Traitement d'images

Travaux pratiques

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

Licence en Intelligence Artificielle

Travaux Pratiques de cours extrait de TP de M. Vandenbroucke Nicolas pour Spécialité Génie Industriel
CING3, parcours Production industrielle, EILCO, ULCO.

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 en python (Optionnellement avec l'outil Matlab. Les TP de traitement d'images réalisés avec Matlab nécessitentainsi 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 $1/1.8''$,
- 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'image de la figure 1 représente le code postal d'une enveloppe acquise par un système d'acquisition.

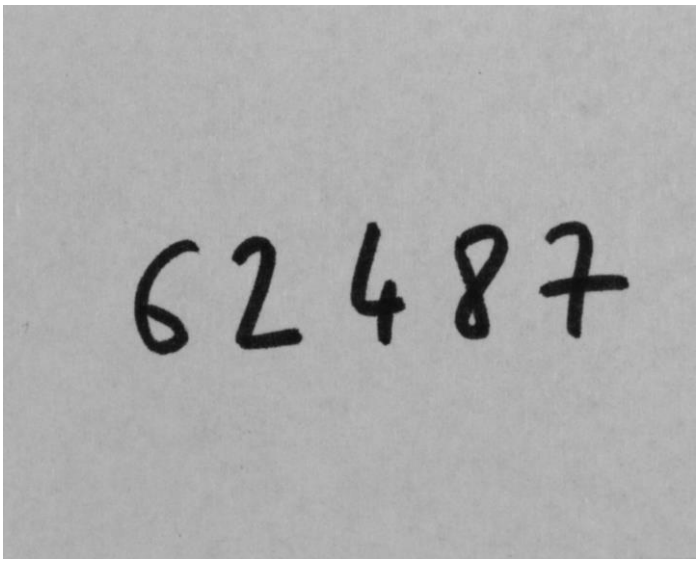


Figure 1 – code postal d'une enveloppe.

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 × 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 2 ((fonctions `imdilate`, `imerode`, `imclose`, `imopen`, `strel`) ainsi que les fonctions précédentes

nécessaires.

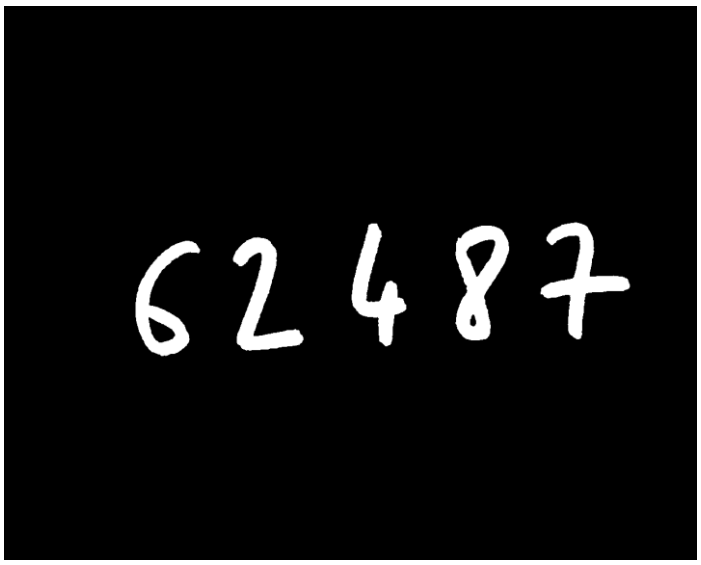


Figure 2 – prétraitement

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 3). 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.

