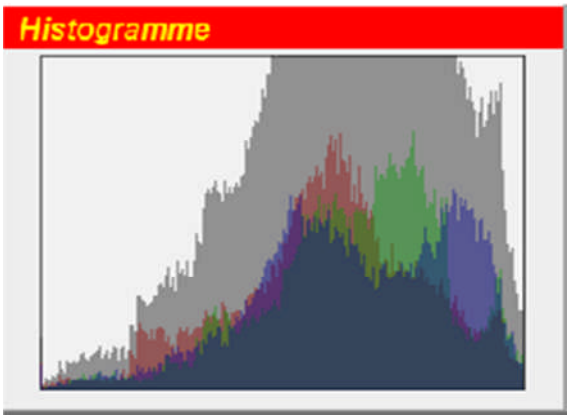


Figure III.5.3- Histogramme qui présente les couleurs de l'image négative

La méthode qui donne l'image négative est :

```
public void négatif () {
byte[ ] inverser = new byte[256];
for (int i=0; i<256; i++) inverser[i] = (byte) (255-i);
ByteLookupTable table = new ByteLookupTable(0, inverser);
LookupOp inversion = new LookupOp(table, null);
inversion.filter(source, source);
calcul();
    historique.ajout ("négatif");
  }
```

Figure III .5.4- code source qui faire le négative

III.5.2Image noir et blanc :

L'image **noir et blanc** (figure III.5.4), est une image dont les couleurs ont étéremplacéspar le gris sauf le noir et le blanc de l'image originel (figure III.5.1).

Il est bien défini dans l'histogramme la disparition des couleurs est la présence de gris et du noire, voire la figure : (figure III.5.5).

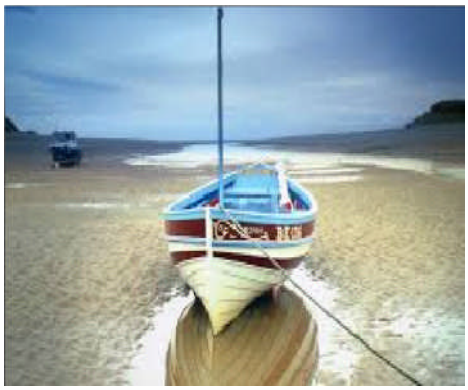


Figure III.5.1- image original

Figure III.5.5- image noire et blanc

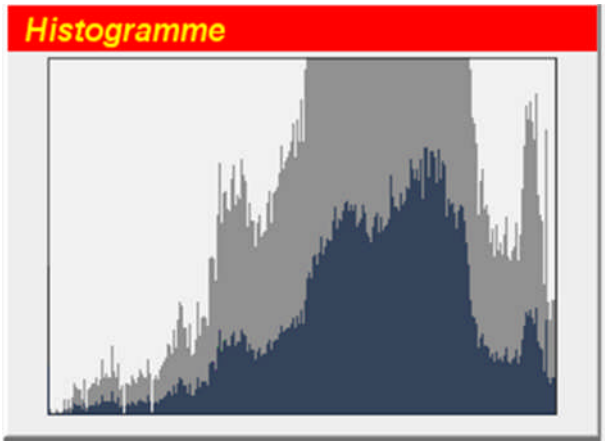


Figure III.5.6- Histogramme qui présente les couleurs de l'image noire et blanc

La méthode qui donne l'image noire et blanc est :

```
public void noirEtBlanc() {
        ColorConvertOpgris new ColorConvertOp (ColorSpace.getInstance
        (ColorSpace.CS_GRAY) ,null);
        gris.filter(source, source);
        calcul();
        historique.ajout("noirEtBlanc");
}
```

FigureIII.5.7-code source qui mettre l'image en noire et blanc

III.5.3Les Rotations :

La **rotation**est le mouvement d'un corps autour d'un point ou d'un axe, avec un angle donné.

