Mini-projet MOP : visionneur d'images astronomiques

8 février 2012

1 Introduction

Le but de ce sujet est de développer un visionneur d'images astronomiques qui propose des transformations de base sur ces images

2 Mais, pourquoi ce genre de matrice ?

Une des premières raisons de l'utilisation de cette représentation est de pouvoir représenter des matrices avec une valeur redondante : l'idée est de ne pas représenter les cases contenant cette valeur.

Une application pratique de ces matrices est la représentation d'images avec une couleur dominante, comme ... des images astronomiques : un fond noir avec des étoiles. Dans ce genre d'image, la matrice contiendra uniquement les pixels qui ne sont pas noirs, ce qui peut entraîner un gain de place considérable.

3 Matrice de compression

Implémenter la matrice de compression vue en TD (et normalement déjà implémentée dans le TP précédent).

4 Sauvegarde/chargement

Vous devez aussi implémenter la sauvegarde et le chargement d'images à partir d'un format défini en XML comme l'exemple suivant :

```
<image>
  <row index=‘‘1’’>
    </pix column=‘‘5’’ color=‘‘240’’>
    </pix column=‘‘5’’ color=‘‘200’’>
    ...
  </row>
```

FIGURE 1 – L'image la plus à gauche est l'image orignale, puis dans l'ordre : rotation à 90°, 180° et 270°

```
  <row index=‘‘7’’>
    </pix column=‘‘1’’ color=‘‘10’’>
    </pix column=‘‘3’’ color=‘‘20’’>
    ...
  </row>
  ...
</image>
```

5 Transformations

Cette section présente des transformations courantes sur la manipulation d'images.

5.1 Rotations

Les rotations sont des tranformations d'images très utilisées : une image de l'espace n'a pas vraiment d'orientation, mais permettre de les pivoter permet de les visualiser de façon plus *esthétique*. C'est le cas des images de la galaxie M104, plus connue sous le nom de galaxie du Sombréro ...

Pour cette fonctionnalité, nous ne ferons que les rotations montrées sur la figure 1, à savoir 90°, 180° et 270°.

5.2 Miroir

Cette transformation permet d'obtenir une image comme si l'image originale était vue dans un miroir : on opère une symétrie par rapport aux bords de l'image. La figure 2 donne le résultat des transformations attendues.

5.3 Elimination du bruit

Cet algorithme à développer permet d'éliminer les défauts des appareils ou du temps : c'est l'élimination du bruit. En effet, les images brutes provenant d'appareils photos ou même de téléscopes n'offrent pas un noir total sur le fond spatial. Perturbations atmosphériques, pollutions lumineuses (hé oui, une ville éclaire loin si le temps n'est pas tout à fait clair), ... rendent un fond presque noir, mais pas totalement. La figure 3 montre la différence entre une image

FIGURE 2 – La figure de gauche est l'original, celle du milieu est l'image miroir par un bord latéral et celle de droite est obtenue par miroir sur le bord haut ou bas.

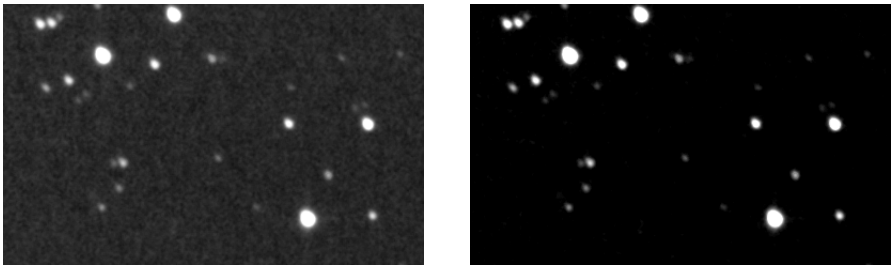

FIGURE 3 – A gauche, l'image originale avec du bruit et à droite, l'image corrigée.

brute et la même image débarassée de ses défauts par élimination du bruit. La différence à été exagérée pour être plus visible (image au voisinage de la galaxie d'Andromède M31).

