N° d'ordre : 1393 ANNÉE 1997

THÈSE

présentée

pour obtenir le titre de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

Spécialité : Traitement du signal

par

Baptiste MARCEL

ÉTUDE SUR LA LECTURE AUTOMATIQUE DE CODES BI-DIMENSIONNELS PAR TRAITEMENT D'IMAGE

soutenue le 17 décembre 1997 devant le jury composé de

Maurice BRIOT	Professeur UPS Toulouse	Président du jury Rapporteur
Dominique BARBA	Professeur IRESTE Nantes	Rapporteur
Michel CATTOËN	Professeur INP Toulouse	Directeur de thèse
Jean BAJON	Professeur émérite INP Toulouse	Examinateur
Philippe MARTHON	Maître de conférences INP Toulouse	Examinateur
Jean-Louis MASSIEU	Directeur R. & D. Intermec - TC Toulouse	Examinateur

« À notre connaissance, il n'existe qu'une seule espèce capable de rivaliser avec l'Homme. Tout le monde a déjà compris, il s'agit des codes Data Matrix. »

D'après « Pièges sur Zarkass » de Stefan Wul.

À ma famille :
mes frères Antonin et Yves, ma sœur Félicie,
mes parents Jean-Claude et Françoise
Kronos, Nicolas et Marion.

Avant propos

L'étude présentée dans ce mémoire a été effectuée au sein du groupe Signaux, Images et Communication (SIC) du Laboratoire d'Électronique (LEN7) de l'École Nationale Supérieure d'Électrotechnique, Électronique, Informatique, et Hydraulique de Toulouse (ENSEEIHT), en collaboration avec la société Réflexion + devenue Intermec - Technologies Corporation. Cette étude a été financée par Réflexion +/Intermec-TC, avec le soutien de l'Association Nationale de la Recherche Technique (ANRT) dans le cadre d'une Convention Industrielle de Formation par la REcherche (CIFRE).

Je remercie vivement monsieur Maurice Briot, professeur à l'UPS, de l'honneur qu'il m'a fait en acceptant d'être président du Jury et rapporteur. Je tiens à remercier monsieur Dominique Barba, professeur à l'IRESTE de Nantes d'avoir accepté de juger mon travail en tant que rapporteur, et monsieur Michel Cattoën, professeur à l'INPT et directeur de thèse, pour ses conseils et sa disponibilité pendant cette étude et lors de la rédaction de ce mémoire.

Je voudrais exprimer ma reconnaissance aux examinateurs de thèse : messieurs Jean Bajon, professeur émérite à l'INPT, Philippe Marthon, maître de conférence à l'INPT, et Jean-Louis Massieu, directeur de la société Intermec-TC. Je les remercie de l'intérêt qu'ils ont porté à mon travail.

Je tiens à remercier également la société Intermec-TC qui est à l'origine de cette étude, et en son sein Jean-Michel Puech pour la confiance qu'il m'a accordé et ses conseils dans les domaines que je maîtrise moins. Merci aussi à Nathalie, Annick, Patrice, Bernard, Francis, Véronique, Franck, John de m'avoir supporté toujours avec le sourire.

Je remercie le LAAS du CNRS et plus particulièrement le groupe RIA de ce laboratoire pour les ressources informatiques et bibliographiques mises à ma disposition pendant la première partie de cette étude.

Je remercie Henri Maurel, ingénieur d'étude, pour sa disponibilité et son aide technique. Pour leur contribution scientifique et technique, je remercie aussi Jean-Yves Tourneret, Roland Lagarrigue, Alban Duverdier. Je tiens également à remercier Corinne Mailhes pour l'effort qu'elle développe pour organiser la "vie sociale" du groupe.

Un merci particulier à Thierry Robert, pour sa gentillesse et sa patience, à Jean-Philippe Scotto et au hasard qui nous a mis dans le même bureau (il n'est pas donné à tout le monde d'avoir un voisin de travail qui partage la même culture et les mêmes passions pour les animés japonais, le cinéma fantastique et les séries télévisées). Merci à Sébastien Délestaing pour sa contribution au travail d'étude dans le cadre de son stage de DEA.

Enfin, je tenais à remercier les participants philanthropes et anonymes du réseau USENET, pour leur aide, publique ou privée, anglo-saxonne ou francophone, toujours à la fois compétente et généreuse. Je remercie aussi les auteurs, universitaires et autres informaticiens qui travaillent à mettre gratuitement à disposition du public par le réseau Internet des bibliothèques de codes de programme et de logiciels.

Table des matières

Introduction

Les informations inscrites sur des supports et destinées à être lues par des machines sont codées par des motifs géométriques selon des codes symboliques, tels les codes à barres que l'on rencontre sur la plupart des produits manufacturés.

Les codes symboliques bi-dimensionnels sont des nouveaux codes, sur lesquels les deux directions de la surface du symbole sont porteuses d'information. Les codes PDF et Data-Matrix en font partie.

Ces codes utilisent les deux dimensions du plan, à la manière d'un damier dont les cases constituent les éléments binaires et peuvent coder beaucoup plus d'informations que les codes conventionnels (dits linéaires) sur la même surface, en fonction de la redondance du codage appliquée. Les symbologies 2D ne sont pas plus coûteuses à l'impression que les codes linéaires. Les résolutions des imprimantes utilisées aujourd'hui sont suffisantes. Par contre, il est plus délicat de réaliser des lecteurs de codes 2D que des lecteurs de codes à barres. Ceci explique le peu d'applications sur le terrain malgré la forte demande du marché.

L'objet de l'étude est de concevoir des algorithmes de lecture de codes symboliques bi-dimensionnels de type Data Matrix par traitement d'image.

Les difficultés de traitement sont les suivantes :

- La lecture doit être insensible aux écarts angulaires entre l'appareil et le code, du moins dans une certaine marge (implication au niveau des algorithmes de repérage).
- La lecture doit s'effectuer dans un délai suffisamment court pour être négligeable pour un utilisateur humain (implication au niveau de la quantité de calculs des algorithmes et de l'implémentation de ces derniers).

Des recherches ont été entreprises au sein du groupe Signaux, Images et Communication (SIC) du Laboratoire d'Électronique de l'ENSEEIHT pour établir les algorithmes de base visant à détecter, localiser et lire un symbole dans une image. Ce mémoire en décrit les différentes étapes.

Le chapitre I décrit les codes symboliques et leurs caractéristiques en termes de traitement d'image. Les différents codes sont passés en revue, avec une description détaillée du code Data-Matrix. Puis le problème de recherche et de lecture de symbole dans l'image est présenté, ainsi que ses contraintes et ses difficultés. Les caractéristiques des symboles dans les images et les effets du processus d'acquisition (échantillonnage, distorsions…) sont observées.

Le chapitre II passe en revue les différentes méthodes de traitement d'image trouvées dans la littérature susceptibles d'être utilisées pour la recherche de symbole. La première partie présente les différentes méthodes étudiées, et en particulier les méthodes d'analyse indirectes (par transformations) et directes de segmentation par luminance ou texture. La seconde partie est consacrée aux méthodes de recherche de contours, en particulier celles utilisant la dérivation (gradient et laplacien). La troisième partie présente des méthodes de détection de lignes établies à partir des méthodes de détection de contour.

Le chapitre III présente les méthodes développées en vue de réaliser la recherche et la lecture de symbole. La première partie présente la recherche de symboles. Cette recherche utilise, selon un procédé détaillé, les détecteurs de contour et de lignes décrits au chapitre II et une méthode de validation par analyse de texture. La seconde partie décrit la procédure de lecture sur un symbole localisé, et décrit en détail les opérations effectuées sur l'image pour identifier les paramètres intrinsèques du symbole et son contenu.

Le chapitre IV permet d'évaluer la méthode en présentant le logiciel réalisé et les résultats obtenus. La première partie décrit le dispositif expérimental, le logiciel réalisé et la façon dont les différentes étapes de la méthode sont accessibles par l'utilisateur. La seconde partie présente les résultats expérimentaux. Nous y exposons les paramètres de la méthode et les valeurs choisies, des exemples de lecture sur des images variées, et des tests visant à évaluer l'efficacité de la méthode. Les différentes causes d'échec sont analysées en terme d'algorithmique et de traitement d'image. Dans la troisième partie, nous étudions les durées d'exécution du programme et de ses composantes, en fonction des paramètres de l'image (nombre de pixels, taille du symbole…).

La conclusion permet d'établir un bilan de l'étude et de présenter ses perspectives de continuation. Nous y dégageons les points critiques de la méthode et proposons des améliorations possibles.

Chapitre I

Codes symboliques et acquisition d'image

I.A. Les différents codes

Le principe des codes symboliques est de représenter un ensemble de données sous une forme lisible pour une machine. Il existe deux familles de codes qui sont les codes symboliques *linéaires* et les codes symboliques *bi-dimensionnels* (ou 2D). La forme la plus connue est le *code à barres* (de la famille des codes linéaires) ; les *codes matriciels* dont nous allons parler dans ce document font partie des codes bi-dimensionnels.

I.A.1. Codes linéaires

Les codes linéaires sont les codes dont les informations ne sont codées que selon une seule direction du plan. Il s'agit essentiellement de codes à barres. Le principe du codage par codes à barres est de représenter des chiffres ou des lettres à partir de barres blanches et noires de largeur variable. Le symbole est composé de mots élémentaires (groupes de barres) qui contiennent en général un élément de donnée et une information de position (ou de contrôle). Des mots de contrôle structurent le symbole selon un format spécifique. Chaque symbologie a ses propres règles quant à la structure du symbole et au codage de l'information et du contrôle dans les mots élémentaires. Ces codes ont une robustesse intrinsèque liée à la redondance de l'information qui peut-être lue par une analyse selon une droite quelconque traversant le symbole (redondance verticale).

Un code à barres peut-être lu par des barrettes de cellules photosensibles ou par le balayage d'un rayon laser associé à un capteur optoélectronique de type photodiode.

Parmi les codes à barres, le code EAN13 est le plus répandu et est celui qui se trouve sur les emballages des produits manufacturés ; il dérive du code UPC (*Universal Product Code*) qui est apparu au début des années 1970. Ce dernier permet de coder 11 chiffres. Un chiffre de contrôle dont la valeur est calculée sur les 11 chiffres est rajouté au moment du codage (pour permettre la détection d'éventuelles erreurs de décodage). Les 12 chiffres ainsi obtenus sont décomposés en deux groupes de 6 chiffres (champs), l'un situé à gauche, l'autre à droite.

Chaque chiffre est représenté par une combinaison de 2 barres noires et 2 barres blanches alternées dont les largeurs sont multiples d'une unité de distance (module) et dont la somme des largeurs est fixée à 7 modules (voir figure I-1). La représentation d'un même nombre diffère selon qu'il est dans la partie gauche ou droite du code. Par convention, un module noir représente la valeur un et un module blanc représente la valeur zéro. Le symbole est

encadré par deux motifs de trois barres structurant le symbole (voir figure I-2). La figure I-3 montre un exemple de code UPC.

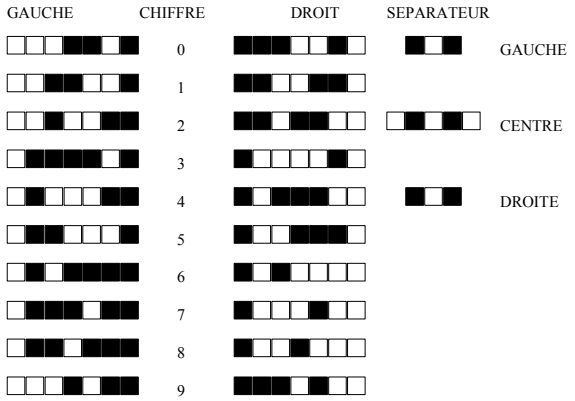


FIG. I-1 - *codage des données dans un code UPC*

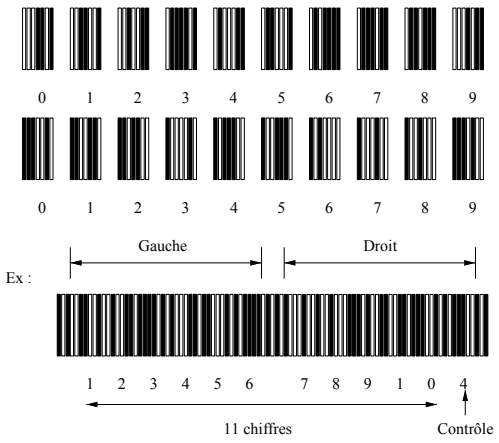


FIG. I-2 - *décomposition d'un code UPC*

FIG. I-3 - *exemple de code UPC*

I.A.2. Codes à barres empilées

Les codes à barres empilés (figure I-4) sont des codes bi-dimensionnels dérivés des codes à barres. Le symbole est constitué d'une matrice de codes à barres courts (des mots). Chacun de ces mots contient un élément de donnée et un élément de contrôle qui permet d'identifier sa place dans la matrice. Ce principe permet aux codes à barres empilées d'être lus en plusieurs fois par les technologies en usage dans la lecture de codes à barres. La redondance verticale disparaît en partie ; elle sera remplacée par une multiplication des codes de contrôle et l'utilisation de codages avec correction d'erreurs.

FIG. I-4 - *code à barres empilées (PDF)*

I.A.3. Codes matriciels

Le principe des codes matriciels est de représenter l'information par la présence ou l'absence de point sur une grille (non imprimée). Cette grille structure l'ensemble du symbole et dispense théoriquement d'avoir des mots élémentaires contenant des informations de contrôle. Des motifs particuliers structurent le symbole pour permettre de le localiser et de définir sa matrice. Il n'y a aucune redondance géométrique, et l'on peut utiliser des codes correcteurs

d'erreurs dont la redondance sera ajustée en fonction des applications. Les figures I-5 et I-6 montrent des exemples de codes matriciels.

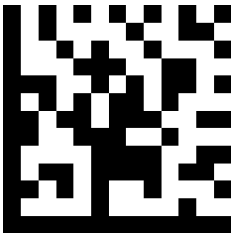


FIG. I-5 - *code matriciel Data Matrix*

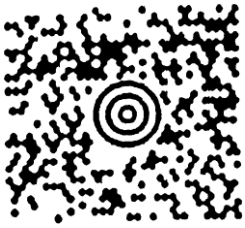


FIG. I-6 - *code matriciel Maxicode (à points)*

I.A.4. Spécificités des différents codes

Les performances des différents codes varient en fonction de la capacité de codage, de la densité et de la redondance de l'information. Les symbologies bi-dimensionnelles nécessitent les mêmes technologies d'impression que les codes linéaires ; par contre, les technologies de lecture doivent être adaptées ([HARM93]).

I.A.4.a. Lecture

Il est plus délicat de réaliser des lecteurs de codes bi-dimensionnels que des lecteurs de codes linéaires ; ceci explique le peu d'applications sur le terrain malgré la forte demande du marché.

Un code à barres est lu par balayage linéaire de la zone d'information ; ceci peut-être réalisé par une barrette de cellules photosensibles (CCD - *Charge Coupled Device*) ou par le balayage d'un faisceau laser associé à un œil électronique. Dans les deux cas, la réflexion de la lumière émise par le système de lecture est différente selon la présence ou non d'une barre sur le support du symbole, et l'élément photosensible reçoit un signal de niveaux alternativement haut et bas correspondant à l'alternance réflexion/absorption.

L'enregistrement de la luminance mesurée sur une ligne du champ de perception fournit un profil qui peut-être directement interprété comme un profil du code et décodé. La hauteur du symbole permet une redondance de l'information dans le cas où une zone du symbole est endommagée, et autorise une tolérance dans l'orientation de la ligne de lecture.

Dans les codes à barres empilées, la redondance verticale est moindre, et une redondance artificielle peut être introduite dans le codage. Chaque mot de code d'un code à barres empilées étant un petit code à barres qui contient des informations sur sa position dans la matrice, le symbole peut-être lu par morceaux, à l'aide des techniques linéaires des codes à barres, et reconstitué peu à peu au fur et à mesure que l'appareil de lecture lit chaque ligne de la matrice. Ceci s'applique surtout à la lecture par faisceau laser, mais ne permet pas une liberté dans la direction du balayage aussi grande que pour les codes linéaires.

Les symboles matriciels ne sont pas nécessairement structurés en mots élémentaires, et ne sont pas adaptés à la lecture par laser ni barrette CCD. La connaissance du symbole doit être globale lors de la lecture et en particulier inclure les éléments qui, dans le champ de vision, le rendent repérable et permettent de déterminer sa structure. La lecture de ces symboles doit être faite à partir d'un capteur surfacique tel qu'une caméra vidéo.

I.A.4.b. Capacité et redondance

Les codes linéaires ont une grande redondance, mais une faible quantité d'information par unité de surface. Les codes bi-dimensionnels ont un meilleur rapport information/surface, avec une redondance explicite réglable.

Considérons un code représenté par un symbole à deux niveaux (blanc et noir), imprimé et lu par des systèmes dont la résolution est limitée. Le plus petit élément significatif du symbole, appelé *point*, code au plus un bit. Si le symbole a une surface S, si la surface du point élémentaire est S_e, alors le nombre de bits que l'on peut coder est au plus S/S_e, ce qui peut être atteint avec un symbole matriciel ayant la densité maximale que permettent les technologies d'impression et de lecture.

Les codes linéaires assurent une bonne robustesse vis-à-vis des agressions sur le support du symbole. Si une tache apparaît et masque une portion de barre, l'information n'est pas perdue car elle reste présente sur le reste de la barre. Dans les codes matriciels, chaque point contient une information (un bit) qui est perdue si le symbole est taché en cet endroit. Les codes matriciels sont donc munis d'un système de redondance au codage qui permet de retrouver l'ensemble des données malgré la perte d'un certain nombre de bits. Cette redondance est introduite par des algorithmes de détection et de correction d'erreur tels celui de Reed-Solomon (méthode par polynômes) et a l'avantage de pouvoir être adaptée aux applications (quantité d'informations à stocker, niveau de protection désiré).

I.A.5. Code Data Matrix

Le code bi-dimensionnels le plus répandu est actuellement le code PDF qui est un code à barres empilées. Cependant, les codes à barres empilées ne représentent qu'une transition entre les codes linéaires et les codes matriciels, dont la forme la plus évoluée semble être le code Data Matrix. Nous allons étudier les caractéristiques des symboles de type Data Matrix, qui feront l'objet de notre étude.

Des spécifications préliminaires pour les symboles Data Matrix ont été publiées (Normes ECC 000-140 et ECC 200, [AIMT95]). La traduction des termes utilisée dans ce mémoire ne fait pas partie de ces spécifications et n'est donc pas normalisée.

Un symbole Data Matrix est constitué d'une *zone de données* (*data region*) dont les éléments sont des carrés appelés *modules* (ce mot désigne aussi la taille des éléments de la zone de données) répartis de manière régulière dans un tableau carré (ou rectangulaire pour la norme ECC 200). La zone de donnée est entourée par un motif de détection (*finder pattern*) lui-même entouré par une zone vide (*quiet zone*). Chaque module peut être marqué (*marked*) ou non-marqué (*unmarked*), et chaque module a une valeur. Un module marqué a la valeur 1, un module non marqué a la valeur 0. Dans les grands symboles de la norme ECC 200, il y a plusieurs zones de données séparées par un motif d'alignement (*alignment pattern*).

Le motif de détection circonscrit la zone de donnée et occupe un module de large. Deux côtés adjacents sont des lignes pleines (*solid lines*) et constituent le motif de détection en *L* (L *shaped finder pattern*). Les deux autres sont des alternances de modules marqués ou non-marqués (*alternating marked and unmarked modules*) formant le motif alternant de détection (*alternating finder pattern*).

La zone de données est aussi appelée *matrice*. Le motif alternant de détection est aussi appelé *zone d'horloge*, et la zone occupée par un module dans la matrice est aussi appelée

cellule (*cell*). Nous appelons *fréquence propre* (ou *fréquence*) d'un symbole le nombre de lignes ou de colonnes, et *fréquence spatiale* le nombre de cellules par unité de longueur (sur le support ou sur l'image). Nous appelons *période propre* et *période spatiale* (ou *période*) les inverses respectifs des fréquence propre et fréquence spatiale. La figure I-7 présente ces divers éléments.

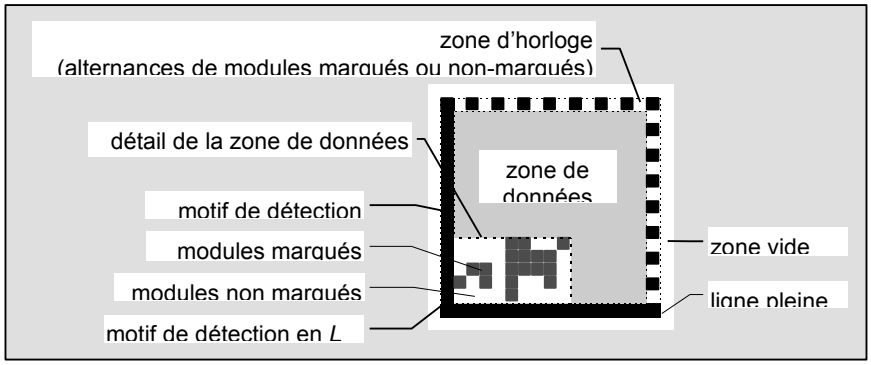


FIG. I-7 - *symbole Data Matrix*

Dans la norme ECC 000-140, les symboles sont carrés, avec un nombre de colonnes et de lignes impair allant de 9 à 49 (zone vide non comprise).

Dans la norme ECC 200, les symboles peuvent être rectangulaires, avec un nombre pair de colonnes et de lignes. Les symboles carrés ont des tableaux de taille 10×10 à 144×144, les symboles rectangulaires, de taille 8×18 à 16×48 (zone vide non comprise).

Les quantités de cellules marquées et non marquées sont globalement équilibrées dans un symbole.

L'étude réalisée porte sur des symboles carrés de fréquence impaire, conformes aux normes proposées par I.D.Matrix en 1992 (voir figures I-8 à I-10). Dans ces symboles, la zone d'horloge commence et finit par une cellule marquée ; cette structure impose à la fréquence d'être impaire.

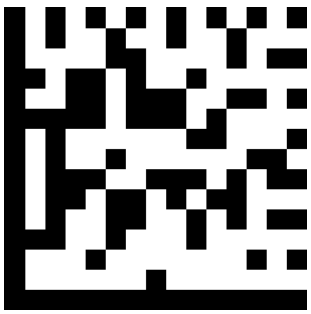


FIG. I-8 - *symbole Notnumb15*

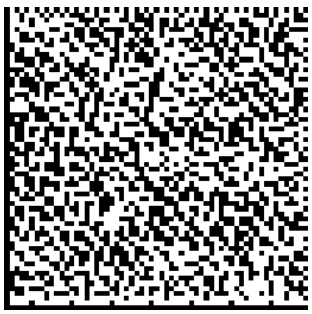


FIG. I-9 - *symbole Clarm61*

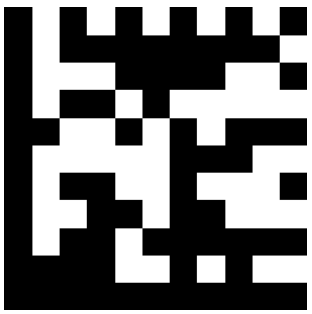


FIG. I-10 - *symbole CR11*

I.B. Étude de l'image d'un symbole

Nous allons étudier les caractéristiques des images de symbole acquises par caméra. Les principes de formation d'image de symboles, selon les type et position relative des supports de symbole, éclairage et caméra ont été détaillées dans [RIYA97]. Les caractéristiques du symbole, de la matrice de pixels du capteur et des conditions d'acquisition entraînent diverses propriétés de l'image numérisée.

Pour le texte qui suit, nous modélisons une image de la manière suivante.

Une image est une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$, où l'ensemble de départ $\mathbb{R}^2$ représente la projection d'une scène, et l'ensemble d'arrivée $\mathbb{R}$ l'éclairement reçu sur le capteur pour chaque point de la scène à travers l'optique. Cette fonction est en général restreinte à une surface du plan, généralement rectangulaire, appelée *support de l'image*. Lorsqu'une image est numérique, c'est une fonction de M×N dans D où M et N sont des intervalles de $\mathbb{N}$ et D un intervalle de $\mathbb{Q}$. L'usage étant de considérer l'image comme une source lumineuse, lorsqu'elle est affichée sur un écran, nous utiliserons le terme de *luminance* pour désigner la valeur associée à chaque pixel (en toute rigueur, un éclairement). Une valeur élevée (resp. faible) de luminance correspond à une zone claire (resp. foncée) de l'image.

Le support de l'image est un ensemble de pixels répartis dans une matrice rectangulaire de taille $N_i \times N_j$ (N_i est le nombre de lignes, N_j le nombre de colonnes). Par extension, l'image désignera aussi l'ensemble des pixels de la matrice, chacun associé à une luminance (scalaire).

Les unités de longueur sur l'image sont le pixel (px) et l'unité d'image (ui). L'unité d'image est telle que l'image mesure 1 ui de côté. Si les pixels ne sont pas carrés, on distingue les pixels de lignes (px_j) des pixels de colonne (px_i). Si la matrice n'est pas carrée ($N_i \neq N_j$), on distingue l'unité d'image en hauteur (ui_i) et l'unité d'image en largeur (ui_j). Afin d'éviter les confusions, nous limiterons le plus possible l'emploi de ces unités anisotropes. Si nous prenons par exemple une matrice carrée de 256 pixels de côtés, dotée de pixels carrés, nous avons $N_i=N_j=1$ ui$=256$ px. Dans la même image, pour les unités de fréquence (ui^{-1} et px^{-1}), nous avons 1 $\mathrm{ui}^{-1}=1/256$ px^{-1}.

La fréquence propre du symbole est notée f (f_i et f_j, nombres de lignes et de colonnes pour les symboles non carrés). Contrairement à cette dernière, la fréquence spatiale (notée f_s) est dépendante des paramètres de la prise de vue et s'exprime en inverse d'une unité de longueur (ui^{-1} ou px^{-1}).

Nous notons T et t respectivement la taille et la période spatiale en pixels du symbole sur son image. Nous avons $T=t\times f$.

Par exemple, avec une résolution de $(N_i\times N_j)=(256\times256$ [en pixels]), un symbole S_7 (voir figures I-11 et I-12) de fréquence propre $f=17$ occupant 47 pixels de côté a une taille de $T=47$ px$=0{,}18$ ui, un module de $t=2{,}76$ px$=0{,}011$ ui, une fréquence spatiale $f_s=17/47$ $\mathrm{px}^{-1}=92{,}60$ ui^{-1}. Le symbole S_3 de 21 cellules de côté (de fréquence propre 21 plus élevée que celle de S_7) occupant 73 pixels de côté a une fréquence spatiale $0{,}29$ $\mathrm{px}^{-1}=73{,}64$ ui^{-1} plus basse que celle de S_7.

Un symbole dans l'image est rarement parfaitement carré (distorsions projectives, optiques… voir § I.I.B). Dans ce cas, nous estimons T par la moyenne des longueurs des côtés calculées par les distances entre les pixels formant les coins, et t par T/f.

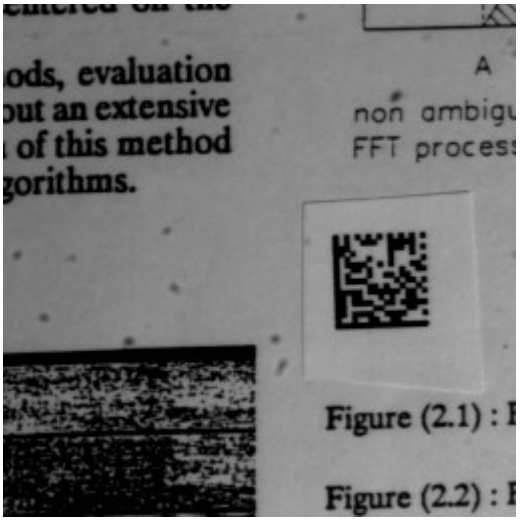


FIG. I-11 - *image du symbole* S_7 *(fréquence 17)*

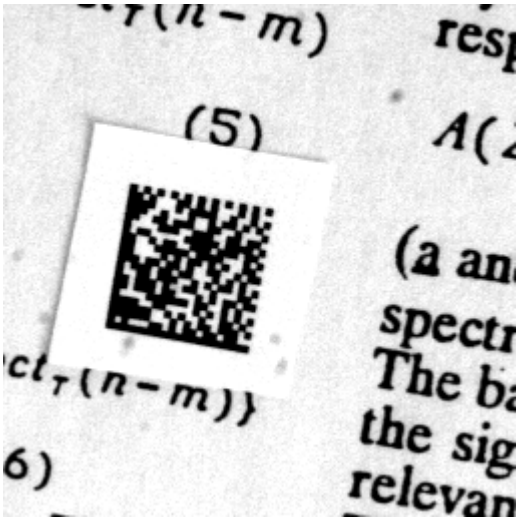


FIG. I-12 - *image du symbole* S_3 *(fréquence 21)*

Le motif de détection ne portant pas d'information, si *f* est la fréquence, alors *f*-2 est le nombre de lignes (et de colonnes) de la zone de données, et $(f\text{-}2)^2$ est le nombre de bits contenus dans le symbole.

I.B.1. Aspect de l'image en luminance

I.B.1.a. Réponse en amplitude

Dans le cas idéal, le symbole est seul dans la scène, et le support de symbole est de luminance uniforme en dehors des zones marquées. Dans ce cas là, nous appelons le *blanc dans la scène*, la zone du support du symbole en dehors du symbole, et le *blanc de l'image*, l'image du blanc de la scène.

L'éclairement du blanc dans la scène est rarement parfaitement homogène, d'une part parce qu'il existe des sources de lumières primaires ou secondaires dans l'environnement du capteur et d'autre part parce que la source de lumière attachée au capteur (s'il y en a une) ne génère pas un faisceau parallèle (elle peut-être ponctuelle) et ne fait pas face de manière parfaitement normale au support.

La luminance du blanc de l'image dépend de l'éclairement de la scène et de la position du capteur par rapport au symbole. Même si l'éclairement de la scène est homogène, le blanc de l'image ne le sera donc pas forcément.

À l'intérieur d'une même image, nous pouvons avoir des niveaux de luminances différentes pouvant correspondre à des luminances identiques sur le support du symbole.

La caméra elle même (optique et capteur) ne réagit pas uniformément à l'éclairement reçu. Les différences de réponses sont dues à l'optique d'une part (phénomène de vignetage qui dépend des objectifs) et à la disparité des éléments photosensibles (taille des pixels, réponse à l'éclairement…).

Nous appelons *profil de ligne* ou *de colonne* la luminance des pixels représentée le long d'une ligne ou d'une colonne. L'image de la figure I-13 montre la réponse en intensité pour un symbole à l'éclairement non homogène. Les figures I-14 et I-15 montrent des profils de ligne de cette image. On voit sur la figure I-14 que les intensités varient de 82 à 88 pour les cellules marquées et de 185 à 216 pour les non marquées.

La figure I-15 qui représente un profil de cette image en un endroit où le blanc de la scène n'est pas uniforme montre bien la disparité de la luminance qui dénote un éclairement plus important en périphérie du champ de vision qu'en son centre.

