TRAVAUX DIRIGES

PARTIE 2 - MATLAB

TRAITEMENT D'IMAGES ET VISION

Cycle ingénieur – ING 2 – Spécialité Informatique

Alice POREBSKI

alice.porebski@eilco-ulco.fr

TD Matlab - Traitement d'images et Vision

Matlab est un logiciel de calcul scientifique permettant de développer des solutions à des problèmes techniques. Il permet de réaliser du calcul numérique et tracer des graphiques pour visualiser et analyser les données. Il dispose d'un langage et d'un environnement de programmation interactifs ainsi que d'outils pour concevoir des interfaces utilisateur graphiques. Matlab est associé à des boites à outils appelées *toolbox* permettant d'accéder à des fonctions spécifiques à un domaine d'application comme le traitement d'images par exemple. Les TD et TP de vision réalisés avec Matlab nécessitent ainsi les toolbox **Image Acquisition** et **Image Processing**.

Après avoir rappelé quelques généralités sur Matlab, nous allons découvrir les fonctions de base permettant d'acquérir, d'afficher et de sauvegarder une image. Les principaux outils de traitements d'images seront ensuite abordés et appliqués à des problématiques concrètes, telles que la compression, la restauration, la segmentation, ...

1 Quelques généralités sur Matlab

La figure 1 montre la décomposition de la fenêtre Matlab en plusieurs onglets :
- un onglet d'édition des commandes (**Command Window**),
- un onglet de visualisation de l'espace des variables (**Workspace**),
- un onglet de visualisation des fichiers du répertoire de travail (**Current Folder**),
- un onglet de visualisation de l'historique des commandes (**Command History**),
- un onglet permettant de visualiser le contenu de fichiers (**Editor**)
- et un onglet permettant de visualiser le contenu des variables (**Variables**).

Les fonctions sont éditées dans la fenêtre de commandes et exécutées en appuyant sur la touche **Entrée**. Le point virgule à la fin d'une fonction permet d'éviter d'afficher les données résultats de la fonction exécutée ou de séparer plusieurs fonctions sur une même ligne de commande. Plusieurs fonctions et commandes peuvent être saisies dans un fichier qui sera enregistré avec l'extension **.m**. En éditant le nom de ce fichier dans la fenêtre de commande, l'ensemble des fonctions déclarées dans ce fichier seront exécutées. L'édition de ce fichier peut s'effectuer en sélectionnant dans le menu **File** : **New** ▶ **M-file**. Il est également possible d'y créer des fonctions en utilisant la commande `function`.

La fonction `figure` permet de générer une fenêtre graphique permettant de visualiser les données (courbes, images, ...). La fonction `close` permet de fermer la fenêtre active et `close all` permet de fermer toutes les fenêtres. Les fonctions `title`, `xlabel`, `ylabel` permettent respectivement d'afficher un titre à la figure, le nom de l'axe des abscisses et le nom de l'axe des ordonnées. La fonction `subplot` permet d'afficher plusieurs graphiques ou images au sein d'une même figure.

La fonction `clear` efface les variables mises en mémoire durant une session Matlab tandis que la fonction `clc` efface le contenu de la fenêtre de commande. La fonction `pause` permet de mettre en veille la