Cependant, sur des images très détaillées comme celles des grands téléscopes, les pixels qui peuvent paraître être du bruit sont en fait des étoiles très lointaines. Pour simplifier, nous considérerons ces étoiles comme du bruit dans l'image.

Indication : L'idée de cet algorithme est de supprimer les pixels qui ont une valeur en dessous d'un certain seuil défini par l'utilisateur.

6 Transformations avancées

Afin de développer des transformations plus complexes, nous introduisons de nouvelles fonctionnalités.

6.1 Négatif

Cette fonctionnalité permet de donner le négatif d'une image. Le négatif est une inversion des couleurs (ou des niveaux de gris dans notre cas) d'une image comme le montre l'image de la figure 4.

Il y a deux possibilités pour cet algorithme :

La méthode brute : il s'agit de prendre chaque pixel et de recomposer une

FIGURE 4 – A gauche, l'image originale et à droite, son négatif.

image avec la couleur inversée. Cette méthode est simple à implémenter, mais comporte un inconvénient majeur : en nous basant sur une image astronomique, tous les pixels noirs deviennent blancs et seront représentés dans la matrice. Nous aurons donc une matrice très chargée.

Méthode plus fine (ou élégante) : cette méthode permet de spécifier, pour une image, la base de son échelle de couleur. La base initiale pour une image normale est 0 (noir) : les pixels noirs ne sont pas représentés. Le négatif permet de changer cette échelle de valeur à 255. Ainsi, la valeur récupérée pour un pixel dépend non seulement de la teinte du pixel mais aussi de la base de l'échelle de valeur choisie.

A vous d'implémenter les deux possibilités (oui, même la première). Quoi qu'il en soit, cette transformation crée une nouvelle matrice.

6.2 Zoom

Fonctionnalité assez explicite, elle consiste à faire un zoom de l'image. Pour simplifier, nous ne donnerons la possibilité de faire que des zooms de 1x (taille de base de l'imgae), 2x (le double), 3x (le triple), 4x (le quadruple), 5x (le quintuple) et 6x (le sextuple).

Pour ce zoom nous ne demandons pas de créer une nouvelle matrice. Tout se passe au moment de l'affichage. A vous de modifier l'affichage en conséquence.

6.3 Agrandissement

L'agrandissement a le même effet que le zoom si ce n'est qu'il crée une nouvelle matrice correspondant à l'agrandissement et donc une nouvelle matrice.

6.4 Flou simple

Cet algorithme est la version basique d'un flou. Pour un pixel, nous considérons tous ses voisins directs, le calcul étant basé sur la moyenne uniforme des pixels qui l'entourent (une zone de 3x3 pixels) et du pixel en question.

5%	10%	5%
10%	40%	10%
5%	10%	5%

FIGURE 5 – Grille des pondérations pour un pixel central dans une zone 3x3.

0.25%	2%	3%	2%	0.25%
2%	3%	5%	3%	2%
3%	5%	39%	5%	3%
2%	3%	5%	3%	2%
0.25%	2%	3%	2%	0.25%

FIGURE 6 – Grille des pondérations pour un pixel central dans une zone 5x5.

6.5 Flou par moyenne pondérée : bilinéaire

Cette fonctionnalité permet de flouter l'image de la même manière que la précédente, à la différence que nous associons un poids différent pour chaque pixel alentour. Ainsi, la valeur du pixel central est défini selon les pondérations suivantes :

- 40% de la valeur du pixel avant modification
- 10% de la valeur de chaque pixel en haut, bas, gauche et droite
- 5% de la valeur des pixels en diagonale

La figure 5 donne la grille des pondérations des valeurs pour les pixels alentours.

Vous donnerez à l'utilisateur la possibilité de modifier ces pondérations.

6.6 Flou par moyenne pondérée : bicubique

Même principe que la précédente, en prenant les voisins dans un carré de 5x5. Les pondérations de base sont les suivantes par rapport aux valeurs des pixels avant modification :

- 39% pour le pixel central
- 32% pour la première couronne
- 29% pour la deuxième couronne

La figure 6 illustre les pondérations pour chaque pixel du voisinage en 5x5. Le pixel du centre garde toujours une plus grande proportion que les autres.