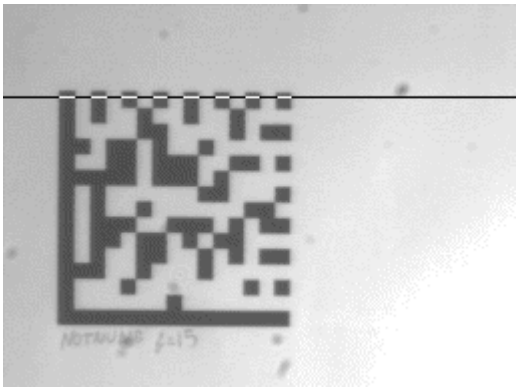


FIG. I-13 - *image d'un symbole à l'éclairement non homogène (la ligne noire représente la ligne utilisée pour le profil de la figure I-14)*

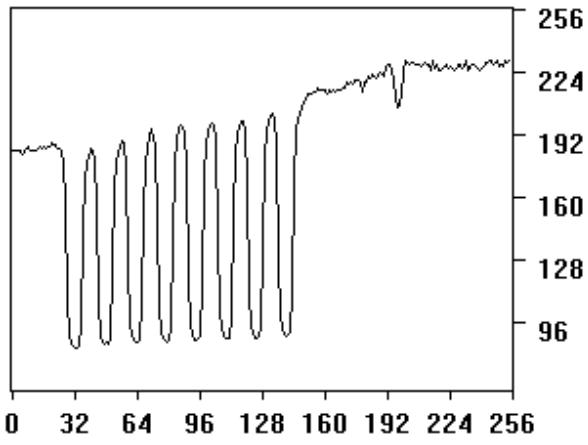


FIG. I-14 - *profil de l'image I-13 le long de la zone d'horloge*

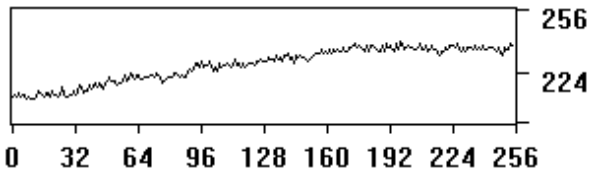


FIG. I-15 - *profil de l'image I-13 mesuré dans le blanc de l'image*

I.B.1.b. Réponse en fréquence et échantillonnage

Sur un symbole imprimé, la transition entre une cellule marquée et une cellule non marquée est franche et nette lorsqu'elle est observée par l'œil, et est marquée par le changement de niveau sur le support du symbole (présence ou absence d'encre).

Au niveau de la caméra, chaque pixel photosensible a une taille non nulle, et fournit une réponse proportionnelle à l'éclairement moyen qu'il reçoit, ce qui provoque un filtrage passe-bas. Cette caractéristique du capteur est la Fonction de Transfert de Modulation (FTM) qui décrit l'amplitude de sa réponse à un éclairement périodique (de fréquence f_s) en fonction de sa fréquence ; elle est de la forme $\sin(f_s)/f_s$ (sinus cardinal).

I.B.1.b.i) Effet sur la dynamique à l'intérieur d'un symbole

La figure I-16 représente un profil le long de la zone d'horloge de l'image d'un symbole de fréquence spatiale élevée (de l'ordre de 0,5 px^{-1}). Nous voyons que la forme du signal

diffère significativement de la forme en créneaux théorique, avec des irrégularités dans les amplitudes, et une dynamique moindre (l'intensité du blanc est bien plus élevée en dehors du symbole qu'à l'intérieur sur les cellules non marquées).

Nous voyons apparaître sur ce profil des harmoniques en basse fréquence qui dénotent du faible battement entre la fréquence d'échantillonnage (2 pixels par cellule) et la fréquence limite (1 pixel par cellule).

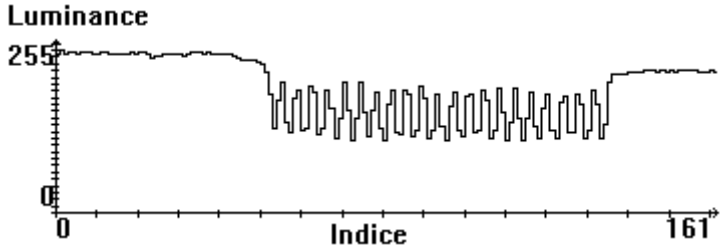


FIG. I-16 - *profil d'une image sur un symbole de haute fréquence le long de la zone d'horloge*

I.B.1.b.ii) Conséquences sur la largeur des contours des cellules

Si par projection, un pixel est contenu tout entier dans une cellule, le pixel associé dans l'image donne la luminance de la cellule ; si le pixel est à cheval sur deux cellules, le pixel associé donne une luminance intermédiaire, qui est la moyenne entre les deux luminances des cellules marquées et non marquées, pondérée par la surface qu'occupent les parties de cellule sur le pixel. Statistiquement, le bord d'une cellule traverse des pixels de l'image. Si l'optique est correctement focalisée et si la transition sur le support du symbole est nette, la transition sur l'image occupe un pixel de large.

Les figures I-17 à I-18 montrent des agrandissements sur des images de symbole. Nous voyons figure I-17 que les transitions entre le noir et le blanc apparaissent en gris, et occupent un pixel de large. Nous visualisons ces valeurs des luminances intermédiaires sur la courbe de la figure I-18.

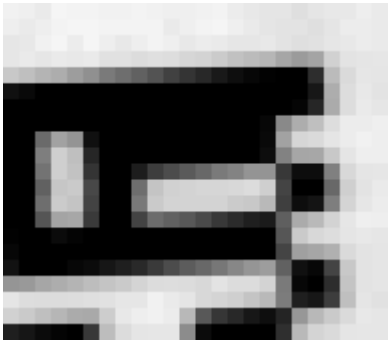


FIG. I-17 - *agrandissement de l'image d'un symbole*

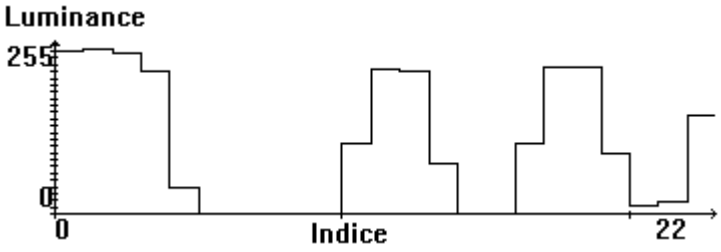


FIG. I-18 - *profil de colonne mesurée au milieu de l'image de la figure I-17*

I.B.1.b.iii) Effets de l'échantillonnage

L'image acquise par la caméra est une forme échantillonnée de la scène, à raison d'un échantillon par pixel. Lorsque nous analysons l'image, nous essayons de reconstituer la matrice du symbole et de déterminer la valeur de chaque cellule. Pour réaliser cette reconstitution, il faut au moins un pixel par cellule. Cette fréquence d'échantillonnage est théoriquement suffisante dans le cas d'un échantillonnage idéal.

I.B.1.b.iv) Combinaison des effets

L'échantillonnage réalisé par la caméra mesure pour chaque échantillon la moyenne de l'éclairement reçu sur la surface du pixel photosensible. Cette opération réalise un filtrage passe-bas qui se combine avec la fonction d'échantillonnage dans les images obtenues.

Nous pouvons voir ceci sur les figures I-19a à I-19c. Lorsque la taille des cellules diminue, il devient difficile de lire la valeur des cellules. La limite se situe entre 1 et 2 pixels par cellule.

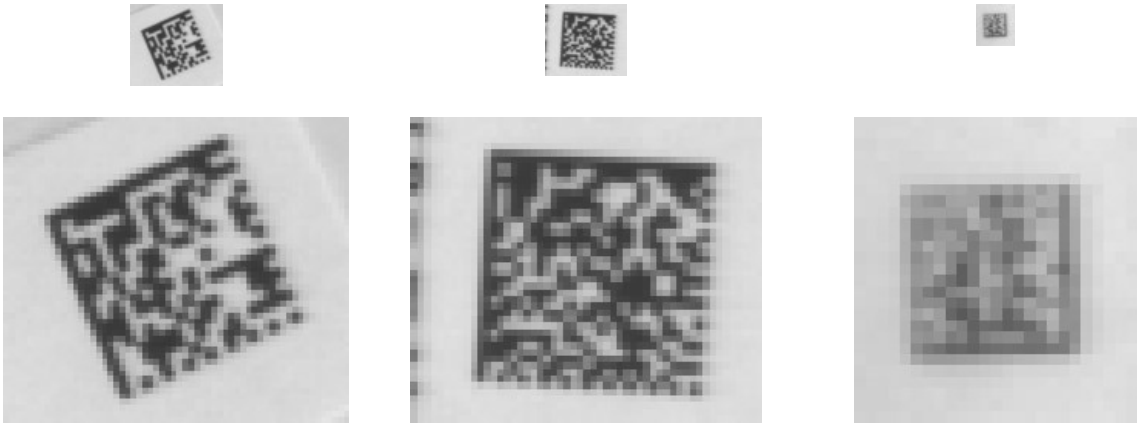


FIG. I-19a - *f=17, t=2,1 px* FIG. I-19b - *f=21, t=1,6 px* FIG. I-19c - *f=15, t=1 px*

FIG. I-19 - *symboles de faible fréquence spatiale : image à l'échelle 0,2 mm/px (en haut) et image agrandie (en bas)*

I.B.2. Distorsions géométriques

Les sources de distorsion géométriques sont de deux types : celles liées à l'optique de la caméra d'une part et celles liées à la transformation projective d'autre part.

Les distorsions optiques sont principalement des distorsions radiales de type coussin ou tonneau (voir figure I-20), et interviennent lorsque nous utilisons des optiques à faible focale (inférieures à 5 mm). Bien que ceci ne soit pas entré dans le champ de notre étude, il est possible de mesurer les paramètres intrinsèques de la caméra et le coefficient de distorsion radiale par calibrage ([ORTE91]), puis de modéliser et corriger les distorsions optiques. Ce type de correction calcule la luminance en chaque pixel à partir de la luminance du point équivalent dans l'image à corriger. Cette luminance est obtenue par interpolation à partir des valeurs des pixels voisins. Cette interpolation provoque un filtrage passe-bas qui, dans notre application, augmenterait la période spatiale limite des symboles.

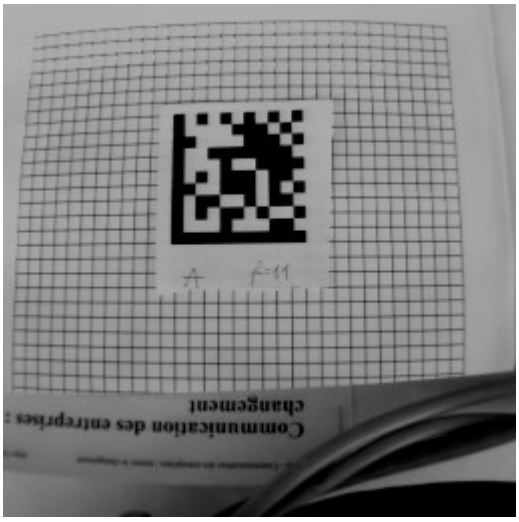


FIG. I-20 - *exemple de distorsions optiques de type tonneau*

Selon le modèle sténopé, les distorsions projectives se modélisent par la projection de l'espace (la scène) à travers un diaphragme ponctuel sur un plan (plan image). Si l'objet est dans un plan parallèle au plan image, cette perspective est cavalière : il n'y a pas de distorsion sur l'image symbole. Lorsque le plan de l'objet est incliné par rapport au plan image, les droites supports des bords du symbole convergent vers des points de fuite, le carré ou rectangle support du symbole se déforme (voir figure I-21), et la taille des cellules dans l'image varie selon une fraction rationnelle dont on peut déterminer les coefficients par la détermination du point de fuite (voir la démonstration dans l'annexe B).

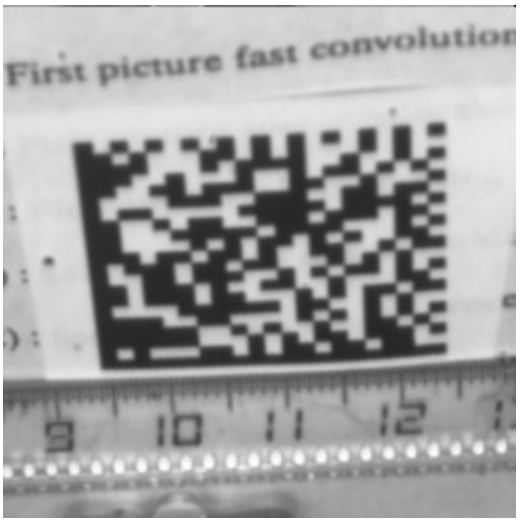


FIG. I-21a - *Symbole S_3 (f=21)*

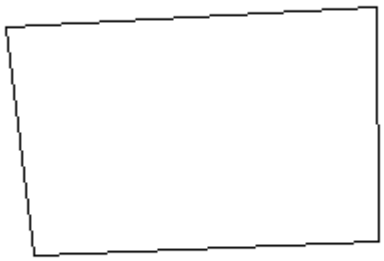


Fig. I-21b - *quadrilatère support du symbole de la figure I-21a*

FIG. I-21 - *exemple de distorsion projective (angle d'incidence de la caméra : 50°)*

I.C. Présentation du problème

L'objet de l'étude est de concevoir une méthode de lecture de codes symboliques bi-dimensionnels de type Data-Matrix par traitement d'image. La lecture consiste à déterminer les caractéristiques intrinsèques du symbole (fréquence) et les valeurs de chaque cellule. Le décodage du message contenu dans le symbole ne fait pas partie de notre étude.

I.C.1. Décomposition du problème

Nous pouvons décomposer le problème en plusieurs étapes et sous-étapes. La lecture se fait à partir d'une image acquise par une caméra matricielle vidéo. La première étape est la recherche de symbole ; elle peut-être décomposée en deux parties : détection et localisation.

- Recherche de symbole :
  - Détection
  - Localisation

La détection consiste à déterminer si un symbole se trouve dans l'image. La localisation détermine la position et l'orientation du symbole ainsi détecté. Détection et localisation peuvent être confondues.

La seconde étape est la lecture ; elle peut-être décomposée en deux parties : analyse et échantillonnage.

- Lecture :
  - Analyse
  - Échantillonnage

Dans l'étape de lecture, l'analyse consiste à étudier le symbole pour en tirer ses principales caractéristiques, notamment sa fréquence. L'échantillonnage consiste à lire en chaque cellule sa valeur afin de reconstituer la zone de données.

Une fois l'ensemble des données reconstitué, les étapes ultérieures, qui ne font pas partie du champ de notre étude, sont le traitement éventuel par les algorithmes de détection et de correction d'erreur, et le décodage afin d'en extraire le message codé.

I.C.2. Contraintes

Les fonctionnalités élémentaires des lecteurs traditionnels de codes linéaires doivent être préservées dans la lecture des codes bi-dimensionnels, au niveau de la qualité des résultats, de la vitesse et de l'ergonomie. Le lecteur doit pouvoir lire le symbole situé dans une certaine limite de distance (de quelques centimètres à près d'un mètre), avec une liberté relativement grande de présentation ; il doit, dans une certaine marge, être insensible aux écarts angulaires entre l'appareil et le code, et réaliser la lecture dans un délai suffisamment court pour être négligeable pour un utilisateur humain (nous parlerons de traitement en *temps réel*) avec une robustesse permettant une bonne lecture dans des conditions difficiles (code déformé, certaines parties non lisibles...).

Dans les symbologies à barres, pour qu'un mot du symbole soit lu, il faut que le faisceau le balaie latéralement de part en part. Si l'orientation du balayage est inclinée par rapport

à celle du symbole, les deux orientations doivent rester suffisamment proches pour que tout le mot soit traversé. Dans les lecteurs où le faisceau laser balaie l'espace dans plusieurs directions successives, le symbole linéaire peut-être lu quelle que soit sa présentation. Cette tolérance, obtenue ici par des moyens géométriques (forme du symbole) ou mécaniques (balayage du faisceau dans plusieurs directions), doit être aussi présente pour la lecture des symboles bidimensionnels et doit être obtenue à partir d'algorithmes de traitement d'image.

I.C.3. Obstacles

Les problèmes rencontrés sont de divers ordres. Le principal problème est de pouvoir détecter et lire le symbole quelles que soient sa fréquence et sa taille. Ceci implique une dynamique très large dans les fonctionnalités de recherche et de lecture qui devront être opérationnelles aussi bien sur de petits symboles à haute fréquence que sur de grands symboles à basse fréquence.

Le contraste de l'image peut être faible si la qualité d'impression du symbole est médiocre, ou si le symbole est placé trop loin de la source lumineuse associée au lecteur. La recherche, puis la lecture, est rendue difficile dans ce cas.

Le blanc de l'image peut être hétérogène et provoquer des variations de luminance dans l'image numérisée entre des pixels représentant des cellules de valeurs identiques (marquées ou non marquées).

Les symboles sont structurés selon des formes géométriques droites. L'image présente des distorsions optiques (en fonction des focales utilisées) qui ont pour effet de courber ces lignes droites, et des distorsions projectives (que nous modélisons) qui rendent convergentes des lignes qui ne devraient pas l'être. Ces effets gênent la recherche tout comme la lecture.

I.D. Conclusion

Les solutions aux problèmes de recherche et de lecture de symbole sont évidemment liées aux caractéristiques des codes utilisées et de leurs images. Les codes matriciels doivent être lus par des capteurs matriciels avec des techniques de traitement d'images. Ils ne peuvent pas êtres lus selon les techniques développées pour les codes linéaires (laser, barrette CCD).

Les codes utilisés comme support de cette étude sont les codes Data Matrix. Nous avons décrit ces codes et étudié les caractéristiques des images de symbole en termes quantitatifs et en termes de luminance et de fréquence spatiale.

Nous avons posé le problème à réaliser et ses contraintes. Nous sommes donc en mesure de rechercher dans la littérature les approches proposées par divers auteurs afin de connaître l'état de l'art et de s'en inspirer pour concevoir nos méthodes de recherche et de lecture de symbole.

Chapitre II

Recherche bibliographique

Dans ce chapitre, nous présentons le résultat des recherches bibliographiques de méthodes susceptibles d'être transposées à notre étude. Ces méthodes ont été étudiées dans le cadre des contraintes auxquelles nous sommes soumis (§ I.C.2).

Notre problème nécessite un traitement rapide, en temps réel, d'une image contenant un symbole de taille et d'orientation quelconque, dans une scène qui peut être complexe et d'un contraste faible.

II.A. Différentes approches

II.A.1. Classification des différentes approches

Les différentes approches peuvent être associées par groupes en fonction des méthodes mises en œuvre. La plupart des méthodes relèvent de la segmentation, qui consiste à analyser l'image afin d'identifier les points ou zones de l'image ayant certaines caractéristiques. L'identification des caractéristiques de symboles permet de les rechercher dans l'image.

Certaines méthodes de segmentation d'image sont basées sur des approches indirectes par transformation dans un autre domaine où les propriétés apparaissent sous d'autres formes (transformation de Hough par exemple) ; d'autres méthodes de segmentation se basent sur la luminance des pixels, et y appliquent des traitements comme le seuillage, l'analyse de texture ou la recherche de contours. Cette dernière peut d'ailleurs être complétée par une recherche de chaînes de contours et de lignes droites de l'image.

La classification que nous avons adoptée est la suivante :

Correspondance de formes (*template matching*)
Segmentation
    Approches indirectes
        Transformation en ondelettes
        Transformation de Hough

Approches directes
- Seuillage
- Segmentation par texture
- Recherche de contours
  - Cartes de contours
  - Chaînes de contours
  - Lignes droites

Dans la première partie (II.II), nous étudions les premières de ces méthodes : correspondance de formes, méthodes de segmentation indirectes, méthodes de détection directes liées au seuillage et aux textures. Dans la seconde partie (II.II), nous étudions d'une manière plus détaillée la méthode de détection de contours. Dans la troisième partie, nous étudions les méthodes de détection de chaînes de contours et de lignes droites (II.II), basées sur les méthodes de détection de contours.

En segmentation d'image, nous sommes amenés à identifier pour chaque point ou zone de l'image des attributs, que nous représentons dans des cartes.

Une *carte* est une fonction similaire à l'image (même ensemble de départ représentant la projection d'une scène), contenant pour chaque pixel non pas l'éclairement reçu par le capteur mais un certain nombre d'attributs, par exemple la couleur, l'intensité ou la direction des contours.

II.A.2. Correspondance de formes

La correspondance de formes (*template matching*) est une méthode qui consiste à rechercher dans une image une corrélation entre des groupes de pixels et une certaine forme prédéfinie dans un modèle (*template*). [VAIL94] utilise des réseaux de neurones pour réaliser cette recherche.

Cette méthode permet rarement une liberté de rotation ou d'échelle dans la forme à détecter ([BENT94]), et est en général trop sensible aux défauts d'éclairage ([SUET92] p. 15). Cette méthode est donc inadaptée à notre problème.

II.A.3. Transformation en ondelettes

La transformation en ondelettes (*wavelet transformation*) est une méthode qui permet d'extraire certaines caractéristiques multi-fréquence, multi-résolution par une transformation globale adaptée aux caractéristiques recherchées. La transformation est appliquée sur des fenêtres d'analyse sur lesquelles ces caractéristiques sont évaluées. Des méthodes de reconnaissance de caractères (avec réseau de neurones : [LAIN93]), de détection de coins ([LEES93], [NEVA80]), de seuillage ([OLIV94]) sont basées sur la transformation en ondelettes. Les transformations en ondelettes pourraient donner de bons résultats pour notre problème, car on ne connaît pas *a priori* la fréquence spatiale des caractéristiques que l'on cherche.

On sait que si le symbole a une fréquence spatiale élevée, il faudra une localisation à grande résolution (petites fenêtre d'analyse), alors que si le symbole a une fréquence spatiale faible, la localisation pourra avoir une résolution moindre (grande fenêtre d'analyse). La résolution nécessaire (nombre de fenêtres d'analyse dans l'image, inversement proportionnel à leur taille) est proportionnelle à la fréquence spatiale du symbole (inversement proportionnelle à la période spatiale). Une analyse espace - échelle par une fonction de décomposition adaptée

à la texture cherchée pourrait permettre de localiser d'une manière assez directe les symboles quelle que soit leur taille dans l'image et quelle que soit leur fréquence spatiale. Cependant, le principal inconvénient de la transformation en ondelettes est la grande quantité de calculs nécessaires qui justifie d'écarter cette méthode.

II.A.4. Transformation de Hough

II.A.4.a. Généralités

La transformation de Hough (*Hough transformation*, prononcer *Hœuf*) a été conçue initialement pour localiser les alignements de pixels dans les images. Des algorithmes récents permettent de repérer les polygones, distinguer les lignes de contours des alignements non jointifs, filtrer les lignes par longueur et orientation. Ces méthodes peuvent soit faire appel à des interprétations particulières des résultats de la transformation de Hough classique ([ROTH94] fait appel à la programmation génétique), soit généraliser les principes de Hough à des équations et formes autres que les droites du plan (transformation de Hough généralisée ou *generalized Hough transform*, [BALL81], [ROTH93], [ROTH94]). La transformation de Hough est assez coûteuse en temps de calcul ; elle est largement étudiée et plusieurs auteurs proposent des implémentations de simplicité et d'efficacité inégales ([DUDA72], [HUNT88], [ILLI88], [LIEC90], [GUPT93], [PALM93], [LEAV93], [FORE94]).

II.A.4.b. Variantes de la transformation de Hough

Dans [DUDA72], les auteurs obtiennent des résultats convaincants avec la transformation de Hough généralisée pour les formes mathématiquement simples telles que le cercle ou l'ellipse. L'application de la transformation de Hough généralisée doit tenir compte des paramètres qui définissent un contour de symbole Data-Matrix. Ces paramètres sont : position (deux dimensions), orientation (une dimension), taille (une dimension) et fréquence (une dimension). On peut ramener ces cinq dimensions à quatre en omettant la fréquence comme paramètre de détection et en utilisant uniquement les zones de localisation du symbole. Mais même à quatre dimensions, la transformation de Hough généralisée n'est pas exploitable dans le cadre de notre projet car trop coûteuse en volume de données.

Dans [ILLI88], les auteurs présentent plusieurs modifications de la transformation de Hough, qui pourraient éventuellement être intéressantes : choix des paramètres et stockage des données adapté au problème, analyse hiérarchique (algorithme *coarse-to-fine*), accélération du calcul (*fast Hough transform*), pondération des votes par la valeur du gradient, utilisation de la direction du gradient, utilisation de la forme des pics, implantation sur processeurs parallèles. L'auteur de [LEAV93] présente lui aussi des améliorations éventuellement intéressantes sous la forme d'étiquettes sur les votes (*look-up tables*).

L'algorithme présenté dans [GUPT93] semble le plus abordable. Cependant, malgré tous leurs aménagements, les algorithmes présentés dans ces publications nécessitent encore de grandes quantités de calcul et des volumes de données importants.

II.A.5. Seuillage d'image

La méthode consiste en une segmentation de l'image en classes déterminées par la luminance des pixels. Si le nombre de classes de luminance est supérieur à deux, on parle de seuillage multi-niveaux ([LEEK95]). Si le seuil n'est pas le même sur toute l'image, on parle de seuil dynamique (dans [KITT86], le seuil est calculé par zone en fonction de l'image).

Dans l'hypothèse [TRIV90] où les cellules marquées du symbole ont des luminances suffisamment homogènes et distinctes du blanc, les zones claire et sombre correspondent à deux pics de l'histogramme des luminances. Un seuillage entre les deux pics permet de séparer les zones.

Si nous utilisons un seuil dynamique, la taille du voisinage sur lequel on calcule le seuil est un problème crucial, car on ne connaît ni la taille ni la fréquence spatiale du symbole. Si ce seuillage devait être utilisé, il ne devrait l'être que pour l'étape de détection et localisation, et non pas pour la lecture (des essais ont montré que la zone intérieure du symbole est très difficile à seuiller en raison de la réduction de la dynamique évoquée au § I.B.1.b.i).

II.A.6. Recherche par textures

La *texture* d'une image (ou d'une partie de l'image) est un ensemble de caractéristiques de l'image invariantes par translation d'une fenêtre d'observation de taille donnée. L'espace des textures est divisé en classes d'équivalence.

L'identification des textures se fait en utilisant des *critères* permettant d'attribuer à chaque zone telle ou telle classe, et amène à établir la carte des textures. Les critères sont en général calculés sur un ensemble de pixels appelé *texel* (de *texture element*). Le fait d'attribuer une certaine classe à un texel s'appelle la *classification*.

La *segmentation d'image* par les textures ([PALP93]) consiste à diviser l'image en régions dont la texture appartient à la même classe. Si la taille des texels n'est pas constante, on parle de texels *dynamiques*.

Le choix des critères est crucial. Ces critères peuvent être basés sur l'image, ou sur des formes dérivées (carte de contours…).

Cette méthode, appliquée à la détection de symbole, consiste à repérer dans l'image les zones dont la texture est semblable à celle d'un code symbolique, sans se préoccuper des différences de texture dans les zones ne relevant pas du symbole. C'est une segmentation binaire.

Les propriétés que nous pouvons utiliser pour le Data Matrix sont la forme bimodale de l'histogramme en niveaux de gris (avec autant de noir que de blanc), la grande quantité de petites lignes de contours de longueur multiple de la période du symbole dans l'image, l'orthogonalité de toutes les lignes…

Plusieurs auteurs décrivent divers critères permettant de caractériser les scènes. Dans [CHEN79], les auteurs utilisent les valeurs de la matrice de cooccurrence comme mesure de texture ; ceux de [CASA92] proposent une série de critères de texture pour des systèmes de classification rapide. Les paramètres de la texture prennent en compte le nombre de fronts abrupts, la variance du gradient (*granularité*), la quantité de coins… Cette méthode est détaillée plus loin (§II.II.A.II).

Pour couvrir les différentes fréquences spatiales que peut présenter un symbole, on peut calculer les critères sur différentes échelles de fréquence.

II.A.6.a. Choix du texel

La segmentation nécessite, en général, de définir une taille de texel. Dans notre cas, le texel n'est pas définissable *a priori* car nous ne connaissons pas la fréquence spatiale du symbole dans l'image. Par exemple, si nous prenons comme critère la concentration en contour, et que nous définissons un texel trop petit, l'intérieur d'une cellule d'un symbole Data-Matrix sera considéré comme une zone homogène et aucun symbole sera détecté. Si le texel est trop

grand, tout le symbole y sera contenu avec d'autres éléments de la scène, la concentration en contours ne sera plus celle du symbole seul, et ne répondra pas au critère. Si l'on veut utiliser la segmentation pour notre problème, il sera nécessaire d'avoir des texels dynamiques.

II.A.6.b. Méthode utilisée sur les codes à barres

La lecture globale de code à barres linéaire de Chandler ([CHAN91]) est une méthode à deux étapes. La première étape est une recherche grossière (*coarse*) sur l'image qui détermine des zones susceptibles de contenir un symbole. Cette première recherche se fait sur des groupes de 8×32 pixels avec des orientations diverses. Le critère utilisé est une fonction proche de la moyenne de l'amplitude de la dérivée directionnelle dans une direction donnée. Lorsque le groupe de pixels est sur un symbole de code à barres, ce critère fournit une valeur élevée pour la direction du symbole. Ceci permet de déterminer approximativement à la fois la position du symbole et son orientation.

La seconde étape effectue un échantillonnage précis de la zone supposée contenir un symbole. En mesurant le décalage du signal entre deux lignes consécutives de pixels prises dans le sens du symbole, l'orientation précise des barres est déterminée. Les informations provenant des groupes voisins sont réunies, les largeurs des barres sont calculées en tenant compte de l'orientation, et la valeur du symbole est ainsi déterminée.

Cette méthode a été brevetée en 1991 par Omniplanar, Inc. Elle est bien adaptée à la lecture des codes à barres. Elle pourrait difficilement être adaptée à notre problème. Les codes symboliques à deux dimensions n'ont pas comme les codes à barres de longues lignes de contours dans une direction unique. Le Data-Matrix, tout comme le PDF, a des lignes dans deux directions différentes, et trop courtes pour que leur direction ne soit pas perturbée par la bande passante du système (les composantes de haute fréquence présentes sur les courtes chaînes de contour disparaissent lors de recherches de contours, comme nous le verrons au § II.II.B.II).

II.A.6.c. Recherche de texture en robotique mobile : algorithme de séparation / fusion

La robotique mobile, dans le domaine de la compréhension d'image, propose des algorithmes de segmentation. L'algorithme de séparation / fusion (*split and merge*, présenté dans [ORTE89]*)* en est un. Il suppose que l'on connaît une segmentation approximative d'une image (la segmentation effectuée à l'instant précédent). L'algorithme étudie chaque région et la sépare en plusieurs petites si elle n'est pas homogène selon certains critères, ou bien la fusionne avec ses voisines si elles sont toutes homogènes elles-mêmes et entre elles. L'ordre selon lequel sont faites les fusions et les séparations permet d'avoir plusieurs variantes de l'algorithme.

Le fait de séparer et de regrouper amène à calculer la texture sur des zones de taille différente. L'algorithme permet donc une certaine dynamique en fréquence spatiale. Dans notre cas, nous cherchons une texture dont la fréquence spatiale n'est pas connue à l'avance. Cette dynamique peut donc nous être utile.

II.A.6.d. Méthode de Casals

Les auteurs de [CASA92] proposent une série de critères de texture simples conçus pour des systèmes devant travailler en temps réel. La méthode donne de bons résultats à condition que la fréquence spatiale des textures à identifier soit notablement plus élevée que la fréquence spatiale de la décomposition en texels.

Les paramètres de la texture prennent en compte le nombre de contours, la variance du gradient, la quantité de coins… Un coefficient de normalisation est appliqué pour harmoniser les amplitudes des différents critères (tous n'ont pas la même réactivité). Ces coefficients se déterminent en premier abord en fonction de la nature du critère, puis sont ajustés par l'expérimentation.

Une combinaison linéaire des critères, suivie d'un seuillage, permet d'obtenir une grande souplesse dans la pondération des critères et d'identifier les zones de l'image où se trouvent les textures recherchées.

Les texels sont de taille 8×8 px. Dans toutes les formules qui suivent, la proximité s'entend au sens du voisinage à 8 pixels ; t_X est la valeur du critère et K_X le coefficient de normalisation.

Les ensembles X_n et $\overline{\mathrm{X}}_n$ représentent les pixels de chaînes de contours de longueur respectivement inférieure ou égale et supérieure à n ; les ensembles E_n et $\overline{\mathrm{E}}_n$ représentent les pixels des extrémités des mêmes chaînes ; les ensembles Λ_n et $\overline{\Lambda}_n$ représentent les pixels de ligne de contours de longueur inférieure ou égale à n.

Pour la traduction des termes américains originaux, nous reprenons celle proposée par [LASS96].