La figure (figure III.5.8) montre la rotation dans le sens gauche avec angle de 90° :

Figure III.5.1- image original**Figure III.5.8-** image après une rotation gauche

La méthode qui donne la rotation d'image vers la gauche est :

```
public void rotationGauche() {
        BufferedImage transfert = new BufferedImage(source.getHeight(),
        source.getWidth(), source.getType());
        double centreDeRotation = source.getWidth()/2;
        AffineTransform pivoter = AffineTransform.getRotateInstance(Math.toRadians(-
        90), centreDeRotation, centreDeRotation);
        AffineTransformOppivoterImage = new AffineTransformOp(pivoter, null);
        pivoterImage.filter(source, transfert);
        source = transfert;
        copieSourceImage();
        historique.ajout("rotationGauche");
 }
```

Figure III.5.9-Code source qui fait la rotation gauche

La figure (figure III.5.7) montre la rotation dans le sens droite avec angle de 90° :

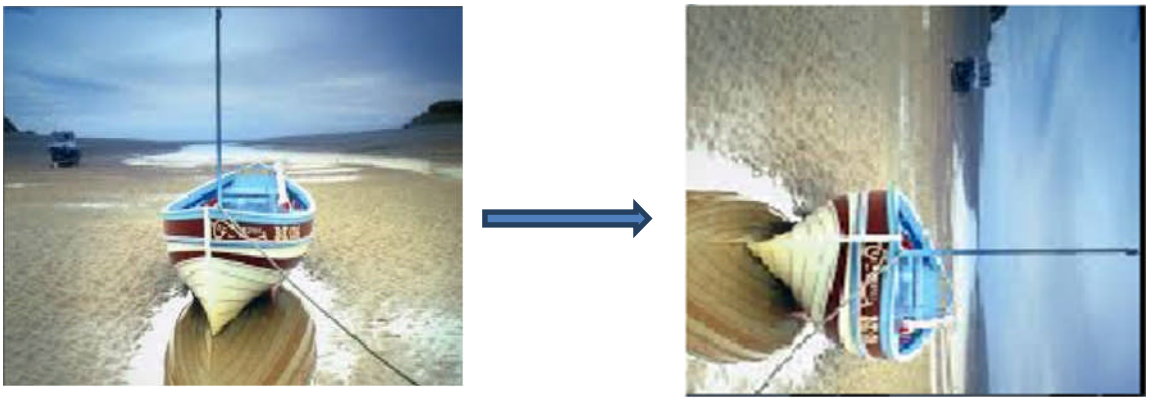


Figure III.5.1- Image original **Figure III.5.10-** image après une rotation droite

L'algorithme qui donne la rotation d'image vers la droite est :

```
public void rotationDroite() {
        BufferedImage transfert = new BufferedImage(source.getHeight(),
        source.getWidth(),source.getType());
        double centreDeRotation = source.getHeight()/2;
        AffineTransformpivoter = AffineTransform.getRotateInstance(Math.toRadians(90),
        centreDeRotation, centreDeRotation);
        AffineTransformOppivoterImage = new AffineTransformOp(pivoter, null);
        pivoterImage.filter(source, transfert);
        source = transfert;
        copieSourceImage();
        historique.ajout("rotationDroite");
}
```

Figure III.5.11- le code source qui fait la rotation droite

III.5.4 Retailler l'image :

C'est une méthode qui permet de changer les pixels (dimensions) d'une image.

Les deux figures (figure III.5.8), (figure III.5.9) montres le rôle de ce bouton :

