Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Filière Systèmes industriels

Orientation Infotronics

Travail de bachelor
Diplôme 2016

Non-confidentiel

Johan Dreydemy

Recherche d'algorithmes de sélection automatique d'un modèle d'alignement d'images en vision

Professeur
Pierre Roduit

Expert
Mario Bellino

Date de la remise du rapport
15.07.2016, 12 :00

Table des matières

1 Description

Une pression accrue sur les règlementations étatiques et internationales oblige les clients de Bobst à assurer la qualité d'impression dans des zones spécifiques des emballages de boîte, principalement dans les zones de posologie (p.ex. domaine pharmaceutique), mais également dans les logos de marque (p.ex. domaine du luxe). Des algorithmes spécifiques (OCR,…) sont déjà intégré dans les machines d'inspection Bobst.

Afin de faciliter le travail des opérateurs, une définition automatique de ces zones aurait une plus-value substantielle : temps de mise-en-route réduit, assurance d'une inspection de haute qualité (p.ex finesse de détection d'occlusion et d'absence de jambage).

2 Objectifs

Dans le but de faciliter le travail des opérateurs il est nécessaire d'implémenter un algorithme qui permettra la détection des zones haute fréquences sur des emballages de type carton ondulé (FLEXO), carton plat (CH), et boîte pliante (PCR) (p.ex. pharmaceutique).

Le choix de l'algorithme sera déterminé dans le but de privilégier la robustesse en garantissant une détection des zones texte semblable à plus de 80% d'une définition à la main pour un temps de processing dans l'ordre de moins d'une dizaine de seconde.

Il faudra alors dans un premier temps, définir une stratégie de définition des zones de haute fréquence pour enfin l'implémenter et la tester en utilisant une librairie OpenCV en langage C++.

Les objectifs ont été définis à la suite des discussions avec Mr. Mario Bellino et Louis-Séverin Bieri.

3 Introduction

Une zone haute fréquences est une région d'une image qui contient des informations de type texte, code-barres et logos. La détection automatique des zones hautes fréquences consiste par un algorithme à délimiter sur une image l'ensemble de ces éléments.

Les principaux efforts pour obtenir une bonne détection sont directement dépendants des performances des méthodes de pre-processing et de processing. Il s'agira alors d'analyser différentes approches pour ensuite les analyser, les comparer et proposer les méthodes qui présentent les meilleures performances.

Une fois des méthodes de pre-processing et de processing établis, il s'agira de filtrer les derniers éléments restant pour ensuite les délimiter comme étant des zones hautes fréquences.

Pour finir, les résultats finaux de l'algorithme seront comparés de façon automatique avec une segmentation effectuée à la main.

4 Théorie

Dans ce chapitre, on expliquera toutes les étapes nécessaires pour arriver à une segmentation.

4.1 Fonctionnement

Un système OCR part de l'image numérique réalisée par un scanner optique d'une page (document imprimé, feuillet dactylographié, …) et produit en sortie un fichier texte en divers formats. Les performances d'un système OCR dépendent directement de la segmentation effectuée au préalable.

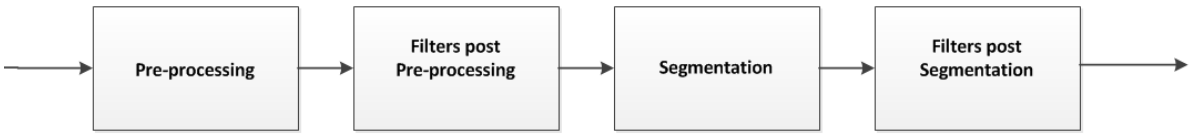

Figure 1 : Pipeline d'une segmentation

La figure 1 présente les différentes étapes nécessaires pour effectuer la segmentation d'une image. Toutes les étapes sont directement dépendantes. L'objectif de chaque étape est d'éliminer au fur et à mesure des informations qui ne sont pas considérées comme étant des zones de textes et code-barre.

4.1.1.1 Pre-processing

L'objectif du pre-processing est de facilité les étapes de processing. Pour cela des filtres permettant de calculer l'intensités de chaque zones permet de faire apparaitre les zones de textes et code-barre. Le pre-procsessing permet aussi de transformer l'image originale en image monocouleur.

4.1.1.2 Filtres pre-processing

Les méthodes de filtres pre-processing visent à éliminer les informations qui ne sont pas considérées comme des zones texte et code-barre. Ces filtres consistent à analyser la géométrie des contours détecter après pre-processing et d'éliminer les premiers éléments indésirables.

4.1.1.3 Processing

Les méthodes de processing consistent à pousser l'analyse des contours restants en analysant cette fois-ci plusieurs critères tels que : histogramme, projection, intensités.

4.1.1.4 Segmentation

Cette étape consiste à regrouper les contours restants et de détourer sur l'image originale ces candidats comme étant une zone texte ou code-barre.

4.1.1.5 Interprétation des résultats

Afin d'interpréter les résultats de chacune des méthodes qui seront présentées dans ce travail, j'ai implémenté une solution qui permet de comparer une segmentation effectuée par un opérateur aux résultats obtenue par nos algorithmes. Je définis les critères de mesures de la façon suivante :

- Rapport entre les surfaces segmenté à la main et segmenter automatiquement (text)
- Rapport entre les surfaces non texte et la surface totale de la boîte (noise)

Le fait d'analyser les résultats de cette façon nous permet d'avoir une constance et une rigueur dans leur interprétation.

5 Etats de l'art

Dans ce chapitre on présentera les méthodes existantes et les algorithmes existants qui effectuent de la reconnaissance optique et de la segmentation de textes.

5.1 Deep Features for Text Spotting[1]

Les méthodes de machine Learning sont énormément utilisées pour la reconnaissance de motif ; par exemple la reconnaissance optique de caractères (OCR) ainsi que dans la segmentation des zones de texte.

Ces méthodes s'appuient sur une structure d'un réseau de neurones qui prend comme paramètre d'entrée les caractéristiques d'une image, pour obtenir en sorti le texte présent ou non sur l'image d'entrée.

Ces méthodes fonctionnent en deux étapes distinctes à savoir :

- La mémorisation
- La généralisation

Une fois le réseau de neurones établi avec ses différentes entrées (abstraction de l'image) il s'agit de présenter au réseau des éléments dont les résultats sont déterminés à l'avance, on appelle cela la phase de mémorisation.

Une fois la mémorisation effectuée, on présente au système d'autre image dont le résultat n'est pas déterminé et on analyse les résultats obtenus à la sortie du réseau, on appelle cela la phase de généralisation.

Le désavantage d'une méthode basée sur un réseau de neurones réside sur le fait qu'il est très compliqué d'expliquer les étapes effectuées à l'intérieur du réseau.

Pour la phase de mémorisation il faut également disposer d'un très grand nombre d'images qui doivent être classifiées par un humain, tout en s'assurant de ne pas générer un sur-apprentissage.

L'avantage de ces méthodes réside principalement dans leurs efficacités.

[1] Deep Learning for Text Spotting

D.Phil Thesis
Robotics Research Group
Department of Engineering Science
University of Oxford

5.2 Méthode « sliding-windows detection »

Une « sliding -windows » est une région rectangulaire que l'on détermine par sa hauteur et largeur parcourant l'ensemble d'une image.

Pour chaque région de cette fenêtre, on applique une classification d'image pour déterminer si cette région nous intéresse ou pas dans notre cas on classifiera une zone haute fréquence texte.

Cette méthode est généralement combinée avec des méthodes « image pyramids » qui permettent de reconnaître un objet avec différentes tailles.

La méthode sliding-windows joue un rôle important dans la détection d'objets sur une image ainsi que dans sa classification.

La force de cette méthode est dans le fait que l'ensemble de l'image originale est parcouru par la fenêtre, elle garantit aucune perte d'information relative à l'image.

Sa faiblesse réside dans le fait que pour obtenir des résultats robustes il faut que la classification des objets soit unique pour chaque type d'objet. Dans notre cas l'objet est une zone de texte qui par sa diversité de caractéristiques ne peut pas être classifiée de façon unique.

De plus cette méthode est dite brute-force, son temps d'exécution est directement lié à la taille de l'image et à la taille de la fenêtre.

5.3 Synthèse

Dans le cadre de notre travail je propose d'utiliser des méthodes dites de machine learning à savoir :

- Arbre de décision
- Réseau de neurones

Après discussion avec Mario Bellino nous avons décidé d'abandonner les méthodes de réseau de neurones qui sont très complexe à analyser et à régler.

Les méthodes de « sliding-windows detection » sont très vite abandonnées. En effet, le fait que ces méthodes sont directement dépendantes de la taille de la fenêtre et que je dispose d'image avec une très grande résolution il paraît évident que ce type de méthode serai beaucoup trop gourmande en temps de processing.

Le fait d'utiliser des méthodes « image pyramids » afin de contrer le problème lié à la taille de l'image serait à mon sens pas efficace car les zones de texte ont des caractéristiques qui dépendent aussi de la résolution de l'image.

6 Pre-processing

Dans ce chapitre on énumèrera les méthodes de pre-processing employées avec pour chacune d'elles leurs forces et faiblesses. On présentera et commentera les résultats obtenus. On expliquera finalement les raisons qui nous ont amené à effectuer certains choix.

6.1 Filtre « Laplacian »

La méthode Laplacian consiste à calculer sur une image la seconde dérivée dans les directions horizontales et verticales. On obtient en sortie une image sur laquelle les intensités des pixels (x,y) est le résultat de la somme des deux dérivées obtenue.

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Figure 2 : Equation Laplacian

Cette méthode est très sensible au bruit. Pour contrer cela, on applique souvent sur l'image un filtre de type Gaussian qui permet de réduire les composantes de bruit à haute fréquence.

La méthode « Laplacian » génère une image de même dimension monocouleur.

Dans nos méthodes nous avons utilisé le filtre Laplacian sur les trois bandes de couleur ainsi que sur des images monocouleur.

6.1.1 Filtre Laplacian sur niveau de gris

La figure ci-dessous montre les étapes effectuées pour calculer un Laplacian sur des images de niveau de gris.

Figure 3 : Pipeline Laplacian niveau de gris

On effectue 3 opérations distinctes avant d'appliquer la méthode Laplacian à savoir :

- Gaussian Filter (5x5)
- Conversion image BGR en image grayscale
- Application d'un threshold de type Gaussian

Le filtre Gaussian permet comme décrit auparavant de réduire les composantes de bruit haute fréquence. Cette étape sera effectuée dans toutes les méthodes de pre-processing.

La conversion BGR en image grayscale consiste à remplacer chaque pixel de l'image par la relation suivante : $Y = 0.587 \cdot green + 0.299 \cdot red + 0.114 \cdot blue$ on obtient alors une image de même dimension codée sur une couleur.

On applique ensuite sur cette image un Threshold dit adaptatif de type Gaussian[2] proposé par la libraire OpenCV 2.4.x. La valeur seuil du Threshold Gaussian est une somme pondérée des pixels voisins délimités par une fenêtre de taille 3x3.

Le temps de processing est d'environ 120 ms par image.

[2] http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0

6.1.2 Filtre Laplacian sur des images niveau de couleurs

La figure ci-dessous montre les étapes effectuées pour faire un Laplacian sur les trois bandes de couleur d'une image.

Figure 4 : Pipeline Laplacian niveau de couleur

Les étapes de cette méthode sont semblables à la méthode Laplacian sur des images de niveau de gris.

On applique la conversion en niveau de gris après le calcul du Laplacian et on y applique cette fois-ci des Thresholds adaptatifs de type Otsu's[3] et toZero[4].

6.1.3 Maximum Laplacian sur les 3 bandes de couleur

La figure ci-dessous montre les étapes effectuées pour calculer un Laplacian sur les trois bandes de couleur pour enfin sélectionner le maximum.

