Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Projet libre DI5
2015-2016

Méthode de dégradation image par inclinaison et décalage gaussien, application à la base Manga109

Tuteurs académiques
Mathieu Delalandre

Étudiants
ZHANG Minghui (DI5)

25 septembre 2016

Liste des intervenants

Nom	Mail	Qualité
ZHANG Minghui	minghui.zhang@univ-tours.fr	Étudiant DI5
Mathieu Delalandre	mathieu.delalandre@univ-tours.fr	Tuteur académique, Département infomatique

Avertissement

Ce document a été rédigé par ZHANG Minghui susnommé les auteurs.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Mathieu Delalandre susnommé les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

Les auteurs reconnaissent assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respects des lois ou des droits d'auteur.

Les auteurs attestent que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

Les auteurs attestent ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

Les auteurs attestent que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

Les auteurs reconnaissent qu'ils ne peuvent diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques.

Les auteurs autorisent l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :
ZHANG Minghui, *Méthode de dégradation image par inclinaison et décalage gaussien, application à la base Manga109*, Projet libre DI5, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
    author={Minghui, ZHANG},
    title={Méthode de dégradation image par inclinaison et décalage gaussien, application à la base
        Manga109},
    type={Projet libre DI5},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2015-2016}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V1.0 (2015/11/18)

Table des matières

Table des figures

1 Introduction générale

1 Sujet général

Manga109 est une base de données libre de droit à des fins d'usage re-cherche en traitement d'image et reconnaissance de forme. Elle se compose de 109 magazines Manga et 200 000 pages environ.

Cette base peut être utilisée pour différents problèmes recherche comme l'indexation d'image, la reconnaissance de texte, etc..

Dans ce projet on s'intéressera au problème de détection de duplicata. La détection de duplicata est un problème spécifique de traitement d'image. Une image et son duplicata représentent visuellement un même contenu, mais le codage des images est différent due aux modes de capture, bruit, modes de compression, contraste, formatages des images, etc.

La détection de duplicata vise à mettre en place des méthodes de traitement d'image capables d'identifier, avec une forte précision, les duplicatas.

2 Objectif du projet

La base Manga109, dans sa version de base, ne fournit pas les duplicatas. L'objectif de ce projet vise à développer une méthode automatique permettant la dégradation des images. On s'intérressera ici à un cas spécifique de dégradation par inclinaison et décalage. En particulier le projet visera :

- A implémenter un batch de dégradation des images par la librairie ImageJ en Java.
- Les images seront modifiées par inclinaison (paramètre teta) et décalage (en paramètres dx et dy), les deux traitement seront contrôlés par un paramètre généré aléatoirement via une distribution gaussienne
- Etudier les différentes méthodes permettant la génération aléatoire gaussienne.

2 Présentation des outils

Dans un premier temps, nous allons présenter les deux outils principaux qui seront utilisés dans ce projet.

Figure 1 – *Manga109*

1 ImageJ

ImageJ est un logiciel multiplate-forme et open source dans le domaine de traitement et d'analyse d'images développé par les National Institutes of Health. Il est écrit en Java et permet d'ajouter de nouvelles fonctionnalités via des plugins et macros.

3 Analyse des mécanismes de traitement d'image

1 Contexte

Pour générer la duplicata, on simules la situation de scanner les pages de manga par le scanneur, normalement, on met la page dans le scanneur correctement, le côté de papier et le côté du scanneur est aligner. Mais pendant la processus de génération la duplicata, la papier n'est pas toujours bien correspondant mettre dans le scanneur, cela entraînera l'inclinaison et la décalage de la résultat de scanner. Pour les deux traitement, on utilises la méthode d'inclinaison et décalage pour simuler.

l'opération d'inclinaison, ça dépend la dégrée, et l'opération de décalage, ça dépend la déplacement de la direction de X et Y, il va faire plusieurs fois de scanner, donc tous les l'opération de scanner ensemble, on peut utiliser comme un événement aléatoire. La distribution de gaussienne peut bien simuler cette processus aléatoire. la distribution gaussienne est une distribution de probabilité, on peut utiliser le nombre aléatoire pour contrôler les deux traitement, simuler la processus de faire la duplicata.

2 Inclinaison et Décalage

La fonction rotation simules l'inclinaison de papier, quand on indiques une dégrée, l'image tourne autour de son angle de centre. La fonction ci-dessous Figure 1 :

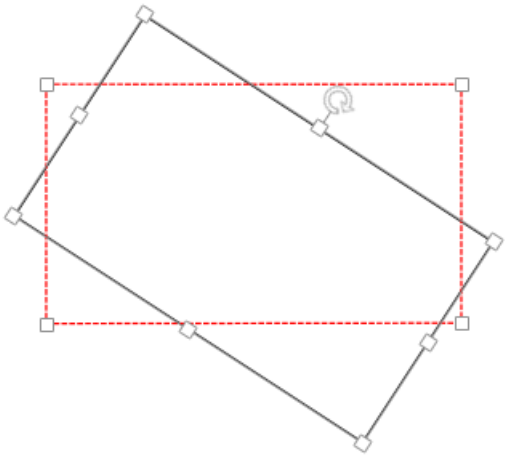

Figure 1 – *Rotation*

Pour réaliser la fonction de décalage, on considéres les sens d'abord, la page il y a 4 angle, donc la papier peut déplacer 4 sens, ces sont en haut à gauche, en haut à droit, en bas à gauche et en bas à droit, il y a quarte conditions :