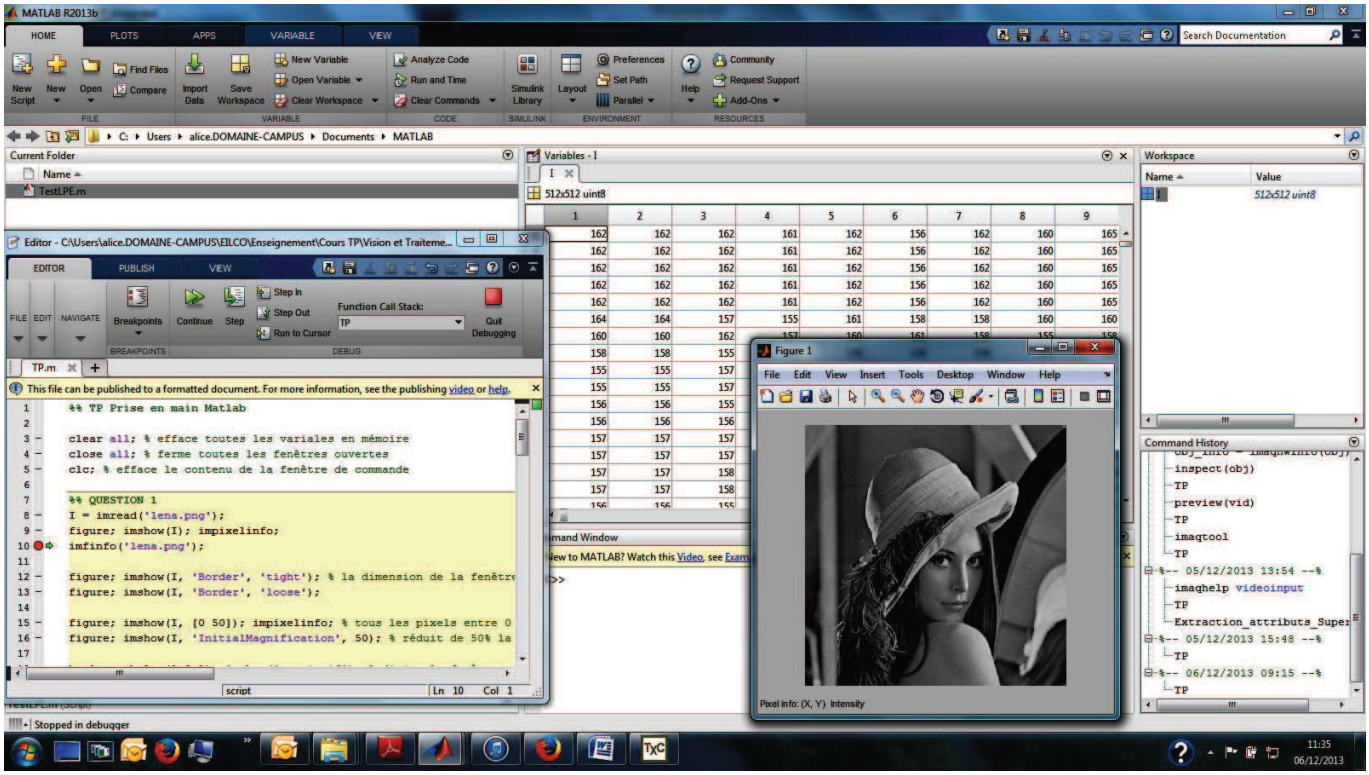

FIGURE 1 – Le logiciel de calcul scientifique Matlab

fenêtre de commandes de Matlab et les fonctions `disp`, `display` ou `celldisp` permettent l'affichage de textes.

Chaque variable déclarée dans Matlab est stockée dans l'espace des variables à partir duquel il est possible de consulter la taille et le type de la variable ainsi que d'éditer son contenu par un double-clic sur le nom de la variable. Les variables Matlab sont des objets de type structure ou des tableaux à n dimensions. Ainsi, un scalaire est un tableau de taille 1×1, un vecteur est un tableau à 1 dimension de taille $1 \times n$, une matrice est un tableau à 2 dimensions de taille $m \times n$, ... Il est ensuite possible d'accéder facilement au tableau, à un élément du tableau ou à une ou plusieurs dimensions particulières du tableau.

Pour créer le vecteur $V = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$ par exemple, il faut saisir l'instruction `V=[1 2 3 4]`. Pour créer la matrice $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ par exemple, il faut saisir l'instruction `M=[1 2; 3 4]`, ...

Certaines opérations arithmétiques sur les tableaux peuvent être effectuées de deux manières :

- de manière matricielle avec les opérateurs : `*`, `/` ou `^`,
- ou élément par élément avec les opérateurs : `.*`, `./` ou `.^`.

Une image en niveaux de gris est un tableau à 2 dimensions. Si I est la variable contenant les données d'une image à niveaux de gris, l'instruction `I(1,1)` renvoie la valeur du pixel de coordonnées $(1,1)$. L'instruction `I(:,1)` renvoie les valeurs des pixels de la première colonne et l'instruction `I(1,:)` renvoie les valeurs des pixels de la première ligne, ... Nous allons voir dans le paragraphe suivant les fonctions de base permettant de manipuler des images sous Matlab.

