ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M.Ing.

PAR
Olivier MOFFETTE

EXTRACTION DES INFORMATIONS NUTRITIONNELLES D'UNE ÉTIQUETTE ALIMENTAIRE PAR VISION ARTIFICIELLE

MONTRÉAL, LE 5 AVRIL 2011

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Jacques-André Landry, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M Tony Wong, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Mohamed Cheriet, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 27 JANVIER 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je voudrais tout d'abord remercier Jacques-André pour ses conseils, sa patience, son soutient constant et son ouverture d'esprit qui m'ont permit d'apprécier cette expérience unique tout en profitant de ma vie familiale.

Je voudrais également remercier Geneviève pour son amour, ses encouragements et sa compréhension.

Finalement, je voudrais remercier mon fils Soran pour sa joie de vivre ainsi que sa présence, qui m'a permis de réaliser jusqu'à quel point, la science est loin de rivaliser avec les capacités d'apprentissage du cerveau humain.

EXTRACTION DES INFORMATIONS NUTRITIONNELLES D'UNE ÉTIQUETTE ALIMENTAIRE PAR VISION ARTIFICIELLE

Olivier MOFFETTE

RÉSUMÉ

Ce projet à pour but le développement d'une application, basée sur un moteur de reconnaissance de caractères, qui permet l'acquisition de l'information contenue dans des photographies représentant des boites d'informations nutritionnelles. Beaucoup de projets de recherche et d'articles traitent de différentes composantes de la reconnaissance de caractère, mais très peu portent sur l'intégration globale de ces différentes composantes afin de former un tout. En plus de la reconnaissance de caractères, ce projet porte également sur le traitement et l'analyse des données une fois l'extraction de l'information complétée.

Pour notre problématique, nous n'avons trouvé aucune banque d'images de test. Nous avons donc développé une banque d'images de boites d'informations nutritionnelles avec l'aide de l'entreprise GS1 Canada. Par la suite, nous en avons extrait manuellement toutes les informations pertinentes. Cette banque d'images sert de référence pour l'évaluation des performances des différentes configurations du système développé. Notre projet n'a pas pour objectif d'évaluer les diverses composantes d'un système de reconnaissance de caractères mais plutôt d'optimiser les résultats du système dans son ensemble. Dans le même ordre d'idée, le principal critère d'évaluation de la performance de notre système est le pourcentage des boites qui sont extraites sans aucune erreur; ce qui est assez différent de l'approche qui est normalement utilisé pour évaluer les moteurs de reconnaissance de caractères soit d'évaluer le pourcentage de caractères adéquatement reconnus.

Dans ce projet, nous utilisons le moteur de reconnaissance de caractères Tesseract qui est parmi les plus performants dans la catégorie des logiciels libres. Le projet a tout de même été conçu de façon à permettre l'interchangeabilité du moteur de reconnaissance de caractères.

Ce projet est séparé en deux grandes phases. La première phase permet de comparer les différentes versions de l'OCR ainsi que de vérifier la pertinence de l'utilisation de certains modules de correction. La deuxième, quant à elle, permet d'ajuster divers paramètres de l'application afin d'optimiser les résultats.

Grâce à cette approche, nous avons réussi à traiter 58% des images sans générer la moindre erreur et au total plus de 83 % des images traitées ont généré deux erreurs ou moins. Ces résultats ne permettent malheureusement pas d'automatiser totalement le procédé, mais ils confirment la possibilité d'utiliser cette application afin d'effectuer un premier traitement massif de l'information avant d'être validée et peaufinée par un utilisateur.

Mots clés : vision artificielle, reconnaissance de caractères, OCR, Tesseract, contenant alimentaire, boite d'information nutritionnelle

NUTRIONNAL INFORMATION EXTRACTION FROM FOOD LABEL BY ARTIFICIAL VISION

Olivier MOFFETTE

ABSTRACT

This project aims at developing an application based on a character recognition engine, which allows the extraction of information contained in photographs of nutritional information boxes. A lot of research projects and articles deal with isolated recognition character components, but few deal with the overall integration of these components. In addition to the character recognition subject, this project also covers the processing and analysis of data after the extraction of the information.

For our problem, we found no bank of test images. We have developed an image database of nutritional information boxes with the help of GS1, a Canadian industrial collaborator. Subsequently, we manually extracted all relevant information. This image bank is used as reference for evaluating the performance of different configurations of our system. Our project did not aim at evaluating the various components of a system for character recognition separately but rather to optimize the results of the system as a whole. In the same vein, the main criterion for evaluating the performance of our system is the percentage of boxes, which are extracted without any error, which is quite different from the percentage of adequately recognized characters which is normally used to assess character recognition engines.

In this project, we use the Tesseract character recognition engine that is among the best within the open source class. Yet the project has been designed to allow the interchangeability of the character recognition engine.

.

This project is separated into two phases. The first phase is aimed at comparing different versions of the OCR and to verify the appropriateness of using some correction modules. On the other hand, the second phase is use to adjust various application settings to optimize the results.

With this approach, we successfully treated 58% of the images without generating any error and in total more than 83% of the processed images generated two errors or less. These results unfortunately do not permit to fully automate the process, but they confirm the possibility of using this application to conduct the first large-scale treatment of information before being validated and refined by a user.

Keywords: artificial vision, character recognition, OCR, Tesseract, food packaging, nutrition facts

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACIA	Agence canadienne d'inspection des aliments
ASCII	American Standard Code for Information Interchange
CCA	Connected components analysis
CCDA	Conseil canadien des distributeurs en alimentation
GÉPA	Guide d'étiquetage et de publicité sur les aliments
KNN	K nearest neighbor (K plus proche voisin)
MADITUC	Modèle d'Analyse Désagrégée des Itinéraires de Transport Urbain Collectif
OCR	Optical character recognition (reconnaissance optique de caractères)

INTRODUCTION

Au cours des dernières années, nous avons pu remarquer l'importance croissante que représente la nutrition aux yeux des citoyens canadiens. « En 2001, près de neuf Canadiens sur dix (88 %) affirmaient que la nutrition était un facteur important […] dans le choix des aliments qu'ils consomment »(Canada, 2002, p. 1). Plusieurs points importants ressortent de cet engouement pour leur alimentation. « Selon Nutrition : évolution et tendances (nutrition, 2001), les principales préoccupations des Canadiens et des Canadiennes en matière de nutrition incluaient les matières grasses (82 %), les vitamines (82 %), les gras saturés (80 %), les fibres (78 %), le calcium (77 %) et les protéines (77 %). »(Canada, 2002, p. 2). Depuis, les entreprises ont répondu à la demande en mettant sur le marché des produits à faible teneur (par exemple : margarine sans sel ou avec moins de gras)(Becel, 2010), enrichis de différents éléments (par exemple : lait avec oméga-3 ADH)(Natrel, 2010) ou encore en affichant, bien en vu, les éléments nutritifs qui avantagent le produit (Gadoua, 2010). Mais comme toutes les entreprises n'ont pas nécessairement le même intérêt à présenter l'information nutritionnelle de leurs produits, il existe deux lois rédigées par le gouvernement du Canada qui permettent à la population d'avoir accès à l'information nécessaire afin d'assurer le suivi de leur alimentation : la Loi sur les aliments et drogues (Canada, 24 mars 2010) et la Loi sur l'emballage et l'étiquetage des produits de consommation (Canada, 25 mars 2010). Malgré le fait que nous ayons accès à l'information de chaque produit, il reste tout de même très difficile de faire un choix éclairé vu la très grande quantité de produits disponibles sur le marché. Afin de donner une idée de l'ampleur du problème, l'épicerie où j'effectue mon marché propose plus de 24 000 produits possédant une boite d'information nutritionnelle, ce qui est cohérent avec l'information que nous avons obtenue du conseil canadien des distributeurs en alimentation (CCDA), qui lui affirme qu'il y a en moyenne 30 000 produits (tous produits confondus) dans une épicerie.

GS1, une entreprise à but non lucratif qui offre un soutient aux entreprises privées, particulièrement au niveau de la validation des contenants alimentaires, nous a approché afin

d'investiguer la possibilité de développer un prototype d'application qui permettrait d'extraire les informations nutritionnelles des étiquettes alimentaires. Cette information permettra de vérifier l'exactitude de l'impression des informations sur les contenants alimentaires. En la jumelant à la liste des ingrédients, il serait également possible de vérifier la cohérence entre ces deux sources d'information. Finalement, il serait également possible de concevoir une base de données qui pourrait aider les nutritionnistes et la population à effectuer un meilleur suivi de leur alimentation. L'information en question serait extraite de photographies numériques prises en studio pour ensuite être regroupées dans une base de données. Il serait très long et ardu d'effectuer cette tâche manuellement, considérant le volume important de produits alimentaires actuellement sur le marché. Nous pouvons prendre en exemple le site web www.les-calories.com qui a répertorié manuellement les calories, les protéines, les glucides et les lipides de 7 333 produits différents grâce à la collaboration de leurs membres, et ce, depuis 2008. D'un autre côté, il est aussi très risqué de construire une base de données de façon entièrement automatique (Sun-Hwa, Joon Ho et Jin-Hyung, 1999) sachant que les moteurs de reconnaissance de caractères actuels sont très performants, mais n'ont pas un taux de reconnaissance de 100 %.

Depuis plusieurs années, l'industrie et le milieu de la recherche tentent d'automatiser les procédés et le traitement de l'information. L'informatique permit d'automatiser le traitement de l'information et par la suite, la réseautique permit la distribution de cette même information. Cependant, à elle seule, l'informatique n'a aucune emprise sur le monde réel; heureusement, la robotique et les automates programmables ont été développés afin de pouvoir interagir sur le monde. Il est malheureusement beaucoup plus facile, pour la machine, d'agir sur le monde que de le percevoir. Par exemple, l'invention de l'imprimante arriva bien avant celle des moteurs de reconnaissance de caractères; la raison en est fort simple, lors de la création (par exemple l'impression) nous contrôlons beaucoup mieux les divers paramètres.

Il est depuis longtemps possible de numériser du texte, cependant celui-ci était initialement sauvegardé sous forme d'image. Cela permettait à l'usager de le consulter, mais non de le traiter numériquement, par exemple d'y effectuer une recherche. Les moteurs de reconnaissance de caractères ont donc permis d'extraire le contenu alphanumérique de ces images. Malheureusement, ces moteurs possèdent de bien meilleures performances lorsque le contexte est simple. Une page remplie de texte telle que celle que vous lisez présentement est beaucoup plus facile à reconnaître qu'une page d'une publication populaire contenant plusieurs colonnes parsemées d'images qui, elles-mêmes, peuvent contenir du texte. La méthode d'acquisition peut, elle aussi, compliquer énormément les choses. La reconnaissance d'une page acquise à l'aide d'un numériseur sera beaucoup plus facile à traiter que la même page acquise à l'aide d'une caméra numérique, car cette dernière peut introduire des artéfacts dus à des problèmes d'éclairage, d'angle et comme la page n'est pas obligatoirement à plat il pourrait aussi y avoir des problèmes de déformations.

Avec le coût toujours de plus en plus faible de la mémoire de stockage, nous voyons apparaitre une tendance qui aurait été inconcevable il y a encore peu d'années; celle d'entreposer de façon numérique toute l'information possible afin de la consulter ou d'y effectuer des recherches. Un exemple flagrant de cette tendance est le « Google Book Library Project » qui a pour but ultime de créer un catalogue de tous les livres avec la collaboration des bibliothèques et des éditeurs (Google, 2010).

Comme il est mentionné ci-dessus, le but de ce projet de recherche est de développer un prototype qui permettra de valider la faisabilité d'une application permettant l'extraction des informations nutritionnelles dans un contexte industriel. L'objectif principal est donc la création d'une application fonctionnelle et efficace, que nous désirons par la suite améliorer en optimisant les critères suivants : maximiser l'exactitude des résultats, minimiser l'intervention humaine et finalement minimiser le temps de traitement. Pour réussir un tel exploit, il nous faudra intégrer plusieurs méthodes les unes avec les autres. Pour ce faire, nous préconisons une approche modulaire qui facilite l'interchangeabilité des méthodes.

Malgré le fait que la majeure partie de la recherche en reconnaissance de caractère est présentement axée vers la reconnaissance de caractères manuscrits, nous nous concentrerons uniquement, dans le cadre de notre recherche, sur la reconnaissance de caractères typographiques, domaine qui a été fortement étudié au cours des dernières décennies.

Afin de pouvoir mener à bien notre projet, nous devons nécessairement en réduire la portée en posant les quelques hypothèses suivantes:

1) toutes les boites d'information nutritionnelle répondent aux exigences du chapitre 5 du « Guide d'étiquetage et de publicité sur les aliments 2003. »;

2) il n'y a qu'une seule boite d'information nutritionnelle par produit;

3) toutes les boites d'information nutritionnelle ont été conçues à partir du modèle standard bilingue (simplifié ou non).

Ce mémoire est organisé de la façon suivante : le premier chapitre présente la revue de la littérature. Dans le deuxième chapitre, nous décrivons la méthodologie qui est employée afin de parvenir à nos résultats. Dans le troisième chapitre, nous présentons le fonctionnement de notre prototype, les algorithmes qui le composent, ainsi que les diverses modifications qui ont été apportées en cours de développement. Les chapitres quatre et six présentent les différents tests que nous avons effectués ainsi que les critères d'évaluations. Les chapitres cinq et sept, quant à eux, présente les résultats que nous avons obtenus ainsi que leur interprétation. Dans le chapitre huit, nous discutons de divers éléments importants de ce projet. Ce mémoire se termine bien entendu par une conclusion et quelques recommandations pour l'avenir du projet.

CHAPITRE 1

REVUE DE LITTÉRATURE

Ce chapitre a pour objectif de présenter la littérature de base nécessaire à la compréhension du défi que représente notre problématique. Ce chapitre est séparé en 5 parties, certaines plus imposantes que d'autres. La première partie survole le *Guide d'étiquetage et de publicité sur les aliments* qui contraint, entre autres, la création des boites d'informations nutritionnelles sur les produits alimentaires préemballés. En second lieu, nous présentons quelques aspects de la recherche opérationnelle qui nous permettent d'optimiser nos calculs. La troisième partie de ce chapitre présente les divers aspects de la reconnaissance optique de caractère dont nous avons besoin pour résoudre notre problématique. La quatrième partie présente le moteur de reconnaissance optique de caractère Tesseract que nous utilisons dans le cadre de ce projet. Finalement, nous introduisons le concept de l'approche orientée résultat qui guidera l'analyse de nos résultats.

1.1 Guide d'étiquetage et de publicité sur les aliments

Le *Guide d'étiquetage et de publicité sur les aliments* (ACIA, 2003) est un document qui a été rédigé en 2003 par l'Agence canadienne d'inspection des aliments ainsi que Santé Canada afin d'aider les fabricants qui importent, fabriquent et/ou vendent des aliments préemballés au Canada à respecter les différents lois et règlements qui s'appliquent à l'étiquetage et à la publicité (ACIA, 2003). Les deux principales lois sont : la *Loi sur les aliments et drogues* (Canada, 24 mars 2010) et la *Loi sur l'emballage et l'étiquetage des produits de consommation* (Canada, 25 mars 2010). Suite à la parution du guide, les deux lois citées ci-dessus ont subi des modifications. Ne pouvant juger l'impact de ces modifications sur le guide, nous avons décidé de baser la présente recherche sur les informations contenues dans le guide d'étiquetage et de publicité sur les aliments. Il est cependant logique de croire que si des modifications majeures avaient été apportées, un nouveau guide aurait été produit.

La *Loi sur les aliments et drogues* (Canada, 24 mars 2010) « … interdit à quiconque d'étiqueter, d'emballer, de traiter, de préparer, de vendre ou d'annoncer un aliment de manière fausse, trompeuse ou mensongère ou susceptible de créer une fausse impression quant à sa nature, sa valeur, sa quantité, sa composition, ses avantages ou sa sureté. » (ACIA, 2003, p. 1.1), tandis que la *Loi sur l'emballage et l'étiquetage des produits de consommation* (Canada, 25 mars 2010) « assure l'uniformité de la méthode d'étiquetage et d'emballage des produits préemballés de consommation » (ACIA, 2003, p. 1.2). Le but de cette législation est d'assurer au citoyen canadien la possibilité de « faire des choix alimentaires éclairés et [de] sélectionner les aliments menant à une alimentation plus saine » (aliments, 22 avril 2009). Dans le cadre de ce projet, nous nous intéressons principalement aux chapitres 5 et 6 du *Guide d'étiquetage et de publicité sur les aliments* (ACIA, 2003) qui résume et condense les restrictions et les latitudes applicables à l'étiquetage nutritionnel des aliments préemballés qui seront vendus au Canada. Nous pouvons donc, grâce à ce document, déterminer les diverses caractéristiques exigées ainsi que les variations permises concernant les boites d'informations nutritionnelles.

Tout d'abord, il est à noter que l'ordre d'apparition des éléments dans le tableau de la valeur nutritive ne peut pas être modifié. Un certain nombre d'éléments sont obligatoires tandis que la plupart sont optionnels.

La Figure 1.1 présente la liste des éléments qui doivent obligatoirement se retrouver dans le tableau de la valeur nutritive. On y retrouve tout d'abord l'échantillon utilisé pour calculer les quantités des éléments nutritifs. Celui-ci étant présenté en gramme ou en millilitre. L'échantillon est suivi du nombre de calories, de la quantité de lipides (en spécifiant les quantités de lipides saturés et trans), de cholestérol, de sodium, de glucides (en spécifiant les quantités de fibres et de sucres) et de protéines. Ces éléments sont tous présentés en gramme ou en milligramme, à l'exception des calories. Certains de ces éléments présentent également le pourcentage de la consommation quotidienne recommandée pour un régime quotidien moyen de 2000 calories. Une deuxième section présente les vitamines et minéraux; les seules

qui sont obligatoires sont la vitamine A, la vitamine C, le calcium et le fer. Cette deuxième catégorie est uniquement représentée par le pourcentage de la consommation quotidienne recommandée.

Nutrition Facts
Valeur nutritive
Per 1 bowl (300 g) / Pour 1 bol (300 g)

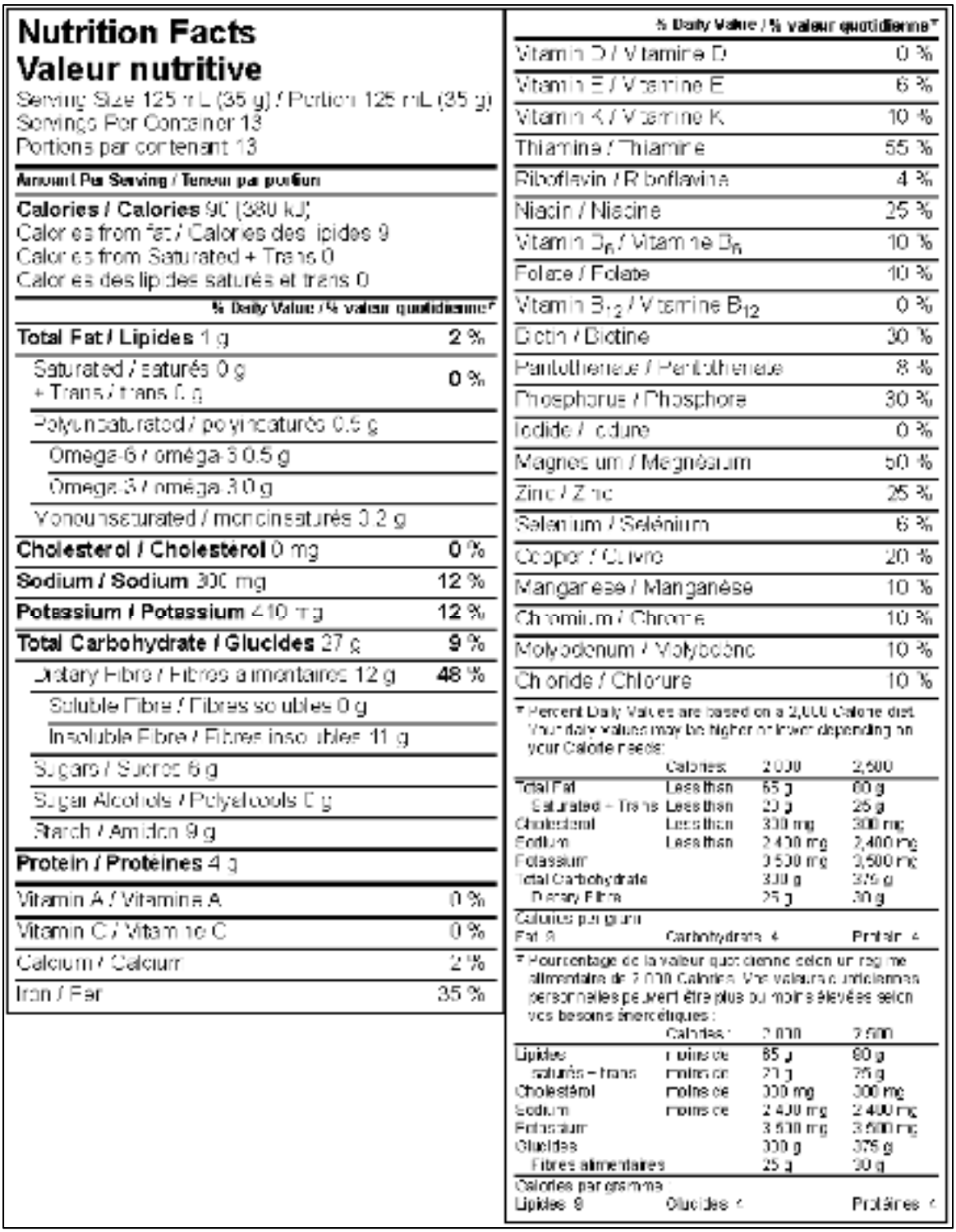
Amount / Teneur	% Daily Value / % valeur quotidienne
Calories / Calories 380	
Fat / Lipides 15 g	**23** %
Saturated / saturés 5 g + Trans / trans 0.3 g	**27** %
Cholesterol / Cholestérol 75 mg	
Sodium / Sodium 750 mg	**31** %
Carbohydrate / Glucides 41 g	**14** %
Fibre / Fibres 4 g	**16** %
Sugars / Sucres 3 g	
Protein / Protéines 21 g	
Vitamin A / Vitamine A	8 %
Vitamin C / Vitamine C	10 %
Calcium / Calcium	20 %
Iron / Fer	15 %

Figure 1.1 Boite d'information nutritionnelle (éléments obligatoires)

Lorsque certains éléments obligatoires sont absents ou présents en trop faible quantité, il est alors possible de les exclure du tableau. Il faut alors ajouter une section dans le bas du tableau qui énumère la liste des éléments qui ont été exclus; comme le montre la Figure 1.2.

Figure 1.2 Boite d'information nutritionnelle simplifiée

Un certain nombre d'éléments supplémentaires peuvent être ajoutés au tableau. Il est à noter que seuls les éléments présentés à la Figure 1.3 peuvent se retrouver dans le tableau.



Nutrition Facts
Valeur nutritive

Serving Size 125 mL (35 g) / Portion 125 mL (35 g)
Servings Per Container 13
Portions par contenant 13

Amount Per Serving / Teneur par portion	
Calories / Calories 90 (380 kJ)	
Calories from fat / Calories des lipides 9	
Calories from Saturated + Trans 0	
Calories des lipides saturés et trans 0	
	% Daily Value / % valeur quotidienne*
Total Fat / Lipides 1 g	2 %
Saturated / saturés 0 g + Trans / trans 0 g	0 %
Polyunsaturated / polyinsaturés 0.5 g	
Omega-6 / oméga 6 0.5 g	
Omega-3 / oméga 3 0 g	
Monounsaturated / monoinsaturés 0.2 g	
Cholesterol / Cholestérol 0 mg	0 %
Sodium / Sodium 300 mg	12 %
Potassium / Potassium 410 mg	12 %
Total Carbohydrate / Glucides 27 g	9 %
Dietary Fibre / Fibres alimentaires 12 g	48 %
Soluble Fibre / Fibres solubles 0 g	
Insoluble Fibre / Fibres insolubles 11 g	
Sugars / Sucres 6 g	
Sugar Alcohols / Polyalcools 0 g	
Starch / Amidon 9 g	
Protein / Protéines 4 g	
Vitamin A / Vitamine A	0 %
Vitamin C / Vitamine C	0 %
Calcium / Calcium	2 %
Iron / Fer	35 %

	% Daily Value / % valeur quotidienne*
Vitamin D / Vitamine D	0 %
Vitamin E / Vitamine E	6 %
Vitamin K / Vitamine K	10 %
Thiamine / Thiamine	55 %
Riboflavin / Riboflavine	4 %
Niacin / Niacine	25 %
Vitamin B_6 / Vitamine B_6	10 %
Folace / Folate	10 %
Vitamin B_{12} / Vitamine B_{12}	0 %
Biotin / Biotine	30 %
Pantothenese / Pantothenese	8 %
Phosphorus / Phosphore	30 %
Iodide / Iodure	0 %
Magnesium / Magnésium	50 %
Zinc / Zinc	25 %
Selenium / Sélénium	6 %
Copper / Cuivre	20 %
Manganese / Manganèse	10 %
Chromium / Chrome	10 %
Molybdenum / Molybdène	10 %
Chloride / Chlorure	10 %

* Percent Daily Values are based on a 2,000 Calorie diet. Your daily values may be higher or lower depending on your Calorie needs:

	Calories	2,000	2,500
Total Fat	Less than	65 g	80 g
Saturated + Trans	Less than	20 g	25 g
Cholesterol	Less than	300 mg	300 mg
Sodium	Less than	2,400 mg	2,400 mg
Potassium		3,500 mg	3,500 mg
Total Carbohydrate		300 g	375 g
Dietary Fibre		25 g	30 g

Calories per gram:
Fat 9 Carbohydrate 4 Protein 4

* Pourcentage de la valeur quotidienne selon un régime alimentaire de 2 000 Calories. Vos valeurs quotidiennes personnelles peuvent être plus ou moins élevées selon vos besoins énergétiques :

	Calories	2 000	2 500
Lipides	moins de	65 g	80 g
saturés + trans	moins de	20 g	25 g
Cholestérol	moins de	300 mg	300 mg
Sodium	moins de	2 400 mg	2 400 mg
Potassium		3 500 mg	3 500 mg
Glucides		300 g	375 g
Fibres alimentaires		25 g	30 g

Calories par gramme :
Lipides 9 Glucides 4 Protéines 4

Figure 1.3 Boite d'information nutritionnelle (éléments permis)
Tirée de GÉPA (2003, p. 5-7)

De plus, la présence de certains éléments complémentaires doit être ajoutée en groupe. Par exemple, les acides gras monoinsaturés, oméga-3 et oméga-6 doivent tous être présents dès que l'un d'eux est mentionné. De plus, la présence des acides gras polyinsaturés entraine automatiquement la présence du groupe susmentionné.

L'information de la boite d'information nutritionnelle doit être affichée dans les deux langues officielles du Canada, soit le français et l'anglais. L'information peut soit être contenue dans une boite nutritionnelle bilingue ou deux boites unilingues.

