

# TP de morphologie mathématique

Hugues Talbot

16 décembre 2011

## 1 Règles et principes

Ce TP n'est pas noté, mais je vous engage à tenter de le terminer afin d'être préparé pour le deuxième TP d'après les vacances, qui lui le sera.

## 2 Prise en main du système PINK-PYTHON

Dans cette section nous allons nous familiariser avec le système PINK-PYTHON sous LINUX.

PINK-PYTHON est un logiciel d'analyse d'images en sources ouvertes (open-source) produit par une équipe de chercheurs du département informatique de l'ESIEE.

### 2.1 Architecture du système

Ce système est basé sur Python, et fait appel à un module dédié incluant la morphologie mathématique, la visualisation et d'autres opérateurs par exemple de géométrie et de topologie discrète.

#### 2.1.1 Prérequis

Avant de pouvoir commencer, il faut suivre les étapes suivantes :

1. Lancer un browser web, par exemple mozilla.
2. Se loger sous Linux sur une des lames de calcul :  
`http://blade08.esiee.fr/blade/tunnel/ujoimro`
3. Choisir `blade-vm11` ou `blade-vm12`, les deux machines sont virtuelles et quasi-identiques.
4. Logez vous sur ces lames avec votre login / mot de passe ESIEE.
5. Une fois logé, lancez un terminal, par exemple `konsole`.
6. Dans le terminal, lancez `pink` par la commande  
`pink.sh`
7. Pointer le browser web vers le site suivant :  
`http://www.pinkhq.com/`
8. Lancez un éditeur, par exemple `nedit` ou `kate`. Toutes les commandes que vous allez taper devront être sauvegardées dans un fichier dont vous vous servirez pour faire le rapport.
9. Au fur et à mesure que vous progresserez dans le TP, vos solutions devront apparaître comme des fonctions de Python.

#### 2.1.2 Images

Les images que nous traiterons sont toujours au format standard PGM. Les images pour ce TP sont toutes disponibles dans le répertoire source suivant :

`/user/talboth/Public/ISBS/TP/Images/tp1/`

## 2.2 Commandes Unix/Linux et PINK

Pour continuer avec ce TP, il vous faut un minimum de connaissance des commandes UNIX. Par exemple, créez un répertoire pour ce TP dans votre répertoire principal `mkdir tplmorpho`. Copiez l'image `cells.pgm` du répertoire source vers le répertoire que vous venez de créer (commande `cp <source> <destination>`). Puis changez de répertoire vers celui que vous venez de créer (`cd tplmorpho`).

### 2.2.1 Documentation UNIX et PINK

Le système UNIX dispose d'un très grand nombre de commandes, qui vont bien au delà de ce TP. En général vous pouvez utiliser l'aide en ligne (cliquer sur le bouton avec une bouée comme icône). À partir de la console vous pouvez invoquer l'aide en ligne par la commande `man`, par exemple :

```
man mkdir
man cp
```

Vous pouvez aussi utiliser le navigateur de KDE ou de GNOME qui sont bien adaptés à une utilisation interactive, similaire à celle de Windows.

Notez que les commandes Unix/Linux restent disponible dans l'environnement PINK.

### 2.2.2 Commandes pink

Le système PINK utilise l'interpréteur python qui doit vous être familier, mais on ne peut pas se contenter de démarrer l'interpréteur python, c'est pourquoi nous lançons le script "pink.sh" pour bénéficier d'un environnement plus adapté.

Le module de PINK s'appelle simplement 'pink'. Pour démarrer une session pink, importez le module de la façon suivante (attention au majuscules !) :

Pink-python : importation des éléments du module

```
# la partie principale du module
from pink import cpp as pink
# la partie affichage
from pink import Imview
```

Les commandes de PINK ont toutes la structure suivante (le % représente l'invite) :

```
resultat = pink.une_commande(donnees)
```

Par exemple lisons une image :

```
image= pink.readimage("cells.pgm")
```

L'aide en ligne PINK est disponible sur le site <http://pinkhq.com>. Prenez un moment pour vous y familiariser.

