Traitement d'images

Travaux pratiques

Reconnaissance des formes
Application à la reconnaissance de caractères et au tri postal

---

Spécialité Génie Industriel
CING3, parcours *Production industrielle*

Vandenbroucke Nicolas nicolas.vandenbroucke@eilco-ulco.fr

La reconnaissance automatique de l'écriture est un domaine de recherche qui a trouvé une application à grande échelle dans le tri du courrier. Les enveloppes passent devant une caméra, et chaque image est traitée automatiquement par une machine qui localise le code postal et le reconnaît.

Les images seront acquises et analysées à l'aide du logiciel Matlab. Matlab est associé à des boites à outils appelé TOOLBOX permettant d'accéder à des fonctions spécifiques à un domaine d'application comme le traitement d'images par exemple. Les TP de traitement d'images réalisés avec Matlab nécessitent ainsi la toolbox **Image Acquisition** et la toolbox **Image Processing**.

1 Acquisition

1.1 Mise au point du système de vision

Le matériel d'acquisition est composé de :

- une caméra monochrome IDS uEye de modèle UI-1240ML-C, de résolution 1280×1024 et équipée d'un capteur $1/1.8''$ (voir datasheet),
- un objectif Fujinon de focale $f = 25$ mm,
- deux sources à Leds alimentées en courant alternatif 220 V-50 Hz.

La caméra est fixée sur un statif et est reliée au PC par le port USB.
Un pilote Windows spécifique permet de communiquer entre le PC et la caméra.
La fonction `imaqhwinfo` permet l'obtention d'informations sur le matériel et les pilotes installés.

L'acquisition d'une image sous Matlab peut-être réalisée de deux manières :

- par la création d'un objet d'entrée vidéo en utilisant la fonction `videoinput`. Cette fonction affiche également les principales propriétés de l'objet d'entrée vidéo créé.
- par l'utilisation d'une interface en utilisant la fonction `imaqtool`.

La première méthode nécessite d'utiliser, dans un script, les instructions permettant l'acquisition d'une image. Elle permet d'ajouter dans ce même script les instructions qui permettent de traiter l'image dès l'acquisition effectuée. Cette méthode est présentée en annexe (voir page 15).

La seconde méthode lance une application permettant d'acquérir une image mais oblige à enregistrer cette image avant de la traiter. C'est cette méthode qui sera utilisée dans ce TP. Le lancement de cette application ouvre la fenêtre de la figure 1