Toute la boite d'information nutritionnelle doit être produite à l'aide d'une seule police de caractère. Le choix de la police de caractères est à la discrétion du fabricant, mais ses caractères doivent être normalisés, sans empattements et sans ornements. **Helvetica** est un très bon exemple d'une police de caractère respectant ces critères. Ce critère facilitera la reconnaissance des caractères, car il assurera une meilleure segmentation des mots et des caractères.

Bien qu'il existe plusieurs modèles de boites nutritionnelles, ils possèdent chacun leur gabarit et sont tous répertoriés afin de s'assurer de la standardisation de la présentation finale. En voici quelques exemples : modèle standard bilingue (Figure 1.4), modèle double bilingue (Figure 1.5) et modèle horizontal bilingue (Figure 1.6).

Nutrition Facts
Valeur nutritive

Per 1 bowl (300 g) / Pour 1 bol (300 g)

Amount Teneur	% Daily Value % valeur quotidienne
Calories / Calories 380	
Fat / Lipides 15 g	**23** %
Saturated / saturés 5 g + Trans / trans 0.3 g	**27** %
Cholesterol / Cholestérol 75 mg	
Sodium / Sodium 750 mg	**31** %
Carbohydrate / Glucides 41 g	**14** %
Fibre / Fibres 4 g	**16** %
Sugars / Sucres 3 g	
Protein / Protéines 21 g	
Vitamin A / Vitamine A	8 %
Vitamin C / Vitamine C	10 %
Calcium / Calcium	20 %
Iron / Fer	15 %

Figure 1.4 Modèle standard bilingue

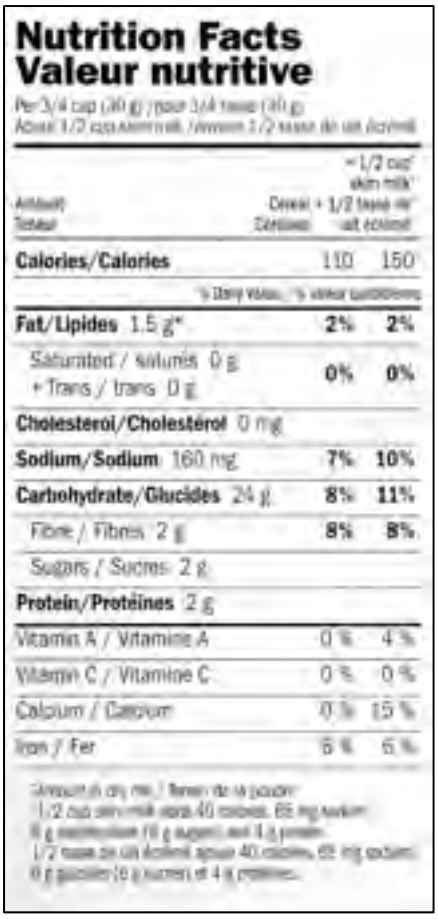


Figure 1.5 Modèle double bilingue

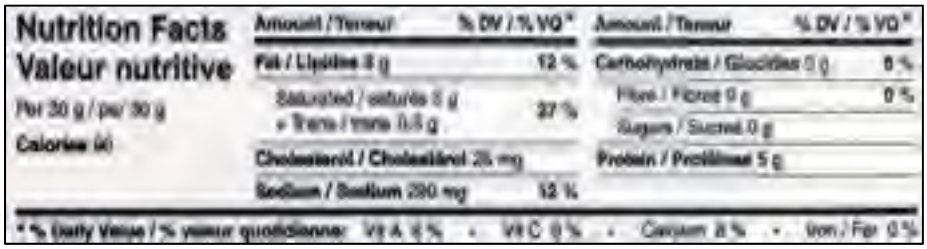


Figure 1.6 Modèle horizontal bilingue

Afin d'assurer une bonne lisibilité de l'information, plusieurs règles ont été établies quant à la dimension des caractères, des interlignes, des filets et des retraits selon chacun des

modèles de boite d'information. De plus, les caractères et les filets doivent obligatoirement reposer sur un fond uniforme et quasi blanc (un maximum de 5 % d'aplat) (Voir Figure 1.7).

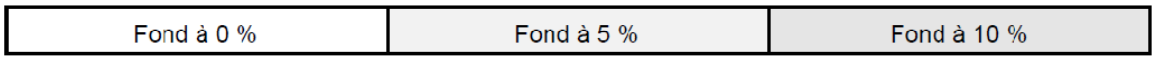


Figure 1.7 Contraste exigé dans l'affichage (en % d'aplat)

1.2 Recherche opérationnelle

La recherche opérationnelle est un domaine d'étude qui aide à la prise de décision. Les différentes méthodes qui la composent permettent d'obtenir la solution la plus optimale selon le type de problème et les contraintes qui lui sont rattachées (dépendant du problème rencontré, il n'est pas toujours possible d'atteindre une solution optimale). Que ce soit pour minimiser le temps de déplacement dans les transports en commun (à l'aide par exemple de l'application « tous azimuts »; développée grâce à une entente entre la société des transports de Montréal et le groupe MADITUC de l'École Polytechnique de Montréal : http://www.stcum.qc.ca/azimuts/index.htm) ou encore pour maximiser la rentabilité de fabrication à partir d'une même quantité de matière brute (par exemple, tenter d'obtenir les meilleures coupes de bois à partir d'un seul arbre). Certains problèmes pourraient très bien être résolus sans l'aide de la recherche opérationnelle; cependant, cette dernière permet de minimiser le temps nécessaire pour obtenir une solution optimale.

1.2.1 Programmation dynamique

La programmation dynamique est une méthode générale de résolution de problème de la recherche opérationnelle dans le cas où une série de décisions sont interreliées. Cette méthode permet de trouver la séquence optimale sans pour autant avoir à tester toutes les combinaisons possibles. Pour y parvenir, cette méthode utilise l'adage : diviser pour régner. Au lieu de tenter de résoudre le problème dans son entier, la programmation dynamique tentera plutôt de trouver la solution optimale pour un sous-problème. Par la suite, elle

augmentera légèrement la dimension du sous-problème afin de trouver la solution optimale. La méthode sera ainsi itérée jusqu'à l'obtention de la solution optimale finale. De façon générale, cette méthode est appliquée à reculons, c.-à-d. de la destination jusqu'à l'origine. Regardons un exemple afin de faciliter la compréhension. L'exemple que nous étudierons est fortement inspiré de (Hillier, 1996). Il est à noter que ce problème aurait pu être résolu par d'autre méthode de la recherche opérationnelle, soit par exemple, la méthode du chemin le plus court.

Soit un voyageur qui désire se rendre en autobus de la ville A à la ville G. Malheureusement aucun autobus n'effectue le trajet d'un seul coup. Il devra donc prendre plusieurs autobus pour se rendre à destination, mais celui-ci désire minimiser ces coûts de transport. Le schéma ci-dessous représente la carte des transports (les bulles représentent les villes, les lignes représentent les trajets d'autobus et les nombres représentent les coûts pour prendre ledit autobus)

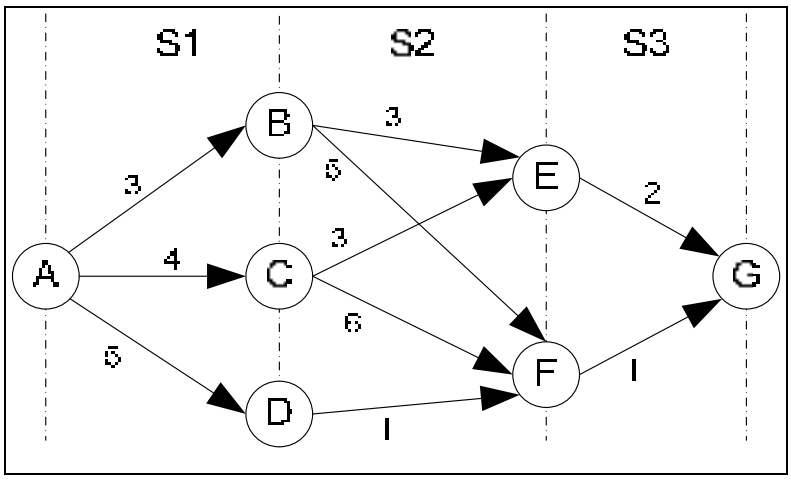


Figure 1.8 Possibilité de transport entre les différentes villes
Adaptée de Hillier (1996, p. 426)

Le voyage s'effectue en trois segments (S1, S2, S3). Comme nous en avons discuté précédemment, nous commencerons par S3 (le dernier segment). Le voyageur arrive à la ville G en provenance de la ville E ou F.

Tableau 1.1 Coût du segment S3

Source	**Coût**	**Dest.**
E	2	G
F	1	G

Par la suite, nous évaluons les coûts pour les deux derniers segments (S2, S3). Le voyageur peut donc provenir des villes B, C ou D et se diriger vers E ou F. À partir de ces destinations, nous utilisons les solutions optimales obtenues précédemment.

Tableau 1.2 Coût du segment S2

Source	**$V2=C+V3_{opt}$**		**Dest.**
	E V3=2	F V3=1	
B	5	6	E→G
C	5	7	E→G
D	X	2	F→G

Pour finir, nous considérons tous les segments (S1, S2, S3). Le voyageur provient de A et se dirige vers B, C ou D. Encore une fois, à partir de ces destinations, nous utilisons les solutions optimales obtenues précédemment.

Tableau 1.3 Coût du segment S1

Source	**$V1=C+V2_{opt}$**			**Dest.**
	B	C	D	
A	8	9	7	D→F→G

Maintenant nous obtenons le voyage optimal A→D→F→G pour un coût de 7.

1.3 Reconnaissance optique de caractère

Bien que la reconnaissance optique de caractère (tiré de l'anglais « Optical Character Recognition » OCR) soit souvent présentée comme une seule étape, il est possible d'en

séparer le processus en quatre grandes étapes : le prétraitement, qui inclut toutes les opérations qui permettront d'améliorer la qualité de l'image et d'en faciliter le traitement ; l'extraction des caractéristiques, qui consiste à récupérer les informations clés relatives au texte ; par la suite, il est possible de classifier ces caractéristiques afin de déterminer le ou les caractères auxquels ils s'apparentent le plus ; et finalement, en jumelant ensemble les caractères hypothétiques obtenus il est possible de reconstruire des mots puis des phrases. Ces quatre étapes seront donc étudiées de façon indépendante. Il est à noter qu'il est impossible dans le cadre de ce travail de passer en revue ce domaine de façon exhaustive. Nous désirons plutôt présenter au lecteur l'information pertinente au projet ou encore l'information qui l'aidera à situer certaines approches par rapport aux autres méthodes connexes.

1.3.1 Prétraitement

Le prétraitement peut être constitué de plusieurs opérations selon les besoins. Il en existe cependant trois grandes catégories qui s'appliquent à notre problématique : le seuillage; la réduction du bruit; et la correction de l'inclinaison (« skew correction »). Le seuillage permet de passer d'une image couleur ou en tons de gris à une image binaire, beaucoup plus facile à traiter et qui ne conserve que les objets d'intérêt. La réduction du bruit permet de retirer les artéfacts insérés involontairement par la méthode d'acquisition ou par la qualité de la source. La correction de l'inclinaison permet de gérer la courbure dans les documents causés par la méthode d'acquisition. (Bombage à proximité d'une reliure, acquisition inclinée, déformations géométriques, etc.)

1.3.1.1 Seuillage

Le seuillage est un processus qui tente de différencier, généralement de façon binaire, l'arrière-plan (dans la problématique qui nous concerne, il s'agit de la feuille) de l'avant-plan (dans notre contexte, le texte qui nous intéresse). Le seuillage peut être associé à la

segmentation de ligne et de caractère, mais nous avons préféré le présenter ici afin de concorder avec l'ordre d'utilisation dans l'application. Une multitude de méthodes de seuillage ont été développées au cours des années. Sezgin et Sankur ont testé quarante méthodes de seuillage (Sezgin et Sankur, 2004). Nous ne présentons pas ici, toutes ces méthodes, mais bien celles qui nous semblent les plus prometteuses. Les méthodes de seuillage peuvent généralement être séparées en deux grandes catégories. Les méthodes dites globales, qui utilisent de l'information présente dans toute l'image afin de déterminer une valeur seuil qui sera ensuite utilisée pour décider si un pixel donné est classé comme arrière-plan ou comme avant-plan. Les méthodes locales, quant à elle, peuvent utiliser des approches similaires aux méthodes globales, mais elles n'utilisent cependant qu'une quantité d'informations limitées qui se trouvent au voisinage du pixel ou de la zone de pixels à classifier. Il faut cependant faire attention, car il existe certaines méthodes qui sont reliées aux deux groupes. Par exemple, la méthode d'Abutaleb (Abutaleb, 1989) est une méthode globale qui utilise de l'information locale afin d'améliorer la prise de décision. Les méthodes globales obtiennent généralement des résultats moins performants que les méthodes locales, mais elles nécessitent un temps de calcul considérablement inférieur. C'est pourquoi les revues sur l'état de l'art (Trier et Taxt, 1995) comparent les méthodes dans leur catégorie respective.

Dans les méthodes globales, la méthode d'Otsu (Otsu, 1979) semble toujours faire ses preuves. Elle est considérée par certains comme meilleur et la plus rapide méthode globale (Cheriet *et al.*, 2007; Trier et Taxt, 1995). Elle est simple, demande peu de temps de calcul et elle est tout de même très efficace; on la retrouve d'ailleurs composante de plusieurs méthodes locales ou encore dans certaines applications comme fonction de binarisation, par exemple dans Matlab. Cette méthode pose l'hypothèse que l'histogramme de l'image est une distribution bimodale composée de deux classes : l'arrière-plan et l'avant-plan. Elle détermine ensuite la valeur seuil optimale en minimisant l'erreur de classification d'un pixel dans la mauvaise classe. Cette méthode donne des résultats impeccables lorsque l'histogramme est bel et bien bimodal et que les deux classes ne s'entrecroisent pas.

Cependant, l'efficacité de cette méthode diminue lorsque nous nous éloignons d'au moins un de ces deux critères; ce qui est souvent le cas. Afin de pallier au problème de la distribution bimodale, (Cheriet, Said et Suen, 1998) proposent une méthode récursive de l'approche d'Otsu qui permet d'effectuer un seuillage multiple. Il est à noter que l'image résultante n'est cependant pas binaire si nous conservons les résultats de chacun des seuillages. Cependant, en ne conservant que le dernier seuillage, il est possible d'extraire le texte d'une image si celui-ci fait parti des éléments les plus sombres de l'image. Grâce au seuillage multiple, il est possible d'extraire le texte, malgré un arrière-plan complexe.

Pour les méthodes locales, nous présentons ici cinq méthodes. Premièrement, la méthode de Bernsen (présentée par (Trier et Jain, 1995)) utilise une fenêtre de voisinage centré sur le pixel à seuiller. Elle détermine la valeur minimum et maximum de ce voisinage, puis à l'aide de l'équation (1.1) elle détermine la valeur seuil à utiliser.

$$T = \frac{V_{max} + V_{min}}{2} \tag{1.1}$$

V_{max} :Valeur maximum dans le voisinage
V_{min}:Valeur minimum dans le voisinage
T:Valeur de seuillage

Si la valeur du pixel est inférieure à la valeur seuil, celui-ci est considéré comme de l'avant-plan, sinon il est considéré comme de l'arrière-plan. Dans l'éventualité où l'écart entre les valeurs minimum et maximum est trop faible (inférieur à une valeur déterminée par expérimentation), le pixel est attribué à l'arrière-plan.

Deuxièmement, la méthode de Eikvil et al. (présenté par (Trier et Jain, 1995)) utilise deux fenêtres. Le seuillage est effectué sur les pixels contenus dans une première fenêtre de petite taille à partir de l'information d'une plus grande fenêtre qui est centrée sur la première. Une valeur seuil par la méthode d'Otsu (Otsu, 1979) est calculée sur la grande fenêtre. À l'aide des

moyennes de classe estimées ($\hat{\mu}_1$ et $\hat{\mu}_2$), une vérification est faite afin de s'assurer que les deux classes sont suffisamment séparées pour être considérées distinctes. Si ce n'est pas le cas, tous les pixels de la petite fenêtre sont attribués à l'arrière-plan; sinon, chacun des pixels est seuillé à l'aide du seuil obtenu par la méthode d'Otsu. Les fenêtres sont ensuite déplacées d'une distance égale à la largeur de la petite fenêtre.

Contrairement aux autres méthodes présentées ici, la troisième méthode, celle de Chow et Kaneko (Chow et Kaneko, 1972) calcule une surface de seuillage. Pour ce faire, elle divise l'image en zone adjacente. Pour chacune de ces zones, elle détermine une valeur seuil en approximant l'histogramme à l'aide d'une mixture de deux distributions gaussiennes (par exemple, à l'aide de la méthode d'Otsu). Une fois toutes les valeurs seuils déterminées, celles-ci sont lissées, puis la surface de seuillage est générée à l'aide d'une interpolation bilinéaire des valeurs seuils obtenues. Les pixels peuvent ainsi être seuillés selon la valeur de la surface correspondant à leurs propres coordonnées.

Finalement, nous présentons la quatrième et la cinquième méthode, qui sont en fait deux méthodes similaires. La méthode de Niblack (présentée par (Trier et Jain, 1995)) détermine un seuil en fonction de la moyenne et de l'écart-type d'une fenêtre environnant le pixel à évaluer.

$$\text{Seuil}(x,y) = m(x,y) + k * s(x,y) \quad (1.2)$$

$k_{\text{suggéré}}$: -0.2

m(x,y):moyenne des valeurs de la fenêtre centrée en (x,y)

s(x,y):écart-type des valeurs de la fenêtre centrée en (x,y)

Seuil(x,y):valeur de seuillage pour le pixel situé en (x,y)

La méthode de Sauvola (Sauvola et Pietikäinen, 2000), qui est une version améliorée de la méthode de Niblack (présentée par (Trier et Jain, 1995)), détermine l'appartenance d'un pixel à l'arrière-plan ou à l'avant-plan en comparant la valeur du pixel avec un seuil local

déterminé en fonction de la moyenne et de l'écart-type du voisinage du pixel; selon l'équation suivante :

$$Seuil(x,y) = m(x,y)\left[1 + k\left(\frac{s(x,y)}{R} - 1\right)\right] \quad (1.3)$$

$k_{suggéré}$: [0.2 , 0.5]

R :128 pour une image en ton de gris quantifiée sur 8 bits

m(x,y):moyenne des valeurs de la fenêtre centrée en (x,y)

s(x,y):écart-type des valeurs de la fenêtre centrée en (x,y)

Seuil(x,y):valeur de seuillage pour le pixel situé en (x,y)

Le seuil du pixel (x,y) est fonction de la moyenne (m) d'une fenêtre de largeur fixe (w) de pixel centrée en (x,y), de l'écart-type (s) de cette même fenêtre, de R qui correspond à la valeur maximum de l'écart-type (R=128 pour une image en ton de gris quantifiée sur 8 bits) et k qui est un paramètre défini entre 0.2 et 0.5. Il faut donc calculer une moyenne locale et un écart-type local pour chacun des pixels de l'image. Cette méthode est très efficace sur les images dont l'éclairage n'est pas uniforme ou encore qui possèdent un arrière-plan complexe, mais elle nécessite une capacité de calcul de beaucoup supérieur aux méthodes globales (Cheriet *et al.*, 2007).

Ces deux types de méthodes, globale et locale, nous laissent malheureusement un choix difficile : la rapidité d'exécution ou l'exactitude du résultat. Dans une multitude de situations, il serait en fait nécessaire d'obtenir les deux, soit l'efficacité de la méthode locale avec le temps d'exécution de la méthode globale. Une nouvelle approche (Shafait, Keysers et Breuel, 2008) permet d'obtenir l'efficacité des méthodes locales pour un temps de calcul se rapprochant de celui des méthodes globales. Ils sont arrivés à ce résultat en utilisant les tables de sommation (Crow, 1984), popularisé en vision informatique par (Viola et Jones, 2004). Cette approche permet de diminuer de façon notable le temps de calcul des moyennes et des écarts-types locaux. Pour ce faire, nous devons tout d'abord générer l'image intégrale (I) (« integral image »). La valeur de l'image intégrale à un pixel donné (x,y) est égale à la

somme de tous les pixels qui se trouvent à gauche et au-dessus dans l'image d'origine (g) (incluant le pixel à la position (x,y)).

$$I(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{y} g(i,j) \tag{1.4}$$

g(i,j):Pixel (i,j) de l'image d'origine

I(x,y):Pixel (x,y) de l'image intégrale

Nous pouvons facilement répéter ce processus pour obtenir l'image intégrale du carré des pixels de l'image d'origine.

$$I_2(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{y} g^2(i,j) \tag{1.5}$$

$g^2(i,j)$:Le carré du pixel (i,j) de l'image d'origine

$I_2(x,y)$:Pixel (x,y) de l'image intégrale

Une fois que nous avons obtenu nos deux images intégrales, nous pouvons rapidement calculer la moyenne et l'écart-type d'une fenêtre carré de largeur (w) centrée en (x,y).

$$m(x,y) = \begin{pmatrix} I(x+w/2, y+w/2) + I(x-w/2, y-w/2) \\ -I(x+w/2, y-w/2) - I(x-w/2, y+w/2) \end{pmatrix} / w^2 \tag{1.6}$$

w: *Largeur de la fenêtre*

(x,y): *centre de la fenêtre*

I: *Image intégrale*

m: *moyenne des valeurs de la fenêtre*

$$s^2(x,y)=\frac{1}{w^2}\sum_{i=x-w/2}^{x+w/2}\sum_{j=y-w/2}^{y+w/2}g^2(i,j)-m^2(x,y)$$

w: *Largeur de la fenêtre*

(x,y): *centre de la fenêtre*

$g^2(i,j)$:Le carré du pixel (i,j) de l'image d'origine

m^2: *moyenne au carré des valeurs de la fenêtre*

$s^2(x,y)$: *Variance de la fenêtre centrée en* (x,y) (1.7)

Tel que:

$$s^2(x,y)=h(x,y)-m^2(x,y)$$

(x,y): *centre de la fenêtre*

m^2: *moyenne au carré des valeurs de la fenêtre*

h: *moyenne du carré des pixels de la fenêtre*

s^2: *variance de la fenêtre* (1.8)

$$h(x,y)=\frac{1}{w^2}\sum_{i=x-w/2}^{x+w/2}\sum_{j=y-w/2}^{y+w/2}g^2(i,j)$$

w: *Largeur de la fenêtre*

(x,y): *centre de la fenêtre*

$g^2(i,j)$: *le carré du pixel*(i,j) *de l'image d'origine*

h: *moyenne du carré des pixel de la fenêtre* (1.9)

$$h(x,y)=\begin{pmatrix}I_2(x+w/2,y+w/2)+I_2(x-w/2,y-w/2)\\-I_2(x+w/2,y-w/2)-I_2(x-w/2,y+w/2)\end{pmatrix}/w^2$$

w: *Largeur de la fenêtre*

(x,y): *centre de la fenêtre*

$I_2(x,y)$:Pixel (x,y) de l'image intégrale

h: *moyenne du carré des pixel de la fenêtre* (1.10)

Les équations 1.4 à 1.8, considèrent que la fenêtre utilisée est carrée et de largeur (w). Il serait cependant assez simple d'adapter ces équations afin d'utiliser des fenêtres rectangulaires.

Cette approche rend la complexité du calcul indépendante de la dimension de la fenêtre choisie afin de calculer la moyenne et l'écart-type local. De plus, elle améliore considérablement le temps de traitement de la méthode de Saulova (Shafait, Keysers et Breuel, 2008) .

1.3.1.2 Réduction du bruit

Le bruit est un problème inhérent à presque tous les systèmes d'acquisition d'image. Il peut provenir de l'acquisition elle-même, de la conversion analogue/numérique ou de la compression de l'image. Des artéfacts peuvent aussi avoir été créés suite à certaines opérations. La réduction du bruit peut donc avoir lieu à deux moments distincts; soit entre l'acquisition et le seuillage afin d'uniformiser l'image pour en faciliter le traitement, soit après le seuillage afin de retirer les artéfacts qui auraient pu avoir été créés durant celui-ci. Il existe trois grandes méthodes afin de réduire le bruit dans les images en tons de gris : le filtrage par l'application de masques, le filtrage par opérateurs morphologiques et le filtrage dans le domaine fréquentiel. Ces méthodes peuvent aussi être appliquées, dans une certaine mesure, aux images binaires. Mais il est souvent plus approprié, avec ces dernières, de favoriser les opérations morphologiques. Les paragraphes suivants présentent les méthodes les plus populaires.

Application de masques

Le filtrage par l'application de masques permet, en appliquant une règle (linéaire ou non) basée sur le voisinage d'un pixel, de déterminer la nouvelle valeur de celui-ci. Ce processus peut ainsi être répété pour chacun des pixels de l'image. Le voisinage est habituellement

carré, centré autour du pixel cible, de dimensions impaires et de taille réduite (fenêtre de 3x3, 5x5, 7x7).

W(-1,1)	W(0,1)	W(1,1)
W(-1,0)	W(0,0)	W(1,0)
W(-1,-1)	W(0,-1)	W(1,-1)

Figure 1.9 Masque moyennant
(concept)

Les masques linéaires utilisent des poids W(x,y) afin de déterminer l'importance de chacun des pixels dans le calcul de la valeur de sortie. Si la somme de tous les poids d'un masque est égale à 1, ce masque est dit normalisé. Ce qui signifie que l'intensité globale de l'image ne sera pas altérée par l'application de ce masque. Par exemple, un masque moyennant normalisé 3x3 ressemblerait à ceci :

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 1.10 Masque moyennant uniforme
Adaptée de Cheriet *et al.* (2007, p. 30)

Ceci signifie que la valeur du pixel dans l'image de sortie à la position centrale sera égale à la moyenne de celui dans l'image d'entrée et des 8 pixels qui l'entourent. Tous les calculs utilisent les valeurs des pixels de l'image originale pour ensuite être sauvegardés dans l'image de sortie.

Il est à noter que nous pouvons aussi utiliser une moyenne pondérée non uniforme afin de donner des importances différentes à chacun des pixels du voisinage. Dans tous les cas, le masque moyennant va permettre de lisser l'image (ce qui diminuera le bruit), mais il fera aussi disparaitre les petits détails de l'image.

Les masques non linéaires utilisent eux aussi le concept de voisinage, mais ils l'exploitent différemment. Par exemple, le voisinage peut être utilisé comme une liste, de laquelle on extrait la valeur médiane, la valeur maximale, la valeur minimale, le mode, etc. Ces différents filtres sont efficaces contre certains types de bruits uniquement. Par exemple, l'utilisation d'un filtre maximum est très efficace contre le bruit poivre (valeurs faibles) mais très peu efficace contre le bruit sel (valeurs élevées), par opposition le filtre minimum est très efficace contre le bruit sel, mais très peu contre le bruit poivre. À l'aide d'un filtre médian, puisque nous sélectionnons la valeur centrale, nous pouvons éliminer les bruits sel et poivre.