- Netteté (*blurriness*)

$\tau_B = K_B \dfrac{(A-D)+(C-B)}{1+\mathrm{card}\left(\overline{\mathrm{X}}_0\right)}$ où A, B, C et D sont les moyennes des luminances dans les quatre sous-régions du texel.

- Granularité (*granularity*)

$\tau_G = K_G \mathrm{card}\left(\overline{\mathrm{X}}_2\right)$.

C'est la quantité de petites chaînes de contours. Par exemple, l'image d'un chandail en laine est fortement granulaire, celle d'un document imprimé l'est peu.

- Discontinuité (*discontinuity*)

$\tau_D = K_D \mathrm{card}\left(\overline{\mathrm{E}}_2\right)$.

C'est le nombre d'extrémités de longues chaînes de contours.

- Escarpement (*abruptness*)

$\tau_A = K_A \alpha$, où α est l'angle de la ligne de contour la plus courte contenue dans le texel.

Ce critère n'est valable que pour une image non bruitée (la plus petite ligne est souvent du bruit) et dont l'orientation a un sens (par rapport à l'orientation des briques du mur).

- Rectitude (*straightness*)

$$\tau_L = K_L \mathrm{card}\left(\overline{\Lambda}_0\right).$$

C'est le nombre de pixels alignés. Ce critère est difficile à mettre en œuvre car il suppose que l'on sache si, dans le voisinage limité que constitue un texel, les pixels sont alignés. En fonction de la taille du texel, ceci n'est pas toujours possible.

- Courbure (*curviness*)

$$\tau_C = K_C\left(\mathrm{card}\left(\overline{\mathrm{X}}_3\right) - \mathrm{card}\left(\overline{\Lambda}_0\right)\right).$$

C'est le nombre de pixels de chaînes de contours courbes.

II.B. Détection de contours

La recherche de contour est une étape très utilisée, voire systématique, dans les systèmes de vision et de compréhension d'image. Dans notre cas, une méthode qui semble directe pour détecter un symbole sur l'image consiste à détecter certains de ses éléments caractéristiques ; parmi ceux-ci, on trouve les contours, mais aussi leurs dérivés : chaînes et lignes de contours, coins de l'image…

Ces derniers termes englobent des définitions qui peuvent varier d'un auteur à l'autre, et nous avons adopté celles qui nous semblent les plus simples ; elles seront en usage tout au long de ce document.

Nous convenons pour la suite des définitions suivantes :

- *Contour* (*edge*)

Zone de l'image qui présente une discontinuité importante en luminance ; l'amplitude de cette discontinuité est l'amplitude du contour. Un contour sera en général matérialisé sur un ensemble de pixels.

- *Chaîne*

Groupe connexe de pixels de contour.

- *Ligne*

Chaîne où les pixels sont alignés.

- *Coin*

Point d'une chaîne où celle-ci change brusquement de direction, ou bien (selon les auteurs et les applications) est coupée par une autre chaîne.

Une chaîne étant un groupe de contours connexe, nous avons de part et d'autre de chaque chaîne deux zones de luminance différente, et une variation d'intensité élevée sur la

chaîne dans le sens transversal. Par exemple, l'image d'un triangle plein contient une chaîne fermée composée de trois lignes, qui se rencontrent en trois coins ; l'image représentant un trait de crayon curviligne large de plusieurs pixels contient, sans compter les extrémités, deux chaînes, une à chaque bord du trait.

II.B.1. Recherche bibliographique

Une volumineuse bibliographie existe sur les recherches de contours ([ROSE74], [DAVI75], [PELI82], [HUER86], [ZHOU89], [FLEC92]), et de nouveaux articles paraissent régulièrement.

Certains auteurs réalisent des détections de lignes et de coins sans utiliser de détection de contours : ceux de [LEES93] et [PEIH94] présentent des détections de coins sur un contour connu par transformation en ondelettes ; ceux de [HIGG94] cherchent les caractéristiques des coins dans une fenêtre flottante ; ceux de [WANG94] cherchent des correspondances de formes à partir de l'image seuillée.

La plupart des méthodes se basent sur la dérivation pour détecter les contours, mais pas toutes. Les auteurs de [NALW86] cherchent à modéliser, par des méthodes de corrélation, les profils de contour par une courbe en tangente hyperbolique.

La quantité de calcul et de mémoire que nécessitent les algorithmes, les résultats sur des images de faible contraste et les déformations sur les coins (qui sont souvent arrondis, décalés ou disloqués) sont les éléments les plus significatifs de ces méthodes en ce qui concerne notre application. Il s'avère cependant qu'il est difficile de juger une méthode sur la base d'une publication, car le succès dépend aussi bien de ses principes que des paramètres choisis pour sa mise en œuvre.

Nous avons relevé quelques exemples d'applications qui montrent comment la détection de contour peut-être utilisée pour la détection de formes particulières, et nous nous en inspirons pour concevoir une méthode similaire pour les symboles Data Matrix.

II.B.1.a. Application à la détection de pistes d'aérodromes

Les auteurs de [BENQ93] présentent une méthode pour détecter les aérodromes dans des images aériennes par la détection de longues lignes droites à partir de lignes issues d'une détection par approximation polygonale des chaînes. Une étude détaillée de la zone autour de ces lignes indique s'il s'agit des pistes d'atterrissage (égalité des luminances de part et d'autre, présence de *taxi-ways*…).

II.B.1.b. Application à l'extraction de caractères

L'extraction de caractères de Lee et Kankanhalli ([LEEK95]) est une méthode à deux étapes. La méthode permet de lire des caractères imprimés dans des images de scène complexe ; elle est appliquée à l'identification de containers par leur numéro dans le port de Singapour.

La première étape est une détection de contours par un changement de résolution dynamique en luminances. Les pixels de contours, situés sur les transitions entre luminances, sont assemblés en groupes en suivant leur connexité. Les groupes de pixels ne répondant pas à certains critères de forme sont rejetés. Les groupes individuels sont sensés représenter des chiffres de taille et d'orientation inconnue. Un réseau neuronal à propagation arrière identifie

chaque chiffre, avec des procédures particulières pour distinguer par exemple le 7 du chiffre 1 ou de la lettre *I* (*i*).

Cette méthode fonctionne correctement et est utilisée dans un site industriel. Elle est elle aussi très adaptée au problème pour lequel elle a été définie. L'algorithme de détection de contour est assez simple, établi sur des bases heuristiques. Par contre, la méthode de regroupement de contour est originale, basée sur les relations des pixels de contour entre eux et sur les caractéristiques intrinsèques de la forme recherchée. L'algorithme est trop complexe pour être résumé en quelques lignes, mais il est sans doute possible de s'en inspirer pour détecter des symboles Data Matrix ou autres. Cependant, le manque de bases théoriques rend difficile l'appréciation *a priori* de son efficacité sur notre problème.

II.B.1.c. Application éventuelle aux symboles Data-Matrix

Pour les symboles Data-Matrix, la recherche des deux lignes orthogonales de même longueur qui bordent le symbole peut amener à la localisation du symbole.

Ceci nécessite une recherche des longues lignes de l'image. Pour chacune de ces lignes (qui sont inspectées par exemple par ordre de taille décroissante), elles doivent vérifier les unes après les autres certaines conditions : il doit y avoir à l'une des extrémités une seconde ligne de taille similaire jointe par son extrémité et relativement perpendiculaire ; ces deux lignes encadrent une zone dominée par des lignes selon une texture particulière (orthogonales, de longueur multiple d'une unité de base) et dont la répartition de la luminance est bimodale, avec des modes équiprobables.

L'interprétation des éléments caractéristiques détectés peut se faire selon des manières moins directes : [MEIS90], [CANN94] font appel à la programmation symbolique (Prolog), basée sur une connaissance de l'image sous forme de primitives de type cercle ou segments, acquises dans une phase préparatoire.

II.B.2. Méthode générale par dérivation

La discontinuité en luminance qui marque la présence d'un contour peut être détectée en effectuant des mesures de la dérivée de l'image de premier ordre (gradient de Sobel, Prewitt…) ou de second ordre (laplacien de Marr-Hildreth…), mais d'autres méthodes sont envisageables (Canny). La détection de contour mène en général à l'établissement d'une carte de contours, voire d'une carte de contours binaires (qui contient comme information l'absence ou la présence de contour, obtenue en général par seuillage de l'intensité de la réponse d'un détecteur de contour).

Certaines méthodes de recherche de contours utilisent des procédés liés indirectement à la dérivation. Les auteurs de [ROSE71] font la différence entre luminances moyennes de part et d'autre du pixel étudié. Les auteurs de [CHEN91] font la différence entre signal et signal lissé tandis que celui de [PARK94] fait un seuillage par la moyenne locale. Ces méthodes permettent un traitement rapide, mais ne s'adaptent pas forcément à d'autres applications particulières.

La plupart des méthodes de détections de contour suivent quatre étapes majeures : lissage, dérivation, seuillage et amincissement.

Le lissage, ou filtrage passe-bas, est nécessaire en premier lieu pour éviter que le détecteur de contour ne détecte des pixels dont le contraste élevé est dû à du bruit. La fréquence

de coupure du lissage est le paramètre qui détermine la capacité à détecter les contours proches les uns des autres et à les localiser.

La dérivation permet de repérer les pixels de fort contraste (de valeur très différente de leurs voisins). Lorsque l'on utilise un masque comme celui de Sobel ou Prewitt, on effectue en même temps le lissage sur une fenêtre plus ou moins grande.

Le seuillage permet d'éliminer les fronts jugés de trop faible amplitude et considérés comme non significatifs. Des techniques évoluées peuvent être utilisées. Le seuillage par hystérésis, par exemple, permet d'éliminer les contours non significatifs en fonction non seulement de leur intensité, mais aussi de leur environnement. Un intervalle d'intensité est choisi comme seuil, les contours dont l'intensité est dans cet intervalle ne seront considérés comme significatifs (et conservés) que s'ils ont comme voisin (le long de leur chaîne) un contour significatif. Les contours dont l'amplitude est deçà de l'intervalle sont éliminés, ceux dont l'amplitude en est au-delà sont conservés. Un tel seuillage amène naturellement des contours minces.

L'amincissement permet de localiser les contours en ne les représentant, le long d'une chaîne, qu'avec un pixel d'épaisseur (même si la pente associée en luminance dans l'image est plus large).

L'amincissement dont nous parlons ici est différent de la squelettisation qui réduit un objet dans toutes ces directions. L'amincissement d'une chaîne ne la réduit que dans la direction de sa largeur, sans affecter sa longueur.

L'amincissement peut utiliser l'intensité de la dérivée, la forme du signal au voisinage du contour, la direction du gradient. Lorsque l'on dérive par l'opérateur laplacien, on cherche ensuite les passages à zéro et on obtient des chaînes minces (ces opérateurs sont détaillés plus loin, § II.II.B). Une technique d'amincissement classique est l'élimination dite des contours non maximaux dans la direction du gradient. Pour chaque pixel contour, celui-ci est gardé si dans la direction du gradient, ses voisins ont une intensité de gradient inférieure, c'est-à-dire si le pixel considéré est un maximum local dans la direction du gradient.

II.B.3. Dérivation par gradient

Plusieurs méthodes basées sur le gradient existent . Les auteurs de [TANL93] font une recherche hiérarchique (à plusieurs échelles) par un gradient simple ; ceux de [PARK95] déterminent la variance locale de l'image pour faire un masque adaptatif. Les auteurs de [WHIT83] implémentent un masque réduit de gradient qui diminue la fréquence de coupure sans augmenter les temps de calcul (avec un risque d'augmentation de sensibilité au bruit) ; ceux de [NELS94] utilisent les masques directionnels de Kirsh (puis une squelettisation par l'amplitude du contour).

Mais la façon la plus directe d'obtenir une carte de contours est d'appliquer à l'image un masque de convolution qui réalise une opération de dérivation [WANG94].

Il est recommandé d'utiliser un masque symétrique afin d'avoir dans la mesure du possible des propriétés isotropes. Le choix de la taille du masque (2×2, 1×3, 3×3…) influe sur la fréquence de coupure du détecteur.

Nous présentons ci-dessous une série de masques classiques et les équations de dérivations associées COCQ95. Dans ces matrices, l'élément souligné est l'élément associé au pixel en cours lors de la convolution. On note g l'image à analyser et $\hat{\nabla} g$ l'estimation de son gra-

dient. g est défini sur une grille de points repérée par les indices (i,j). ∇g_x, ∇g_y représentent les dérivées directionnelles selon les directions indiquées, $\tilde{\nabla} g_x$, $\tilde{\nabla} g_y$ leurs estimations.

- Masques linéaires 1×2 et 2×1 : $\begin{pmatrix} \underline{-1} & 1 \end{pmatrix}$ et $\begin{pmatrix} \underline{-1} \\ 1 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i+1,j) - g(i,j) \text{ et } \tilde{\nabla} g_y = g(i,j+1) - g(i,j) \quad \text{(II-1)}$$

- Roberts 2×2 : $\begin{pmatrix} \underline{1} & 0 \\ 0 & -1 \end{pmatrix}$ et $\begin{pmatrix} \underline{0} & -1 \\ 1 & 0 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i,j) - g(i+1,j+1) \text{ et } \tilde{\nabla} g_y = g(i+1,j) - g(i,j+1) \quad \text{(II-2)}$$

- Masques linéaires 3×3 : $\begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$ et $\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i+1,j) - g(i-1,j) \text{ et } \tilde{\nabla} g_y = g(i,j+1) - g(i,j-1) \quad \text{(II-3)}$$

- Roberts 3×3 : $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$ et $\begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i-1,j-1) - g(i+1,j+1) \text{ et } \tilde{\nabla} g_y = g(i+1,j-1) - g(i-1,j+1) \quad \text{(II-4)}$$

- Sobel 3×3 : $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ et $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i-1,j-1) + 2g(i-1,j) + g(i-1,j+1) - g(i+1,j-1) - g2(i+1,j) - g(i+1,j+1) \quad \text{(II-5)}$$

$$\tilde{\nabla} g_y = g(i-1,j-1) + 2g(i,j-1) + g(i+1,j-1) - g(i-1,j+1) - 2g(i,j+1) - g(i+1,j+1) \quad \text{(II-6)}$$

- Prewitt 3×3 : $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$ et $\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$

$$\tilde{\nabla} g_x = g(i-1,j-1) + g(i-1,j) + g(i-1,j+1) - g(i+1,j-1) - g(i+1,j) - g(i+1,j+1) \quad \text{(II-7)}$$

$$\tilde{\nabla} g_y = g(i-1,j-1) + g(i,j-1) + g(i+1,j-1) - g(i-1,j+1) - g(i,j+1) - g(i+1,j+1) \quad \text{(II-8)}$$

À partir des composantes du gradient, on peut connaître l'intensité et la direction du gradient.

- Norme 1 : $\tilde{\nabla} g = \left|\tilde{\nabla} g_x\right| + \left|\tilde{\nabla} g_y\right|$ (II-9)

- Norme 2 : $\vec{\nabla} g = \sqrt{\vec{\nabla} g_x{}^2 + \vec{\nabla} g_y{}^2}$ (II-10)
- Norme ∞ : $\vec{\nabla} g = \max\left(\left|\vec{\nabla} g_x\right|, \left|\vec{\nabla} g_y\right|\right)$ (II-11)
- Direction : $\arg\left(\vec{\nabla} g\right) = \arctan_{xy}\left(\vec{\nabla} g_x, \vec{\nabla} g_y\right)$ (II-12)

II.B.3.a. Cartes de contours

Nous allons comparer ces divers masques. Les figures II-1 et II-2 représentent des symboles Data-Matrix, pour lesquels la fréquence spatiale est respectivement faible (0,13 px^{-1}) et forte (0,49 px^{-1}). Nous avons calculé le gradient de l'image II-1 selon Sobel et Prewitt (figures II-3 et II-4), et des images II-1 et II-2 selon Roberts (figures II-5 et II-6).

FIG. II-1 - *image de Notnumb15*

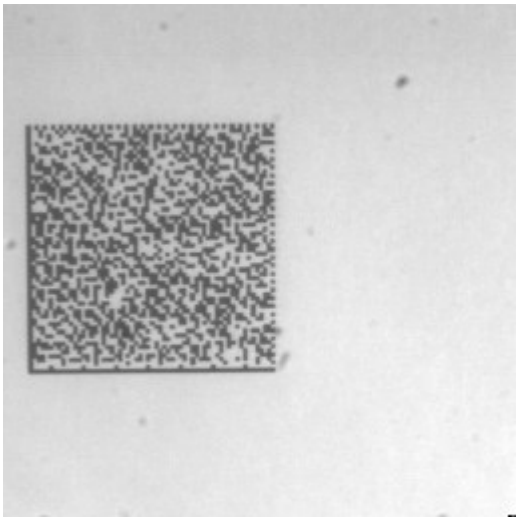


FIG. II-2 - *image de Clarm61*

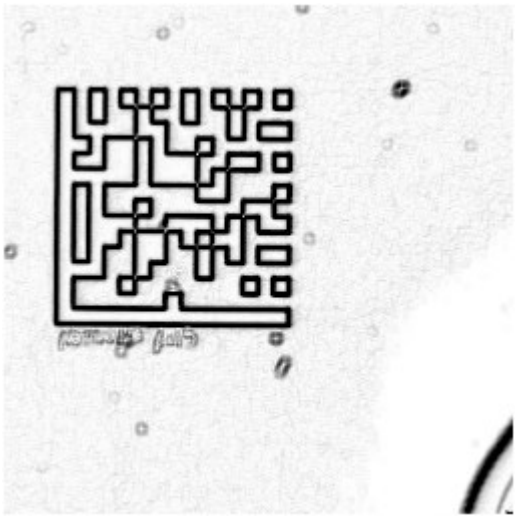


FIG. II-3 - *contours par Sobel de Notnumb15*

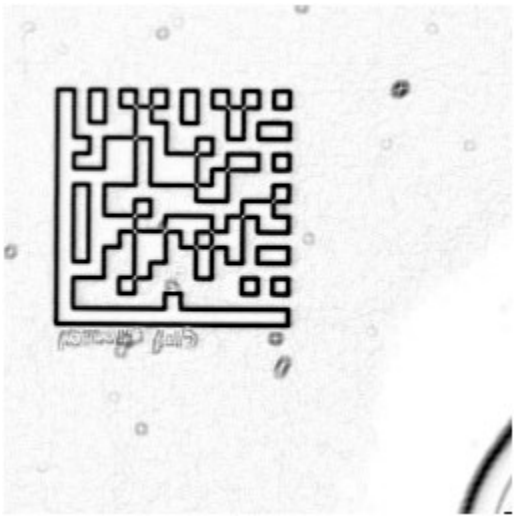


FIG. II-4 - *contours par Prewitt de Notnumb15*

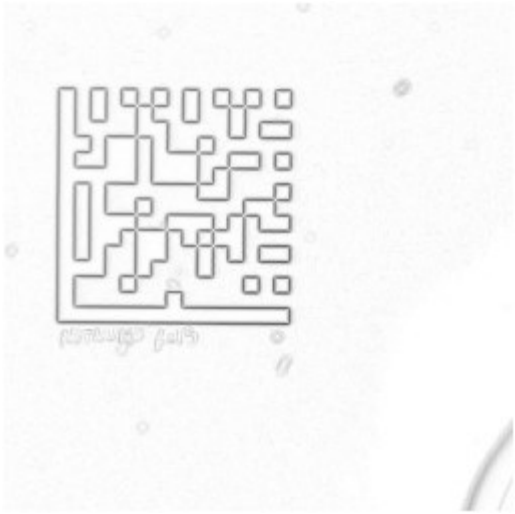


FIG. II-5 - *contours par Roberts 2×2 de Notnumb15*

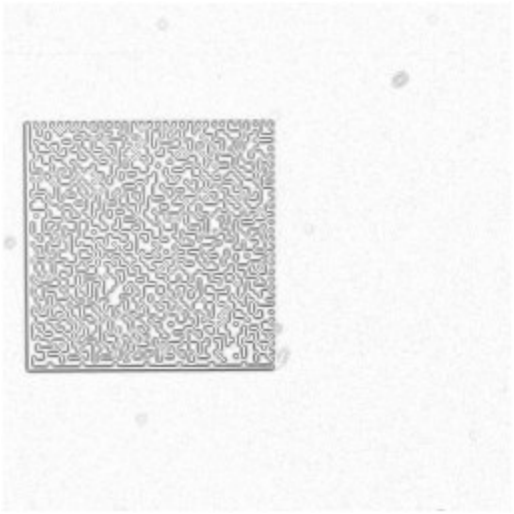


FIG. II-6 - *contours par Roberts 2×2 de Clarm61*

Les masques de Sobel et Prewitt (figures II-3 et II-4) conduisent à des contours épais et manquent de précision. Les contours délivrés par le masque de Roberts sont plus fins. En haute fréquence, le masque de Roberts (figure II-6) arrive difficilement à décrire le contour du symbole au niveau de la zone d'horloge qui semble être uniformément grise. A l'intérieur du symbole, les bords des cellules ne sont pas lisibles ; on a du mal à les reconstituer à l'intérieur des contours.

Pour obtenir une carte de contours binaires, les résultats des détecteurs de contours de Roberts sont seuillés (figures II-7 et II-8).

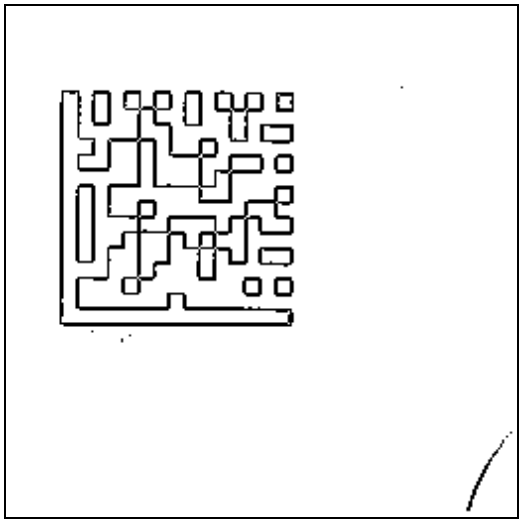


FIG. II-7 - *contours binaires par Roberts 2×2 de Notnumb15*

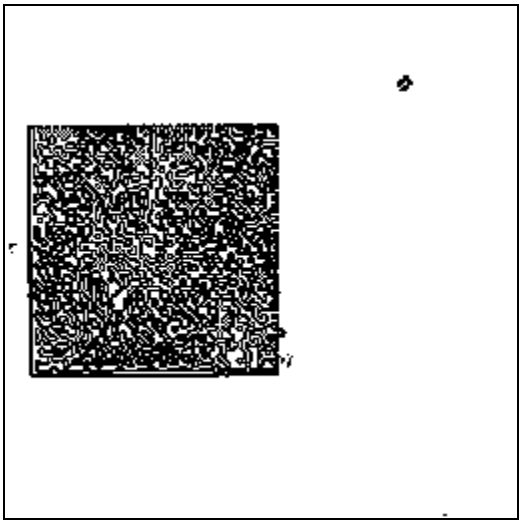


FIG. II-8 - *contours binaires par Roberts 2×2 de Clarm61*

Comme nous le voyons sur les images précédentes, les contours de l'image en basse fréquence ont été identifiés sans ambiguïté, tandis que les contours de l'image en haute fréquence sont mêlés les uns aux autres et la zone d'horloge du symbole en est indiscernable. Le seuil choisi n'est peut-être pas optimal, mais aucun seuil n'aurait pu rendre nettement les contours de cette zone à partir de la carte de la figure II-6.

II.B.4. Autres méthodes de dérivation

Plusieurs méthodes font appel à la dérivation seconde voire troisième. Ces méthodes utilisent des dérivations partielles directionnelles ou des masques à symétrie centrale. En fonction des champs d'application et des critères d'efficacité de chacun, telle ou telle méthode sera la mieux adaptée à tel ou tel problème.

La méthode de Canny ([CANN86]) se base sur le calcul d'un filtre de forme optimale calculé numériquement selon certains critères. Plusieurs auteurs reprennent cette méthode en précisant et étendant les résultats théoriques ([DERI87], [BOUR93]).

Haralick propose une méthode basée sur le passage à zéro de la dérivée seconde directionnelle appliquée à une approximation polygonale des chaînes de l'image ([HARA84], [TORR86]) ; les auteurs de [ZHOU89] calculent cette dérivée sur l'image elle-même (suivi par un chaînage vectoriel). Marr et Hildreth calculent le laplacien sur l'image filtrée par un filtre gaussien (approximation du filtre optimal de Canny, [CHEN87], [SOTA89], [DEFE91], [BASU94], [BERZ84], [ZIOU93]).

Ces méthodes ont le défaut de nécessiter une grande quantité de calculs. Le choix chez chaque auteur entre mode de calcul directionnel ou non directionnel n'est pas toujours clairement justifié. De plus, les techniques de chaînage qui en découlent n'utilisent pas la direction du gradient (celle-ci n'est pas calculée). En effet, le gradient d'une image porte des informations aussi bien sur l'amplitude des contours que sur leur direction (respectivement amplitude et direction du gradient). Le gradient peut servir à la fois pour la recherche de contour et pour la vectorisation ; il est donc préférable de choisir des méthodes basées sur le gradient (tant qu'elles sont efficaces) pour réduire la quantité de calculs.

II.B.5. Méthodes par Laplacien

Le laplacien est la somme des dérivées secondes partielles dans les directions du repère ; c'est un scalaire et il correspond à la dérivée seconde de l'image. Entre deux zones de luminances différentes, les points de la transition où la discontinuité est la plus forte présentent un extremum du gradient (point d'inflexion), et donc un changement de signe de la dérivée seconde dans cette direction (c'est à dire du laplacien). Les contours sont donc localisés sur des passages à zéro du laplacien.

La méthode consiste à calculer le laplacien de l'image ; les passages à zéro du laplacien seront considérés comme des contours. En fait, tous les passages à zéro du laplacien ne sont pas nécessairement des contours, comme nous allons le voir par la suite.

II.B.5.a. Calcul du laplacien

On trouve dans la littérature plusieurs opérateurs relativement simples de calcul du laplacien. Nous en présentons quelques-uns ci-dessous. On note g l'image à analyser et $\vec{\nabla}^2 g$ l'estimation de son laplacien. g est défini sur une grille de points repérée par les indices (i,j). $\partial^2 g_{\partial x^2}$, $\partial^2 g_{\partial y^2}$, $\partial^2 g_{\partial x \partial y}$ et $\partial^2 g_{\partial y \partial x}$ représentent les dérivées directionnelles secondes selon les directions indiquées.

Les dérivées directionnelles premières sont notées ∇g_x et ∇g_y dans l'ensemble de ce document. Par souci de clarté, nous utilisons ici les notations $\partial g_{\partial x}$ et $\partial g_{\partial y}$, similaires à celles de la dérivée seconde.

- Opérateur non directionnel (noté $\partial^2 / \partial n^2$) de [TORR86].

Cet opérateur n'est pas basé sur les dérivées directionnelles selon x et y, mais correspond à la dérivation seconde dans la direction du gradient (notée $\vec{h}$). Il n'est pas applicable lorsque la dérivée première est localement nulle, car le gradient n'y est pas défini. On donne alors la valeur zéro. En dehors de ce cas, cet opérateur se décompose de la façon suivante :

$$\nabla^2 g = \frac{\partial^2 g}{\partial n^2} = \frac{\partial g_{\partial x}{}^2 \partial^2 g_{\partial x \partial x} + 2\partial g_{\partial x}\partial g_{\partial y}\partial^2 g_{\partial x \partial y} + \partial^2 g_{\partial x \partial y}{}^2 \partial^2 g_{\partial y \partial y}}{\partial g_{\partial x}{}^2 + \partial g_{\partial y}{}^2} \quad \text{(II-13)}$$

- Masque 3×3 à 4 voisins : $\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$

$$\frac{\partial g^2(i,j)}{\partial x \partial y} = g(i+1,j) + g(i-1,j) + g(i,j+1) + g(i,j-1) - 4g(i,j) \quad \text{(II-14)}$$

- White-Rohrer à 4 voisins ([WHIT83]) : $M_{WR} = \begin{pmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{pmatrix}$

$$\frac{\partial g^2(i,j)}{\partial x \partial y} = g(i+2,j) + g(i-2,j) + g(i,j+2) + g(i,j-2) - 4g(i,j) \quad \text{(II-15)}$$

- Marr-Hildreth à 8 voisins : $M_{MH} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & -4-2\sqrt{2} & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix}$

$$\begin{aligned} \frac{\partial g^2(x,y)}{\partial x \partial y} &= g(i+1,j) + g(i-1,j) + g(i,j+1) + g(i,j-1) - 4g(i,j) \\ &+ \tfrac{1}{\sqrt{2}}\big(g(i+1,j+1) + g(i-1,j+1) + g(i+1,j-1) + g(i-1,j-1) - 4g(i,j)\big) \end{aligned} \quad \text{(II-16)}$$

II.B.5.b. Cartes et propriétés du laplacien

Les cartes II-9 et II-10 représentent le laplacien des images II-1 et II-2.

Une carte du laplacien indique des valeurs relatives positives ou négatives. Nous avons opté pour une représentation dans laquelle la luminance médiane (gris) représente la valeur zéro, et les pixels plus clairs ou plus sombres représentent respectivement les valeurs négatives et positives. Les contours sont situés sur le passage à zéro du laplacien (lorsque nous avons une transition clair ↔ sombre).

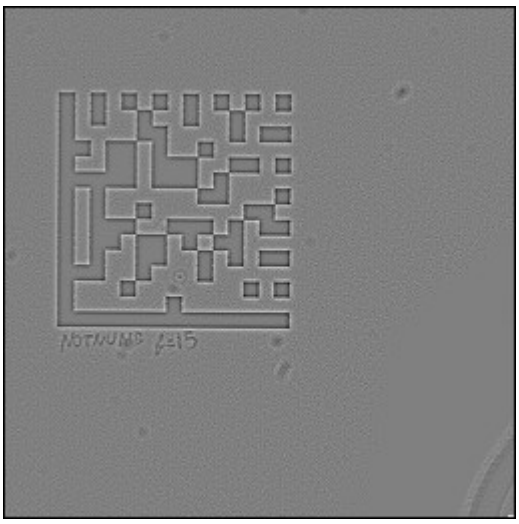


FIG. II-9 - *laplacien de Notnub5*

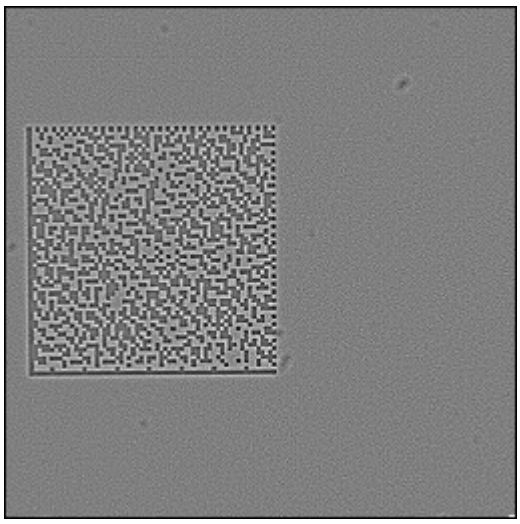


FIG. II-10 - *laplacien de Clarm61*

Contrairement à la carte des contours de Roberts (figure II-5 et II-6), les contours restent bien délimités et bien lisibles en haute fréquence, car le laplacien introduit un filtrage passe-haut. Si on seuille ces deux cartes au niveau zéro, nous obtenons une carte binaire où les transitions noir ↔□blanc représentent les passages à zéro du laplacien et donc les contours détectés. Les cartes obtenues se trouvent figures II-11 et II-12.