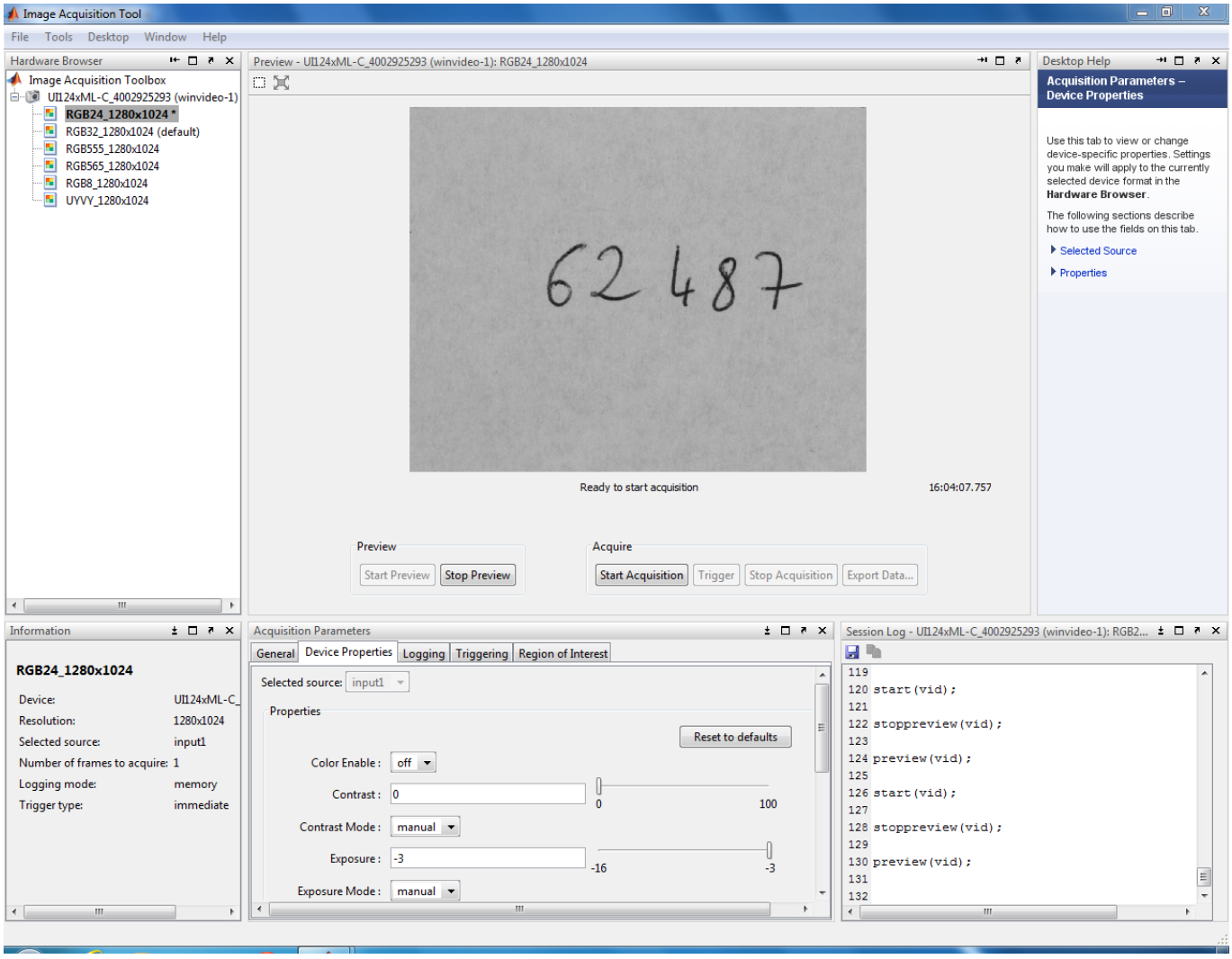


Figure 1 – Application d'acquisition d'image sous Matlab

Pour acquérir une image ou une vidéo (séquence d'images), il faut alors suivre les étapes suivantes :

1. Choisir le format d'image : on utilisera des images 1280×1024 au format RGB codées sur $3 \times 8 = 24$ bits.
2. Fixer les paramètres d'acquisition (voir figure 2) : on veillera en particulier à effectuer tous les réglages en mode **manuel**, à minimiser les paramètres de gain et de contraste qui ont pour effet d'accentuer le bruit et à maximiser le temps d'exposition afin de fermer l'objectif et afin d'atténuer l'effet de scintillement de la source lumineuse.
3. Régler le nombre d'images à acquérir et le moment de leur déclenchement : une seule image devra être acquise immédiatement après l'appui sur le bouton Start Acquisition.
4. Démarrer la prévisualisation : l'appui sur le bouton Start Preview permet de visualiser l'image. Le réglage du système peut alors être effectué et les paramètres ajustés si nécessaire.
5. Acquérir l'image : l'appui sur le bouton Start Acquisition déclenche l'acquisition de l'image.
6. Enregistrer l'image : l'appui sur le bouton Export Data permet d'enregistrer l'image dans un fichier. On choisira les fichiers au format PNG.

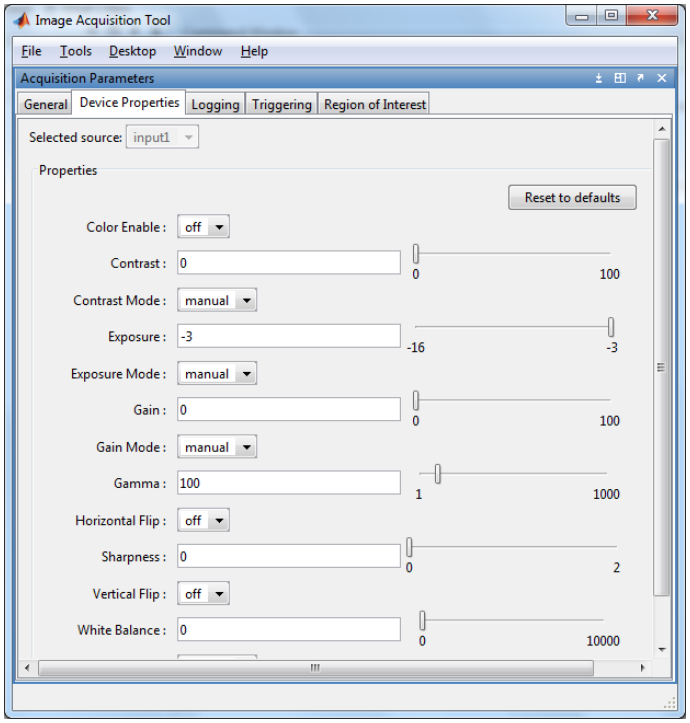


Figure 2 – Paramètres d'acquisition

L'image de la figure 3 représente le code postale d'une enveloppe acquise par un système d'acquisition.

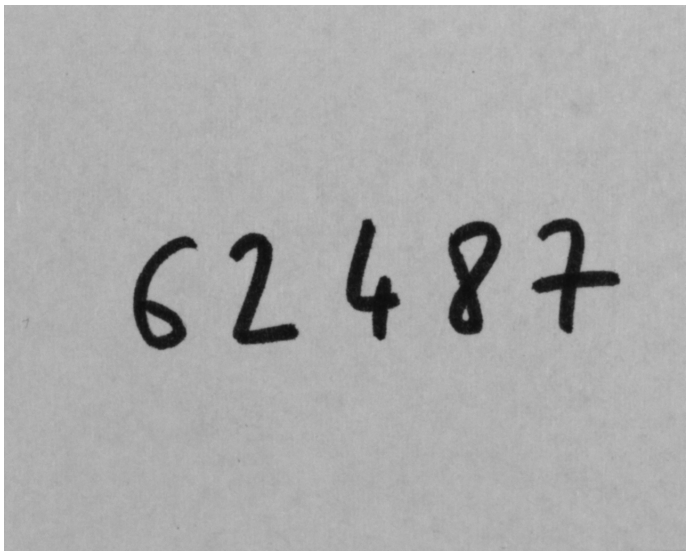


Figure 3 – code postal d'une enveloppe.