Vous donnerez à l'utilisateur la possibilité de modifier ces pondérations.

Indication : Attention aux pixels en bordure d'image !

6.7 Flou de zone

Ce flou correspond à celui qui est utilisé, par exemple, pour garantir l'anonymat de personnes dans les vidéos. Il s'agit de placer d'énormes "pixels" sur le visage de ces personnes.

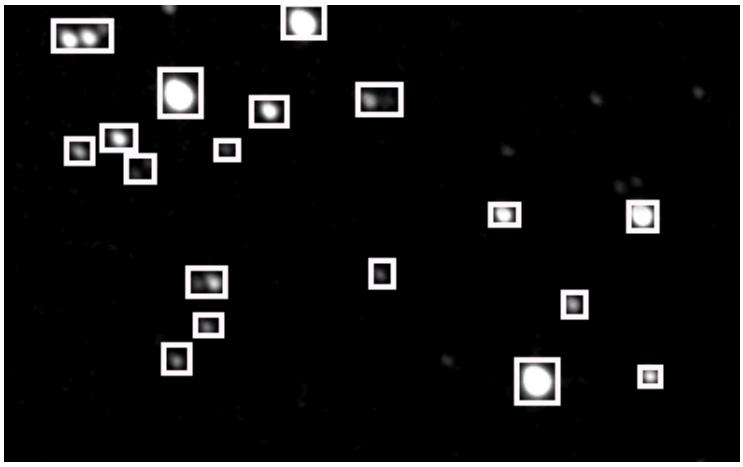

FIGURE 7 – Les étoiles avec leur halo sont encadrées. Le problème de détection en haut à gauche.

L'idée est de découper l'image en zones et, pour chaque zone, de remplacer chaque pixel par la moyenne des valeurs des pixels de la zone. Vous devrez proposer à l'utilisateur la possibilité de spécifier la taille des zones.

7 Transformations (très) avancées

Les transformations précédentes sont des transformations simples sur les images. Cependant, nous nous intéressons à des images bien spécifiques : les images astronomiques. Les algorithmes demandés dans cette section s'intéressent à des transformations plus orientées vers la maniuplation d'images astronomiques.

7.1 Détection d'étoiles

Le but de cette transformation est de détecter les étoiles présentes dans l'image. Puisque le bruit a (normalement) déjà été éliminé de l'image, nous allons pouvoir encadrer les étoiles présentes dans l'image de la figure 7. Cette image montre qu'il peut y avoir des problèmes de détection quand des étoiles sont très proches : nous ne nous occuperons pas de ce problème.

Indication : Détectez les pics de luminosité et cernez les pixels alentours qui représentent son halo.

7.2 Raffinement de l'élimination du bruit

L'algorithme présenté au tout début du projet est un peu brut en l'état. En effet, en l'appliquant, il n'y a pas de distinction entre les pixels générés par le

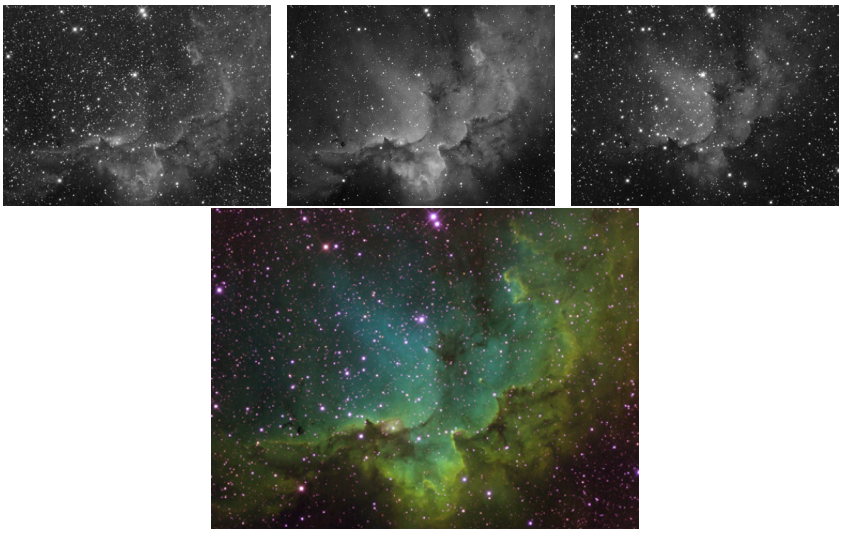

FIGURE 8 – En haut, les différentes niveaux de couleur pour le rouge SII (à gauche), le vert Ha (au milieu) et le bleu OIII(à droite). La composition de ces trois images avec ces intensités donne l'image colorée du dessous.

