Traitement d'images - Rapport de projet

Félix Abecassis - Julien Marquegnies

Résumé: Ce rapport présente le travail effectué dans la cadre du cours de traitement d'images dispensé à l'EPITA. L'objectif du projet a été le développement d'un algorithme permettant la localisation, l'extraction puis la lecture de codes barre dans des photos naturelles.

Auteurs : Félix Abecassis - felix.abecassis@epita.fr
Julien Marquegnies - julien.marquegnies@epita.fr
Date : Dimanche, 15 janvier 2012

Table des matières

Table des figures

Chapitre 1

Introduction

Chapitre 2

Localisation des codes barre

L'étape de localisation des codes barre implique deux phases principales. La première consiste à mettre en avant les éléments qui sont potentiellement des codes barre en repérant les zones à fort contraste. La seconde étape a pour objectif d'étudier les différents éléments récupérés lors de la première phase afin de déterminer par l'intermédiaire de critères géométriques (proximité, parallélisme, ...) les différents éléments qui sont suceptibles d'être des codes barre.

Ci-après se trouve la chaîne de traitements pour la localisation et l'extraction des codes barre dans l'image.

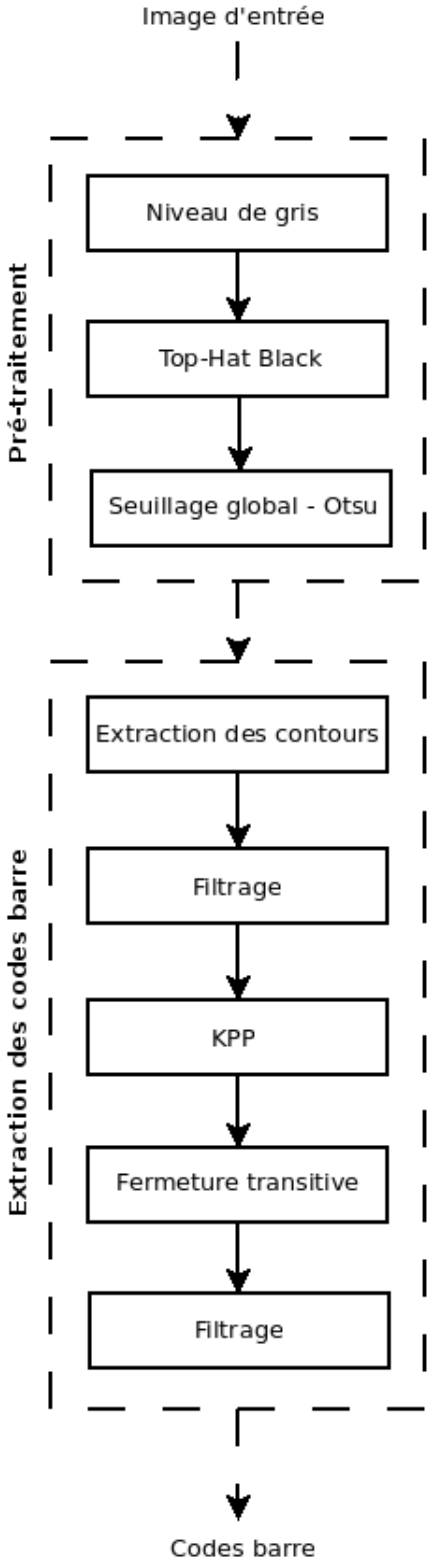

Figure 2.1 – Chaîne de traitements.

2.1. Pré-traitement

Afin de préparer l'étape de localisation des codes barre, l'image couleur d'entrée est dans un premier temps convertie en image en niveau de gris. Un réhaussement des contrastes peut également être appliqué afin de renforcer la distinction entre les objets clairs et foncés.

Les codes barre possèdent diverses caractéristiques qui leur permettent d'être différenciés des autres éléments présents dans l'image. L'une d'entre elles est la présence d'un fort contraste entre les barres composant le code et le fond sur lequel il est imprimé. Dès lors, afin de mettre en évidence ces zones de fort contraste, un filtrage morphologique de type Top-Hat Black est appliqué sur l'image en niveau de gris. Le Top-Hat Black est défini comme étant la fermeture de l'image à laquelle est soustraite l'image originale. Etant donné que la fermeture de l'image tend à étendre les zones blanches de l'image sans pour autant altérer de façon significative ces zones, il résulte de la soustraction de l'image originale les zones à l'origine noires entourées de blanc, ce qui met en avant les codes barre. L'application du Top-Hat Black a été réalisée avec un élément structurant carré de taille 25×25, ce qui est suffisant pour bien faire ressortir les codes barre.