Il existe aussi une aide interactive, par exemple essayez :

```
help(pink.readimage]
```

vous devriez voir :

```
Help on built-in function readimage:
```

```
readimage(...)
```

```
  readimage( (str)filename) -> object :
```

```
    reads an image from a 'pgm' file and returns an object with the appropriate type
```

```
  C++ signature :
```

```
    boost::python::object readimage(std::string)
```

```
(END)
```

### 2.2.3 Interactivité

Le programme `pink.sh` est interactif avec historique des commandes tapées (avec les flèches de direction hautes et basses), et raccourcis par la touche “TAB” (à gauche de la lettre A).

Par exemple, essayez :

```
I=pink.read<TAB>
```

où `<TAB>` correspond simplement à presser la touche TAB.

## 2.3 Visualisation

La visualisation des images s’opère par la commandes `Imview`, avec la syntaxe suivante :

Pink-python : lecture et affichage d’une image

```
image= pink.readimage(" cells .pgm")
# ceci cr\`ee un port de visualisation et visualise 'image'
viewer=Imview(image)
# ensuite on peut r\`utiliser ce port , par exemple:
resultat=pink.monFiltre(image)
viewer.show(resultat , " resultat ")
```

Dans le cas où on réutilise un même port plusieurs fois, il est possible de passer d’une image à l’autre par les touche d’espace et effacement. Le zoom avant et arrière s’effectue avec les touches “>” et “<” respectivement.

## 2.4 Exemple d’interaction aide-programmation

### Exercice 2.1 (La fonction `erosion`)

1. Trouvez la documentation de la fonction `pink.erosion`
2. Trouver comment fonctionne la fonction `pink.erosion`. Expliquer les paramètres.
3. Trouvez la documentation de la fonction `pink.genball`. A quoi sert cette fonction ? A quoi sert l’argument `dimension` ? que veut dire le fait qu’il soit par défaut égal à 2 ?
4. Réaliser l’érosion de l’image `cells.pgm` par un disque de rayon 2.
5. Visualisez le résultat et comparez avec l’image de départ.
6. Trouvez la documentation de la fonction `pink.erosball`. Réalisez l’opération précédente avec cette fonction.

## 2.5 Fichier programme

Par la suite, il est nécessaire de créer un fichier executable contenant toutes les suites de commandes qui sont solution des exercices. Le format est le suivant :

- Vous devez l’appeler `nom1_nom2.py` où `nom1` et `nom2` sont les noms des composantes de votre binôme.
- Chaque ligne (sauf la première) commençant par un caractère # est un commentaire. N’hésitez pas à commenter vos fichiers.
- Chaque ligne non-commentée est une commande executable
- A chaque fois que vous faite un changement dans ce fichier, vous pouvez le recharger et l’exécuter par la commande  
`%run nom1_nom2.py`

N’oubliez pas que vous devez rendre un tel fichier avec votre rapport.

### 3 Filtrage de l'image `cells.pgm`

L'image `cells.pgm` est une image binaire de certaines cellules, avec présence de bruit tant dans le fond que dans les cellules elles-mêmes.



FIGURE 1 – Image initiale `cells.pgm`.

#### 3.1 Filtrage du bruit

Dans cette partie on va s'efforcer d'éliminer le bruit dans le fond et dans les cellules sans pour autant affecter la forme des cellules.

##### Exercice 3.1 (Filtrage avec reconstruction)

1. Cherchez la documentation des fonctions `pink.geodilat` et `pink.georerod`.
2. Erodez l'image de départ avec un ES de bonne taille pour éliminer le bruit blanc. Montrez le résultat.
3. Reconstituez l'image érodée en utilisant l'image de départ comme masque. Montrez le résultat. Commentez.
4. Proposez une approche similaire pour éliminer le bruit noir dans les cellules, sans pour autant éliminer les trous importants et sans changer la forme des cellules. Montrez le résultat.
5. Donnez la suite des opérations pour éliminer le bruit noir et blanc dans l'image de départ et montrez le résultat

Par la suite cette image est appelée `cells_filt.pgm`.

#### 3.2 Bords et trous

En général on ne peut pas utiliser les objets touchant le bord de l'image car les mesures faites sur ces objets ne peuvent être que biaisées ou qu'incomplètes.

##### Exercice 3.2 (Élimination des objets touchant le bord)

1. Cherchez la documentation des fonctions `pink.frame`
2. Cherchez comment réaliser une soustraction d'images dans PINK-Python
3. Proposez une solution utilisant la reconstruction géodésique pour trouver, puis éliminer les objets touchant le bord de l'image.
4. Appliquez cette procédure sur l'image `cells_filt.pgm` et montrez le résultat.