bruit et ceux qui sont de faible intensité car provenant du halo d'une étoile.

Indication : Utilisez la détection d'étoile au moment de la suppression du bruit.

7.3 Coloration

Vous avez certainement déjà vu de belles images astronomiques et très colorées : elles ne présentent pas leurs vraies couleurs, d'où l'expression *coloration en fausse-couleur*. Ces photos sont en fait le résultat de la superposition de trois photos distinctes. Par exemple, le téléscope spatial Hubble prend trois photos avec des filtres sur une longueur d'onde particulière. Dans le cas de Hubble, il filtre les rayonnements d'hydrogène-α (Ha), d'ions d'oxygène O^{2-} (OIII) et d'ions sulfure S^{-} (SII). Chaque image est ensuite associée à une couleur : ainsi, le SII donne les niveaux d'intensité pour le rouge, le Ha pour le vert et le OIII pour le bleu. La figure 8 montre une image de NGC7830 décomposée dans ses différentes composantes rouge, verte et bleue.

Vous devez implémenter cette fonctionnalité. Elle devra permettre à l'utilisateur de charger trois images, chacune associée à une couleur, et de les superposer pour créer une nouvelle image. De plus, l'utilisateur doit pouvoir ajuster la quantité de rouge, de vert et de bleu dans l'image.

Indication : Pour une représentation graphique de l'ajutement des niveaux de chaque couleur, regardez le composant *JSlider* en Swing.

Indication : L'ajustement du niveau d'une couleur, se résume à ramener les niveaux de gris à une autre échelle (plus petite) que celle de base en 256 niveaux.

Indication : Pensez qu'il existe la méthode setRGB(...) dans la classe BufferedImage ...

8 Et si vous vous ennuyez ...

Le titre est assez explicite : les transformations suivantes ajoutent plus de finesse et de compléxité) aux algorithmes précédents.

8.1 Autres colorations

La première coloration d'image présentée précédemment (SII :Ha :OIII) n'est pas unique. Bien que très utilisée, il est possible d'associer une image non pas à un seul canal couleur mais de l'associer à plusieurs canaux. Ainsi, la palette Ha :sG :OIII ,une autre palette couleur, défini les niveaux de couleur de cette façon :

- Rouge : Ha
- Vert : 70%OIII et 30%Ha
- Bleu : 90%OIII et 10%Ha

Dans ce cas, l'image SII n'est pas utilisée et le vert est synthétisé (sG) avec des proportions d'OIII et de Ha. Intuitivement, cette palette donne une autre tonalité à l'image.

Une autre technique de coloration Ha :OIII :SII est utilisée par l'équipe Canada-France-Hawaï (CFHT : Canadian-France-Hawaï Team) qui associe :

- Rouge : Ha
- Vert : OIII
- Bleu : SII

Donnez à l'utilisateur la possibilité de choisir entre ces trois palettes pour la coloration d'une image.

Bonus : Donnez en plus la possibilité de définir les proportions des images pour chaques couleur comme dans la palette Ha :sG :OIII.

8.2 Interpolations

L'interpolation est un outil mathématique qui permet de déduire des valeurs (de pixels par exemple) en ne connaissant que les valeurs voisines.

Un exemple d'interpolation : imaginons que vous connaissez la température extérieure (20°) à 11h. Plus tard, à 13h, vous regardez la température qui est montée à (22°). Il serait tentant de penser que la température a suivi une progression linéaire, ce qui donnerait, à midi, une température de 21°, image gauche de la figure 9. Cependant, une mesure de la température à midi donne 21,5°. La progression n'est pas linéaire et il convient d'ajuster la courbe de l'évolution de la température comme dans l'image droite de la figure 9. Cette courbe (ici de Bézier) permet une nouvelle interprétation de l'évolution de la température et est plus proche de la réalité que l'évolution linéaire.