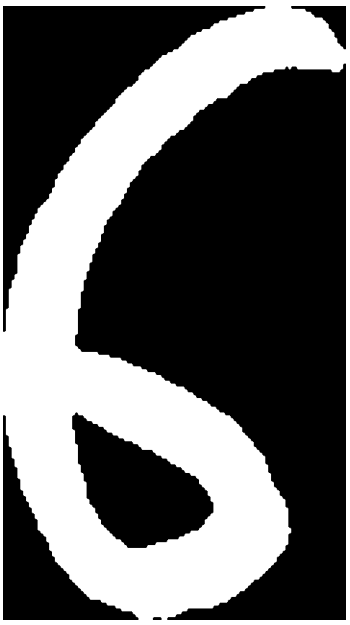


Figure 3 – Chiffre localisé et isolé

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 :

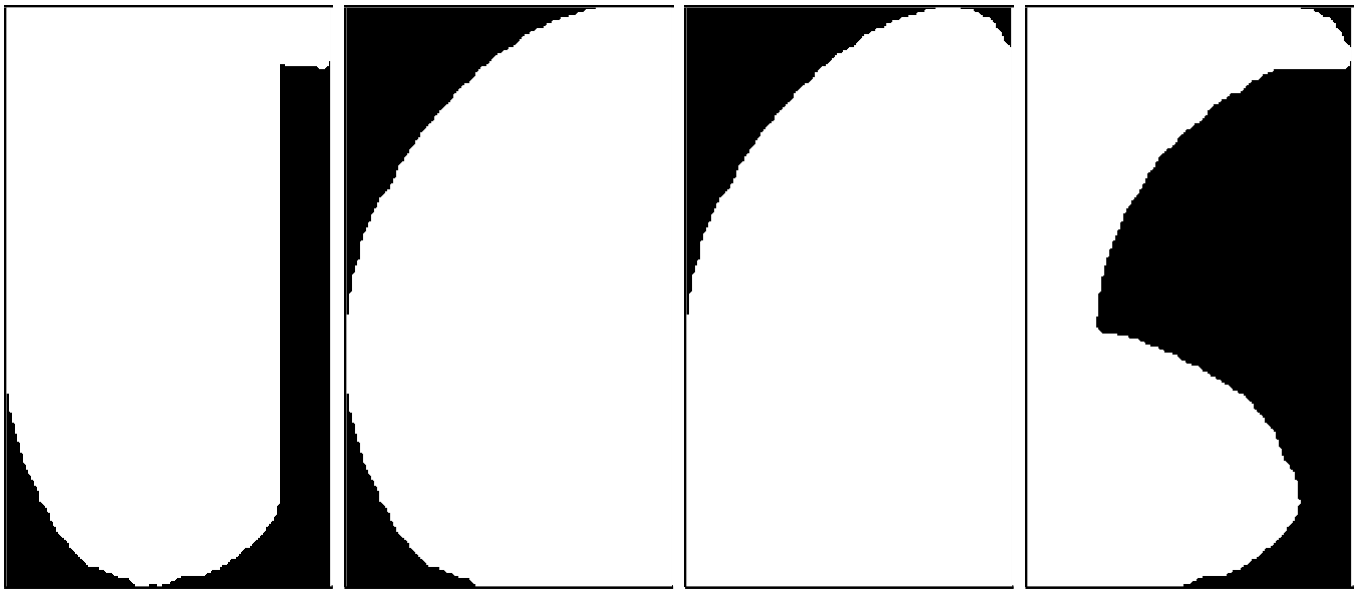
- ce pixel n'appartient pas au tracé du chiffre ;

- 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 3. La figure 4 montre le résultat de ces traitements.

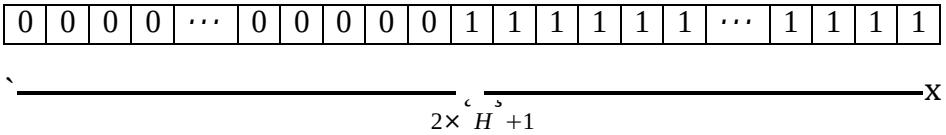


(nord) (est) (sud) (ouest)

Figure 4 – traitements réalisés sur l'image de la figure 3.

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").



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).