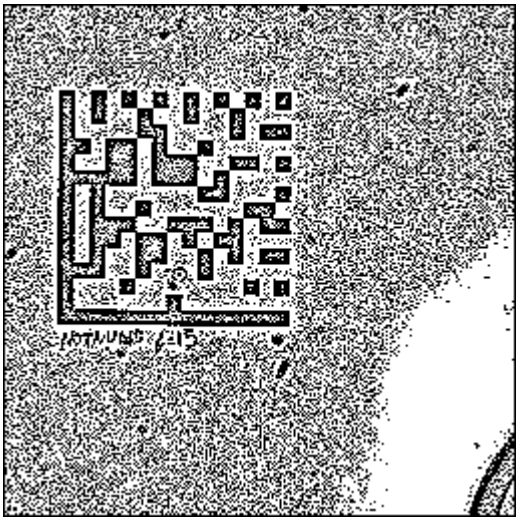


FIG. II-11 - *contours binaires par laplacien de Clarm61*

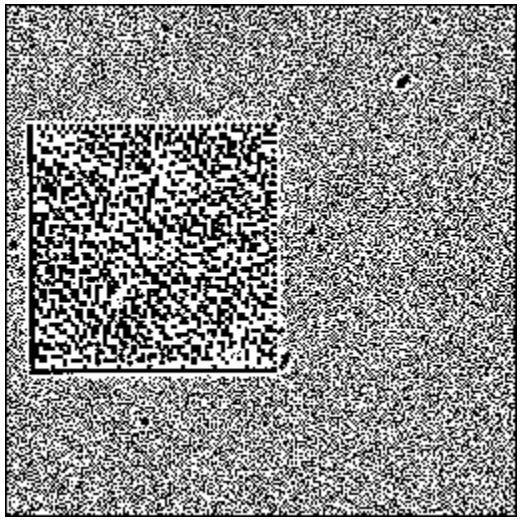


FIG. II-12 - *contours binaires par laplacien de Clarm61*

En étudiant les cartes de contour binaire, on se rend bien compte que le seuillage a séparé les pixels de l'image en deux classes. Si l'on observe les contours des groupes connexes de pixels d'une classe donnée, on voit que ceux-ci sont minces (pas plus d'un pixel de large) et fermés.

Ceci est une propriété importante des contours obtenus par le passage à zéro du laplacien, qui vient du fait que pour un signal continu (non échantillonné), les passages à zéro de la dérivée seconde constituent des courbes de niveau d'une fonction du plan. Cette propriété est intéressante si nous utilisons la détection de contours comme une étape préalable à des procédures de compréhension d'image basées sur les chaînes ou les lignes de l'image. Cependant, cette propriété peut disparaître lorsque l'on calcule numériquement les contours (on quitte le

domaine théorique). Ceci arrive notamment si l'on seuille les contours sur leur intensité, ou si l'on calcule la carte de contours sans tenir compte des contours entre-pixels.

L'utilisation d'un filtrage préalable par un filtre gaussien réduit la bande passante du signal-image, élimine les points d'inflexion de faible intensité et conserve en pratique les propriétés de fermeture et de minceur. Un tel filtre permet en outre d'arrondir les coins, ce qui réduit les erreurs d'estimation du laplacien (un coin étant une discontinuité du contour, le gradient y est indéterminé, et le calcul du laplacien entaché d'incertitudes).

Si nous désirons avoir une carte des contours, il suffit de marquer chaque transition de la carte binaire donnée par le laplacien. Nous obtenons les cartes II-13 et II-14.

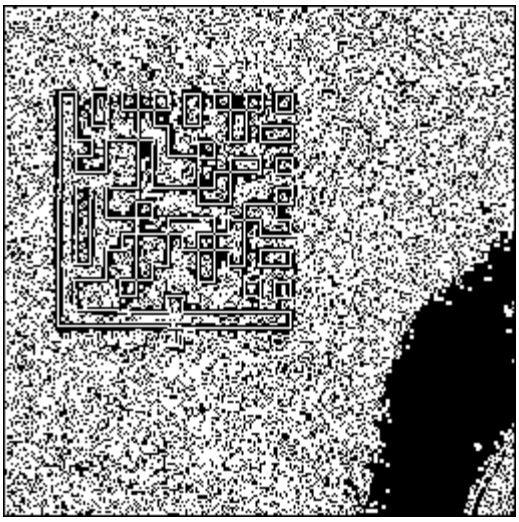


FIG. II-13 - *contours binaires d'après laplacien de Notnumb15*

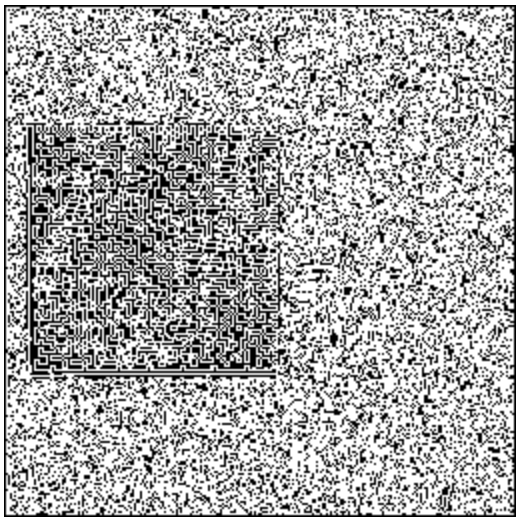


FIG. II-14 - *contours binaires filaires d'après laplacien de Clarm61*

Contrairement à ce que nous avions figure II-12, la carte de contour du symbole Clarm61 n'est pas nette à l'intérieur du symbole. Nous n'avons plus de contours minces ni fermés, mais des contours amalgamés qui rendent mal compte des cellules du symbole. Ceci est dû au fait que les contours que l'on devinait sur la carte II-12 étaient des contours inter-pixel, alors que ceux représentés figure II-14 sont des contours sur-pixel. Nous reviendrons sur ce point plus loin (§ II.II.B.II).

II.B.5.c. Élimination du bruit (filtrage / seuillage)

On peut voir sur les figures II-11 à II-14 que les cartes de détection comportent en certains endroits des pixels qui ne correspondent pas à des contours significatifs des images. Ces pixels correspondent à la détection des irrégularités du niveau uniforme, mais bruité, de luminance des images II-1 et II-2. Ces réponses de la carte de contours sont appelées *fausses alarmes*.

L'élimination de ces fausses alarmes, ainsi que des contours non désirés de faible intensité peut se faire par seuillage de la carte selon l'amplitude de la transition. Un tel seuillage a été appliqué sur les cartes de contour binaire du gradient (figures II-7 et II-8).

On peut aussi éliminer ce bruit par un filtrage préalable, effectué sur l'image avant le calcul du gradient ou laplacien.

Si l'on décide de s'abstenir d'utiliser un seuil (II-12 à II-13), on voit que les fausses alarmes ont une fréquence spatiale stationnaire (indépendante de celle du symbole) de l'ordre

de 0,3 px^{-1} à 1 px^{-1}. Un filtrage préalable avec une fréquence de coupure de 0,3 px^{-1} en éliminerait une grande partie, mais limiterait aussi la résolution du détecteur. De nombreux auteurs (dont ceux de [TORR86]) recommandent un filtrage préalable à toute recherche de contours par convolution, en général par un filtre gaussien.

Si l'on décide d'utiliser un seuil pour éliminer le bruit (figures II-7 et II-8), la plupart des fausses alarmes disparaissent, (car elles sont pour la plupart dues à des faibles changements de luminance dans l'image), mais ce seuillage risque d'entraîner une disparition des contours de faible intensité. Nous pouvons aussi envisager d'éliminer le bruit par l'utilisation d'un filtre médian.

Nous avons là un dilemme contraste / précision. Si on veut une forte sensibilité au contraste, on réduit les valeurs des seuils, ce qui augmente le bruit. On peut alors l'éliminer par un filtrage passe-bas ce qui diminue la résolution du détecteur de contours. Si on veut une bonne résolution, on augmente la fréquence de coupure du détecteur, et on doit donc éliminer le bruit en mettant des seuils sur l'amplitude des contours, ce qui diminue la sensibilité aux faibles contours.

II.B.5.d. Contours sur pixel / contours entre pixels.

Un détecteur qui cherche le maximum du gradient identifie un certain nombre de pixels comme pixels de contour. Dans cette approche, la carte de localisation présente des contours localisés sur des pixels.

En revanche, un détecteur qui mesure le passage à zéro du laplacien identifie comme contour des passages à zéro de la dérivée seconde. Lorsque deux pixels voisins ont des dérivées secondes de signe différent, cela veut dire qu'il y a entre eux un contour. Dans cette approche, la carte de localisation présente (ou devrait présenter) des contours localisés entre des pixels ([FLEC92]).

Une carte de contours sur pixels permet de déterminer la position avec une précision d'un pixel.

On peut évaluer la précision d'une carte de contours entre pixels en comptant le nombre de positions possible d'un contour dans cette image. En omettant les effets de bord, on trouve avec une topologie à 4 voisins (on parlera de contours entre 2 pixels) qu'une carte de contours entre pixels peut présenter deux fois plus de contours que de pixels (en prenant pour chaque pixel par exemple le bord de droite et le bord du haut). La précision de la localisation est donc d'½ pixel. En prenant une topologie à 8 voisins (on parle de contours entre 4 pixels) au lieu de 4, on trouve des contours sur les nœuds de l'échantillonnage et une précision d'⅓ de pixel.

On peut donc voir que par nature, les détecteurs liés au passage à zéro du laplacien ont une résolution supérieure aux détecteurs sur-pixel. Par contre, la capacité de stockage de la carte de détection doit être plus grande, puisqu'il faut mémoriser jusqu'à 3 contours par pixels.

Il ne faut pas en conclure que les détections de contour par laplacien soient plus précises que celles par gradient. En raison de ses propriétés, le laplacien a une fréquence de coupure à 1 px^{-1} alors que le Roberts l'avait à 0,5 px^{-1}. On peut construire un masque de gradient générant des contours entre les pixels et atteindre la précision du laplacien ; et réciproquement, réduire la fréquence de coupure du laplacien ramène sa performance au niveau de celle du Roberts (figure II-14).

II.B.5.e. Inflexion et contour

D'une manière générale, un passage à zéro de la dérivée seconde d'une fonction correspond à un point d'inflexion de la fonction. Cependant, tous les points d'inflexion d'une fonction ne correspondent pas nécessairement à un contour ([FLEC92]). Nous allons le montrer par des exemples sur des fonctions de $\mathcal{C}^3$ continûment dérivables au voisinage du point considéré.

Un point d'inflexion d'une fonction $f(x)$ est déclaré contour dans les conditions suivantes.

1) Pour f croissant, le point d'inflexion est un maximum local de f', c'est à dire f' positif, f'' nul décroissant et f''' négatif (voir figure II-15A).

2) Pour f décroissant, le point d'inflexion est un minimum local de f', c'est à dire f' négatif, f'' nul croissant et f''' positif (voir figure II-15C).

Dans ces deux cas :

$$f'(x) \neq 0, \quad f'''(x) \neq 0 \ ; \text{(II-17)}$$

$$\operatorname{sgn}(f'(x)) = -\operatorname{sgn}(f'''(x)) \, . \text{(II-18)}$$

Dans les deux autres cas (figures II-15B et II-15D), le point d'inflexion ne présente pas de contour.

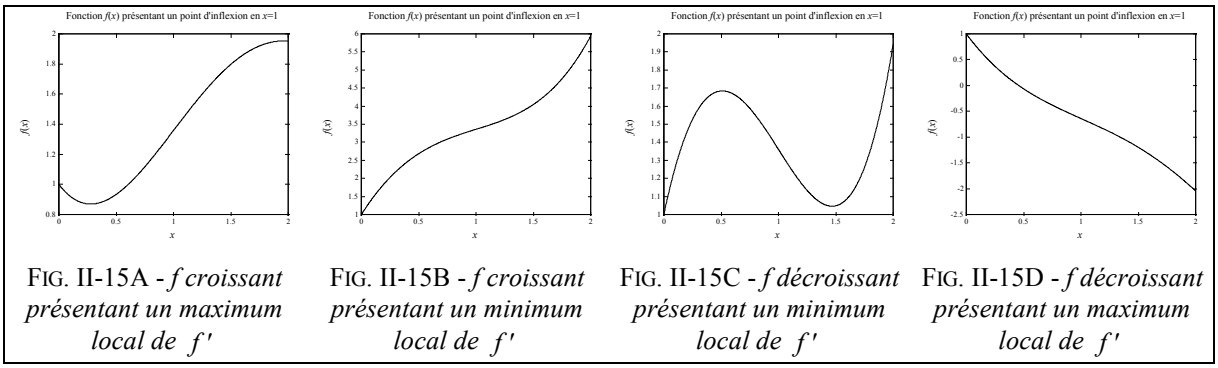


FIG. II-15A - *f croissant présentant un maximum local de f'*

FIG. II-15B - *f croissant présentant un minimum local de f'*

FIG. II-15C - *f décroissant présentant un minimum local de f'*

FIG. II-15D - *f décroissant présentant un maximum local de f'*

FIG. II-15. *Les quatre fonctions ci-dessus présentent un point d'inflexion en x=1. Seules les fonctions des figures II-15A et II-15C présentent un point de contour en ce point*

Puisque le laplacien est la dérivée directionnelle de l'image dans la direction du gradient, on peut appliquer cette propriété. Un passage à zéro du laplacien sera un contour si le gradient et la dérivée du laplacien sont non nuls (éventuellement de valeur absolue supérieure à un certain seuil) et de signe différent. Pour un contour sur pixel, cette vérification est faite pour le pixel en question ; pour un contour entre pixels, la validation est faite sur les deux pixels voisins concernés. Ces propriétés du voisinage d'un contour sont appelées les *propriétés gages de contour*.

II.B.5.f. Méthode de Marr-Hildreth

La méthode de Marr-Hildreth [MARR80] est une méthode basée sur le passage à zéro du laplacien. Les passages à zéro du laplacien sont considérés comme des contours si le gradient et la dérivée troisième possèdent en ces points les propriétés gages de contour.

La méthode nécessite l'utilisation de deux cartes qui sont une *carte de localisation* (qui contient la réponse de l'opérateur de recherche de contour) et une *carte de détection* (contenant les propriétés gages de contour). D'une manière générale, on a un contour en un point si les deux cartes indiquent en ce point et dans son voisinage les propriétés caractéristiques d'un contour.

La localisation consiste à chercher les passages à zéro significatifs du laplacien (c'est à dire d'amplitude supérieure à un seuil T_2). Pour les contours entre pixels, il faut que $\nabla^2 g$ soit élevé en valeur absolue et de signe différent de part et d'autre. Pour les contours sur pixel, il faut que $\nabla^2 g$ soit nul sur le pixel en question (ce qui correspond à un passage à zéro, dans la mesure où la carte de détection a aussi une réponse sur ce pixel).

La détection consiste à vérifier que pour chaque pixel susceptible d'être un contour, les dérivées première et seconde ont des valeurs absolues élevées ($\left|\vec{\nabla}^3 g\right|$ et $\left|\vec{\nabla} g\right|$ supérieures à des seuils respectivement T_3 et T_1). Ces propriétés sont proches des propriétés gages de contour vues au § II.II.B.II.

II.B.5.g. Détecteur de contours de Fleck

L'auteur de [FLEC92] présente des méthodes basées sur le passage à zéro du laplacien. La base théorique en est établie avec précision, et l'ensemble ne nécessite pas une grande quantité de calculs. L'auteur présente les deux façons de déterminer l'emplacement d'un contour : contours sur pixel ou contours entre pixels (*on-cell edges* et *inter-cell edges*), les propriétés gages de contour liées aux valeurs des dérivées seconde et troisième du signal, puis utilise ces concepts pour analyser quelques algorithmes classiques de détection de contour (algorithmes de Canny, Boie-Cox, Marr-Hildreth, Marr-Hildreth directionnel et *Max-Second*). Les défauts notoires de ces algorithmes sont présentés, ainsi que des remèdes pour s'en affranchir.

II.B.5.g.i) Algorithme de Fleck

L'image à analyser est notée g. p représente un pixel. Les deux cartes sont *Détection*(p) et *Localisation*(p).

Algorithme :

Pour chaque pixel p en (i,j) :
  Calcul de $\vec{\nabla}^2 g(p)$
  Calcul de $\vec{\nabla}^3 g(p)$ et $\vec{\nabla} g(p)$
  Si $\left|\vec{\nabla} g(p)\right| > T_1$ et $\left|\vec{\nabla}^3 g(p)\right| > T_3$ et si $\mathrm{sgn}(\vec{\nabla} g(p)) = -\mathrm{sgn}(\vec{\nabla}^3 g(p))$ alors
    Détection(p) ← Vrai
  sinon
    Détection(p) ← Faux
  Si *Détection*(p) et $\left|\vec{\nabla}^2 g(p)\right| < T_2$ alors
    Localisation(p) ← Vrai
    Il y a un contour sur le pixel p.
  Pour les voisins p_k de p pour lesquels le détecteur est calculé :

si *Détection*(p) et *Détection*(p_k) et $\left|\vec{\nabla}^2 g(p)\right| > T_2$ et $\left|\vec{\nabla}^2 g(p_k)\right| > T_2$ et

$\mathrm{sgn}(\vec{\nabla}^2 g(p)) \neq \mathrm{sgn}(\vec{\nabla}^2 g(p_k))$ alors

Il y a un contour entre les pixels p et p_k.

Fin de boucle *pour*.

Fin de boucle *pour*.

On remarque que les cartes de détection et de localisation de l'algorithme n'ont besoin d'être connues qu'au voisinage du pixel considéré (ce qui peut permettre une économie en volume de mémoire nécessaire). On remarque aussi que le calcul des dérivées première, seconde et troisième ne peut pas se faire en parallèle. En effet, on calcule d'abord le laplacien $\vec{\nabla}^2 g(p)$, puis son gradient $\vec{\nabla}^3 g(p)$, puis la dérivée première de l'image dans la direction de $\vec{\nabla}^3 g(p)$: $\vec{\nabla} g(p)$.

Cet algorithme délivre à la fois des contours sur pixels et des contours entre pixels, ce qui peut poser des problèmes de continuité entre les différents types de contour. Une étape postérieure consiste à unifier les deux cartes de contour en spécifiant que les bords de chaque pixel contour sont aussi des contours (entre pixels), et que les deux nœuds de chaque contour entre 2 pixels sont aussi des contours entre 4 pixels.

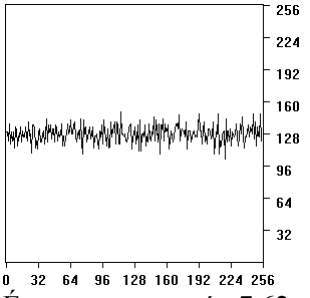
II.B.5.h. Comparaison des masques de laplacien

Parmi les quatre méthodes de calcul présentées au § II.II.B.II, deux d'entre elles présentent les meilleurs résultats pour le type d'image utilisée : le masque de White-Rohrer (M_{WR}) et le masque de Marr-Hildreth (M_{MH}). Nous allons les comparer :

$$M_{WR} = \begin{pmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{pmatrix}, \; M_{MH} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & -4-2\sqrt{2} & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

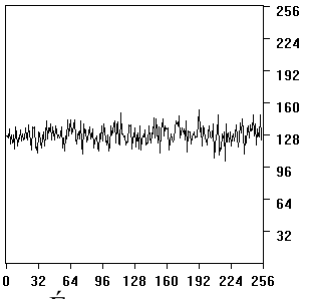
Les figures II-16 et II-17 sont des profils extraits d'une carte de contours dans une zone sans contour. La réponse au masque de White-Rohrer a été corrigée par un facteur multiplicatif afin d'égaliser les niveaux de bruit sur les deux échantillons (mesurés par l'écart type du signal).

Les figures II-18 et II-19 sont des profils de l'image extraits d'une zone où se trouvaient des contours. Avec la même correction, et donc des niveaux de bruit égaux, on voit que l'intensité des pics correspondant aux contours est plus élevée. Le masque de Marr-Hildreth délivre des résultats ayant un rapport signal-sur-bruit plus élevé.



Écart type mesuré : 7,62

FIG. II-16 - *échantillon de bruit (White-Rohrer corrigé)*



Écart type mesuré : 7,59

FIG. II-17 - *échantillon de bruit (Marr-Hildreth)*

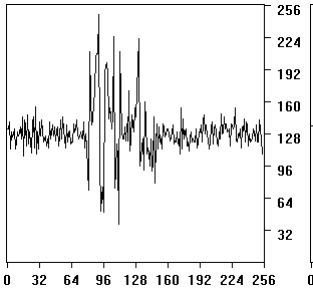


FIG. II-18 - *laplacien par masque de White-Rohrer corrigé*

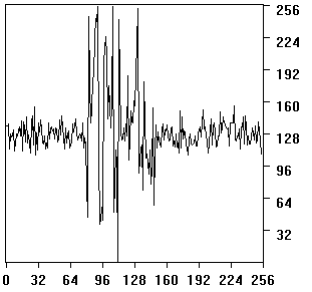


FIG. II-19 - *laplacien par masque de Marr-Hildreth*

Cependant, d'un point de vue numérique, le masque de White-Rohrer nécessite une convolution avec une matrice d'entiers, tandis que le masque de Marr-Hildreth nécessite une convolutions avec une matrice de flottants, ce qui est plus couteux en temps de calcul. Mais si l'on compare les équations II-15 et II-16 (page II-37), on voit que la convolution selon Marr-Hildreth peut être ramenée à environ le double de calculs de la convolution selon White-Rohrer.

En fonction de la capacité de détection souhaitée et de la quantité de calculs qu'il en résulte, nous choisirons l'un ou l'autre détecteur. Il s'est avéré dans notre cas que le laplacien à 4 voisins donne des résultats satisfaisants.

II.C. Détection de lignes

La détection de ligne est une étape fréquente dans les algorithmes de compréhension d'image. Des méthodes comme la transformation de Hough (décrite § II.II.A) ou le détecteur de Burns sont possibles, mais la plupart nécessitent une recherche de contours, suivie par une étape dite de *composition de lignes* qui regroupe les pixels de contours en chaînes puis en lignes (le terme de *vectorisation* est parfois utilisé ici, mais nous le réservons pour une étape particulière de la composition de ligne).

II.C.1. Détecteur de lignes de Burns

II.C.1.a. Présentation

La méthode de Burns ([BURN86]) calcule le gradient de l'image et cherche les lignes directement, sans passer par une carte de contours. L'étape de lissage est effectuée en même temps que la dérivation lors du calcul du gradient par un masque. Un seuillage sur la longueur des lignes permet de les distinguer du bruit.

II.C.1.b. Définitions

Les définitions suivantes diffèrent de celles adoptées pour l'ensemble du document, et ne concernent que la méthode de Burns.

- *Ligne* : groupe connexe de pixels de contour alignés et de même direction.

- *Partition* : division de l'ensemble des directions possibles ([0;2π]) en un certain nombre d'intervalles égaux (appelés *classes*). L'algorithme demande l'utilisation de deux partitions décalées.
- *Classe* : intervalle d'une partition.
- *Classification* : action d'attribuer à chaque pixel une classe de direction en fonction de la direction de son gradient.
- *Région* : ensemble connexe de pixels dont la direction de gradient appartient à la même classe.
- *Segmentation* : action de découper la carte des contours en régions.
- *Support de ligne* : région dont les caractéristiques sont telles qu'elle est susceptible de représenter une ligne dans l'image.
- *Crédit* : information indiquant si une région est acceptée ou rejetée comme support de ligne.
- *Représentation* : segmentation basée sur une des deux partitions.
- *Demi-classe* : intervalle d'amplitude moitié de l'amplitude d'une classe, résultant de l'intersection de deux classes des deux partitions.

II.C.1.c. Algorithme

1. Calcul du gradient vectoriel de l'image (intensité et direction) et classification de chaque pixel.
2. Segmentation en fonction de la connexité des classes.
3. Analyse des caractéristiques des régions, acceptation ou rejet comme support de ligne.
4. Calcul des caractéristiques des supports de ligne (longueur, largeur, contraste…)

Il n'y a pas de seuillage sur l'intensité du gradient. De cette manière, une ligne peu contrastée est bien détectée.

La méthode nécessite le choix d'une partition des directions. Pour éviter qu'une ligne ayant une direction proche de la limite entre deux classes ne soit fragmentée lors de la segmentation, l'algorithme utilise deux partitions décalées d'un angle égal à la moitié de l'amplitude des classes. Ces partitions mènent, après segmentation, à deux représentations différentes. Une ligne fragmentée dans une représentation ne l'est pas dans l'autre. Un système de vote en fonction de la longueur des régions permet localement d'opter pour l'une ou l'autre représentation. L'ensemble des supports de lignes obtenus à l'issue des deux classifications comprend des régions de l'une et l'autre représentation.

L'acceptation ou le rejet d'une région comme région support de ligne se fait en tenant compte de sa longueur et d'un *critère de raideur*. Le critère de raideur peut être par exemple le coefficient de corrélation de l'ajustement plan (moindres carrés) de l'image sur la région, ou de l'ajustement linéaire de la courbe de niveau moyen (en géométrie spatiale, l'intensité lumineuse des pixels forme la troisième dimension).

Le calcul des caractéristiques des supports de ligne utilise le calcul d'une droite porteuse de la ligne. Cette droite est choisie passant par le centre de gravité de la région, pondérée par l'intensité du gradient, et perpendiculaire à la direction du gradient. Les extrémités de la ligne seront les extrémités du nuage de points formé par la projection des points de la région sur la droite porteuse (sur le plan de l'image en géométrie plane). Du bruit sur l'image peut avoir entraîné une erreur de classification, et donc des erreurs de segmentation. Ces erreurs pourront allonger la ligne de quelques pixels sans modifier les caractéristiques de direction ni de contraste.

II.C.2. Composition de lignes

La composition de lignes consiste à détecter, à partir d'un ensemble de pixels de contours, les lignes présentes dans l'image.

Plusieurs publications existent sur le sujet et les méthodes sont variées. On trouve néanmoins dans la plupart des cas trois étapes principales opérant à partir de cartes de contours fins. Une étape de *chaînage*, pour associer les contours continus en groupes, une étape de *vectorisation* qui sépare les groupes en suites de lignes et une étape de *restauration* qui consiste à prolonger de quelques pixels des lignes proches afin de reconstituer des lignes coupées ou des coins qui auraient disparu lors d'une étape précédente.

II.C.2.a. Méthode générale

Le chaînage consiste à regrouper les pixels de contours en chaînes (groupes connexes). Les pixels isolés et les chaînes de longueur inférieure à un certain seuil de longueur minimale seront éventuellement rejetées.

La vectorisation sépare ces chaînes en un ensemble de lignes jointives. Un critère de rectitude, associé à un seuil de rectitude, permet d'évaluer pour une chaîne s'il est possible de l'assimiler à une ligne. La surface entre une chaîne et sa corde est appelée *déviation surfacique* et peut être calculée de manière algébrique (lorsque la corde coupe la chaîne). La déviation surfacique et la flèche de la chaîne (critère de Pavlidis) peuvent entre autres être utilisées pour l'établissement du critère de rectitude.

Les étapes de chaînage et de vectorisation sont parfois associées en une seule procédure. L'information de la direction du gradient, qui est théoriquement orthogonale à celle du contour (et donc de la ligne) peut-être utilisée.

Il n'est pas rare qu'une ligne soit coupée ou disjointe d'une voisine avec laquelle elle devrait former un coin (notamment parce que certains opérateurs de dérivation rendent mal compte des contours sur ces zones). La restauration consiste à repérer les lignes dont les extrémités sont très proches (de l'ordre d'un ou deux pixels) et à les prolonger pour les joindre et les unifier si leurs orientations sont semblables. Ceci fait intervenir un seuil de distance maximale autorisant la jonction et un seuil d'écart angulaire maximal autorisant la fusion. Certains auteurs incluent la restauration dans l'étape de chaînage.

II.C.2.b. Méthodes existantes

Les auteurs de [NALW87] proposent une technique d'amincissement, puis de chaînage, puis d'ajustement de la conique des moindres erreurs (ellipse, hyperbole, cercle…) ; les auteurs détectent aussi les coins et segments de droite des formes obtenues.

D'autres auteurs présentent des techniques plus particulières, comme l'estimation par primitives de la forme non parasitée de l'image par la mesure du bruit ([CHUA93]), la minimisation de l'énergie sur des lignes ajustées ([NELS94]), le chaînage par prédiction / vérification ([MANS87])… Les auteurs de [SARK91] et [CHER93] développent des techniques de chaînage et de vectorisation à partir du détecteur de Canny.

Les trois méthodes présentées contiennent des particularités dont nous nous inspirerons.

Les auteurs de [DEVI93] et [NEVA80] font le calcul du gradient selon un certain nombre de directions données par convolution. L'amincissement est réalisé par élimination

des contours non maximaux dans la direction du gradient. Le chaînage est fait à partir de la connexité locale des pixels de contours. Les auteurs de [WALL84] présentent une méthode d'approximation polygonale d'une carte de chaînes.

II.C.2.b.i) Méthode de [DEVI93]

Le calcul du gradient ($\vec{\nabla} g(p)$) est effectué selon les directions horizontale et verticale. Ceci amène à l'établissement de deux cartes de contours : $\nabla g_x(p)$ et $\nabla g_y(p)$. Cette méthode a été conçue pour le repérage en robotique mobile en environnement structuré où l'abondance des lignes verticales et horizontales permet de ne pas utiliser les contours obliques. L'inconvénient est qu'elle sépare les contours verticaux des contours horizontaux, et le chaînage ne se fait pas en se basant sur les deux cartes mais sur chacune d'elle séparément. Les seuils de détections des contours ne sont intrinsèquement pas les mêmes selon leur direction. On peut démontrer en effet que pour deux lignes de même caractéristique, et d'orientation 0° et 45°, la ligne à 45° a dans les cartes de contours de cette méthode un gradient plus faible que l'autre d'un facteur √2, et donc une probabilité de détection moindre.

La vectorisation est faite de manière itérative le long des chaînes, et le critère de rectitude est incrémental. Cette méthode a l'inconvénient de ne pas donner les mêmes résultats selon que l'on vectorise une chaîne par une extrémité ou l'autre. Il n'y a donc pas de reproductibilité du résultat selon le point d'entrée du contour.

Le critère de rectitude utilisé exige deux conditions. Il demande d'une part que la direction du contour que l'on rajoute soit globalement le même que la direction de la ligne en cours de construction, et d'autre part que la déviation surfacique de la chaîne mesurée sur le point courant soit faible.

II.C.2.b.ii) Méthode de [NEVA80]

Le calcul du gradient est effectué par convolution avec des masques directionnels de pas 30° de taille 5×5.

La condition de chaînage est que les directions de contour soient semblables. La faible valeur du pas (30°) peut amener des jonctions de plusieurs contours qui compliquent le chaînage.

La vectorisation adopte une approche globale. Pour l'ensemble d'une chaîne, si certains points de la chaîne sont trop éloignés de la ligne joignant ses extrémités, alors, la chaîne est séparée en deux et chaque chaîne est alors vectorisée séparément récursivement.

II.C.2.b.iii) Méthode de [WALL84]

L'algorithme calcule une approximation polygonale d'une carte de contours chaînés. Le critère de rectitude utilise la déviation surfacique algébrique construite sur le point courant du parcours de la chaîne. Les dépassements de la chaîne de part et d'autre de sa corde se compensent. Ceci autorise la vectorisation d'une chaîne ondulée en une ligne unique tant que la direction générale est bonne.

L'intérêt de ce critère est la simplicité de sa mise en oeuvre de manière incrémentale. Si O est l'origine de la chaîne, D_s la déviation surfacique algébrique et P_i et P_{i+1} les points courant et suivant de la vectorisation, la déviation surfacique algébrique après ajout du point P_{i+1} se calcule en cumulant la surface algébrique du triangle (O,P_i,P_{i+1}) avec D_s.

II.D. Conclusion

Une analyse des différentes méthodes utilisées pour résoudre des problèmes similaires nous permet d'établir quelques éléments et principes généraux pour réaliser notre propre méthode spécifique au problème posé.

Nous avons vu que les méthodes pouvaient globalement être décomposées en approches indirectes (transformations) ou en approches directes. Les approches indirectes ayant été rejetées car trop coûteuses, nous avons étudié en détail les méthodes à base de segmentation par texture et détection de contours, chaînes de contours et lignes.

Les critères de texture proposés dans [Casa92]-G nous serviront pour élaborer des critères de textures.