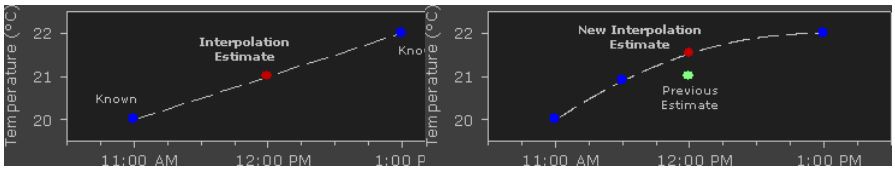

FIGURE 9 – A gauche, évolution linéaire de la température et à droite, évolution plus proche du réel.

Cet outil peut donc être utilisé partout où l'on a besoin de déduire les valeurs des nouveaux pixels, comme dans le cas d'un agrandissement. Vous devez permettre à l'utilisateur de définir la courbe d'interpolation qu'il veut : linéaire ou de Bézier (codées en dur ou à l'aide d'un outil permettant de la définir soi-même comme sur la figure, en "tirant" sur les points), ou toute autre courbe qu'il vous faudra documenter.

8.3 Images au format FITS

En réalité, les images astronomiques utilisées par les astronomes sont conservées dans un format ouvert : Flexible Image Transport System ou *FITS* [1]. Ce format permet de stocker une image (et même plusieurs en extension) et des meta-données sur l'image, l'appareil utilisé, l'orientation, ... Les images stockées ne sont pas non plus nécessairement les images Ha, SII, OIII de Hubble, mais n'importe quelle image peut y être stockée : rayons X, ultraviolet, infrarouge, ...

Il existe plusieurs librairies de lecture de fichiers *FITS* en Java, mais une sort du lot : il s'agit de la librairie *nom.tam*. Vous trouverez sur [2] le .jar contenant les classes et celui contenant les sources si ça vous intéresse. Mettez le .jar dans le même dossier que vos classes. Il y a aussi sur [3] la documentation de la librairie.

8.4 Tester sur du grandeur nature

Une fois la lecture de fichier FITS implémentée, vous pouvez tester votre application sur de vraies images que vous pouvez rechercher sur [5] et vous amuser avec vos transformations.

9 Conseils

Ce qui suit est une liste de conseils valables pour ce projet, mais aussi pour les suivants.

Soyez créatifs ... : Si vous pensez à un moyen élégant de représenter graphiquement des fonctionnalités ou de transformer la matrice, faites-le !

... mais restez efficaces : Pensez à faire une interface intuitive, accessible à l'utilisateur lambda. Une solution élégante ne doit pas se faire au détriment

d'autres fonctionnalités (i.e. ralentir d'autres algorithmes, ralentissement de l'interface, ...)

Documentez-vous : Interessez-vous à la javadoc, les tutoriels de Sun/Oracle sont aussi très bons. Tout ce qui est en rapport avec les transformations est bon à prendre pour une meilleure compréhension.

Commentez votre code : Un code non-commenté est bon à jeter. Tout élégant qu'il puisse être, les correcteurs n'ont pas envie de passer des heures pour comprendre ce que vous avez écrit, et ce, même si ça marche.

Respectez les conventions : le respect des conventions permet d'identifer à quel genre d'élément nous avons à faire (variables, classes, constantes, ...) et améliore la lisibilité du code.

Références

[1] Le document qui défini le standard du format FITS `http://fits.gsfc.nasa.gov/fits_standard.html`

[2] Librairie Java pour intéragir avec des fichiers FITS : `http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/v1.08.0/`

[3] Documentation de la librairie Java nom.tam : `http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/FitsLib.pdf`

[4] Récupération d'images au format FITS : `http://archive.stsci.edu/cgi-bin/dss_form`

[5] Recherche coordonnées d'objets spatiaux : `http://archive.stsci.edu/`