

TS225

Compte rendu - Projet en Traitement des Images

Maxime PETERLIN - `maxime.peterlin@enseirb-matmeca.fr`

Gabriel VERMEULEN - `gabriel@vermeulen.email`

ENSEIRB-MATMECA, Bordeaux

1^{er} janvier 2014

Contents

1	Introduction	2
2	Démarche algorithmique	2
3	Implémentation	4
4	Résultats	4
5	Conclusion	4

1 Introduction

Les codes barres permettent de représenter facilement une information alphanumérique sous forme d'image. On les retrouve au quotidien sous de multiples formes. L'objectif de ce projet sera l'étude et le décodage de codes-barres unidimensionnels suivants la norme **EAN 13** nous permettant ainsi de nous familiariser avec les techniques fondamentales du traitement des images.

Premièrement, nous expliquerons notre démarche, ainsi que nos choix algorithmiques. Puis, nous passerons à l'implémentation. Enfin, nous étudierons les résultats obtenus à l'issue de ce projet.

2 Démarche algorithmique

Cette première section traite des algorithmes utilisés.

- **Détermination de la région d'intérêt**

La première étape pour arriver au décodage des codes-barres a été la détection de leur zone d'intérêt. Afin de simplifier le traitement et la lecture du code-barres, on commence par passer l'image en niveau de gris en moyennant les composantes R, G et B de chaque pixel.

Le principe de détection repose sur deux actions au préalable de l'utilisateur. En effet, il devra déterminer manuellement les bords droits et gauche du code-barres. Le programme se chargera ensuite de déterminer l'extension horizontale du code-barres.

On appelle $A(x_1, y_1)$ et $B(x_2, y_2)$ les deux points déterminés par l'utilisateur. Grâce à ces derniers, on peut déterminer la région \mathcal{R}_0 dont les coins supérieur gauche et inférieur droit ont pour coordonnées respectives (x_{min}, y_{min}) et (x_{max}, y_{max}) . L'objectif, à présent, est d'étendre cette région verticalement, pour arriver à la région d'intérêt \mathcal{R}_T .

Pour ce faire, on part de la ligne de numéro y_{min} et on somme toutes les composantes de cette dernière ayant des abscisses comprises entre x_{min} et x_{max} . On va ensuite décrémenter la valeur du numéro de ligne y qui était initialement à y_{min} et on réitère l'opération précédente sur la ligne actuelle. On compare alors les résultats obtenus et suivant la différence entre les lignes on considère ou non que l'on est sorti de la zone d'intérêt. On obtient finalement notre nouveau y_{min} .

Le même raisonnement peut s'appliquer pour la recherche de la nouvelle valeur de y_{max} .

Cette méthode est résumée par la formule mathématique suivante, où I est la matrice représentant l'image du code-barres et ϵ un paramètre de tolérance.

$$1 - \epsilon < \left| \frac{\sum_{x=x_{min}}^{x_{max}} I(x, y)}{\sum_{x=x_{min}}^{x_{max}} I(x, y_{min})} \right| < 1 + \epsilon$$

- **Estimation de la signature**

La signature du code-barres est une représentation monodimensionnelle des informations qu'il porte. En effet, en théorie, il suffit d'avoir une seule ligne de pixels du code-barres pour pouvoir le décoder. Cependant, nous allons nous servir de cette redondance d'information pour pallier aux différentes erreurs que le code-barres pourrait comporter (rayures, flou, etc...).

On commence par projeter l'image sur l'axe horizontale. Cela se traduit par un moyennage des valeurs de chaque colonnes comprises entre x_{min} et x_{max} .

Ensuite, on va binariser l'image pour qu'elle ne soit formée plus que par des pixels blancs ou noirs. Nous allons devoir trouver un seuil délimitant les valeurs qui nous intéressent et, pour ce faire, nous appliquerons l'algorithme de Otsu qui repose sur le calcul d'un histogramme h sur N . N sera le nombre de valeurs différentes présent par les pixels de l'image.

On commence par calculer $w : k \mapsto w(k)$ et $\mu : k \mapsto \mu(k)$ avec $k \in \llbracket 0, N-1 \rrbracket$.

Notre seuil s est alors donné par $s = \max_{k \in \llbracket 0, N-1 \rrbracket} w(k)[\mu(N-1) - \mu(k)]^2 + (1 - w(k))\mu(k)^2$

On peut, à présent, passer à l'étape de décision pour binariser l'image : 0 correspond à un pixel noir et 1 à un pixel blanc.

La prochaine étape est la suppression des bits inutiles sur les bords de l'image venant de la région d'intérêt sélectionnée approximativement par l'utilisateur. Maintenant que l'image a été binarisée, cette étape est simple, car il suffit de supprimer les pixels blancs les plus au bord de la signature s'ils existent.

- **Identification des chiffres**

Nous savons qu'un code-barres est composé de 95 éléments de taille unitaire (i.e. les lignes le représentant). La première étape va être la séparation de la ligne obtenue précédemment en ces 95 éléments. De plus, il va falloir identifier la couleur de chaque élément, car ces derniers sont composés de plusieurs pixels et il se peut que leur couleur ne soit pas uniforme, d'où la nécessité de prendre une décision sur cette dernière.

Notre approche est relativement directe dans le sens où la couleur d'un élément donné sera celle des pixels de cette couleur qui sont majoritaires en nombre, ce qui nous permet de nous convertir un ensemble de pixels à un seul bit pour la description d'une barre.

On arrive alors à ramener la signature du code-barres obtenue précédemment en un vecteur de 95 bits.

L'étape suivante est la reconnaissance des chiffres. La signature d'un chiffre est donné par la concaténation de 7 éléments unitaires du code-barres. Dans le vecteur obtenu précédemment, les chiffres sont compris entre les indices 4 et 45, ainsi qu'entre les indices 51 et 92.

Pour identifier les chiffres, nous utiliserons une matrice de 30 lignes et 7 colonnes comportant les signatures théoriques des chiffres. Nous comparons alors la signature observée à la matrice des signatures théoriques pour obtenir la valeur du chiffre encodé.

La critère utilisé est le suivant.

$$c(s_{th}, s_p) = \frac{\langle s_{th} - \overline{s_{th}}, s_p - \overline{s_p} \rangle}{\|s_{th} - \overline{s_{th}}\| \cdot \|s_p - \overline{s_p}\|}$$

La signature retenue sera celle maximisant ce critère.

- **Cas des codes-barres *orientés***

Les codes-barres étudiés n'auront pas toujours leurs lignes parallèles aux bords de l'image. Il va donc

falloir trouver un angle pour effectuer une rotation sur l'image afin qu'elle devienne *droite*.

Pour se faire, il faudra détecter les lignes grâce à la transformée de Hough, puis trouver leur orientation afin de trouver un angle de rotation.

3 Implémentation

Maintenant que nous avons expliqué les différents algorithmes que nous utiliserons pour décoder des codes-barres, nous allons passer à l'implémentation de ces derniers à l'aide de Matlab.

- Détermination de la région d'intérêt
- Estimation de la signature
- Identification des chiffres
- Cas des codes-barres *orientés*

4 Résultats

5 Conclusion