* deltaX >= 0 et deltaY >= 0 (voir Figure 2)
* deltaX < 0 et deltaY > 0 (voir Figure 3)
* deltaX > 0 et deltaY < 0 (voir Figure 4)
* deltaX < 0 et deltaY < 0 (voir Figure 5)

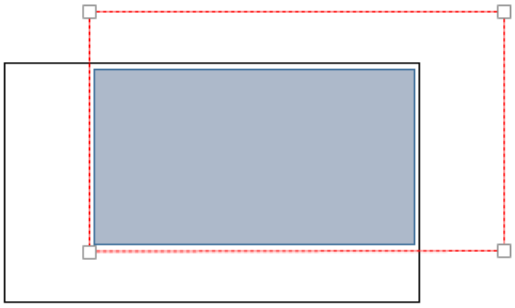

Figure 2 – *deltaX >= 0 et deltaY >= 0*

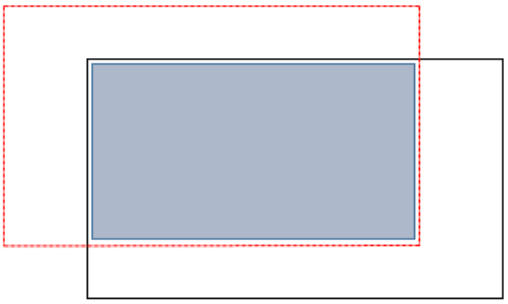

Figure 3 – *deltaX < 0 et deltaY > 0*

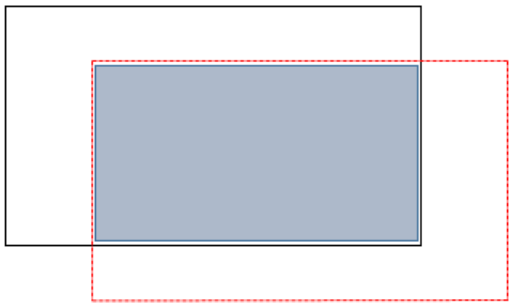

Figure 4 – *deltaX > 0 et deltaY < 0*

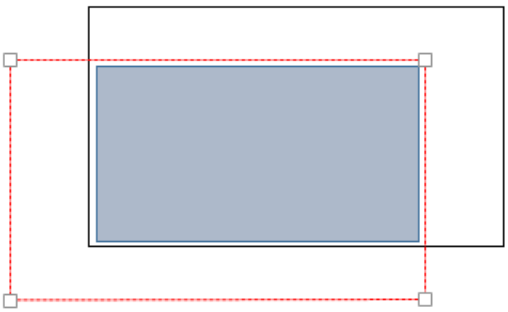

Figure 5 – *deltaX < 0 et deltaY < 0*

3 Distribution Gaussienne

On sait que la processus de faire la duplicata est une processus de aléatoire, en l'ensemble, ce processus aléatoire est subordonné à la distribution gaussienne. Donc chaque fois, on utilise le nombre aléatoire générer par la distribution gaussienne pour contrôle les traitement de l'inclinaison et la décalage. Donc j'ai étudié les différente méthodes permettant la génération aléatoire gaussienne.

3.1 Théorème central limite

Le théorème central limite établit la convergence en loi de la somme d'une suite de variables aléatoires vers la loi normale. Intuitivement, ce résultat affirme que toute somme de variables aléatoires indépendantes et identiquement distribuées tend vers une variable aléatoire gaussienne.

Soit X1, X2, ... Xn sont indépendantes et identiquement distribuées séquence variable aléatoire, signifie μ, variance σ^2, puis

$$Z_n = \frac{X_1 + X_2 + \cdots + X_n - n\mu}{\sigma\sqrt{n}}$$

Figure 6 – *distribution N(0, 1)*

Zn a la distribution de N(0, 1), c'est à dire que quand n est infini, donc

$$P\left\{\frac{X_1 + X_2 + \cdots + X_n - n\mu}{\sigma\sqrt{n}} \leq x\right\} \to \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$$

Figure 7 – *probabilité de distribution N(0, 1) approchée la distribution gaussienne*

En d'autres termes, la distributon de somme de n événement mutuellement indépendantes similaire à la distribution gaussienne. n est plus grand, une meilleure approximation. L'exemple est ci-dessous :

On peut voir, quand n égale 1, il est loi uniforme. Lorsque n augmente, l'histogramme du contour proche de la distribution gaussienne.