La réalisation d'un détecteur de contour est nécessaire. Nous avons étudié les opérateurs de détection de contour à base de dérivation du premier ordre (gradient) ou du second ordre (laplacien). Nous avons vu pour chacun les propriétés de détection, précision, réponse au bruit, minceur des résultats… Ces observations nous permettront de déterminer le ou les détecteurs les plus adéquats pour notre problème, en fonction de la nature des images traitées et des informations que nous voulons extraire de la carte de contour (texture, lignes…).

La détection de lignes dans l'image nécessite que les contours détectés soient minces, afin de pouvoir les chaîner et les vectoriser, en utilisant éventuellement les directions du contour.

Nous allons reprendre un certain nombre des méthodes présentées dans ce chapitre et les développer plus en détail. L'étude détaillée de ces méthodes fait l'objet du chapitre suivant.

Chapitre III

Méthodes développées

À partir de méthodes trouvées dans la littérature, nous avons développé des algorithmes dédiés à la lecture de codes Data Matrix.

L'ensemble est constitué de deux algorithmes indépendants : la recherche de symbole et la lecture.

Nous allons étudier en détail les méthodes mises en œuvre par ces algorithmes. La plus grande partie de ce chapitre est consacrée à l'algorithme de recherche de symbole qui fait appel à deux méthodes issues de principes différents. Nous présentons d'abord la méthode basée sur la recherche de lignes (dont certains éléments ont été exposés dans [MARC97]), puis la méthode basée sur l'analyse de texture. Ces deux méthodes sont basées sur des cartes contours calculées par le passage à zéro du laplacien. Nous décrivons ensuite comment ces deux méthodes sont associées pour réaliser la recherche de symbole. Nous abordons en quatrième partie la lecture de symbole.

III.A. Méthode par recherche de lignes

Le symbole Data Matrix se caractérise par deux lignes pleines (voir § I.A.5). L'approche par analyse de lignes consiste à chercher dans l'image les lignes susceptibles de représenter les deux lignes pleines du symbole.

La recherche de ligne se fait selon les cinq étapes suivantes.

1. Calcul des directions de contour par estimation du vecteur gradient par convolution.
2. Détection de contours fins aux passages à zéro du laplacien.
3. Chaînage en courbes de contour en utilisant la direction du gradient.
4. Vectorisation de courbes de contour en lignes.
5. Restauration de lignes : jonctions des lignes quasi jointives et unification des lignes jointives quasi parallèles.

La recherche de ligne est suivie par une recherche de symbole, qui comprend les étapes suivantes.

6. Recherche de coins
7. Recherche de couples de lignes en *L*

La figure III-1 présente l'enchaînement des opérations.

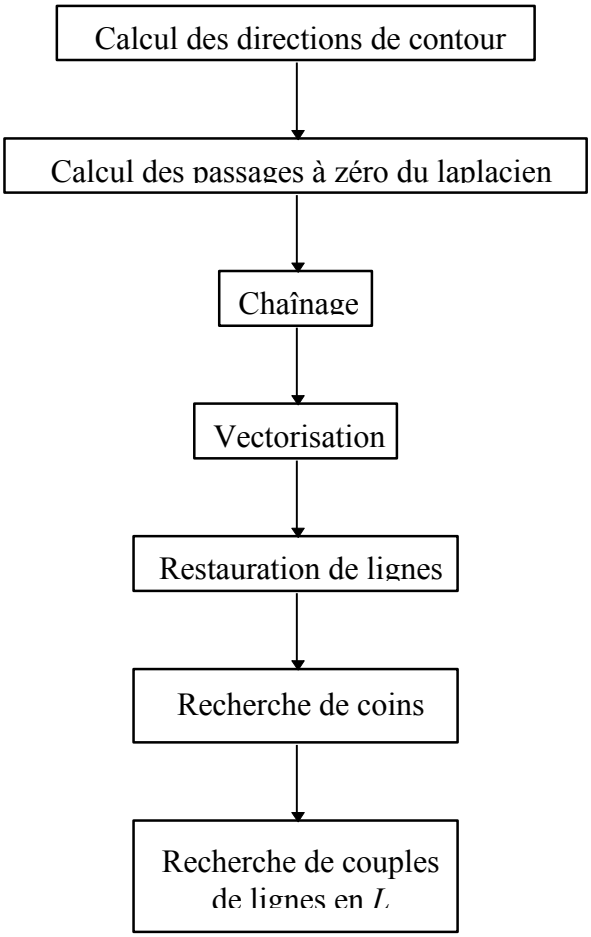


FIG. III-1 - *Recherche de symboles par recherche de lignes*

III.A.1.Calcul des directions de contour

La direction du contour est la direction des lignes de niveau de luminance sur la transition. La direction du gradient est orthogonale à ces lignes de niveau.

Un contour est caractérisé par sa position, sa direction, son amplitude et son sens. On peut donc représenter un pixel de contour par un point d'application (position) et un vecteur de contour (direction, sens, amplitude). Un tel couple (point, vecteur) est équivalent à un segment orienté.

La figure III-2 représente un contour, avec sa direction et son gradient. Si on mesure l'amplitude du contour par la valeur de son gradient, on peut indifféremment utiliser le vecteur de contour ou le vecteur gradient pour représenter le contour.

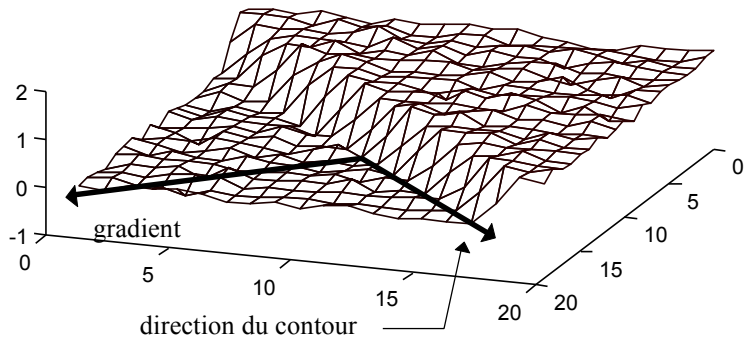


FIG. III-2 - *contour et gradient*

Le sens du vecteur indique le sens de la transition en luminance et est conventionnel. Si l'on veut représenter le gradient, on utilise le sens donné par la dérivation $\frac{\partial g(x,y)}{\partial x}$ et $\frac{\partial g(x,y)}{\partial y}$. Ce sens donne un vecteur orienté vers le bas de la transition. Si l'on veut représenter le contour, on prend le sens du vecteur orthogonal au gradient faisant avec lui un angle de $+\pi/2$ orienté. Si $\vec{c}$ représente le vecteur contour, alors $(\vec{\nabla}g, \vec{c}) = +\pi/2$ selon le repère $(\vec{x}, \vec{y})$ évoqué (représenté figure III-6).

Le calcul des directions de contour se fait par convolution avec deux masques qui donnent les deux coordonnées du gradient pour chaque point de l'image. Les matrices utilisées sont les matrices $\begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$ et $\begin{pmatrix} 1 & -1 & 0 \end{pmatrix}$ qui correspondent aux opérations de dérivation : $\frac{\partial g(x,y)}{\partial x} = g(i, j+1) - g(i,j)$ et $\frac{\partial g(x,y)}{\partial y} = g(i-1, j) - g(i,j)$ en prenant $\vec{x} = \vec{j}$ et $\vec{y} = -\vec{i}$. Ces directions sont représentées plus loin (figure III-6) ; on peut noter qu'elles mènent à un calcul de gradient légèrement différent de celui réalisé par les masques linéaires présentés au chapitre II (§ II.B.5.a).

Ces opérateurs sont assez sensibles au bruit et peu précis en localisation. On peut le voir sur les images des figures III-3 et III-4 qui représentent les réponses à la détection de contour par ces masques et par le masque du laplacien de White-Rohrer (voir comparaison détaillée dans le chapitre II, § II.B.5.h). Pour cette raison, ils ne sont pas utilisés pour faire de la détection de contour ; par contre, ils permettent de connaître la direction des contours avec suffisamment de précision pour réaliser cette étape.

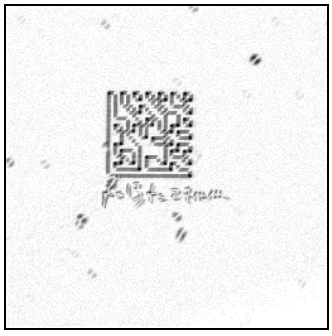


FIG. III-3 - *détection de contour par gradient selon x et y*

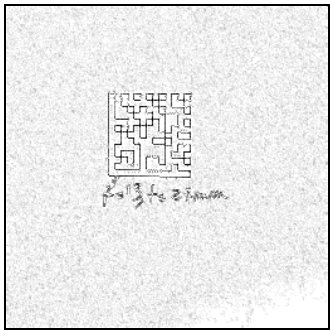


FIG. III-4 - *détection de contour par laplacien*

Dans la suite des opérations, nous n'avons pas besoin d'une grande précision sur la direction du gradient ; nous découpons le cercle trigonométrique en huit classes de directions représentées figure III-5. En affectant une classe de direction à chaque pixel de contour, nous avons une erreur maximum de 22,5°, huit directions possibles, et un codage de la direction sur trois bits.

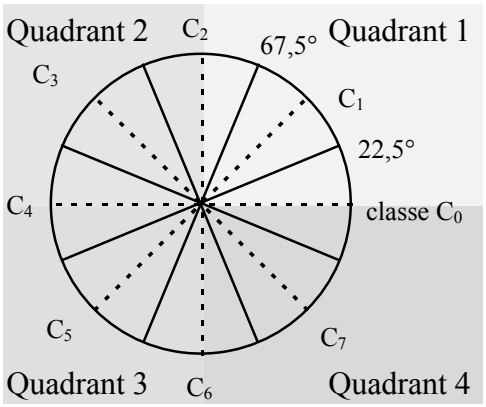


FIG. III-5 - *répartition des classes de direction*

L'attribution d'une direction de contour à un pixel se fait à partir de la valeur des deux coordonnées (signes et valeur du quotient) du gradient.

Soit un vecteur $\vec{u}$ de coordonnées (x, y) et sa direction $\vec{e}_u$. En fonction du signe des coordonnées horizontale (x) et verticale (y), nous déterminons dans quel quadrant se situe $\vec{e}_u$. Nous comparons le rapport $|y|/|x|$ aux tangentes des angles 22,5° et 67,5° et déterminons la classe de la direction du vecteur ($|x|$, $|y|$). En combinant cette classe avec la connaissance du quadrant de $\vec{e}_u$, nous déterminons rapidement sa classe. Cette solution est plus rapide que celle qui consiste à faire le calcul de $\arctan\left(y/x\right)$.

À l'issue de ce calcul, nous avons une carte des directions de contour indiquant pour chaque pixel la direction de son gradient sur trois bits. Cette carte est établie pour tous les pixels, quelle que soit l'amplitude du gradient. Cette carte n'indique donc pas quels pixels sont des pixels de contours.

III.A.2.Détection de contours

La recherche de contour se fait en utilisant les opérateurs de détection de passage à zéro du laplacien évoqués dans le chapitre II. Ceci se fait en deux étapes : calcul du laplacien à partir du masque de White-Rohrer à 4 voisins puis calcul de ses passages à zéro.

III.A.2.a. Calcul des passages à zéro du laplacien

Pour chaque pixel $p_{(i,j)}$ de l'image (à l'exception de ceux du bord de l'image), on lit la valeur du laplacien aux points $p_{(i,j)}$, $p_{v(i+1,j)}$ et $p_{h(i,j+1)}$ (voir figure III-6). S'il y a un changement de signe entre deux voisins $p_{(i,j)}$ et p_k, on dit qu'il y a entre ces deux voisins un passage à zéro ZC de valeur $ZC=|p_{(i,j)}-p_k|$. Le passage à zéro du laplacien au point $p_{(i,j)}$ sera le plus grand des passages à zéro entre (p, p_v) et (p, p_h).

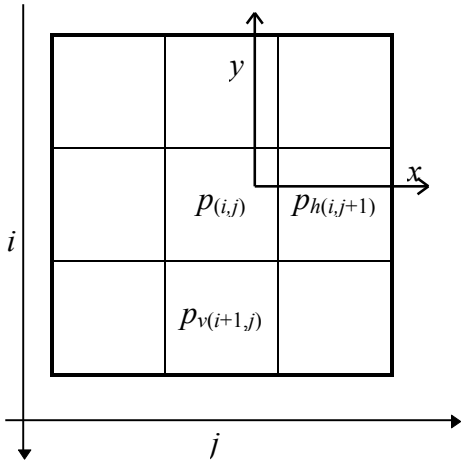


FIG. III-6 - *voisinage considéré pour la recherche du passage à zéro du laplacien*

Cette méthode ne détecte que les passages à zéro sur ligne ou colonne, sans tenir compte du fait que le passage à zéro peut être un point d'inflexion sans contour (lorsque le signe de la dérivée première est différent du signe de la dérivée troisième, voir chapitre II, figures II-15.b et II-15.d). Leur identification et leur rejet nécessiteraient une dérivation du troisième ordre (f'''), mais il n'est pas apparu nécessaire de l'effectuer, car ces fausses alarmes disparaissent dans les étapes ultérieures.

III.A.3.Amincissement

Comme on l'a vu dans le chapitre II, la recherche du passage à zéro du laplacien donne théoriquement des contours fins. Cependant, en raison de la discrétisation du signal, il arrive que ce ne soit pas le cas. En particulier, le fait d'utiliser une carte ayant le même nombre de pixels que l'image fait projeter sur un même pixel de la carte plusieurs contours possibles (contour sur le voisinage vertical et sur le voisinage horizontal). Ceci est en fait une réduction d'une carte de contour entre-pixel en une carte de contour sur-pixel (voir § II.B.5.d). Des pixels de contour qui n'auraient pas du être voisins se trouvent ainsi rapprochés, et des contours voisins se trouvent confondus ; tout ceci crée des agglomérations de contours, notamment aux alentours des intersections de chaîne et des lignes antiparallèles proches (lignes parallèles de sens différents).

Ayant besoin de contours fins pour effectuer le chaînage, nous corrigeons l'image par la suppression des contours non maximaux (voir § II.C.2.a). Le nombre de pixels ainsi éliminés est relativement faible.

Nous avons représenté l'extrait d'une image où se trouvait un coin sur la figure III-7 et les directions et amplitudes des contours de cet extrait sur la figure III-8. Lorsqu'il y a un contour entre deux pixels, le contour est attribué au pixel de plus faible intensité (le plus sombre). Les contours d'amplitude nulle ne sont pas représentés. Pour chacun de ces contours, l'intensité est indiquée ainsi que la direction. La longueur de la flèche est proportionnelle à l'amplitude. On voit que le pixel de contour qui fait le coin (d'amplitude 164) a, dans la direction de son gradient, un voisin pixel de contour d'amplitude plus élevée (204). Ce pixel du coin ne sera pas présent dans la carte de contours minces.

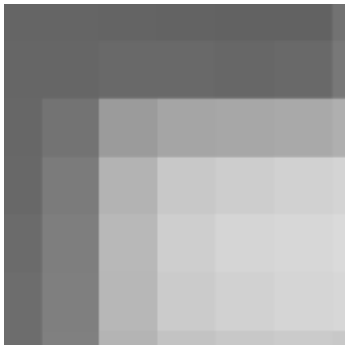


FIG. III-7 - *extrait d'une image au niveau d'un coin*

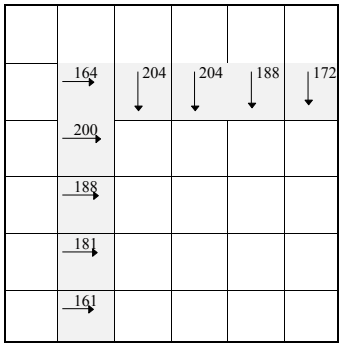


FIG. III-8 - *direction et amplitude du gradient dans cet extrait*

III.A.4.Chaînage

Une fois que les contours ont été amincis, les pixels de contour sont regroupés en chaînes. Le chaînage est effectué selon un voisinage à 8 pixels, comme celui représenté figure III-6.

Le chaînage entre un pixel de contour et son voisin ne sera effectué que si les directions de gradient sont compatibles entre elles et avec la direction du voisinage considéré (même classe ou de classes voisines).

La figure III-9 est un extrait d'une image au voisinage d'un coin. La figure III-10 indique pour chaque pixel de cet extrait les amplitude et direction du gradient.

Sur les pixels ***d***, ***e*** et ***q*** de la figure III-10, nous avons représenté en pointillé et en longueur constante la direction du contour à côté du vecteur gradient. La direction du contour sur le pixel ***d*** est telle qu'elle ne permet comme successeur que l'un des trois pixels ***b***, ***e*** ou ***q***. Le plus proche de ces trois (***e***) présente un contour dont la direction est telle qu'elle ne permet comme prédécesseur que l'un des trois pixels ***b***, ***a*** ou ***d***. Le chaînage ***d-e*** est donc compatible avec la direction de chacun des deux contours. D'autre part, les directions de contour des pixels ***d*** et ***e*** appartiennent à des classes voisines (respectivement 0 et 7 modulo 8). Nous pouvons alors chaîner dans cet ordre ***d*** et ***e***.

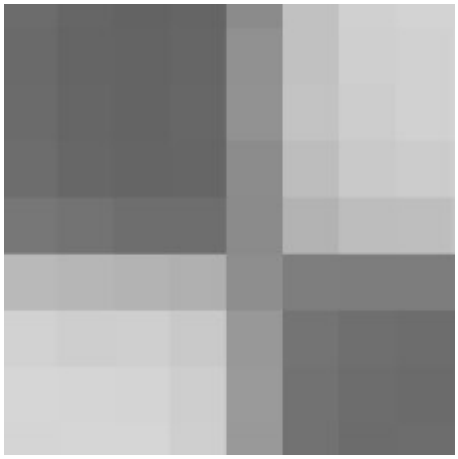


FIG. III-9 - *extrait d'une image*

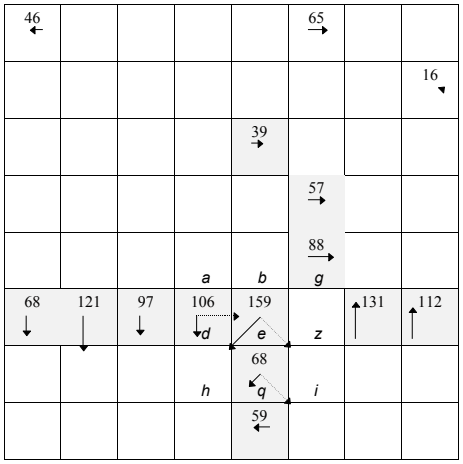


FIG. III-10 - *direction et amplitude du gradient dans cet extrait*

Le pixel suivant est q car celui-ci est un pixel de contour appartenant aux trois successeurs possibles de e (z, i et q), e est un prédécesseur possible de q (parmi e, d et h), et les directions de contours des deux pixels e et q sont identiques.

Pour analyser l'image de contour et en extraire les courbes de contour, nous parcourons l'image selon un balayage à deux dimensions en omettant les pixels déjà analysés. Nous interrompons le balayage à chaque fois que nous trouvons un pixel appartenant à une nouvelle courbe de contour pour chaîner ses éléments en la suivant et en mémorisant chaque pixel traité au fur et à mesure.

Le chaînage d'un pixel de contour avec ses voisins selon sa direction dans un sens particulier est une procédure récursive. On détermine les trois pixels voisins dans la direction du gradient et de ses deux classes voisines dans cet ordre. Le premier de ces voisins qui est un pixel de contour et dont la direction de contour est compatible avec la direction du voisinage est déclaré être le pixel suivant (ou précédent) dans la courbe de contour. On mémorise le pixel qui vient d'être traité et on cherche le pixel suivant (ou précédent) du pixel que l'on vient de rajouter à la chaîne. Lorsque l'on arrive à la fin de la chaîne, on reprend le balayage là où il s'était arrêté.

Il n'y a pas de seuillage dans la recherche des passages à zéro du laplacien. Il s'ensuit que la carte de contours, avant le chaînage, contient de nombreux pixels correspondant à de faibles variations de luminance ne représentant pas de contours significatifs de l'image. Ces fausses alarmes sont relativement aléatoirement distribuées et s'associent rarement en blocs connexes de plus de trois pixels. L'élimination des petites chaînes (de longueur inférieure à 3 ou 4 pixels) à l'issue de l'étape de chaînage élimine les pixels de bruit qui étaient présents dans la carte de contours, tout en préservant les lignes (de longueur supérieure à 3 ou 4 pixels) de faible contraste.

Les pixels représentant des contours minces, le résultat est un ensemble de chaînes d'un pixel de large. Le chaînage se fait selon la direction des contours. La chaîne résultante sera elle-même orientée et peut être interprétée comme délimitant deux parties de l'image de luminances différentes. En prenant la même convention d'orientation que pour les pixels de contour, on prend le sens de la chaîne tel que la zone de faible luminance (zone foncée) se trouve à sa droite. On peut voir cette orientation sur la figure III-2.

III.A.5. Vectorisation

Les chaînes sont analysées une à une selon la méthode de Pavlidis. Une corde est construite sur chacune. Chaque chaîne est fragmentée en deux en son point le plus éloigné à la corde si cette distance est supérieure à un certain seuil appelé *seuil de rectitude*. Le processus est répété récursivement.

La figure III-11 représente une chaîne joignant deux points *A* et *B*. Le point de la courbe le plus éloigné de la corde [*AB*] est le point *C*. On construit alors les cordes [*AC*] et [*CB*] et on fait la vectorisation des courbes joignant *B* à *C* et *C* à *A*. Cette vectorisation amène à couper ces courbes respectivement en *E* et *D* et ainsi de suite autant de fois que nécessaire. La ligne [AB] sera orientée de la même façon que la chaîne (la zone de faible luminance, foncée, limitée par la ligne se trouve à sa droite).

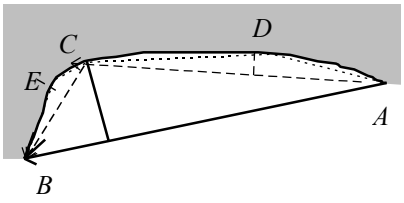


FIG. III-11 - *vectorisation d'une chaîne*

Les résultats de la vectorisation sur l'ensemble de l'image représentent un ensemble de lignes qui seront corrigées dans l'étape suivante. Les figures III-12 à III-17 présentent deux images ainsi que les lignes extraites à partir de ces images représentées seules et en surimpression. Nous voyons notamment sur les lignes de la figure III-13 que certaines lignes sont discontinues (par exemple, la ligne horizontale sous la ligne pleine du motif en *L*).

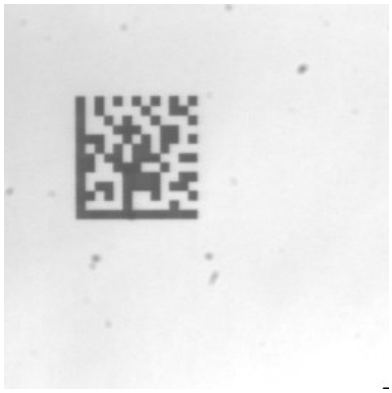


FIG. III-12 - *image de symbole*

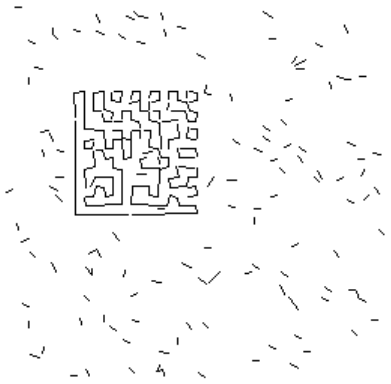


FIG. III-13 - *lignes détectées*

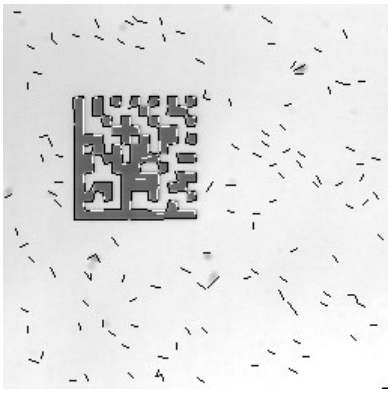


FIG. III-14 - *lignes représentées sur l'image*

FIG. III-15 - *James West (Robert Conrad) dans* The Wild Wild West *(1965)*

FIG. III-16 - *lignes détectées*

FIG. III-17 - *lignes représentées sur l'image*

III.A.6.Restauration de lignes

Les lignes obtenues à l'étape précédente présentent des défauts par rapport aux lignes recherchées. Le principal est la disparition des coins. Puisque nous utilisons des contours minces et de directions de classes voisines de proche en proche, un pixel de contour ne peut y avoir que deux voisins et ne peut pas former avec eux d'angle droit. Nous ne pouvons donc pas avoir d'intersection de lignes, ni de lignes jointives avec un angle aigu inférieur à 45°. L'image originale étant la représentation d'une image binaire pour la zone du symbole, il ne peut pas y avoir d'intersections de lignes. Par contre, deux lignes qui devraient former un angle droit seront disjointes d'un pixel au moins dans notre représentation.

Sur la figure III-13, les lignes pleines du motif en *L* sont représentées par deux longues lignes orthogonales. La disjonction évoquée au paragraphe précédant ne se voit pas car les deux extrémités qui forment l'angle sont sur deux pixels voisins. Ces deux extrémités sont donc distantes d'exactement 1 pixel et semblent former deux segments sécants.

La restauration de ligne consiste à rechercher et raccorder dans l'ensemble de lignes les lignes aux extrémités puis directions proches (voir § II.C.2.a).

Dans notre application, seules les lignes qui sont parallèles entre elles (ligne fragmentée) ou orthogonales (coin d'un symbole) sont intéressantes. Nous travaillons d'autre part avec des lignes orientées, représentant des contours de formes fermées. Ces contraintes nous permettent d'établir quelques règles sur la façon dont les lignes se joignent.

La figure III-18 montre des exemples d'objets présents dans l'image et les lignes résultantes (elles ont été légèrement décalées pour plus de lisibilité).

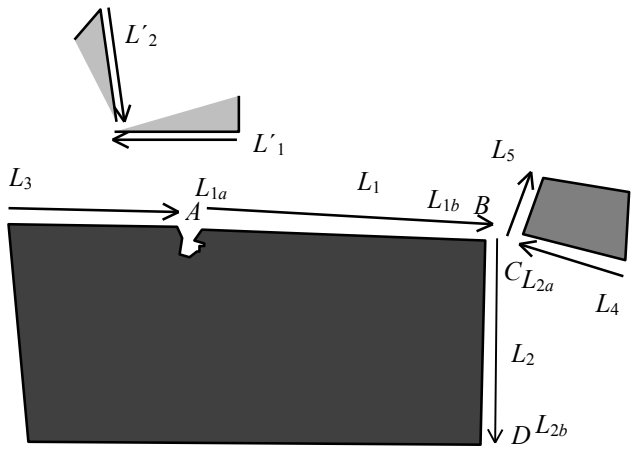


FIG. III-18 - *objets, contours, et lignes associées*

Les lignes étant orientées, on peut voir facilement que si deux lignes représentent deux parties jointives du contour d'un même objet, elles s'enchaînent dans le même sens (L_3, L_1, L_2). Les lignes L'_1 et L'_2 de l'exemple ne doivent pas être jointes, alors qu'elles ont bien une extrémité commune

Soient deux lignes $[\overrightarrow{AB}]$ et $[\overrightarrow{CD}]$. Pour une ligne L, nous notons L_a et L_b respectivement l'origine (ou première extrémité) et la seconde extrémité de la ligne.

Pour réaliser la restauration des lignes, les lignes sont examinées une à une, après avoir été classées par ordre de longueurs décroissantes. Lorsque L_1 est examinée, nous cherchons un point proche de L_{1b} d'une ligne L_2 tel que (entre autres) L_{1b} et L_{2a} soient proches. Si la ligne L_1 est susceptible d'être jointe à une ligne L_3 par son point L_{1a}, cette jonction est réalisée lorsque L_3 est examinée (ces détails peuvent être suivis avec la figure III-18).

La jonction entre L_1 et L_2 se fait dans les conditions suivantes.

Nous cherchons au voisinage de L_{1b} la ligne L_2 telle que L_2 n'ait pas déjà été jointe (test Rl), et pour laquelle la distance $L_{1b}L_{2a}$ est minimale. Une fois la ligne L_2 obtenue, nous effectuons la jonction si la distance $L_{1b}L_{2a}$ est suffisamment faible (test Rd) et si l'angle (L_1, L_2) est proche de l'angle plat (0°, test R0) ou de l'angle droit (±90°, test R9). Si ces critères sont vérifiés, nous calculons le milieu du segment [$L_{1b}L_{2a}$] et décalons chacune des extrémités L_{1b} et L_{2a} des lignes concernées pour réaliser la jonction sur ce milieux.

Lors de cette jonction, la longueur des lignes peut changer, mais l'ordre d'inspection de lignes décidées au début de la procédure (par longueurs décroissantes) reste inchangé.

Si lors de la suite de l'inspection, une ligne L_4 semble susceptible d'être jointe à L_2, cette jonction ne se fait pas (suite au test Rl), car L_2 a déjà été jointe avec L_1. Effectuer la jonction L_4-L_2 reviendrait à briser la jonction L_1-L_2, or cette dernière est plus intéressante car L_1 est plus long que L_4 (en raison de l'ordre d'inspection des lignes).

Les lignes des figures III-19 et III-20 montrent l'ensemble de lignes avant et après la restauration. Nous voyons par exemple l'effet de la restauration sur les deux lignes du motif en L.

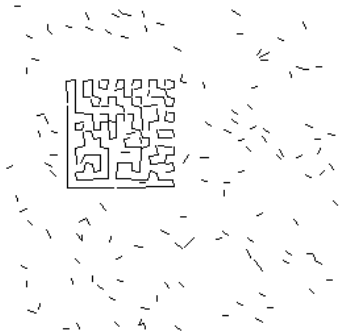


FIG. III-19 - *image de lignes avant la restauration (rappel de la figure III-13)*

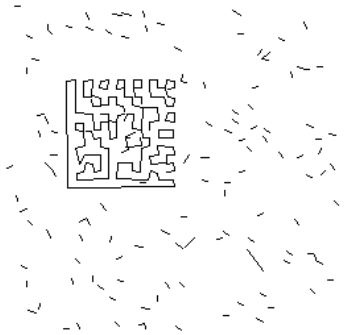


FIG. III-20 - *image des lignes après la restauration*

III.A.7.Recherche du motif en L

L'ensemble des lignes trouvées à l'étape précédente est analysé pour rechercher les lignes susceptibles de représenter le motif en *L*. Si de telles lignes L_1 et L_2 sont présentes dans l'image, elles ont les propriétés suivantes :

- Propriétés géométriques des lignes : les deux lignes sont jointives, orthogonales et de longueurs semblables ;
- Propriétés géométriques de la zone délimitée : la longueur des lignes est suffisamment importante pour que la zone délimitée puisse contenir un symbole ;
- Propriétés de l'image dans la zone délimitée : le coin délimité par les deux lignes est dans la zone sombre que délimite chacune d'entre elles (ou dans la zone claire si l'on recherche des symboles imprimés en négatif).

L'évaluation de ces propriétés passe par un certain nombre de calculs et de comparaisons à des seuils. Pour les deux premières, la connaissance des longueurs de ligne est nécessaire (elles sont calculées au moment du tri).

Propriété 1

- La jonction se fait avec deux lignes ayant une extrémité commune compatible avec leur sens (voir figure III-21).
- Leur orthogonalité est estimée en comparant l'angle qu'elles font entre elles avec un seuil. On se contente en fait de comparer leurs cosinus, plus faciles à calculer (voir annexe D).
- Comparaison des longueurs : le rapport des longueurs doit appartenir à un certain intervalle fixé à $\left[\frac{3}{5} ; \frac{5}{3}\right]$ (à peu près $[0{,}6 ; 1{,}67]$). Les deux bornes de l'intervalle sont inverses (l'intervalle est symétrique par rapport à l'origine dans l'espace logarithmique : $\left[\log(\frac{3}{5}) ; -\log(\frac{3}{5})\right]$) afin que la condition soit la même pour (L_1, L_2) et (L_2, L_1).