1) Sur la feuille de papier où sont imprimés des codes postaux avec des polices différentes, ajouter un code postal en écriture manuscrite. Placer ce code dans le champ de vision de la caméra et, en utilisant l'application Matlab :

- Prévisualiser l'image **monochrome** et ajuster les paramètres de réglage.
- Ajuster la distance de travail de sorte à obtenir un champ de vision de surface de 5 cm $\times$ 4 cm.
- Effectuer le réglage de la mise au point et de l'ouverture de l'objectif de la caméra. On veillera

à régler correctement la caméra et les paramètres d'acquisition afin d'obtenir une image de bonne qualité avec le minimum d'ombres et de reflets. Indiquer les valeurs de réglage. **Les images seront acquise directement en niveau de gris**.
- Enregistrer l'image au format PNG sous le nom `Code0.png`.

1.2 Acquisition des images

Afin de développer et tester l'algorithme de reconnaissance selon une approche de classification supervisée, il est nécessaire de constituer une base de données d'images où sont présents différents chiffres à identifier. Cette base est divisée en deux parties :
- une base d'apprentissage dans laquelle figure des prototypes (observation de classe connue) des 10 chiffres à reconnaitre,
- une base de test dans laquelle figure différents exemples de code postaux à lire.

2) Répéter les acquisitions afin de constituer la base de test sous la forme d'une séquence de douze images qui sera utilisée pour mettre au point le programme de localisation et de tri. Chaque image doit contenir un code avec une écriture différente et sera numérotée de 0 à 11.

3) Répéter les acquisitions afin de constituer la base d'apprentissage sous la forme d'une séquence de dix images de telle sorte à ce que chaque image contienne cinq prototypes d'un même chiffre. Chaque image doit donc contenir le même chiffre avec une écriture différente et sera numérotée de 0 à 9 sous le nom `Chiffre0.png`...

Dans la suite des manipulations, on mettra au point les algorithmes d'abord sur la première image uniquement avant de les tester sur l'ensemble des images ensuite, une fois les paramètres correctement ajustés.

2 Prétraitement

Afin de détecter les chiffres présents, on propose de procéder à une binarisation. Généralement, on attribue les pixels blancs (égales à 1) à la **forme** de l'objet présent dans une image binaire (avant plan) et les pixels noirs (égales à 0) au **fond** (arrière plan).

4) Ecrire un nouveau script permettant :
- d'ouvrir et afficher une image de la séquence (on utilisera tout d'abord l'image `Code0.png`),
- transformer l'image couleur acquise en image monochrome si nécessaire et d'afficher cette image,
- de calculer et d'afficher l'histogramme de l'image monochrome,
- de binariser cette image de telle sorte à obtenir les chiffres en blanc et un fond en noir. Si plusieurs binarisations sont nécessaires, utiliser les opérateurs logiques (fonctions `imcomplement` (`~`), `or` (`|`), `xor` et `and` (`&`)) pour obtenir l'image. Indiquez la valeur du ou des seuil(s) de binarisation.

La fonction `imclearborder` est une fonction 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 qui permet de combler les "trous" dans les régions d'une image binaire.

5) Utiliser les fonctions morphologiques nécessaires pour obtenir une image similaire à celle de la figure 4 ((fonctions `imdilate`, `imerode`, `imclose`, `imopen`, `strel`) ainsi que les fonctions précédentes

nécessaires.

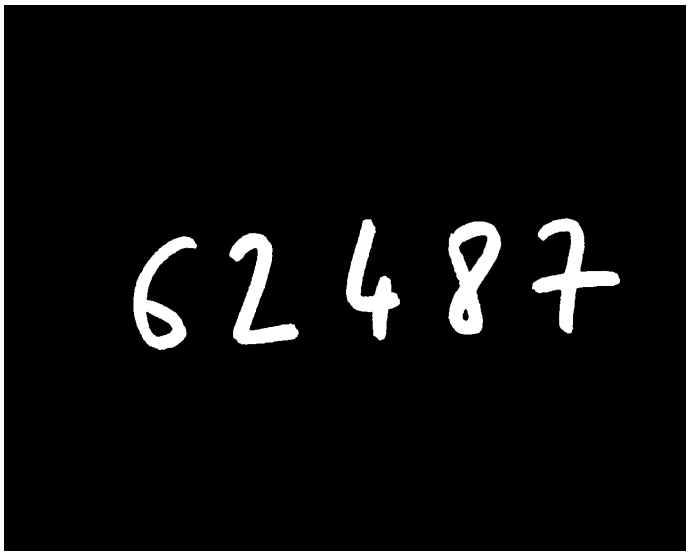


Figure 4 – prétraitement

3 Localisation des chiffres du code postal

La fonction `bwlabel` détermine les régions de pixels connexes dans une image binaire tandis que la fonction `bwboundaries` réalise la même opération mais donne les coordonnées des points de contour de chaque région. Les images résultantes de ces deux fonctions sont des images indexées où chaque index correspond à une étiquette (label), c'est à dire une région. Ainsi, les pixels d'une même région possède la même étiquette. Cette image peut également être convertie en une image couleur grâce à la fonction `label2rgb`. Enfin, la fonction `bwselect` permet de sélectionner une ou plusieurs régions particulières d'une image binaire et donc de supprimer toutes les autres.