3.2 Box-Muller

La méthode de Box-Muller consiste à générer des paires de nombres aléatoires à distribution normale centrée réduite, à partir d'une source de nombres aléatoires de loi uniforme. On peut également utiliser

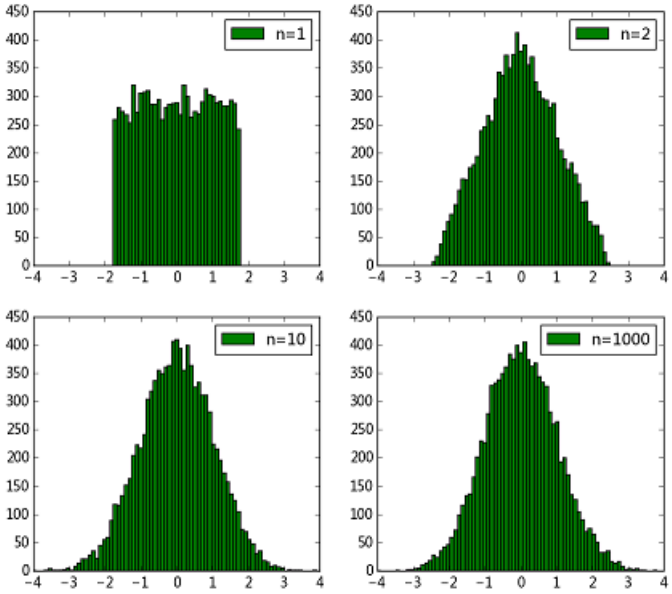

Figure 8 – *Les résultats de différente valeur de n*

la méthode de la transformée inverse pour générer des nombres normalement distribués, la méthode de Box-Muller a été mise au point pour être algorithmiquement plus efficace.

Soit U1 et U2 sont deux variables aléatoires indépendantes uniformément distribuée dans [0, 1], soient

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

et

$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$

Figure 9 – *algorithme de Box-Muller*

Alors Z0 et Z1 sont des variables aléatoires indépendantes suivant une loi normale de variance 1.

Si on teste en plusieurs fois, algorithme de Box-Muller est plus rapide. Le résultat est ci-dessous :

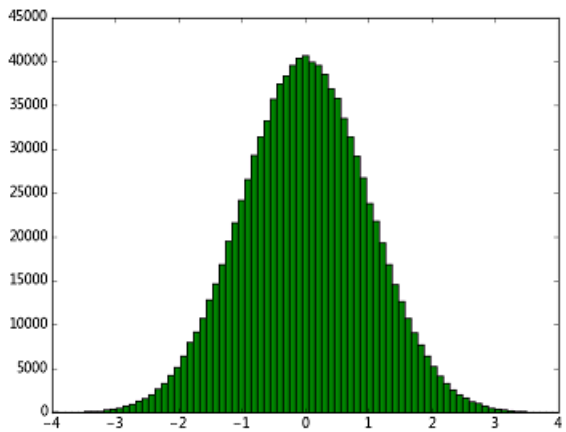

Figure 10 – *Le résultat d'algorithme Box-Muller*

3.3 Ziggurat

L'algorithme de Ziggurat est plus pratique, la méthode est comme ça. Utiliser une fonction q(x) et quantité constante k, pouvoir mettre p(x) dessous la fonction kq(x).

En construisant une fonction très subtile q(x), l'algorithme de Ziggurat nous propose une performance efficace. On utilise plusieurs rectangles empilés, assurant que les zones ombrées sont toujours plus petit. C'est aussi la raison de sa efficacité.

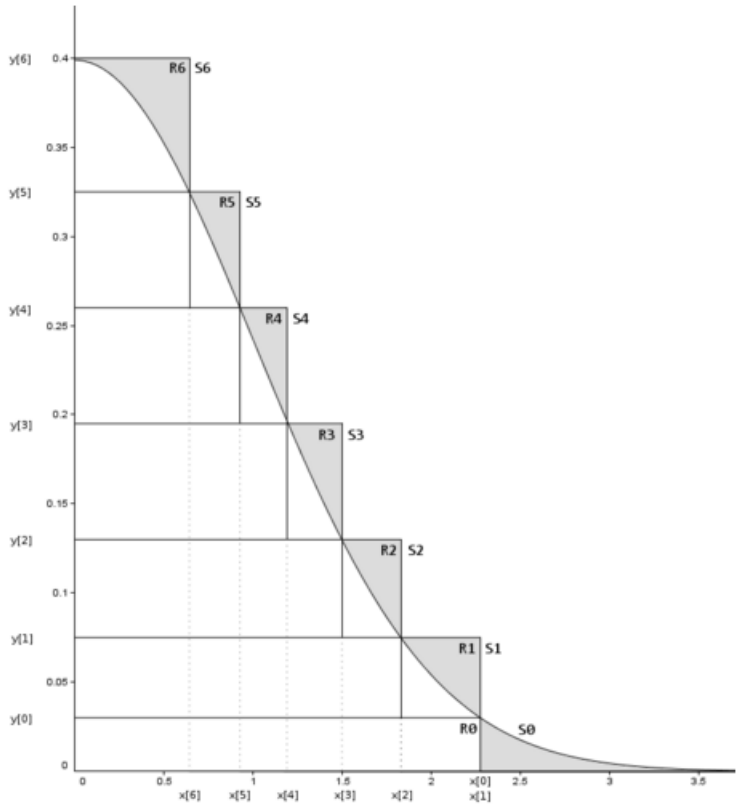

Figure 11 – *La méthode de Ziggurat*

Je vais vous expliquer la figure :