Filtrage fréquentiel

Le filtrage dans le domaine fréquentiel permet de traiter le bruit de nature cyclique en prenant une approche différente. Pour ce faire, nous devons transposer notre image dans le domaine fréquentiel, effectuer les différentes manipulations désirées et finalement ramener l'image résultante dans le domaine spatial. Considérant qu'une image n'est pas une fonction périodique, il ne nous est pas possible d'utiliser les séries de Fourrier. Cependant, une image est une fonction finie dont l'aire sous la courbe est finie, ce qui nous permet d'utiliser la transformée de Fourrier (Gonzalez et Woods, 2002, p. 148). Étant donné qu'une image est une fonction bidimensionnelle discrète, nous devons donc également utiliser la version discrète de la transformée de Fourrier à deux dimensions.

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \tag{1.11}$$

$$x = 0,1,2,\dots,M-1$$

$$y = 0,1,2,\dots,N-1$$

$$M: largeur\ de\ l'image$$

$$N: hauteur\ de\ l'image$$

Nous utilisons bien sûr la transformée inverse correspondante afin de ramener notre image dans le domaine spatial.

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N)} \quad (1.12)$$

$$x = 0,1,2,\dots,M-1$$

$$y = 0,1,2,\dots,N-1$$

$$M: largeur\ de\ l'image$$

$$N: hauteur\ de\ l'image$$

Afin de faciliter les manipulations, nous multiplions la fonction d'entrée par l'équation (1.11) avant d'appliquer la transformation de Fourrier, cela a pour effet de « centrer » la fonction dans le domaine fréquentiel. Ce qui signifie que la composante continue se retrouve au centre et que plus on s'éloigne du centre, plus les fréquences augmentes. Il ne faut cependant pas omettre de multiplier le résultat de la transformée inverse de fourrier par cette même équation.

$$(-1)^{x+y} \quad (1.13)$$

À l'aide des équations présentées ci-dessus, il nous est possible de passer du domaine spatial au domaine fréquentiel et vice et versa.

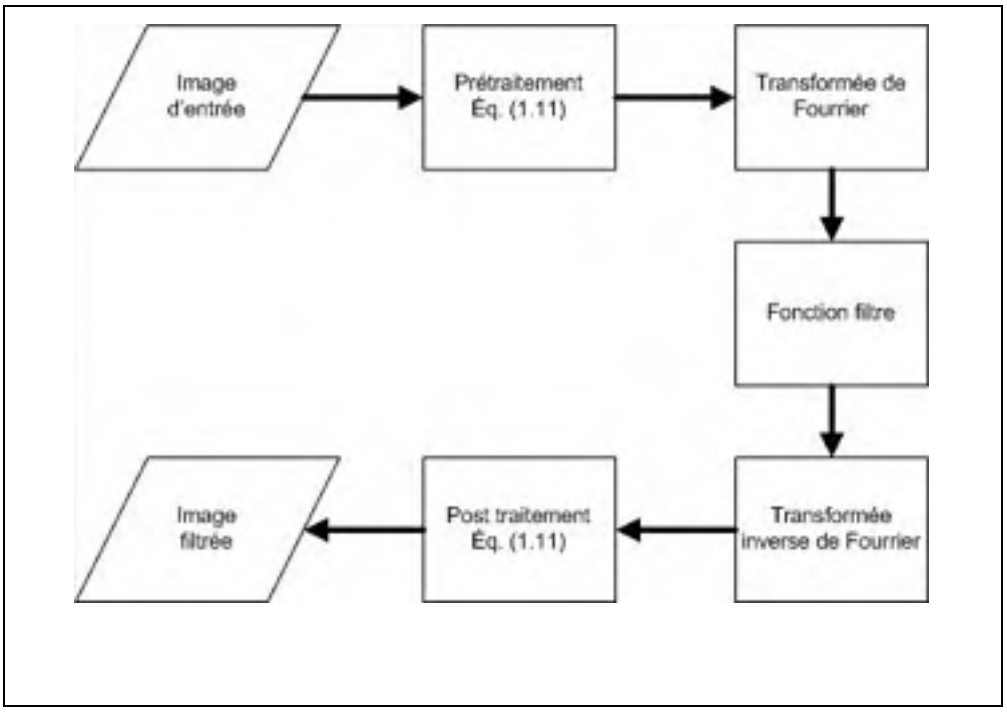


Figure 1.11 Séquence d'opération pour le filtrage fréquentiel

Concentrons-nous maintenant sur la représentation de l'information dans le domaine fréquentiel. Dans ce dernier, nous discernons trois catégories de composantes :

- l'élément de fréquence nulle (composante continue) situé au centre de l'image fréquentielle qui représente la valeur moyenne de l'image spatiale;
- les basses fréquences situées à une faible distance du centre de l'image fréquentielle qui représentent les surfaces douces;
- les hautes fréquences situées loin du centre de l'image fréquentielle qui représentent les détails (arêtes, bruits, pixel isolé, …).

Connaissant ces informations, il nous est possible de cibler l'information qui nous importune et de la supprimer. Dans le cas qui nous concerne, nous désirons principalement cibler les hautes fréquences afin d'éliminer le bruit de notre image. Cela a également pour effet de lisser les contours et d'atténuer les petites aspérités. Pour ce faire, nous utilisons ce que l'on appelle un filtre passe-bas (qui laisse passer les basses fréquences et coupe les hautes

fréquences). Comme nous venons de l'expliquer, les fréquences varient en fonction de la distance par rapport au centre que nous représentons ici par l'équation (1.12).

$$D(u,v) = \left[\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2\right]^{1/2} \tag{1.14}$$

$$M: largeur\ de\ l'image$$

$$N: hauteur\ de\ l'image$$

$$D(u,v): distance\ du\ pixel\ (u,v)\ par\ rapport\ au\ centre\ du\ filtre$$

De plus, la valeur D_0 représente la fréquence de coupure. Nous présentons ici trois différents filtres passe-bas : idéal, Gaussien et Butterworth.

Le filtre idéal est le plus simple, comme le montre l'équation (1.13), si la distance D(u,v) est inférieure ou égale à la fréquence de coupure D_0 la valeur de F(u,v) sera conservée, sinon elle sera réduite à zéro.

$$H(u,v) = \begin{cases} 1\ si\ D(u,v) \leq D_0 \\ 0\ si\ D(u,v) > D_0 \end{cases} \tag{1.15}$$

$$D(u,v): distance\ du\ pixel\ (u,v)\ par\ rapport\ au\ centre\ du\ filtre$$

$$D_0: distance\ de\ coupure, représantant\ la\ fréquence\ de\ coupure$$

Malheureusement, le fait d'utiliser une coupure de fréquence aussi nette introduit un effet de halo dans l'image spatiale connu sous le nom de réverbération.

Figure 1.12 Réverbération
Tirée de Gonzalez et Woods (2002, p. 172)

En utilisant le filtre Gaussien (1.14) nous supprimons complètement l'effet de réverbération. Cependant, nous perdons beaucoup de précision sur le contrôle de la fréquence de coupure D_0.

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$
$$D(u,v): distance\ du\ pixel\ (u,v)\ par\ rapport\ au\ centre\ du\ filtre \tag{1.16}$$
$$D_0: distance\ de\ coupure, représantant\ la\ fréquence\ de\ coupure$$

Si nous désirons un compromis entre le filtre idéal et le filtre gaussien, nous pouvons utiliser le filtre Butterworth (1.15). L'ordre du filtre Butterworth (n) permet de faire varier le comportement du filtre pour qu'il agisse plus comme un filtre idéal ou plus comme un filtre Gaussien. L'utilisation d'un ordre égale à 1 donne un filtre avec des effets similaires à ceux du filtre Gaussien tandis que si nous utilisons un ordre élevé les effets du filtre se rapprochent de ceux d'un filtre idéal.

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$
$$D(u,v): distance\ du\ pixel\ (u,v)\ par\ rapport\ au\ centre\ du\ filtre \tag{1.17}$$
$$D_0: distance\ de\ coupure, représantant\ la\ fréquence\ de\ coupure$$
$$n: ordre\ de\ Butterworth$$

Opérations morphologiques

Les opérations morphologiques utilisent un élément structurant (B) afin de modifier la forme d'un objet (A) dans une image binaire. L'élément structurant ressemble en quelque sorte aux masques que nous avons vus précédemment. Il peut prendre une multitude de formes (carré, rond, losange, ligne, etc.). L'élément structurant possède une origine qui est généralement située en son centre (représenté par un point noir dans la Figure 1.13).

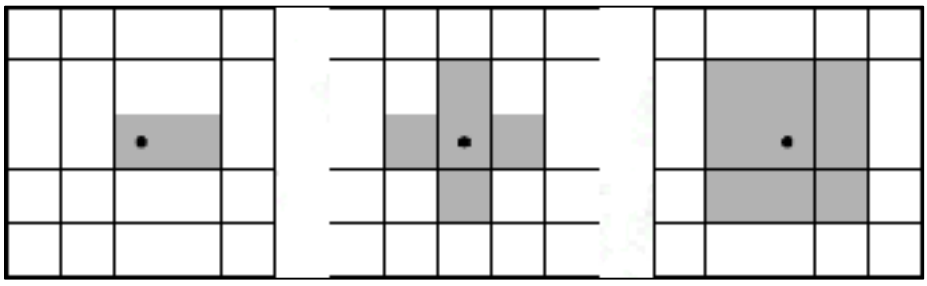


Figure 1.13 Exemple d'éléments structurants
Tirée de Gonzalez et Woods (2002, p. 550)

Il est à noter que nous discutons ici uniquement des opérations qui pourraient nous être utiles. Il y a deux opérations de base que nous pouvons utiliser. La dilatation, qui comme son nom l'indique augmente la surface de l'objet. Elle permet, par le fait même, de combler les petits espaces présents dans l'objet (par exemple : un caractère)(Gonzalez et Woods, 1992, p. 525).

$$A \oplus B = \left\{ Z \middle| \left[\left(\hat{B} \right)_z \cap A \right] \neq \emptyset \right\} \tag{1.18}$$

Par opposition, l'érosion gruge les contours de l'objet ce qui permet entre autres d'uniformiser les contours, de supprimer les pixels isolés et de séparer des objets qui sont faiblement rattachés.

$$\mathrm{A} \ominus B = \{ Z | (B)_z \subseteq A \} \tag{1.19}$$

À partir de l'utilisation alternée de ces deux opérations, deux autres opérations ont été développées : l'ouverture et la fermeture. « L'ouverture permet de supprimer de petites régions, de lisser des contours ou de détacher les objets faiblement liés »(Landry, 2007, p. 6.38). L'ouverture s'effectue en appliquant une érosion suivie d'une dilatation, et ce, en utilisant le même élément structurant pour les deux opérations; à l'exception que la dilatation utilise en fait la réflexion de l'élément structurant afin d'éviter qu'il y ait déplacement des objets dans l'image.

$$A \circ B = (A \ominus \mathrm{B}) \oplus B \tag{1.20}$$

« La fermeture permet de supprimer de petits trous ou de refermer des contours. »(Landry, 2007, p. 6.38). La fermeture s'effectue en appliquant une dilatation suivie d'une érosion. Tout comme pour l'ouverture, l'élément structurant reste le même pour les deux opérations, si ce n'est la réflexion de l'élément structurant lors de la dilatation.

$$A \bullet B = (A \oplus \mathrm{B}) \ominus B \tag{1.21}$$

Malgré le fait que les effets de l'ouverture et de la fermeture ressemblent beaucoup à l'érosion et la dilatation, il est à noter que pour l'ouverture et la fermeture, l'objet garde généralement les mêmes dimensions; contrairement à l'érosion et à la dilatation qui diminue et augmente la taille de l'objet respectivement.

1.3.1.3 Correction de l'inclinaison

L'inclinaison est habituellement causée par une mauvaise orientation entre le document original et le système d'acquisition. Malheureusement, cette situation peut entrainer beaucoup de problèmes au niveau de la segmentation des lignes, des mots et des caractères; ce qui aura des répercussions sur le reste du processus. Pour régler ce problème, il faut tout d'abord déterminer s'il y a, ou non, présence d'une inclinaison. Si c'est le cas, il est requis

d'évaluer l'angle de cette inclinaison avant d'effectuer une rotation de l'image pour l'annuler.

La méthode la plus simple pour y arriver est de projeter l'image selon plusieurs angles. La projection qui aura la plus grande variance sera utilisée comme angle d'inclinaison. La projection est calculée en déterminant plusieurs lignes parallèles selon un angle α, puis en additionnant tous les pixels appartenant à des caractères que chaque ligne traverse. Si l'alignement est correct, certaines lignes auront une très grande quantité de pixels (lignes de textes), tandis que d'autres n'en auront aucun (espace entre les lignes). En contrepartie, si l'alignement est incorrect, les lignes auront une répartition plus uniforme puisqu'elles seront toutes composées de segments de lignes et d'espaces blancs ce qui en diminuera la variance. Une fois l'angle α connu, il suffit d'effectuer une rotation matricielle de l'image.

Certaines applications gèrent la segmentation de ligne en tenant compte de l'inclinaison possible. D'autres vont même jusqu'à segmenter les lignes en tenant compte d'une courbure quadratique (Rice, Nagy et Nartker, 1999) ou même cubique. Cela permet de gérer la courbure dans le document; ce qui serait impossible avec la simple correction de l'inclinaison.

1.3.1.4 Segmentation

La segmentation consiste à isoler les éléments, les paragraphes, les lignes puis finalement les caractères. Dans la présente section, nous traitons uniquement de la segmentation des caractères à l'intérieur d'une ligne qui a déjà été segmentée. Les lignes peuvent être segmentées à l'aide des mêmes méthodes utilisées pour la segmentation des caractères à pas fixes que nous présentons dans la présente section. Il est à noter que l'isolation des caractères est absolument nécessaire afin de pouvoir extraire adéquatement les caractéristiques. Il existe deux grandes catégories de police de caractère, les polices à pas fixe « fixed pitch » et les polices à espacement proportionnel « Proportionnal spacing ». Dans l'exemple ci-dessous

nous remarquons facilement que l'espace horizontal occupé par la lettre « i » est aussi important que celui occupé par les autres lettres dans la police de caractères « Courrier New » alors que ce n'est vraiment pas le cas pour la police de caractère « Times New Roman ».

Tableau 1.4 Types de police de caractère

Police à pas fixe	**Polices à espacement proportionnel**
Courrier New	Times New Roman
POLICE	POLICE

De plus, si le document à analyser est de bonne qualité il est en général assez facile d'isoler chacun des caractères. En contrepartie, lorsque le document est de mauvaise qualité, deux problèmes majeurs peuvent survenir : un même caractère peut être brisé en plusieurs morceaux ou au contraire, plusieurs caractères peuvent être attachés ensemble.

Figure 1.14 Problème de segmentation
Tirée de Lu (1995, p. 68)

Dans l'exemple ci-dessus (Voir Figure 1.14), l'on peut remarquer que les caractères « w », « a », « h », « n » et « o » sont brisés alors que les caractères « as » et « gt » sont attachés ensemble.

Si la situation est idéale, une simple projection verticale suffit afin de segmenter les caractères. Cette méthode consiste à additionner tous les pixels noirs qui se trouvent dans une colonne, et à répéter le processus pour chacune des colonnes de la ligne. Une fois que nous

avons toutes ces valeurs, il est possible de déterminer les espaces entre les caractères en localisant les zones dont les valeurs de projection sont nulles. Nous savons également que les zones dont les valeurs sont très élevées représentent des traits verticaux. Il est aussi possible d'afficher ces résultats sous forme d'histogramme afin de visualiser les zones nulles et les pointes, comme le montre l'exemple ci-dessous (Voir Figure 1.15).

Figure 1.15 Projection verticale
Tirée de Lu (1995, p. 70)

Si la police de caractère utilisée est à pas fixe, il est possible de jumeler à la projection verticale une analyse statistique (Lu *et al.*, 1992) afin de déterminer si la largeur des caractères, l'espace entre les caractères et la distance entre le centre d'un caractère et le centre du caractère suivant sont suffisamment constants. Lorsque l'analyse rencontre une information disparate, il y a de fortes chances qu'il y ait présence d'un caractère brisé, si la largeur du caractère est trop petite, ou de caractères attachés, si la largeur du caractère est trop grande. Afin d'augmenter la robustesse de cette méthode, (Lu, 1995) propose d'appliquer la projection verticale à plusieurs lignes simultanément; il faut bien sûr que toutes ces lignes soient alignées afin que cette méthode soit efficace. L'utilisation de la projection à plusieurs lignes permet de minimiser l'effet des caractères brisés et des caractères attachés dans la projection, mais rend la méthode plus sensible au bruit qui pourrait se retrouver entre deux lignes.

Afin de tenter de réparer les caractères brisés, l'application localise le plus petit espace entre deux caractères (nous utilisons ce terme pour garder la cohérence, car il s'agit en fait de la

distance entre deux parties de caractères) puis tente de supprimer celui-ci. L'analyse est effectuée à nouveau afin de réévaluer la situation pour s'assurer qu'il y a eu amélioration et pour passer au prochain plus petit espace. Si jamais la modification a détérioré le résultat, elle est supprimée.

Pour la séparation des caractères attachés, si nous utilisons toujours une police de caractères à pas fixes, il est possible de localiser les caractères attachés en analysant la largeur des caractères. Par la suite, nous pouvons tenter de déterminer le point de séparation entre les caractères en utilisant la largeur normale de la police ainsi que les creux présents dans la projection verticale.

Malheureusement, le problème se complexifie rapidement lorsque nous passons d'une police de caractère à pas fixes à une police de caractère à espacement proportionnel. Les approches dont nous avons discuté ci-dessus sont toujours applicables, mais il faut cependant considérer que la largeur des caractères ainsi que celle des espaces n'est plus uniforme ce qui rend l'identification des caractères brisés et attachés beaucoup plus difficile. Tout d'abord, il n'est plus possible de traiter plusieurs lignes simultanément considérant que les espaces ne sont plus alignés d'une ligne à l'autre. De plus, considérant qu'il n'y a plus de valeurs fixes pour la largeur des caractères et des espaces inter caractères nous devons confirmer les résultats de la projection verticale à l'aide du classificateur. Donc, une fois la projection effectuée nous effectuons la reconnaissance et lorsque le résultat n'est pas satisfaisant nous pouvons tenter de séparer les caractères trop longs ou d'assembler les caractères trop courts; jusqu'à l'obtention d'un meilleur résultat de reconnaissance.

1.3.2 Extraction des caractéristiques (« Feature extraction »)

Les caractéristiques sont utilisées afin de décrire et ainsi différencier les objets. Par exemple, l'être humain différencie habituellement les fruits par leurs formes, leurs couleurs et leurs textures. Dans le cas des applications automatisées, il est possible d'utiliser des

caractéristiques reconnues par l'homme, car celles-ci sont faciles à conceptualiser. Il est aussi possible de concevoir des caractéristiques pertinentes pour classifier des objets, mais qui ne seraient d'aucune utilité pour l'homme. De plus, comme le choix des caractéristiques est une étape cruciale dans la création d'une application de reconnaissance et que les caractéristiques sélectionnées varieront d'une problématique à l'autre (reconnaissance de caractères, maturité des fruits, identification faciale, …); certaines recherches visent à développer des algorithmes permettant à l'application de choisir elle-même les caractéristiques qui seront utilisées (Kharma, 2004; Siedlecki et Sklansky, 1993)

De plus, certaines caractéristiques sont sensibles à la position, à l'échelle et/ou à l'orientation de l'image, ce qui peut en restreindre l'utilisation selon la situation. Il est donc préférable de favoriser l'utilisation de caractéristiques invariantes. Par exemple, si une caractéristique consiste à compter le nombre de pixels noir dans la fenêtre englobant un caractère, elle varierait en fonction de la grosseur de la fenêtre choisie. En contrepartie, si une caractéristique consistait à trouver la proportion de pixels noirs dans la fenêtre elle deviendrait invariante à l'échelle sans pour autant changer la nature de l'information recueillie.

Il existe un très grand nombre de caractéristiques proposées dans la littérature. Cependant, pour des raisons de temps de calcul, il n'est pas possible, ni souhaitable, d'utiliser la totalité des caractéristiques dont nous disposons. Il est donc nécessaire de sélectionner un nombre restreint de caractéristiques, les plus pertinentes à la classification d'un problème donné. Nous effectuerons donc, ci-dessous, un survol de certaines caractéristiques fréquemment citées dans la littérature.

Étant donné que cette étape survient après la segmentation de l'image d'origine, lorsque nous parlons d'une ou des caractéristiques d'une image nous parlons dans le cas qui nous concerne, des caractéristiques d'un seul caractère.

Les moments sont des caractéristiques décrivant l'ensemble de l'image. Il est possible de différencier deux grandes catégories : les moments géométriques et les autres. En imagerie, les moments géométriques peuvent être représentés par l'équation générale suivante :

$$m_{pq} = \sum_{-\infty}^{+\infty} \sum_{-\infty}^{+\infty} x^p y^q f(x,y) \tag{1.22}$$

p, q : entier dans l'intervalle [0,∞

Ces moments représentent certaines informations physiques telles que la surface, le centre de masse et le moment d'inertie (Cheriet *et al.*, 2007, p. 55). Malheureusement, ces moments ne sont pas orthogonaux; ce qui signifie qu'il est très difficile et coûteux en temps de calcul de reconstruire l'image à partir de ses moments géométriques. C'est pour cela que d'autres moments, ceux-ci orthogonaux, ont été élaborés. Nous pouvons, entre autres, citer les moments de Zernike (Khotanzad et Hong, 1990), de Legendre (présenté par (Cheriet *et al.*, 2007)) et de Tchebichef (Mukundan, Ong et Lee, 2001).

Dans le même ordre d'idée que les moments, il est aussi possible de recueillir plusieurs autres informations géométriques. L'axe majeur et l'axe mineur représentent respectivement la plus grande ligne droite dont les extrémités sont contenues dans l'objet et la plus grande ligne droite, perpendiculaire à l'axe majeur, dont les extrémités sont également contenues dans ce même objet. Ces deux éléments permettent, en général, de constituer un rectangle de surface minimale pouvant englober l'objet.

Un grand nombre de caractéristiques sont basées sur la forme du contour de l'objet. Nous pouvons encoder le contour, en parcourant tous les pixels qui forment le périmètre et en notant les changements de direction. Il est possible de travailler avec uniquement les verticales et les horizontales (connectivité-4) ou encore en incluant les diagonales (connectivité-8 ou -m). Il est de plus, possible de travailler avec un référentiel absolu (le repère de l'image) ou encore selon l'orientation du dernier mouvement.

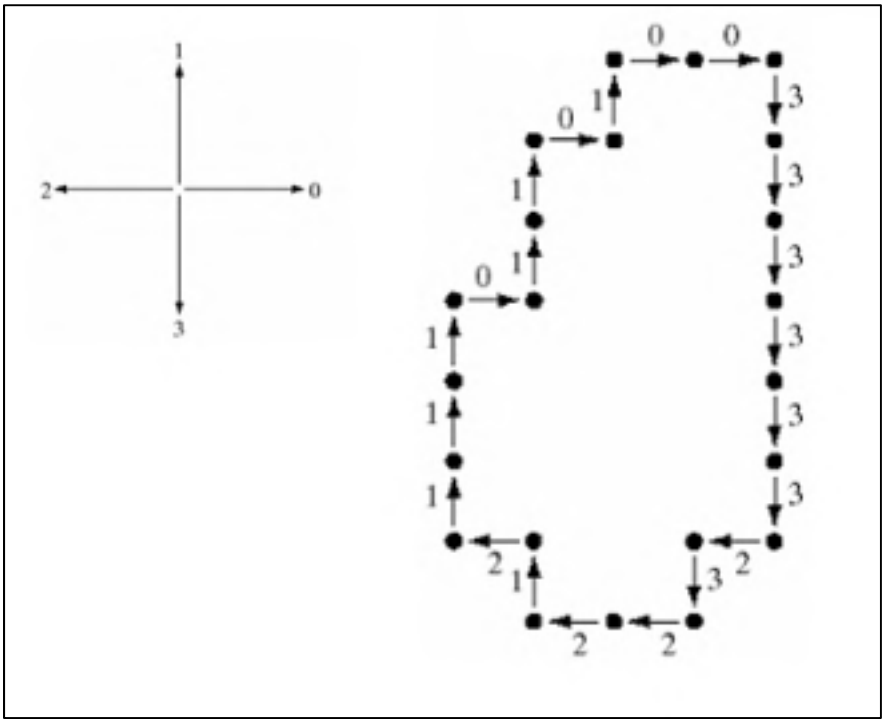


Figure 1.16 Encodage de contour (connectivité-4 absolu)
Adaptée de Gonzalez et Woods (2002, p. 645)

Il est aussi possible de localiser les lignes droites de l'image à l'aide de la transformée de Hough. Cette dernière transforme les coordonnées spatiales (x, y) vers un espace paramétré (ρ, θ) en respectant l'équation suivante :

$$\rho = x \cos\theta + y \sin\theta \tag{1.23}$$

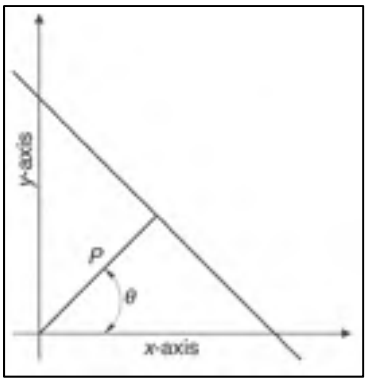


Figure 1.17 Représentation de Hough
Tirée de Cheriet *et al.* (2007, p. 69)

Il est à noter que pour un seul point (x, y) une multitude de correspondances seront présentes dans l'espace paramétré. Cependant, si plusieurs points des coordonnées spatiales obtiennent un même couple (ρ, θ), nous pouvons affirmer qu'il y a présence d'une ligne. La transformée de Hough peut également être utilisée pour localiser des courbes ou autres formes géométriques pouvant être décrites par une équation paramétrique.

Il est également possible d'utiliser les équations de courbes polynomiales, telles que les courbes de Lagrange, de Bézier (Bézier, 1987), les splines de Bézier (Bézier, 1987), les descripteurs génériques avec Bézier (Sohel, Karmakar et Dooley, 2005) et plusieurs autres afin de caractériser les contours de la forme.

La caractérisation d'un contour par courbes polynomiales exige beaucoup de temps de calcul et cette précision n'est pas toujours nécessaire. Nous pouvons simplement utiliser l'approximation polygonale. Celle-ci consiste à représenter la forme à l'aide d'une multitude de segments de droite. Le concept est similaire à l'encodage du contour, mais au lieu de mémoriser uniquement la direction, nous conservons le point initial (x, y), la longueur et l'orientation de chacun des segments.

Les caractéristiques topologiques, très souvent utilisées dans les classificateurs structuraux, fournissent un autre type d'information. Certains points caractéristiques du contour sont recherchés. Il s'agit des extrémités d'arêtes, des embranchements (trois arêtes qui se rejoignent à leur extrémité ou une ligne qui se termine sur une autre) et des croisements (deux arêtes qui se croisent). Il peut également être pertinent d'identifier les boucles ainsi que la courbure des arcs.

1.3.3 Classification

Une fois que les caractéristiques ont été adéquatement choisies, il est maintenant possible de déterminer de quel caractère il s'agit. Le classificateur compare justement les caractéristiques qui ont été extraites avec l'information qu'il a recueillie lors de sa phase d'apprentissage. Cette tâche fort simple pour l'être humain peut s'avérer compliquée pour le classificateur surtout lorsque des éléments similaires peuvent être choisis. Prenons l'exemple de la lettre O et du chiffre 0 qui peuvent même parfois poser problème à l'être humain lorsque le contexte est difficile; dans un numéro de série par exemple.