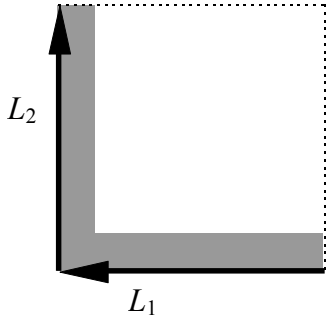


FIG. III-21 - *position des lignes par rapport aux lignes pleines du motif de détection en L*

Propriété 2

- Les longueurs de lignes doivent être supérieures à un certain seuil.

Propriété 3 :

Celle-ci se décompose en deux propriétés faciles à vérifier sur le couple (L_1, L_2).

- L_1 et L_2 forment un coin : les deux lignes sont jointives et telles que la pointe L_1 est confondue avec l'origine de L_2. Si c'était la seconde extrémité de L_2 qui était confondue avec l'origine de L_1, on parlerait du couple (L_2, L_1).
- Le coin formé doit être une cellule marquée (c'est à dire des pixels sombres). La zone sombre est par convention située à la droite d'une ligne orientée ; la ligne L_2 est donc orientée vers la droite par rapport à la ligne L_1.

Nous cherchons les couples possibles (L_1, L_2) en commençant par les lignes les plus longues. Nous pouvons donc dresser une liste de candidats, où les premiers de la liste correspondent aux symboles probables ayant les plus grandes dimensions.

III.A.8.Détermination du coin d'horloge : recherche de coins

III.A.8.a. Détermination du coin d'horloge

Les deux lignes permettent de connaître trois des quatre points qui définissent la position du symbole. Le quatrième point est appelé le *coin d'horloge*, car il correspond à l'intersection des zones d'horloge.

On détermine le coin d'horloge *nominal* en construisant un parallélogramme sur les trois premiers points comme indiqué figure III-21. Si les distorsions géométriques ne sont pas très importantes, le coin d'horloge est proche du coin nominal.

Dans l'image, si la fréquence du symbole est impaire, la cellule associée au coin d'horloge est marquée et visible, et ce point correspond au coin de cette cellule le plus éloigné du centre du symbole. Dans le cas contraire, le coin d'horloge n'est pas matérialisé.

Nous avons développé une méthode pour rechercher les coins dans une image à partir de la carte de contours et des autres informations disponibles. Nous cherchons des coins au voisinage du coin d'horloge nominal. Ce voisinage est construit pour un rayon donné avec la distance d_∞ (valeur maximale des distances sur x et y) ; il est donc de forme carrée. Les coins trouvés correspondent aux coins des cellules de cette zone, et le coin d'horloge est celui

d'entre eux le plus éloigné du centre du symbole. Si le symbole est de fréquence paire, il n'y a pas de coin dans le voisinage du coin d'horloge nominal.

III.A.8.b. Recherche de coins

Les coins qui nous intéressent sont les coins formés par des lignes orthogonales. Pour détecter ces coins, nous cherchons les zones de l'image présentant des contours dans deux directions orthogonales. Les coins sont classés en deux catégories : les coins droits et les coins obliques. Les coins droits sont les coins dont les lignes porteuses ont des directions de classe paire (selon le découpage en classes de direction vu au § III.III.A et figure III-5), c'est à dire des directions proches de 0° ou 90°, dites *directions droites*. Les coins obliques sont les coins dont les lignes porteuses ont des directions de classe impaire, c'est à dire des directions proches de ±45°, dites *directions obliques*. Les deux lignes d'un coin étant orthogonales, leurs directions sont ensemble soit droites, soit obliques.

Nous effectuons des convolutions de l'image avec des masques de type compass qui répondent avec une valeur absolue élevée sur des directions de contour particulières. Ils sont utilisés pour détecter les points qui présentent deux contours dans deux directions orthogonales, soit droites, soit obliques (voir table III-1). Nous n'utilisons que quatre des huit masques compass car le sens n'est pas significatif (nous utilisons la valeur absolue de la réponse du masque).

Nom du masque	Masque appliqué	Contours détectés
Sud	$M_S = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	contours horizontaux, directions de classe 0 et 4.
Est	$M_E = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$	contours verticaux, directions de classe 2 et 6.
Sud-ouest	$M_{SO} = \begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$	contours à +45°, directions de classe 1 et 5.
Nord-ouest	$M_{NO} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$	contours à -45°, directions de classe 3 et 7.

TABLE III-1 - *masques directionnels pour la détection de coins*

Les figures III-22 et III-23 montrent les classes de direction de contour au voisinage d'un coin et d'une ligne. Sur un coin, les quatre détecteurs directionnels réagissent, alors que seulement trois d'entre eux réagissent sur une ligne ou une chaîne. En particulier, un seul détecteur droit (resp. oblique) réagit sur une ligne droite (resp. oblique).

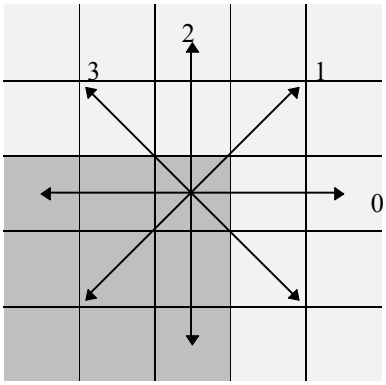


FIG. III-22 - *directions des masques directionnels par rapport à un coin droit*

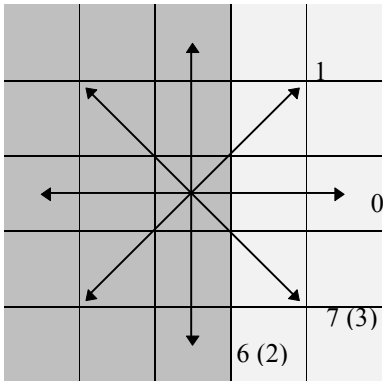


FIG. III-23 - *directions des masques directionnels par rapport à une ligne droite*

Si nous voulons détecter des coins droits (resp. obliques), nous cherchons les pixels pour lesquels les deux masques droits M_S et M_N (resp. obliques M_{SO} et M_{NO}) réagissent. L'orientation du quadrilatère étant connue, nous sommes donc en mesure de savoir si les coins à chercher sont droits ou obliques.

Nous déterminons les deux diagonales du quadrilatère, puis les deux bissectrices des diagonales. Ces deux diagonales sont orthogonales. Si elles ont des directions droites, nous supposons que le quadrilatère est oblique, et à l'inverse, si les directions des bissectrices sont obliques, que le quadrilatère est droit. Selon le cas, les convolutions avec les matrices M_S et M_N ou M_{SO} et M_{NO} sont effectuées.

Nous voyons figures III-24 à III-26 les résultats d'une recherche de coins obliques et droits sur des images présentant des coins des deux catégories. Selon les masques utilisés (M_S et M_N ou bien M_{SO} et M_{NO}), la carte obtenue n'est pas la même.

FIG. III-24 - *image contenant des coins droits (symbole du bas et images d'arrière plan) et des coins obliques (symbole du centre)*

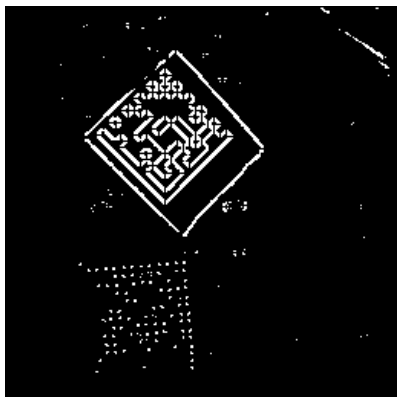


FIG. III-25 - *carte des coins droits (ceux du symbole du bas sont détectés)*

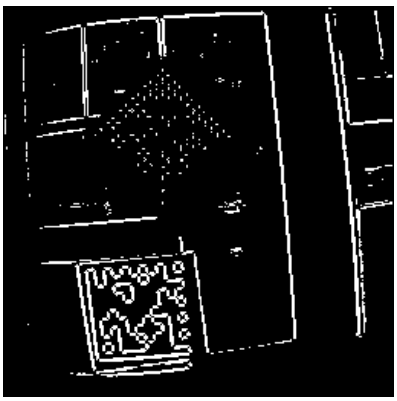


FIG. III-26 - *carte des coins obliques (ceux du symbole du centre sont détectés)*

III.A.9.Résultats de la recherche par les lignes

Les figures III-27 et III-28 montrent les résultats de recherche et de détection de symbole par détection de lignes. Le symbole présent dans l'image a été correctement détecté et localisé à partir des lignes comme on le voit figure III-29.

FIG. III-27 - *image d'un symbole Data Matrix*

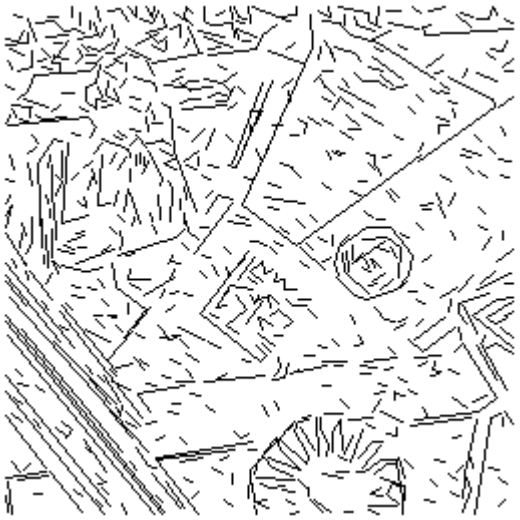


FIG. III-28 - *détection des lignes sur l'image de la figure III-27*

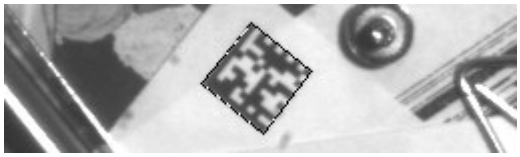


FIG. III-29 - *localisation du symbole*

III.B.Méthode par analyse de texture

L'approche par texture est basée sur le calcul d'un certain nombre de critères de texture dont la combinaison permet d'estimer la présence ou non de symbole.

Pour couvrir les différentes fréquences spatiales que peut présenter un symbole, on calcule les cartes de texture sur différentes échelles dont les valeurs sont en inverse de puissances de 2 (½, ¼, ⅛ …). Nous notons h l'échelle utilisée et appelons *niveau d'échelle* la valeur n_h telle que $h=2^{-n_h}$.

Pour calculer une carte de texture à une échelle donnée, nous réalisons un sous-échantillonnage de l'image. L'image à une échelle de niveau n_h s'obtient à partir de l'image à l'échelle de niveau n_h-1 en donnant à chaque pixel la valeur moyenne du bloc de 2×2 pixels correspondant. Le support de l'image obtenue a une taille deux fois plus petite, et nous pouvons y appliquer les critères de texture.

À l'échelle h=1 (n_h=0), nous utilisons des texels de taille t_x. Une classification par texture à une échelle 2^{-n_h} est équivalente à une classification avec des texels de taille $2^{-n_h}\ t_x$ et des critères dont la normalisation a été ajustée pour tenir compte de la taille des texels.

L'expérimentation nous a permis de déterminer et de développer trois critères significatifs pour notre application [DÉLE96]. Les critères sélectionnés sont tous les trois calculés à partir de la carte de contour. Cette carte de contour est calculée par la même méthode que pour la détection de lignes (passage à zéro du laplacien), avec un seuil d'amplitude de contour qui élimine les passages à zéro de faible amplitude. Nous choisissons un seuil relativement faible afin d'obtenir les contours, même avec une image de faible contraste. Ce choix reste critique,

car un seuil trop bas augmente le nombre de fausses alarmes, et un seuil trop haut élimine les contours significatifs.

III.B.1.Analyse de texture

Les critères utilisés s'appliquent sur l'image de contours. Ils représentent la quantité dans chaque texel des pixels de contours ayant telle caractéristique.

Les critères dits *positifs* réagissent à la présence de symbole (leur valeur absolue est élevée dans les zones où un symbole est présent) ; les critères dits *négatifs* répondent de manière inverse et sont des critères d'exclusion. Les figures III-30 et III-31 représentent des images de contour sur lesquelles ces critères sont appliqués.

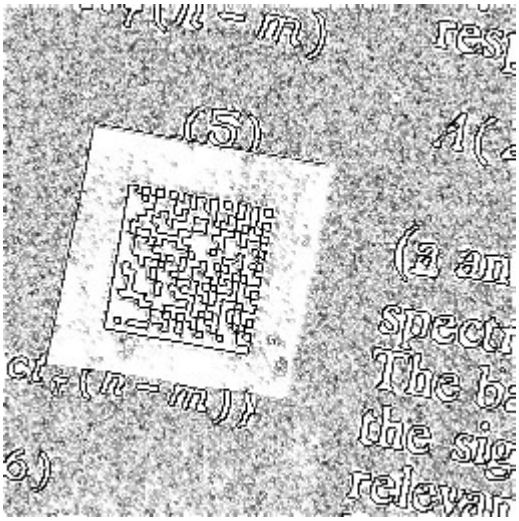


FIG. III-30 - *contours de l'image de la figure* **Erreur ! Source du renvoi introuvable.**-1

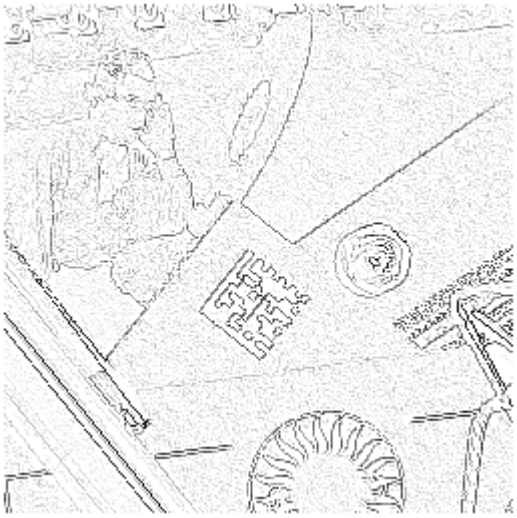


FIG. III-31 - *contours de l'image de la figure III-27*

Les critères utilisés sont la *continuité*, la *densité de contour*, et la *proximité*. Ils sont inspirés de ceux utilisés dans [CASA92] (voir § II.A.6.D), et nous utilisons le même formalisme pour les décrire. La proximité s'entend au sens du voisinage à 8, les texels sont de taille 8×8 (t_x=8). Pour chaque critère X, t_X est sa valeur du critère et K_X son coefficient de normalisation.

Pour chaque texel, nous définissons les ensembles suivants :

- C_N (resp $\overline{X}_N$) est l'ensemble des pixels des chaînes de longueur inférieure ou égale (resp. supérieure) à N.
- P_N (resp $\overline{\Pi}_N$) est l'ensemble des pixels de contour agglomérés en blocs de dimensions inférieures ou égales (resp. supérieures) à $N \times N$.
- D_N est l'ensemble des pixels de contour distants d'exactement N pixels.

D'un point de vue algorithmique, ces critères ne sont pas calculés *stricto sensus* mais sont estimés, et l'approximation faite suffit pour tirer des conclusions sur la probabilité qu'a un texel d'appartenir à un symbole.

- Continuité

$\tau_O = K_O \text{card}\left(\overline{X}_2\right)$ (quantité de pixels des chaînes de longueur supérieure à 2). C'est un critère positif : il réagit dans les zones où se trouvent des contours, et donc là où se trouve un symbole. La limitation à deux pixels minimise les détections dues au bruit.

Pour chaque texel, on compte dans la carte de contours binaires le nombre de pixels isolés et ceux présents qui n'ont qu'un seul voisin, tels que ce voisin ait lui même un seul voisin (qui est nécessairement le pixel que l'on compte). On retranche cette quantité du nombre de pixels de contours du texel. Ce calcul permet d'estimer le nombre de pixels de contour appartenant à des chaînes de longueur supérieure à 2.

- Densité de contours

$\tau_N = K_N \text{card}\left(\overline{\Pi}_1\right)$ (quantité de pixels de contour agglomérés en blocs de taille 2×2 ou plus). C'est un critère négatif : les symboles ne présentant pas de traits fins, il ne devrait pas y avoir de contours agglomérés (avec une carte de contours minces).

Pour chaque texel, on compte dans la carte de contours binaires le nombre de pixels présents ayant des voisins dans les trois directions 0, 1 et 2. Ce calcul représente moins le nombre de pixels agglomérés que le nombre de blocs : un seul pixel est compté pour un bloc isolé de taille 2×2 (les blocs sont rarement plus grand étant donné que nous utilisons des contours fins), et quelle que soit la forme du bloc, certains pixels du bord du bloc ne sont pas comptés.

- Proximité

$\tau_P = K_P \text{card}\left(\Delta_1\right)$ (quantité de pixels de rapprochés). C'est un critère négatif : les symboles présentent des contours relativement éloignés les uns des autres.

On compte dans la carte de contours binaires les pixels de non-contour ayant deux contours dans les directions 0 et 4 ou dans les directions 2 et 6. Ceux qui ont quatre contours dans les directions 0, 2, 4 et 6 comptent double.

III.B.2.Carte de détection

Les figures III-33 à III-35 montrent les cartes de continuité, densité et proximité calculées à partir de l'image III-32. Nous voyons que la carte de présence probable (figure III-36) se déduit par la suppression dans la carte de continuité des texels marqués dans les deux autres cartes, qui représentent des critères d'exclusion (densité et proximité).

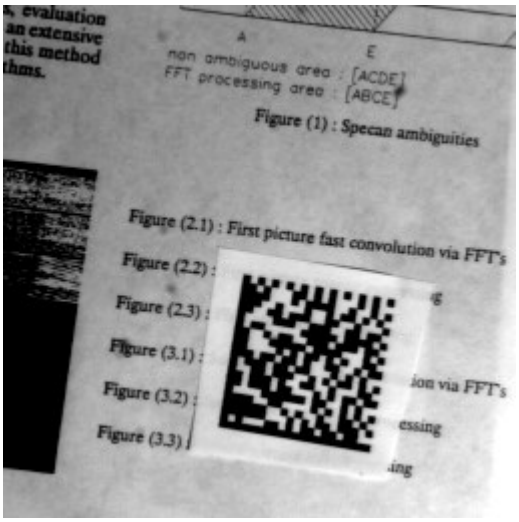


FIG. III-32 - *image comportant un symbole*

FIG. III-33 - *carte de continuité (de l'image III-32)*

FIG. III-34 - *carte de densité (de l'image III-32)*

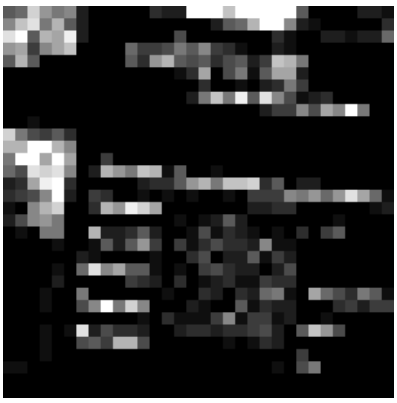


FIG. III-35 - *carte de proximité (de l'image III-32)*

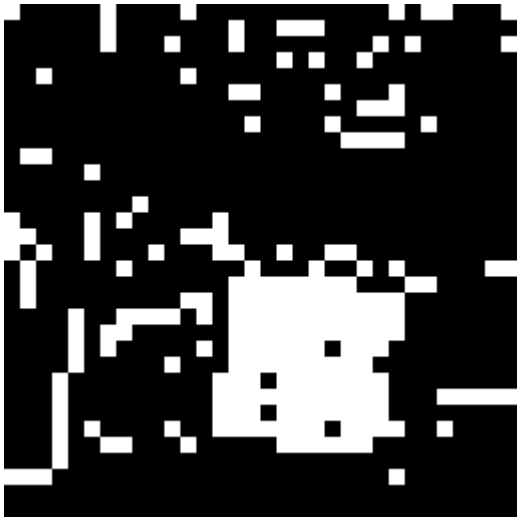


FIG. III-36 - *carte de présence probable (de l'image III-32)*

Par exemple, on voit sur ces images que le critère de continuité sélectionne la plupart des zones de l'image contenant beaucoup de contours, y compris les zones présentant un symbole et les zones de texte. Le critère de densité ne sélectionne pas les zones du symbole, mais réagit aux zones où se trouve du texte, ce qui permet de les éliminer.

Les cartes de critère sont combinées en une carte unique où chaque texel marqué correspond à une présence probable de symbole en cet endroit La figures III-38 représente la carte de présence probable de l'image III-37 (nous avons reproduit les luminances de l'image sur les texels marqués).

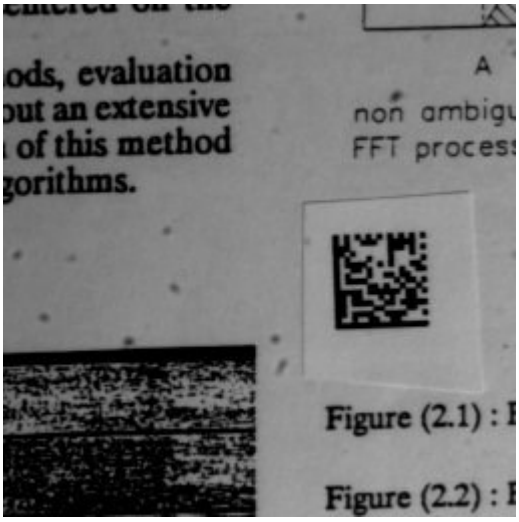


FIG. III-37 - *image d'un symbole*

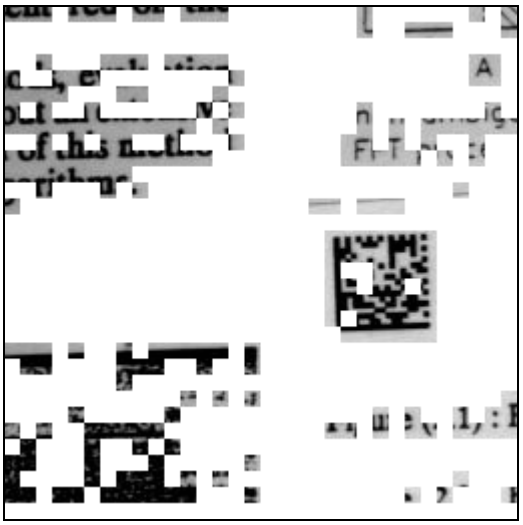


FIG. III-38 - *carte de présence probable pour l'image de la figure III-37*

Une combinaison linéaire des critères, suivie d'un seuillage, permettrait une grande souplesse dans la pondération des critères, mais le choix des coefficients serait difficile à réaliser (chaque critère n'a pas la même importance). Une méthode équivalente a été préférée qui consiste à seuiller d'abord séparément chaque critère, puis à en faire la combinaison booléenne. Un texel sera inscrit dans la carte de détection s'il est présent dans les cartes des critères positifs et absent des cartes des critères négatifs. Cette méthode permet de choisir séparément, et avant combinaison, le seuil à appliquer sur un critère en fonction des qualité et amplitude de sa réponse.

Le choix du seuil se fait selon les critères suivants :

- Pour un critère positif :
  - Préserver les régions appartenant au symbole.
  - Éliminer les régions extérieures.
- Pour un critère négatif :
  - Éliminer les régions appartenant au symbole.
  - Préserver les régions extérieures.

III.B.3.Localisation dans une image et validation d'une région

S'il y a un symbole dans l'image, les texels correspondant à cette zone dans la carte de présence probable sont en majorité marqués. Puisque la zone où se trouve un symbole est carrée, le groupe de texel marqué et correspondant à cette zone doit avoir une forme convexe proche d'un carré. Un traitement morphologique (ouverture et fermeture) y est appliqué afin d'éliminer les texels isolés.

Les texels restants sont regroupés en objets (groupes connexes) dont on mesure les caractéristiques géométriques. Les groupes dont l'excentricité est trop élevée, ou dont la surface est trop petite, sont rejetés. Les contours des groupes restants sont extraits et analysés afin de déterminer la position du carré de la zone du symbole. Cette dernière méthode consiste à déterminer le polygone contour de ces groupes et à chercher à l'ajuster à un carré par la suppression progressive de ses petits côtés.

Cette méthode a été développée dans [DÉLE96] et donne des résultats incertains (voir aussi figure III-41). C'est pourquoi, plutôt que de chercher un symbole dans toute l'image,

nous avons préféré utiliser la carte de texture pour s'assurer que telle partie de l'image contient bien un symbole.

Une région comprenant un certain nombre de texels sera validée si dans cette région, la proportion de texels présents dans la carte de présence probable est supérieure à un certain seuil. Les régions à tester étant polygonales (et pas forcément carrées), nous utilisons l'algorithme "Generic Convex Polygon Scan Conversion and Clipping" de Graphics Gems ([KIRK95], vol. 1, section II) afin d'établir une carte des pixels intérieurs au polygone soumis.

III.C.Structure participative des méthodes

Les deux méthodes de recherche appliquées séparément permettent de détecter et de localiser les symboles, chacune présentant néanmoins des faiblesses.

III.C.1.Résultats de détection de symboles par lignes et textures

La méthode par lignes sélectionne tous les couples de lignes ayant les caractéristiques du motif en *L* et détermine le coin d'horloge. Certains de ces couples peuvent ne pas être le bord d'un symbole, surtout avec les types d'images structurées dans lesquels se trouvent les symboles.

Les candidats des figures III-27, III-42, III-37 sont uniques dans l'image, mais dans la figure III-39, un autre candidat est proposé à la lecture. La méthode n'a aucun moyen de trancher en faveur de l'un ou l'autre candidat et les deux doivent être proposés à la lecture.

FIG. III-39 - *image d'un symbole avec motif perturbateur*

FIG. III-40 - *candidats retenus*

III.C.2.Méthode par analyse de texture

La méthode par texture est efficace en détection, mais peu précise en localisation. La nature de la texture (et donc la présence ou non d'un symbole) est établie sur des texels de 8 pixels de côté ; ces pixels, selon l'échelle, peuvent à leur tour représenter plusieurs pixels de l'image d'origine. Les position et localisation du symbole doivent être connues avec exactitude afin de mener à bien les étapes suivantes de lecture. Si la position du symbole est erronée, il est fort probable qu'il ne soit pas possible de déterminer la fréquence du symbole, et *a*

fortiori d'en faire un échantillonnage correct. Les symboles des figures III-27, III-42, sont convenablement localisés, mais ce n'est pas le cas de celui de la figure III-41.

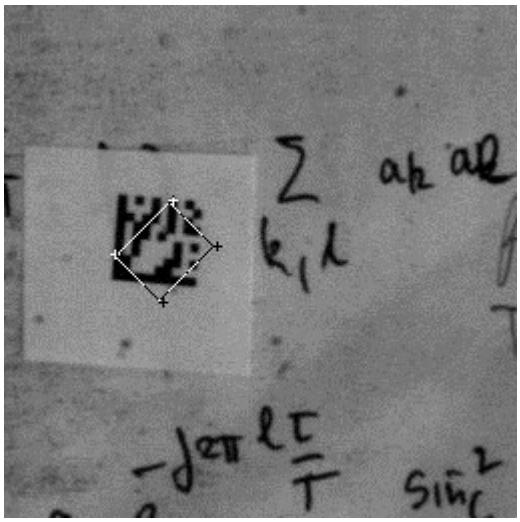


FIG. III-41 - *détection d'un symbole avec échec de la localisation*

III.C.3.Conclusion de la recherche de lignes

En dehors des objectifs recherchés pour la recherche de symboles Data Matrix, nous pouvons émettre quelques remarques sur cette méthode. La méthode fonctionne bien avec des images contenant des lignes droites, et délivre des suites de petites lignes jointives lorsque des contours courbes sont présents dans l'image. Cette méthode ne fonctionne pas bien pour la détection de petites lignes (2 à 3 pixels) qui ne peuvent pas être correctement déterminées à partir des réponses des masques de taille 2×2 ou 3×3 qui sont utilisés, puisque les erreurs de localisations inévitables de l'ordre d'un pixel entraînent des erreurs importantes dans la direction de ces lignes. Cette incompatibilité permet d'établir que l'élimination des courtes chaînes de la fin de l'étape de chaînage n'entraîne pas une dégradation des résultats de la détection, les lignes issues de ces chaînes étant de toute façon non fiables.

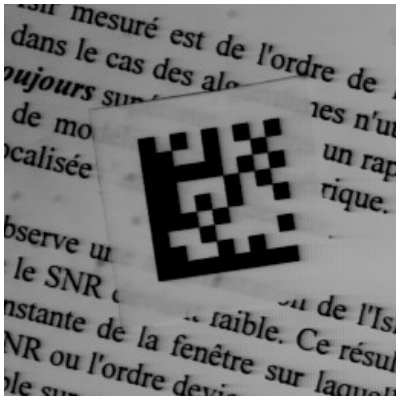


FIG. III-42 - *image d'un symbole Data-Matrix*

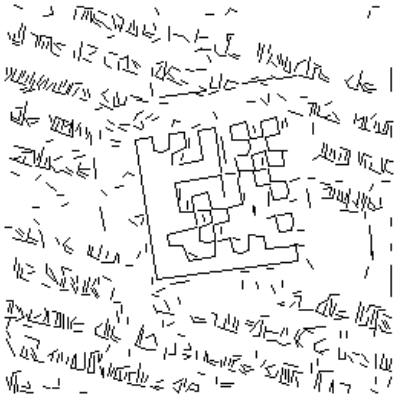


FIG. III-43 - *résultat de la détection de ligne*

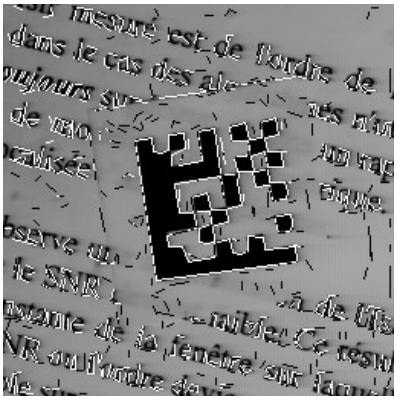


FIG. III-44 - *résultat de la détection de ligne superposé à l'image*

III.C.4.Combinaison des méthodes

Nous voyons que les défauts de chaque méthode sont assez différents. La méthode par lignes donne de bons résultats mais peut-être faussée par certains éléments de l'image. La méthode par texture réalise la détection, mais n'est pas fiable pour la localisation.

Nous avons donc conçu une méthode qui réalise la détection et la localisation, en s'appuyant sur les performances positives de chacune des approches. Elle procède de la façon suivante.

La méthode par recherche de lignes est tout d'abord appliquée. L'ensemble de lignes délivrées est analysé afin d'en tirer un certain nombre de candidats ayant les propriétés du motif de détection en *L*. Pour chacun de ces couples de lignes, nous analysons sur plusieurs échelles la texture à l'intérieur de la zone délimitée ; les couples sont rejetés si la proportion de texels marqués est trop faible dans cette zone (procédé de validation décrit § III.III.B). De cette manière, les échecs décrits à la figure III-40 ne se produisent plus. L'étape de localisation dans la méthode par texture, qui constituait sa principale faiblesse, n'est plus réalisée.

L'enchaînement des opérations est présenté figure III-45.

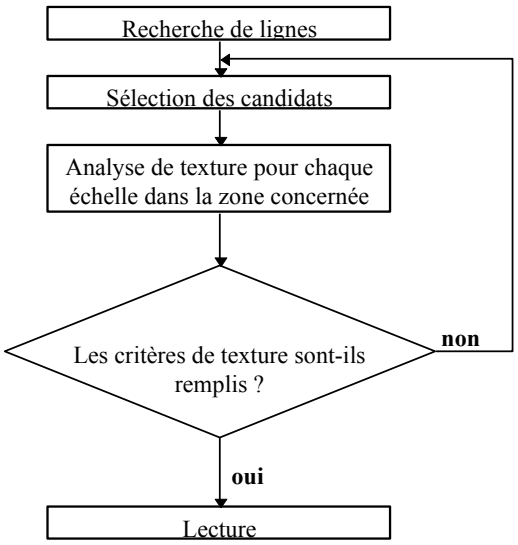


FIG. III-45 - *organigramme de la méthode générale*

L'approche initiale avait été de chercher quelle propriété intrinsèque des images de symbole pourrait donner de meilleurs résultats pour la détection. Il s'est avéré que l 'utilisation conjointe de plusieurs propriétés (motif en *L* et texture) était nécessaire pour obtenir de bons résultats de reconnaissance.

