Traitement d'Images

Rapport de Projet

Caroline Beyne – Perrine Brunel
25/09/2013

Ce rapport témoigne du projet réalisé pour le cours de Traitement d'Images, de M. Fabrizio, dispensé à l'EPITA. Le sujet choisi pour le développement du projet est la détection et lecture de code barres dans une image.

SOMMAIRE

TABLE DES FIGURES

1. PRESENTATION

Pour le projet de Traitement d'Images, nous avions à implémenter un sujet parmi plusieurs possibles. Nous avons choisi celui de lecture de code barre dans une image, car c'est celui qui nous paraissait le plus « concret ». En effet, il existe par exemple des applications pour Smartphones quotidiennes qui permettent de faire ses courses en décodant le code barre de ses emplettes. Hors les téléphones n'ont pas de tête de lecture laser comme les caisses des supermarchés mais seulement des caméras et le traitement doit se faire à même l'image. Ainsi pour mieux comprendre le procédé, nous avons choisi ce sujet.

Nous avons réalisé le projet dans le langage C++, en utilisant la bibliothèque OpenCV, très performante en ce qui concerne le traitement d'images, ce qui nous a permis de la découvrir, pour notre culture personnelle et pour des besoins sur d'autres projets.

Ce rapport se découpe en 3 parties :

1. **La préparation de l'image de base pour l'extraction des codes barre**
2. **L'extraction des fameux codes barre**
3. **Le décodage desdits codes barre pour vérifier qu'ils soient bien valides**

   Le résultat sera alors affiché sur l'image d'origine suivant la validité.

2. PREPARATION

La préparation de l'image est une étape tout aussi importante que les 2 autres, car sans un bon prétraitement de l'image, aucun code barre ne peut être décodé.

Tout d'abord, un code barre est caractéristique par ses couleurs (noire et blanche - bien qu'il existe des originaux qui remplacent le noir par de la couleur) très contrastées. Il s'agit donc, dans un premier temps, de faire ressortir ces zones de fort contraste pour pouvoir envisager de les trier par la suite.

Une première étape consiste à passer l'image en niveau de gris, car les couleurs ne nous intéressent pas et de manière générale le traitement d'images s'effectue souvent en niveau de gris.

Figure 1 - Image en niveau de gris

Figure 2 - Image en niveau de gris

Comme dit précédemment, les codes barre sont caractéristiques par leur zone de forts contrastes, que nous voulons faire ressortir. De plus, sur l'image de départ, les barres sont (normalement) noires, or nous préférons les avoir blanches, car dans la logique arbitraire du traitement d'images : le noir représente le fond et le blanc les objets. Ainsi, nous appliquons un Black Top Hat qui fait ressortir les zones sombres et nos barres apparaissent alors blanchies. En effet, un Top Hat représente la fermeture de l'image à laquelle on soustrait l'image originale, faisant ressortir notre code barre. Dans notre programme, après plusieurs essais, suivant les tailles d'images, l'élément structurant le meilleur se situait entre 20x20 et 30x30, que nous avons finalement mis à 25x25 pour être au mieux.

Figure 3 - Black Top Hat

Une binarisation est effectuée ensuite pour faciliter le traitement en n'ayant que 2 couleurs bien distinctes. Comme le résultat du Top Hat est assez sombre, après plusieurs essais, nous mettons tous les pixels au dessus de 50 en blanc.

Figure 4 - Binarisation

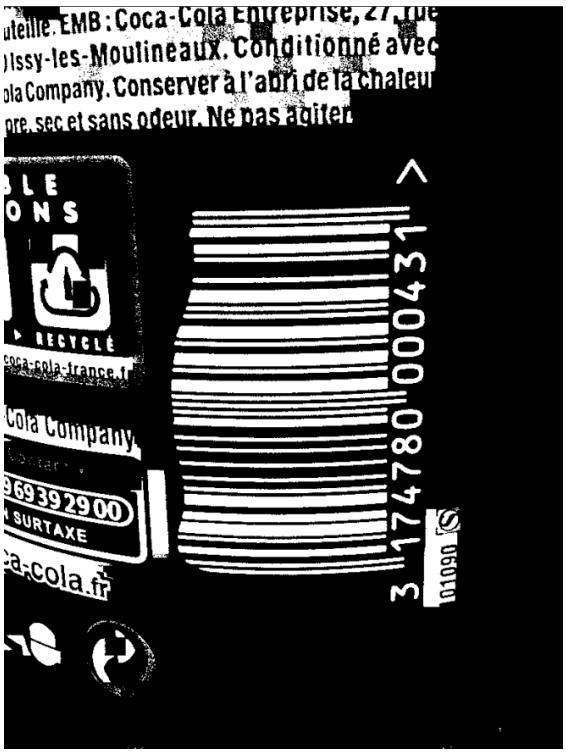


Figure 5 - Binarisation

Pour résumer, lors du prétraitement, nous effectuons les opérations suivantes :

Une fois notre image nettoyée et binarisée, nous passons à la recherche puis l'extraction des codes barre.