Avant qu'un classificateur soit fonctionnel, il faut lui fournir des données d'apprentissages. Il existe deux grands types d'apprentissage : l'apprentissage supervisé et le non supervisé. L'apprentissage supervisé consiste à fournir au classificateur plusieurs images de chaque classe présentées séparément (par classe, nous entendons un ensemble d'éléments similaire; par exemple une lettre de l'alphabet.). En contrepartie, l'apprentissage non supervisé consiste à fournir les mêmes images, mais cette fois-ci sans spécifier à quelle classe elles appartiennent. Dans ce dernier cas, le classificateur doit donc lui-même départir les classes selon leurs caractéristiques. Ce type de classificateur est utilisé lorsque nous ne possédons pas l'information à priori ou encore, qu'il est trop onéreux d'étiqueter chacun des éléments (Duda, Hart et Stork, 2000, p. 517). Dans le cas qui nous concerne, nous nous attarderons uniquement à l'apprentissage supervisé.

Nous pouvons considérer qu'il existe trois grandes familles de classificateurs : les classificateurs statistiques, les classificateurs structuraux et les réseaux de neurones. (Duda, Hart et Stork, 2000, p. 7).

Les classificateurs statistiques, qu'ils soient paramétriques ou non paramétriques, sont basés sur la théorie de décision de Bayes. Celle-ci tente de minimiser les mauvaises classifications. Malheureusement, cette théorie nécessite de connaitre toute l'information à priori; ce qui est rarement le cas dans les problèmes concrets. Puisque nous ne possédons pas cette information, nous devons fournir des échantillons d'apprentissages afin d'estimer cette information. Dans le cas des méthodes paramétriques, les échantillons d'apprentissages sont utilisés afin d'estimer les paramètres décrivant la distribution de densité de chaque classe. Il est fréquent que ce genre de classificateur utilise des distributions normales (gaussiennes) car l'estimation des paramètres est simple et plusieurs problèmes réels peuvent être approximés par une distribution normale (Cheriet *et al.*, 2007, p. 132). En utilisant l'approche du maximum de vraisemblance (« maximum likelihood ») il est possible d'estimer les paramètres de chacune des caractéristiques (i) à l'aide des équations suivantes, sachant que l'ensemble d'apprentissage d'une classe est représenté par :

$$\chi = \{x^1, \dots, x^N\} \tag{1.24}$$

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} x^n \tag{1.25}$$

$$\hat{\Sigma}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} (x^n - \hat{\mu}_i)(x^n - \hat{\mu}_i)^T \tag{1.26}$$

Malheureusement, il arrive que les classes ne forment pas des distributions uniformes ou encore qu'il n'y ait pas assez d'échantillons pour évaluer les paramètres. Les méthodes non paramétriques, elles, n'estiment tout simplement pas les distributions de densité des classes;

ce qui contourne ce genre de problème. Bien qu'elles n'estiment pas directement les paramètres, ces méthodes nécessitent tout de même l'utilisation d'échantillons d'apprentissages qui seront plutôt utilisés comme objet de référence. Il existe deux méthodes très répandues : la fenêtre de parzen (« parzen window ») et les K plus proches voisins (« K nearest neighbor » (KNN)). Les deux méthodes observent les objets de référence qui se trouvent à proximité de l'élément à classifier, et l'assigne à la classe possédant le plus d'objets de références dans la zone donnée. Il s'agit en fait d'un vote de majorité selon la distribution spatiale des objets de référence. Les deux méthodes diffèrent sur la sélection des objets de référence. La fenêtre de parzen utilise tous les objets de référence qui se trouvent à l'intérieur d'un volume qui entoure l'élément à classifier, tandis que les K plus proches voisins, comme son nom l'indique, utilisent un nombre fini d'objets qu'elle que soit leur distance par rapport à l'élément à identifier.

Les classificateurs structuraux, comme leur nom l'indique, utilisent la structure qui compose un caractère afin de le classifier. Cette structure est composée de plusieurs primitives (ligne, courbe, boucle, extrémité, jonction, …) qui sont inter reliées les unes aux autres. Ce type de classificateur est beaucoup plus utilisé en reconnaissance de caractères manuscrits en temps réel (online) car il permet de connaître l'ordre de création des traits. Cette structure peut être représentée par une chaine de caractère où chaque « lettre » représente une primitive possédant des attributs. Dans ce cas, il est possible de calculer la distance entre un élément à classifier et un prototype à l'aide de la distance de Levenstein. Cette structure peut également être représentée par un graphe constitué de nœuds (primitives) et d'arêtes (relations entre les primitives). Les nœuds et les arêtes peuvent contenir des caractéristiques afin d'augmenter l'information qu'ils contiennent (voir Figure 1.18). Vu la complexité potentielle de la structure des graphes, il devient beaucoup plus difficile de les comparer les uns aux autres (comparer le graphe à classifier avec les graphes de références). Pour ce faire, deux méthodes sont préconisées : la recherche par arborescence (« tree search ») et la relaxation d'étiquetage (« relaxation labeling »)

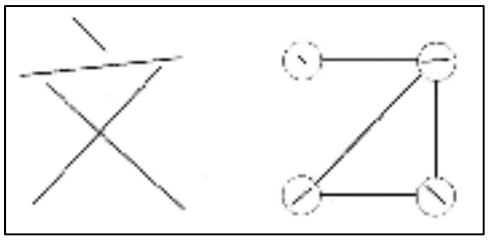


Figure 1.18 Exemple de graphe
Tirée de (Cheriet *et al.*, 2007, p. 175)

L'étude des réseaux de neurones est un vaste domaine de recherche lié à l'intelligence artificielle. Cette approche tente de reproduire le fonctionnement du cerveau humain afin de créer des applications de reconnaissance performantes, tel que des classificateurs. Tout comme le cerveau humain, ils fonctionnent grâce à l'agencement de neurones. Ces neurones artificiels sont en fait constitués de plusieurs entrées auxquelles nous attribuons un poids. La somme de ces entrées, ainsi que la fonction d'activation du neurone, entrainera une certaine réponse à la sortie du neurone.

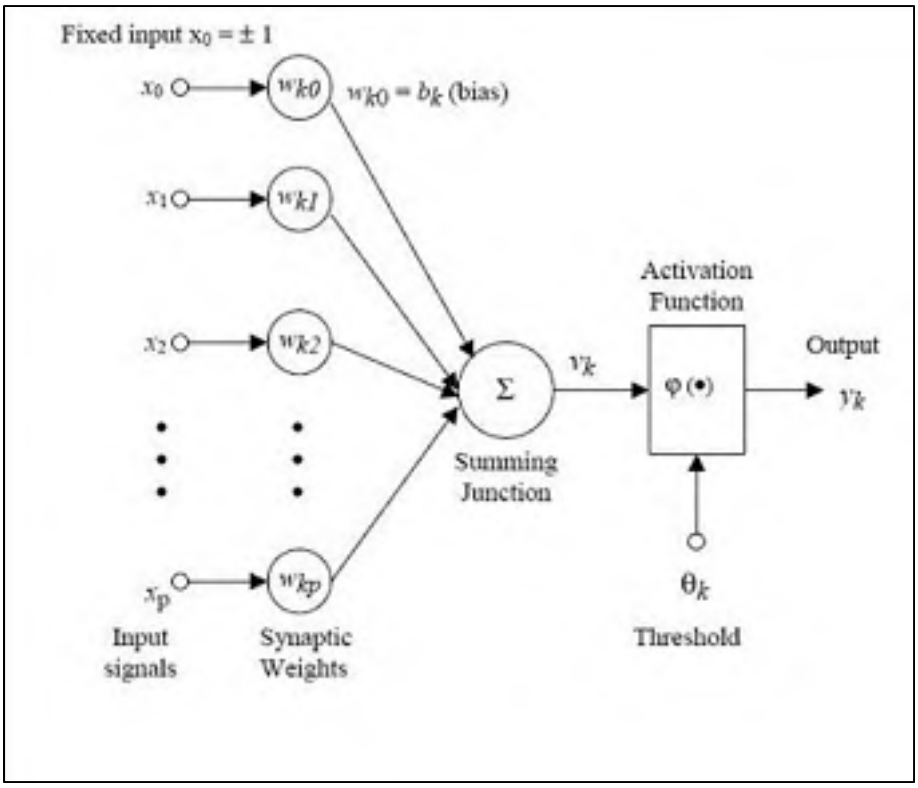


Figure 1.19 Modèle mathématique d'un neurone
Tiré de Artificial neural networks : a neural network tutorial (2010)

Cette réponse peut ensuite être acheminée vers un autre neurone afin d'être utilisée comme entrée ou encore être directement utilisée comme sortie du système. Il est possible de concevoir des réseaux à une, deux ou plusieurs couches (par couche nous entendons généralement une colonne de neurones dont les entrées proviennent de la couche précédente et les sorties sont dirigées vers la couche suivante) (Figure 1.19). Un réseau trop simple aura de la difficulté à classifier des ensembles dont la distribution est complexe et en contrepartie, un réseau trop complexe aura de la difficulté à généraliser pour les échantillons qui n'auront pas été testés lors de l'apprentissage. Afin de pouvoir classifier adéquatement les éléments d'un problème donné, le réseau de neurones doit, tout comme la plupart des autres méthodes, passer par une phase d'apprentissage. Lors de cet apprentissage, les valeurs des poids d'entrées des neurones sont modifiées afin de minimiser l'erreur de classification. Afin d'effectuer cet apprentissage, il existe différentes méthodes selon le type de réseau concerné.

Il est, par exemple, possible d'utiliser pour un réseau à une seule couche, la méthode de la pseudo inverse ou encore la méthode de la descente du gradient; pour les réseaux multicouches, la méthode de la rétropropagation (ou l'une de ces versions modifiées) est généralement utilisée.

1.4 Moteur Tesseract

L'élaboration d'un moteur de reconnaissance de caractère n'est pas une tâche facile. Elle requiert une profonde expertise dans les différents domaines que nous avons survolés précédemment. Considérant que notre objectif est de concevoir un prototype d'application pour extraire de l'information à partir d'une image et non de concevoir le moteur de reconnaissance de caractère lui-même, nous avons décidé de nous tourner vers un moteur déjà existant. Nous avons choisi le projet Tesseract et ce, pour plusieurs raisons. Tout d'abord, il s'agit d'un logiciel libre : cela nous permet donc d'encourager la communauté libre, toute personne qui le désire peut se procurer cette application afin de reproduire nos expériences et de plus, si le besoin s'en fait sentir, nous avons aussi la possibilité de modifier le code source pour l'adapter à nos besoins. Malgré le fait que ce projet soit libre, il est aussi très mature. Celui-ci a vu le jour avec le projet de recherche de Doctorat de M. Ray Smith (Smith, 1987) en 1984 dans les laboratoires de HP à Bristol. Il fut développé sur une période de 10 années, puis présenté à l'« UNLV Annual Test of OCR Accuracy » en 1995 où il termina dans les 3 premiers (Rice, Jenkins et Nartker, 1995). Par la suite, le projet fut abandonné. Finalement, HP donna le projet à la communauté libre en 2005 où son développement reprit de plus belle. Il est maintenant possible de trouver ce projet sur Google Code :

http://code.google.com/p/tesseract-ocr

Il est aussi possible de trouver un excellent résumé des principaux points innovateurs de l'application dans cet article publié par l'auteur original du projet, M. Smith (Smith, 2007) De plus, le projet continu d'apparaitre au sein de la communauté scientifique (Smith, Antonova et Lee, 2009; Unnikrishnan et Smith, 2009).

Originalement, une des faiblesses de l'application était qu'elle n'effectuait aucune analyse de la mise en page des documents (« page layout analysis »). Vu la structure des documents que nous tentons de traiter, l'utilisation ou non de l'analyse de mise en page ne devrait avoir que très peu d'effet sur nos résultats. Il n'en reste pas moins que ce défaut est en voie d'être éliminé dans la version en cours de développement (Smith, 2009).

1.4.1 Méthodes utilisées dans Tesseract

L'utilisation d'une application existante permet de réduire la charge de travail, cependant il peut être difficile de connaitre les méthodes qui sont implémentées dans cette application. Nous pouvons nous référer à l'article (Smith, 2007) qui présente certains aspects du fonctionnement de l'application et heureusement, le code source de l'application Tesseract est disponible sur le site web du projet (*http://code.google.com/p/tesseract-ocr*) tel que mentionné dans l'article (Smith, 2007)

Tout d'abord, l'application considère que l'image d'entrée est binaire et, si ce n'est pas le cas, utilise la méthode d'Otsu sur chacun des canaux conservant celui qui possède le plus de contraste. Par la suite, l'application différentie l'avant-plan de l'arrière-plan en considérant que ce dernier couvre une plus grande surface que l'avant-plan; ce qui permet de traiter aussi bien les textes noirs sur fond blanc que blanc sur fond noir.

L'utilisation d'une analyse en composantes connectées (Connected Components Analysis : CCA) permet de segmenter les différents caractères. Par la suite, à l'aide de la méthode des moindres carrés, l'application tente de faire correspondre une courbe quadratique, la courbe guide, avec les composantes connectées afin d'identifier chaque ligne de texte, même si celles-ci contiennent des déformations.

Ensuite, Tesseract détermine si la police de caractère est à pas fixe ou à espacement proportionnel. Dans le cas où elle est à pas fixe, l'application effectue la segmentation selon le pas identifié; ce qui permet de segmenter chacun des caractères et de localiser les espaces entre les mots. Dans le cas contraire, Tesseract utilise une bande horizontale de faible hauteur suivant la ligne guide afin de tenter d'identifier les espaces inter-caractères et les espaces entre les mots. Cette méthode permet de maximiser les résultats lorsqu'aucun espace n'est visible à l'aide d'une projection verticale.

Une première classification des caractères et des mots est effectuée (dans le cas des polices à pas fixe, la classification ne va pas plus loin). Par la suite, si certains mots ont un taux de confiance trop faible, l'application tente de découper les caractères dont le taux de confiance n'est pas satisfaisant afin de séparer les caractères qui seraient fusionnés. Lorsque les possibilités de découpage ont été épuisées et que le taux de confiance est toujours insuffisant, l'application tente ensuite de réassocier les morceaux de caractères brisés.

Une des particularités de l'application Tesseract est qu'elle n'utilise pas les mêmes caractéristiques durant la phase d'apprentissage que durant la phase de reconnaissance. Lors de la phase d'apprentissage, l'application utilise une approximation polynomiale, tandis que lors de la phase de reconnaissance, l'application utilise plutôt de petits segments de droite de longueur fixes qui sont comparés à plusieurs contre un avec les segments de l'approximation polynomiale. Cette approche permet entre autres d'obtenir de bon résultat lors de la reconnaissance de caractères brisés.

1.5 Approche orientée résultat

Afin de développer une application performante, nous devons tenter d'utiliser les modules les plus efficaces possible. Il est cependant très difficile de déterminer si un module est plus performant qu'un autre. Tout d'abord, nous devons considérer le contexte dans lequel ces modules ont été évalués. Par exemple, les moteurs de reconnaissance de caractères sont généralement évalués en analysant des pages de textes (revues, journaux, publications

scientifiques) alors que notre application nécessite l'analyse de tableaux. Par la suite, nous devons considérer les critères d'évaluation utilisés. Certains modules (généralement des modules de bas niveau, le seuillage en est un bon exemple) sont difficiles à évaluer, il est fréquent de voir dans la littérature des comparaisons (évaluations) de méthodes qui utilisent des critères visuels (fragmentation de caractères ou de lignes en plusieurs segments, disparition d'éléments importants, zones d'ombre, etc.) afin de noter les différentes méthodes. Finalement, il ne faut pas oublier que même si un module est beaucoup plus performant qu'un autre, il est difficile d'évaluer, à première vue, l'impact que celui-ci aura sur le résultat final de l'application.

Donc, afin d'évaluer nos différents modules ainsi que les modifications que nous leurs apporterons, nous avons décidé d'utiliser une approche dite orientée résultat (« Goal-directed evaluation ») (Trier et Jain, 1995). Cette méthode consiste à évaluer l'efficacité d'une méthode en la plaçant préalablement dans le contexte dans lequel nous prévoyons l'utiliser et en l'évaluant grâce aux résultats réels obtenus à l'aide de l'application.

CHAPITRE 2

MÉTHODOLOGIE

Dans ce chapitre, nous présentons la méthode utilisée actuellement pour l'extraction de l'information puis la méthode automatique que nous avons développée. Par la suite, nous étudions les phases du développement de l'application et finalement, nous présentons l'approche d'expérimentation que nous utilisons afin de caractériser et de valider les performances de l'application.

2.1 Méthode manuelle couramment utilisée

Afin de pouvoir adéquatement développer notre application d'extraction d'information, nous devons tout d'abord observer de quelle façon nous effectuons cette tâche présentement, ce qui dans notre contexte consiste en l'extraction manuelle d'information à partir d'une image numérique.

Figure 2.1 Extraction manuelle (boite)

Tout d'abord, nous devons localiser la boite d'information nutritionnelle dans l'image. Nous identifions le format linguistique (Figure 2.1:i) de la boite (français, anglais, français/anglais ou anglais/français). Ensuite, nous extrayons l'information de la quantité de l'échantillon (Figure 2.1:ii). Nous excluons les informations qui, bien qu'à l'intérieur de la boite, nous sont d'aucunes utilités (Figure 2.1:iii,iv). Finalement, nous extrayons les lignes d'informations pertinentes (Figure 2.1:v).

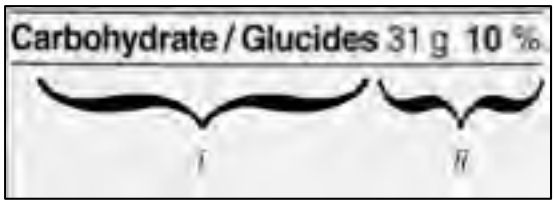


Figure 2.2 Extraction manuelle (ligne)

Pour chacune des lignes pertinentes, nous séparons le nom de l'élément nutritif (Figure 2.2:i) et les informations numériques qui s'y rattachent (Figure 2.2:ii). À l'aide du nom nous pouvons bien sûr identifier l'élément; puis, en segmentant les informations numériques, nous pouvons répertorier les quantités.

Malheureusement, malgré la facilité avec laquelle nous sommes capables d'effectuer cette tâche, il y a un grand risque d'erreur causée par les manipulations humaines (retranscription) ainsi que par la perte d'attention résultant de la répétitivité de la tâche.

2.2 Méthode automatique proposée

Les étapes de la méthode automatique que nous proposons sont très fortement inspirées de celles de la méthode manuelle puisque ces dernières sont efficaces et répondent à nos besoins. Il faut bien sûr localiser la boite, en déterminer le format linguistique, extraire l'information relative à l'échantillon et déterminer la pertinence de chaque ligne. Pour chaque ligne pertinente, nous identifions l'élément selon le maximum de vraisemblance avec les éléments contenus dans le dictionnaire (présenté à la section 3.2) de l'application. Afin de déterminer le maximum de vraisemblance, nous traitons le nom de l'élément dans chaque langue de façon indépendant; il faut également s'assurer que les caractères ne sont pas utilisés dans plus d'une langue. Si la vraisemblance est trop faible, nous considérons que la ligne est finalement non pertinente. Lorsque l'élément est identifié, nous segmentons l'information numérique selon le type d'élément qui a été identifié.

2.3 Les phases de développement

Le développement de ce projet est séparé en deux phases, qui sont : la phase d'exploration qui a permis de déterminer la faisabilité du projet; et la phase de développement de l'application elle-même qui a permis de peaufiner les résultats.

Lors de la phase d'exploration, nous avons numérisé une dizaine de boites d'information nutritionnelle afin de vérifier le concept, localiser les généralités dans la structure du modèle d'information et identifier les faiblesses de l'application. Ensuite, nous avons photographié onze produits à neuf résolutions différentes afin de tester le domaine d'applicabilité, en termes de résolution spatiale, de notre application. Lors de cette partie, nous avons aussi dû améliorer la robustesse de l'application face aux informations partielles ou manquantes causées par la performance médiocre du moteur de reconnaissance de caractère lorsqu'on lui fournit une image de basse résolution. À la fin de cette phase, nous considérons que l'application est fonctionnelle, mais plusieurs choix restent encore à faire afin d'optimiser les résultats. Étant donné que nous désirons que ces choix soient optimaux par rapport aux images que l'application rencontrera réellement, nous devons absolument utiliser des images qui seront représentatives de la réalité.

Donc, pour la seconde phase, la validation et l'optimisation de l'application, l'entreprise GS1 nous a fourni une banque d'images qu'ils ont recueillie au cours de la dernière année. La banque d'images originale contenait 1028 images, desquelles nous en avons conservé 927 (les autres ne respectant pas le modèle standard bilingue : voir l'hypothèse 3 présentée dans l'introduction)(Voir annexe 3). Après avoir déterminé plusieurs critères d'évaluation, nous avons testé les différentes configurations possibles afin d'obtenir le résultat optimal. Les critères et les configurations sont expliqués en détail aux chapitres 4 et 6.

Vous remarquerez que la première phase n'est que très peu discutée dans le présent document considérant qu'elle constitue la phase d'exploration de l'application.

2.4 Validation des résultats

Afin de déterminer la performance des différentes configurations de notre application, nous devons connaitre l'information réellement contenue dans les images de la banque d'image de la deuxième phase. Pour ce faire, nous avons extrait manuellement les informations nutritionnelles d'un ensemble d'images et créé une version informatique de chaque boite d'information nutritionnelle. Cela nous permet donc de comparer les résultats de l'application à ceux obtenus par l'humain afin de comptabiliser les erreurs commises par l'application ainsi que le type de celles-ci. Il est à noter que lors de l'extraction manuelle de ces informations, nous avons volontairement laissé les erreurs que nous avons rencontrées dans les images. Par exemple, certaines images utilisaient de mauvaises unités pour certains éléments nutritifs (ex. : l'utilisation des grammes « g » pour le sodium). Ceci aura pour effet, dans certains cas, de diminuer la performance de notre application. Si nous conservons l'exemple des unités erronées, l'application pourrait corriger l'unité erronée à l'aide du module de validation des unités ce qui aurait pour effet d'introduire une erreur entre l'image d'origine et l'information à la sortie de l'application. L'application émettra également un avertissement pour signaler qu'il a modifié la ligne en question afin de vérifier si l'erreur provient de l'image elle-même ou de l'extraction, cette approche permettra à l'utilisateur d'identifier les erreurs qui seraient contenues dans l'image et de les corriger.

2.5 Plan d'expérimentation

Nous présentons ici, l'approche qui est utilisée pour les tests à proprement parler. Nous présentons le test de répétabilité, l'approche orientée résultat, l'expérimentation exhaustive ainsi que les critères d'évaluation que nous utilisons afin d'évaluer la performance de notre application.

2.5.1 Test de répétabilité

Afin de s'assurer de la validité des résultats, nous devons tout d'abord vérifier que, si nous effectuons un même test à répétition, nous obtenons le même résultat. Pour ce faire, nous effectuerons un même test plusieurs fois (20 répétitions), puis nous comparons les résultats à deux étapes différentes dans le processus : premièrement, nous vérifions si les résultats de l'application sont constants et dans quelle mesure ils le sont, puis nous observons les informations extraites à même le moteur de reconnaissance. Il sera ainsi possible d'évaluer la répétabilité autant à la sortie du moteur de reconnaissance de caractères qu'à la sortie de notre application. Cette méthode nous permet de localiser sommairement une éventuelle variabilité du test.

2.5.2 Approche orientée résultat

Comme nous en avons discuté dans la revue de la littérature, nous utilisons une approche orientée résultat. Cette approche consiste à évaluer la performance d'une composante du système en la plaçant dans ledit système et en évaluant la performance du système lui-même. Par la suite, nous pouvons évaluer la performance du système en utilisant une autre variante de la même composante. Il est ainsi possible de déterminer laquelle de ces composantes est la plus efficace pour notre système.

2.5.3 Expérimentation exhaustive

Considérant que nous devons tester plusieurs paramètres en parallèle, nous ne pouvons pas affirmer que l'effet de la variation d'un paramètre dans une configuration donnée (en fixant chacun des autres paramètres) resterait constant pour les autres configurations. Afin de pallier à ce problème, nous avons décidé de tester toutes les configurations possibles. Cette approche nécessite beaucoup plus de tests, mais permettra une meilleure analyse des résultats; il est possible, après avoir effectué la batterie de tests, d'étudier l'effet de la

variation d'un paramètre pour toutes les configurations données et ainsi de vérifier si la tendance du paramètre est généralisée pour toutes les configurations.

2.5.4 Critères d'évaluation

Les critères serviront à évaluer l'exactitude des résultats obtenus à l'aide de l'application. Chacun de ces critères sera représenté par trois informations : le nombre d'erreurs, le nombre total d'éléments concernés et finalement le pourcentage d'erreur (soit la proportion entre la première et la deuxième valeur). Nous n'avons pas seulement évalué le pourcentage d'erreur, car dans certains cas, le nombre d'éléments concernés est très faible (par exemple, les chlorures n'apparaissent que dans 3 boites de notre banque d'images). Les critères sont divisés en 4 catégories : la première représente les boites dans leur ensemble, la seconde les lignes, la troisième les colonnes d'information (les valeurs, les unités et les pourcentages) et la quatrième les éléments nutritifs (calories, lipides, etc.).

Il est important de noter que ces catégories ne sont pas exclusives les unes des autres. La première catégorie donne une vue très globale de l'exactitude (une boite qui contiendrait une seule valeur erronée, peu importe le nombre total d'éléments qui la composent, serait tout de même considérée comme étant erronée), la deuxième catégorie donne une vue plus spécifique, mais toujours générale (encore là, dès qu'une ligne contient une erreur elle est considérée comme erronée), les deux dernières catégories fournissent les mêmes informations, mais présentées selon deux points de vue orthogonaux l'un à l'autre. L'une comptabilise les erreurs selon la colonne où elles sont survenues (valeur, unité, pourcentage) tandis que l'autre comptabilise ces mêmes erreurs selon l'élément nutritif où elles sont survenues. Vous trouverez ci-dessous la liste des critères d'évaluations ainsi que ce qu'ils impliquent :

Tableau 2.1 Critères d'évaluations

Critère	**Description**
Boite non complète	Représente toutes les boites qui contiennent des erreurs ou qui n'ont pas été traitées
Boite non traitée	Lorsque l'application ne localise pas les mots « information nutritionnelle/nutrition fact » dans l'image, cette dernière est rejetée; elle n'est donc pas traitée. Il est à noter que les éléments subséquents dans la liste ne sont pas comptabilisés pour les boites qui ne sont pas traitées.
Ligne	Représente toutes les lignes qui contiennent des erreurs, qui sont manquantes ou en trop.
Ligne manquante	Représente toutes les lignes qui sont présentes dans la boite, mais qui ne se retrouvent pas dans la réponse de l'application.
Ligne en trop	Représente toutes les lignes qui sont présentes dans la réponse de l'application, mais qui ne se retrouvent pas dans la boite. (Très souvent, les lignes en trop sont associées à des lignes manquantes dans la même boite. Il s'agit en fait d'un élément qui a été mal identifié)
Valeur	Représente toutes les erreurs de la colonne des valeurs.
Unité	Représente toutes les erreurs de la colonne des unités.
Pourcentage	Représente toutes les erreurs de la colonne des pourcentages.
Valeur (sans manquante et en trop)	*
Unité (sans manquante et en trop)	*
Pourcentage (sans manquante et en trop)	*
calories\calories (tous les éléments nutritifs)	Représente les erreurs contenues dans chacun des éléments nutritifs

*Il s'agit des mêmes critères que les trois critères précédents, à l'exemption que ceux-ci ne comptabilisent pas les erreurs provenant des lignes manquantes et en trop.