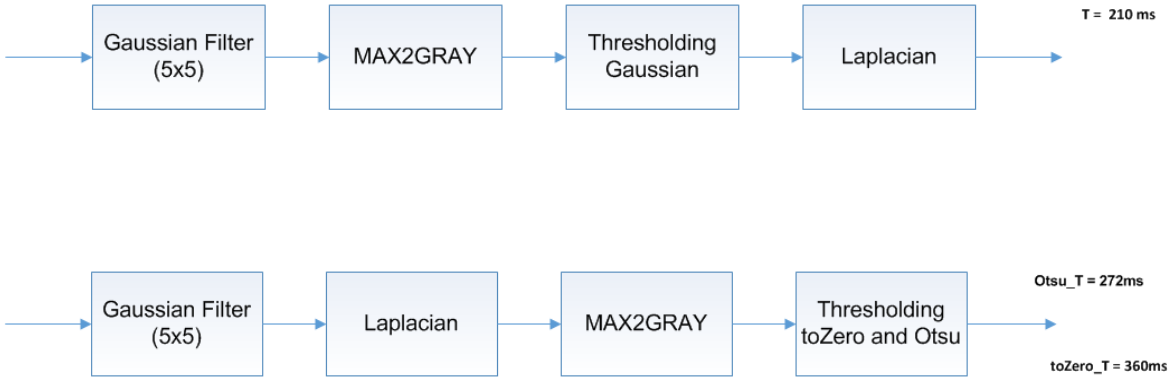

Figure 5 : Pipeline Laplacian MAX2GRAY

[3] https://fr.wikipedia.org/wiki/M%C3%A9thode_d%27Otsu (consulté le 07.07.2016)
[4] http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

L'unique différence de ces méthodes réside sur le calcul du maximum des trois bandes de couleur de l'image que l'on effectue soit avant le Laplacian, soit après.

On remarque que le temps de processing de ces méthodes est sensiblement plus important que les méthodes Laplacian présentées auparavant cela est dû essentiellement aux opérations effectuées pour séparer les trois bandes de couleur de l'image.

6.2 Filtre de Sobel[5]

La méthode de Sobel consiste à calculer la première dérivée sur les axes x et y d'une image. Cette méthode combine un lissage gaussien et le calcul différentiel. Elle permet d'obtenir une approximation de l'intensité d'une image.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

$$G = \sqrt{G_x^2 + G_y^2}$$

Figure 6 : Sobel équation

L'avantage de la dérivée première de Sobel est le fait qu'elle est beaucoup moins sensible au bruit qu'une dérivée seconde obtenue en utilisant un filtre Laplacian.

[5] http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

6.2.1 Filtre de Sobel sur image niveau de gris

La figure ci-dessous montre les étapes effectuées pour faire un Sobel sur des images de niveau de gris.

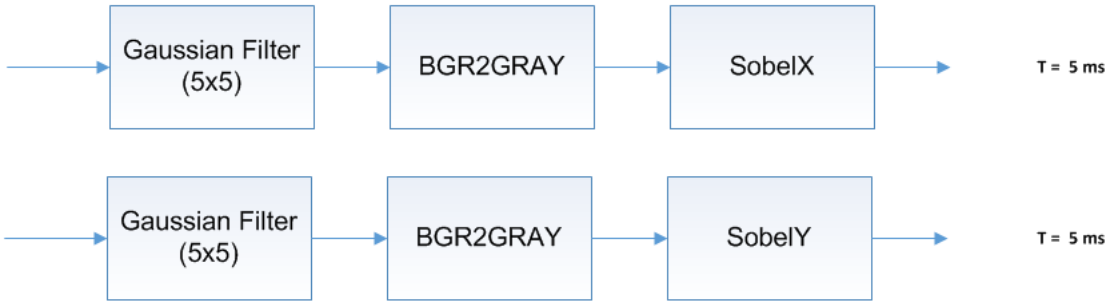

Figure 7 : Pipeline méthodes Sobel

La dérivée de Sobel sur des images monocouleurs peut engendrer une perte d'information dans le cas où la colorimétrie d'une image est telle que sa représentation en niveau de gris présente un contraste très faible.

6.2.2 Filtre de Sobel Maximum et Somme

La figure ci-dessous montre les étapes effectuées pour faire un Sobel sur chacune des trois bandes de couleur et pour sélectionner ensuite soit le maximum soit la somme.

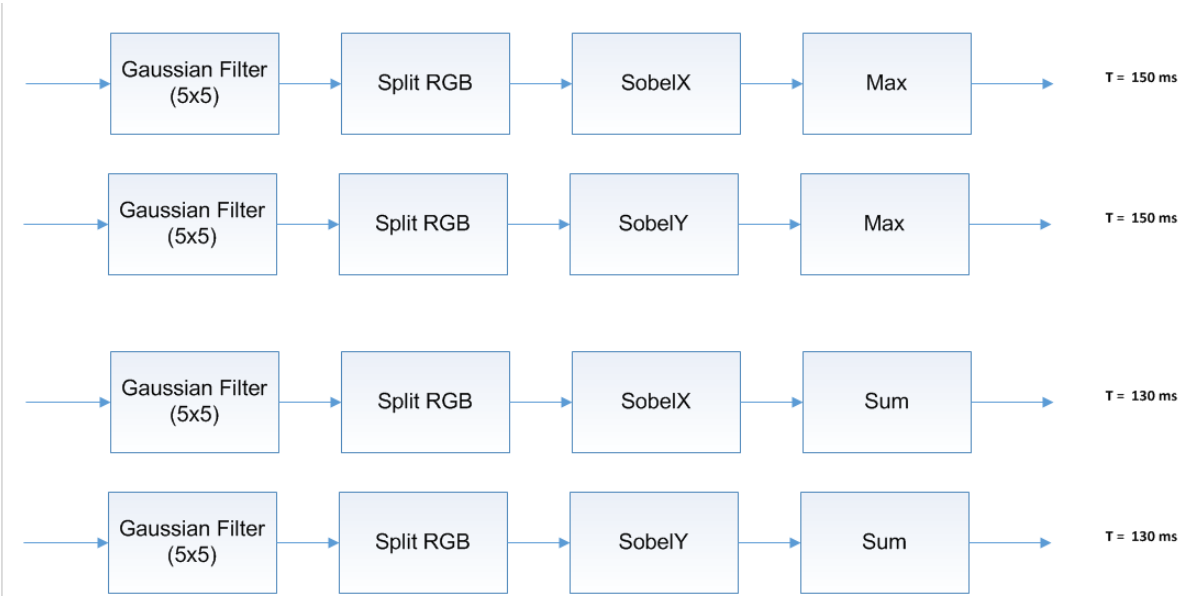

Figure 8 : Pipeline Sobel MAX et SUM

L'objectif de ces méthodes est d'effectuer le calcul du maximum et de la somme sur les 3 bandes de couleurs. Cela permet d'atténuer les faiblesses d'un filtre Sobel sur des images monocouleur ou le contraste est très faible.

6.3 Résultats pre-processing

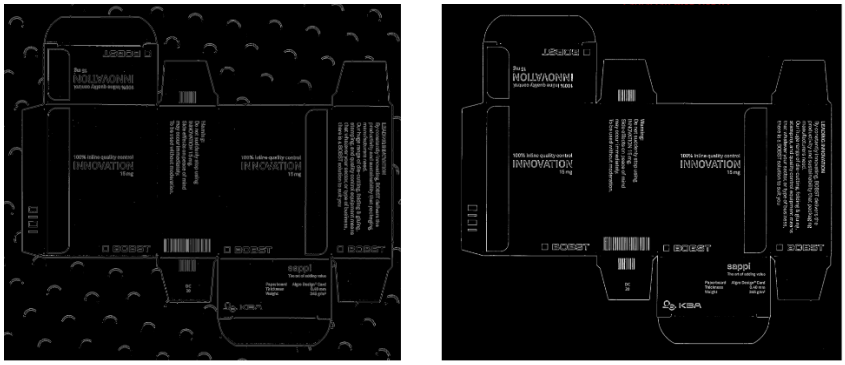

Figure 9 : Laplacian (MONOCOULEUR & COULEUR)

La figure 9 présente les résultats obtenus après un pre-processing de type Laplacian sur une image monocouleur et sur une image couleur. On remarque immédiatement par l'absence d'apparition du tapis de fond que le calcul Laplacian sur une image couleur est beaucoup moins sensible au bruit.

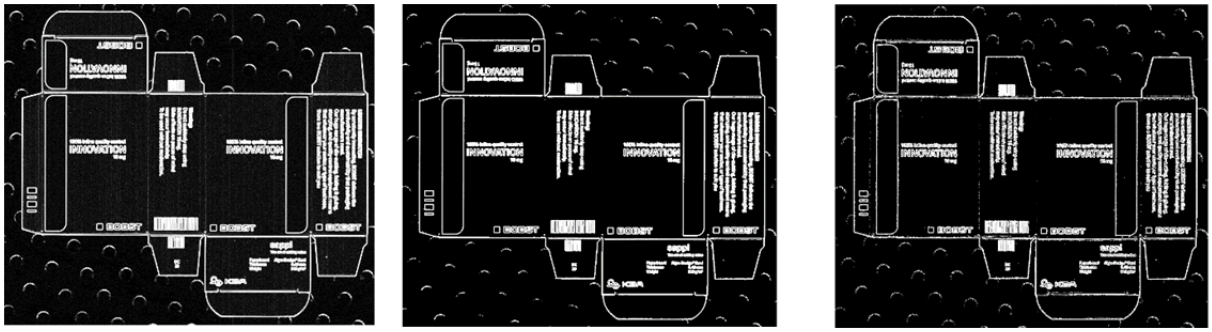

Figure 10 : Sobel (MONO & MAX & SUM)

La figure 10 présente les résultats obtenus après un pre-processing de type Sobel, sur l'ensemble des images. La méthode qui somme les résultats du Sobel sur les trois bandes de couleur génère plus de bruit que les deux autres méthodes. Quant aux deux autres méthodes elles donnent des résultats qui sont à première vue semblables. Toutefois il faut comprendre qu'avec la méthode « MONO » on risque de perdre des informations sur des images dont la représentation monocouleur est très faiblement contrasté. On retient donc comme méthode de pre-processing un filtre de Sobel sur les trois bandes de couleur et sélection du maximum obtenus.

6.4 Synthèse

Sur la base d'expériences empiriques de plusieurs méthodes de pre-processing mentionnées ci-dessus ainsi que par la comparaison des résultats obtenus pour chaque méthode. Je décide d'utiliser des méthodes basées sur le maximum du filtre Sobel sur les 3 bandes de couleur.

L'avantage du filtre Sobel est qu'il est beaucoup moins sensible au bruit qu'un filtre Laplacian.

La somme du filtre Sobel sur les 3 bandes de couleurs génère une accumulation de bruit des 3 bandes.

Les résultats obtenus par le maximum du filtre de Sobel sur les 3 bandes de couleur et un filtre Sobel sur des images monocouleurs présentent des résultats similaires.

L'avantage principal du maximum du filtre de Sobel sur les 3 bandes de couleur réside dans le fait qu'il sera beaucoup plus performant pour des images qui une fois transformé en monocouleur perdent du contraste sur les zones texte et codes-barres.

Concernant la binarisation de l'image je décide d'utiliser un Threshold de type Otsu's qui a l'avantage d'adapter la valeur de seuillage pour toutes les zones de l'image.

7 Filtres pre-processing

Les résultats des méthodes de pre-processing génèrent des images comportant encore beaucoup de zones hautes fréquences qui ne sont pas du texte. L'objectif des filtres post pre-processing est de supprimer ces informations tout en gardant les zones de texte.

