Lecture d'une image ou d'une vidéo

1. Lecture d'une image

On commence par charger le module puis on ouvre l'interface :

```
Needs['CV`'];
CVOpen[];
```

Pour lire une image contenue dans un fichier (JPG, PNG) :

```
img=CVLoadImage['baboon.jpg',CVLoadImageUnchanged];
```

Le fichier est cherché dans le répertoire courant, qu'il faut définir au préalable avec `SetDirectory`. On peut aussi indiquer le chemin complet d'accès au fichier. La fonction renvoit une référence (un entier) à une structure IplImage. Le deuxième argument peut ausi prendre la valeur `CVLoadImageGrayScale`. Dans le cas présent, l'image lue comporte 3 couches de 8 bits chacune. Ces couches sont B (Blue) G (Green) et R (Red).

Pour obtenir la taille de l'image :

```
s=CVGetSize[img]
```

```
{300, 300}
```

2. Affichage d'une image

Il existe trois moyens d'afficher une image. Le premier consiste à utiliser l'interface graphique de OpenCV. Une fenêtre portant un nom doit être créée puis l'image lui est fournie :

```
CVNamedWindow['baboin',CVWindowAutosize];
CVShowImage['baboin',img];
```

On remarquera que cette fonction est prévue pour afficher correctement une image dont les couches sont dans l'ordre B, G et R. Avec des images RGB, SHV et autres l'affichage sera incorrect.

La deuxième méthode consiste à convertir l'image en un objet `Image`. Pour cela, il faut obtenir un tableau des valeurs de l'image :

```
array = CVGetImageArray[img,1,-1];
```

Le deuxième argument indique que les valeurs doivent être normalisées entre 0 et 1. Le troisième argument indique que l'ordre des couches doit être inversé, afin d'obtenir des triplets dans l'ordre RGB.

```
Image[array]
```

La fonction `CVGetImageArray` renvoit trois matrices correspondant aux couches R, G et B. Si l'image BGR initiale est utilisée, ces matrices ne sont pas dans le bon ordre pour être affichées correctement. L'objet `Image` ainsi obtenu peut être utilisé avec les fonctions de traitement d'image de Mathematica.

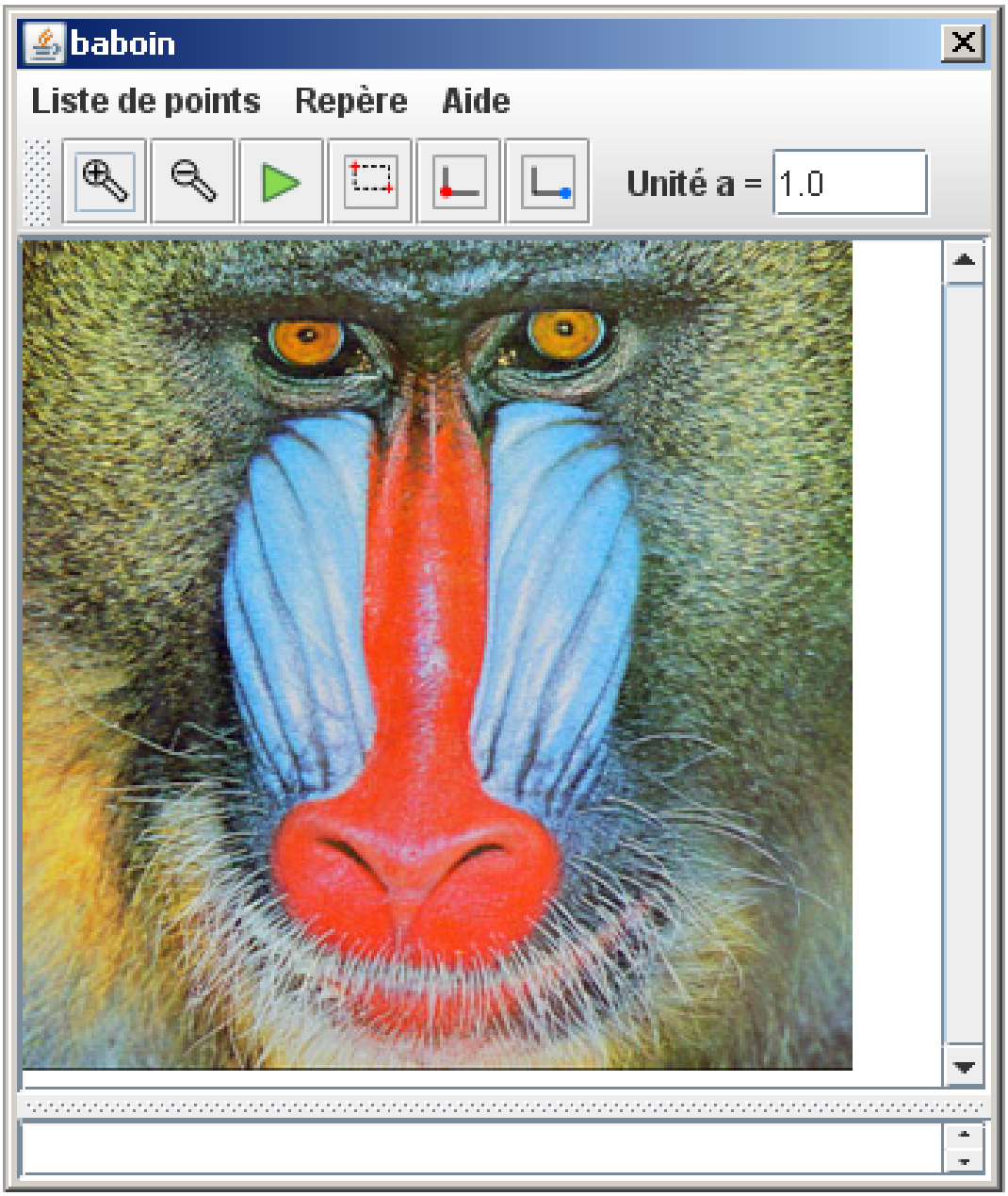
La troisième possibilité consiste à utiliser l'interface java. Pour cela, on commence par créer un éditeur d'image :

```
edit=CVJImageEditor['baboin',True];
```