CHAPITRE 3

FONCTIONNEMENT DE L'APPLICATION

Dans cette section, nous expliquons le fonctionnement des différentes parties de l'application. Nous tentons aussi de souligner les différentes améliorations que nous avons apportées afin de résoudre les impasses ou les défauts de l'application.

En premier lieu, nous faisons un survol de l'application en regardant le fonctionnement et l'interaction des différents modules; puis nous regardons chacun des modules individuellement. L'approche que nous utilisons est composée de procédés séquentiels distincts et bien définis. En conservant les entrées et les sorties des modules, il nous est possible d'améliorer un module sans compromettre le reste du processus.

3.1 Langage de programmation

L'application a été développée à l'aide du logiciel MatLab (version 7.8.0.347, Matworks, Massachusetts, États-Unis). Ce logiciel permet de traiter et de gérer aisément les images puisqu'il utilise une représentation matricielle et possède plusieurs boites d'outils pour le traitement d'image et de signaux. De plus, son langage de programmation est très adapté au le développement et à la recherche puisqu'il s'agit d'un langage de programmation interprété; il n'a donc pas besoin d'être recompilé à chaque altération du code source. Malheureusement, la flexibilité qu'apporte le langage interprété amène un contre coup. Les langages interprétés sont beaucoup plus lents que les langages compilés. Matlab offre une option de « compilation », mais à la lumière de l'information que nous avons obtenue (http://www.mathworks.com/products/compiler/description1.html), le compilateur ne semble pas générer du code compilé, mais semble plutôt encapsuler un interpréteur de commandes Matlab et toutes les fonctions nécessaires à notre application. De cette façon, le programme peut être exécuté sur des ordinateurs qui ne possèdent pas Matlab. Dans notre cas, vu la faible charge de calcul que nous demandons à l'application, ce défaut s'en trouve amoindri.

Nous considérons tout de même que lorsque l'application aura atteint sa maturité, il serait judicieux de la reprogrammer dans un langage compilé. Dans le cas où cette option serait rejetée, l'utilisation du compilateur Matlab serait tout à fait appropriée pour permettre la portabilité de notre application. Il est à noter que le module de reconnaissance lui-même, Tesseract, est une application programmée en C que nous utilisons sous sa forme compilée.

3.2 Dictionnaire

Pour assurer une performance optimale de notre application, nous y avons inclus un dictionnaire spécialisé. Celui-ci contient tous les éléments nutritifs qui peuvent se retrouver dans une boite nutritionnelle, donc tout ce que l'application devrait rencontrer; pour ce faire, il a été conçu en se basant sur la Figure 1.3. Ce dictionnaire permettra à l'application de comparer les mots rencontrés lors de la reconnaissance avec ceux présents dans le dictionnaire. Pour chaque entrée, le dictionnaire possède le terme en français, en anglais ainsi que la configuration numérique de l'élément (discutée à la section 3.8). Étant donné que notre dictionnaire est de très petite taille, moins de 50 entrées, nous utilisons une comparaison exhaustive de chaque terme présent dans le dictionnaire.

3.3 Survol

L'image est chargée en mémoire à l'aide de fonctions Matlab. La boite d'information nutritionnelle est ensuite localisée par une des quatre méthodes disponibles (présentées à la section 3.4). L'application sauvegarde l'image de la boite et effectue ensuite un appel de commande externe afin d'activer l'OCR qui effectue la reconnaissance des caractères puis génère un fichier ASCII (.txt) contenant le résultat. L'application charge le fichier ASCII puis compare chacune de ses lignes avec chacune des entrées du dictionnaire afin de déterminer la correspondance. En minimisant l'erreur, il est possible de classifier chacune des entrées de la boite d'information nutritionnelle avec une ligne du dictionnaire. Par la suite, en se basant sur le format de l'élément nutritif associé à la ligne, l'application extrait les informations numériques et unitaires correspondantes. Si l'application détermine la

présence possible d'incohérences, il affiche, via l'interface à l'usager, l'information afin que celui-ci puisse comparer et corriger les informations recueillies par l'application avec l'image d'origine. Finalement, lorsque les informations sont considérées valides, l'application enregistre les informations dans un format base de données.

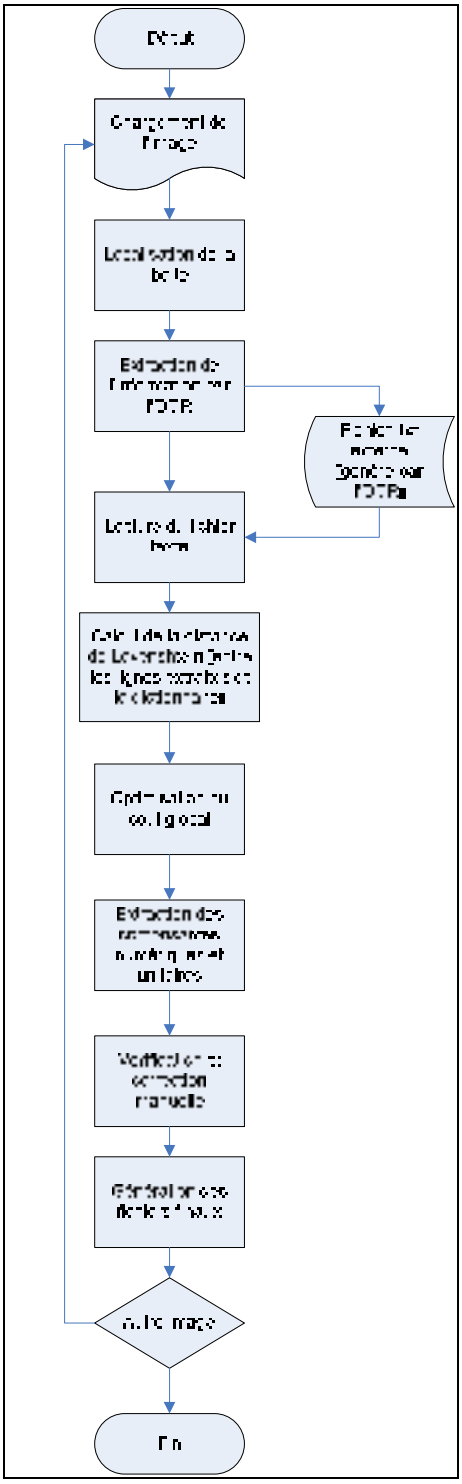


Figure 3.1 Organigramme du fonctionnement de l'application

3.4 Localisation de la boite

Afin de localiser la boite d'information nutritionnelle, quatre possibilités s'offrent à l'usager. La première survient quand l'image contient uniquement la boite d'information nutritionnelle, il n'est donc pas nécessaire de la localiser. L'application analysera uniquement l'entête de la boite afin de déterminer de quel type de boite il s'agit (Français/Anglais ou Anglais/Français). Il ne faut pas oublier qu'il existe également deux autres types de boite, soit français, soit anglais, mais ceux-ci ne sont pas inclus considérant l'hypothèse 3 présentée dans l'introduction. La deuxième possibilité est lorsque l'usager désire sélectionner lui-même la boite d'information nutritionnelle dans l'image à l'aide de l'interface mise à sa disposition. Cette interface offre des outils simples pour sélectionner une zone dans une image d'origine afin de ne pas avoir à utiliser une tierce application pour éditer l'image; de telles applications n'étant pas toujours disponibles sur le poste de travail. La troisième possibilité est lorsque l'usager désire laisser à l'application le soin de localiser la boite d'information nutritionnelle. Celle-ci créera une version binaire de l'image à traiter, puis identifiera et classera en ordre croissant toutes les zones blanches de plus de 50000 pixels. Cette valeur a été choisie de façon arbitraire tout en s'assurant qu'elle soit inférieure aux surfaces à localiser. Pour chacune des zones, l'application détermine les coordonnées minimales et maximales selon les deux axes afin d'obtenir une région plus générale, puis tente de retrouver la présence d'une boite (en localisant les termes Valeur Nutritive / Nutrition Facts) selon chacun des quatre axes orthogonaux (0°, 90°, 180°, 270°). Finalement, lorsqu'une boite est confirmée, l'application détermine le type de boite. Une quatrième méthode a été développée après l'observation des images qui nous ont été fournies par GS1 pour effectuer nos tests. Ces images sont constituées de la boite d'information entourée d'une grande bordure parfaitement noire (Voir Figure 3.2).

Figure 3.2 Boite d'information avec bordure noire

Étant donné que le noir est représenté par la valeur 0, nous pouvons facilement trouver où commence et où se termine la boite d'information. Pour ce faire, nous additionnons tous les pixels formant une même colonne, et ce, pour toutes les colonnes de l'image. Ensuite, nous n'avons qu'à déterminer la première et la dernière valeur non nulle (ne comportant pas uniquement des pixels noirs) ce qui représente respectivement les bornes de gauche et de droite de la boite d'information. Nous pouvons répéter le même processus en utilisant les lignes afin d'obtenir les bornes supérieure et inférieure de la boite. Nous, travaillons ensuite comme si nous avions uniquement la boite d'information.

3.5 Extraction par l'OCR

Une fois la boite d'information nutritionnelle identifiée, l'application sauvegarde l'image en ton de gris sur le disque en format BMP (Windows Bitmap) afin de s'assurer que l'OCR puisse lire l'image. Par la suite, l'application exécute une commande externe afin de lancer l'OCR qui, suite au traitement de l'image, génère un fichier ASCII (.txt). Par la suite, l'application charge ce fichier ASCII. Une attention particulière doit être portée à l'encodage du fichier afin de ne pas insérer d'erreurs de format dans le système.

Tel que mentionné précédemment, nous avons décidé d'utiliser l'application d'OCR Tesseract (cette application à licence ouverte (open source) est la propriété de Google) pour ce projet, ce qui permet à quiconque de l'utiliser. Malgré le fait que cette application est très performante, il s'agit tout de même d'une application gratuite en cours de développement.

Par contre, le module d'OCR est appelé en commande externe, et il serait possible en changeant une ou deux lignes de code de notre application de passer à une autre application d'OCR. La seule contrainte est que nous devons pouvoir utiliser cette application en ligne de commande (« command prompt ») et que nous devons connaitre l'encodage du fichier de sortie.

3.6 Calcul de la distance de Levenshtein

La distance de Levenshtein permet de calculer le coût de modification entre deux chaines de caractères. À l'origine, l'algorithme assigne un coût de traitement d'une unité afin d'insérer, de supprimer ou de modifier un caractère de la première chaine (celle générée par l'OCR) pour la faire correspondre à la deuxième chaine (chacune des entrées du dictionnaire).

Nos lignes à comparer (provenant de l'OCR) contiennent le nom de l'élément nutritif dans les deux langues officielles, ainsi que les informations numériques relatives à l'élément nutritif. Pour extraire les deux noms, nous devons calculer la distance de Levenshtein à deux reprises sur la même chaine de caractère. Pour ce faire, nous avons dû ajouter une quatrième opération à la méthode de Levenshtein : la suppression d'extrémité. Toutes séquences consécutives de suppression qui inclut le premier ou le dernier caractère de la chaine s'effectuent avec un coût nul.

De plus, nous devons calculer la distance de Levenshtein dans les deux langues sans qu'il y ait de superposition entre les réponses; ce qui signifie qu'un caractère ne peut pas être utilisé simultanément pour le calcul des distances en français et en anglais. Nous effectuons donc le deuxième calcul de la distance à partir de la partie du premier calcul qui a été catégorisé comme suppression d'extrémité de la fin de la chaine. Puis nous conservons encore la suppression de l'extrémité de la fin du deuxième calcul comme étant les informations numériques relatives à cet élément nutritif.

Voici un exemple qui facilitera la compréhension. Pour faciliter la lecture, nous avons utilisé les symboles suivants :

Tableau 3.1 Symboles des opérateurs de Levenshtein

Symboles	**Opérations**
E	Suppression d'Extrémité
S	Suppression
I	Insertion
M	Modification

Nous commençons avec une chaine de caractères à comparer :

L	i	p	i	c	l	e	s		I		F	t		0		g		0		%

Figure 3.3 Chaine générée par l'OCR à comparer aux entrées du dictionnaire

Cette chaine (Figure 3.3) sera comparée à chacun des termes du dictionnaire. Pour l'exemple, nous comparons avec le mot ''Lipides''. Pour les Figure 3.4 et Figure 3.5, la première ligne représente la chaine à comparer, la deuxième représente les opérateurs de Levenshtein utilisés pour chaque caractère et la dernière ligne représente l'information retenue. Selon l'algorithme de Levenshtein, la solution gagnante est la suivante :

L	i	p	i	c	l	e	s		I		F	t		0		g		0		%
				M	S			E	E	E	E	E	E	E	E	E	E	E	E	E
L	i	p	i	d		e	s													

Figure 3.4 Calcul de la distance de Levenshtein (français)

À partir de la suppression d'extrémité de la fin de la chaine (suite consécutive du symbole « E » se terminant au dernier caractère de la chaine), nous comparons avec le mot anglophone du même terme; dans ce cas-ci, il s'agit du mot ''Fat'' :

		I		F		t		0		g		0		%
	E	E	E		I		E	E	E	E	E	E	E	E
				F	a	t								

Figure 3.5 Calcul de la distance de Levenshtein (anglais)

Nous pouvons donc, dans le cas de notre exemple, conclure que la distance ‘globale’ de Levenshtein entre notre chaine de caractère et la ligne ‘Lipides / Fat’ du dictionnaire est de 3 (1 modification et 1 suppression dans le calcul français et 1 insertion dans le calcul anglais). En conservant encore une fois la suppression d’extrémité de la fin de la chaine, nous obtenons l’information numérique du nutriment :

	0		g		0		%

Figure 3.6 Information numérique

En effectuant ce processus pour chacune des lignes d’entrée générées par l’OCR par rapport à chacune des entrées du dictionnaire, il nous est possible de construire le tableau de comparaison des coûts (voir Figure 3.7) dont chaque ligne est une ligne d’entrée de l’OCR et chaque colonne un mot du dictionnaire.

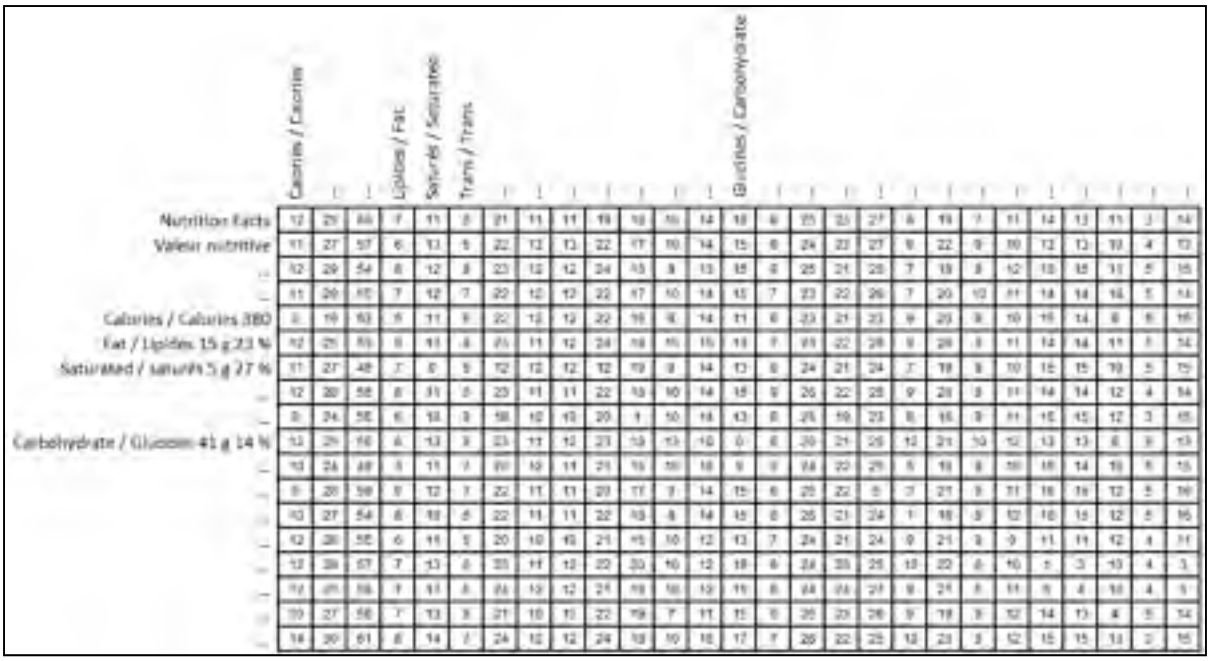


Figure 3.7 Comparaison des coûts

Chaque cellule à l'intersection d'une ligne et d'une colonne représente la distance de Levenshtein entre ces deux éléments. Dans un autre tableau possédant les mêmes dimensions que le tableau de comparaison des coûts, nous conservons toutes les possibilités d'information numérique. Il est à noter que dans le cadre de l'explication du fonctionnement de l'application, nous présentons les valeurs du tableau de comparaison des coûts comme étant la distance de Levenshtein. Cependant, dans l'application réelle, nous utilisons une version normalisée de cette valeur. Nous divisons en fait la distance de Levenshtein par le nombre de caractères présent dans les noms (Français/anglais) de l'élément. Dans l'exemple précédent, 'Lipides / Fat' contient 10 caractères, ce qui donnerait une valeur normalisée de 0.3.

3.7 Optimisation du coût global

Dans le tableau de comparaison des coûts (Figure 3.7), les premières lignes contiennent des informations qui ne se retrouvent pas dans le dictionnaire (Boite Nutritionnelle, teneur, etc.).

Il s'agit en fait des lignes précédant la ligne Calories. Afin de trouver la ligne Calories, nous localisons la valeur minimale dans la première colonne (qui correspond aux calories) et considérons la ligne correspondante comme étant le début du tableau. En plus de déterminer le début des informations pertinentes, nous devons aussi, s'il y a lieu, en trouver la fin. Certaines boites possèdent une zone de texte situé sous le tableau qui énumère les éléments qui ont volontairement été omis du tableau puisque leur quantité était négligeable. Selon la langue première utilisée nous devrions donc rencontrer les termes : «Source négligeable de » ou « Not a significant source of » s'il s'agit d'une boite qui possède cette particularité. Nous recherchons donc la plus haute ligne qui contienne une de ces deux informations et nous considérons que le tableau se termine à la ligne qui précède cette dernière. Dans les exemples qui suivent, nous considérons qu'aucune de ces deux lignes n'a été identifiée, et nous utilisons donc le tableau jusqu'à la fin.

Afin de minimiser le coût global de traitement de la boîte, plusieurs algorithmes ont été développés. Le but étant de trouver un résultat idéal tout en minimisant le temps de traitement. Nous présentons donc ici uniquement deux méthodes, dont celle de l'algorithme gagnant. Avant de continuer, il ne faut pas oublier que nous considérons que les lignes apparaissent dans le même ordre que celui présenté dans le *Guide d'étiquetage et de publicité sur les aliments* (voir section 1.1) qui est le même que l'ordre d'apparition des éléments dans le dictionnaire. Donc, les indices (I, J) d'une valeur retenue doivent obligatoirement être plus grands que les indices correspondants à la valeur retenue précédemment (équation 3.1). Cette valeur se retrouve donc sur une colonne plus à droite et sur une ligne plus basse que la valeur sélectionnée précédemment. Ceci implique que les valeurs sélectionnées formeront un effet de cascade descendante vers la droite (voir Figure 3.8).

$$I_n < I_{n+1}$$
$$J_n < J_{n+1}$$
$$I_n : indice\ de\ ligne\ de\ l'élément\ actuel \quad (3.1)$$
$$I_{n+1} : indice\ de\ ligne\ de\ l'élément\ suivant$$
$$J_n : indice\ de\ colonne\ de\ l'élément\ actuel$$
$$J_{n+1} : indice\ de\ colonne\ de\ l'élément$$

Tout d'abord, nous considérons qu'une distance de Levenshtein de 0 est automatiquement considérée comme exacte (en vert).

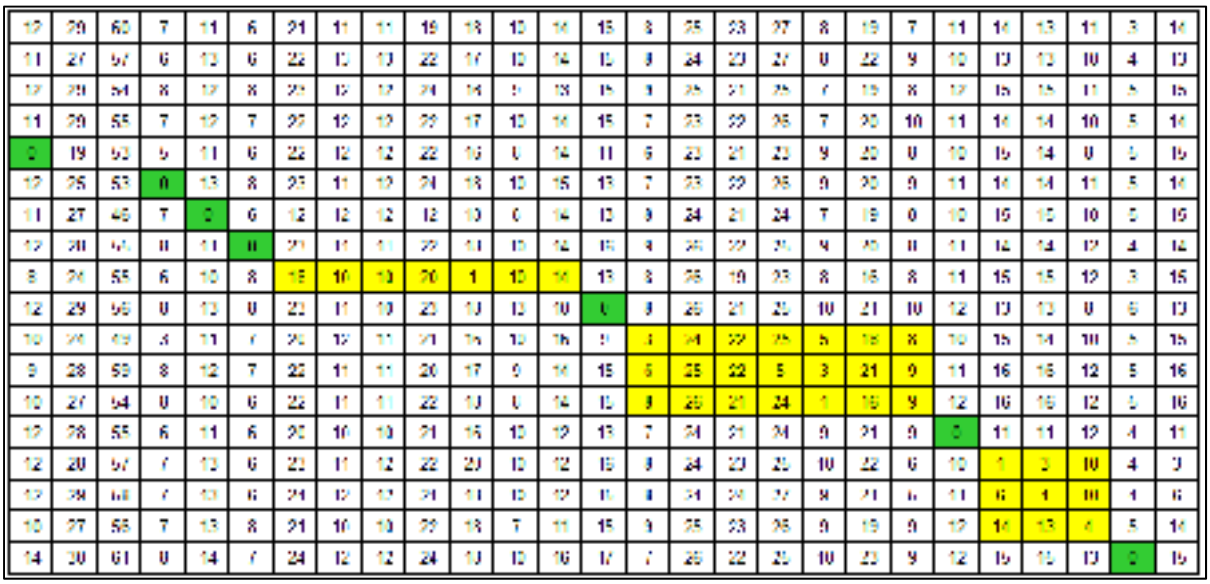


Figure 3.8 Comparaison des coûts (Éléments sans erreur)

Il ne reste plus qu'à optimiser les groupes de lignes (en jaune) qui se trouvent entre deux lignes idéales (en vert). Pour la suite de l'explication, nous prendrons en exemple le deuxième groupe de lignes en jaune dans la Figure 3.8:

3	21	22	25	5	18	8
6	25	22	5	3	21	9
8	26	21	21	1	16	9

Figure 3.9 Deuxième groupe
de lignes en jaune de la figure 3.8

Cette problématique d'optimisation peut être représentée par une modélisation par arbre. Nous pouvons observer cet arbre à la Figure 3.10:

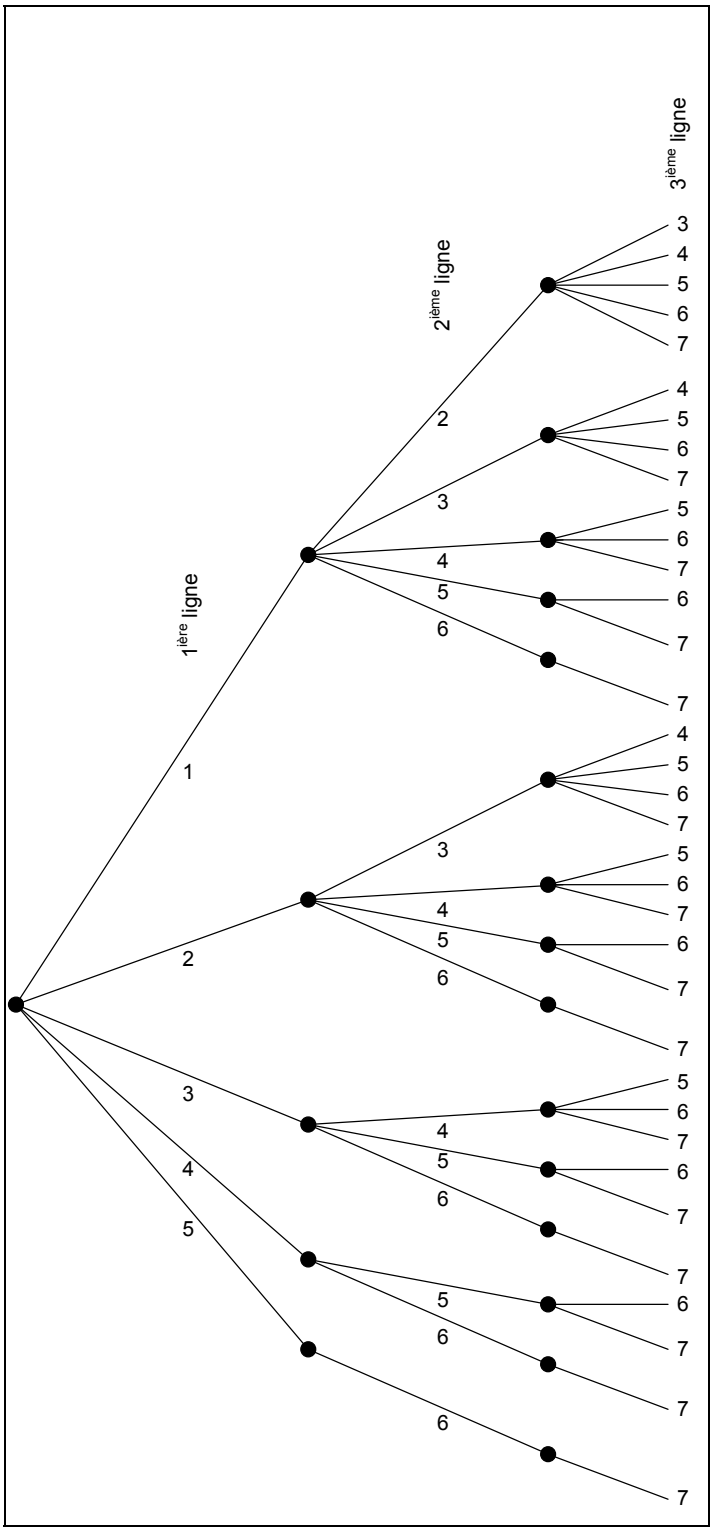


Figure 3.10 Arbre de possibilité

L'arbre de la Figure 3.10 possède trois niveaux. Chacun de ces niveaux représente une ligne du groupe à optimiser. Chaque branche de l'arbre représente le choix d'une cellule (indice de la colonne) pour une ligne donnée. Comme nous en avons discuté précédemment, chaque cellule sélectionnée doit se trouver à la droite de la cellule sélectionnée sur la ligne précédente. Cette contrainte est clairement observable dans la Figure 3.10. Afin d'optimiser ce groupe, nous avons tout d'abord expérimenté avec une fonction récursive qui gère un seul niveau (une seule ligne) et appelle une itération d'elle-même pour le niveau suivant (la ligne suivante). Malheureusement, nous travaillons en langage interprété, ce qui implique que cette approche demande beaucoup de ressources informatiques. De plus, nous avons remarqué que la méthode récursive recalculait un grand nombre de fois les mêmes informations simplement parce qu'elle ne gardait pas l'information en mémoire.