Un seuillage global de l'image avec l'algorithme d'Otsu est finalement appliqué afin de binariser l'image.

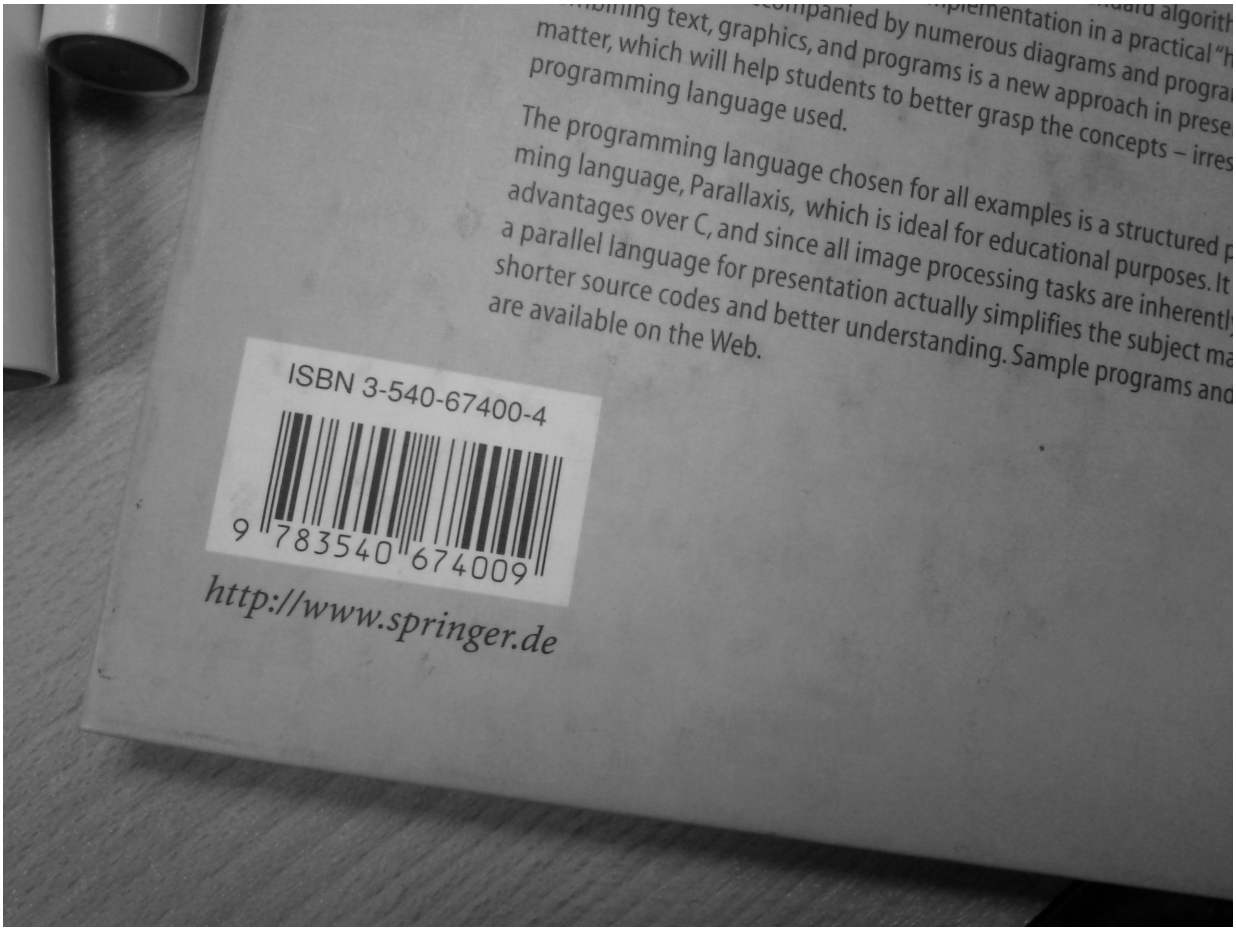

FIGURE 2.2 – Image d'entrée convertie en niveau de gris.