Le deuxième argument indique si la fenêtre doit être modale ou pas. Si elle est modale, l'exécution reste en attente de la fermeture de la fenêtre : il est impossible de continuer à travailler sur le notebook tant que la fenêtre est ouverte. Dans le cas contraire, le fil d'exécution est rendu au noyau de Mathematica dès que la fenêtre est ouverte. Si la fenêtre est modale, elle reste invisible tant qu'aucune image ne lui a été fournie.

On fournit ensuite l'image à afficher :

```
CVJShowImage[edit,img];
```



Outre la possibilité de zoomer sur l'image, cet éditeur permet de définir un repère, de saisir des points et des rectangles.

3. Lecture d'une vidéo

La lecture d'une vidéo à partir d'un fichier se fait de la manière suivante :

```
video=CVCaptureFromFile['chocBilles.wmv'];
```

Différentes informations sur la vidéo sont accessibles. La plus importante en pratique est la taille des images :

```
width=CVGetCaptureProperty[video,CVCapPropFrameWidth]
height=CVGetCaptureProperty[video,CVCapPropFrameHeight]
```

```
640
```

```
480
```

Le taux d'image (frame rate) et le nombre d'image sont en principe accessibles :

```
fps=CVGetCaptureProperty[video,CVCapPropFps]
```

```
count=CVGetCaptureProperty[video,CVCapPropFrameCount]
```

```
47.583333333333336
```

```
2472
```

Dans le cas présent, cette dernière imformation est fausse. La vidéo dure environ 3 secondes, ce qui fait au maximum 150 images pour le taux indiqué. Ce taux lui-même est douteux puisque cette vidéo a été prise avec une webcam fonctionnant au maximum à 30 fps.

4. Lecture d'une image d'une vidéo

La fonction suivante permet d'obtenir une copie d'une image d'une vidéo. Le premier argument est la référence à la vidéo, le second le nombre d'images à sauter. Par exemple, pour obtenir la 5 ième et la 10 ième image en partant du début :

```
img5=CVGetFrame[video,5];
img10=CVGetFrame[video,5];
```

Lorsque la fin de la vidéo est atteinte avant l'image demandée, la fonction `CVGetFrame` renvoit une valeur nulle.

Lorsque ces images ne sont plus utilisées, il est bon de libérer la mémoire correspondante par :

```
CVReleaseImage[img5];
CVReleaseImage[img10];
```

À noter que pour relire la vidéo depuis le début, il faut la recharger à nouveau, non sans avoir au préalable désaloué l'espace mémoire correspondant :

```
CVReleaseCapture[video];
video=CVCaptureFromFile['chocBilles.wmv'];
```

5. Déroulement d'une vidéo

La première possibilité pour obtenir le déroulement dans le temps d'une vidéo (image par image) est d'utiliser la fonction `CVPlayVideo`, qui utilise l'interface graphique d'OpenCV :

```
CVNamedWindow['billes',CVWindowAutosize];
CVPlayVideo['billes',video,400];
```

Le dernier argument est le délai entre l'affichage de deux images consécutives, en millisecondes.

La fonction précédente est utile pour avoir un aperçu de la vidéo. Pour obtenir un déroulement image par image, on utilisera l'éditeur d'image introduit plus haut en lui fournissant la référence de la vidéo :

```
CVReleaseCapture[video];
video=CVCaptureFromFile['chocBilles.wmv'];
edit=CVJImageEditor['billes',True];
CVJShowVideoFrames[edit,video,1];
```

Cette fois-ci, l'affichage est statique. Le dernier argument est le nombre d'images à sauter à chaque fois que l'utilisateur appuie sur la flèche verte.

6. Fin de la session

Pour finir, la fonction suivante libère toutes les ressources mémoire utilisées et ferme l'interface :

```
CVClose[];
```