Pour cela on proposera et on implémentera des méthodes de filtre simple dans le but d'éliminer les plus grand nombre de zone haute fréquence de type non texte.

7.1 Détection est suppression contours de boîtes

Dans certains cas, les contours de boîtes qui apparaissent en sortie de pre-processing sont directement reliés à une zone de texte cela pose un certain nombre de problème pour les opérations futur de processing. Il s'agit alors de proposer une méthode permettant de détecter le contour principal de la boîte pour le supprimer.

En partant du principe que le contour de la boîte est le plus grand contour présent sur l'image il est alors simple de le détecter et de le supprimer.

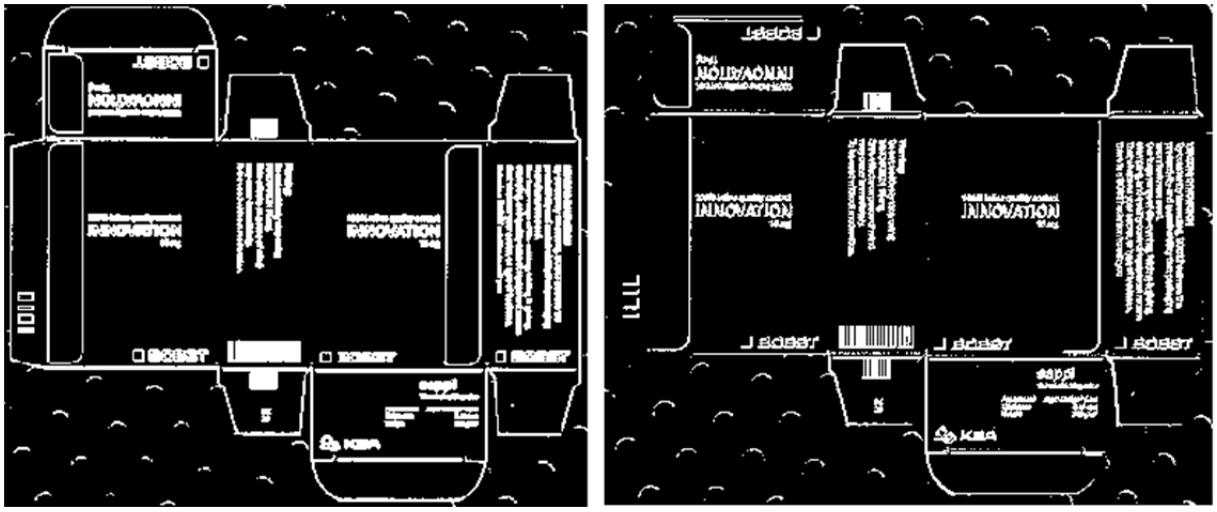

Figure 11 : Suppression contours de boites image monocouleur à gauche suppression de contours à droite

La figure 11 présente une image monocouleur avant suppression de contours et après suppression. On remarque que la méthode comporte des faiblesses pour les contours qui sont arrondi. Toutefois sachant que les machines d'inspection de Bobst implémentent déjà des algorithmes permettant la détection de contours nous pouvons négliger ce problème.

7.2 Détection est suppression des « crease »

Les « crease » ou refoulage en français, sont les zones qui vont servir au pliage de la boîte. Elles apparaissent uniquement pour les boîtes de type PCR. Les « crease » provoquent dans certains cas les mêmes problèmes que le contour de la boîte.

7.2.1 Détection des « crease » méthode n°1

Après quelques observations et en partant du principe que les « crease » sont toujours noirs, on élabore une première méthode qui permet de supprimer les « crease » qui traverse complètement la boîte horizontalement et verticalement.

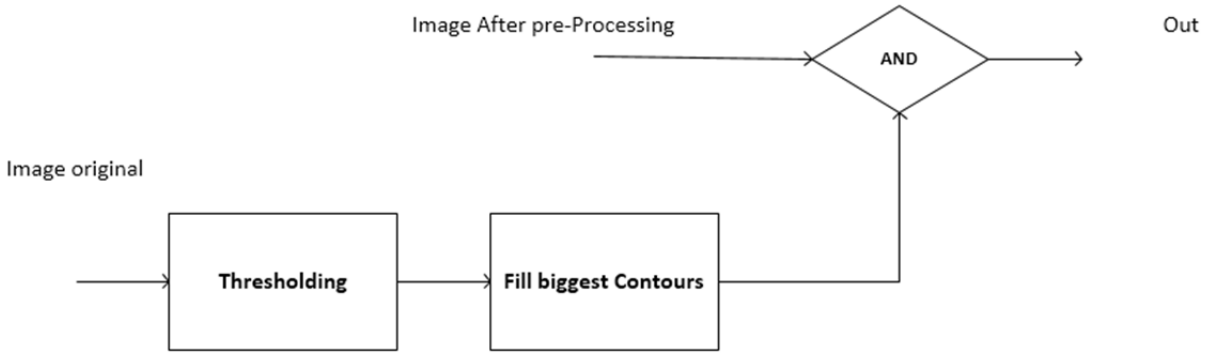

Figure 12 : Méthode suppression de « crease »

On remarque qu'avec des boîtes dont le bord et l'intérieur sont composés de noir nous perdons toutes les informations présentes sur la boîte. En effet, la valeur du seuil est définie de telle sorte que le tapis de fond (couleur noire) soit filtré. Dans le cas où la couleur de la boîte est semblable à celle du tapis il arrive que certaines informations présentent sur la boîte soit perdu.

La détection est suppression des « crease » reste cependant une méthode robuste pour l'ensemble des boîtes qui sont d'une couleur autre que noir.

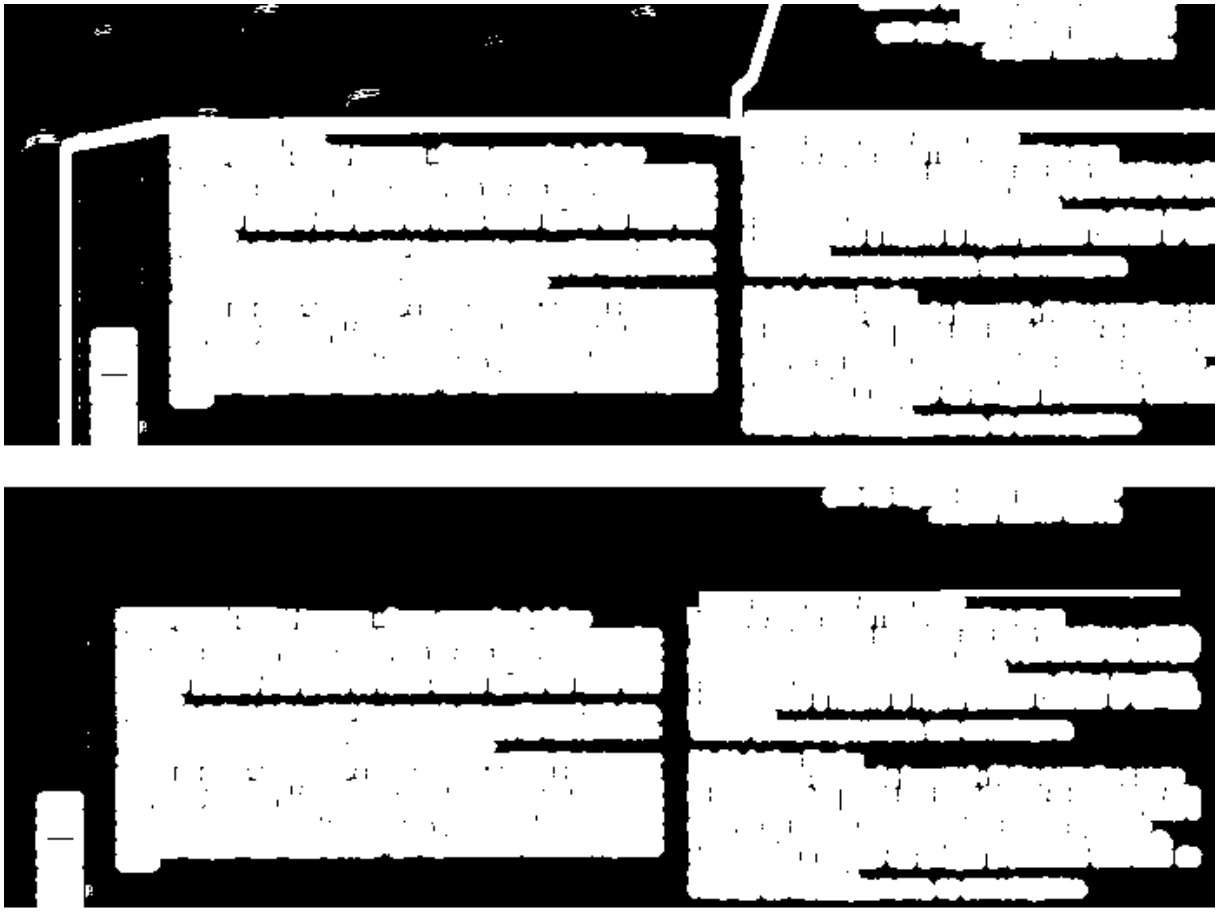

Figure 13 : Image sans détection de « crease » (haut) et avec détection de « crease » (bas)

7.2.2 Détection des « crease » méthode n°2

Afin de compenser le problème de la détection est suppression des « crease » lié à la couleur de la boîte on élabore une seconde méthode qui s'appuie sur la détection et l'approximation de rectangles. En effet, après observations on remarque que les « creases » forment généralement avec le contour de la boîte un rectangle.

Cela permet non seulement d'éliminer les rectangles formés par les « crease » à l'intérieures des boîtes mais également les motifs rectangulaires qui peuvent apparaitre sur certaines boîtes.

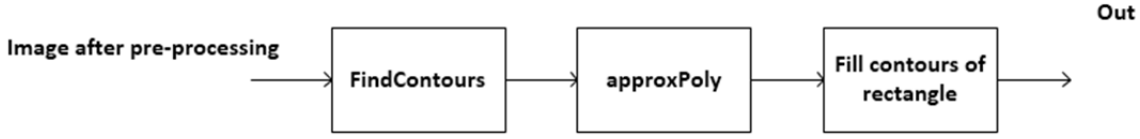

Figure 14 : Méthode suppression de « crease »

Après analyse visuelle des résultats on constate que la combinaison des deux méthodes permet d'avoir les meilleurs résultats. En effet, la deuxième méthode permet de compenser les faiblesses de la première méthode sur les boîtes composées de noire de plus elle permet de supprimer davantage de « crease » (crease intérieure).