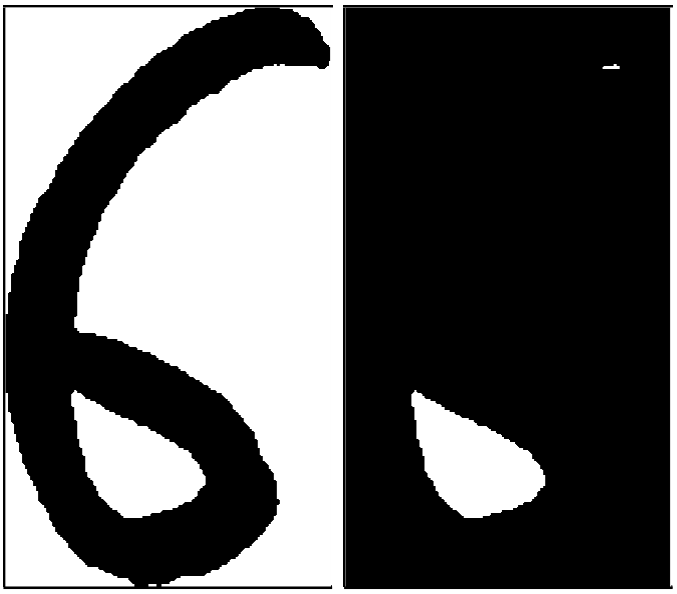
7) Compléter votre script afin d'appliquer ces traitements sur un chiffre localisé d'un code postal et appliquer cette procédure pour chaque chiffre détecté.

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



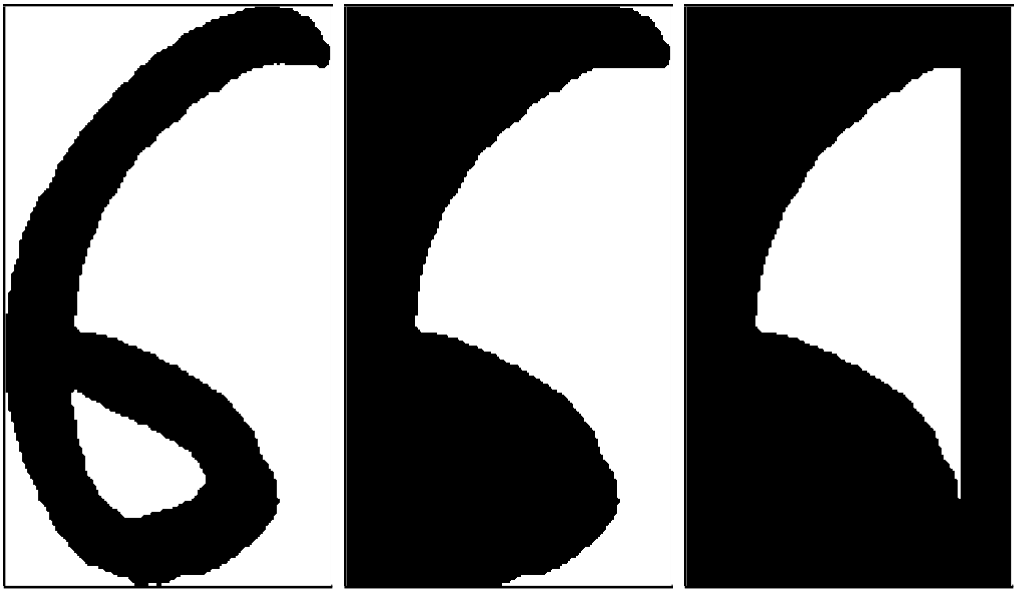
(a) (b)

Figure 5 – 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 6 montre le résultat de ces traitements pour la détection des pixels appartenant à des cavités *Est*.



(a) (b) (c)

Figure 6 – traitements réalisés pour la détection des cavités *Est*.

8) A partir des traitements effectués sur les images des figures 5 et 6, 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.