Reconnaissance de chiffres
Compte-rendu de TP - IN54

Adrien Berthet et Camille Mougin

UTBM - Automne 2014

1 Pré-traitements : localisation et extraction des chiffres manuscrits

Dans l'exemple étudié, la façon la plus simple de localiser les chiffres consiste d'abord à rechercher, à partir de l'histogramme de projections horizontales de pixels noirs, les plages correspondant à un nombre de pixels noirs non nul. Le début et la fin de chaque plage détectée sur l'histogramme des projections horizontales définissent une ligne de chiffres dans l'image. En appliquant le même principe (mais à partir d'un histogramme de projections verticales) sur chaque ligne détectée, on peut alors déterminer l'emplacement de chaque chiffre de l'image.

1.1 Recherche des lignes

Le principe est très simple, puisqu'il suffit dans un premier temps de parcourir chaque ligne pour vérifier qu'elle contient un pixel noir ou non. Pour accélérer le processus et parcourir rapidement l'image, un histogramme du nombre de pixels noirs est récupéré. Ceci est possible facilement car l'image est déjà binaire. On peut alors remarquer de façon évidente les différentes lignes. Les valeurs de celles-ci (début et fin) sont alors sauvegardées pour être réutilisées lors de la recherche des colonnes.

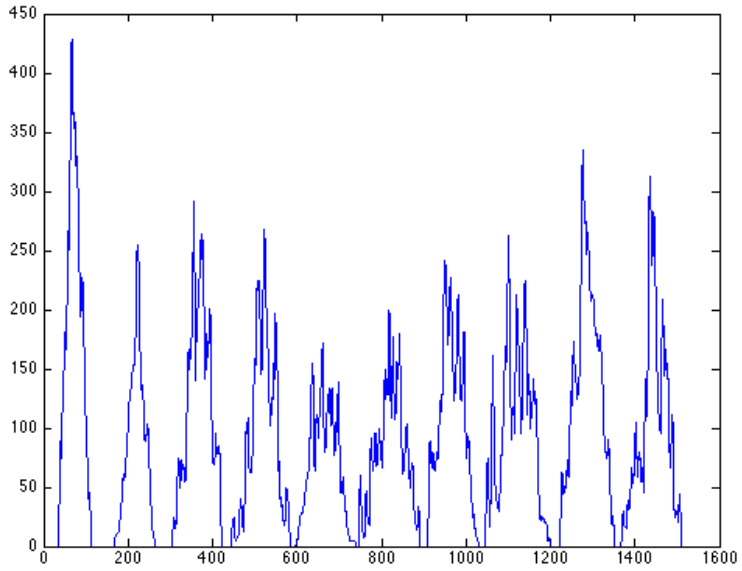

FIGURE 1 – Histogramme du niveau de noir

1.2 Recherche des colonnes pour chaque ligne

Le même principe utilisé précédemment est recommencé, avec cette fois un histogramme du niveau de noir sur chaque colonne, pour une ligne unique (soit une ligne de chiffre). On obtient alors plusieurs histogrammes semblables au précédent, qui permettent d'obtenir un tableau avec les colonnes englobant chaque chiffre pour chaque ligne.

1.3 Détermination du rectangle englobant

On pourrait penser qu'il n'y a plus d'étapes de découpe et qu'il faut uniquement assembler les coordonnées colonnes avec celles des lignes, mais il est nécessaire de repasser pour chaque chiffre dans le découpage des lignes, car ils possèdent un profil vertical beaucoup trop grand pour certains (puisqu'il s'agit du plus haut et plus bas chiffre pour chaque ligne). Ainsi, en effectuant une découpe plus précise, on obtient au final le résultat suivant.

FIGURE 2 – Résultat des découpes

1.4 Avantages et inconvénients d'une telle méthode

Cette méthode de localisation par projection est avantageuse puisque dans le cas ci présent, elle fournit un résultat parfait quant à la découpe des chiffres. Seulement, celle-ci peut perdre énormément en pertinence si les lignes de chiffres n'étaient pas aussi droites.
Imaginons deux lignes consécutives, écrites de façon à ce qu'elles montent toutes les deux vers le haut. On se retrouve avec le premier chiffre de la première ligne qui se retrouve à la même hauteur que le dernier chiffre de la deuxième ligne. Ainsi, la méthode de l'histogramme ne convient pas du tout car on ne pourrait faire la différence entre les deux lignes.
On peut également remarquer que cette méthode devient vite lourde pour les images de grande taille et comprenant un nombre important de caractères (en plus de chiffres), et il serait donc pertinent de trouver une méthode plus optimisée.

2 Classifieur 1 : profils et classifieur par distance euclidienne minimum

2.1 Extraction des profils

Chaque chiffre récupéré précédemment, il est désormais possible d'extraire les profils gauche et droit pour chacun, sur d composantes. Ainsi, pour chaque chiffre, on utilise l'algorithme suivant :

découper en 5 lignes avec *linspace*
for *chaque ligne* **do**
  while *pixel_gauche != noir* **do**
    profil gauche = pixel_gauche + 1
  end
  while *pixel_droit != noir* **do**
    profil droit = pixel_droit - 1
  end
end
normaliser le profil obtenu

Enfin, on récupère tous ces profils et on les classe en fonction du chiffre auquel ils appartiennent, pour ensuite effectuer la moyenne des profils d'un même chiffre.

2.2 Apprentissage du classifieur

2.3 Décision du classifieur

3 Classifieur 2 : Densités et K plus proches voisins

Liste des fonctions mentionnées dans ce chapitre : getdensity, computepdensities, learningclassifier2, decisionclassifier2

3.1 Principe et implémentation

Ce second classifieur vise à identifier un chiffre à partir des densités de pixels noirs obtenues dans les différentes zones qui le compose. La première étape consiste à diviser le rectangle encadrant le chiffre à identifier en m x n zones, puis de calculer la densité de pixels noirs dans chacune de ces zones.

Comme précédemment, le classifieur nécessite de passer par une phase d'apprentissage afin d'obtenir une base de densités de référence. Il est ensuite possible d'identifier le chiffre en comparant les densités obtenues avec les densités de référence. Les probabilités d'appartenance du chiffre à chacune des classes est finalement calculée en fonction du nombre de représentants de chaque classe parmis ses k plus proches voisins.

3.1.1 Fonctions utilisées

getdensity(rectangle, m, n) : density Entrées : Calcule la densité de pixels noirs dans chaque zone de l'image. Les zones sont obtenues par division du rectangle contenant le chiffre en m parties sur la hauteur et n parties sur la largeur. Les densités sont calculées en divisant le nombre de pixels noirs décomptés par la hauteur x la largeur du rectangle. Cette fonction retourne un vecteur de m x n composantes contenant les densités normalisées relatives à chaque zone.
computepdensities(vectordensity, vectordensitylearning, nbrectangleslearning, k) : pbelonging Entrées : Calcule la différence entre le vecteur de densités du chiffre à identifier et les vecteur de densités de chaque élément contenu dans la base d'apprentissage respectivement.

3.1.2 Phase d'apprentissage

learningclassifier2(rectangleslearning, learningimag, m, n) : vectordensitylearning

3.1.3 Phase de décision

decisionclassifier2(rectangles, image, vectordensitylearning, m, n, k) : pbelonging

3.2 Analyse et conclusion

3.2.1 Résultats obtenus en fonction des paramètres m et n

3.2.2 Influence du paramètre k