- on utilise R[i] pour représenter un rectangle, x[i]est la limite droite de R[i], y[i] est sa limite haute. S[i] représente une découpage, alors que S[0]=R[0]+tail si i=0, sinon S[i]=R[i].
- sauf que R[0], les surfaces de chaque rectangle est égale à la valeur A. Et la surface de R[0]=A-tail. Cela garantit que la probabilité de récupérer les échantillons (x, y) dans chaque découpage est équivalente.
- Quant on récupère une échantillon dans un R[i] sélectionné, le y se situera en dessous de la courbe si x <x [i + 1], il reçoit le x. Si x<x[i+1], il faut avoir d'autre jugement. C'est à dire continuer l'algorithme. Avec la même façon, les échantillons dans R[0] et tail nécessitent une manipulation spéciale

Ici, pour faciliter l'explication de l'algorithme, on a utilisé seulement quelque rectangles. Lorsque dans le vrai programme, on implémente cette procédure normalement avec 128 ou 256 rectangles.

4 Détail d'implémentation et résultats d'expérimentation

1 distribution gaussienne

J'ai utilisé deux méthode pour réaliser la distribution gaussienne, ces sont la méthode de central limite et la méthode de Box-Muller. Les détail est ci-dessous.

1.1 Méthode de central limite

```
double temp = 12;
double x = 0;
do{
    for (int i = 0; i < temp; i++)
      x = x + (Math.random());
  x = (x - temp / 2) / (Math.sqrt(temp / 12));
  x = a + x * Math.sqrt(b);
}while(x <= 0);

return x;
```

1.2 Méthode de Box-Muller

```
double pi = 3.1415926535;
double r1 = Math.random();
double r2 = Math.random();
double z = 0;
do{
  double u = Math.sqrt((-2) * Math.log(r1)) * Math.cos(2 * pi * r2);
  z = a + u * Math.sqrt(b);
}while(z <= 0);

return (z);
```

Quand on utilise la méthode pour générer le nombre aléatoire x en la distribution gaussienne, on vais utiliser cette paramètre pour contrôle les deux traitement, et utiliser équation +/-(1-x)*paramètres, ici les paramètre sont teta de l'opération d'inclinaison et dx, dy de l'opération de décalage.

2 Les fonctions

Tout d'abord, on choisit une répertoire, après lire tous les fichiers dans cette répertoire et stocker dans une liste, après on fait les opérations sur tous les images.

La fonction batch :

```
String[] list = new File(dir1).list();
if(list == null) return;
for(int i = 0; i < list.length; i++){
  IJ.showProgress(i, list.length);
  boolean isDir = (new File(dir1 + list[i])).isDirectory();
  if (!isDir && !list[i].startsWith(".")) {
    ImagePlus img = IJ.openImage(dir1+list[i]);
    //traitement
    ip = img.getProcessor();
    double aleatoir = normalRandom(moyen, variance);
    ip = traitement(aleatoir, teta, dx, dy);
    img.setProcessor(img.getTitle(), ip);
    IJ.saveAs(format, dir2+list[i]);
  }
}
```

La fonction inclinaison :

```
ip.rotate(teta);
```

La fonction décalage :

```
if(deltaX >= 0 && deltaY >= 0){
  //get pixel cible
  for(int i = 0; i < width - deltaX; i++){
    for(int j = 0; j < height - deltaY; j++){
      cible.add(temp[i][j]);
    }
  }
  //put pixel cible sur l'image blanc
  for(int i = deltaX; i < width; i++){
    for(int j = deltaY; j < height; j++){
      ip.putPixel(i, j, (int)cible.get(flag));
      flag++;
    }
  }

}else if(deltaX <= 0 && deltaY >= 0){
  deltaX = Math.abs(deltaX);
  for(int i = deltaX; i < width; i++){
    for(int j = 0; j < height - deltaY; j++){
      cible.add(temp[i][j]);
    }
  }
  for(int i = 0; i < width - deltaX; i++){
    for(int j = deltaY; j < height; j++){
      ip.putPixel(i, j, (int)cible.get(flag));
      flag++;
    }
  }
}else if(deltaX >= 0 && deltaY <= 0){
  deltaY = Math.abs(deltaY);
```

```
31    for(int i = 0; i < width - deltaX; i++){
32      for(int j = deltaY; j < height; j++){
33        cible.add(temp[i][j]);
34      }
35    }
36    for(int i = deltaX; i < width; i++){
37      for(int j = 0; j < height - deltaY; j++){
38        ip.putPixel(i, j, (int)cible.get(flag));
39        flag++;
40      }
41    }
42  }else{
43    //deltaX < 0 && deltaY < 0
44    deltaX = Math.abs(deltaX);
45    deltaY = Math.abs(deltaY);
46    for(int i = deltaX; i < width; i++){
47      for(int j = deltaY; j < height; j++){
48        cible.add(temp[i][j]);
49      }
50    }
51    for(int i = 0; i < width - deltaX; i++){
52      for(int j = 0; j < height - deltaY; j++){
53        ip.putPixel(i, j, (int)cible.get(flag));
54        flag++;
55      }
56    }
57  }
```