Après l'appel de la fonction `bwlabel` (ou `bwboundaries`) qui effectue une analyse en composantes connexes et retourne le nombre de régions présentes dans l'image, il est nécessaire d'utiliser une structure répétitive (boucle `for`) afin d'accéder à chaque région indépendamment et isoler ainsi le chiffre correspondant. Le squelette de programme suivant indique la trame du code à utiliser pour réaliser ces opérations :

```
[L,N] = bwlabel(BW); % BW représente l'image binaire pré-traitée
hold all; % permet de ne pas effacer la figure en cours
for k = 1:N
    bin = (L==k); % bin contient les pixels dont l'étiquette (label) est k
        % (k variant de 1 à N)
    % Calcul de paramètres...
    % ...
    % Affichage des paramètres...
    % ...
end
```

La fonction `find` retourne les coordonnées des cellules d'un tableau (pixels d'une image) qui vérifie une condition (éléments non nuls). Ainsi, la commande suivante permet d'affecter aux variables x et y respectivement les abscisses et les ordonnées des pixels d'une image dont les valeurs sont différentes de 0.

```
[x y] = find(bin);
```

Attention, les coordonnées des pixels de l'image ne sont pas exprimées dans le même repère que celui de la figure dans laquelle on souhaite superposer du texte ou des graphiques.

6) En utilisant les fonctions précédentes, proposer et développer une méthode qui permet de localiser et isoler chacun des chiffres présents dans le code postal de l'image (voir figure 5). On pourra notamment utiliser les fonctions `min` et `max` afin de déterminer les coordonnées minimum et maximum des pixels de chaque chiffre afin de les délimiter.

Figure 5 – Chiffre localisé et isolé

4 Extraction des caractéristiques des chiffres

La reconnaissance des chiffres du code postal est un problème difficile, surtout lorsqu'il s'agit d'écriture manuscrite car chaque ligne de chiffres est écrite par une personne différente et il existe donc une grande variabilité de l'écriture, lorsque l'on passe d'une personne à l'autre. Même pour une personne donnée, l'écriture n'est jamais parfaitement stable.

Les méthodes de reconnaissance de chiffres fonctionnent généralement en deux étapes. La première étape consiste à caractériser la forme du chiffre, en détectant dans l'image des zones particulières.

La caractéristique utilisée ici est la cavité. Les cavités se définissent par leur direction d'ouverture. Cinq types de cavités sont ainsi définis :

- cavité *Nord* : vers le haut,
- cavité *Est* : vers la droite,
- cavité *Sud* : vers le bas,
- cavité *Ouest* : vers la gauche,
- cavité *Centrale* : au centre.

Par exemple, un pixel de l'image appartient à une cavité *Est* si, et seulement si, les trois conditions suivantes sont vérifiées :

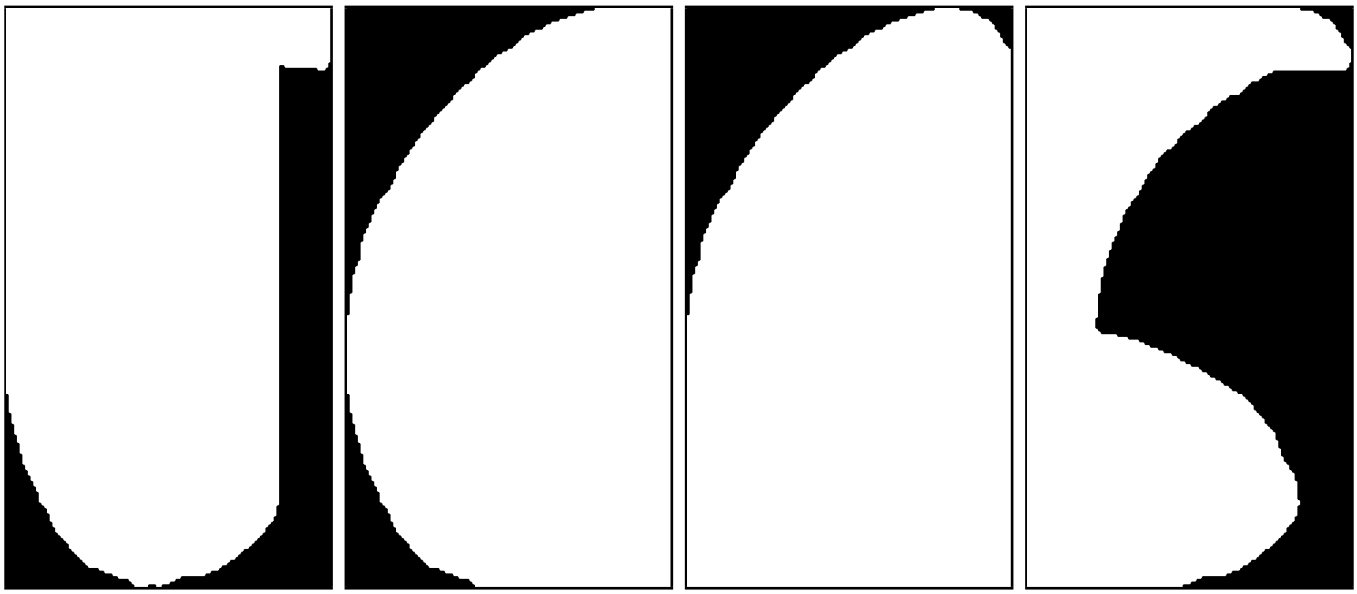
- ce pixel n'appartient pas au tracé du chiffre ;

- en se déplaçant en ligne droite vers l'est, à partir de ce pixel, on ne rencontre pas le tracé ;
- en se déplaçant en ligne droite vers le sud, l'ouest, ou le nord, à partir de ce pixel, on rencontre le tracé.