3. EXTRACTION

L'extraction de code barre est la partie la plus délicate du programme. En effet, il s'agit de bien sélectionner les morceaux et de les extraire correctement pour pouvoir être lus et validés.

Pour cette partie, le travail est divisé en plusieurs étapes :

- détection de boite englobante (bounding box)
- association d'axe principal à sa boite englobante
- recherche de groupes de boites suivant des critères géométriques
- extraction de chaque groupe en une image binaire horizontale

3.1. DETECTION DE BOITE ENGLOBANTE

Une boite englobante est un polygone qui englobe au pixel près la figure qu'on lui dit d'encadrer. Ainsi, pour trouver les boites englobantes de nos barres de code barres, nous effectuons d'abord la recherche des contours, ce qui nous permet de définir un premier jet de boite englobante. Ces boites sont alignées sur le repère orthonormé et donc les barres obliques ne sont pas bien englobées.

Un deuxième passage est effectué où à l'intérieur de chaque boite englobante, on cherche l'aire minimum d'une figure (et donc de notre fameuse barre oblique). Nous avons alors un RotatedRect, un rectangle orienté, dont nous extrayons les 4 points. Nous profitons de ce passage pour faire une première sélection, où les barres trop fines proches d'une ligne (si la largeur est plus de 10 fois inférieure à la longueur) et les barres minuscules (si la longueur n'atteint même pas les 20 pixels). Les rectangles orientés sélectionnés deviennent alors nos boites englobantes.

Figure 6 - Boite englobante

3.2. ASSOCIATION D'AXE PRINCIPAL

Ensuite, pour une meilleure recherche de nos barres en nous appuyant sur des critères géométriques, nous décidons de créer un axe principal pour

chacune de nos boites englobantes. Un axe principal est la médiatrice des 2 plus petits côtés de notre boite, à priori la largeur. Ainsi un axe principal représente sa boite en longueur en étant au milieu de celle-ci, tout en étant plus facile à travailler en comparaison que la boite elle-même.

Pour une boite donnée, on cherche alors la longueur la plus courte (suivant l'orientation de la barre), et l'on calcule les coordonnées des points au milieu des 2 extrémités.