Regardons tout de même son fonctionnement. L'algorithme récursif (Niveau 1) choisit le premier terme de la première ligne (Figure 3.11, en bleu) et appelle une autre instance (N2) pour la ligne suivante (Figure 3.11, en rose) qui appelle aussi une autre instance (N3) pour trouver l'optimal de la dernière ligne (Figure 3.11, en orange).

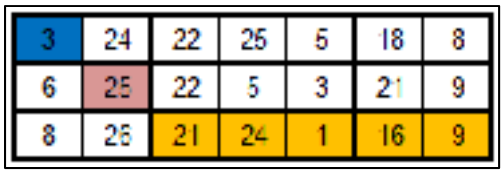


Figure 3.11 Première itération de l'algorithme récursif

Une fois que N2 obtient l'information optimale de N3 (soit le coût minimal de 1 à la colonne 5) il se déplace d'une colonne et appelle un nouveau N3.

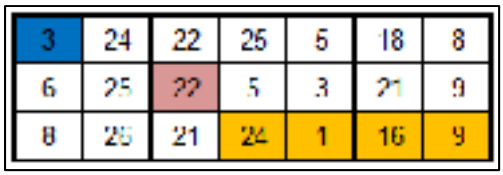


Figure 3.12 Deuxième itération
de l'algorithme récursif

Comme nous l'avons noté précédemment, cet algorithme recalcule plusieurs fois les mêmes informations, car il ne conserve pas le résultat. Prenons l'exemple de la Figure 3.13, le N3 qui y est présenté est calculé 6 fois dans notre exemple (3 fois où N3 contient exactement cette séquence, et 3 fois où N3 contient une séquence plus grande qui inclus la séquence en jaune).

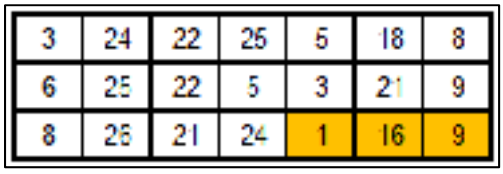


Figure 3.13 Exemple de redondance
de calcul

Nous pouvons également observer la répétition de cette séquence dans l'arbre de possibilité Figure 3.14. Pour la visibilité, nous avons mis les séquences exactes en gras et nous avons atténué la partie de l'arbre qui ne nous intéressait pas. Il est à noter que pour les séquences identiques, la répétition inclut également l'élément du niveau 2 comme il est possible de le remarquer dans la Figure 3.14.

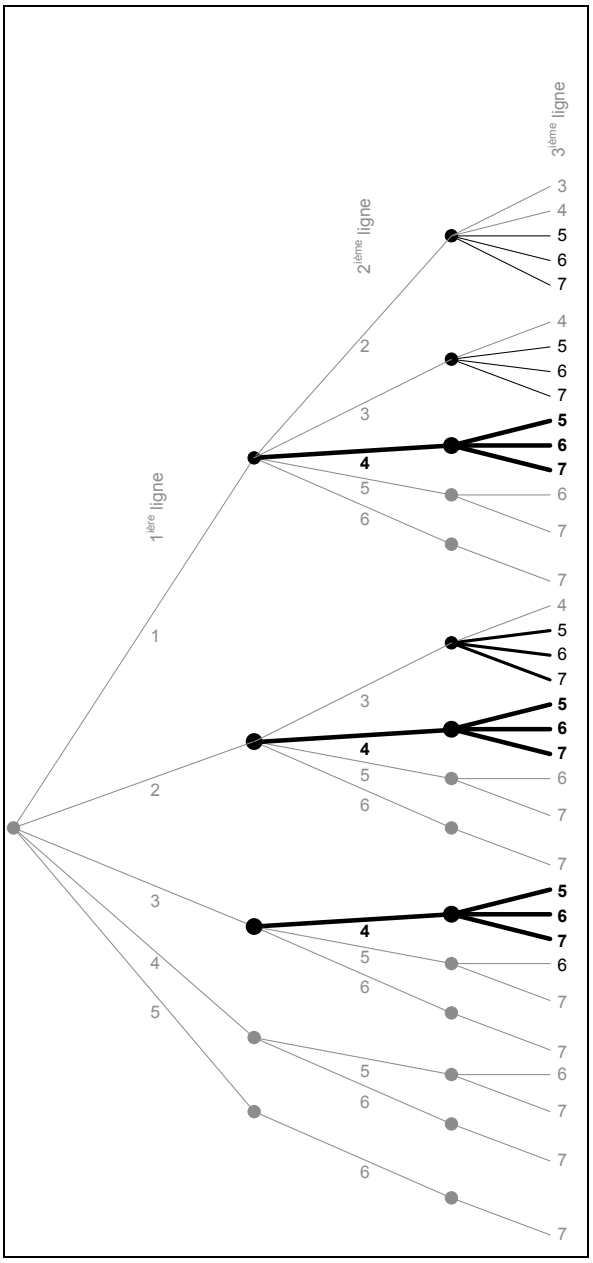


Figure 3.14 Exemple de redondance de calcul (arbre)

De plus, d'une itération à l'autre, la longueur de la chaine à comparer pour la troisième ligne variera, mais la plupart du temps la réponse sera la même; le coût de 1 à la 5e colonne. Il y a donc un gaspillage de la capacité de calcul. Ici, le problème est peu critique, mais s'il s'agissait d'un groupe de plusieurs lignes et plusieurs colonnes (ex. : 12 lignes x 20 colonnes)

cette perte augmenterait de façon exponentielle. Nous avons déterminé que le volume de calcul nécessaire pour la méthode récursive en fonction du nombre de colonnes (c) et du nombre de lignes (l) est de :

$$V(c,l) = \sum_{i=0}^{l-1} \sum_{j=1}^{(c-l+1)-i} j \qquad (3.2)$$

c: *nombre de colonnes*

l: *nombre de lignes*

Bien qu'il aurait été possible, mais compliqué, de modifier la méthode récursive afin de lui permettre de conserver l'information, nous avons plutôt développé une méthode de notre cru afin de minimiser le volume de calcul. Après conception de cette méthode, nous avons remarqué qu'il s'agissait en fait d'une méthode dérivée de la programmation dynamique (voir section 1.2.1). Le but étant de ne pas avoir à recalculer la même information plusieurs fois; afin d'y parvenir, nous attaquons le problème par la fin. Afin de présenter cette méthode, nous réutilisons le même exemple que pour l'algorithme récursif (Figure 3.9).

Afin de faciliter la compréhension de cette autre méthode, nous allons tout d'abord représenter cette problématique d'une nouvelle façon, mais toujours en utilisant la représentation par arbre (Figure 3.15). Dans cette représentation, chaque cellule est représentée par un nœud de l'arbre. Tout comme dans le tableau, les cellules sont réparties sur 3 lignes et il y a qu'une seule occurrence par cellule. Au centre des nœuds se trouve l'indice de la colonne de la cellule correspondante. Les branches présentent les agencements possibles entre les cellules. Finalement, nous pouvons encore une fois observer l'effet de cascade causé par l'obligation de l'ordre d'apparition des éléments.

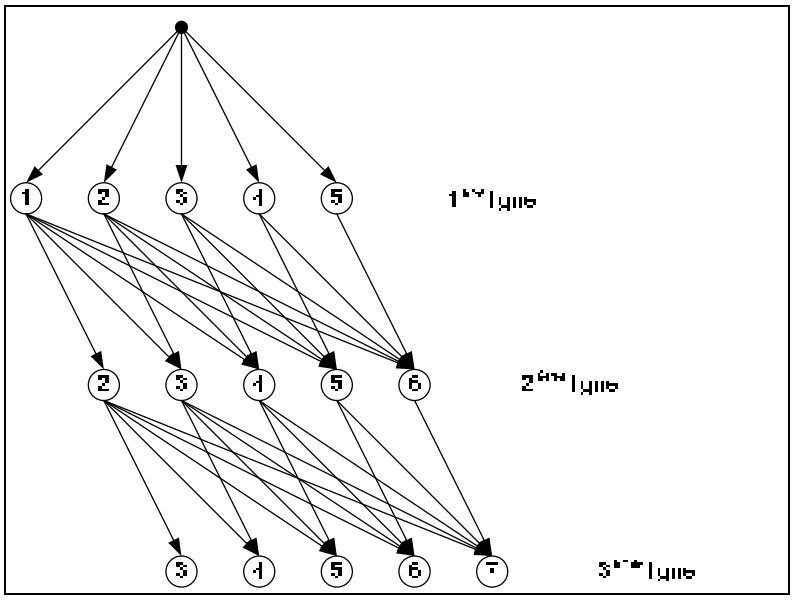


Figure 3.15 Représentation par arbre

Dans un tableau (Figure 3.16) aux mêmes dimensions que la zone à optimiser (Figure 3.9), nous initialisons la dernière ligne (en bleu) en mémorisant le coût de la cellule correspondante (dans la Figure 3.9) ainsi que l'indice de sa colonne. Il ne faut pas oublier que certaines cellules ne sont pas utilisables pour s'assurer de conserver l'effet de cascade.

		21 [3]	24 [4]	1 [5]	16 [6]	9 [7]

Figure 3.16 Méthode dynamique (première itération)

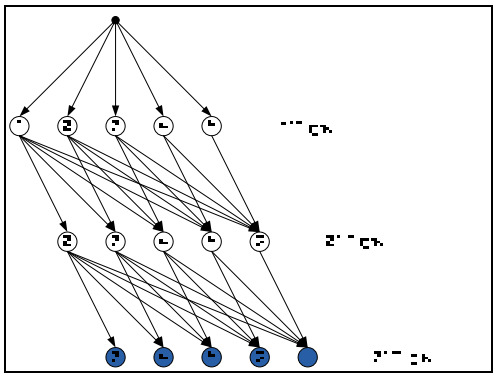


Figure 3.17 Représentation par arbre
(première itération)

Par la suite, nous traitons chaque ligne, une à la fois, en remontant vers le haut. Pour chaque cellule, nous additionnons son coût avec le coût minimum de la section de la ligne inférieure qui se trouve à droite de la cellule que nous calculons. Une fois cette valeur minimale localisée, nous concaténons la position de la cellule actuelle avec la liste de la cellule minimale. Donc lorsque nous traitons la cellule (2, 4) (en bleu dans la Figure 3.18), nous travaillons sur la zone en jaune. Nous localisons, dans la zone jaune, la cellule dont le coût est minimal (dans ce cas-ci, il s'agit d'un coût de 1 à la colonne 5). Donc la valeur de la cellule est de 6 (coût de la cellule (2, 4) = 5 (selon Figure 3.9) + coût minimum de la section de la ligne précédente = 1). Puis nous concaténons l'indice de la colonne actuelle (4) avec la liste de la cellule minimale de la ligne précédente [5]. Ce qui résulte en [4, 5].

			6 [4,5]	12 [5,7]	30 [6,7]	
		21 [3]	24 [4]	1 [5]	16 [6]	9 [7]

Figure 3.18 Méthode dynamique (itération ultérieure)

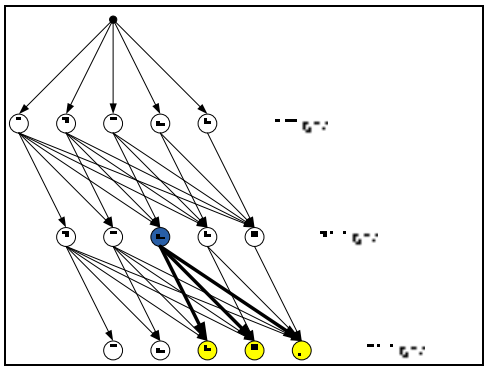


Figure 3.19 Représentation par arbre
(itération ultérieure)

Nous appliquons cette procédure jusqu'à remplir le tableau final (Figure 3.20). Par la suite, nous trouvons la valeur minimale de la première ligne, et nous obtenons par le fait même la séquence des indices des colonnes à utiliser pour minimiser le coût du groupe.

9 [1,4,5]	30[2,4,5]	28 [3,4,5]	37 [4,5,7]	35 [5,6,7]		
	26 [2,5]	23 [3,5]	6 [4,5]	12 [5,7]	30 [6,7]	
		21 [3]	24 [4]	1 [5]	16 [6]	9 [7]

Figure 3.20 Méthode dynamique (solution finale)

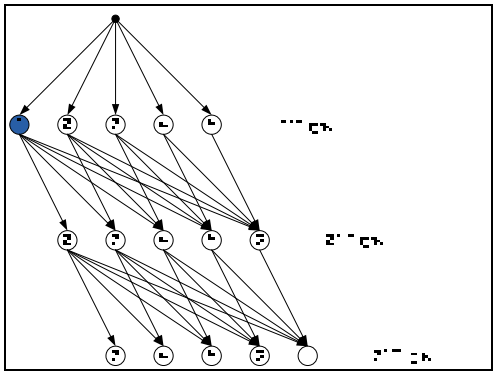


Figure 3.21 Représentation par arbre
(solution finale)

Nous obtenons toujours une solution minimale, tout comme avec la méthode récursive, mais pour un volume de calcul plus faible :

$$V(c, l) = l * (c - l + 1) \quad (3.3)$$

De plus, les deux équations (3.2 , 3.3) de volume de calcul ne prennent pas en compte que notre méthode n'utilise qu'une seule fonction. Il y a donc beaucoup moins d'allocations de mémoire, ce qui pourrait procurer un gain de rapidité supplémentaire surtout si la mémoire vive est limitée.

3.8 Extraction des caractéristiques

Maintenant que nous possédons la correspondance optimale entre les lignes de l'OCR et le dictionnaire, nous pouvons récupérer les informations numériques du nutriment (obtenues à la fin du calcul de distance de Levenshtein) correspondant au couple ligne/colonne.

Il existe 4 configurations numériques dans les boites d'information nutritionnelle (Tableau 3.2).

Tableau 3.2 Configuration numérique

Config	**Format**	**Ex.**
0	###	calories/calories **250**
1	### unit	sugars/sucres **7 g**
2	### %	vitamin a/vitamine a **4 %**
3	### unit ### %	sodium/sodium **580 mg 24 %**

Il est possible d'associer une configuration à chaque nutriment d'une boite d'information nutritionnelle. Lors de nos expérimentations, nous avons observé que le cholestérol peut correspondre à la configuration 1 ou 3; soit une valeur avec unité, avec ou sans pourcentage. Pour pallier à ce problème, nous avons donc créé la configuration 4 qui consiste à analyser les configurations 1 et 3 et à conserver celle qui génère le moins d'erreurs. Si l'information

numérique ne correspond pas à la configuration de la ligne sur laquelle nous travaillons (qu'il y ait trop ou pas assez d'information), un avertissement sera émis pour cibler le problème, mais le processus continuera.

Pour vérifier cette information, nous ne pouvons pas utiliser la distance de Levenshtein, puisque nous ne connaissons pas la valeur que nous recherchons. Donc, la seule vérification que nous pouvons faire consiste à comparer la chaine de caractères que nous avons avec la configuration correspondante. Pour ce faire, l'application crée des groupes de caractères numériques et des groupes de caractères alphabétiques. Il est à noter que les séparateurs décimaux ('.' et ',') sont considérés comme étant des caractères numériques.

Afin d'améliorer les performances, nous avons ultérieurement ajouté un module qui valide l'information alphabétique correspondant aux unités. Sachant que celle-ci devrait prendre uniquement les valeurs « g » ou « mg » selon l'élément nutritif visé, si une autre valeur est détectée, elle est remplacée par la valeur par défaut et un avertissement est émis. Aux chapitres 4 et 5, nous faisons référence à ce module comme étant la validation des unités. Ce dernier peut être ou ne pas être utilisé afin d'évaluer le gain en performance qu'il procure.

Un deuxième module facultatif a également été développé. Durant la vérification de la configuration, le module identifie l'absence d'un pourcentage (dans une configuration de type 2 ou 3) et, au lieu de laisser l'espace vide, pose l'hypothèque qu'il s'agit d'une valeur nulle, il inscrit donc la valeur 0 dans l'espace vide. Comme nous considérons qu'il y a eu une erreur de vérification, un avertissement est tout de même émis. Tout comme le module précédent, il peut être ou ne pas être utilisé afin d'évaluer son effet sur la performance globale.

3.9 Vérification humaine

Dans le cas où un ou plusieurs avertissements ont été émis, l'application considère que le risque d'erreur est trop grand. Il affichera donc l'information extraite via l'interface à

l'usager afin que celui-ci prenne la décision finale. Cette interface (Figure 3.22) contient l'image d'origine qui a été traitée ainsi que toutes les informations pertinentes qui en ont été extraites. L'usager peut donc corriger l'information avant la sauvegarde finale.

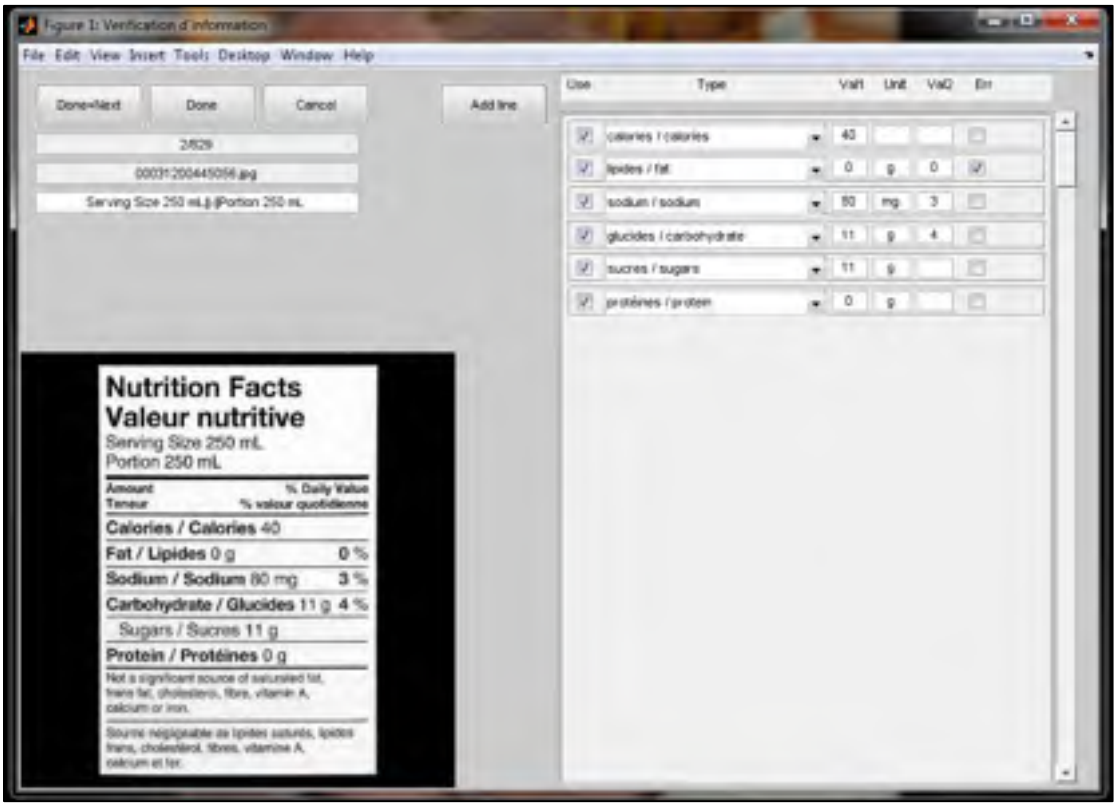


Figure 3.22 Interface de vérification

3.10 Sauvegarde vers la base de données

Une fois que toutes les lignes ont été traitées, et qu'au besoin l'usager est intervenu, l'information peut être sauvegardée vers la base de données. Étant donné que nous ne travaillons pas ultérieurement avec la base de données, nous avons, dans le cadre de notre projet, simplement sauvegardé l'information sous un format csv. Sachant pertinemment qu'un fichier csv de 180 colonnes n'est pas très facile à consulter, nous avons également généré un fichier ASCII (Figure 3.23), contenant uniquement les éléments qui sont présents, dont le format rappelle celui de la boite d'information nutritionnelle.

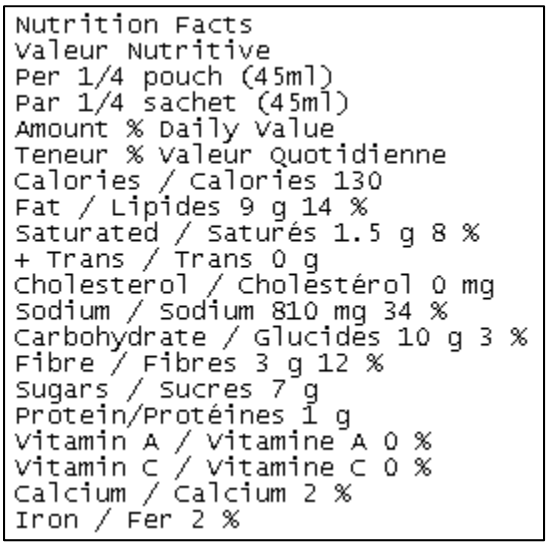


Figure 3.23 Exemple de fichier ASCII

CHAPITRE 4

PREMIÈRE PHASE DE TEST

Maintenant que le concept de l'application a été élaboré et que son fonctionnement a été validé, nous devons évaluer l'efficacité de notre application ainsi que l'impact des différentes modifications que nous pourrions y apporter.

4.1 Test de répétabilité

Avant de pouvoir évaluer l'efficacité de notre application, nous devons tout d'abord nous assurer qu'il est possible de répéter les tests tout en conservant une variabilité minimale. Pour ce faire, nous effectuons un test 20 fois et nous comparons les résultats. Comme nous l'avons décrit dans la méthodologie, nous observons les résultats à deux endroits. Tout d'abord à la sortie de l'application afin de vérifier s'il y a une variabilité en ce qui concerne les résultats réels de l'application, puis à la sortie du moteur de reconnaissance pour déterminer si le post traitement de notre application altère la variabilité du moteur de reconnaissance.

4.2 Variables d'optimisation

Dans le chapitre de la méthodologie, nous avons établi les critères d'évaluations, il nous est donc maintenant possible d'évaluer l'efficacité de notre application. Nous ne voulons cependant pas en rester là, puisque notre but est d'obtenir la meilleure efficacité possible. Pour ce faire, nous ferons varier divers paramètres de l'application afin d'obtenir le résultat optimal.

La première variable est la version du moteur de reconnaissance de caractère (OCR). Comme nous avons discuté précédemment, nous utiliserons l'OCR Tesseract. Depuis le début de ce projet, une nouvelle version de l'OCR a été publiée (2.04) et une autre est actuellement en développement. Nous effectuerons donc nos tests à l'aide des versions 2.01,

2.04 et 3.0. Par la suite, nous effectuerons un unique test à l'aide du moteur Omnipage 16, qui est une application de reconnaissance de caractère commerciale qui a fait ses preuves. Ce dernier test permettra de situer l'efficacité du moteur Tesseract par rapport à une application commerciale.

La deuxième variable que nous ferons varier est la méthode de localisation de la boite. Tel que présenté au CHAPITRE 3, l'usager a à sa disposition 4 méthodes pour localiser les boites : laisser l'application localiser la boite, considérer que l'image en entier est une boite, considérer que l'image est entourée d'une bordure noire ou encore utiliser la petite interface de l'application pour spécifier manuellement la localisation de la boite. Étant donné que cette dernière méthode implique que l'usager sélectionne des points à l'écran, il serait assez difficile d'assurer la reproductibilité du test. Nous avons donc choisi de tester uniquement les trois premières méthodes. À première vue, nous croyons que cette variable devrait surtout avoir un impact sur le temps d'exécution.

La troisième variable est l'utilisation ou non de la validation des unités. Lors du traitement de l'information, l'application tente de trouver une zone de caractères pour l'associer aux unités. Elle ne se préoccupe cependant pas de savoir s'il s'agit réellement d'une unité. Nous savons que pour un élément nutritif donné, il n'y a qu'une seule valeur valable pour les unités (rien, g ou mg). Nous pouvons donc écraser l'unité lue par l'unité supposée, ce qui devrait pratiquement éliminer les erreurs dues aux unités. Il reste cependant la possibilité que la boite n'ait pas été produite en respectant le guide de l'étiquetage et que d'autres unités soient utilisées (kg par exemple).

La quatrième variable est l'utilisation ou non du remplissage des pourcentages. Lors du traitement de l'information, l'application tente de déterminer le pourcentage, mais s'il n'en trouve pas, il laisse l'espace vide et averti d'un risque d'erreur pour la ligne. Comme plusieurs de ces erreurs ont lieu lorsque la valeur de la boite est zéro (0), et considérant aussi que l'espace vide et la valeur de zéro peuvent être considérés comme équivalents, nous

proposons de remplir ces espaces par des zéros. Il va de soit que ce remplissage sera effectué uniquement pour les éléments nutritifs qui devraient posséder un pourcentage.

L'investigation de l'effet de chacune de ces 4 variables constitue la première phase de test. Nous effectuerons le traitement des images pour chacune des combinaisons possibles. En jumelant les 3 versions de l'OCR avec les 3 méthodes de localisation de boite ainsi que l'utilisation ou non de la validation des unités et de l'utilisation ou non du remplissage des pourcentages, nous devons donc effectuer 36 tests différents (3x3x2x2), excluant le test de comparaison à l'OCR commercial Omnipage. En analysant les résultats de ces tests (disponible à l'annexe 1), nous pourrons choisir de façon éclairée la meilleure combinaison de variables.

Afin de pouvoir comparer et analyser les résultats, nous présentons les résultats sous forme de tableaux. Comme nous en avons discuté dans la méthodologie, ces tableaux sont divisés en 4 parties : des informations sur les boites dans leur ensemble, sur les lignes, sur les colonnes ainsi que sur chaque élément nutritif.