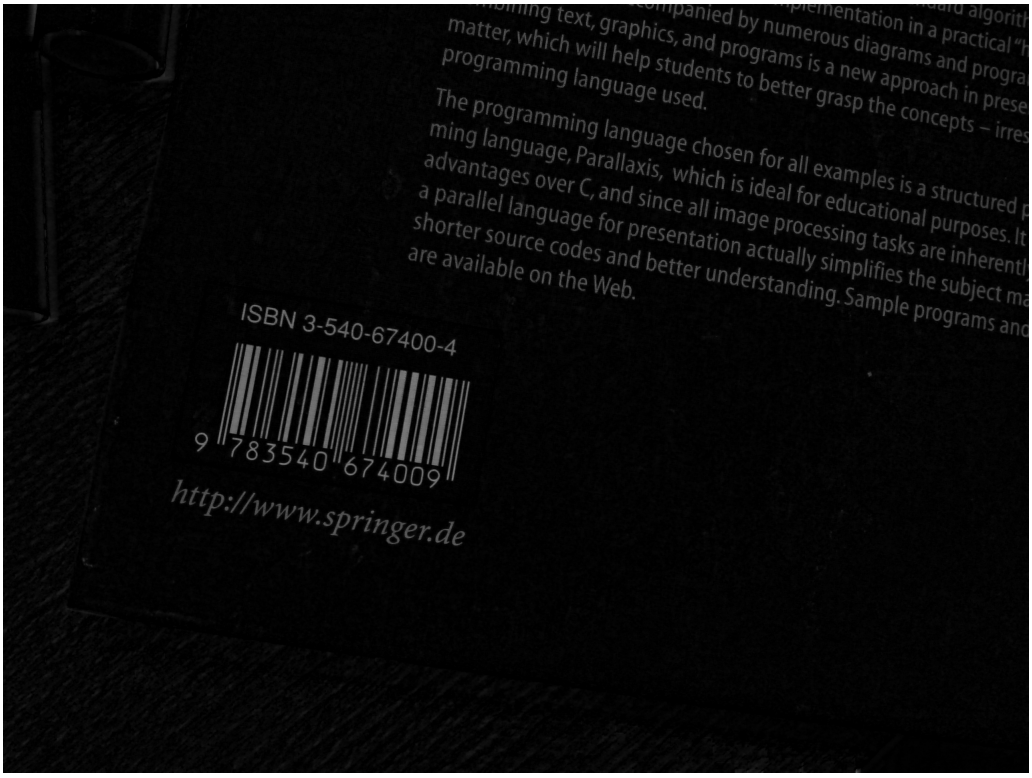

Figure 2.3 – Image résultante après l'application d'un Top-Hat Black.

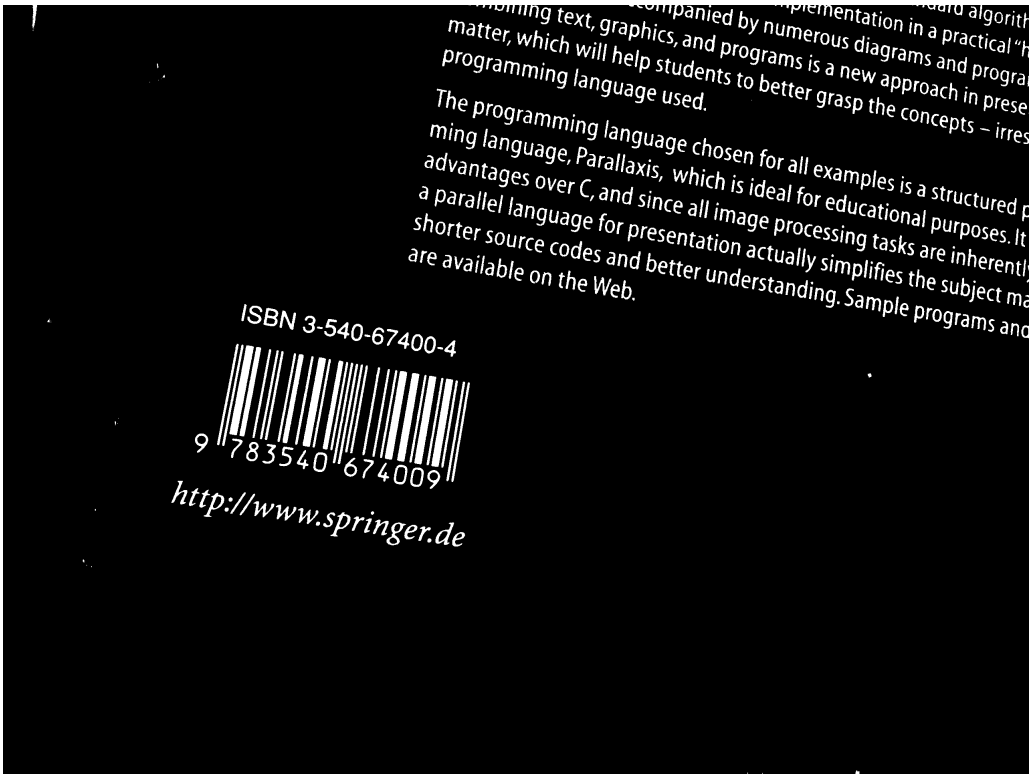

Figure 2.4 – Image résultante après la binarisation.