Un pixel appartient à une cavité *Centrale* si, et seulement si, les deux conditions suivantes sont vérifiées :

- ce pixel n'appartient pas au tracé du chiffre ;
- en se déplaçant en ligne droite vers l'est, le sud, l'ouest, ou le nord, à partir de ce pixel, on rencontre le tracé.

Afin de mettre en évidence les pixels appartenant aux différents type de cavité, on réalise une première série de traitements morphologiques à partir de l'image 5. La figure 6 montre le résultat de ces traitements.



(nord) (est) (sud) (ouest)

Figure 6 – traitements réalisés sur l'image de la figure 5.

Cette première opération consiste à dilater chaque image de chiffre selon les 4 directions : nord – sud – est – ouest. Pour cela, il faut utiliser la fonction `imdilate`.

Afin d'étirer le tracé selon une direction donnée, on utilise des éléments structurants qui représentent des lignes. Les lignes horizontales ont une taille impaire 2 fois supérieure à la résolution horizontale de l'image du chiffre, notée H. Les lignes verticales ont une taille impaire 2 fois supérieure à la résolution verticale de l'image du chiffre, notée V (voir exemple ci-dessous pour la direction "Est").

0	0	0	0	$\cdots$	0	0	0	0	0	1	1	1	1	1	1	$\cdots$	1	1	1	1

$\underbrace{\qquad}_{2\times H+1}$

La création d'un élément structurant prédéfini peut être réalisée avec la fonction `strel` mais également en utilisant les fonctions `zeros` et `ones` qui créent respectivement des tableaux de 0 et de 1.

La taille de l'image d'un chiffre peut-être calculée à partir des coordonnées minimums et maximums des pixels ou en utilisant les fonctions `size` ou `length`.

Enfin, à partir d'un premier élément structurant, on peut déduire les trois autres en utilisant les fonctions `fliplr` (symetrie gauche–droite), `flipud` (symétrie haut–bas), `rot90` (rotation de 90 °) ou `transpose` (transposition).

Pour l'image d'un même chiffre, on obtient ainsi 4 images dilatées (une par direction) appelée respectivement (nord), (est), (sud) et (ouest).

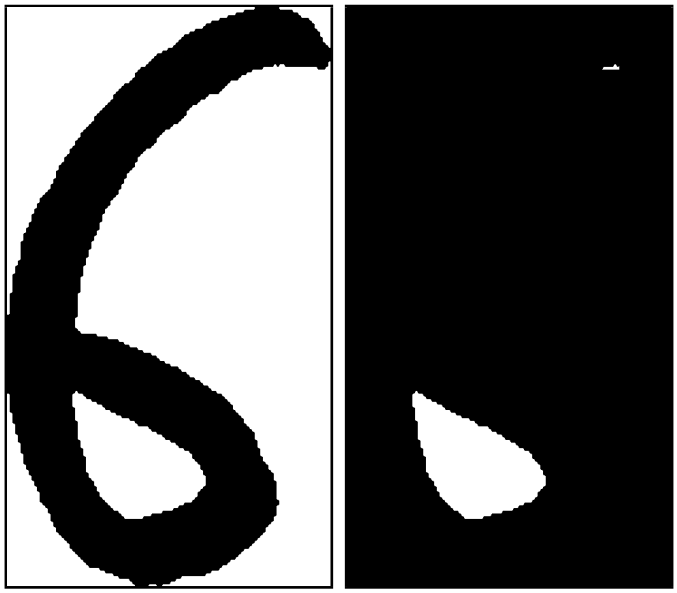
7) Compléter votre script afin d'appliquer ces traitements sur un chiffre localisé d'un code postal et appliquer cette procédure pour chaque chiffre détecté.

Pour **CHAQUE** cavité, une seconde série de traitements est ensuite effectuée à partir des images (nord) à (ouest).

Par exemple, pour déterminer la cavité *centrale*, on effectue l'opération logique suivante ou "Chiffre" représente l'image du chiffre analysé :

CENTRE = (est) ET (ouest) ET (sud) ET (nord) ET inverse(Chiffre)

La figure 7 montre le résultat de ces traitements pour la détection des pixels appartenant à des cavités *Centrale*.



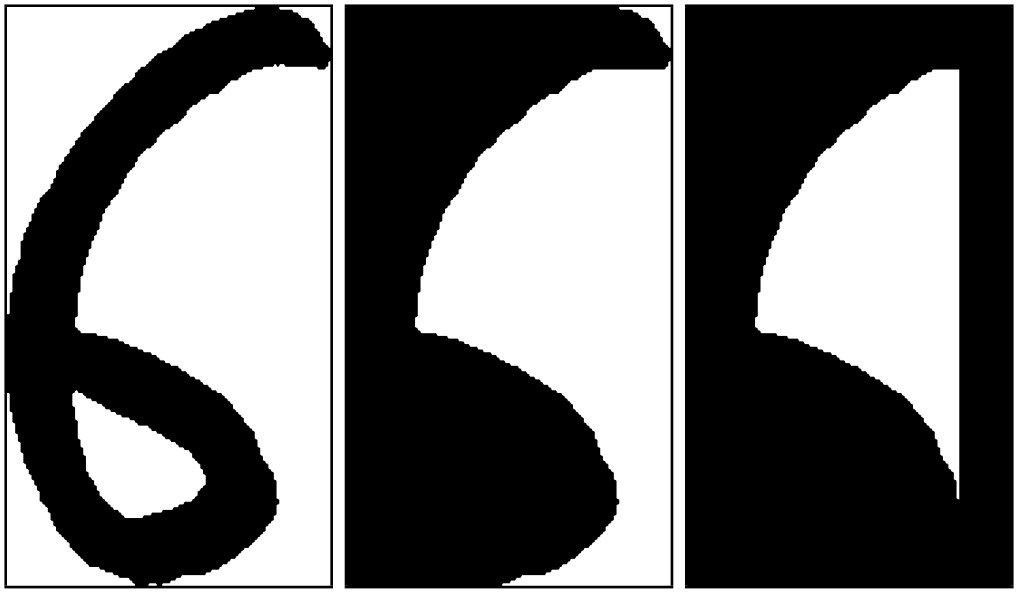
(a) (b)

Figure 7 – traitements réalisés pour la détection des cavités *Centrale*.