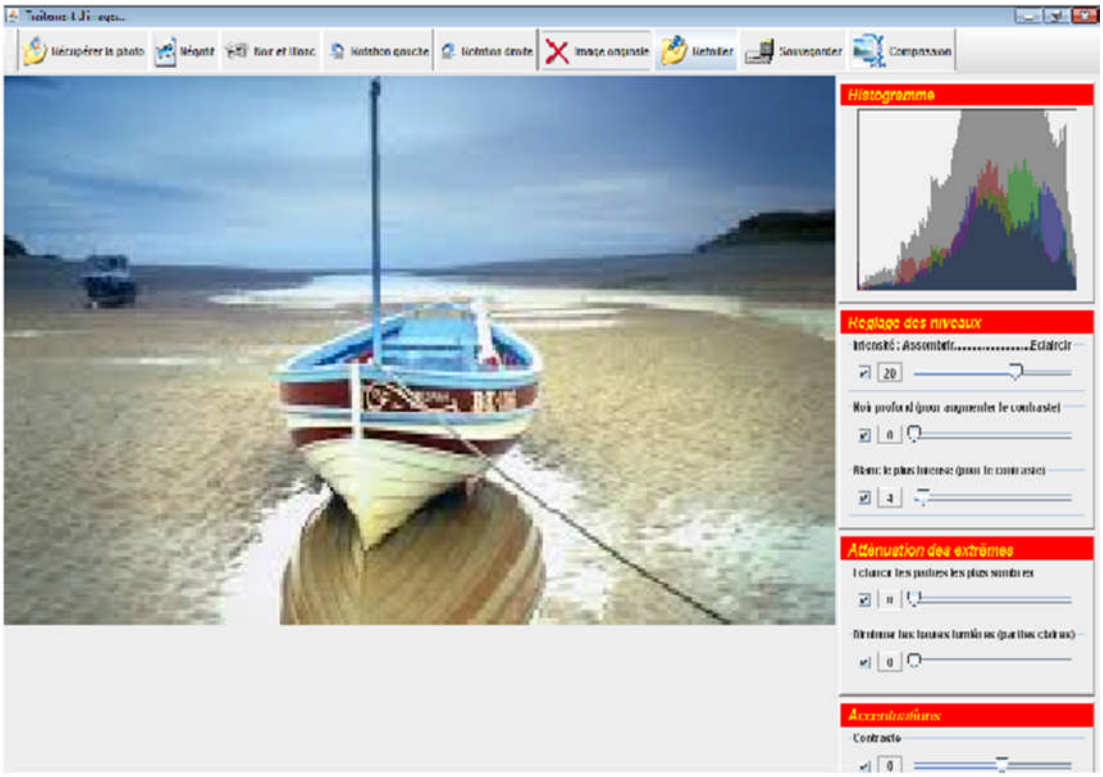


Figure III.5.12- Image non retaillé

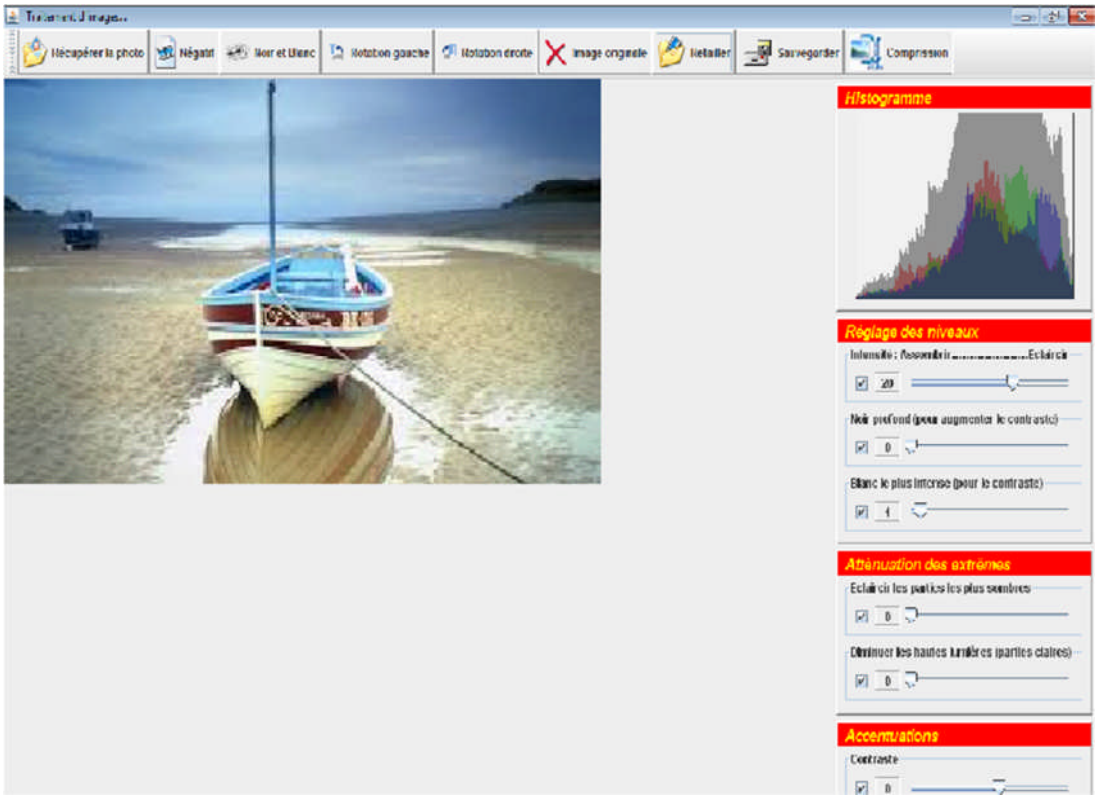


Figure III.5.13- Image retaillé

La méthode qui permet de retailler l'image est :

```
Public void changerTaille () {
        String valeur = JOptionPane.showInputDialog("Nouvelle largeur de l'image ?",
        1024);
        réduction =  Double.parseDouble(valeur)/imageOriginale.getWidth();
        source = new BufferedImage((int)(imageOriginale.getWidth()*réduction),
        (int)(imageOriginale.getHeight()*réduction), imageOriginale.getType());
        AffineTransform retailler = AffineTransform.getScaleInstance(réduction,
        réduction);
        int interpolation = AffineTransformOp.TYPE_NEAREST_NEIGHBOR;
        AffineTransformOpretaillerImage = new AffineTransformOp(retailler,
        interpolation);
        retaillerImage.filter(imageOriginale, source);
        copieSourceImage();
        historique.ajout("changerTaille");
}
```

Figure III .5.14- code source qui permet de retailler l'image

III.5.5 Saturation :

La saturation ou pureté est l'intensité d'une teinte spécifique, L'objectif était de définir des couleurs dans l'espace RVB à l'aide d'un moyen plus intuitif que les valeurs RVB.