Figure 7 - Boite englobante avec axe principal

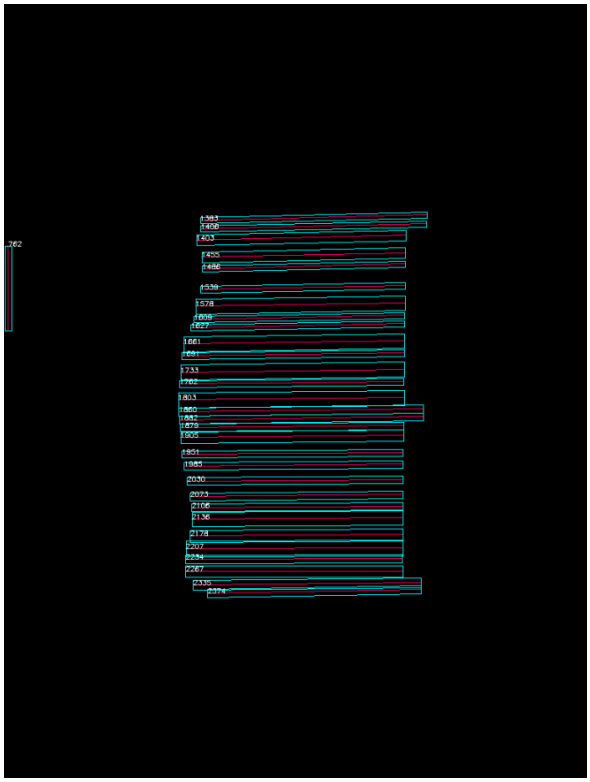


Figure 8 - Boites englobantes pas groupées

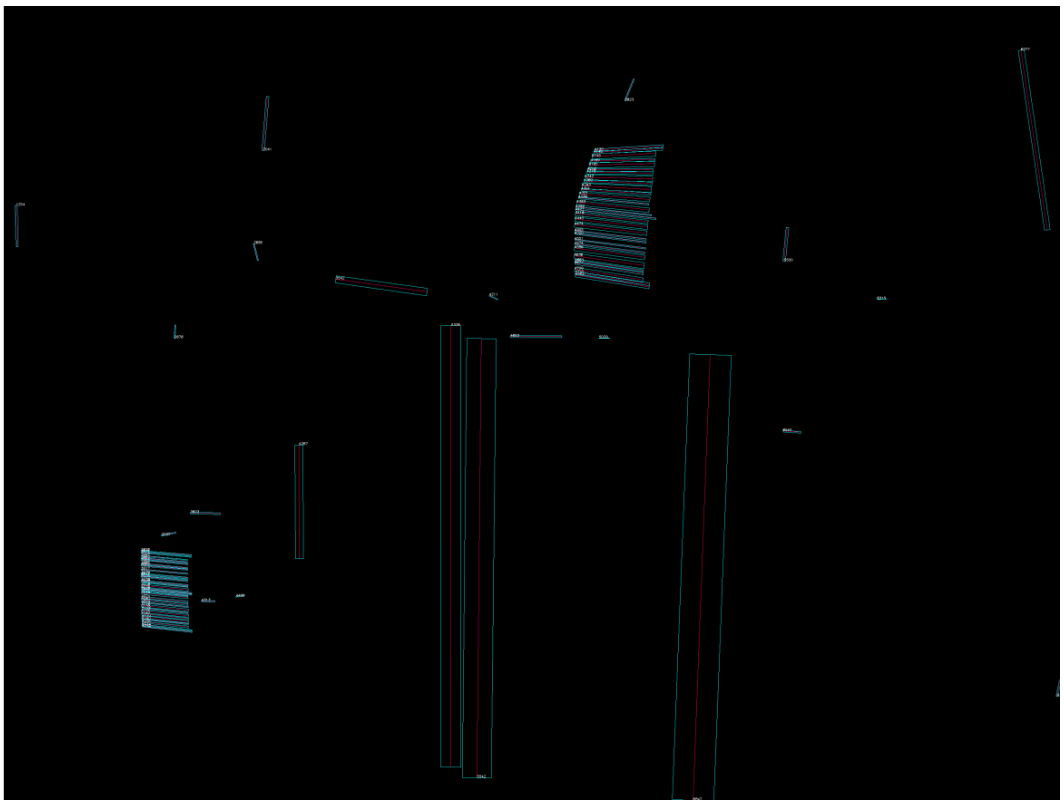


Figure 9 - Multitude de boites englobantes pas groupées

3.3. RECHERCHE DE GROUPES DE BOITES

Vient ensuite une partie délicate, la recherche des groupes de boites. Cette recherche s'effectue principalement grâce aux axes précédemment calculés, les boites englobantes sont mises à jour après ça en fonction des résultats.

Nous parcourons alors chacun des axes et nous tentons de découvrir s'il a des voisins aux mêmes caractéristiques. Ainsi pour chacun des autres axes, autres que celui sur lequel s'effectue le test, nous comparons les distances entre les points des 2 axes, deux à deux. Ainsi, si nous testons l'axe_1 avec l'axe_i, nous allons calculer la distance_1 entre le point_1 de l'axe_1 et le point_1 de l'axe_i, idem pour le point_2.

Comme les barres d'un code barre sont censées être alignées, la différence entre ces deux distances devrait être de 0. Pour palier aux éventuelles erreurs si jamais le code est un peu froisé, nous acceptons que cette différence soit <= à 20 pixels.

En plus d'être alignées, les barres sont rapprochées. Nous vérifions alors que soit la distance_1 soit la distance_2 soit <= à 50 pixels (à cause d'un éventuel froissage).

Si notre axe courant reconnaît des voisins alors il est stocké temporairement en attendant de le regrouper avec tous ses voisins. S'il est tout seul, il est le maillon faible et nous le supprimons.

Comme les barres ne sont pas forcément détectées par OpenCV, lors de la recherche de contours, dans l'ordre dans lequel notre esprit les assimile en les voyant, la fusion des groupes de voisins est assez délicate. Ainsi, pour chaque petit groupe, s'il existe au moins un axe qui appartient à deux petits groupes, nous les fusionnons dans le premier et le deuxième disparait, et la recherche reprend du début, au cas où grâce à la dernière fusion, on puisse récupérer d'autres voisins précédents.