Pour déterminer la cavité *est*, on effectue l'opération logique suivante :

EST = (est) ET inverse((ouest)) ET (sud) ET (nord) ET inverse(Chiffre)

La figure 8 montre le résultat de ces traitements pour la détection des pixels appartenant à des cavités *Est*.



(a) (b) (c)

Figure 8 – traitements réalisés pour la détection des cavités *Est*.

8) A partir des traitements effectués sur les images des figures 7 et 8, permettant de détecter respectivement les pixels des cavités *Centrale* et les pixels des cavités *Est*, déduire les traitements à effectuer pour détecter les pixels des cavités *Nord*, *Ouest* puis *Sud*.

9) Compléter le script précédent afin d'estimer les 5 types de cavités associés à chaque chiffre.

5 Classification des chiffres

Une fois que les cavités sont détectées, elles sont analysées par un système que l'on appelle un classifieur. Ce système décide si le chiffre est un zéro, un un, un deux, etc... La classification s'effectue en deux étapes :

- Un apprentissage hors-ligne permettant de construire les classes en présence à partir de l'ensemble des prototypes. Pour cela, les chiffres présents dans les images prototypes sont décrits par un ensemble d'attributs. Chacun de ces chiffres est ainsi représenté par un vecteur d'attributs dans un espace de décision. Un classifieur est ensuite élaboré afin de déterminer la classe d'appartenance d'un chiffre en fonction de ses caractéristiques.
- Une classification en-ligne consistant à appliquer les règles de décision déterminées lors de l'apprentissage afin de classer chacun des chiffres d'un code postal en lui assignant une étiquette.

5.1 Apprentissage hors-ligne

10) Sachant que, pour chaque chiffre détecté, il est possible de déterminer la surface de chaque type de cavité, proposer un espace de décision permettant de repérer chaque chiffre par un point dans cet espace.

L'apprentissage consiste à ouvrir chacune des images de la base d'apprentissage, calculer et enregistrer dans une variable le vecteur d'attributs de chaque prototype ainsi que la classe correspondante. Pour cela, nous pouvons utiliser une boucle répétitive de la manière suivante :

```
%% Reconnaissance de caractères

clear all; % efface toutes les variables en mémoire
close all; % ferme toutes les fenêtres ouvertes
clc; % efface le contenu de la fenêtre de commande

nb_classe = 10; % défini le nombre de classes

%% Apprentissage

tic % prise du temps pour chronomètrage

nb_image = 0; % initialise le nombre d'images
for i=1:nb_classe % boucle de parcours des classes
        % Constitution du nom du fichier image
        % ...
        % Ouverture de l'image d'une classe de chiffres
        % ...
        % Localisation des chiffres (avec N, le nombre de chiffres dans l'image)
        % ...
        for k=1:N % boucle de parcours des images

                % Enregistrement du numéro de la classe dans un tableau et
                % incrémentation du nombre d'images
                % ...
                % Accès à chaque chiffre k
                % ...
                % Calcul des attributs pour chaque prototype
                % (pourcentage de cavité + autres)
                % ...
                % Affectation à une variable mise à jour
                % ...

        end %fin de la boule
end %fin de la boule

toc % Mesure du temps de traitement
```

11) Compléter le script précédent afin de réaliser l'apprentissage sur la base de données images réalisée précédemment et déterminer une matrice à 2 dimensions des caractéristiques de chaque chiffre (valeurs des attributs pour chaque prototype), un vecteur qui enregistre la classe d'appartenance de chaque prototype ainsi qu'une matrice des caractéristiques moyennes de chaque chiffre (valeurs moyennes des attributs pour chaque classe).

5.2 Classification en-ligne

12) Proposer une méthode simple de classification, basée sur la recherche du plus proche voisin dans l'espace de décision précédemment déterminé, permettant de classer chaque chiffre dans cet espace et mettre en oeuvre cette méthode afin d'afficher dans l'image initiale le code postal lu.

13) Valider votre programme sur les codes postaux de la base de test et conclure.

14) Proposer une méthode simple de classification, basée sur la recherche de la plus proche moyenne dans l'espace de décision précédemment déterminé, permettant de classer chaque chiffre dans cet espace et mettre en oeuvre cette méthode afin d'afficher dans l'image initiale le code postal lu.

15) Valider votre programme sur les codes postaux de la base de test et comparer avec la méthode précédente.

La fonction `regionprops` permet de mesurer différents paramètres de régions contenues dans une image d'étiquettes ou une image binaire, et donc obtenus après une analyse en composantes connexes tels que : la surface, le périmètre, le diamètre équivalent, la longueur de l'axe principal d'inertie (longueur de la région), la longueur du second axe d'inertie (largeur de la région), l'angle de l'axe principal d'inertie (orientation de la région), l'excentricité, les coordonnées du centre de gravité, les coordonnées du cadre circonscrit à la région (boundingbox), le niveau de gris moyen...