2 Fonctions de base permettant de manipuler des images sous Matlab

2.1 Lecture, affichage et sauvegarde d'une image

Le chargement en mémoire d'une image se fait avec la fonction `imread`. Par exemple, la fonction suivante permet de lire une image et de placer son contenu dans une variable de type matrice :

```
I = imread('lena.png');
```

Cette variable est alors visible dans le Workspace (espace des variables) de Matlab. La fonction `whos` permet d'afficher toutes les informations relatives aux variables en mémoire et la fonction `imfinfo` affiche les informations relatives à un fichier image.

L'affichage de l'image (ou de la variable) est réalisé par la fonction `imshow`. Ainsi les fonctions suivantes ouvrent une nouvelle fenêtre pour y afficher l'image I.

```
figure; imshow(I);
```

1) Créer un fichier .m qui lit et affiche l'image 'lena.png'. Donner les caractéristiques de cette image.

Les fonctions `imwrite` et `print` permettent la sauvegarde, respectivement, des images et des figures sous différents formats (tif, jpg, bmp, png, gif, eps, ...).

2) Enregistrer l'image 'lena.png' sous les formats suivants : JPEG, BMP, GIF et le format TIFF.

2.2 Mise au point du système de vision et acquisition des images

Le matériel d'acquisition est composé de :

- une caméra couleur IDS uEye de modèle UI-1240ML, de résolution 1280×1024, équipée d'un capteur $1/2''$ (6.784 mm $\times$ 5.427 mm),
- un objectif Fujinon de focale $f = 25$ mm,
- deux ampoules à LEDs blanc pur.

La caméra est fixée sur un statif et est reliée au PC par port USB. Un pilote Windows spécifique permet de communiquer entre le PC et la caméra.

La fonction `imaqtool` permet d'accéder à l'interface Matlab d'acquisition d'images. Sur cette interface, nous retrouvons les informations relatives au matériel et aux pilotes installés.

La figure 2 illustre l'interface 'Image Acquisition Tool'. Cette interface est décomposée en plusieurs onglets :

- un onglet affichant les différentes entrées vidéo ainsi que les formats d'image disponibles (Hardware Browser),
- un onglet permettant de régler les paramètres d'acquisition (Acquisition Parameters), comme par exemple l'espace de codage de l'image (RGB, YCbCr ou niveaux de gris), le contraste (Contrast), le temps d'exposition (Exposure), la correction gamma (Gamma), la netteté (Sharpness), ...
- un onglet d'information (Information) récapitulant les principales informations sur l'entrée vidéo sélectionnée et les paramètres d'acquisition fixés,
- un onglet permettant de prévisualiser et d'acquérir une image ou une vidéo (Preview),

- un onglet affichant la retranscription en code Matlab des différentes opérations effectuées à l'aide de l'interface (**Session Log**) :
  - `videoinput` : affiche les principales propriétés de l'objet d'entrée vidéo créé,
  - `preview` : permet de créer une fenêtre d'aperçu afin de visualiser la scène observée,
  - `closepreview` : permet de fermer cette fenêtre.