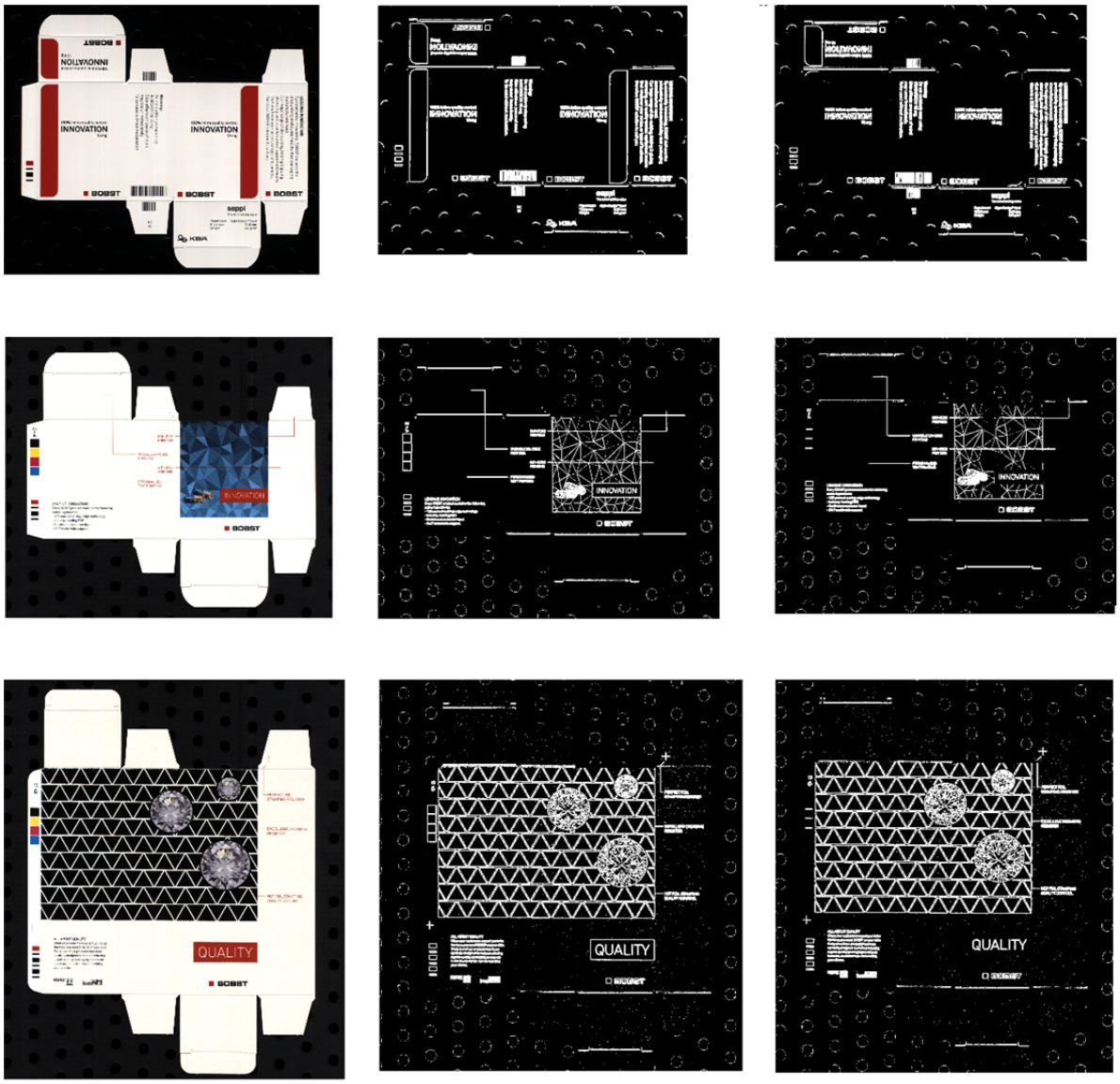

Figure 15 : Résultats méthodes suppression des « crease »
Gauche : Image originale. Milieu : Image avec crease. Droite : Image avec suppression des « crease »

7.3 Détection de code-barres et datamatrix

En utilisant uniquement des moyens de transformation morphologique et de filtre Sobel j'ai mis en place une méthode qui permet de mettre en évidence sur une image uniquement les éléments de type code-barres et datamatrix.

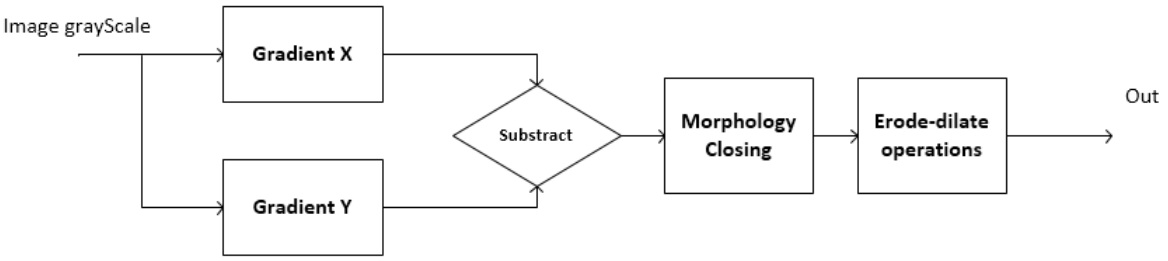

Figure 16 : Méthode détection code-barres et datamatrix

Un code-barre ou datamatix est composé uniquement d'éléments verticaux ou horizontaux. Il s'agit alors de mettre en évidence ces éléments par des filtres Sobel.

Le fait de soustraire la valeur absolu du gradient X au gradient Y permet d'isoler les informations de type code-barres et datamatrix.

Après observation des premiers résultats je remarque que certaine zone de texte apparaissent. On remarque également que les zones de textes ont beaucoup moins de surface que les zones codes-barres et datamatrix.

Pour éliminer les éléments restants autres que code-barres et datamatrix j'applique des transformations morphologiques sur la base d'un kernel de fermeture (CLOSE) puis par une série d'erosion et dilatation j'obtiens en résultats une image monocouleur avec uniquement les codes-barres et les datamatrix en évidence.

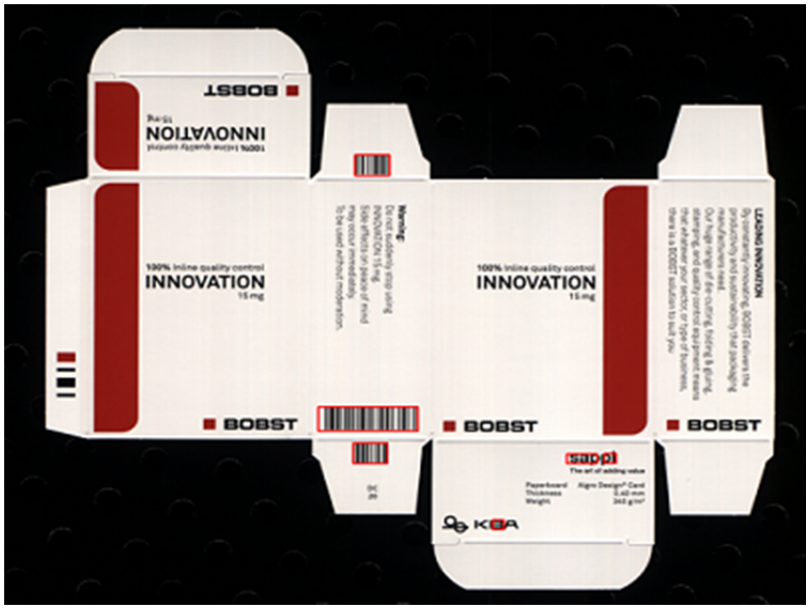

Figure 17 : Résultats code-barres détection

7.4 Synthèse

Les principaux efforts des filtres pre-processing s'articulent sur la détection et la suppression des « crease ». En effet, ces éléments sont souvent récurrents et critiques.

J'ai remarqué aussi par la suite que les caractéristiques des codes-barres sont très souvent similaires à une zone non-texte c'est pourquoi je propose et implémente une méthode basé sur le pre-processing pour mettre en évidence ces éléments.

8 Processing

L'objectif du processing est d'analyser les éléments restants pour déterminer si une zone est considérée comme étant du texte ou non. Pour cela nous avons élaboré plusieurs stratégies qui permettent d'identifier une zone non-texte pour pouvoir la supprimer.

8.1 Abstraction d'une ROI

La ROI (Region of Interest) est délimitée par des bounding-box qui sont générés suite à la détection des contours sur une image après pre-processing.

Dans un premier temps nous avons extrait et classifié chacune des bounding-box générée en fonction de leur appartenance à savoir :

- Zones de textes
- Zones de bruit

Ensuite nous avons fait une abstraction de chacune de ces zones en définissant les éléments suivants :

- Caractéristiques géométriques
- Zones spatiale
- Histogramme
- Projection orthogonale

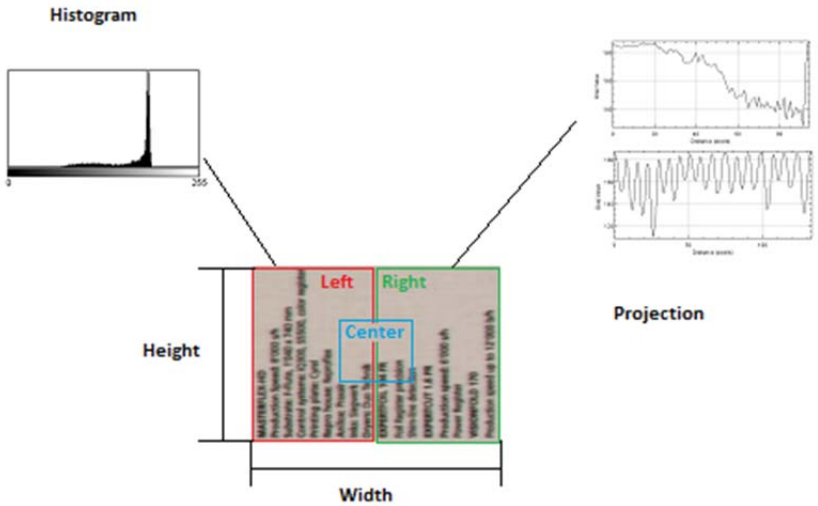

Figure 18 : Abstraction du ROI

8.1.1 Caractéristiques géométriques

Par les caractéristiques géométriques nous pouvons déterminer le périmètre et l'aire d'une bounding-box. On détermine également les rapports entre la hauteur et la largeur ainsi qu'entre la surface et le périmètre.

Les caractéristiques géométriques nous permettent de filtrer les éléments de type longues lignes droite ou encore quelques « crease » restantes.

Figure 19 : Elimination de crease grâce aux caractéristiques géométrique

8.1.2 Zones spatiales

La séparation de la ROI en trois zones spatiales (gauche, centre et droit) nous permet d'analyser chacune de ces zones et de trouver une corrélation pour déterminer si la ROI est une zone de texte ou non.

Après analyse de zone spatiale de texte et de bruit on remarque qu'une ROI de texte à une plus grande intensité dans l'ensemble de ces zones spatiale par rapport à une ROI non-texte.

L'analyse de l'intensité dans les trois zones spatiales nous permet de filtrer des éléments de types bruits tels que présenté sur la figure ci-dessous

Figure 20 : Détection bruits par analyse des zones spatiale

8.1.3 Histogramme

L'histogramme nous permet d'analyser la répartition des couleurs sur l'image. On analysera de ces histogrammes les éléments suivants :

- La position de la plus grande pointe.
- Le nombre de pointes.
- La distance entre les deux bords.
- La variance entre les deux bords.

La position de la plus grande pointe permet de déterminer quelle couleur est la plus représentée dans la ROI. En combinant la position de cette pointes avec la distance entre les deux bords de l'histogramme on peut déterminer le contraste maximum présent sur la ROI et déterminer s'il s'agit d'une zone texte ou non.

Le nombre de pointes permet d'analyser la répartition et l'intensité des couleurs sur la ROI. Si on admet qu'une zone texte est composée uniquement de deux couleurs on peut également déterminer si la ROI est du texte ou non.

La variance entre les deux bords permet dans certains cas d'identifier des zones non texte qui ont les mêmes caractéristiques qu'une zone texte.

L'analyse de tous les éléments de l'histogramme nous permet d'éliminer les zones telles que présentées sur la figure ci-dessous.