La figure (figure III.5.1) donne l'image saturée, et aussi Il est bien défini dans l'histogramme la disparition des couleurs, voire : (figure III.5.10)

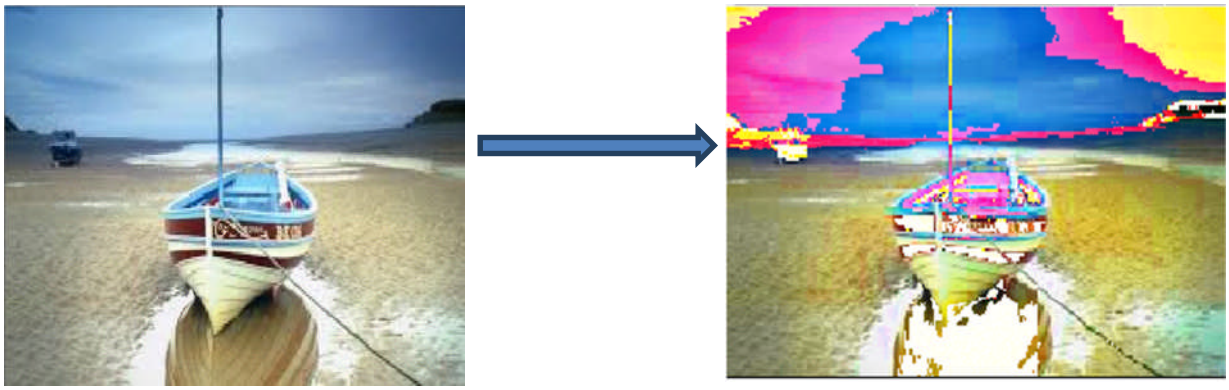
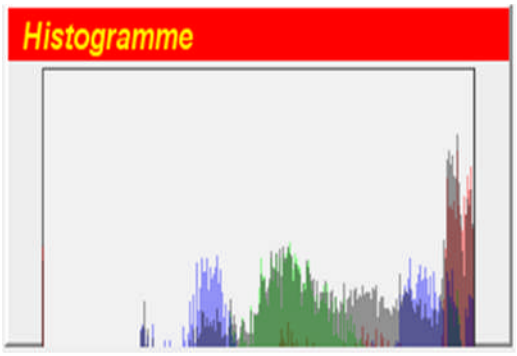


Figure III.5.1- image original **Figure III.5.15-** image saturé



FigureIII.5 .16- Histogramme d'image saturé

La méthode qui permet de saturé l'image est :

```
public void saturation(int valeur) {
        float saturation = (valeur+100)/(float)100F;
        WritableRaster trame = source.getRaster();
        ColorModel modèle = source.getColorModel();

        for (int y=0; y<source.getHeight(); y++)
        for (int x=0; x<source.getWidth(); x++) {
        Object données = trame.getDataElements(x, y, null);
        float[] hsb = new float[3];
        Color.RGBtoHSB(modèle.getRed(données),
        modèle.getGreen(données), modèle.getBlue(données), hsb);
```

```
            Object changement = modèle.getDataElements
            (Color.HSBtoRGB(hsb[0], hsb[1]*saturation, hsb[2]), null);
            trame.setDataElements(x, y, changement);
            }
            calcul ();
}
```