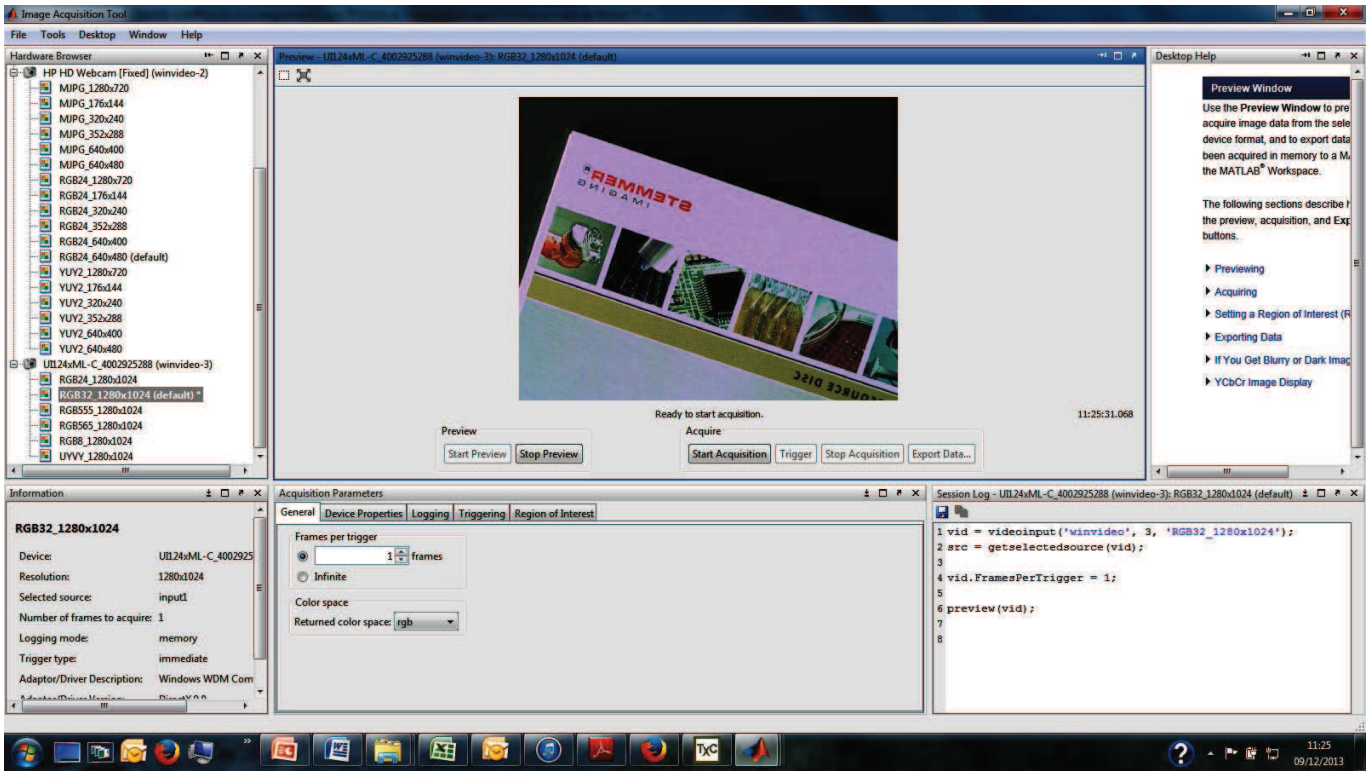

FIGURE 2 – L'interface d'acquisition d'images de Matlab

3) Utilisation de la toolbox **Image Acquisition** :

- Placer un objet quelconque sous la caméra et orienter correctement cette dernière par rapport à l'objet à observer.
- Prévisualiser une image et ajuster la distance de travail de sorte à obtenir un champ de vision incluant une surface de 130 mm $\times$ 105 mm.
- Ajuster la mise au point et de l'ouverture de l'objectif de la caméra.
- Mesurer la distance de travail et la confronter avec la théorie. On utilisera la formule suivante :

$$f = \frac{C \times D}{H},$$

avec f, la focale,
C, la largeur ou la longueur du capteur,
D, la distance de travail,
H, la largeur ou la longueur du champs de vision.

- Régler les paramètres d'acquisition afin d'obtenir une image de bonne qualité avec le minimum d'ombres et de reflets. Veiller en particulier à effectuer tous les réglages en mode manuel et à minimiser les paramètres de gain et de contraste qui ont pour effet d'accentuer le bruit.
- Indiquer les valeurs de réglage de l'objectif (paramètres extrasèques) et de la caméra (paramètres intrinsèques) et expliquer brièvement ces différents paramètres.

- Calculer la précision (résolution spatiale) P de votre système.

4) Placer deux objets de couleurs différentes sous la caméra et acquérir une image couleur. Enregistrer l'image au format PNG et donner les caractéristiques de cette image.

5) Changer l'espace de représentation RGB en un espace de luminosité et faire l'acquisition d'une nouvelle image sans modifier les réglages précédents et sans déplacer les objets observés. Enregistrer également cette image au format PNG.