Une fois que tous nos voisinages sont complets, on redéfinit les boites englobantes qui englobent tous les voisins, toujours avec un rectangle orienté. Nous profitons de cet ajustement pour éliminer les voisinages donc l'englobement n'a pas la forme d'un code barre, à savoir donc la largeur est plus de 6 fois inférieure à la longueur.

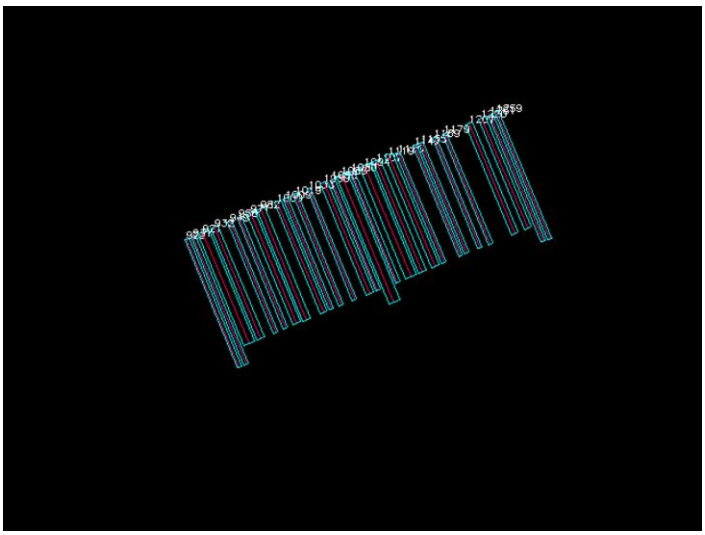


Figure 10 - Groupe de boites englobantes avec tri

3.4. EXTRACTION

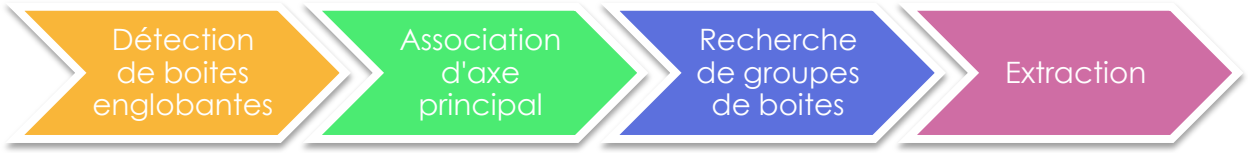
Enfin vient la partie la plus délicate : l'extraction. Après avoir défini nos boites englobantes de groupes de barres (et donc, nous espérons, de codes barre), nous allons les transformer pour les passer au décodeur. Chaque boite englobante est alors extraite de l'image binaire d'origine et remis à l'horizontale en fonction de l'angle qu'avait le rectangle orienté de la boite englobante.

Les barres presque à la verticale se retrouvent donc alignées à la verticale, il s'agit alors de les coucher. Nous effectuons une rotation de 90° dans le cas où la longueur serait à l'horizontale et pas à la verticale (et donc le grand côté de haut en bas au lieu de gauche à droite).

Enfin, l'affination de l'extraction d'une barre très orientée peut laisser du bruit en niveau de gris, nous repassons alors une binarisation de l'extrait pour pouvoir l'envoyer au décodeur.

Figure 11 - Image envoyée au décodeur

Récapitulatif des étapes du traitement :



4. DECODAGE

4.1. FONCTIONNEMENT DES CODES BARRES

Un code barre est composé de traits noirs séparés par des espaces blancs.

On considèrera ici pour plus de facilité qu'une partie noire sera appelée BARRE et qu'une partie blanche sera appelée ESPACE, bien que lors du traitement ces couleurs soient inversées.

Le code barre se sépare en plusieurs parties.