16) En utilisant la fonction `regionprops`, ajouter un ou plusieurs attributs à votre espace de décision qui sont invariants au changement d'echelle afin d'améliorer les résultats de votre programme.

On suppose que les codes postaux peuvent apparaitre écrit avec un angle non nul par rapport à l'horizontal et que les chiffres peuvent se toucher dans l'écriture.

17) Améliorer votre programme en recalant en rotation les images acquises après avoir mesuré l'angle de l'axe principal d'inertie du code postal inscrit.

18) Améliorer votre programme en séparant des chiffres qui pourraient se toucher.

Annexe

Acquisition d'images sous Matlab

Un objet d'entrée vidéo est un objet de type structure sur laquelle il est possible de régler plusieurs propriétés :

- les paramètres liés au périphérique comme :
  - le format de l'image,
  - l'espace de codage de l'image (RGB, YCbCr, niveaux de gris, ...)...
- Les paramètres liés à l'acquisition comme :
  - la luminosité (Brightness),
  - le contraste (Contrast),
  - le temps d'exposition (Exposure),
  - la correction gamma (Gamma),
  - la netteté (Sharpness)...

La fonction `propinfo` (ou les fonctions `get` et `set`) ainsi que la fonction `inspect` permettent d'accéder et de connaître les caractéristiques détaillées de chaque propriété. Une propriété peut également être un objet de type structure avec ses propres propriétés. L'accès à cet objet s'adresse de la façon suivante : *nom.propriete*. Attention, certaines propriétés ne sont accessibles qu'en lecture seule selon qu'une acquisition est en cours ou non.

La fonction `preview` permet de créer une fenêtre d'aperçu afin de visualiser la scène observée et la fonction `closepreview` permet de fermer cette fenêtre. C'est ainsi qu'il est possible de régler certain paramètre d'acquisition.

Réglages des paramètres intrinsèques et extrinsèques

Le programme suivant utilise les fonctions précédentes afin de configurer l'acquisition d'un image et effectuer les réglages nécessaire avant d'acquérir l'image :

```
%% ACQUISITION D IMAGES SOUS MATLAB

clear all; % efface toutes les variables en mémoire
close all; % ferme toutes les fenêtres ouvertes
clc; % efface le contenu de la fenêtre de commande

%% Affichage des informations logicielles et matérielle

info_id = imaqhwinfo('winvideo')
info_dev = imaqhwinfo('winvideo',1)
formats = info_dev.SupportedFormats % affiche les formats d'images possible

%% Création d'un objet d'entrée vidéo

vid = videoinput('winvideo',1,'RGB8_1280x1024'); % associe un objet d'entrée vidéo
        % au périphérique matériel de la caméra en transmettant le numéro
        % d'identification de ce périphérique avec un format d'image 1280 x 1024.

% Informations sur les propriétés de "Source"
information = get(vid)

% Réglage du périphérique
propinfo(vid,'VideoFormat') % affiche les informations sur le format vidéo
propinfo(vid,'ReturnedColorSpace') % affiche les informations sur l'espace couleur
set(vid,'ReturnedColorSpace') % permet de visualiser les réglages possibles
        % ou : get(vid,'ReturnedColorSpace')
set(vid,'ReturnedColorSpace','grayscale') % sélectionne l'espace de représentation
        % (ici monochrome)
        % get(vid,'ReturnedColorSpace') pour affichage

% Affichage des informations sur les propriétés
propinfo(vid,'Source')

%% Accès à la source vidéo

Source = getselectedsource(vid); % <=> Source = vid.Source;

% Informations sur les propriétés de la source vidéo
information = get(Source) % <=> set(Source)
% Affichage des valeurs minimales et maximales de réglage des principales propriétés
info = propinfo(Source,'Gain'); disp('Gain'); disp(info.ConstraintValue);
info = propinfo(Source,'Contrast'); disp('Contrast'); disp(info.ConstraintValue);
info = propinfo(Source,'Exposure'); disp('Exposure'); disp(info.ConstraintValue);
info = propinfo(Source,'Gamma'); disp('Gamma'); disp(info.ConstraintValue);
info = propinfo(Source,'Sharpness'); disp('Sharpness'); disp(info.ConstraintValue);

% Affichage et réglage des propriétés de la la source vidéo
inspect(Source); % ouvre une fenêtre avec les différents paramètres à régler
        % et permet le réglage de ces paramètres de manière interactive

%% Réglage du système
preview(vid); % ouvre la fenêtre de prévisualisation de l'image
pause; % attend que l'utilisateur appui sur une touche depuis l'éditeur de commande
        % avant de continuer afin d'effecuer les réglages
closepreview; % ferme la fenêtre de prévisualisation
```