Info	# erreurs	# total	% erreur
Boite non complète	518	927	55.88
Boite non traite	18	927	1.94

Figure 4.1 Information sur les boites

Info	# erreurs	# total	% erreur
Ligne	1802	11830	15.23
Ligne manquante	719	11830	6.08
Ligne en trop	53	11830	0.45

Figure 4.2 Information sur les lignes

Info	# erreurs	# total	% erreur
Valeur	904	8731	10.35
Valeur	1125	7864	14.31
Pourcentage	1317	7821	16.84

Figure 4.3 Information sur les colonnes

Info	# erreurs	# total	% erreur
Valeur (sans manquante et en trop)	132	7959	1.66
Unite (sans manquante et en trop)	353	7092	4.98
Pourcentage (sans manquante et en trop)	545	7049	7.73

Figure 4.4 Information sur les colonnes

Info	# erreurs	# total	% erreur
calories\calories	68	909	7.48
calories from fat\calories des lipides	4	4	100
calories from saturated + trans\calories des lipides saturés et trans	0	0	0
fat\lipides	271	909	29.81
saturated\saturés	294	737	39.89
trans\trans	114	736	15.49
polyunsaturated\polyinsaturés	8	34	23.53
omega-6\oméga-6	4	31	12.9
omega-3\oméga-3	3	31	9.68
monounsaturated\monoinsaturés	5	37	13.51
cholesterol\cholestérol	88	725	12.14
sodium\sodium	117	865	13.53
potassium\potassium	8	42	19.05
carbohydrate\glucides	149	908	16.41
fibre\fibres	178	733	24.28
dietary fibre\fibres alimentaires	3	3	100
soluble fibre\fibres solubles	0	2	0
insoluble fibre\fibres insolubles	1	2	50
sugars\sucres	64	856	7.48
sugar alcohols\polyalcools	0	14	0
starch\amidon	3	4	75
protein\protéines	70	909	7.7
vitamin a\vitamine a	54	729	7.41
vitamin c\vitamine c	61	763	7.99
calcium\calcium	63	741	8.5
iron\fer	100	744	13.44
vitamin d\vitamine d	8	26	30.77
vitamin e\vitamine e	10	15	66.67
vitamin k\vitamine k	0	0	0
thiamin\thiamine	8	44	18.18
riboflavin\riboflavine	8	50	16
niacin\niacine	2	36	5.56
vitamin b6\vitamine b6	5	16	31.25
folate\folate	4	34	11.76
vitamin b12\vitamine b12	5	26	19.23
biotin\biotine	1	12	8.33
pantothenate\pantothénate	3	20	15
phosphorus\phosphore	5	26	19.23
iodide\iodure	2	9	22.22
magnesium\magnésium	1	19	5.26
zinc\zinc	2	15	13.33
selenium\sélénium	2	2	100
copper\cuivre	3	5	60
manganese\manganèse	0	3	0
chromium\chrome	0	0	0
molybdenum\molybdène	0	0	0
chloride\chlorure	3	4	75

Figure 4.5 Information sur les éléments

CHAPITRE 5

ANALYSE DE LA PREMIÈRE PHASE

5.1 Test de répétabilité

Pour le test de répétabilité, nous avons utilisé une configuration de base : nous avons utilisé la version 2.01 de Tesseract et le mode de localisation automatique des boites d'information nutritionnelle. Nous n'avons utilisé ni la validation des unités ni la correction des pourcentages, car ces deux méthodes risquaient d'altérer les résultats. Par exemple, deux erreurs d'unité différentes seraient remplacées par la même valeur si le module de validation des unités était utilisé. Les résultats de ce test sont surprenants. Nous ne pensions pas obtenir une grande variabilité, mais nous ne nous attendions pas aux résultats que nous avons obtenus. C'est-à-dire, aucune variation entre les 20 tests que ce soit pour l'application elle-même ou encore le moteur de reconnaissance.

5.2 Versions de l'OCR Tesseract

À la suite de nos multiples tests, aucune des trois versions de l'OCR ne s'est démarquée de façon significative. Les versions 2.0X de l'OCR traitent plus de boites sans introduire d'erreur et rejettent moins d'images comme n'étant pas des boites d'information nutritionnelle que ne le fait la version 3.0. Cependant, la version 3.0 de l'OCR omet beaucoup moins de lignes. Nous observons également que la version 2.04 est légèrement plus performante que la version 2.01. Un nombre beaucoup plus important d'images de test serait nécessaire afin de tenter de départager les différentes versions de l'OCR. Étant donné qu'il s'agit en fait de la même application, il est également possible que l'on ne puisse pas mettre en évidence la supériorité d'une des versions. Donc, malgré que les écarts soient très faibles entre les différentes versions, si nous avions à choisir une seule version, nous opterions pour la version 2.04.

Nous avons été déçus de la proportion des boites qui contenaient des erreurs. Toutes les combinaisons des versions du moteur de reconnaissance et des méthodes de localisation sans l'utilisation des modules de corrections obtiennent plus de 55% de boites contenant des erreurs. Cette valeur nous semble catastrophique. Cependant, le test que nous avons effectué à l'aide de l'OCR commercial Omnipage 16 a obtenu plus de 60 % de boites qui contenaient des erreurs. Ce qui signifie que certaines de nos configurations ont mieux performé que l'application commerciale qui nous servait de comparaison. À partir de cette information, nous considérons que la faible efficacité obtenue serait attribuable à la configuration particulière de l'information que nous avons à extraire, plutôt qu'au fonctionnement du moteur de reconnaissance lui-même.

5.3 Méthode de localisation

Les trois méthodes que nous avons utilisées ont donné des résultats acceptables. Cependant, la méthode qui considère la boite uniquement a donné des résultats beaucoup moins concluants. Ce résultat est tout à fait compréhensible considérant que les images n'étaient justement pas uniquement des boites d'information nutritionnelle.

Pour les deux autres méthodes, les résultats sont très similaires. La méthode qui considère un fond noir entourant la boite nutritionnelle donne des résultats légèrement plus performants que la méthode de localisation de la boite (automatique) pour les versions 2.0X alors que la situation est inversée pour la version 3.0. Nous ne nous expliquons pas cette différence entre les deux méthodes considérant qu'après les diverses manipulations sur l'image d'origine, ces deux méthodes devraient obtenir, pour ainsi dire, la même sous-image.

Bien que ce ne soit pas un critère très déterminant, la méthode sur fond noir effectue le traitement environ deux fois plus rapidement que la méthode automatique. Ce qui nous fait pencher pour cette seconde méthode pour les prochaines phases de test.

5.4 Validation des unités

En observant le tableau ci-dessous, nous remarquons l'efficacité de l'utilisation de la validation des unités. Elle permet, pour toutes les versions et méthodes de localisation confondues, de diminuer le nombre de lignes contenant des erreurs de plus de 18%. Cette amélioration se répercute par une diminution du nombre de boites contenant des erreurs. Les seules erreurs d'unité qui subsistent suite à l'utilisation de la validation des unités sont causées par des boites ne respectant pas le guide d'étiquetage et de publicité sur les aliments.

Tableau 5.1 Diminution d'erreur de ligne en fonction de la version et du mode de localisation (validation des unités)

Version	**Mode de localisation**	**# de ligne en erreur sans validation**	**# de ligne en erreur avec validation**	**% de diminution des lignes en erreur**
2.01	Localiser	1802	1459	19.0%
	Fond noir	1727	1406	18.6%
	Boite uniquement	2278	1833	19.5%
2.04	Localiser	1772	1442	18.6%
	Fond noir	1665	1340	19.5%
	Boite uniquement	2337	1893	19.0%
3.00	Localiser	2489	1193	52.1%
	Fond noir	3158	1749	44.6%
	Boite uniquement	259	107	58.7%

Tableau 5.2 Diminution des boites non complètes en fonction de la version et du mode de localisation (validation des unités)

Version	Mode de localisation	# de boite non complète sans validation	# de boite non complète avec validation	% de diminution des boites non complètes
2.01	Localiser	518	510	1.54%
	Fond noir	523	516	1.34%
	Boite uniquement	608	591	2.80%
2.04	Localiser	511	505	1.17%
	Fond noir	517	513	0.77%
	Boite uniquement	608	596	1.97%
3.00	Localiser	728	554	23.9%
	Fond noir	771	598	22.4%
	Boite uniquement	926	918	0.86%

5.5 Correction des pourcentages manquants

Bien que le remplissage des pourcentages n'obtienne pas le même gain en performance que la validation des unités (considérant que le remplissage ne touche que les lignes contenant un élément de pourcentage dont le champ est resté vide), il n'en reste pas moins que les résultats sont prometteurs.

Tableau 5.3 Diminution d'erreur de ligne en fonction de la version et du mode de localisation (remplissage des pourcentages)

Version	Mode de localisation	# de ligne en erreur sans validation	# de ligne en erreur avec validation	% de diminution des lignes en erreur
2.01	Localiser	1802	1580	12.3%
	Fond noir	1727	1510	12.6%
	Boite uniquement	2278	1971	13.5%
2.04	Localiser	1772	1565	11.7%
	Fond noir	1665	1456	12.6%
	Boite uniquement	2337	2038	12.8%
3.00	Localiser	2489	2190	12.0%
	Fond noir	3158	2706	14.3%
	Boite uniquement	259	213	17.8%

Nous obtenons une diminution du nombre de lignes erronées de plus de 11 %, qui entraine, encore une fois, une diminution du nombre de boites contenant des erreurs.

Tableau 5.4 Diminution des boites non complètes en fonction de la version et du mode de localisation (remplissage des pourcentages)

Version	**Mode de localisation**	**# de boite non complète sans validation**	**# de boite non complète avec validation**	**% de diminution des boites non complètes**
2.01	Localiser	518	501	3.28%
	Fond noir	523	504	3.63%
	Boite uniquement	608	576	5.26%
2.04	Localiser	511	497	2.74%
	Fond noir	517	500	3.29%
	Boite uniquement	608	570	6.25%
3.00	Localiser	728	719	1.24%
	Fond noir	771	765	0.78%
	Boite uniquement	926	919	0.76%

Malgré l'écart original entre la validation des unités et le remplissage des pourcentages, lorsque nous comparons ces deux modifications au niveau des boites non complètes, nous observons que le gain en performance est du même ordre.

5.6 Combinaison vérification des unités et des pourcentages

Une méthode n'empêchant pas l'autre, nous pouvons aussi tenter d'appliquer les deux méthodes simultanément. Nous avons observé que l'amélioration obtenue par la combinaison des méthodes au niveau des lignes contenant des erreurs est égale à la somme de l'amélioration de chacune des méthodes prises individuellement.

Cependant, en effectuant la même vérification au niveau des boites non complètes, nous pouvons remarquer que l'utilisation des deux méthodes simultanément offre de bien meilleurs résultats que la somme des deux méthodes prise individuellement; comme en témoignent les résultats présentés au tableau ci-dessous.

Tableau 5.5 Pourcentage de diminution des boites non complètes selon la méthode utilisée

Version	**Mode de localisation**	**% de diminution avec validation**	**% de diminution avec remplissage**	**Somme des 2 % précédent.**	**% de diminution avec les 2 méthodes**
2.01	Localiser	1.54%	3.28%	4.8%	18.7%
	Fond noir	1.34%	3.63%	5.0%	21.6%
	Boite uniquement	2.80%	5.26%	8.1%	20.2%
2.04	Localiser	1.17%	2.74%	3.9%	18.8%
	Fond noir	0.77%	3.29%	4.1%	20.9%
	Boite uniquement	1.97%	6.25%	8.2%	20.1%
3.00	Localiser	23.9%	1.24%	25.1%	39.4%
	Fond noir	22.4%	0.78%	23.2%	37.1%
	Boite uniquement	0.86%	0.76%	1.6%	3.67%

Cette amélioration remarquable est due au fait que les méthodes prises individuellement ne peuvent résoudre toutes les erreurs d'une boite, que si toutes ces erreurs correspondent à cette méthode. Dans le cas des deux méthodes combinées, il est donc possible de résoudre les boites qui contiennent les deux types d'erreur. Dans la même logique, en considérant que les lignes en erreur contiennent habituellement une seule erreur, il nous est impossible d'observer cet effet au niveau des lignes en erreur.

5.7 Évaluation de l'efficacité

Maintenant que nous avons tenté d'améliorer l'efficacité de l'application, nous pouvons jeter un coup d'œil à la configuration optimale. Dans notre cas, il s'agit de la version 2.04 utilisant la méthode de localisation qui considère l'arrière-plan noir et qui utilise les deux modules d'améliorations; soit la validation des unités et le remplissage des pourcentages. Cette combinaison nous permet d'obtenir un taux de 44.12 % des boites qui contiennent des erreurs

dont 4.64 % n’ont pas été traitées par l’application; cette dernière ne les ayant pas reconnues comme des boites d’informations nutritionnelles.

CHAPITRE 6

DEUXIÈME PHASE DE TEST

Maintenant que nous avons été en mesure d'identifier la configuration optimale, soit la version 2.04 du moteur de reconnaissance utilisant la localisation de boite sur arrière-plan noir, avec l'utilisation de la validation des unités et le remplissage des pourcentages. Nous tentons ici d'optimiser des variables de plus bas niveau. Durant certaines étapes de notre procédé, nous utilisons des seuils dont la valeur a été déterminée de façon arbitraire lors de la conception de l'application. Nous tentons donc ici d'ajuster leur valeur afin d'optimiser le résultat. Finalement, nous tentons d'améliorer l'extraction de l'information en testant diverses méthodes de prétraitement et de rehaussement de l'image.

6.1 Seuil d'identification des boites

Le cinquième paramètre est donc l'ajustement de la valeur seuil qui permet à l'application de déterminer si l'image, ou la sous image, sélectionnée représente une boite d'information nutritionnelle. La valeur par défaut utilisée lors de la première phase de test est un coût de transformation total de 8 pour les deux lignes : « Valeur Nutritive » et « Nutrition Facts ». Ce qui représente une erreur de moins de 26 %. La diminution de cette valeur devrait augmenter le nombre de boites d'information nutritionnelle qui ne seront pas détectées comme telles. En contrepartie, bien que l'augmentation de cette valeur permette de diminuer le nombre de boites non détectées, si la valeur augmente trop, elle pourrait inclure des images qui ne sont pas des boites d'information nutritionnelle. Cependant, en considérant l'utilisation actuelle de l'application, toutes les images introduites possèdent une boite d'information ce qui empêche l'application d'inclure des boites qui n'en seraient pas. Il faut cependant noter que cette valeur seuil est également utilisée par la méthode de localisation afin de déterminer laquelle des sous images contient la boite. L'utilisation d'une valeur seuil trop élevée pourrait nuire à cette méthode. Pour l'évaluation de cette valeur seuil, nous considérons uniquement le nombre de boites qui ne sont pas traitées.

6.2 Seuil d'identification des lignes

Le sixième paramètre est très similaire au cinquième; il s'agit de la valeur seuil pour déterminer si une ligne détectée représente un élément nutritif ou s'il s'agit uniquement d'une ligne bruitée. La valeur seuil par défaut est de 60 % d'erreur. Donc, s'il y a moins de 60 % d'erreur dans la ligne, elle sera conservée, sinon elle sera rejetée. En augmentant cette valeur, le nombre de lignes manquantes devrait diminuer. Il ne faut cependant pas oublier que le nombre de lignes en trop, lui, risque d'augmenter. Donc, par opposition, si l'on diminue cette valeur, le nombre de lignes manquantes devrait augmenter et celui des lignes en trop diminuer. Il faut cependant considérer que lors de l'utilisation de l'interface graphique pour vérifier et corriger les erreurs, il est beaucoup plus facile d'éliminer une ligne en trop (il suffit simplement de décocher une case) que d'ajouter une ligne manquante (ajout de ligne, sélection d'éléments, inscription des valeurs)

6.3 Prétraitement

Par la suite, nous tenterons d'appliquer différentes méthodes de prétraitement afin d'améliorer la performance de l'application.

En considérant que la présence des filets (lignes horizontales séparant certains éléments nutritifs) puisse nuire au bon fonctionnement du moteur d'OCR, nous effectuons un test dans lequel nous remplaçons ces filets par la couleur de l'arrière-plan afin de vérifier la nécessité ou non de filtrer ces filets. Pour ce faire, nous utiliserons la transformée de Hough afin de localiser les lignes horizontales possédant un nombre minimal de pixels supérieur à la moitié de la largeur de la boite.

Par la suite, nous tenterons d'améliorer les résultats en appliquant un filtre qu'il soit moyennant, médian ou encore Laplacien.

Finalement, nous appliquons un seuillage. Nous testons premièrement la méthode d'Otsu (Otsu, 1979), puis la méthode de Niblack (présentée par (Trier et Jain, 1995)) dont les calculs sont accélérés à l'aide de la méthode des images intégrales (Viola et Jones, 2004). Par la suite, nous les comparons avec la méthode de base qui consiste à fournir une image en tons de gris au moteur de reconnaissance.

CHAPITRE 7

ANALYSE DE LA DEUXIÈME PHASE

7.1 Seuil d'identification des boites

Comme expliquée précédemment, cette valeur détermine si une image contient une boite d'information nutritionnelle ou non. Considérant le nombre de caractères présent dans les deux lignes « Valeur Nutritive » et « Nutrition Facts », la valeur seuil peut varier de 0 (les deux lignes ne contiennent aucune erreur) à 31 (aucun des caractères de ces deux lignes n'a été identifié). Nous avons donc fait varier cette valeur, ce qui nous a permis de générer la Figure 7.1. La Figure 7.2 met l'emphase sur les valeurs de 10 à 31 afin d'en faciliter la lecture.

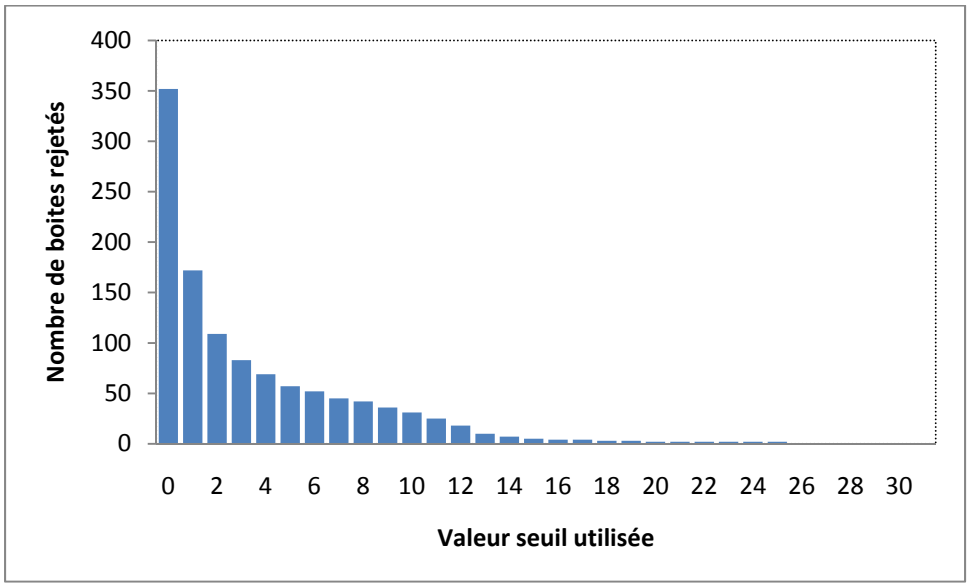


Figure 7.1 Nombre de rejets en fonction de la valeur seuil d'identification des boites

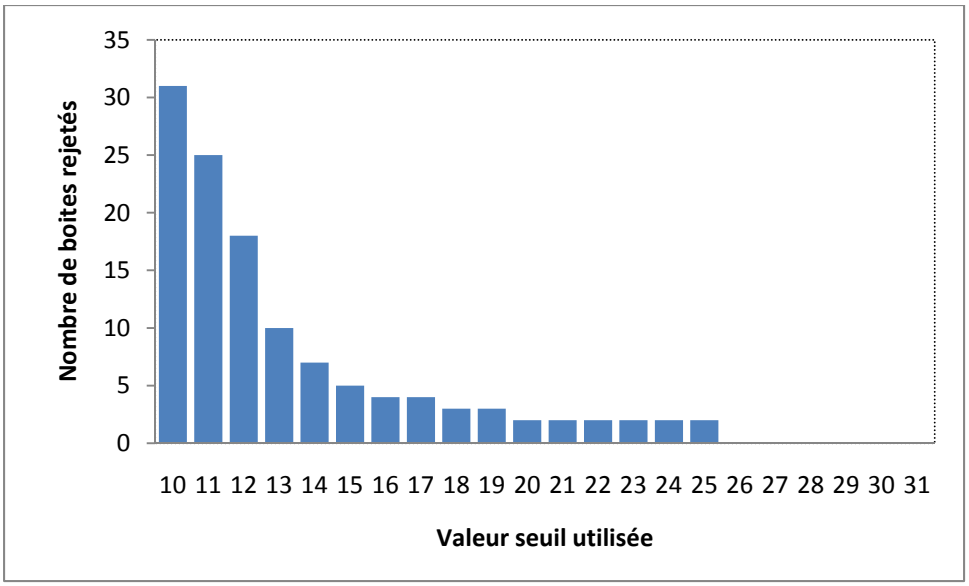


Figure 7.2 Nombre de rejets en fonction de la valeur seuil d'identification des boites (pour les valeurs de 10 à 31)

Il apparait clairement que la valeur seuil de 8, sélectionnée arbitrairement pour les précédents tests, n'est vraiment pas optimale. En analysant les Figure 7.1 et Figure 7.2, nous pouvons sélectionner la valeur qui permettra de minimiser le nombre de boites rejetées tout en minimisant la valeur seuil elle-même afin d'éviter d'accepter des boites qui n'en seraient pas. Nous avons opté pour la valeur seuil de 20, car bien qu'elle ne permet pas d'éviter tous les rejets, elle devrait permettre de rejeter la plupart des images qui ne sont pas des boites d'information nutritionnelle considérant qu'une image qui n'est pas une boite d'information nutritionnelle devrait contenir 11 caractères dans l'ordre (mais pas nécessairement consécutifs) des chaines « Valeur Nutritive » et « Nutrition Facts » afin d'être incluse.

7.2 Seuil d'identification des lignes

Cette valeur permet de déterminer si une ligne appartient réellement à la boite ou si elle ne représente que des caractères artéfacts générés lors de la reconnaissance de l'image. Cette valeur étant normalisée, nous pouvons la faire varier de 0 à 1 (0 à 100%). Nous avons donc décidé de tester toutes les valeurs comprises entre 0.05 et 0.95 à intervalle de 0.05; bien que

nous anticipons que plusieurs valeurs d'extrémité seront rejetées (les résultats de ces tests sont disponibles à l'annexe 2). La Figure 7.3 présente le nombre de boites non complètes (contenant des erreurs) en fonction de la valeur utilisée comme seuil de ligne. Nous observons que les valeurs minimales se situent autour de 0.50 ce qui est très près de la valeur 0.60 choisie arbitrairement pour le début des expériences. Il est à noter qu'à l'exception des valeurs d'extrémité, toutes les valeurs de seuil donnent des résultats similaires. Il n'est pas impossible que le résultat optimal de 0.50 puisse être simplement causé par l'ensemble de tests lui-même. Une plus grande banque d'image serait nécessaire afin de valider cette valeur. Nous avons tout de même décidé d'utiliser la nouvelle valeur seuil de 0.50 pour le reste de nos expériences.

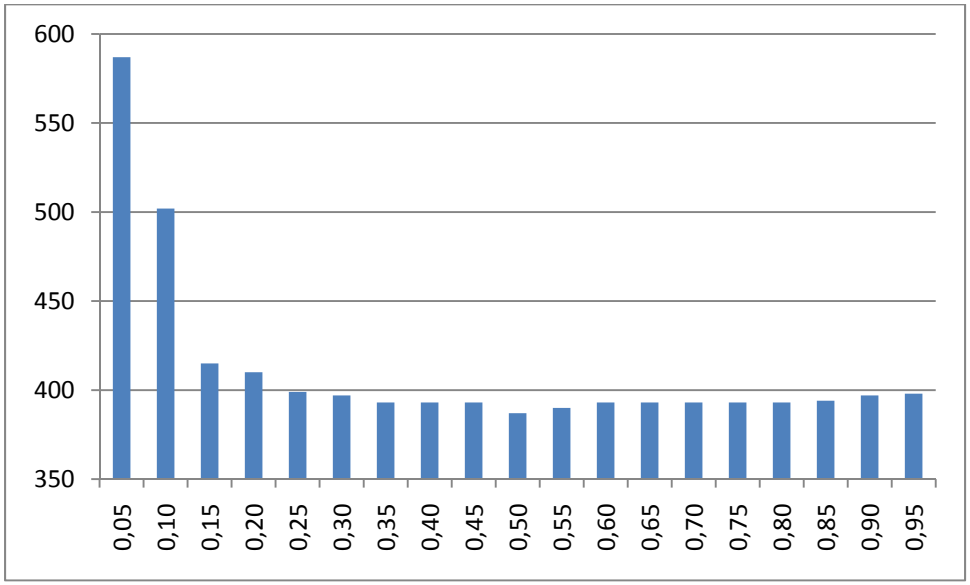


Figure 7.3 Nombre de boites non complètes selon le seuil de ligne

7.3 Prétraitement

Afin d'améliorer les résultats de l'application, nous avons tenté différentes opérations de prétraitement. Nous avons tout d'abord tenté d'éliminer les filets à l'aide de la transformée du Hough; cependant, cette approche n'a apporté aucune amélioration.

Par la suite, nous avons appliqué divers filtres afin d'améliorer la qualité de l'image, mais encore une fois cette approche n'a pas porté fruit.

Finalement, nous avons appliqué deux méthodes de seuillage (otsu et niblack) que nous avons comparées avec la méthode d'origine du moteur de reconnaissance. Nous avons effectuer ce test dans trois situations différentes : la configuration optimale de la première phase de test, la configuration optimale utilisant le seuil d'identification des boites et le seuil d'identification des lignes optimisées et finalement, sachant que la méthode du moteur de reconnaissance pose l'hypothèse que lors du seuillage, la classe qui possède le plus d'éléments représente l'arrière-plan, nous avons donc appliqué ce même concept à nos deux méthodes de seuillage.

Tableau 7.1 Comparaison du pourcentage de boite non complète

	Configuration phase 1	Avec optimisation des seuils.	Avec la gestion de l'arrière-plan
Tesseract	44.12 %	41.75 %	41.75 %
Otsu	49.95 %	47.46 %	47.46 %
Niblack	49.51 %	47.68 %	47.68 %

Nous pouvons remarquer que l'utilisation de la gestion de l'arrière-plan (selon l'hypothèque que la classe qui possède le plus d'éléments représente l'arrière-plan) n'apporte aucune amélioration. Ceci pourrait s'expliquer par le fait que le moteur de reconnaissance utilise une analyse par composantes connectées et qu'il ainsi assez facile d'identifié un texte blanc sur fond noir et de l'inverser.

Malheureusement, que ce soit les opérations de seuillage ou encore les tentatives d'amélioration de l'image, aucune des approches que nous avons utilisées n'a permit n'améliorer nos résultats. Dans la plupart des cas, les modifications n'ont eu aucun effet sur

les résultats, si ce n'est une faible dégradation des performances. Il ne faut cependant pas oublier que l'utilisation de ces opérations augmente le temps de traitement de l'image.

Deux éléments principaux peuvent expliquer le peu de succès suite à l'application des divers traitements, la redondance des opérations ainsi que la qualité des images d'origine. Nous avons vu à la section 1.4.1 que l'application Tesseract utilise déjà une fonction de seuillage. Donc le fait d'effectuer préalablement une étape de seuillage ne devrait pas fournir un gain substantiel. Étant donné le contraste prononcé et la répartition de l'histogramme de la plupart des images, l'utilisation d'une méthode de seuillage local n'est pas d'une grande utilité. L'utilisation des filtres d'améliorations rencontre la même problématique. Vu la qualité déjà très grande des images d'origine, l'application des filtres ne semble pas procurer d'amélioration notoire au niveau des résultats. Il ne faut cependant pas exclure la possibilité de leur utilisation dans l'éventualité de l'utilisation d'images de plus basse qualité. Dans un tel cas, l'application de filtres devrait permettre d'obtenir un gain de performance.