2.3 Types des images en mémoire

Matlab supporte 4 formats d'images :
- les images binaires,
- les images d'intensités (à niveaux de gris),
- les images couleur RGB,
- les images couleur indexées.

Il est possible de changer de format en utilisant les fonctions suivantes :
- `ind2gray` : indexé → intensité,
- `ind2rgb` : indexé → RGB,
- `rgb2ind` : RGB → indexé,
- `rgb2gray` : RGB → intensité,
- `im2bw` : intensité, indexé, RGB → binaire : c'est l'opérateur de binarisation. Une image binaire peut être également obtenue en utilisant des opérateurs de comparaison et des opérateurs logiques. Par exemple, les instructions `(I==seuil)` ou `((I>=seuil_bas) & (I<seuil_haut))` permettent d'obtenir des images binaires par comparaison des niveaux des pixels d'une image `I` à des valeurs de seuils.

Le passage d'une image couleur à une image d'intensité correspond à la transformation des composantes R, G, B en la composante Y des systèmes de transmissions YIQ et YUV qui séparent l'information de luminance et de chrominance.

6) Ouvrir l'image couleur acquise et enregistrée précédemment et la convertir en une image d'intensité. Comparer cette image avec celle précédemment acquise directement en niveaux de gris en calculant l'erreur quadratique moyenne entre ces deux images.

7) Ouvrir l'image couleur acquise et enregistrée précédemment et extraire et afficher en niveaux de gris puis en couleur les images correspondant aux composantes R, G et B. Que peut-on observer ?

2.4 Autres fonctions de base

Les valeurs des pixels des images peuvent être de différents types :
- logique (0 ou 1 pour les images binaires),
- entier non signé codé sur 8 bits (entre 0 et 255),
- entier non signé codé sur 16 bits (entre 0 et 65535),
- réel (entre 0 et 1).

Il est possible de changer le type des variables en utilisant les fonctions suivantes :
- `im2double` : codage d'image en type réel,
- `im2uint8` : codage d'image en type entier non signé sur 8 bits,
- `im2uint16` : codage d'image en type entier non signé sur 16 bits,

- `double` : conversion de données en type réel,
- `uint8` : conversion de données en type non signé sur 8 bits,
- `uint16` : conversion de données en type non signé sur 16 bits.

Les images binaires sont codées en entier logique ou en réel logique et leurs pixels ont des valeurs uniquement égales à 0 ou à 1. L'affichage d'une image peut se faire, soit en réel et les valeurs des pixels sont alors comprises entre 0.0 et 1.0, soit en entier et les valeurs des pixels sont alors comprises, par exemple, entre 0 et 255 pour des entiers non signés codés sur 8 bits.

Certaines fonctions ou certains outils de Matlab permettent des manipulations interactives sur une image contenue dans une figure ou non :

- `imageinfo` : retourne les informations de l'image dans la figure ou d'un fichier image,
- `zoom` : zoom sur une zone de l'image de la figure,
- `imcrop` : sélectionne une zone de l'image,
- `improfile` : affiche le profil d'une ligne sélectionnée,
- `impixel` : retourne les valeurs des pixels sélectionnés,
- `impixelinfo` : affiche la position et les valeurs d'un pixel pointé avec la souris,
- `impixelregion` : affiche les valeurs des pixels dans une région sélectionnée avec la souris,
- `imdistline` : affiche la distance entre deux pixels sélectionnés,
- `imdisplayrange` : affiche l'intervalle des valeurs des pixels de l'image,
- `imcontrast` : réajuste une image,
- `imtool` : outil qui utilise les outils précédents.

D'autres fonctions permettent des opérations géométriques sur l'image :

- `imresize` : ré-échantillonnage de l'image (homothétie),
- `imrotate` : rotation de l'image.

8) Tester les outils `imtool`, `impixelinfo` et `imdisplayrange` sur l'image monochrome acquise et enregistrée précédemment.

Après avoir présenté les fonctions de base permettant de manipuler les données image, nous allons utiliser Matlab pour répondre à des problématiques concrètes, comme la compression d'image.