2.2. Extraction des codes barre

La localisation et l'extraction des codes barre à partir de l'image pré-traitée se déroule en 5 étapes.

Dans un premier temps les contours des différentes composantes présentes dans l'image binaire sont extraits. Ceci permet ainsi de déduire la boîte englobante orientée de chacune des composantes. Un premier filtrage sur les composantes peut donc être effectué. En effet, les barres composant un code barres sont toutes allongées, i.e une de leur dimensions (largeur ou longueur) est bien plus grande que la seconde. Ainsi le premier filtrage effectué tient compte des dimensions, largeur et longueur, de chacune des boîtes englobantes. Ainsi, pour qu'un composante soit gardée pour le reste du traitement, les deux conditions suivantes doivent être satisfaites :

- $\max(largeur, longueur) \geq 20$
- $\dfrac{\max(largeur, longueur)}{\min(largeur, longueur)}$

A l'issue de ce premier filtrage, un grand nombre de composantes ont pu être retirées, considérées comme ne pouvant être une partie d'un code barres.

FIGURE 2.5 – Boîtes englobantes restantes à l'issue du premier filtrage.

Dès lors que le premier filtrage a été appliqué, un algorithme de K-Plus-Proches Voisins est appliqué sur la composantes restantes. Les barres d'un code barres étant particulièrement rapprochée, l'application de cet algorithme permet de trouver, pour chaque barre, un voisin à sa gauche et un à sa droite. Dans notre algorithme nous avons fixé $K = 5$, ce qui permet de récupérer suffisamment d'information sans trop s'éloigner de la zone initiale de recherche.

Une fermeture est dès lors appliquée à l'aide des voisins récupérés à l'étape précédente afin de former les codes barres. Un certain nombre de critères ont été définis afin que deux composantes dans un même voisinage puissent être liées entre elles.

Pour chacune des composantes de l'image, les moments d'inertie sur les axes x et y sont calculés. Pour que deux composantes soit liées entre elles il est donc nécessaire dans un premier temps que l'intensité de leurs moments d'inertie sur chaque axes soient du même ordre. Si la composante $c1$ tend à être allongée horizontalement (moment d'inertie important selon l'axe x), il doit en être de même pour la composante $c2$.

Parmi les critères utilisés se trouvent également des critères géométriques qui nécessitent la mise en place d'un segment virtuel dans la boîte englobante, qui correspond à l'axe principal de celle-ci, comme illustré dans la Figure 2.6.

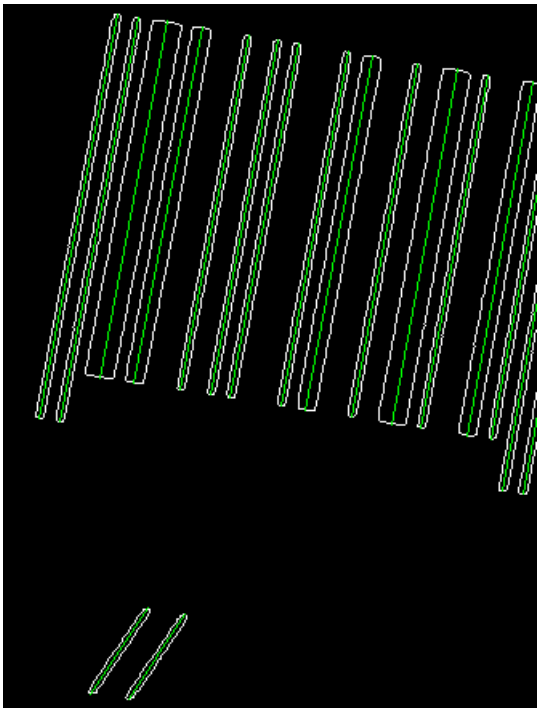

FIGURE 2.6 – Axes principaux (en vert) des différentes boîtes englobantes.

Dès lors, il est possible de calculer l'angle formé par les axes des composantes $c1$ et $c2$. Les deux composantes peuvent alors être liées si ce dernier est inférieur à $\frac{\pi}{8}$.

De plus, une projection des deux points limitants le segment de l'axe principale de la composante $c2$ sont projetés orthogonalement sur l'axe principal de la composante $c1$. Les deux composantes ne peuvent alors être liées que si il y a chevauchement de l'axe principal de $c1$ et du projeté de $c2$.