La combinaison des deux méthodes présentées permet de réussir des détections là où l'une ou l'autre méthode, prise séparément, aurait échoué. La quantité de calculs est optimisée (l'image de contours n'est calculée qu'une seule fois ; les caractéristiques de textures ne sont pas calculées pour toute l'image). La validation des détections de symboles par l'analyse de texture permet d'éviter de tenter des opérations de lecture vouées à l'échec qui ralentiraient le fonctionnement du dispositif.

Les faiblesses d'une méthode sont contournées par la participation de l'autre. Les points forts de chaque méthode sont conservés.

III.D.Lecture

Dans les sections précédentes, nous avons établi comment est déterminée la position du symbole. La suite consiste à :

- Déterminer la fréquence du symbole.
- Déterminer la valeur des cellules à partir de cette fréquence.

III.D.1.Identification de la fréquence

Afin de déterminer la fréquence du symbole, nous effectuons des échantillonnages de la luminance sur des segments de droite. Nous appelons ces échantillonnages des *profils*.

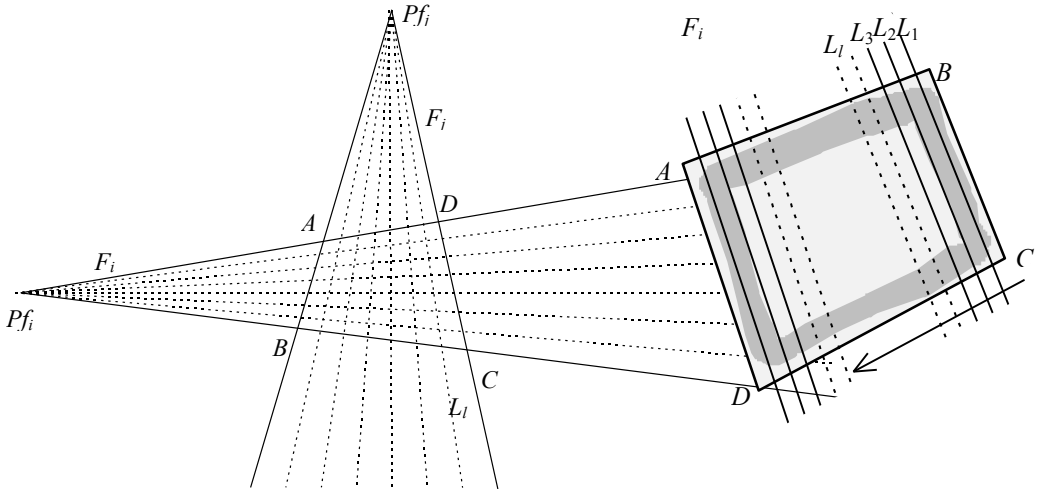


FIG. III-46 - *faisceaux construits sur ABCD*

FIG. III-47 - *recherche de la zone d'horloge*

Nous appelons faisceau de droite un ensemble de droites concourantes en un point appelé point de fuite, ou parallèles selon une direction appelée *direction de fuite*.

Au quadrilatère *ABCD*, nous pouvons associer deux faisceaux de droites : le faisceau formé par les droites (*AB*) et (*CD*) (appelé faisceau F_j) et celui formé par les droites (BC) et (AD) (F_i,). Ces faisceaux sont représentés figure III-46.

Nous savons que la zone d'horloge se trouve sur le bord du symbole, représenté en grisé sur la figure III-47. Les droites L_1, L_2, L_3, L_l. de cette figure appartiennent à F_i.

Pour déterminer la zone d'horloge située le long de [*AD*] ou [*BC*], nous utilisons les directions du faisceau F_i pour sélectionner des profils dans le quadrilatère. Ces profils seront transversaux à (*AB*) et (*CD*). La zone d'horloge pouvant être assez étroite, nous extrayons des profils proches les uns des autres, tels que les échantillons sur le plus grand des segments [*AB*] et [*CD*] soient distants d'un pixel. Sur la figure, ces échantillons sur [*CD*] sont volontairement très espacés pour une plus grande lisibilité.

Nous analysons ensuite chacun de ses profils et cherchons ceux qui ont les caractéristiques de la zone d'horloge. Ces profils sont représentés par une courbe de luminance. Les figures III-50 et III-49 montrent de telles courbes (III-50 un profil quelconque et III-49 le long de la zone d'horloge).

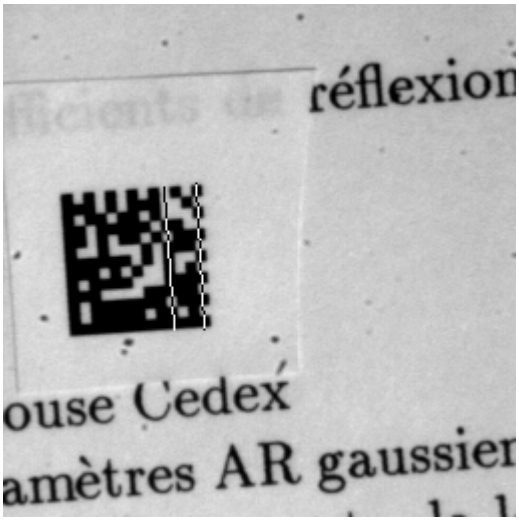


FIG. III-48 - *image de symbole d'où sont extraits deux profils (figures III-49 et III-50)*

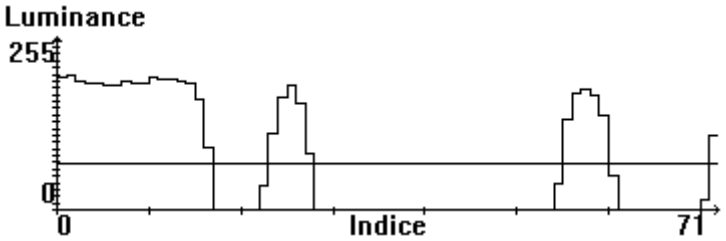


FIG. III-49 - *profil le long d'une ligne quelconque*

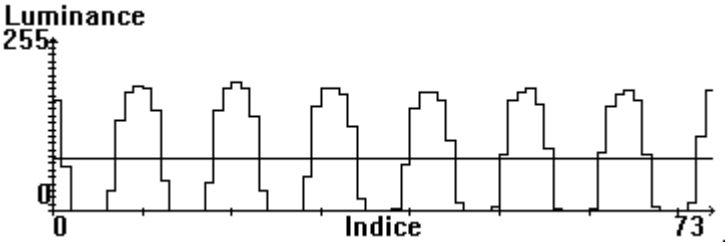


FIG. III-50 - *profil le long de la zone d'horloge*

Chaque profil est interprété comme celui d'un signal d'horloge, et est seuillée selon sa valeur moyenne (visible sur les figures), pour permettre d'estimer ses caractéristiques.

Le profil est découpé en intervalles délimités par les franchissements du seuil. Ces intervalles peuvent représenter des cellules de la zone d'horloge ou des effets de bords comme on le voit à la figure III-50.

Les longueurs de ces intervalles, à l'exception des bords, sont analysées, leurs moyenne $\bar{t}$ et écart type s sont calculés et permettent d'évaluer un critère, dit *critère d'alternance*, indiquant si le profil présente un signal proche ou non de celui d'une zone d'horloge.

Nous évaluons pour chaque intervalle s'il est de taille semblable aux autres intervalles. Pour cela, nous vérifions que, pour un intervalle de largeur t : $\bar{t} - k_{zh}\sigma \le t \le \bar{t} + k_{zh}\sigma$ ($k_{zh} \in \mathbb{N}^+$). Cette inégalité est vérifiée avec un arrondi en entier de $\bar{t} - k_{zh}\sigma$ et $\bar{t} + k_{zh}\sigma$. Par exemple, si k_{zh}=2, $\bar{t}$ =6 et s=0,4, la valeur t=7 est acceptée bien qu'étant supérieure à $\bar{t} + 2\sigma$ (=6,8).

Si les intervalles extrêmes ne satisfont pas à cette inégalité, ils sont considérés comme des effets de bord. Le nombre d'intervalles restant donne la fréquence.

Le critère d'alternance est un critère d'exclusion. Si un intervalle du profil, en dehors de ceux des bords, ne satisfait pas à cette inégalité, le critère d'alternance est fixé à $+\infty$. Sinon, il est fixé à s.

Dans la figure III-50, les intervalles des bords sont exclus, ce qui laisse 6 intervalles.

À l'issue de cette série d'échantillonnage, nous disposons, pour chaque droite du faisceau, d'un critère d'alternance et d'une fréquence. Nous cherchons dans cet ensemble le profil présentant le meilleur critère (de valeur la plus basse), et pour laquelle certaines conditions de cohérences sont remplies :

- Le profil doit être extrait près du bord (proximité qui dépend de la période spatiale) ;
- La fréquence doit être inverse de la période (la période spatiale est la largeur de la zone d'horloge). L'extremum du critère doit donc être un extremum local d'un pic ayant une valeur basse du critère sur une largeur égale à une période.

Nous montrons figures III-52 et III-53, obtenues à partir de la figure III-51, comment ces conditions sont vérifiées.

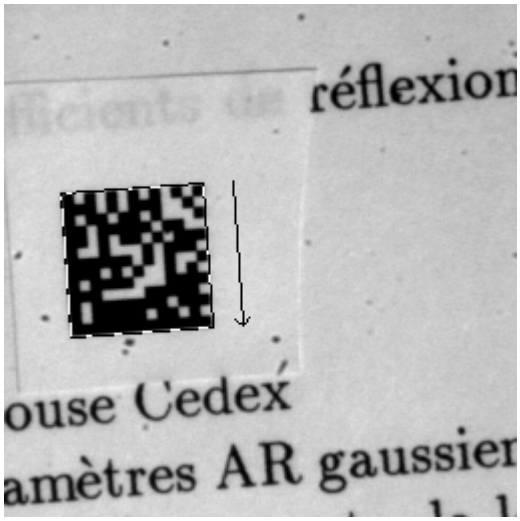


FIG. III-51 - *image de symbole d'où sont extraits les profils (figures III-52 et III-53)*

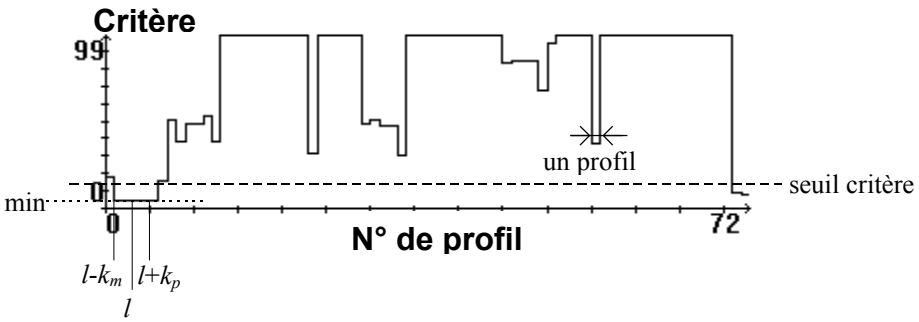


FIG. III-52 - *critères sur les profils du quadrilatère*

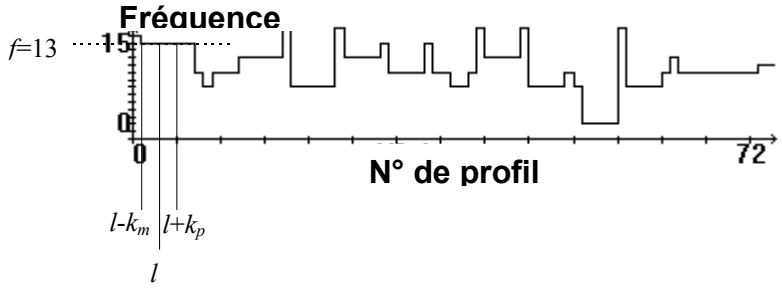


FIG. III-53 - *fréquences sur les profils du quadrilatère*

Le critère d'alternance présente un minimum absolu donnant une fréquence *f*. *l* est le numéro de profil au milieu de ce minimum. Il y a autour de ce minimum un intervalle de profils pour lesquels le critère est inférieur à un certain seuil et dont la fréquence associée est aussi *f*. Nous définissons k_m et k_p tels que cet intervalle s'étend de l-k_m à l+k_p. La largeur de cet intervalle est $t_h=k_m+k_p+1$.

La période spatiale le long de [*CD*] est CD/f. Le groupe de profils de l-k_m à l+k_p est susceptible de représenter un signal d'horloge si t_h est proche de CD/f, et si la distance entre le profil *l* et la plus proche extrémité de [*AB*] est faible par rapport à CD/f.

Dans les figure III-52 et III-53, l-k_m=1, l+k_p=5, *f*=13 et *CD*=72 (nombre de profils). Nous avons bien la largeur de la zone d'horloge t_h=5 proche de la période spatiale *CD*/*f*=5,5. Cette zone d'horloge est proche de l'extrémité de la courbe, à exactement un pixel (l-k_m).

Une fois la zone d'horloge localisée, la zone de l'image associée est à nouveau examinée afin de localiser avec exactitude le centre des cellules d'horloge. Cette localisation permet par la suite de construire la grille d'échantillonnage pour lire les cellules à l'intérieur du symbole.

Cette reconnaissance de la fréquence est effectuée dans les deux directions, selon F_i et F_j. Dans le cas des symboles carrés, les deux fréquences doivent être identiques.

III.D.2.Lecture du symbole

III.D.2.a. Calcul de la grille d'échantillonnage

Nous connaissons à ce stade la localisation du symbole, sa fréquence et les positions des cellules de la zone d'horloge.

Nous construisons les deux faisceaux F_i et F_j, constitués de *f* droites issues des points ou directions de fuite, chacune passant seule par le centre d'une cellule de la zone d'horloge. Sur les intersections de ces droites se situent théoriquement les centres des cellules de la matrice. Cet ensemble de points s'appelle la *grille d'échantillonnage* ; chacun de ces points est un *nœud* de la grille.

La figure III-54 représente les deux faisceaux. Le profil L_l qui a servi de support au faisceau F_i est représenté, le faisceau F_j a été construit de la même manière.

La figure III-55 représente les centres des cellules reconstitués pour une fréquence 15.

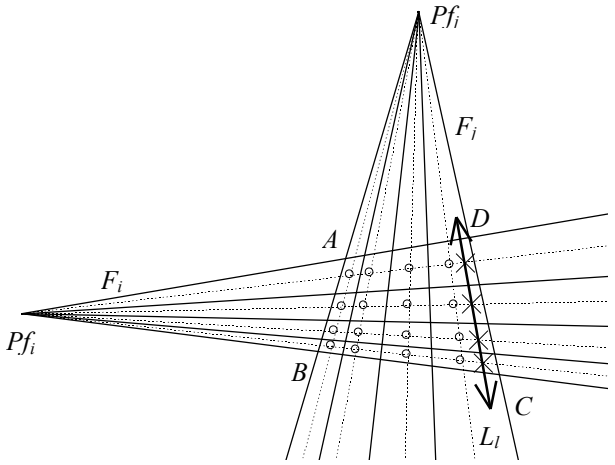


FIG. III-54 - *représentation de la grille*

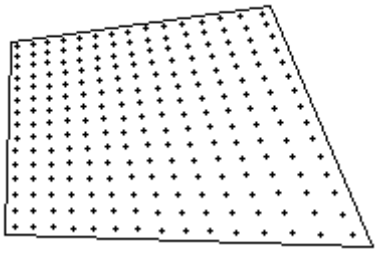


FIG. III-55 - *exemple de grille construite par la méthode à partir des quatre coins (fréquence 15)*

III.D.2.b. Lecture du symbole

Connaissant la position estimée des centres de chaque cellule, nous pourrions y lire directement la luminance pour décider de la valeur de la cellule. Cependant, l'image présente en général du bruit, et la lecture sera plus efficace si on détermine la valeur de chaque cellule en s'appuyant sur la luminance de plusieurs pixels voisins.

Autour de chaque nœud de la grille, nous définissons un *disque d'échantillonnage* contenant les pixels qui serviront à l'estimation de la valeur des cellules. Les luminances des pixels de ce disque sont lues dans l'image et leur moyenne est calculée.

La distance utilisée pour établir ce disque est la somme des valeurs absolues des distances sur x et y (distance d_1). Étant donné les faibles valeurs du rayon de ce disque, il est inutile d'utiliser la distance euclidienne classique d_2 qui nécessite des calculs de racines carrées.

Le rayon du disque r_e est calculé en fonction de la fréquence spatiale de façon à ce qu'il ne chevauche pas une cellule voisine. La période spatiale minimale dans le symbole est $t_m = \frac{1}{f}\min(AB, BC, CD, DA)$. La distance minimale entre le centre d'une cellule et une cellule voisine vaut $D_w = t_m/2$. Nous choisissons r_e légèrement inférieur à D_w afin d'être certain ne pas lire des pixels appartenant à des cellules voisines. Nous définissons $k_r < 1$ et nous fixons $r_e = k_r D_w$.

À l'issue de l'échantillonnage, nous avons une estimation de la luminance moyenne de chaque cellule. Nous faisons la moyenne de ces valeurs et nous l'utilisons comme seuil de luminance afin de décider pour chaque cellule si elle est marquée ou non. Ceci est justifié par le fait que les quantités de cellules marquées et non marquées sont globalement équilibrées.

La figure III-56 montre des résultats de lecture sur deux images de symbole (la grille d'échantillonnage a été superposée aux images).

FIG. III-56 - *Lecture de symboles : grilles et résultats (symboles redressés)*

III.E.Conclusion

La méthode adoptée pour la détection et la localisation se base à la fois sur une recherche de lignes et une analyse de texture.

La recherche de lignes utilise le passage à zéro du laplacien pour détecter et localiser les pixels de contours, et les masques de Roberts pour déterminer leurs directions. Les pixels sont chaînés puis assemblés en lignes.

Les lignes ainsi obtenues présentent des défauts (fragmentation, coins disloqués…). Afin de pouvoir représenter la scène de manière cohérente, une étape de restauration est nécessaire. Cette étape analyse une à une les différentes lignes obtenues et les ajuste, en fonction de leurs distances et de leurs orientations.

Les méthodes à base de détection de lignes obtiennent une bonne précision de la localisation, mais peuvent détecter des motifs en L ne correspondant pas à un symbole. Les méthodes à base de texture ont les propriétés inverses : elles se trompent peu, mais sont moins précises en localisation.

La méthode retenue combine les deux approches. À partir de l'ensemble des lignes, le motif en *L* caractéristique du symbole est recherché. À l'intérieur de la zone qu'il délimite, la texture est étudiée afin de déterminer s'il s'agit d'un symbole ou non.

Cette méthode permet d'exploiter la bonne précision du détecteur de ligne et la bonne discrimination du détecteur par analyse de texture.

La lecture nécessite d'identifier la zone d'horloge et de déterminer la fréquence du symbole. La localisation de la zone d'horloge permet de construire la grille d'échantillonnage.

Cette méthode ayant été programmée et mise au point, nous allons étudier son efficacité, en particulier les résultats obtenus sur un ensemble d'images, les limites, la complexité et les durées de traitement correspondant.

Chapitre IV

Mise en œuvre et résultats

IV.A.Mise en œuvre

Pour mettre en œuvre et valider les méthodes décrites précédemment, nous avons réalisé des programmes de traitement d'images qui ont été implémentés sur un dispositif expérimental. Celui-ci est constitué d'un système d'acquisition d'images et d'un système de traitement de données (un micro-ordinateur et les programmes de traitement des images).

IV.A.1. Système d'acquisition

Le banc de numérisation et de traitement d'images comprend deux cartes reliées à un micro-ordinateur de type PC par l'intermédiaire d'une carte d'entrées-sorties comprenant deux adaptateurs parallèles (PIA).

L'ensemble de ses éléments est constitué de :

- une carte d'acquisition d'images N/B ou couleur ;
- une carte de visualisation d'images ;
- une caméra achrome ou trichrome reliée à la carte d'acquisition ;
- un moniteur de visualisation relié à la carte de visualisation.

La configuration matérielle de l'ensemble du dispositif expérimental est présentée à la figure IV-1.

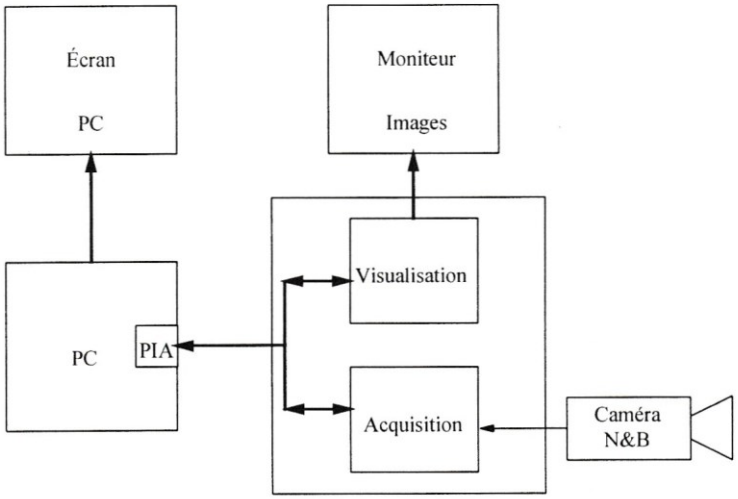


FIG. IV-1 - *configuration matérielle du dispositif expérimental : système d'acquisition et micro-ordinateur*

Pour cette application, nous n'avons effectué que des acquisitions en noir et blanc. Le format des images numérisées est de 256×256 ; chaque pixel est codé sur 8 bits.

IV.A.2. Système de traitement

Le micro-ordinateur utilisé est de type PC, muni d'un processeur Pentium 100 MHz.

Le logiciel de traitement d'image est écrit en C++, avec OWL de Borland pour l'environnement M$-Windows 3.1*x*. Il est constitué d'une base logicielle développée au laboratoire (le *noyau*) et d'une couche dédiée à l'application de recherche et de lecture de symboles (la *couche applicative*).

Le noyau est destiné au développement d'applications en traitement d'images. Il est constitué d'une couche conversationnelle qui gère les relations avec l'utilisateur et qui communique avec le système d'exploitation et d'une couche calculatoire, constituée par une bibliothèque de fonctions appelées par la couche conversationnelle.

Ces fonctions permettent de réaliser des opérations de traitement d'image :

- Mesures (histogrammes, statistiques, profils…)
- Traitements simples (seuillage, transformations géométriques…)
- Traitements complexes (filtrages, modifications d'histogramme, caractérisation d'objets…)

La couche applicative a été rédigée pour l'étude. Nous avons appelé le logiciel ainsi constitué *Iris*. Il est composé lui aussi, selon le même principe, d'une partie conversationnelle (le *superviseur*, qui communique aussi avec la couche conversationnelle du noyau) et d'une partie calculatoire. La figure IV-2 montre les différentes couches, leurs composantes et leurs interconnexions.

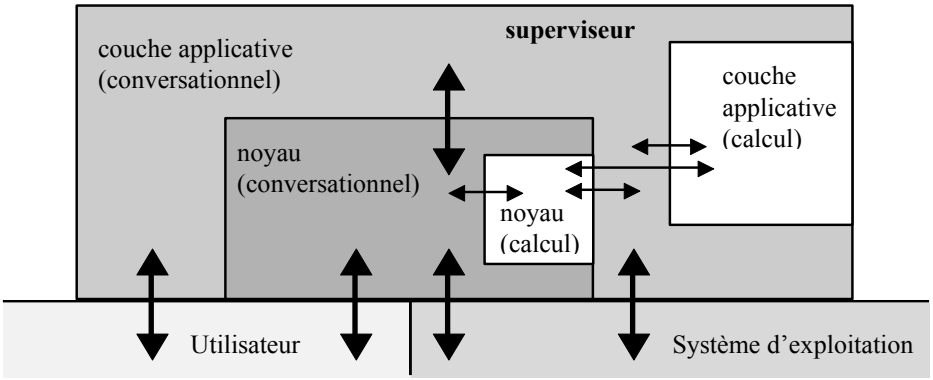


FIG. IV-2 - structure du logiciel iris

Les fonctions du noyau ou de la couche applicative sont accessibles à partir de menus déroulants, regroupées selon des rubriques ; certaines sont également activables par des boutons. Ces fonctions sont appelées par les couches conversationnelles.

IV.A.3. Le logiciel Iris

Ce logiciel permet, sous certaines conditions de bonne présentation que nous préciserons plus loin, de rechercher et de lire des symboles Data Matrix.

IV.A.3.a. Structure générale

Les fonctionnalités d'Iris sont découpées en plusieurs sous-applications distinctes, codées dans des modules différents et matérialisées par des fenêtres, qui communiquent entre elles. Elles sont gérées par le superviseur.

Les principales sous-applications du logiciel Iris sont les suivantes :

- Le détecteur de lignes
- L'analyseur de texture
- Le lecteur de symbole

Chaque sous-application, ainsi que le superviseur, est munie d'un certain nombre de fonctions de traitement, de paramètres réglables, et d'une interface utilisateur permettant de modifier les paramètres et d'appeler les fonctions.

La séquence d'opérations de recherche puis de lecture peut être exécutée soit manuellement en pas à pas, soit de manière automatique. Les paramètres utilisés par le superviseur ou par les sous-applications sont réglables par l'utilisateur à chaque étape et peuvent être mémorisés. Le fonctionnement automatique utilise les paramètres que l'utilisateur a mémorisés, et procède selon la méthode détaillée au chapitre III.

Le superviseur (voir figure IV-3) dispose des fonctions permettant de calculer divers types de contours (gradient, laplacien, passages à zéro du laplacien…) dont les paramètres sont réglés par l'utilisateur (type de filtres, type de voisinage, seuils…). Ces fonctions permettent d'étudier les caractéristiques des différents types de contour afin de choisir ceux qui sont les mieux adaptés.

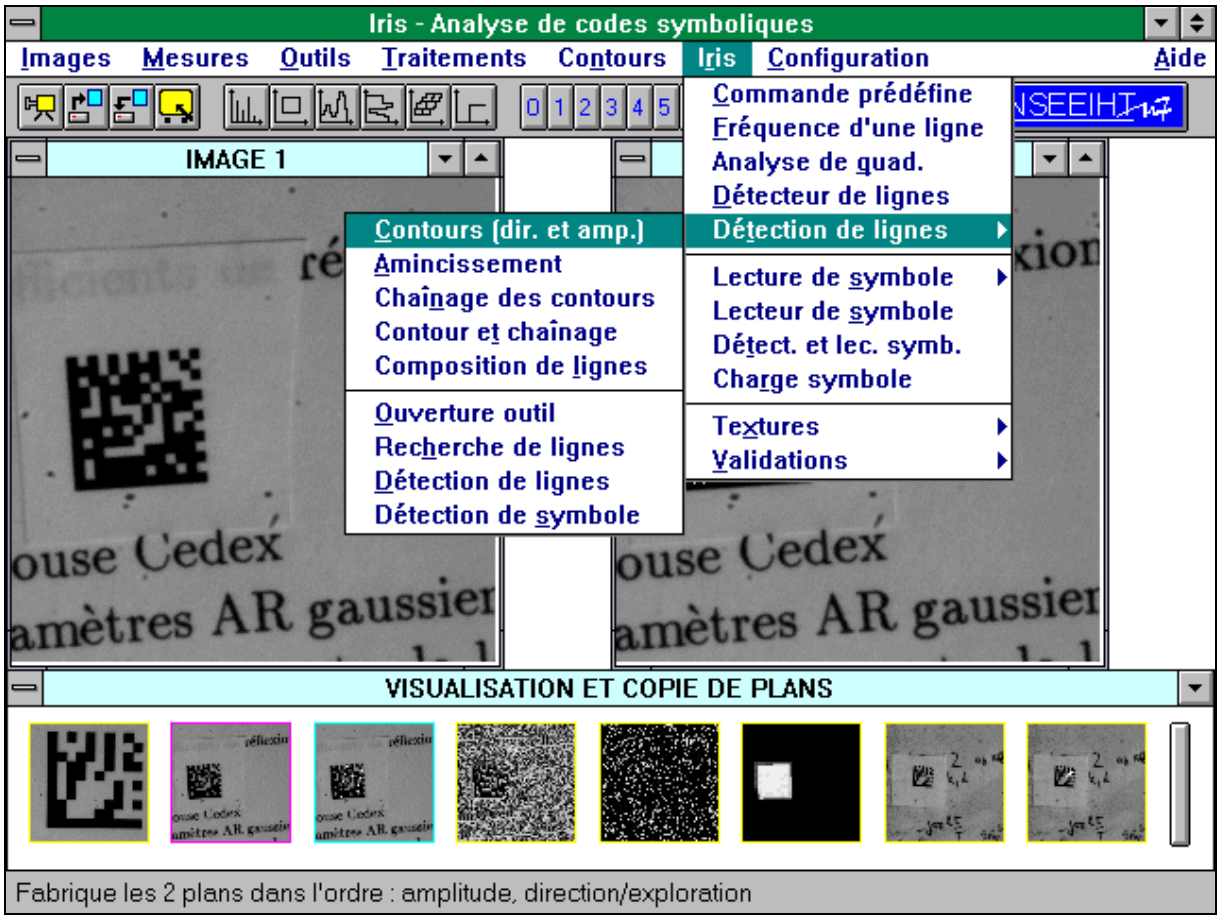


FIG. IV-3 - *aperçu du programme superviseur (menus et plans image visible)*

IV.A.3.b. Détecteur de lignes

Le détecteur de ligne (voir figure IV-4) appelle les fonctions de contour de la couche calculatoire, et utilise les résultats obtenus pour construire une liste de lignes. Les résultats intermédiaires (carte de directions, chaînes de pixels de contours, table de lignes avant restauration) sont accessibles et le détecteur de ligne propose des outils pour les étudier et les archiver.