Figure III.5.17- code source qui fait la saturation d'image

III.5.6 intensité

C'est un décalage entre deux technique assombrir et éclaircir, les figure suivante montre ces technique

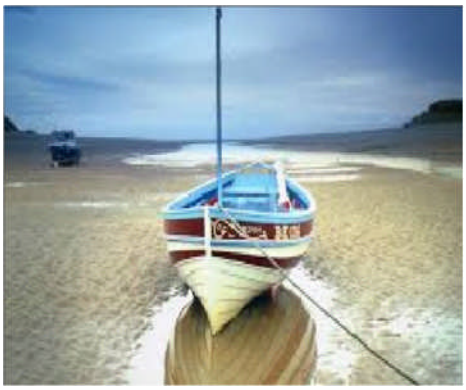


Figure III.5.1 -image original

Figure III .5.18- image assombri

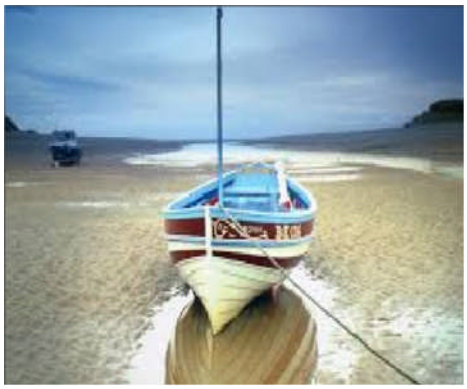


Figure III.5.1- image original

Figure III.5.19- image éclairci

Conclusion générale

Conclusion générale

Dans ce mémoire nous avons atteint notre but, celui de traitement d'une image, le traitement a été menée sous le java, et appliquée sur des images numérique et réelles. On a vu aussi comment il est possible de mettre en œuvre des fonctions de traitements d'images.

J'ai expliqué le code d'une application fenêtrée, qui permet de faire plusieurs opérations sur les images, comme la lecture, l'enregistrement, et le dimensionnement.etc
en général, et la restauration en particulier.

Nous envisageons la réalisation d'une interface, permettant de généraliser l'amélioration de l'image, avec des techniques bien définie, et d'implémenter d'autre technique de traitement, comme la segmentation, et la reconnaissance, et l'Acquisition de l'image.

Le traitement des images est un domaine très actif, Pour terminer, il faut dire que ce traitement en Java n'est pas limité aux services de l'API Java 2D, mais les dépassent pour utiliser l'API Java Advanced Image ou couramment JAI, qui étend les services offerts par Java 2D tout en étant compatible avec celle-ci.

Résumé :

Cette thèse aborde les principaux traitements d’image A travers une approche progressive dans le but d'obtention d'une plus grande lisibilité, avec l’utilisation des technique de restauration Puis l’implémenter en java, et grâce cette étude nous avons bien compris le traitement d'image car notre objet a centré d'apprendre l'utilisation d'orienté objet et java, l'utilisation de NetBeans comme un environnement pour implémenter l’application.

Bibliographie :

[1]- A. Gagalowic, de cours dispensés à l'ESIEA, l'année 2002.

[5]- Pierre Crescenzo Rapport de recherche ISRN I3S/RR, Université de SOPHIA ANTIPOLIS, l'année 28juin 2002-FR

Web graphie :

[2]- Le site : http://www.crdp.ac-grenoble.fr/image/general/general.htm, visitez le 10 avril 2013.

[3]- Le site : http:// www.ac-nice.fr/ia06/iencagnes/file/tice/tuto/**Image**_numerique.pdf, visitez le 15 avril 2013.

[4]- Le site : http://foad.refer.org/IMG/pdf/D226_Chapitre-5.pdf,visitez le 21 avril 2013

[6]- le site : http://images.math.cnrs.fr/Compression-d-image.html, visitez le 22 avril 2013.

[7]- le site : http://slim-boukettaya.developpez.com/tutoriels/traitement-images-java/, visiter le 27 avril 2013.

[8]- le site : http://java.developpez.com/faq/gui/?page=graphique_general_images, visitez le 03mail 2013.

[9]- le site : http://www.techno-science.net/?onglet=glossaire&definition=5346, visitez le 08 mai 2013.

Les Mots clé :

Traitement d'image, orienté objet, java, NetBeans, Restauration.

Listes des Figures

Chapitre I :

Chapitre II :

Chapitre III :

Listes des Figures

Listes des Figures

Liste des tableaux

Chapitre I :