Enfin, la distance entre $c1$ et $c2$ est calculée avant la fusion des deux composantes. Si cette distance est supérieure à 3 fois l'espacement moyen entre les composantes du code barres en

formation, la fusion entre $c1$ et $c2$ est rejetée.

L'étape de fermeture terminée, un certain nombre de codes barres ont été formés à partir de l'image binaire. Cependant, un certain nombre dentre eux sont des faux positifs. Pour exemple, il est impossible qu'un code barres ne soit constitué que de 4 composantes, soit 4 barres. C'est pourquoi, un filtrage final est appliqué afin de supprimer au maximum le nombre de faux positifs trouvés avant la phase de lecture des codes qui, elle, certifiera ou non la validité des codes barres extraits par l'algorithme de segmentation.

Parmi tous les codes barre formés, seuls ceux possédant plus de 2 composantes et donc l'espacement moyen des composantes est inférieur à 8 fois la taille moyenne des composantes du code barres sont gardés. On définit la taille d'une composante comme étant le minimum entre sa largeur et sa longueur.

Enfin, une tentative de fusion de codes barres est également réalisée. En effet, il peut arriver que certains codes barres se retrouvent coupés à l'issue de l'étape de segmentation. Une recherche d'intersection entre boîtes englobantes est effectuée et une fusion est dès lors envisagée. Pour que la fusion puisse être réalisée il est nécessaire que l'orientation des deux codes barre soit la même ainsi que celle des éléments qui les composent.