3 Compression

9) L'image 'lena.png' a été enregistrée précédemment sous les formats suivants : JPEG, BMP, GIF et le format TIFF. Ouvrir et afficher chacune de ces images, observer leurs différences en taille et en qualité et comparer les avec l'image d'origine. Mesurer ces différences avec l'image d'origine en calculant l'erreur quadratique moyenne.

10) Enregistrer l'image 'lena.png' au format JPEG avec différents niveaux de compression (mettre le paramètre `'Quality'` à 0, 25, 50, 75 et 100). Ouvrir et afficher ensuite chacune de ces images, observer leurs différences en taille et en qualité et comparer les avec l'image d'origine en calculant également l'erreur quadratique moyenne.

4 Réduction du bruit

Dans cette partie, nous allons nous intéresser aux techniques qui permettent de réduire le bruit dans une image. L'image 'lena_noisy.png' correspond à l'image 'lena.png' qui a été bruitée.

11) La fonction `imhist` permet de calculer l'histogramme d'une image en niveaux de gris. Appliquer cette fonction sur l'image 'lena.png' et afficher l'histogramme dans une nouvelle fenêtre. On utilisera pour cela la fonction `plot` qui permet d'afficher un graphique au sein d'une figure.

12) Afficher l'image 'lena_noisy.png' ainsi que son histogramme. Relever les différences et calculer l'erreur quadratique moyenne entre l'image originale et l'image bruitée.

La fonction `imfilter` permet d'appliquer un filtre sur une image. Pour définir le filtre qui va être appliqué, nous utilisons la fonction `fspecial`.

13) Appliquer un filtre gaussien sur l'image bruitée. Observer l'effet de ce pré-traitement sur l'image et sur l'histogramme. Mesurer l'amélioration en calculant l'erreur quadratique moyenne entre l'image originale et l'image filtrée.

14) Même question avec le filtre moyenneur. Conclure.

5 Restauration d'image

L'image 'lena_abimee.png' correspond à l'image 'lena.png' qui a été dégradée. Le filtre médian (fonction `medfilt2`) peut être utilisé pour lisser une image mais il peut également être utilisé dans le cadre de la restauration d'image.

15) Appliquer le filtre médian sur l'image 'lena_abimee.png' et observer le résultat avant et après restauration.

6 Segmentation

L'objectif de cette partie consiste à analyser des objets extraits d'une image par segmentation en régions. Les objets traités sont des pièces ou jetons de couleurs différentes : rouge, vert, bleu et jaune. La segmentation d'une image consiste à isoler les différents objets présents dans une image. La méthode la plus simple pour cela est de seuiller ou binariser une image à partir des niveaux de gris des pixels ou de leur couleur.

6.1 Binarisation

Généralement on attribue les pixels blancs (valeurs égales à 1) à la **forme** de l'objet présent dans une image binaire (avant plan) et les pixels noirs (valeurs égales à 0) au **fond** (arrière plan).

16) Mettre plusieurs objets sous la caméra et acquérir une image couleur de taille 1280×1024. Transformer cette image en image monochrome.

17) Binariser l'image acquise précédemment de telle sorte à obtenir des objets en blanc et un fond en noir. Si plusieurs binarisations sont nécessaires, utiliser les opérateurs logiques (fonctions `imcomplement`,

`or`, `xor` et `and`) pour recomposer l'image.

6.2 Érosion et dilatation

Les fonctions `imerode` et `imdilate` permettent respectivement de réaliser des opérations d'érosion et de dilatation sur des images à niveaux de gris à partir d'un élément structurant. Cet élément structurant peut être défini soit par une matrice de 0 et de 1, soit en utilisant la fonction `strel` qui permet de configurer des éléments structurants élémentaires.

6.2.1 Érosion

18) Appliquer une érosion sur l'image binaire en utilisant comme élément structurant :

- un carré de taille :
  - 3×3 pixels,
  - 5×5 pixels.
- un disque de taille :
  - 3×3 pixels,
  - 5×5 pixels.

Commenter et comparer les résultats.

6.2.2 Dilatation

19) Appliquer une dilatation sur l'image binaire en utilisant les éléments structurants utilisés dans la question précédente. Commenter et comparer les résultats.