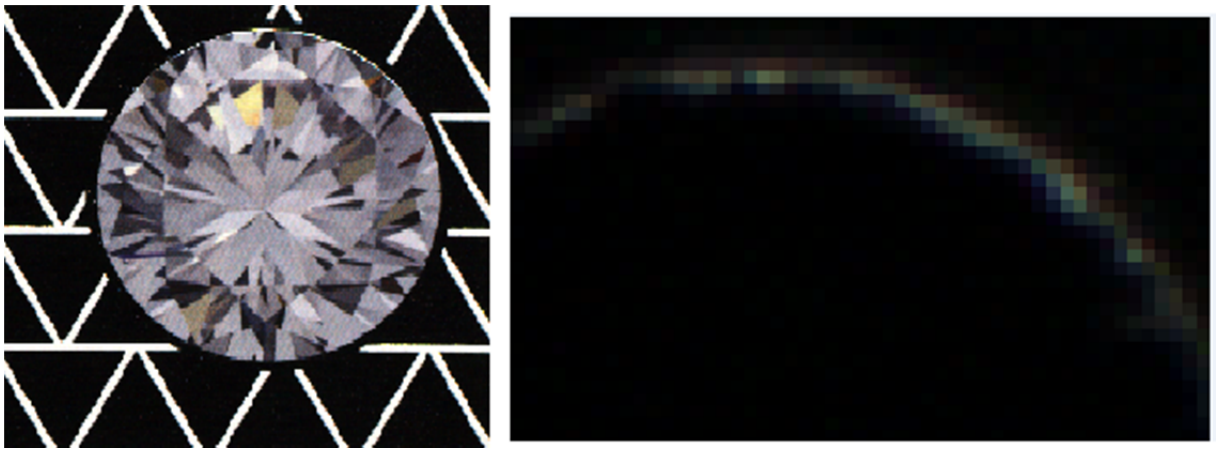

Figure 21 : Elément supprimés par caractéristiques histogramme
Gauche : Bruit type holographique. Droite : Bruit lié tapi de fond.

8.1.4 Projection orthogonale

Après analyse et observation on peut affirmer qu'une zone texte qui contient plusieurs lignes à une fréquence beaucoup plus élevée dans une direction que dans l'autre.

La projection orthogonale nous permet d'analyser la fréquence d'une ROI par rapport à ses coordonnées (x,y). On analysera de ces projections les éléments suivants :

- Nombre de maximum
- Nombre de minimum
- Différence de fréquence entre les projections horizontales et verticales

Les zones non-texte ne présentent pas ces caractéristiques fréquentielles, il est alors possible de séparer les zones de texte aux zones non-texte en s'appuyant sur les analyse ci-dessus.

Ces analyses nous permettent de filtrer les éléments suivants :

Figure 22 : Eléments supprimés par caractéristiques projection

8.2 Stratégie de processing

Sur les bases des abstractions faites sur les ROI nous avons élaboré une stratégie de processing qui privilégie la suppression de zones non-texte tout en minimisant les pertes de zones texte.

8.2.1 Filtre N°1

Le principal objectif du filtre est d'éliminer un grand nombre d'information non-texte tout en gardant un maximum de zones de texte. On propose une première série de conditions à remplir pour qu'une bounding-box soit considérée comme étant du texte. On établit les conditions sur les bases de l'abstraction de la bounding-box effectué au chapitre précédent.

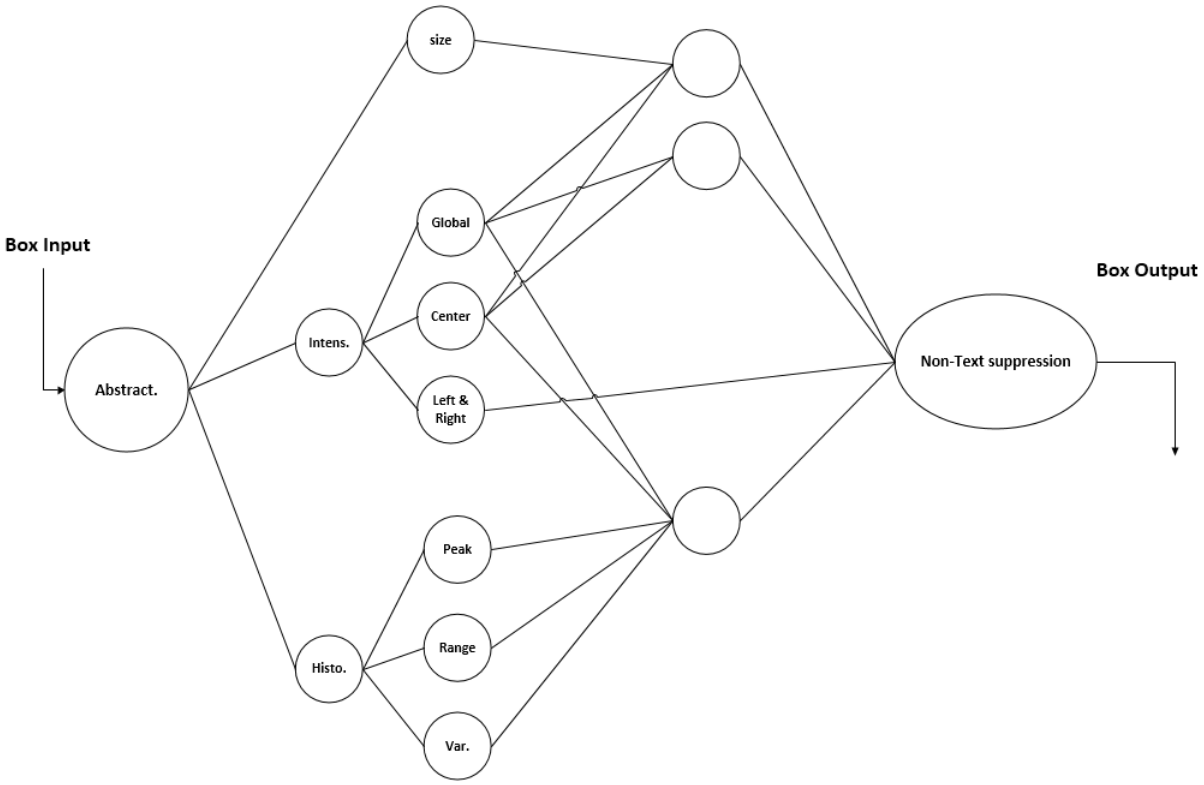

Figure 23 : Filtre N°1
Abstraction des bounding-box et critères d'évaluation

Pour définir les meilleurs paramètres du filtre on effectue plusieurs essais empiriques en utilisant des éléments texte et non-texte classifié à la main.

On analyse les résultats obtenus chaque fois que l'on modifie une contrainte, la difficulté réside dans le fait de trouver des valeurs de paramètres qui respecte le compromis entre l'élimination de zone non-texte tout en perdant le moins possible de zone texte.

Elements	*Total*	*Deleted*	*Critical*	*nCritical*	*[%]*
Overall	723	0	0	0	0.0%
Text	426	9	2	7	0.5%
Code	93	1	1	0	1.1%
Noise	204	144	0	0	70.6%

Figure 24 : Résultats filtre N°1
Comparaison des résultats avec la classification à la main des différents éléments.

La figure 23 montre les meilleurs résultats obtenus en satisfaisant le compromis entre perte de texte et suppression de bruit.

On définit par élément *nCritical* les éléments de texte qui sont considéré comme étant du gros texte et les zones texte qui sont faiblement contrastées.

Sur un ensemble de 723 échantillons analysés et testé on arrive à supprimer 70.6% de zones non-texte pour seulement une perte de 0.5% de texte.

Dans un deuxième temps, on analyse l'impact de chaque contrainte pour identifier les paramètres qui sont les plus sensibles et les moins sensibles.

Les résultats de cette analyse démontrent que les contraintes liées à l'intensité des zones spatiales d'une ROI n'ont aucun effet sur les résultats finaux. Il s'agit alors d'utiliser d'autres contraintes afin d'améliorer encore l'efficacité du filtre.

8.2.2 Filtre N°2

Sur la base et l'analyse des résultats obtenus du premier filtre on propose un deuxième filtre en utilisant cette fois-ci d'autres paramètres.

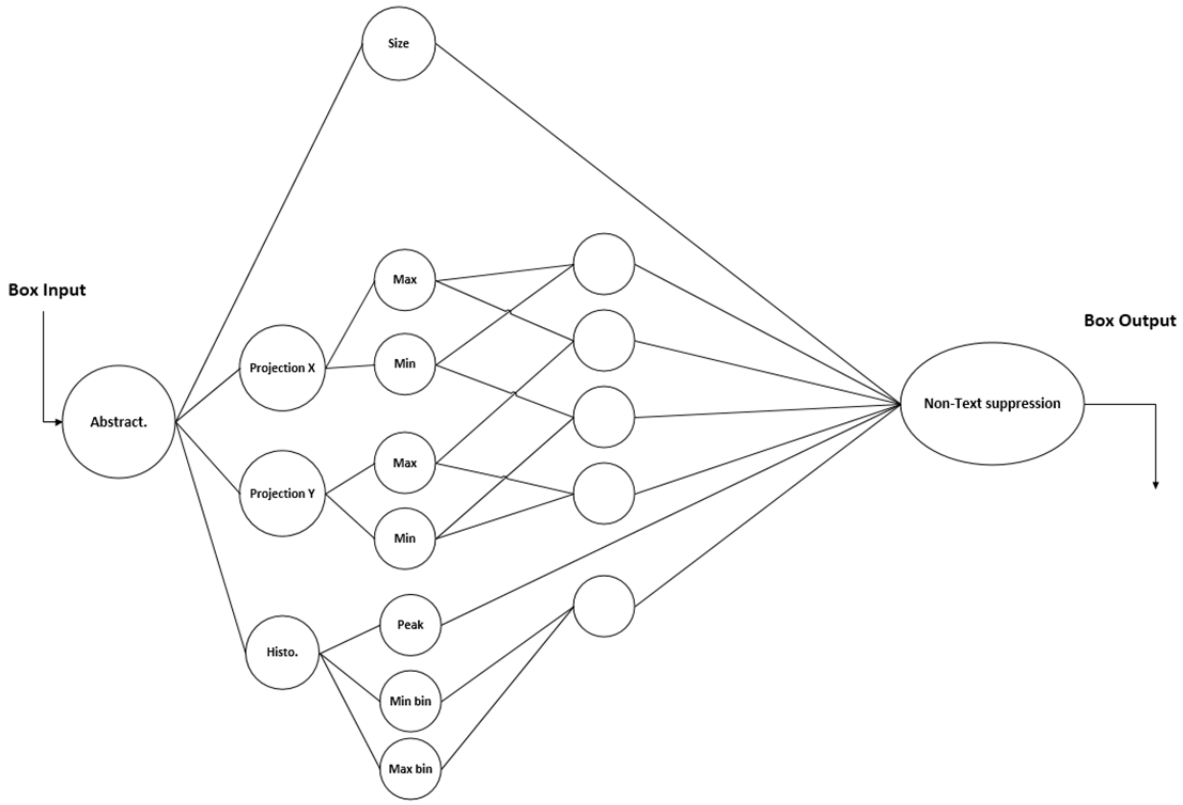

Figure 25 : Filtre N°2
Abstraction des bounding-box et critères d'évaluation

On effectue les essais dans les mêmes conditions que le filtre N°1 avec les mêmes échantillons.

Elements	Total	Deleted	Critical	nCritical	[%]
Overall	723	0	0	0	0.0%
Text	426	13	3	10	0.7%
Code	93	1	1	0	1.1%
Noise	204	154	0	0	75.5%

Figure 26 : Résultats filtre N°2
Comparaison des résultats avec la classification à la main des différents éléments

La figure 25 montre les résultats obtenus, on remarque que l'on supprime 75.5% des éléments non-textes pour une perte d'éléments textes inférieure à 1%.

On effectue également une analyse sur l'impact de chaque contrainte pour arriver à la conclusion que toutes les contraintes utilisées ont un impact sur les résultats finaux.