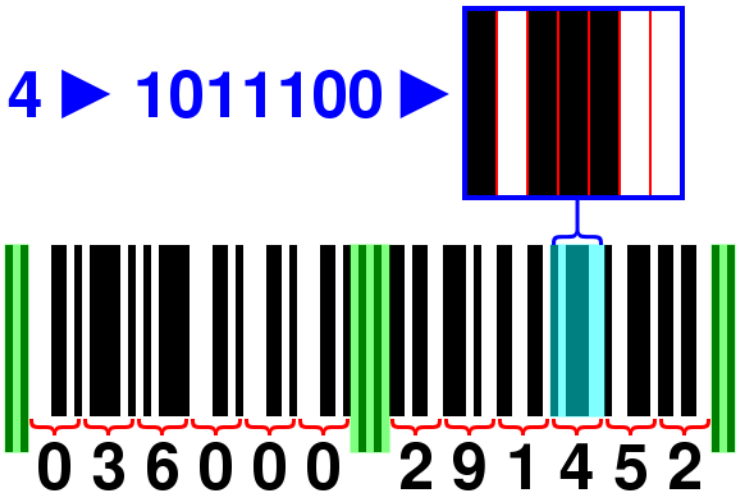
Tout d'abord, le code barre commence par un motif que l'on appelle une garde. Il y a en tout 3 gardes dans un code barre : au début, au milieu et à la fin.

Les gardes du début et de la fin sont construites ainsi : BARRE - ESPACE - BARRE.

Celle du milieu est un peu plus longue : ESPACE - BARRE - ESPACE - BARRE - ESPACE.

Entre 2 gardes, on trouve 6 chiffres de 0 à 9 qui sont codés sur 7 bits. Il y a un codage pour le côté gauche (entre la garde de début et milieu) et le codage du coté droit n'est qu'un complément à un des valeurs du coté gauche (entre la garde du milieu et celle de droite).

	Codage à gauche	Codage droit
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	1011100
5	0110001	1001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100



EXEMPLE DE CODE BARRE:

- **en vert**: les 3 gardes
- **en bleu clair**: un exemple de chiffre sur 7 bits

4.2. ALGORITHME UTILISE

Tout d'abord, nous appliquons notre traitement sur une cv::Mat d'OpenCV qui a été traitée auparavant (en noir et blanc). Sur cette matrice, nous commençons par déterminer où commence le code barre en éliminant toutes les marges éventuelles. D'abord en commençant par les marges en largeur et s'il y a besoin en hauteur.

Ensuite, on cherche la garde du début. Le but va être de vérifier qu'elle est correcte, et on va aussi se servir d'elle pour définir la taille minimum d'une BARRE en pixel. Par défaut, on prendra la taille de la première BARRE de cette garde pour taille minimale.

Une fois que nous avons cette information, nous pouvons regarder les chiffres.

Pour se faire, nous parcourons chaque bit, pixel par pixel. Pour chacun de ces pixels, nous ajoutons 1 à la valeur du bit si celui-ci est une BARRE. Ensuite pour avoir les résultats, nous comparons les valeurs totales de bits avec la moitié de la taille minimale. Cela permet d'avoir la couleur dominante du bit.

Pour finir cette recherche, il faut à la fin réaligner les bits. En effet la mesure de la taille minimale n'étant pas très précise, on peut parfois avoir des erreurs dues au bruit si on ne réaligne pas.

Il y a deux cas qui sont concernés :

- Si (le bit - le dernier pixel) est principalement (par principalement, on entend largement au dessus de la moyenne) un ESPACE et que le dernier pixel est une BARRE.
- Si (le bit - le dernier pixel) est principalement une BARRE et que le dernier pixel est un ESPACE.

Dans les deux cas, vu que le dernier pixel n'a aucun impact sur le bit, on replace le curseur juste avant ce pixel.

Une fois ceci fini, on calcule les bonnes couleurs pour les bits en fonctions de moitié de la taille minimale.

Et on compare celle-ci avec le codage des chiffres de référence qui sont déjà connus pour trouver le chiffre. Si jamais on ne reconnait pas le chiffre, on renvoie -1 et on affiche une erreur, mais on continue le traitement.

Pour finir, on détecte la garde du milieu et on fait la même chose avec les chiffres de droite en les comparant avec le codage droit et on finit avec la garde de fin.

Le décodeur renvoie un vecteur d'entier qui correspondent aux chiffres du code barre. En cas de code invalide, il s'agit d'un vecteur de -1. Le code en question est alors affiché entouré de rouge, tandis qu'un code validé apparait en vert avec le contenu de son code dans l'image résultante.

Figure 12 - Image avec rejets

Figure 13 - Image avec acceptation

5. CONCLUSION

C'est avec plaisir, bien que prises par le temps, que nous avons effectué ce projet intéressant. Notre programme n'est bien évidemment parfait et aurait nécessité plus de temps pour de meilleurs résultats. Néanmoins, au vu des conditions de rendu tardives, nous sommes assez fières du résultat, dont nous aurons beaucoup appris.