ici, ces sont les quatre sens, en haut à gauche, en haut à droit, en bas à gauche et en bas à droit. Chaque fois, la programme prend chaque pixel et stocker dans la liste, après les opérations, il affiche dans la nouvelle image blanc en même taille.

3 Les résultat

J'ai crée une user interface pour faire l'opération et indiquer les paramètre. ci-dessous :

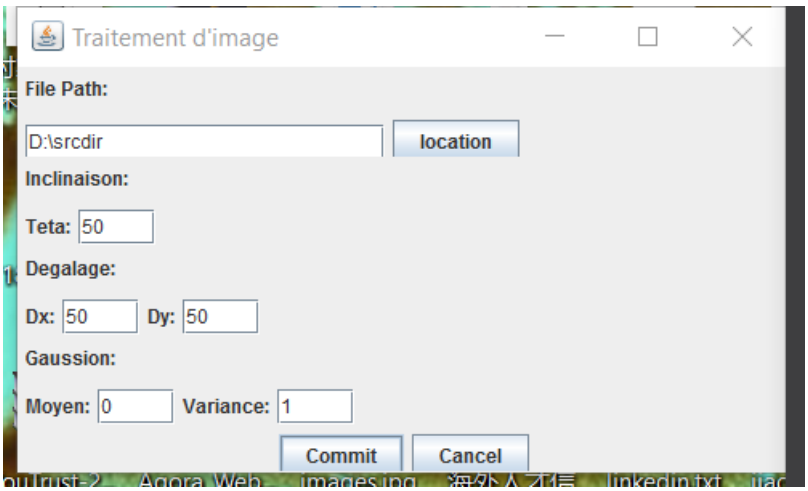

Figure 1 – *User interface*

On peut choisir la répertoire de les images, après indiquer les paramètres, teta est la paramètre de fonction inclinaison, dx et dy sont la paramètres de fonction de décalage, moyen et variance sont les paramètres de distribution gaussienne. Ici on peut voir un exemple, image original et image duplicata :

On peut voir les deux image, l'image duplicata déplace par la sens de en haut à droit et rotation en le sens indirect.

Figure 2 – *Image original*

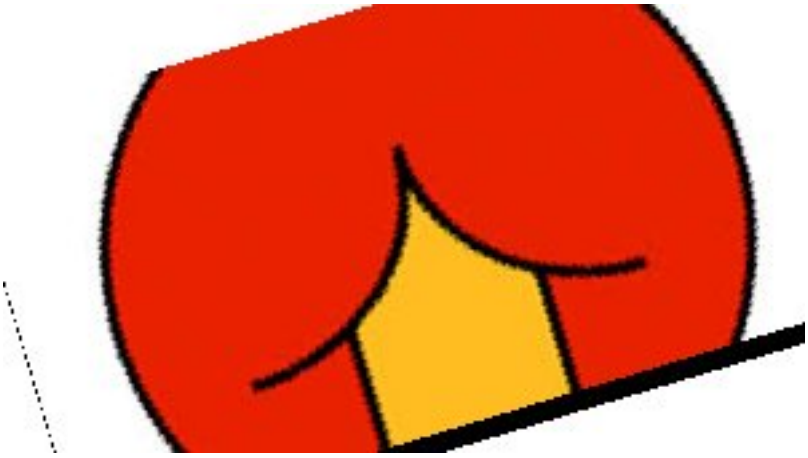

Figure 3 – *Image duplicata*

5 Conclusion

Dans le domaine de traitement d'image, il y a quand même beaucoup de chose à apprendre comme le bruit, le contraste, le niveau de compression et les types d'image, etc. On a bien compris la façon de générer des duplicata.

Dans ce projet, on a utilisé le logiciel ImageJ pour faire les traitements d'image. Donc on a étudié beaucoup de chose pour bien comprendre comment ça marche. L'inclinaison et décalage sont simple et la base de façon de traiter les images. On sait que la processus de faire une duplicata est une processus aléatoire, et il suffisant la distribution gaussienne. La gaussienne est une distribution qui plus base et important dans la domaine de probabilité. Et il pratique dans plusieurs domaine.