Les faiblesses du filtre résident dans l'élimination de zones non texte qui sont des images ; nous avons essayé plusieurs méthodes afin de filtrer ce genre d'éléments à savoir :

- Abstraction des projections par rapport à la longueur de la courbe
- K-mean algorithm[6]
- Adaptation des paramètres de filtrage

Figure 27 : Echantillon de zone « non-texte » non filtré

[6] https://fr.wikipedia.org/wiki/K-moyennes (consulté le 07.07.2016)

8.3 Machine learning « Decision Tree »

Sur la base des analyses des bounding-box et leur classification à la main, on décide d'utiliser une forme de machine learning à savoir un arbre de décision par classification.

Pour cela, on utilise la moitié des échantillons classifiés à la main pour la phase de mémorisation, ainsi que l'autre moitié pour la phase de généralisation.

Les résultats obtenus par cette méthode ne sont pas satisfaisants, en effet on supprime plus de 90% d'éléments non-texte mais en contrepartie on perd plus de 15% de zones de texte ce qui n'est pas tolérable.

Il est très difficile avec ce genre de méthode d'analyser et d'identifier la source de ces mauvais résultats.

Avec une méthode basée sur le machine learning par arbre de décision on s'attend à des résultats très satisfaisants à savoir plus de 90% de réussite.

On remarque que les résultats sont satisfaisants pour la détection de zone non-texte mais pas assez performanta en ce qui concerne la détection de zone texte.

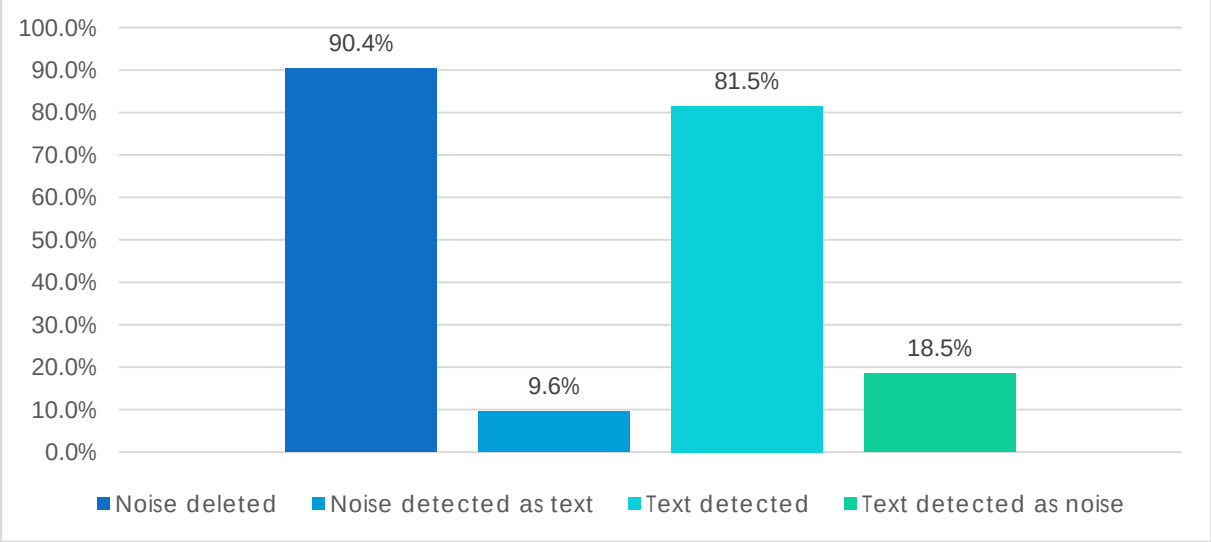

Figure 28 : Résultats Decision Tree
Comparaison des résultats avec la classification à la main des différents éléments

8.4 Synthèse

La principal difficulté du processing était le fait d'implémenter une solution qui permet de filtrer un maximum de zone non-texte tout en gardant un maximum de zone texte.

La méthode proposée sur la base d'un arbre de décision ne respecte pas ce compromis. En effet, bien que nous arrivons à supprimer plus de 90% de zone non-texte il faut remarquer qu'en contrepartie nous perdons plus de 15% du texte.

Les méthodes basées sur des paramètres de filtre établis à la main de manière empirique présentent les meilleurs résultats notamment au niveau du texte perdu qui est inférieur à 1% pour plus de 70% de zone non-texte supprimé.

Il est encore possible d'extraire plus d'informations sur certain élément de la ROI notamment au niveau des projections orthogonales ce qui pourrait améliorer le filtrage d'élément type image (visage, photo).

9 Segmentation

La segmentation est la dernière étape de notre travail ; il s'agit de combiner toutes les méthodes citées dans ce travail pour obtenir comme résultat final une image où toutes les zones hautes fréquences de texte sont délimitées sur l'image.

On présentera plusieurs résultats avec plusieurs combinaisons de méthode. On analysera les résultats de façon visuelle et automatique en utilisant un script qui compare une segmentation effectuée à la main avec une segmentation effectuée par notre algorithme.

9.1 Segmentation simple

On effectue une première segmentation simple en utilisant uniquement les filtres établis lors du processing.

L'objectif de cette première segmentation est de pouvoir généraliser nos méthodes de processing sur l'ensemble des images et de par l'analyse visuelle des résultats pouvoir détecter les faiblesses de segmentation pour enfin proposer une méthode de segmentation robuste.

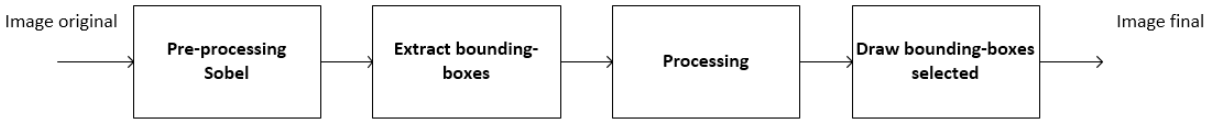

Figure 29 : Pipeline segmentation

9.1.1 Résultats et analyse

Après analyse visuelle des résultats obtenus on classifie les faiblesses de notre méthode de processing :

1. Zone de texte faiblement contrasté.
2. Grands caractères.
3. Zone de texte blanc sur fond bleu.
4. Filtre zone type image

On remarque que les deux premiers points sont des faiblesses que nous avons déjà classifiées comme étant non-critiques, en ce qui concerne le dernier point nous avons déjà traité cette faiblesse dans le chapitre du processing et par manque de temps j'ai pas trouvé de méthodes efficace pour filtrer ces éléments.

Il s'agit alors de revoir les méthodes de processing afin d'améliorer le filtrage de zone de texte blanche sur fond bleu.

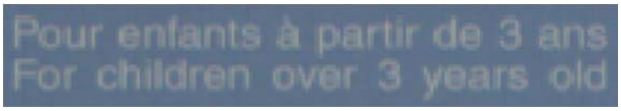

Figure 30 : Faiblesse processing

Pour éliminer ce problème du au processing, nous avons adapté les contraintes de filtrage.

On remarque également qu'une simple segmentation n'est pas la plus efficace au niveau des résultats visuels. En effet, beaucoup de zones sont délimitées à l'intérieur d'autres zone qui englobe une plus grande partie de texte cela est un problème d'over-lap.

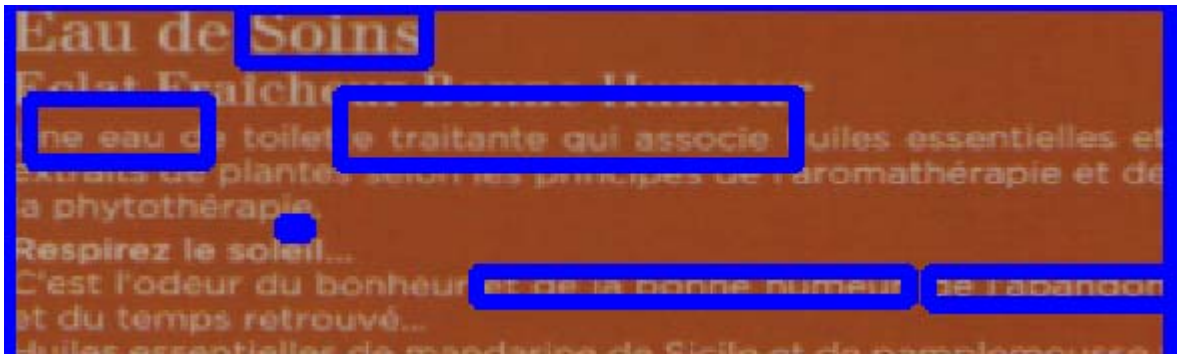

Figure 31 : Problème d'over-lap

L'analyse des résultats visuels démontre également des problèmes de segmentation liés directement au «crease».

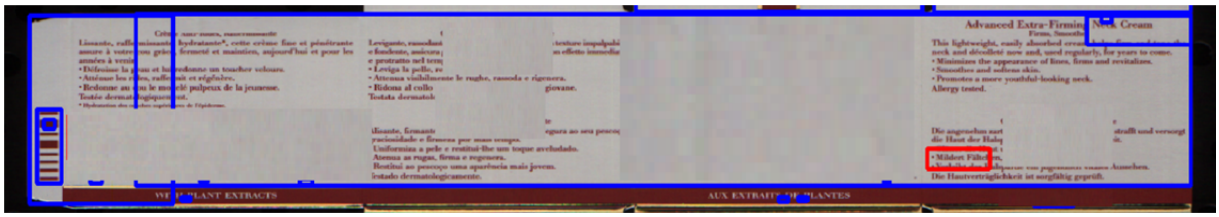

Figure 32 : Problème crease

9.2 Segmentation sur base de masque

Après avoir confirmé la robustesse de notre méthode de processing on effectue une deuxième segmentation en combinant cette fois-ci plusieurs méthodes de filtre pre-processing. Pour supprimer les problèmes liés à l'over-lap je décide de m'appuyer sur une étape intermédiaire qui consiste à segmenter les zones de textes sur des masques.

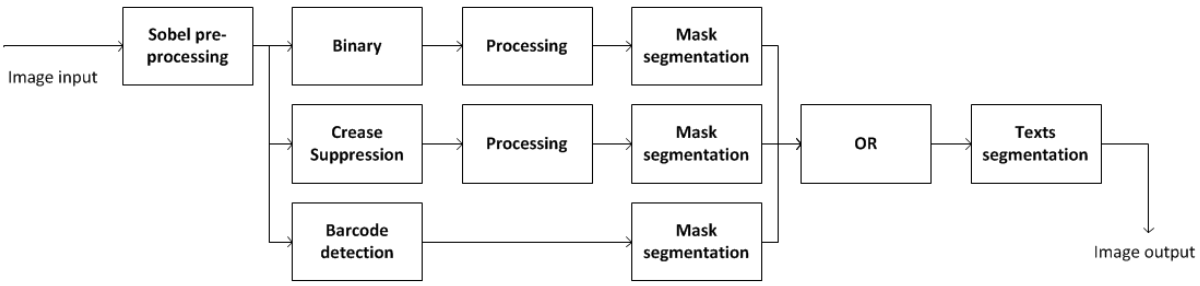

Figure 33 : Segmentation pipeline

En utilisant chaque méthode de pre-processing on effectue une segmentation que l'on sépare sur 3 masques distincts pour finalement regrouper les masques et détourer les zones de texte.