Malgré les résultats très peu concluants des tests de prétraitement, nous avons tout de même réussi, lors de cette phase de test, à améliorer les résultats du système. Nous sommes parvenus à atteindre seulement 41,75 % des boites qui contiennent des erreurs dont moins de 1 % est causé par des boites n'ayant pas été traitées par l'application.

CHAPITRE 8

DISCUSSION

8.1 L'efficacité de l'OCR

Lors de nos démarches, nous avons obtenu des résultats bien inférieurs à nos espérances par rapport au nombre de boites ne contenant aucune erreur. Afin de nous assurer que le problème ne provenait pas du moteur de reconnaissance que nous utilisons, nous avons effectué un test à l'aide d'un moteur de reconnaissance commercial qui nous a donné des résultats similaires, voire moins bons.

Nous désirons donc expliquer l'écart entre l'efficacité du moteur de reconnaissance Tesseract d'environ 98 % tels qu'il est présenté dans (Rice, Jenkins et Nartker, 1995) et celle d'environ 60 % que nous avons obtenu lors de nos expériences.

Il faut tout d'abord noter que l'efficacité présentée dans (Rice, Jenkins et Nartker, 1995) représente la proportion de caractères adéquatement reconnus tandis que notre efficacité représente la proportion de boites reconnues sans aucune erreur. Nous considérons que la majorité des erreurs proviennent des valeurs numériques (quantités et pourcentage) et que l'impact des unités est négligeable en raison du module de correction qui leur est destiné.

En analysant la banque d'image à notre disposition, nous avons établi qu'il y a en moyenne 13 lignes d'éléments par boite et que celles-ci contiennent en moyenne chacune 2 caractères. Ce qui nous donne 26 caractères à risque par boite. Donc, afin de déterminer la probabilité d'obtenir une boite sans erreur, nous devons calculer la probabilité de reconnaitre tous les 26 caractères adéquatement.

$$P_{boite} = (P_{caractères})^{nombre\ de\ caractère} \tag{8.1}$$

L'équation ci-dessus prévoit que nous avons une probabilité de 59.14% d'identifier une boite sans erreur; ce qui correspond très fortement avec les 58.25 % de boites que nous avons identifiées sans erreur à l'aide de la configuration finale de la deuxième phase de test.

8.2 Critère de performance acceptable

Comme nous l'avons présenté au début de ce mémoire, ce projet tente d'automatiser, autant que possible, le processus. Bien que nous ayons tenté, lors du développement, de maximiser le nombre de boites sans erreurs, nous ne croyons pas pouvoir obtenir 100% de boite sans erreurs. Nous avons généré un histogramme présentant la distribution des boites selon le nombre d'erreurs qu'elles contiennent.

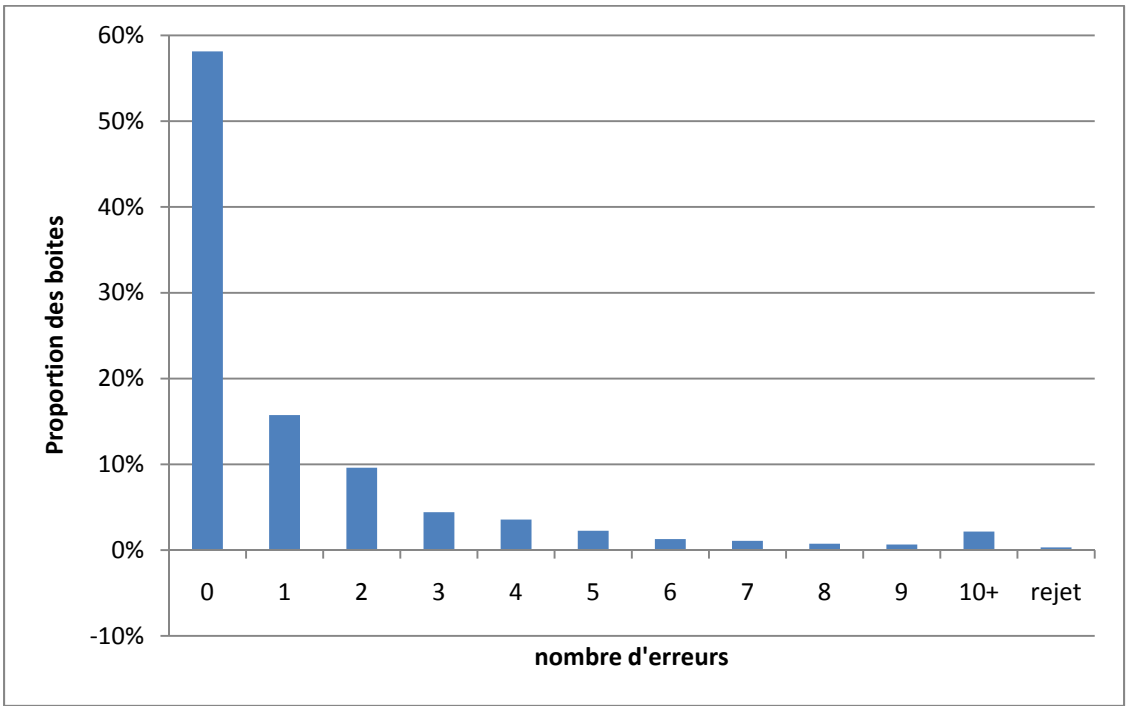


Figure 8.1 Distribution des boites selon le nombre d'erreurs contenues

Nous considérons qu'il est acceptable, dans un cadre industriel, qu'une boite contienne jusqu'à 3 erreurs; en considérant bien sûr qu'il s'agisse d'une minorité, et qu'il est possible pour l'usager de corriger ces erreurs à l'aide de l'interface. Afin de bien nous représenter la

proportion des boites qui sont incluses avec un objectif de N erreurs ou moins, nous avons également généré une figure présentant la distribution cumulative selon le nombre d'erreurs.

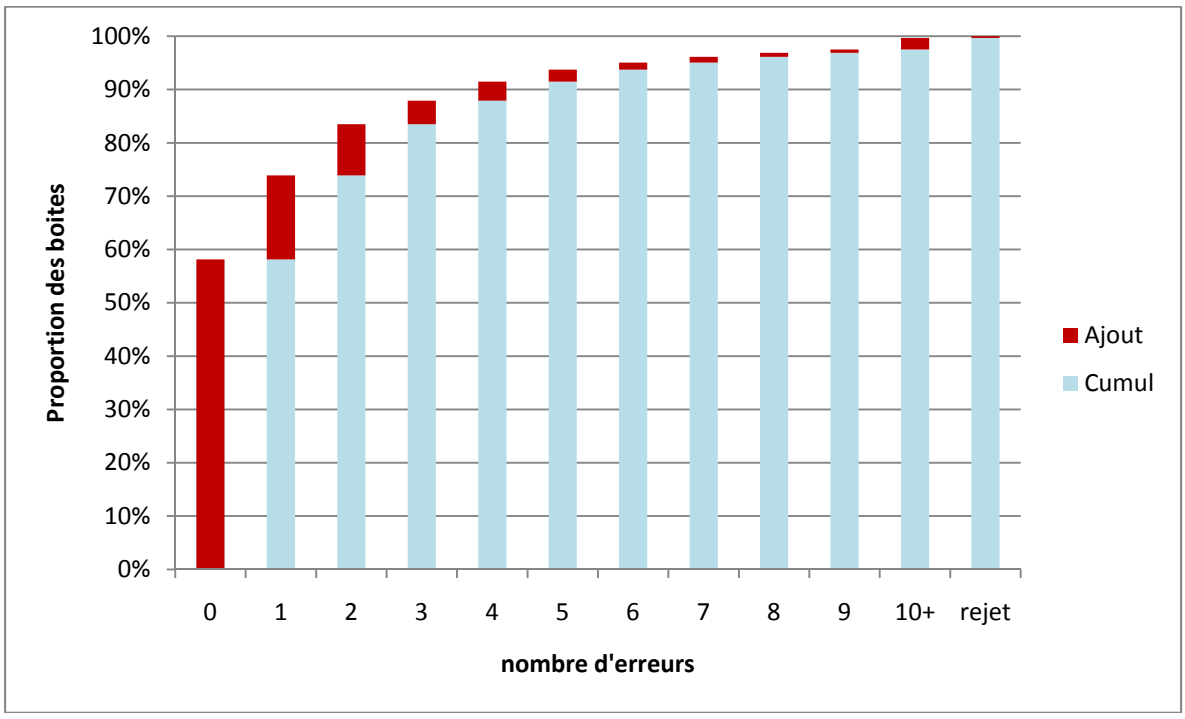


Figure 8.2 Distribution cumulative des boites selon le nombre d'erreurs contenues

Il est possible d'observer dans la Figure 8.2 que près de 83 % des boites que nous avons traitées contiennent 2 erreurs ou moins et que si l'on augmente le nombre d'erreurs acceptable à 3, nous obtenons près de 88% des boites traitées. Bien que ce résultat ne soit pas parfait, il est tout à fait acceptable.

8.3 Erreurs statistiques de type I et de type II

Nous avons expliqué précédemment que l'application génère un avertissement lorsqu'elle détecte un problème dans le format d'un élément nutritif ou encore lorsqu'elle effectue une correction à l'aide d'un des deux modules de correction. Nous avons donc observé dans quelle proportion ces avertissements étaient fondés. Étant donné que les avertissements s'appliquent sur les lignes, nous devons obligatoirement observer la situation à cette échelle.

Il existe deux types d'erreur : l'erreur de type I qui, dans notre cas, consiste à ne pas émettre d'avertissement alors qu'il y a une erreur sur la ligne et l'erreur de type II qui consiste à émettre un avertissement alors qu'il n'y a pas d'erreur. Dans notre problématique, ces 2 types d'erreur, bien qu'en faible nombre, sont non négligeables (Tableau 8.1).

Tableau 8.1Distribution des erreurs statistiques

	L'application signale un avertissement.	**L'application ne signale pas d'avertissement.**
La ligne contient une erreur.	338 (2.81%)	834 (6.94%)
La ligne ne contient pas d'erreur	690 (5.74%)	10148 (84.5%)

Nous savons que plusieurs erreurs de type II surviennent lorsque les modules de correction sont utilisés, car ceux-ci corrigent une erreur tout en créant un avertissement. Cette situation augmente la charge de travail de l'usager, mais ne remet pas en cause la qualité de l'information extraite. En contrepartie, les erreurs de type I, qui sont entre autres générées lorsque l'application effectue une mauvaise reconnaissance (par exemple, reconnaitre un 6 alors qu'il s'agissait en fait d'un 5), ne sont pas perçues par l'application et pourraient éventuellement se retrouver dans la base de données.

8.4 Niveau de confiance de l'information extraite

Plusieurs situations obligent l'application à demander une vérification de la part de l'usager; notons principalement les informations qui ne respectent pas le format de leur élément nutritif, les unités ayant été remplacées par le module de correction correspondant ou encore les pourcentages manquants ayant été remplacés par le second module de correction. Il n'y a pas nécessairement une erreur, mais plutôt un doute raisonnable qui mérite d'effectuer une validation.

Il peut cependant survenir une erreur lorsque tout semble en règle pour l'application (erreur de type I : pas d'erreur de format ni de correction de la part des deux modules de correction). Dans ce cas, afin de s'assurer de l'exactitude de l'information extraite, il incombe à l'usager d'effectuer une vérification exhaustive de toute l'information. Pour renforcer le niveau de confiance de l'information extraite, il serait possible, pour tous les éléments possédant la configuration numérique 3 (Tableau 3.2), de comparer la quantité avec le pourcentage de l'apport quotidien afin de repérer d'éventuelle anomalie. Trois autres méthodes pouvant améliorer le niveau de confiance de l'information extraite seront présentées dans les recommandations à la fin de ce document.

8.5 L'utilisation de test « un contre tous »

Afin d'améliorer la validité statistique de nos résultats, il aurait été préférable d'utiliser une approche de test d'un contre tous. C'est-à-dire, séparer la banque d'image de test en partie égale, utiliser une de ces parties pour effectuer l'apprentissage de l'application, en déterminer les paramètres, puis valider les résultats à l'aide des autres parties.

Dans notre projet, il n'y a pas vraiment d'apprentissage durant la première phase de test. C'est-à-dire que les diverses variations (version du moteur de reconnaissance, méthode de localisation de la boite, utilisation de la validation des unités et l'utilisation de la correction des pourcentages) ont été déterminées préalablement à la connaissance de la banque d'image. Il n'y avait donc aucun paramètre à optimiser afin qu'il soit ensuite validé avec la partie restante de la banque d'image.

Malgré tout, nous aurions pu tenter de séparer la banque d'image en plusieurs parties afin d'effectuer plusieurs tests pour chacune des configurations. Cette approche nous aurait permis d'obtenir des résultats statistiquement plus fiables en présentant les résultats sous forme de moyenne et d'écart-type; ce qui minimise la possibilité que les résultats optimaux soient dus à la banque d'image elle-même.

Deux éléments ont fait que nous avons décidé de ne pas prendre cette approche dans le cadre de ce projet. Tout d'abord, afin que cette approche soit efficace, il faut que chacune des parties soit représentative de la banque d'image dans son ensemble. Il nous semble difficile de rendre ces parties représentatives de la banque si l'on considère toutes les variations que ces images peuvent posséder (nombre d'élément par boite, qualité de l'image, éléments particuliers de la boite, etc.) Deuxièmement, si nous tentons, par exemple, de faire dix parties, cela ne nous laisse qu'environ 93 images par parties, ce qui n'est pas beaucoup. Il aurait bien sûr été possible de diviser la banque d'image en un plus petit nombre de parties, mais encore une fois, si le nombre est trop petit, les informations statistiques ne seront pas représentatives. À la lumière de ces deux informations, nous avons donc décidé de ne pas utiliser cette approche.

CONCLUSION

Dans ce mémoire, nous avons développé une application, programmée dans l'environnement Matlab (version 7.8.0.347, Matworks, Massachusetts, États-Unis), permettant d'extraire les informations pertinentes contenues dans l'image d'une boite d'information nutritionnelle. Ce travail a été possible grâce à l'utilisation du moteur de reconnaissance Tesseract. Bien que nous ayons choisi un moteur de reconnaissance en particulier, l'application a été développée de façon à permettre de changer le moteur de reconnaissance pour n'importe quel autre moteur pouvant être utilisé en ligne de commande et dont nous connaissons le format de sortie.

Pour réaliser ce projet, nous avons analysé la structure, les généralités et les particularités des boites d'information nutritionnelle. Cette connaissance à priori de l'information à extraire nous permet de renforcer le poids d'un choix lorsqu'il y a redondance d'information ou encore de cibler des zones d'erreurs potentielles lorsque le résultat n'est pas conforme à la structure prévue.

Pour parvenir à nos fins, nous avons testé différentes méthodes de localisation de la boite d'information nutritionnelle avec différentes versions du moteur de reconnaissance et l'utilisation ou non de module de correction que nous avons développé. Par la suite, nous avons amélioré nos résultats, en ajustant quelques paramètres de bas niveau. Finalement, pour peaufiner le tout, nous avons testé quelques méthodes de prétraitement.

Dans la configuration finale, l'information de 58 % des images de boite de notre banque d'image de test est extraite sans générer la moindre erreur. De plus, l'extraction de l'information de 83,5% des images s'effectue en générant 2 erreurs ou moins. Ce résultat est inférieur à ce que nous espérions; il ne permettra pas d'automatiser complètement le traitement de l'information. Par contre, l'efficacité que nous avons obtenue est amplement

suffisante pour concevoir une application qui prétraiterait l'information pour permettre à un utilisateur de vérifier les résultats et de les corriger le cas échéant.

L'approche que nous avons utilisée nous a permis de résoudre partiellement notre problème de boites d'informations nutritionnelles. Elle pourrait également servir à l'extraction de la liste des ingrédients ou encore à l'extraction des étiquettes de prix présentes dans les épiceries (Nom, prix, prix par 100 g.). Toutes ces informations pourraient être jumelées afin d'obtenir une basse de données très pratique afin de permettre au consommateur d'effectuer des choix éclairés lors de leurs emplettes hebdomadaires.

RECOMMANDATIONS

Nous présentons ici trois modifications majeures ainsi que méthode de validation que nous n'avons pas eu le temps d'implanter lors de ce projet et qui pourraient améliorer les performances de l'application.

Étant donné que Tesseract est une application libre, nous pourrions utiliser directement les fonctions présentes dans la librairie de Tesseract afin d'avoir accès à toute l'information contenue dans le module de reconnaissance, tel que les coefficients de vraisemblance des caractères ou des mots. En accédant aux fonctions internes de ce module, nous pourrions améliorer ces performances grâce à la connaissance à priori spécifique à notre problématique.

Nous pourrions également utiliser deux ou plusieurs images de la même boite pour créer une redondance d'information. Il serait donc possible de renforcer la confiance dans une information lorsque la reconnaissance de toutes, ou de la majorité, des images génère la même information. Dans le cas contraire, nous aurions un doute raisonnable afin de vérifier cette information manuellement.

Dans le même ordre d'idée, afin de créer une redondance d'information et d'augmenter le niveau de confiance, il serait également possible à partir d'une même image d'effectuer plusieurs reconnaissances à l'aide de différent moteur de reconnaissance (classificateur) (Kittler *et al.*, 1998; Tin Kam, Hull et Srihari, 1994) et de fusionner l'information recueillie.

Tel qu'il a été mentionné dans la discussion, il serait possible de valider les lignes possédant la configuration numérique 3 en comparant le pourcentage de l'apport quotidien avec avec la quantité de l'élément lui-même. Bien que le pourcentage de l'apport quotidien ne soit pas une information très précise (± 0,5%), il permettrait tout de même d'identifier des erreurs.

ANNEXE I

RÉSULTAT DE LA PREMIÈRE PHASE DE TEST

Vous trouverez à l'adresse ci-dessous le document qui contient tous les tableaux de résultats que nous avons générés durant la première phase de test.

http://profs.etsmtl.ca/jalandry/Recherche/Moffette/resultat_premiere_phase.pdf

ANNEXE II

RÉSULTAT DU TEST POUR DÉTERMINER LA VALEUR SEUIL DE LIGNE À UTILISER

Vous trouverez à l'adresse ci-dessous le document qui contient tous les tableaux de résultats que nous avons générés durant l'évaluation de la valeur seuil de ligne (voir Section 7.2).

http://profs.etsmtl.ca/jalandry/Recherche/Moffette/resultat_valeur_seuil_ligne.pdf

ANNEXE III

BANQUE D'IMAGE

Vous trouverez à l'adresse ci-dessous le répertoire Banque_image qui contient toutes les images de tests que nous avons utilisé pour effectuer nos tests.

http://profs.etsmtl.ca/jalandry/Recherche/Moffette/Banque_image/

ANNEXE IV

CODE SOURCE

Vous trouverez à l'adresse ci-dessous le répertoire Code_source qui contient tous les fichiers du code source de l'application. (voir Section 7.2).

http://profs.etsmtl.ca/jalandry/Recherche/Moffette/Code_source/

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Abutaleb, Ahmed S. 1989. « Automatic thresholding of gray-level pictures using two-dimensional entropy ». *Computer Vision, Graphics, and Image Processing,* vol. 47, n° 1, p. 22-32.

ACIA, Agence canadienne d'inspection des aliments. 2003. « GUIDE D'ÉTIQUETAGE ET DE PUBLICITÉ SUR LES ALIMENTS 2003 ».

aliments, Agence canadienne d'inspection des. 22 avril 2009. « Étiquetage nutritionnel, allégations nutritionnelles et allégations relatives à la santé ». In *Agence canadienne d'inspection des aliments*. <http://www.inspection.gc.ca/francais/fssa/labeti/nutrition-pagef.shtml>. Consulté le 21 avril 2010.

« Artificial neural networks : a neural network tutorial ». 2010. In *NeuroAI : Artificial intelligence technologies tutorial*. Consulté le 26 mai 2010.

Becel. 2010. *Becel : Quelle margarine becel choisir ?* En ligne <http://www.becelcanada.com/fr_ca/info-produits/quelle_margarine_becel_choisir.aspx>. Consulté le 14 avril 2010.

Bézier, Pierre. 1987. *Courbe et surface*. Paris: Hermès, 221 p.

Canada. 24 mars 2010. *Loi sur les aliments et drogues*. 1092 p.

Canada. 25 mars 2010. *Loi sur l'emballage et l'étiquetage des produits de consommation*.

Canada, Santé. 2002. « Que pensent les Canadiens et les Canadiennes de la nutrition? ». p. 2.

Cheriet, M., J. N. Said et C. Y. Suen. 1998. « A recursive thresholding technique for image segmentation ». *Image Processing, IEEE Transactions on,* vol. 7, n° 6, p. 918-921.

Cheriet, Mohammed, Nawwaf Kharma, Cheng-lin Liu et Ching Suen. 2007. *Character Recognition Systems: A Guide for Students and Practitioners*. Wiley-Interscience.

Chow, C. K., et T. Kaneko. 1972. « Automatic boundary detection of the left ventricle from cineangiograms ». *Computers and Biomedical Research,* vol. 5, n° 4, p. 388-410.

Crow, Franklin C. 1984. « Summed-area tables for texture mapping ». In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. p. 207-212. ACM.

Duda, Richard O., Peter E. Hart et David G. Stork. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Gadoua. 2010. *Gadoua:vitalité*. En ligne. <http://www.gadoua.qc.ca/vbread.html>. Consulté le 14 avril 2010.

Gonzalez, Rafael C., et Richard E. Woods. 1992. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., 716 p.

Gonzalez, Rafael C., et Richard E. Woods. 2002. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., 716 p.

Google. 2010. « Google Books Library Project ». En ligne. <http://books.google.com/googlebooks/library.html>.

Kharma, N., Kowaliw, T., Clement, E., Jensen, C., Youssef, A., Yao, J. 2004. « Project CellNet: Evolving an autonomous Pattern Recognizer ». *International Journal of Pattern Recognition ans Artificial Intelligence,* vol. 18, n° 6, p. 1039-1056.

Khotanzad, A., et Y. H. Hong. 1990. « Invariant image recognition by Zernike moments ». *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 12, n° 5, p. 489-497.

Kittler, J., M. Hatef, R. P. W. Duin et J. Matas. 1998. « On combining classifiers ». *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 20, n° 3, p. 226-239.

Landry, Jacques-André. 2007. *Notes de cours GPA669 : Vision robotique*.

Lu, Yi. 1995. « Machine printed character segmentation --; An overview ». *Pattern Recognition,* vol. 28, n° 1, p. 67-80.

Lu, Yi, Haist B., Harmon L., Trenkle J. et Vogt R. 1992. « An accurate and efficient system for segmenting machine-printed text ». In *U.S. Postal Service 5th Advanced Technology Conference* (November). Vol. 3, p. A-93-A-105. Washington D.C.

Mukundan, R., S. H. Ong et P. A. Lee. 2001. « Image analysis by Tchebichef moments ». *Image Processing, IEEE Transactions on,* vol. 10, n° 9, p. 1357-1364.

Natrel. 2010. *Natrel : Omega 3 ADH*. En ligne. <http://www.natrel.ca/french/whatsnew/NatrelOmega3ADH.html>. Consulté le 14 avril 2010.

nutrition, Institut national de la. 2001. « Nutrition: évolution et tendances. ».

Otsu, Nobuyuki. 1979. « A Threshold Selection Method from Gray-Level Histograms ». *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 9, n° 1, p. 62-66.

Rice, Stephen V., Frank R. Jenkins et Thomas A. Nartker. 1995. « The fourth annual test of OCR accuracy ». Las Vegas, University of Nevada.

Rice, Stephen V., George L. Nagy et Thomas A. Nartker. 1999. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, 194 p.

Sauvola, J., et M. Pietikäinen. 2000. « Adaptive document image binarization ». *Pattern Recognition,* vol. 33, n° 2, p. 225-236.

Sezgin, Mehmet, et Bülent Sankur. 2004. « Survey over image thresholding techniques and quantitative performance evaluation ». *Journal of Electronic Imaging,* vol. 13, n° 1, p. 146-168.

Shafait, Faisal, Daniel Keysers et Thomas M. Breuel. 2008. « Efficient implementation of local adaptive thresholding techniques using integral images ». In, sous la dir. de Yanikoglu, Berrin A., et Kathrin Berkner, 1. Vol. 6815, p. 681510-6. San Jose, CA, USA: SPIE. <http://link.aip.org/link/?PSI/6815/681510/1>.

Siedlecki, W. , et J. Sklansky. 1993. « On automatic feature selection ». In *Handbook of pattern recognition and computer vision*, sous la dir. de Chen, C. H., L. F. Pau et P. S. P. Wang. p. 984. World Scientific Publishing Co., Inc. <http://books.google.ca/books?hl=en&lr=&id=wnhwKoen4xAC&oi=fnd&pg=PA63&dq=On+automatic+feature+selection&ots=phmIJGWlZ9&sig=dPQIOhaJC1FaEfJb9TlYujD6uwo#v=onepage&q=On%20automatic%20feature%20selection&f=false>.

Smith, R. 1987. « THE EXTRACTION AND RECOGNITION OF TEXT FROM MULTIMEDIA DOCUMENT IMAGES ». University of Bristol, 155 p.

Smith, R. 2007. « An Overview of the Tesseract OCR Engine ». In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*. p. 629-633. IEEE Computer Society.

Smith, R. W. 2009. « Hybrid Page Layout Analysis via Tab-Stop Detection ». In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*. p. 241-245.

Smith, Ray, Daria Antonova et Dar-Shyang Lee. 2009. « Adapting the Tesseract open source OCR engine for multilingual OCR ». In *Proceedings of the International Workshop on Multilingual OCR*. p. 1-8. Barcelona, Spain: ACM.

Sohel, F. A., G. C. Karmakar et L. S. Dooley. 2005. « A generic shape descriptor using Bezier curves ». In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on* (4-6 April 2005). Vol. 2, p. 95-100 Vol. 2.

Sun-Hwa, Hahn, Lee Joon Ho et Kim Jin-Hyung. 1999. « A study on utilizing OCR technology in building text database ». In *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*. p. 582-586.

Tin Kam, Ho, J. J. Hull et S. N. Srihari. 1994. « Decision combination in multiple classifier systems ». *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 16, n° 1, p. 66-75.

Trier, O. D., et A. K. Jain. 1995. « Goal-directed evaluation of binarization methods ». *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 17, n° 12, p. 1191-1201.

Trier, Oivind Due, et Torfinn Taxt. 1995. « Evaluation of Binarization Methods for Document Images ». *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 17, n° 3, p. 312-315.

Unnikrishnan, Ranjith, et Ray Smith. 2009. « Combined script and page orientation estimation using the Tesseract OCR engine ». In *Proceedings of the International Workshop on Multilingual OCR*. p. 1-7. Barcelona, Spain: ACM.

Viola, Paul, et Michael J. Jones. 2004. « Robust Real-Time Face Detection ». *International Journal of Computer Vision,* vol. 57, n° 2, p. 137-154.