Pour moi, je suis plus intéressant dans la domaine de machine learning, et il utilise beaucoup de connaîssance de probabilité et la distribution gaussienne aussi. cette distribution est plus improtant dans la vie aussi, beaucou de événement aléatoire suffisant la distribution gaussienne.

A la fin, on voudrait remercier monsieur Delalandre. Merci de votre aide dans le projet.

Annexes

```java
package com.projet.manga;

import java.io.File;
import java.util.ArrayList;

import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import ij.IJ;
import ij.ImagePlus;
import ij.WindowManager;
import ij.plugin.PlugIn;
import ij.process.ImageProcessor;

public class TraitementImages extends JFrame{

  private ImageProcessor ip = null;
  private String srcDir = null;
  private String destDir = "D:\\test\\";

  // definir les components
  JLabel jlPath, jlInclinaison, jlTeta, jlDegalage, jlDx, jlDy, jlGaussion, jlMoyen, ↩
        jlVariance;
  JTextField jtfPath, jtfTeta, jtfDx, jtfDy, jtfMoyen, jtfVariance;
  JButton jbFile, jbCommit, jbCancel;
  JPanel jp1, jp2, jp3, jp4, jp5, jp6, jp7, jp8, jp9;

  public TraitementImages(){
    // initialiser les components
    jlPath = new JLabel("File Path:");
    jlInclinaison = new JLabel("Inclinaison:");
    jlTeta = new JLabel("Teta:");
    jlDegalage = new JLabel("Degalage:");
    jlDx = new JLabel("Dx:");
    jlDy = new JLabel("Dy:");
    jlGaussion = new JLabel("Gaussion:");
    jlMoyen = new JLabel("Moyen:");
    jlVariance = new JLabel("Variance:");

    jtfPath = new JTextField(20);
    jtfTeta = new JTextField(4);
    jtfDx = new JTextField(4);
    jtfDy = new JTextField(4);
    jtfMoyen = new JTextField(4);
    jtfVariance = new JTextField(4);

    jbFile = new JButton("location");
    jbCommit = new JButton("Commit");
    jbCancel = new JButton("Cancel");

    jp1 = new JPanel(new FlowLayout(0));

    jp2 = new JPanel(new FlowLayout(0));
    jp3 = new JPanel(new FlowLayout(0));
```

```
60      jp4 = new JPanel(new FlowLayout(0));
61      jp5 = new JPanel(new FlowLayout(0));
62      jp6 = new JPanel(new FlowLayout(0));
63      jp7 = new JPanel(new FlowLayout(0));
64      jp8 = new JPanel(new FlowLayout(0));
65      jp9 = new JPanel();
66
67      // configurer le layout
68      this.setLayout(new GridLayout(9, 1));
69
70      jp1.add(jlPath);
71
72      jp2.add(jtfPath);
73      jp2.add(jbFile);
74
75      jp3.add(jlInclinaison);
76
77      jp4.add(jlTeta);
78      jp4.add(jtfTeta);
79
80      jp5.add(jlDegalage);
81
82      jp6.add(jlDx);
83      jp6.add(jtfDx);
84      jp6.add(jlDy);
85      jp6.add(jtfDy);
86
87      jp7.add(jlGaussion);
88
89      jp8.add(jlMoyen);
90      jp8.add(jtfMoyen);
91      jp8.add(jlVariance);
92      jp8.add(jtfVariance);
93
94      jp9.add(jbCommit);
95      jp9.add(jbCancel);
96
97      this.add(jp1);
98      this.add(jp2);
99      this.add(jp3);
100     this.add(jp4);
101     this.add(jp5);
102     this.add(jp6);
103     this.add(jp7);
104     this.add(jp8);
105     this.add(jp9);
106
107     // ajouter surveiller de button
108
109     jbFile.addActionListener(new ActionListener() {
110       @Override
111       public void actionPerformed(ActionEvent e) {
112         JFileChooser chooser = new JFileChooser();
113         chooser.setDialogTitle("Choisir le repertoire");
114         chooser.setMultiSelectionEnabled(false);
115         chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
116         // FileNameExtensionFilter filter = new
117         // FileNameExtensionFilter("jpg", "jpg");
118         // chooser.setFileFilter(filter);
119
120         chooser.showOpenDialog(null);
```

```
121         File file = chooser.getSelectedFile();
122         jtfPath.setText(file.getAbsolutePath());
123         srcDir = file.getAbsolutePath();
124         srcDir += "\\";
125 
126       }
127 
128     });
129 
130     jbCommit.addActionListener(new ActionListener() {
131       @Override
132       public void actionPerformed(ActionEvent e) {
133         double teta = Float.parseFloat(jtfTeta.getText());
134         int dx = Integer.parseInt(jtfDx.getText());
135         int dy = Integer.parseInt(jtfDy.getText());
136         double moyen = Float.parseFloat(jtfMoyen.getText());
137         double variance = Float.parseFloat(jtfVariance.getText());
138 
139         System.out.println(teta + " " + dx + " " + dy + " " + moyen + " " + variance);
140         batchTraitements(srcDir, destDir, "JPEG", teta, dx, dy, moyen, variance);