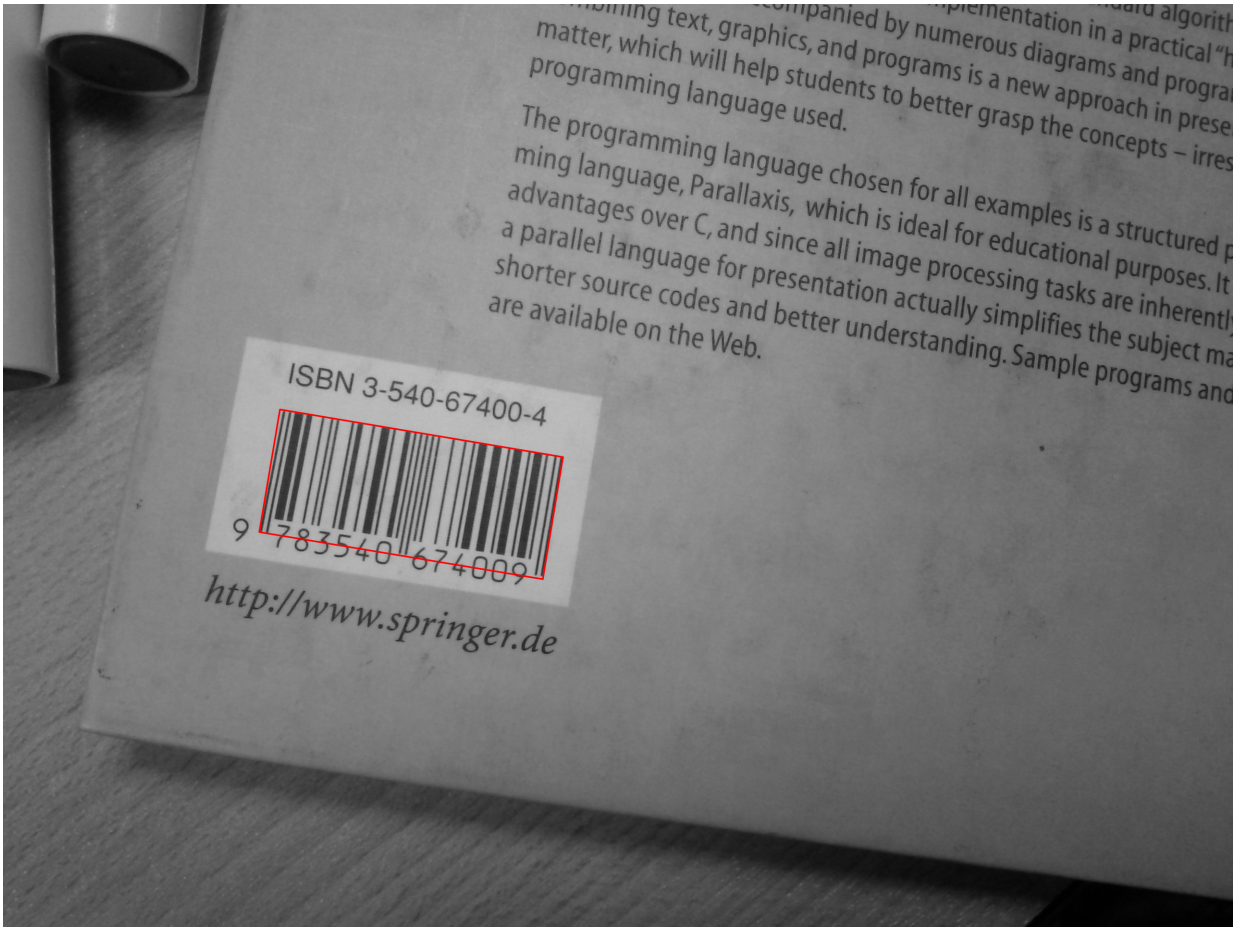

FIGURE 2.7 – Code barres trouvé à l'issue de l'étape de segmentation (non encore validé).

Chapitre 3

Fonctionnement des codes barres

3.1. Code-barres EAN-13

Dans le cadre de ce projet, nous nous sommes limités à la gestion des codes barres de type EAN, et plus particulièrement EAN-13 puisque ce format est quasiment omniprésent sur tous les produits de grande consommation en France (le type d'encodage utilisé peut varier selon le pays). EAN-13 dérive directement du système d'encodage UPC (Universal Product Code), qui fût le premier type de code barres existant, inventé par George Laurer, ancien ingénieur chez IBM. Comme son nom l'indique, un code EAN-13 est composé de 13 chiffres, soit seulement un de plus que le système original UPC.

Les codes barres sont faits pour être lus par des lecteurs lasers, le système d'encodage a donc été concu avec comme objectif la simplicité et la robustesse. Les chiffres écrits en dessous sont seulement présents pour permettre une saisie humaine en cas de problème (barres fortement endommagées, défaillance de la douchette laser, etc.). Lire des chiffres est d'ailleurs une tâche impossible pour un lecteur laser et difficile pour un appareil photo ou une caméra. Un classifieur statistique est nécessaire et donc les temps de calculs sont généralement beaucoup plus importants. Au contraire, un décodeur se basant sur les barres peut atteindre des performances largement temps réel (jusqu'à 90 codes par seconde), il est aussi important de noter que certains code barres ne présentent pas de chiffres en dessous, ceux si sont optionnels.

3.2. Fonctionnement de l'EAN-13

Sur chaque code barres nous pouvons tout d'abord observer qu'il y a toujours une large zone blanche à gauche et à droite du code barre, ceci peut s'observer très bien sur la dernière image de la partie détection (la couverture de livre d'une couleur plus sombre que le code barres).

Cette zone se nomme la "quiet zone" (zone de silence). La taille de cette zone doit être au moins égale à 10 fois la largeur de la plus petite barre du code. Cette zone existe afin de faciliter la détection, mais elle n'a aucun interêt pour le décodage.

Nous pouvons ensuite observer la triple occurrence d'un motif semblable à gauche, à droite et enfin au milieu du code barres. Les barres de ce motif sont généralement plus hautes pour marquer la séparation mais ce n'est pas systématique. Ce motif est appelé **motif de garde** et les trois occurrences de ce motif sont appelées S, M et E (Start, Middle et End). S et E sont constitués du motif noir-blanc-noir (3 barres) et M est constitué du motif blanc-noir-blanc-noir-blanc (5 barres).

FIGURE 3.1 – Motif de garde sur un code barres EAN-13.

Dans un code barres EAN-13, les chiffres sont separés en trois groupes : le premier chiffre, les 6 chiffres de gauche (entre S et M) et les 6 chiffres de droite (entre M et E).

Chaque chiffre est encodé en binaire selon une table parmi 3 au total : L, G et R. La notion de bit pour un code barres correspond à la valeur d'une barre de taille élémentaire. La largeur d'une barre élémentaire est donnée par la lecture des gardes : les barres noires et blanches sont forcément de taille élémentaire. Les motifs de garde servent donc à calibrer le lecteur de codes barres. Une barre élémentaire noire correspond à un bit de valeur 1 et une barre élémentaire blanche correspond à un bit de valeur 0. Chaque chiffre est encodé sur 7 bits, S et E sur 3 bits et M sur 5 bits.

Les chiffres de gauche peuvent être encodés par 2 tables différentes, le schéma d'alternance entre ces tables est determiné par la valeur du premier chiffre du code barres comme montré dans la table suivante.