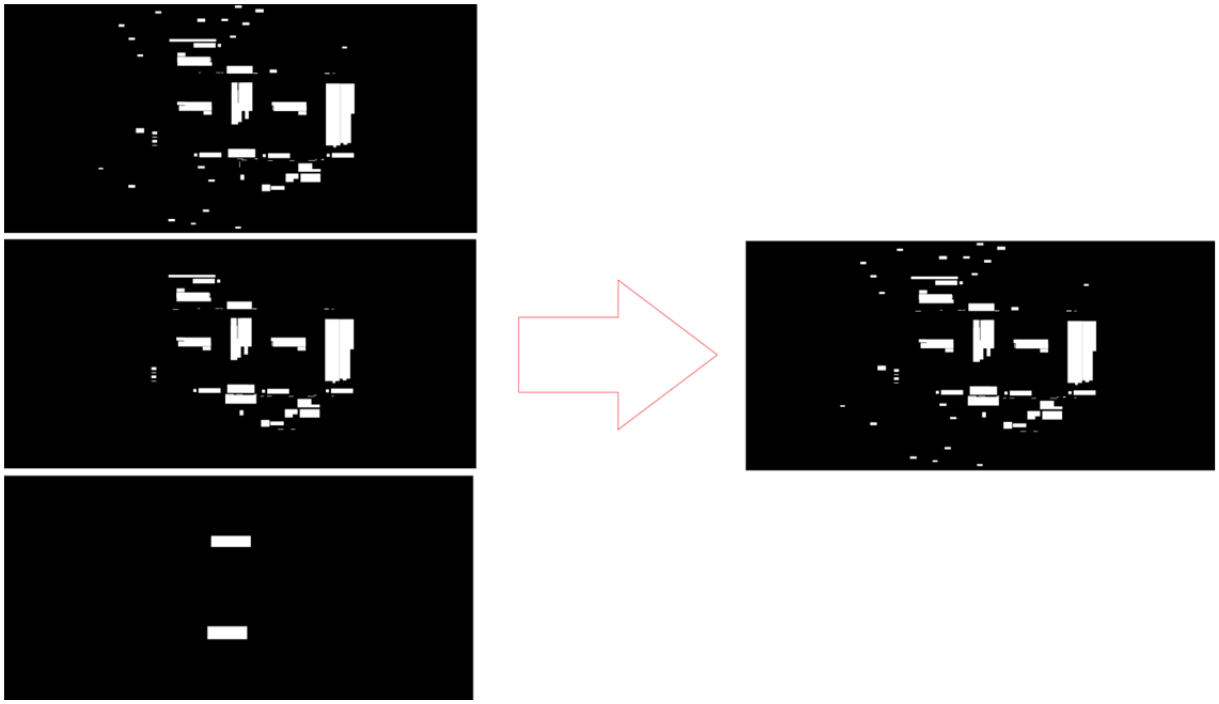

Figure 34 : Segmentation des masques

La figure 32 présente le résultat d'une segmentation basée sur l'utilisation de masque

9.2.1 Résultats et analyse

Pour analyser les résultats, on effectue cette fois-ci une analyse visuelle ainsi qu'une comparaison automatique d'une segmentation faite à la main.

Les résultats de l'analyse automatique permettent de quantifier les résultats obtenus et de vérifier si toutes les zones hautes fréquences de texte sont détourées.

Figure 35 : Résultats de l'analyse par comparaison automatique à une segmentation à la main. Pourcentage de zones texte délimité et de zone non-texte délimité pour chacun des emballages

Les résultats obtenus par comparaison automatique présentent des résultats très satisfaisant notamment pour l'ensemble des boîtes de type PCR.

Les résultats obtenus pour les boîtes de type CH et FLEXO sont moins satisfaisants, cela s'explique dû au fait que nous disposons de beaucoup moins d'échantillons et aussi du fait que les échantillons que nous avons ont les caractéristiques de l'ensemble des faiblesses de notre algorithme :

- Gros caractères
- Texte faiblement contrasté
- Non-texte de type image

Les résultats de l'analyse visuelle permettent de vérifier si les problèmes d'over-lap sont résolus et si les zones de textes sont relier entre-elle de façon logique.

Après analyse des résultats visuels on remarque que les problèmes liés à l'over-lap d'une segmentation simple sont résolus par l'utilisation des masques intermédiaire.

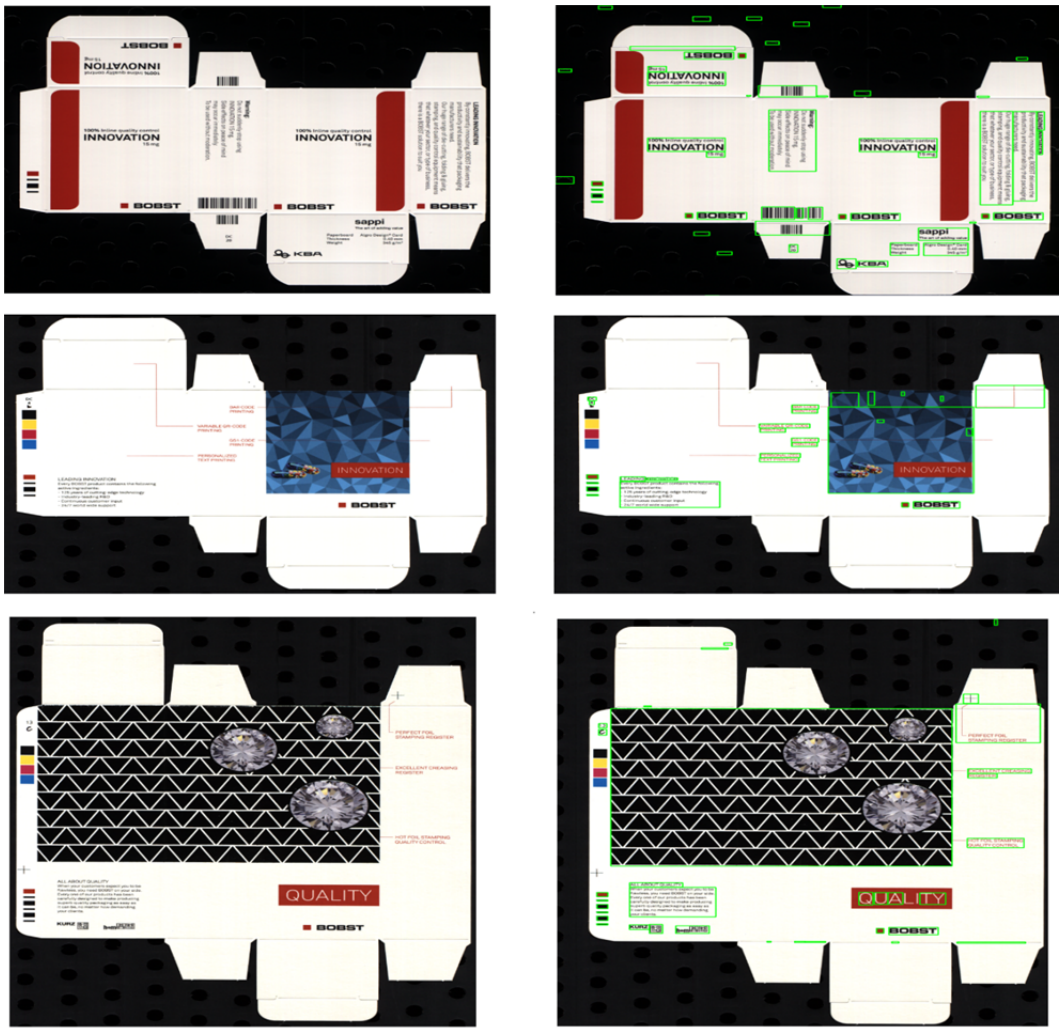

Figure 36 : Résultats visuelle

L'analyse visuelle montre également que l'utilisation et la combinaison des masques font apparaitre certaines zones qui ne sont pas du texte.

En analysant les zones non texte présente nous pouvons les classifier en trois catégorie à savoir :

1. Pattes de certaines boîtes
2. Zones de très grande taille par rapport à la boîte
3. Apparition de « crease »

Après analyse des caractéristiques des pattes on remarque que sur l'ensemble des images aucune zone texte n'apparait sur une patte. En effet, il faut savoir que les pattes permettent l'assemblage final de la boîte. Il est alors très peu probable qu'une zone texte y apparaît.

J'implémente une méthode qui permet de filtrer le résultat des masques générés en s'appuyant cette fois-ci sur des contraintes d'intensités.

Pour éliminer les zones non-textes de très grandes tailles on permet à l'opérateur de fixer des contraintes de taille pour chacune des boîtes.

Finalement l'analyse visuelle montre des faiblesses dans la façon dont les zones textes sont reliées entre-elles.

Pour éliminer cette faiblesse et relier les zones de manière logique on analyse deux approches différentes :

1. Comparaison des caractéristiques (espacement, histogramme, projection)
2. Transformation morphologique des masques obtenus

La première approche permettrait d'avoir un résultat final robuste en évitant de relier une zone de texte avec soit une zone code-barre ou avec une zone non-texte.

Sa principale faiblesse réside dans la complexité d'implémentation et son grand temps d'exécution. Il sera également compliqué de différencier une zone code-barre d'une zone non-texte ce qui est dû à leurs caractéristiques similaires.

La deuxième méthode à l'avantage des transformations morphologiques qui permet de relier immédiatement deux zones qui sont proches entre-elles.

De plus il est impossible de prouver une méthode universelle sur la façon de relier les zones de texte entre-elles. Cette approche laisse donc une certaine flexibilité est souplesse à l'opérateur sur la segmentation finale.

La faiblesse de la deuxième méthode est que la sensibilité des transformations morphologiques peut relier des zones de texte de façon non-logique.

De plus, chaque boîte dispose d'une posologie différente ce qui ne permet donc pas de généraliser une solution unique.

Nous avons choisi d'utiliser la deuxième solution car elle a l'immense avantage de laisser une certaine flexibilité et souplesse sur les résultats finaux.

Pour pallier la faiblesse de la sensibilité des transformations morphologiques on admet que les zones de texte peuvent uniquement être reliées de façon verticale.

9.3 Segmentation finale

Sur la base de l'analyse des résultats des deux segmentations précédentes on implémente la solution finale en considérant toutes les améliorations décrites auparavant.

On élabore une solution pour chaque type de boîtes PCR, FLEXO et CH en laissant à l'opérateur uniquement un contrôle sur la sensibilité de la segmentation finale.

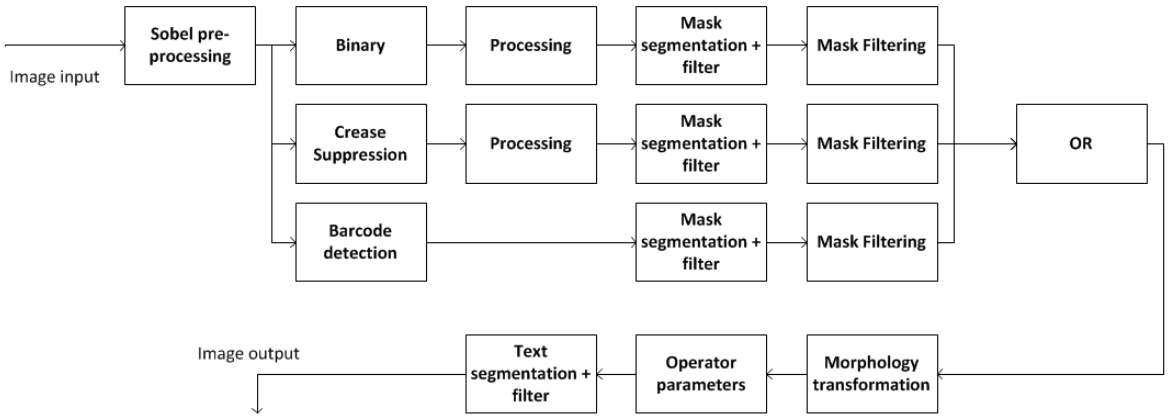

Figure 37 : Pipeline algorithme

La différence entre les solutions PCR, FLEXO et CH sont présentes uniquement sur les paramètres des transformations morphologiques.

On filtre les bounding-box présentes sur le dernier masque afin d'éliminer quelques éléments non-texte restant.

Pour finir on détour les rectangles du masque sur l'image originale pour obtenir une image où toutes les zones haute-fréquence texte et code-barres sont détourées.