141         /*batchTraitements(srcDir, destDir, "JPEG", 0.5, 50, 50, 0, 1);*/
142 
143         //ajouter
144         JOptionPane.showConfirmDialog(null, "Success!", "Information", ↩
                JOptionPane.YES_NO_OPTION);
145 
146       }
147 
148     });
149 
150     jbCancel.addActionListener(new ActionListener() {
151       @Override
152       public void actionPerformed(ActionEvent e) {
153         jtfTeta.setText("");
154         jtfDx.setText("");
155         jtfDy.setText("");
156         jtfMoyen.setText("");
157         jtfVariance.setText("");
158 
159       }
160 
161     });
162 
163     this.setSize(500, 300);
164     this.setTitle("Traitement d'image");
165     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
166     this.setVisible(true);
167     this.setLocation(300, 300);
168   }
169 
170   public void batchTraitements(String dir1, String dir2, String format, double teta, int ↩
          dx, int dy, double moyen, double variance) {
171     IJ.log("\\Clear");
172     IJ.log("Converting to " + format);
173     IJ.log("dir1: " + dir1);
174     IJ.log("dir2: " + dir2);
175     String[] list = new File(dir1).list();
176     if(list == null) return;
177     for(int i = 0; i < list.length; i++){
178       IJ.showProgress(i, list.length);
179       IJ.log((i+1) + ": " + list[i] + "  " + WindowManager.getImageCount());
```

```
180       IJ.showStatus(i + "/" + list.length);
181       boolean isDir = (new File(dir1 + list[i])).isDirectory();
182       if (!isDir && !list[i].startsWith(".")) {
183         ImagePlus img = IJ.openImage(dir1+list[i]);
184         System.out.println("path... : " + dir1+list[i]);
185 
186         //traitement
187         ip = img.getProcessor();
188 
189         double aleatoir = normalRandom(moyen, variance);
190         IJ.log("distribution gaussienne: " + aleatoir);
191         ip = traitement(aleatoir, teta, dx, dy);
192 
193         img.setProcessor(img.getTitle(), ip);
194 
195         WindowManager.setTempCurrentImage(img);
196         IJ.saveAs(format, dir2+list[i]);
197       }
198     }
199     IJ.showProgress(1.0);
200     IJ.showStatus("");
201   }
202 
203   //distribution gaussienne
204   public double normalRandom1(double a, double b) {
205     double temp = 12;
206     double x = 0;
207     do{
208       for (int i = 0; i < temp; i++)
209         x = x + (Math.random());
210       x = (x - temp / 2) / (Math.sqrt(temp / 12));
211       x = a + x * Math.sqrt(b);
212     }while(x <= 0);
213 
214     return x;
215   }
216 
217   public double normalRandom2(double a, double b) {
218     double pi = 3.1415926535;
219     double r1 = Math.random();
220     double r2 = Math.random();
221     double z = 0;
222     do{
223       double u = Math.sqrt((-2) * Math.log(r1)) * Math.cos(2 * pi * r2);
224       z = a + u * Math.sqrt(b);
225     }while(z <= 0);
226 
227     return (z);
228   }
229 
230   public double normalRandom3(double a, double b) {
231     double f = 0;
232     double c0 = 2.515517, c1 = 0.802853, c2 = 0.010328;
233     double d1 = 1.432788, d2 = 0.189269, d3 = 0.001308;
234     double w, z;
235     double r = Math.random();
236     if (r <= 0.5)
237       w = r;
238     else
239       w = 1 - r;
240     if ((r - 0.5) > 0)
```

```
241       f = 1;
242     else if ((r - 0.5) < 0)
243       f = -1;
244     do{
245       double y = Math.sqrt((-2) * Math.log(w));
246       double x = f * (y - (c0 + c1 * y + c2 * y * y) / (1 + d1 * y + d2 * y * y + d3 * y
                * y * y));
247       z = a + x * Math.sqrt(b);
248     }while(z <= 0);
249
250     return (z);
251   }
252
253   public double normalRandom(double a, double b) {
254     /*double r = Math.random() * 9;
255     switch ((int) r / 3) {
256       case 0:
257         IJ.log("normalRandom1");
258         return normalRandom1(a, b);
259       case 1:
260         IJ.log("normalRandom2");
261         return normalRandom2(a, b);
262       case 2:
263         IJ.log("normalRandom3");
264         return normalRandom3(a, b);
265     }
266     return 0.0;*/
267     return normalRandom1(a, b);
268   }
269
270   public ImageProcessor decalage(int deltaX, int deltaY){
271     int width = ip.getWidth();
272     int height = ip.getHeight();
273     int [][] temp;
274     int [] imagePixels;
275     ArrayList imagePix = new ArrayList();
276
277     temp = new int[width][height];
278     for(int i = 0; i < width; i++){
279       for(int j = 0; j < height; j++){