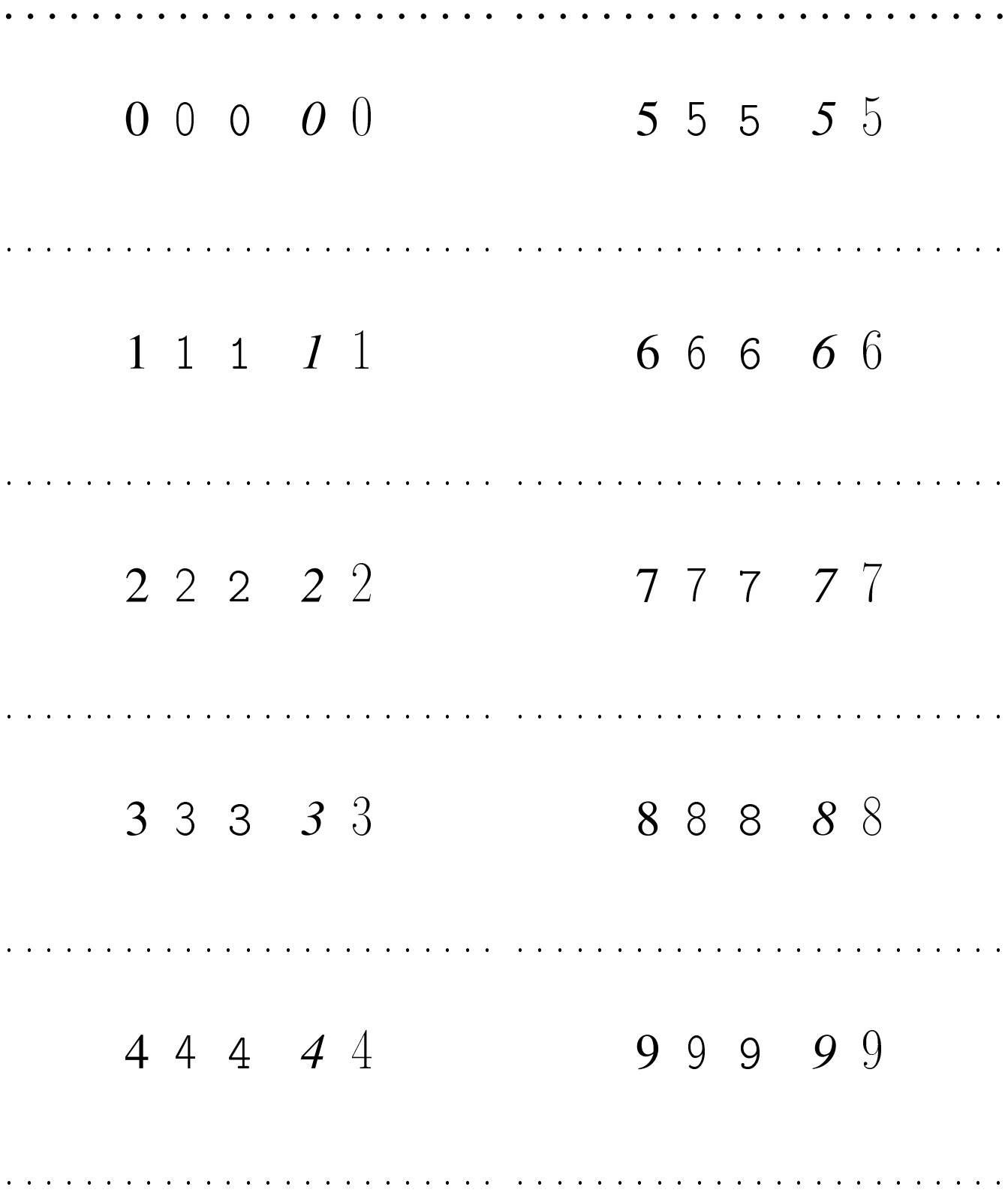
Acquisition d'une image

L'acquisition d'UNE seule image se fait avec la fonction `getsnapshot`. La fonction `pause` peut être utilisée afin de mettre le programme en attente pendant le temps du réglage et avant d'acquérir l'image.

Le programme suivant permet de fixer les paramètres de réglages définis précédemment et acquérir une image avant de l'enregistrer au format brut :

```
%% ACQUISITION D IMAGES SOUS MATLAB

clear all; % efface toutes les variables en mémoire
close all; % ferme toutes les fenêtres ouvertes
clc; % efface le contenu de la fenêtre de commande


%% Création d'un objet d'entrée vidéo

vid = videoinput('winvideo',1,'RGB24_1280x1024'); % associe un objet d'entrée vidéo
        % au périphérique matériel de la caméra en transmettant le numéro
        % d'identification de ce périphérique avec un format d'image 1280 x 1024.

% Réglage du périphérique
set(vid,'ReturnedColorSpace','grayscale') % sélectionne l'espace de représentation
        % ('rgb' ou 'grayscale')


%% Accès à la source vidéo

Source = getselectedsource(vid); % <=> Source = vid.Source;

% Affichage et réglage des propriétés de la la source vidéo
inspect(Source); % ouvre une fenêtre avec les différents paramètres à régler
        % et permet le réglage de ces paramètres de manière interactive


%% Acquisition
preview(vid); % ouvre la fenêtre de prévisualisation de l'image

% Réglage des paramètres
set(Source,'ExposureMode','manual')
set(Source,'Exposure',-10)
set(Source,'GaineMode','manual')
set(Source,'Gain',0)
set(Source,'Contrast',0)
set(Source,'Sharpness',0)
set(Source,'Gamma',100)
information = get(Source) % Affichage des réglages
pause;

% Prise d'images
Image = getsnapshot(vid);
closepreview; % ferme la fenêtre de prévisualisation

% Affichage de l'image acquise
figure; imshow(Image);
% ou : imshow(Image,'Border','tight','InitialMagnification',100);

% Enregisrement de l'image acquise au format BMP
imwrite(Image,'Image.bmp');
```

Chiffres imprimés



62487 59130

62487 59130

62487 59130

62487 *59130*

62487 59130