6.2.3 Ouverture et fermeture

Une ouverture est une érosion suivie d'une dilatation et une fermeture est une dilatation suivie d'une érosion. Pour effectuer ces opérations, il est possible d'utiliser les fonctions définies précédemment ou de manière plus simple les fonctions `imopen` et `imclose`.

6.3 Autres opérations sur images binaires

La fonction `imclearborder` est une fonction morphologique qui permet de supprimer des régions qui sont au contact des bords de l'image binaire. La fonction `bwareaopen`, basée sur une analyse en composantes connexes, permet de supprimer des régions de trop petites tailles dans une image binaire. La fonction `imfill` est une fonction morphologique qui permet de combler les "trous" dans les régions d'une image binaire.

20) En utilisant ces trois fonctions, traiter l'image acquise et binarisée afin d'obtenir une image dans laquelle les formes correspondent au mieux aux objets de la scène réelle. Utiliser d'autres fonctions morphologiques si nécessaire.

6.4 Analyse des régions

Les fonctions suivantes permettent d'analyser les objets dans une image binaire :

- `bwlabel` : attribue une étiquette aux pixels de chaque région représentant la forme (pixels blancs connexes) présente dans une image binaire et retourne le nombre de ces régions (analyse en composantes connexes),
- `bwselect` : sélectionne une ou plusieurs régions représentant la forme dans une image binaire,
- `bwperim` : détermine les pixels contours d'une image binaire avec un voisinage 4 ou 8 (périmètre des objets dans une image binaire),
- `bwarea` : calcule le nombre de pixels représentant la forme dans une image binaire (surface des objets dans une image binaire),
- `bweuler` : retourne le nombre de régions représentant la forme moins le nombre de trous dans ces formes dans une image binaire.

21) Sur l'image binaire précédemment obtenue, utiliser les fonctions `bwlabel`, `bwperim`, `bwarea` pour afficher sur l'image binaire et à proximité de chaque région, sa surface, son périmètre et son numéro. Pour cela, utiliser la fonction `text` permettant de superposer du texte à la figure et la fonction `int2str` qui permet de transformer une valeur entière en une chaîne de caractère.

7 Extraction de caractéristiques et représentation dans l'espace d'attributs

On considère 9 images issues de la base d'images de texture couleur VisTex ('1.bmp' à '9.bmp') :
- les images 1 à 3 représentent une écorce d'arbre,
- les images 4 à 6 représentent un pelage,
- les images 7 à 9 représentent un métal.

Le code suivant est implémenté sous Matlab :

```
chemin = 'C:\... chemin vers les images VisTex ...\';
Att1=[];
for i=1:9
    fichier_image = [chemin int2str(i) '.bmp'];
    num_classe(i) = floor((i-1)/3) + 1;
    I = imread(fichier_image);
    IG = rgb2gray(I);
    Att = mean(mean(IG));
    Att1 = [Att1 Att];
end
Att1_1=Att1(num_classe==1);
Att1_2=Att1(num_classe==2);
Att1_3=Att1(num_classe==3);

plot(Att1_1,1,'c*',Att1_2,1,'b*',Att1_3,1,'r*');
```

FIGURE 3 – Extraction de caractéristiques et représentation dans l'espace d'attributs.

22) Exécuter ce code et expliquer ce que chaque ligne effectue en vous aidant de l'aide Matlab. Quelle conclusion peut-on tirer de l'analyse du graphique ?

La matrice de co-occurrences est un descripteur permettant de caractériser la texture dans une image. Cette matrice est calculée à partir d'une image en niveaux de gris grâce à la fonction `graycomatrix`. Une

fois la matrice calculée, on peut en extraire des statistiques comme le contraste, l'homogénéité ou l'énergie, qui vont permettre de caractériser la texture contenue dans l'image. Ces statistiques sont obtenues grâce à la fonction `graycoprops`.

23) Reprendre le code précédent en extrayant maintenant l'homogénéité de la matrice de co-occurrences à la place de la moyenne des niveaux de gris. Analyser la représentation des images dans ce nouvel espace.