280         temp[i][j] = ip.getPixel(i, j);
281       }
282     }
283
284     for(int i = 0; i < width; i++){
285       for(int j = 0; j < height; j++){
286         ip.putPixel(i, j, 0);
287       }
288     }
289
290     //test copy
291     /*for(int i = 0; i < temp.length; i++){
292       for(int j = 0; j < temp[0].length; j++){
293         ip.putPixel(i, j, temp[i][j]);
294       }
295     }*/
296     ArrayList cible = new ArrayList();
297     int flag = 0;
298
299     if(deltaX >= 0 && deltaY >= 0){
300       //get pixel cible
```

```
301       for(int i = 0; i < width - deltaX; i++){
302         for(int j = 0; j < height - deltaY; j++){
303           cible.add(temp[i][j]);
304         }
305       }
306       //put pixel cible sur l'image blanc
307       for(int i = deltaX; i < width; i++){
308         for(int j = deltaY; j < height; j++){
309           ip.putPixel(i, j, (int)cible.get(flag));
310           flag++;
311         }
312       }
313
314       /*for(int i = 0; i < deltaX; i++){
315         for(int j = 0; j < height; j++){
316           ip.putPixel(i, j, 0);
317         }
318       }
319
320       for(int i = 0; i < width; i++){
321         for(int j = 0; j < deltaY; j++){
322           ip.putPixel(i, j, 255);
323         }
324       }*/
325
326     }else if(deltaX <= 0 && deltaY >= 0){
327       deltaX = Math.abs(deltaX);
328       for(int i = deltaX; i < width; i++){
329         for(int j = 0; j < height - deltaY; j++){
330           cible.add(temp[i][j]);
331         }
332       }
333       for(int i = 0; i < width - deltaX; i++){
334         for(int j = deltaY; j < height; j++){
335           ip.putPixel(i, j, (int)cible.get(flag));
336           flag++;
337         }
338       }
339     }else if(deltaX >= 0 && deltaY <= 0){
340       deltaY = Math.abs(deltaY);
341       for(int i = 0; i < width - deltaX; i++){
342         for(int j = deltaY; j < height; j++){
343           cible.add(temp[i][j]);
344         }
345       }
346       for(int i = deltaX; i < width; i++){
347         for(int j = 0; j < height - deltaY; j++){
348           ip.putPixel(i, j, (int)cible.get(flag));
349           flag++;
350         }
351       }
352     }else{
353       //deltaX < 0 && deltaY < 0
354       deltaX = Math.abs(deltaX);
355       deltaY = Math.abs(deltaY);
356       for(int i = deltaX; i < width; i++){
357         for(int j = deltaY; j < height; j++){
358           cible.add(temp[i][j]);
359         }
360       }
361       for(int i = 0; i < width - deltaX; i++){
```

```
362         for(int j = 0; j < height - deltaY; j++){
363           ip.putPixel(i, j, (int)cible.get(flag));
364           flag++;
365         }
366       }
367     }
368 
369     return ip;
370   }
371 
372   public ImageProcessor inclinaison(double teta){
373     ip.rotate(teta);
374     return ip;
375   }
376 
377   public ImageProcessor traitement(double variable, double teta, int dx, int dy){
378     teta = operator() * (Math.random() * teta);
379     dx = operator() * ((int)(Math.random() * dx));
380     dy = operator() * ((int)(Math.random() * dy));
381     System.out.println(teta + " " + dx + " " + dy);
382     ip = decalage((int)(dx * (1 - variable)), (int)(dy * (1 - variable)));
383     ip = inclinaison(teta * variable);
384     return ip;
385   }
386 
387   public int operator(){
388     if(Math.random() > 0.5){
389       return 1;
390     }else{
391       return -1;
392     }
393   }
394 
395   public static void main(String args[]) {
396     /*JFileChooser fileChooser = new JFileChooser(".");
397     fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
398     fileChooser.setDialogTitle("open directory ...");
399     int ret = fileChooser.showOpenDialog(null);
400     String srcDir = null;
401     if(ret == JFileChooser.APPROVE_OPTION){
402       srcDir = fileChooser.getSelectedFile().getAbsolutePath();
403       srcDir += "\\";
404       System.out.println("path:" + srcDir);
405     }
406     String destDir = "D:\\test\\";
407     //new Batch_Converter_().convert(srcDir, destDir, "TIFF");
408     System.exit(0);*/
409     new TraitementImages();
410   }
411 
412 }
```

A Webographie

[0] Introduction ImageJ, UFE Observatoire de Paris, 2014
[1] http ://blog.csdn.net/rns521/article/details/6953591
[2] http ://blog.sina.com.cn/s/blog4cbb6b0c0100080a.html
[3] http ://www.tuicool.com/articles/ZFv6Rbe
[4] http ://blog.csdn.net/amberroom/article/details/6527805
[5] http ://www.zhihu.com/question/29971598