On appellera cette image `cells_nohole.pgm`.

Il est souvent utile de boucher tous les trous d'un objet binaire, par exemple lorsque ces trous sont produits par du bruit, mais également par différence si on souhaite détecter les cellules à trou.

### Exercice 3.3 (Bouchage de trous)

1. Cherchez la documentation de la fonction `pink.inverse`.
2. Proposez une solution pour boucher tous les trous des cellules en utilisant la reconstruction géodésique.
3. Appliquez cette procédure sur l'image `cells_nohole.pgm` et montrez le résultat.

On appellera cette image `cells_filled.pgm`.

### Exercice 3.4 (Détection des objets comportant au moins un trou)

1. Cherchez la documentation de la fonction `pink.min`.
2. En utilisant le résultat de l'exercice précédent (`cells_filled.pgm`), proposez une solution pour détecter les cellules qui ont au moins un trou.
3. Appliquez cette procédure sur l'image `cells_filt.pgm` et montrez le résultat.

On appellera cette image `cells_final.pgm`.

## 4 Extraction de pistes sur circuit électronique

L'image initiale pour cet section est l'image `circuit.pgm`, disponible dans le répertoire source (voir section 2.2.2).

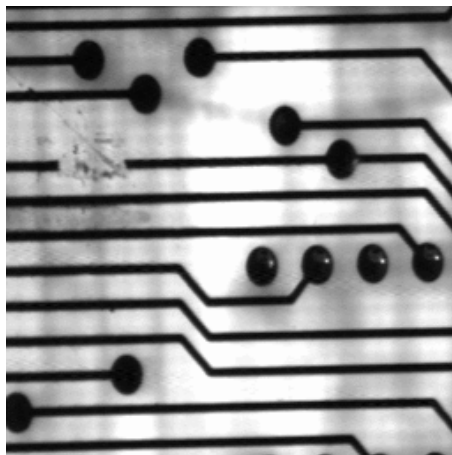


FIGURE 2 – Image initiale `circuit.pgm`.

### Exercice 4.1 (Extraction de pistes)

1. Voir la documentation de la fonction `pink.seuil`
2. À l'aide de cette fonction, déterminez manuellement un seuil de niveau de gris sur l'image initiale qui donne une image binaire contenant comme objets les pistes et les pastilles (vous devrez sans doute inverser l'image).
3. Suivant une méthodologie inspirée par celle des cellules, éliminez le bruit blanc et noir.
4. Vous connaissez maintenant l'opérateur `pink.erosion`. Regardez la documentation de l'opérateur `pink.dilatation` (dilatation en anglais). Comment peut-on maintenant détecter les pastilles ?
5. Proposez (et utilisez) une idée pour retrouver autant que possible la forme des pastilles sans pour autant reconstruire les pistes.
6. Extrayez les pistes.

## 5 Microscopie électronique à balayage

L'image initiale pour cette section est l'image `meb.pgm`. On se propose d'extraire la "corne" en bas à droite, composées de petites billes.

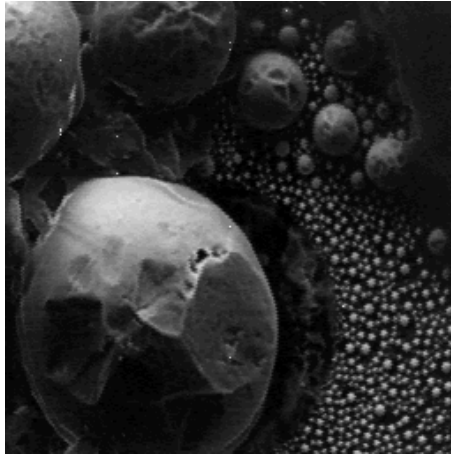


FIGURE 3 – Image initiale `meb.pgm`.

### Exercice 5.1 (Extraction des petites billes)

1. Voir la documentation des fonctions `pink.dilatball` et `pink.erosball`.
2. Voir la documentation de la fonction `manipulate`
3. Trouvez un bon seuil de départ pour cette image.
4. Détectez les petites billes.