Premier chiffre	**Encodage chiffres de gauche**
0	LLLLLL
1	LLGLGG
2	LLGGLG
3	LLGGGL
4	LGLLGG
5	LGGLLG
6	LGGGLL
7	LGLGLG
8	LGLGGL
9	LLGLGL

Pour les chiffres de droite, l'encodage de type R est toujours utilisé. L'encodage L est le complément bit à bit de l'encodage R.

Les tables d'encodage pour L, G et R sont presentées dans le tableau suivant :

Chiffre	**L**	**G**	**R**
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Les encodages G et R étant symétrique, lire un code de type R à l'envers donne un code de type L et inversement. Cependant un code de type L lu à l'envers est invalide, ceci permet de déterminer si un code est en train d'être lu à l'envers.

Le dernier chiffre est une clé de contrôle calculée à partir des autres chiffres (un checksum). Cette clé est calculée en calculant la somme ponderée des autres chiffres puis en prenant le complément à 10 du chiffre des unités. Les poids utilisés dans la somme sont alternativement 1 et 3.

Le principe de fonctionnement de l'EAN-13 est resumé sur ce graphique provenant de Wikimedia Commons :

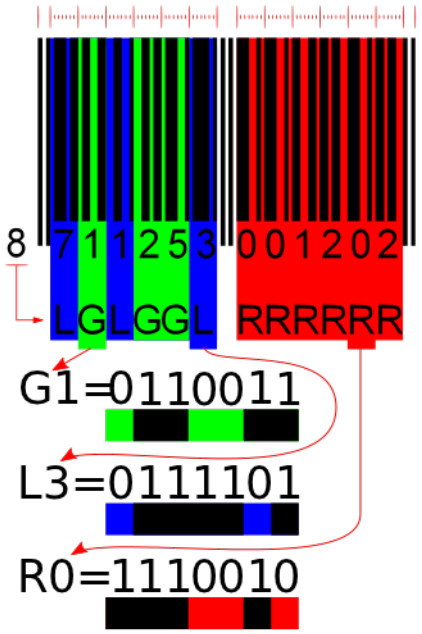

FIGURE 3.2 – Resumé du fonctionnement de l'EAN-13.

Chapitre 4

Implémentation du décodage

4.1. Principe général

Les codes barres EAN-13 sont des codes à une seule dimension, la hauteur importante du code barre n'est présente que pour servir de mécanisme de redondance contre le bruit qui peut survenir dans l'image. Le décodage du code barres peut donc se réaliser en ne prenant en compte qu'une seule ligne de l'image. Il faut donc ensuite parcourir cette ligne dans un sens et trouver successivement quels sont les chiffres qui correspondent aux alternances de blanc et de noir rencontrées.

FIGURE 4.1 – Ligne de scan d'un code barres

Comme nous l'avons vu, les codes barres possédent un double mécanisme de redondance : le codage du chiffre gauche dépend du premier chiffre ce qui diminue les risques d'erreurs, de plus le dernier chiffre est un checksum calculé à partir des autres chiffres. De plus décodage est un processus rapide qui ne demande qu'un parcours horizontal de l'image. Pour ces raisons, la segmentation peut très bien extraire des images qui ne sont en fait pas des codes barres puisque ces images seront rejetées lors du décodage : le code ne sera pas valide.

4.2. Extraction du code barres

Il est d'abord nécessaire d'extraire le code barres depuis l'image originale. Grâce à l'étape précédente nous avons une ou plusieurs boîtes englobantes par image, l'orientation des codes barres est indéterminée : ils peuvent être horizontaux, verticaux ou orientés à 45 degrés. La première étape du décodage est donc d'isoler chaque boite englobante et d'effectuer la rotation nécéssaire pour obtenir des code barres dans le sens horizontal (le code barres peut toujours être à l'envers par contre, ce cas est geré plus tard dans notre algorithme).

4.3. Prétraitement

Pour arriver à une bonne résistance au bruit et aux variations d'illumination, il est indispensable d'utiliser une binarisation efficace, sinon des barres vont tout simplement être perdues lors de cette phase. En effet lorsque le code barres est endommagé la différence d'intensité lumineuse entre une barre blanche et une barre noire peut être faible. La luminosité peut également grandement varier au sein d'un même code barre, nous avons donc besoin d'utiliser un algorithme de binarisation robuste.

Nous avons expérimenté plusieurs algorithmes différents. Certains articles proposaient de calculer tout d'abord une moyenne élaguée de chaque pixel sur une hauteur de 10. Cependant cette méthode ne fonctionne pas du tout lorsque les barres présentent une forte distorsion, c'est par exemple le cas lorsque que le code se trouve sur une surface facilement maléable (par exemple une feuille de papier ou encore un sac plastique). Nous avons ensuite essayé un seuillage global selon la ligne de scan : en prenant la valeur moyenne ou médiane de cette ligne. Nous avons également testé l'algorithme d'Otsu sur toute l'image du code barres. Ces deux méthodes n'ont pas donné des résultats satisfaisants.