9.4 Résultats

L'implémentation de la méthode de filtre pour chacun des masques permet non seulement d'éliminer les zones non-textes de type pattes, mais également les zones non-textes qui apparaissent en arrière-plan de chacune des boîtes (p.ex tapis de fond).

Le fait d'avoir utilisé la méthode basée sur les transformations morphologiques laisse une certaine souplesse à l'opérateur sur la façon de regrouper les zones texte.

On a encore des faiblesses concernant le filtrage des zones non-texte de type image notamment les visages humain ou encore les photos en arrière-plan des emballages.

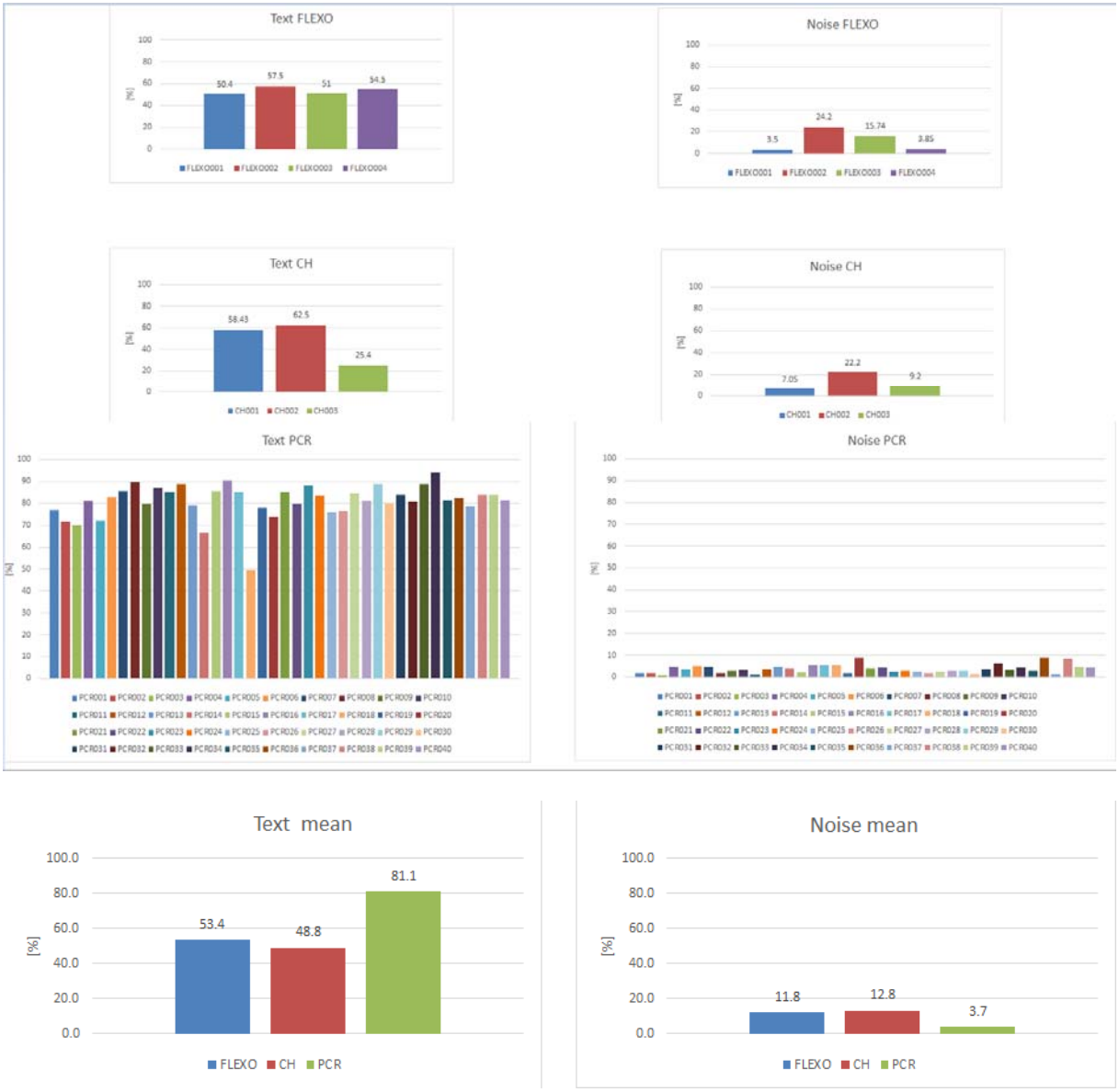

Figure 38 : Résultats de l'analyse par comparaison automatique à une segmentation à la main. Pourcentage de zones texte délimité et de zone non-texte délimité pour chacun des emballages

La figure 36 montre l'analyse automatique des résultats obtenue. On remarque qu'ils sont légèrement meilleurs que ceux présentés précédemment. En effet, on constate une augmentation de zone de texte détecté de 0.6% à 5.4% pour une diminution des zones non texte de 0.3% à 7%.

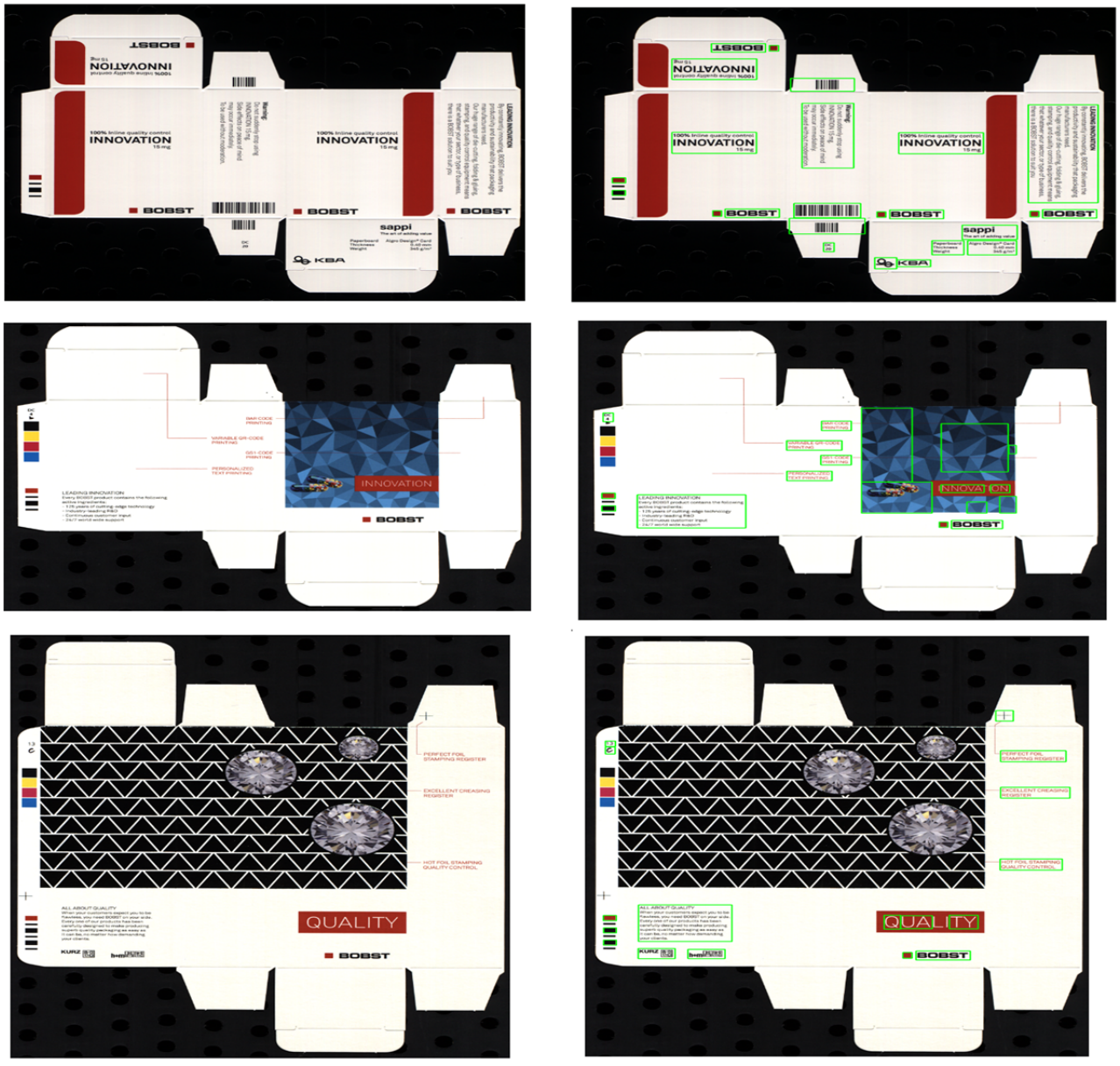

Figure 39 : Résultats segmentation finale

La figure 37 montre les résultats de l'algorithme sur l'échantillon de quelques images de type PCR.

Figure 40 : Faiblesse de l'algorithme

La figure 37 l'ensemble des images sur lesquelles on aperçoit les faiblesses de filtrage de notre algorithme à savoir (arrière-plan type photo, visage humain).

10 **Conclusion**

Dans le but de faciliter le travail d'un opérateur, mon travail consistait à détourer automatiquement les zones de texte.

J'ai présenté une solution permettant de détecter 81.1% des zones sur des boîtes de type PCR, tout en limitant les fausses détections à 3.7%. Les résultats obtenus pour les autres type de boîtes ne sont pas représentatif de la réalité car je ne dispose que de très peu d'échantillon.

De plus la solution proposée permet à l'opérateur une certaine liberté sur la façon de relier les zones de texte entre-elle tout en garantissant la robustesse de l'algorithme.

La principale faiblesse réside dans le fait que notre algorithme ne filtre pas les zones non-texte de type image, nous avons exploré plusieurs méthodes pour résoudre ce problème mais aucune d'entre elles nous à amener vers des résultats satisfaisants.

Nous n'avons pas non plus exploré des méthodes se basant sur un réseau de neurones qui pourrait peut-être donner des résultats encore plus satisfaisants.

Une première solution pour éliminer les derniers éléments indésirables serait de partir sur des méthodes permettant de classifier les éléments qui sont uniquement horizontaux et verticaux.

Il est également possible d'effectuer d'autres approche sur l'abstraction des projections orthogonales qui selon moi permettront d'éliminer assez facilement les éléments non-texte de type visage humain.

Finalement des méthodes de détection de visage humain basé sur des algorithmes de « Face Detection Haar Cascades » permettrait également d'éliminer certaines zones non-texte encore présentent.

La solution présentée a été développé avec OpenCV sous Python il s'agit maintenant de trouver une méthode pour filtrer les éléments restants de type image pour finalement traduire l'implémentation Python en langage C++.

Sion, le 14 juillet 2016 Johan Dreydemy

11 Bibliographie

1. http://www.holehouse.org/mlclass/18_Application_Example_OCR.html
2. https://fr.wikipedia.org/wiki/Reconnaissance_optique_de_caract%C3%A8res (consulté le 17.02.2016)
3. https://fr.wikipedia.org/wiki/K-moyennes (consulté le 07.07.2016)
4. http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html
5. http://docs.opencv.org/
6. Volker Märger Haikal El Abed Editors Guide to OCR for Arabic Scripts
7. Chen HUIZHONG, Sam S. TSAI, SCHROTH Georg, David M.CHEN, Radek GRZESZ
   (Robust text detection in natural images with edge-enhanced maximally stable extremal region)
8. D. MANJULA Dept. of Computer Science and Engineering College of Engineerin, Anna University Chennar, India
   (Sliding windows approach based text binarisation)
9. Deep Learning for Text Spotting D.PHIL THESIS Robotics Research Group Department of Engineering University of Oxford