Finalement nous nous sommes orientés vers des techniques de seuillage local (ou seuillage adaptatif). Ces méthodes calculent une valeur de seuil selon le voisinage proche du pixel courant. La taille de ce voisinage et la méthode d'accumulation utilisée peut varier (par exemple moyenne, médiane, moyenne ponderée). Après experiméntation, nous en avons conclu que la meilleure technique est d'utiliser comme seuil local la moyenne ponderée par un noyau gaussien. Ainsi les points les plus proches sont favorisés pour le calcul de la valeur du seuil. La taille de la fenêtre (glissante) pour le calcul du seuil est environ la largeur d'un chiffre du code (soit la largeur de l'image divisée par 12).

4.4. Lecture des chiffres

Maintenant que l'image est binarisée nous pouvons lire les chiffres. Chaque moitiée du code barre est seuillé et lue indépendamment.

Initialement, les gardes étaient utilisées pour trouver la largeur élémentaire des barres blanches et des barres noires, c'est à dire que les gardes étaient utilisées pour leur fonction initiale de calibrage. Nous multiplions ensuite ces valeurs dans les 3 tables d'encodage pour les faire correspondre à l'échelle de notre code barres. Un chiffre est toujours encodé par l'alternance blanc-noir-blanc-noir, il suffit donc de lire la largeur de chacune de ces 4 barres en avancant dans la ligne de scan. Nous calculons ensuite la distance entre le code trouvé et tous les codes de référence. Nous prenons ensuite le chiffre dont le code avait la plus faible distance.

Cependant, cette méthode ne s'est pas revelée être la plus efficace. Au lieu de faire correspondre les tables d'encodage aux observations, nous faisons l'inverse. Les tables ne sont pas modifiées mais à la place la largeur du code est normalisé pour obtenir une norme L1 (la somme) de taille 7.

4.5. Vérification

Lors du décodage de la partie gauche nous ne modifions pas notre algorithme en fonction de la valeur du premier chiffre puisque la valeur de celui ci pourrait avoir été mal lue. Ce mécanisme sert à la place de contrôle : nous vérifions si le premier chiffre correspond bien à l'encodage trouvé pour les chiffres suivants. Nous calculons ensuite le checksum et le comparons par rapport au dernier chiffre. Si le premier ou le dernier chiffre est invalide, le code est erroné, nous essayons quand même de "sauver" ce code en remplacant le chiffre qui avait la plus petite différence entre le premier et le deuxième résultat.

Lorsqu'une ligne donne un code invalide, nous essayons sur une autre ligne plus bas, jusqu'à que 6 lignes soient testées avant de rejeter le code.

Dans notre programme, les codes barres valides sont entourés d'un rectangle vert et le code est inscrit à l'intérieur du rectangle. Les détections rejetées sont elles entourées d'un rectangle rouge.

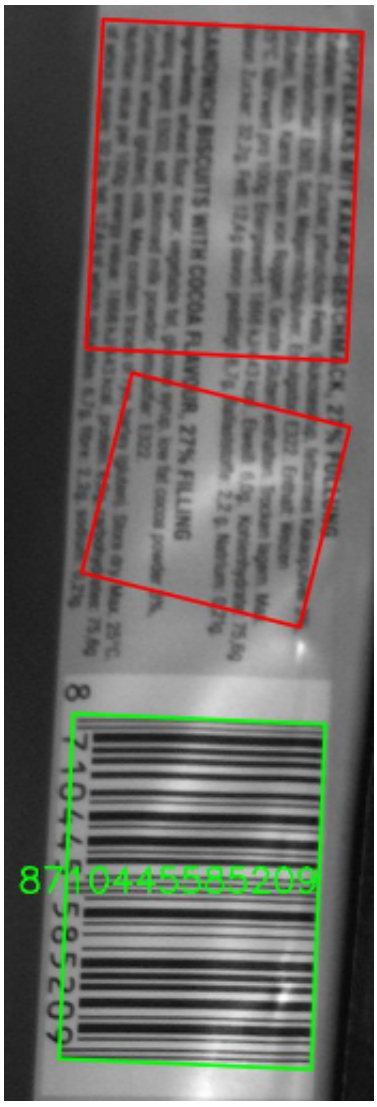

FIGURE 4.2 – Résultat du décodage.