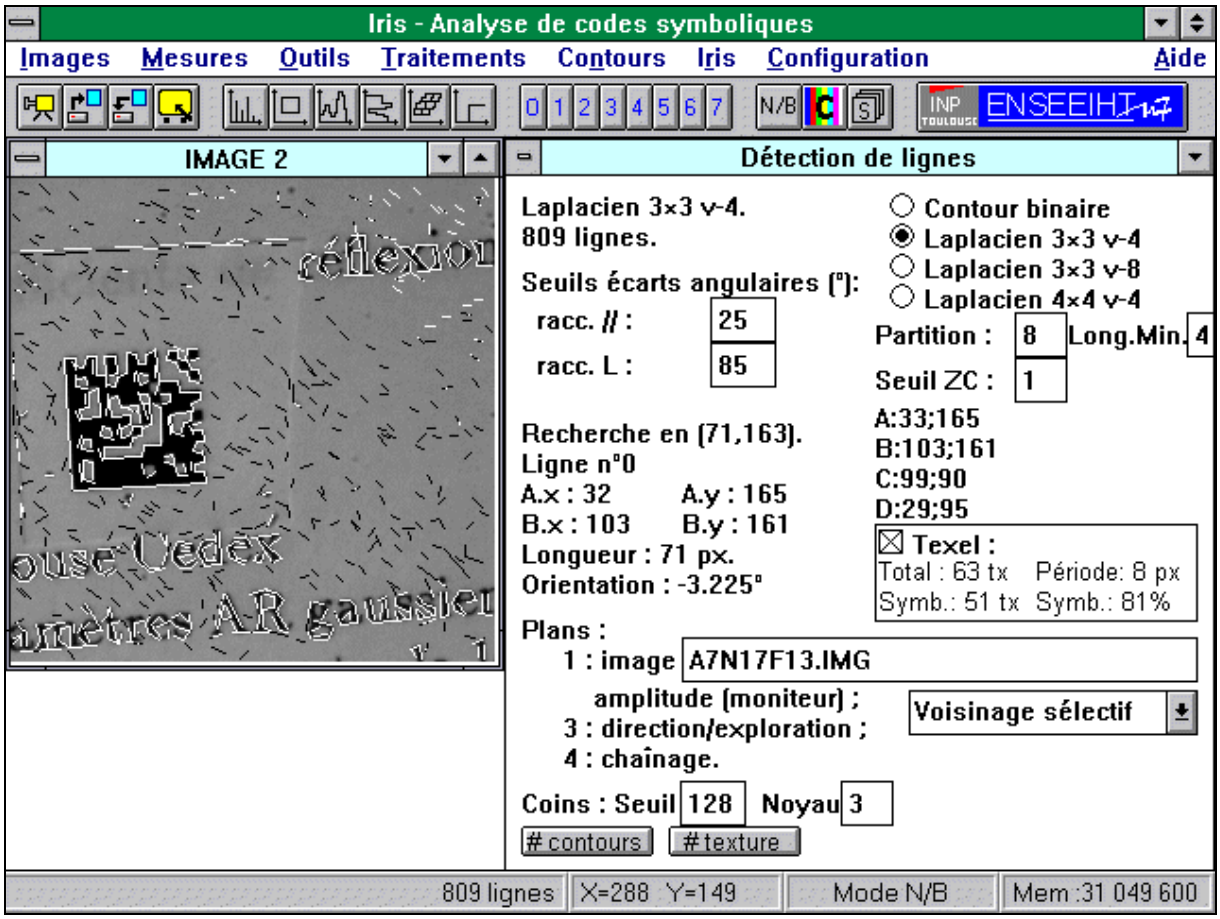


FIG. IV-4 - *aperçu du détecteur de ligne*

Les lignes sont analysées, en commençant par les plus longues. Les positions des couples de lignes susceptibles d'être le motif en *L* d'un symbole sont soumises à l'analyseur de texture qui détermine si la texture contenue dans la zone concernée peut-être celle d'un symbole. L'analyseur de texture peut-être désactivé (il n'est pas exécuté lors de la chaîne de traitement automatique et tous les couples de lignes candidats sont validés). Ses résultats intermédiaires sont aussi accessibles.

Le premier couple de lignes répondant aux critères de texture est passé en paramètre au lecteur de symbole.

IV.A.3.c. Analyseur de texture

Le détecteur de ligne appelle l'analyseur de texture en lui passant en paramètre un quadrilatère représentant la localisation possible d'un symbole. Ce dernier analyse la texture dans le quadrilatère désignée en utilisant plusieurs critères appliqués à différentes échelles de fréquence. S'il s'avère que la texture est compatible avec une texture de symbole, alors cette information est transmise au détecteur de ligne qui a son tour transmet la position supposée du symbole au lecteur de symbole.

Des fonctions de détection et de localisation de symbole à partir des cartes de texture ont été écrites dans la couche calculatoire et sont accessibles à l'utilisateur. Ces fonctions cependant ne font plus partie de la chaîne des opérations de détection de symbole.

IV.A.3.d. Lecteur de symbole

Le lecteur de symbole (voir figure IV-5) reçoit une position supposée d'un symbole, et analyse l'image en cet endroit pour déterminer la fréquence propre de l'éventuel symbole. Si une fréquence cohérente est trouvée en bordure (selon un certain nombre de critères), ce profil est mémorisé et sert à l'établissement de la grille d'échantillonnage. Le symbole est lu selon cette grille. Une fois le symbole lu, son contenu est affiché à la fois sous forme matricielle, sous forme de texte ou de suite de valeurs numériques.

Afin de permettre de réaliser des tests de manière automatique, nous avons construit une bibliothèque d'images pour lesquelles nous connaissons le contenu du symbole. À partir du nom du fichier de l'image utilisée, ou d'une liste d'associations, la valeur exacte du symbole est retrouvée, et comparée avec la valeur déterminée par le lecteur de symbole.

Le lecteur de symbole propose des outils pour manipuler les symboles : fabrication, écriture, visualisation, modification et transfert dans des fichiers.

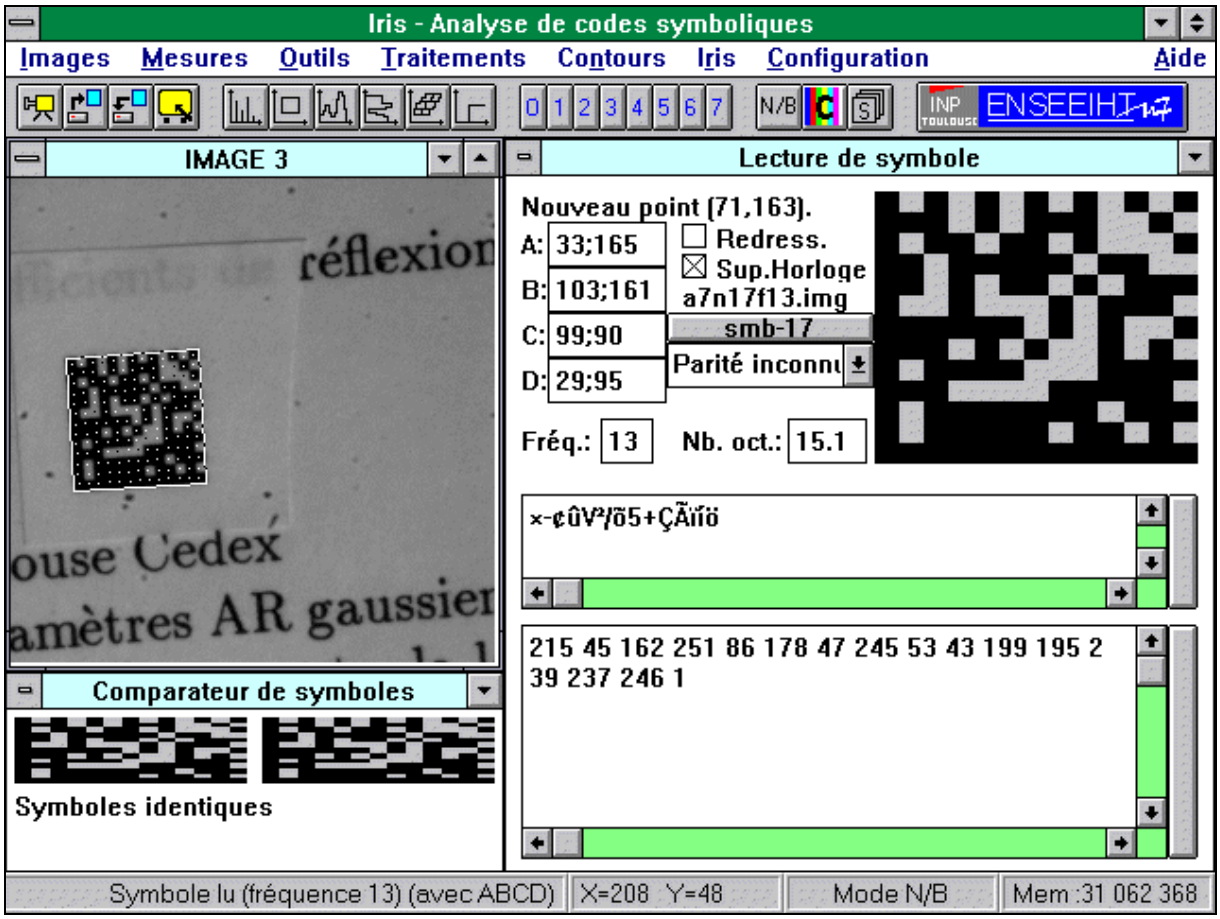


FIG. IV-5 - *aperçu du lecteur de symbole (comparaison avec la valeur théorique à gauche)*

IV.B.Résultats expérimentaux

La caméra vidéo utilisée est une caméra développée autour d'un module type 56474 CCIR à balayage entrelacé doté d'un capteur CCD NXA 1011/03 de taille H×V 604×588 pixels. Nous avons utilisé trois objectifs, de focale 10, 25 et 50 millimètres. Les symboles, imprimés par une imprimante LaserJet 4 de Hewlett Packard, ont une taille variant environ de 5 à 50 millimètres, pour des fréquences impaires de 5 à 81. La distance entre la caméra et les symboles varie de 10 à 80 centimètres.

Ceci nous a permis d'obtenir dans les images des fourchettes de taille et de fréquences de respectivement 5 à 256 pixels et 0,1 à 50 pixels.

Nous avons présenté au chapitre III la méthode complète de recherche et lecture. Nous appelons *le processus* l'ensemble des calculs effectués pour la recherche et la lecture de symbole, selon cette méthode.

IV.B.1. Valeurs des paramètres de la méthode

La méthode utilise un certain nombre de paramètres que nous avons présenté au chapitre III. Nous les rappelons ici en indiquant les effets de leur valeur sur le processus. Des essais avec divers types d'image et de symbole nous ont amené à choisir les valeurs que nous présentons.

IV.B.1.a. Seuil de rectitude

Ce seuil détermine comment une chaîne doit être fragmentée en lignes lors de l'étape de composition de lignes. Il correspond à la flèche maximale autorisée dans une chaîne. Les images ne présentent pas de distorsions optiques susceptibles de courber les symboles et eux-mêmes sont disposés sur des supports plans.

Nous avons choisi une valeur de 1 pixel.

IV.B.1.b. Seuils de restauration

Ces seuils déterminent d'une part les distances limites entre deux extrémités de lignes à joindre (S_R), et d'autre part les angles limites entre deux lignes pour permettre une jonction (S_{R0}) ou une unification (S_{R9}). Si ces trois valeurs sont trop proches des valeurs nominales des angles (distance 0 et angles 0° et 90°), des lignes risquent de n'être pas assemblées correctement et d'empêcher la détection. Une erreur d'un pixel sur une ou deux extrémités d'une petite ligne modifie beaucoup sa direction. Le choix des seuils angulaires deviennent déterminant lorsque le niveau de bruit augmente, et que l'erreur de localisation des contours augmente.

Nous avons choisi les valeurs S_R=5 px, S_{R0}=25° et S_{R9}=65°. Ces valeurs d'angles correspondent à un écart de 25° à la valeur nominale, c'est à dire à une erreur de localisation de 1 pixel sur chaque extrémité d'une ligne de 5 pixels de longueur.

IV.B.1.c. Voisinage de la recherche de coin

Lorsque nous cherchons un coin d'horloge au voisinage du coin d'horloge nominal, le rayon du voisinage est un paramètre. Ce paramètre est difficile à fixer, car il devrait dépendre de la période spatiale. Nous avons fixé ce paramètre à 3 pixels, ce qui favorise la localisation

des symboles de haute fréquence spatiale au dépends de celle des symboles de basse fréquence spatiale.

IV.B.1.d. Coefficients et seuils de texture

Les coefficients de texture sont les coefficients qui permettent de normaliser les critères de texture. Les valeurs de critère constituent une carte que l'on seuille avant de la combiner de manière booléenne avec les autres cartes. Les choix des coefficients et des seuils se font de manière conjointe.

Soit un critère $t_X(p)=K_X F(p)$, où $F(p)$ est une fonction scalaire de l'image calculée sur le texel de centre p (p est un pixel). Le seuil S_X est utilisé pour déterminer la carte de texture de ce critère. Soit α un réel quelconque strictement positif, les texels marqués dans la carte sont ceux pour lesquels :

$$t_X(p)>S_X \Leftrightarrow K_X F(p)>S_X \Leftrightarrow \alpha K_X F(p)>\alpha S_X \quad \text{(IV-1)}$$

$$t_X(p)>S_X \Leftrightarrow F(p)>S_X / K_X \quad \text{(IV-2)}$$

Nous obtenons donc la même carte en utilisant αK_X et αS_X à la place de K_X et S_X. Si $\alpha=1/K_X$, cela revient à seuiller directement $F(p)$ par S_X / K_X .

Cependant, la valeur mémorisée dans la carte est la valeur $K_X F(p)$. Dans notre implémentation, la carte de texture est numérique et a la même dynamique que les images (256 niveaux). Il conviendra de choisir un coefficient K_X tel que le critère utilise au mieux cette dynamique. Une fois K_X fixé, il faut déterminer les valeurs de seuil à adopter pour obtenir les meilleurs résultats

Nous avons déterminé les valeurs des seuils de texture par une méthode heuristique, en faisant des essais sur divers types d'images, avec des symboles de fréquence et taille différentes, et des environnements différents. En même temps que ces valeurs et de la même manière, nous avons déterminé le seuil du détecteur de contour (laplacien) qui génère les cartes utilisées pour le calcul des critères. De ce seuil dépend la quantité de contours présents dans la carte, et donc la valeur des critères.

- Continuité

K_O=15, S_O=70, S_O/K_O=4,67 (quantité par texel de pixel appartenant à des chaînes)

- Densité (critère d'exclusion)

K_N=40, S_N=140, S_N/K_N=3,5 (quantité par texel de contours agglomérés)

- Proximité (critère d'exclusion)

K_P=15, S_P=140, S_P/K_P=9,33 (quantité par texel de contours rapprochés)

- Seuil du passage à zéro pour la détection de contours : 60.

Il n'est pas nécessaire de détecter tous les texels de texture, mais une certaine proportion qui est la proportion jugée acceptable lors de l'étape de validation (le seuil de validation de texture). Il n'est pas nécessaire non plus d'empêcher complètement les fausses alarmes, à condition que leur proportion dans une zone donnée reste en deçà du seuil de validation de texture.

IV.B.1.e. Seuil de validation de texture

Les zones délimitées par les couples de lignes formant un motif en *L* sont analysées en vue d'évaluer les critères de texture à partir desquels une carte de présence probable est déterminée.

Ces couples sont rejetés si la proportion de texels marqués est inférieure au seuil de validation de texture (procédé de validation décrit § III.B.3).

Tout comme pour la détermination des coefficients et seuils de texture, nous avons procédé de manière heuristique, à partir de diverses images. Le seuil adopté est de 0,8 (80 %). Les zones dans lesquelles moins de 80 % des texels ont une texture de symbole (selon les critères de symboles adoptés) sont rejetées, les autres sont acceptées.

IV.B.1.f. Coefficient de régularité du signal d'horloge

Lors de l'identification de la fréquence, un profil est accepté comme signal d'horloge si les largeurs des cellules supposées sont toutes telles que $\bar{t} - k_{zh}\sigma \le t \le \bar{t} + k_{zh}\sigma$, où $\bar{t}$ est leur largeur moyenne et σ leur écart type.

Avec k_{zh}=1, il est certain que presque la moitié (statistiquement) des largeurs sera en dehors de cet intervalle, même si elles sont très régulières.

Nous avons choisi k_{zh}=2.

IV.B.1.g. Rayon du disque d'échantillonnage

Le paramètre k_r détermine le rayon r_e du disque d'échantillonnage utilisé pour évaluer la valeur des cellules, par rapport à D_w, la distance minimale entre le centre d'une cellule et une cellule voisine. $r_e=k_r D_w$, le rayon est arrondi à l'entier inférieur.

Ce paramètre est surtout significatif lorsque la taille des cellules est faible, car dans ce cas, le risque de chevaucher les cellules voisines est plus grand, et le nombre des pixels sur lesquels repose l'estimation de la valeur d'une cellule est réduit.

Nous avons obtenu de bons résultats en prenant $k_r=⅔$. Lorsque la période minimale t_m est inférieure à 6, D_w est inférieur à 3, et r_e est inférieur à 2. Nous prenons dans ce cas r_e=1.

IV.B.2. Exemples illustratifs

Lorsqu'une recherche et lecture de symbole est réalisée, trois types de résultats peuvent survenir :

1. Le processus échoue et s'interrompt,
2. Le processus délivre une matrice résultat ne correspondant pas au symbole,
3. Le processus délivre une matrice résultat correspondant au symbole, avec éventuellement des erreurs de lecture.

En général, à cause de l'identification de la fréquence qui exige une cohérence des deux zones d'horloge, le cas 2 arrive rarement.

Il peut arriver cependant, dans le cas 2, que la fréquence soit mal identifiée, ce qui rend la matrice résultat complètement fausse. Si le processus faisait partie d'une chaîne complète de lecture avec décodage du message du symbole, ce genre de résultat serait rejeté dans une étape ultérieure grâce à la détection d'erreur utilisée dans le code Data Matrix.

Si la localisation et l'identification de la fréquence sont bonnes, la matrice est en général correctement lue, mais peut cependant présenter quelques erreurs sur certaines cellules, à cause notamment de petites erreurs de localisation, de faible contraste ou de bruit.

Nous mesurons la validité d'un résultat en comparant la matrice lue avec la matrice originale. f est la fréquence propre du symbole, $(f\text{-}2)^2$ est le nombre de cellules de données. Soit E le nombre de cellules de données ayant des valeurs erronées. La qualité d'une lecture se mesure à l'erreur relative, c'est à dire au nombre de cellules erronées rapporté au nombre de cellules total, parfois exprimé en pourcentage : $E/(f-2)^2$.

Nous présentons figure IV-6 plusieurs images contenant un symbole soit isolé, soit entouré d'autres éléments. Les contraste, luminance moyenne, tailles et fréquences varient. Tous ces symboles sont convenablement détectés, localisés et lus par le processus.

FIG. IV-6A - *symbole S_{12}, période t=6,4 px*

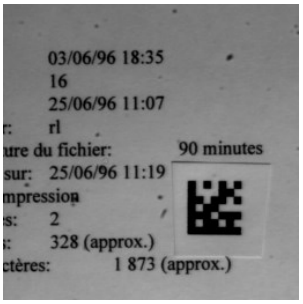


FIG. IV-6B - *symbole S_{12}, période t=6,7 px*

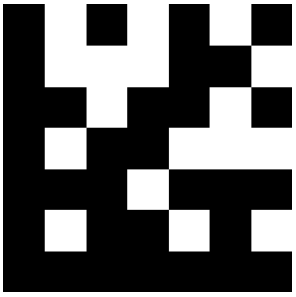


FIG. IV-6C - *symbole S_{12}, fréquence f=7*

FIG. IV-6D - *symbole Spiff, période t=3,9 px*

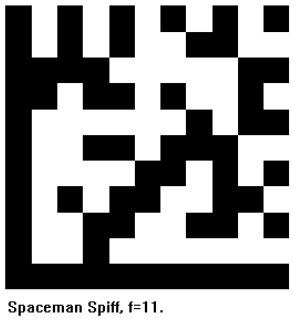


FIG. IV-6E - *symbole Spiff, fréquence f=11*

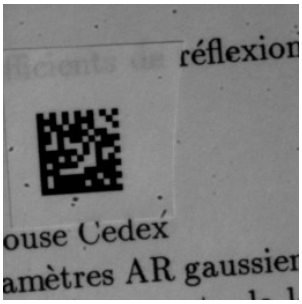


FIG. IV-6F - *symbole S_{17}, période t=5,4 px*

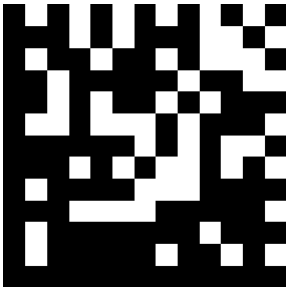


FIG. IV-6G - *symbole S_{17}, fréquence f=13*

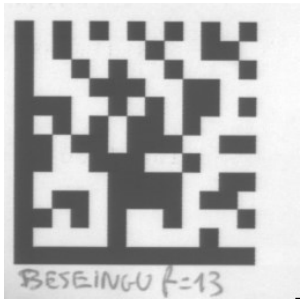


FIG. IV-6H - *symbole Beseingu, période t=16,1 px*

FIG. IV-6I - *symbole Beseingu, période t=5,0 px*

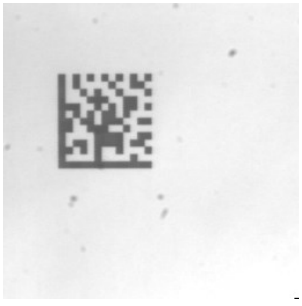


FIG. IV-6J - *symbole Beseingu, période t=6,2 px*

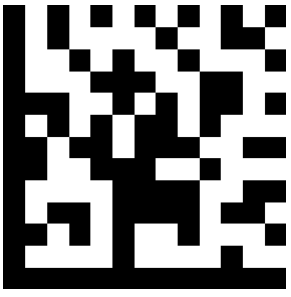


FIG. IV-6K - *symbole Beseingu, fréquence f=13*

FIG. IV-6L - *symbole S_{20}, période t=4,0 px*

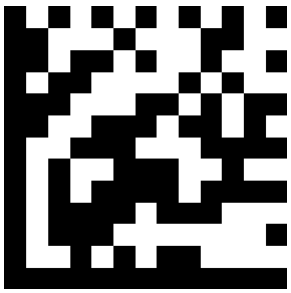


FIG. IV-6M - *symbole S_{20}, fréquence f=13*

FIG. IV-6N - *symbole S_7, période t=5,5 px*

FIG. IV-6O - *symbole S_7, fréquence f=17*

FIG. IV-6P - *symbole S_{11}, période t=6,2 px*

FIG. IV-6Q - *symbole S_{11}, période t=3,0 px*

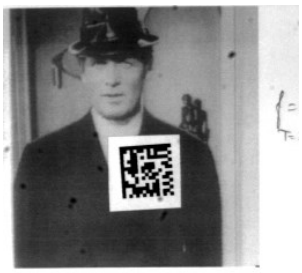


FIG. IV-6R - *symbole S_{11}, période t=3,0 px*

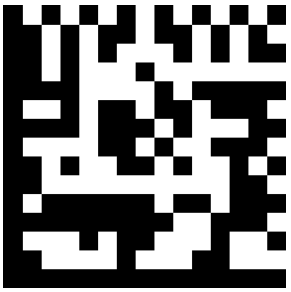


FIG. IV-6S - *symbole S_{11}, fréquence f=15*

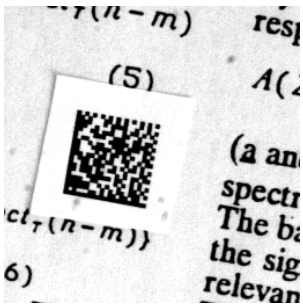


FIG. IV-6T - *symbole S_3, période t=3,5 px*

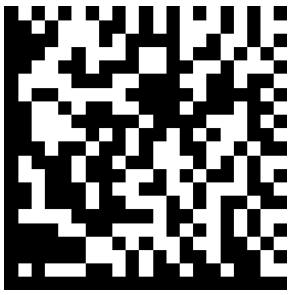


FIG. IV-6U - *symbole S_3, fréquence f=21.(ce symbole a été lu avec une erreur sur 9 cellules, soit 2,5 %)*

FIG. IV-6V - *symbole Verne43, période t=2,9 px*

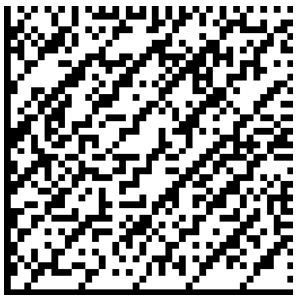


FIG. IV-IV-6W - *symbole Verne43, fréquence f=43*

FIG. IV-6 - *diverses images de symboles trouvés et lus et les symboles associés*

IV.B.3. Étapes critiques de la méthode

Nous avons vu au chapitre III comment les étapes s'enchaînent et dépendent les unes les autres. Les résultats d'une étape servent en général de base aux calculs de l'étape suivante.

Certains résultats intermédiaires sont plus déterminants que d'autres. Par exemple, si un certain nombre de pixels de contours ne sont pas détectés, les lignes peuvent tout de même être construites. Mais si les lignes du motif en *L* ne sont pas détectées, alors le symbole ne peut pas être détecté ni lu.

Nous avons décomposé la méthode en un certain nombre d'étapes dites *critiques* à l'issue desquelles le résultat, s'il n'est pas correct, entraîne l'échec du processus, soit en l'interrompant, soit en propageant un résultat faux.

Lorsqu'il y a un échec du processus, il pourra être attribué à l'une de ces étapes, ce qui permettra une meilleure compréhension des limites de la méthode.

Étapes critiques	Élément(s) calculé(s) par cette étape
Recherches de lignes	Ensemble de lignes dont deux forment le motif en *L* d'un symbole (parmi les candidats)
Validation de texture	Validation d'un candidat
Recherche de coin	Localisation précise du quadrilatère
Identification de la fréquence	Fréquence du symbole
Lecture	Grille d'échantillonnage puis valeur des cellules

TABLE IV-1 - *décomposition de la méthode par la dépendance des étapes critiques*

Nous passons en revue les étapes de la table IV-1 pour mettre en évidence leurs dépendances et les raisons possibles d'échec. Nous nous intéressons pour l'instant aux points de vue algorithmiques qui identifient l'endroit du processus où l'erreur apparaît, sans évoquer les raisons pour lesquelles ces erreurs apparaissent. Ces raisons sont liées aux propriétés de l'image et sont développées section IV.IV.B.IV.

IV.B.3.a. Recherche de lignes

Si trop de pixels de contours n'ont pas été détectés, les lignes du motif en *L* sont fractionnées même après l'étape de restauration. Le motif en *L* ne peut alors pas être détecté.

S'il y a trop de fausses alarmes dans la carte de contours, alors de courtes lignes parasites peuvent apparaître à proximité du motif en *L*, et s'assembler indûment avec ses deux lignes principales. Les deux lignes peuvent alors n'avoir plus les propriétés requises pour être détectées, ou bien être détectées mais avec une localisation erronée.

IV.B.3.b. Validation de texture

Deux cas d'erreur sont possibles : la localisation proposée ne correspond pas à un symbole mais est acceptée, ou bien c'est celle d'un vrai symbole mais elle est rejetée. Les deux cas empêchent le symbole d'être détecté.

IV.B.3.c. Recherche de coin

Si le coin d'horloge du symbole n'est pas détecté, ou si le coin détecté est un autre point, en particulier un des trois autres coins de la cellule marquant le coin d'horloge, la localisation du symbole présente une erreur de localisation.

IV.B.3.d. Identification de la fréquence

Le processus comprend l'identification de la fréquence dans les deux directions, les résultats trouvés devant être identiques. Il est donc fort peu probable, en cas d'erreur, d'obtenir une fréquence fausse.

Une erreur de localisation peut entraîner l'échec de l'identification de fréquence, dès que l'erreur de localisation sur les positions de la zone d'horloge n'est pas négligeable par rapport à la période spatiale.

Si la fréquence n'est pas convenablement identifiée, le symbole ne peut pas être correctement lu.

IV.B.3.e. Lecture

La lecture peut échouer si la grille d'échantillonnage n'est pas alignée avec celle de la matrice. Dans ce cas, les pixels échantillonnés ne sont pas lus sur la bonne cellule, ou sont lus sur plusieurs cellules différentes.

La lecture utilise la luminance moyenne des pixels à l'intérieur d'une cellule par rapport à la luminance sur l'ensemble du symbole pour évaluer la valeur de la cellule (marquée ou non marquée). La lecture peut échouer aussi si les luminances des cellules marquées et des cellules non marquées ne sont pas nettement séparées. On ne peut pas dans ce cas décider de la valeur d'une cellule à partir de sa luminance.

IV.B.4. Sensibilité de la méthode

Nous allons maintenant passer en revue les différentes causes d'erreurs de la méthode, et indiquer pour chacune quelles étapes critiques sont concernées.

IV.B.4.a. Période spatiale

L'estimation du laplacien se fait à partir d'un masque de taille 5×5, et ses passages à zéro par un masque de taille 2×2, ce qui fait pour la détection de contour un masque équivalent de taille 6×6. La recherche de contour se base donc sur les luminances d'un groupe de pixels non ponctuel. Lors de cette estimation, la présence de contours à des distances proches du point considéré (2 à 3 pixels) peut la fausser. Avec une petite période spatiale, les lignes pleines du motif en *L* deviennent étroites, et ont le long de leur bord intérieur de nombreux contours rapprochés de la ligne à détecter, qui peuvent empêcher sa détection.

Le coin nominal d'horloge est parfois éloigné du coin d'horloge, en fonction de la distorsion projective. Si la fréquence spatiale est faible et de l'ordre de l'erreur sur le coin d'horloge, alors le détecteur de coin risque de détecter dans le voisinage du coin nominal le coin d'une autre cellule, et de provoquer une erreur de localisation.

Plus la période spatiale est faible et plus la largeur de la zone d'horloge est étroite La fréquence en est d'autant difficile à identifier, et requière une grande précision de localisation. Par exemple, avec une période spatiale de 3 pixels, seule une bande d'un à deux pixels de large contient des valeurs significatives. Cette bande ne pourra être échantillonnée que si la position des coins du symbole est précise au pixel près.

Si la période spatiale est faible, la distance entre les nœuds de la grille est faible. D'une part, cela réduit le nombre d'échantillons utilisés pour estimer la valeur des cellules, et donc augmente la sensibilité au bruit. D'autre part, cela nécessite une précision d'autant plus importante pour la localisation du symbole, car la grille d'échantillonnage est construite sur cette localisation.

IV.B.4.b. Contraste

Pour mesurer le contraste de l'image d'un symbole, nous mesurons un certain nombre de propriétés de l'image, illustrées figure IV-7.

Nous notons $\bar{L}$ la luminance moyenne sur l'image, $\bar{l}$ sa valeur exprimé en pourcentage de la dynamique de l'image : $\bar{l} = \bar{L}/256$. L_M et L_m sont respectivement les luminances minimales et maximales dans la zone du symbole, soit $\Delta L = L_M - L_m$. .

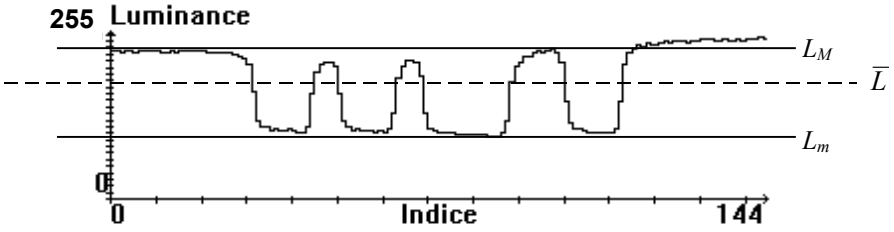


FIG. IV-7 - *profil le long d'un symbole*

Nous définissons C le contraste absolu : $C = \Delta L$ et c le contraste relatif exprimé en pourcentage de la dynamique de l'image : $c = (\Delta L)/256$.

Le contraste de l'image détermine l'amplitude des contours. Si l'image est peu contrastée, les contours peuvent ne pas être détectés, où avoir leur direction mal interprétée. La non-détection des contours peut empêcher la détection de lignes mais aussi l'évaluation des textures.

Pour un niveau de bruit donné, un faible contraste entraîne un rapport signal-sur-bruit faible, ce qui peut causer des problèmes de détection de lignes.

Avec un faible contraste, l'écart entre les luminances des cellules marquées et non marquées est aussi faible, ce qui peut occasionner des erreurs de lecture sur quelques cellules, surtout si ce faible contraste se cumule avec du bruit ou une faible période spatiale.

Si la période spatiale est faible, il est aussi plus difficile avec un faible contraste d'identifier la fréquence.

IV.B.4.c. Bruit

Nous quantifions le bruit σ comme étant l'écart type du signal dans le blanc de l'image, sous l'hypothèse d'un blanc globalement uniforme.

Les images ayant un blanc hétérogène, nous estimons σ par la moyenne de quatre mesures d'écart type dans des fenêtres de taille 16×16 prises dans du blanc aux quatre coins de l'image.

Pour comparer ce bruit à l'amplitude du symbole, nous calculons d'abord l'écart type d'un signal d'horloge nominal d'amplitude donnée ΔL. Cet écart type vaut $\frac{1}{2}\Delta L$.

Nous appelons *rapport signal sur bruit b* le rapport des écarts types : $b = \Delta L/2\sigma$.

Les transitions qui indiquent la présence de contours peuvent être masquées par des discontinuités locales dues au bruit si ce dernier est important. La forme du signal au voisinage d'un point de contour peut aussi être modifiée par la présence de bruit. Ces deux cas peuvent entraîner la non-détection de pixels de contours ou une estimation erronée de la direction de contour (les chaînes ne s'assembleront pas convenablement) ce qui peut empêcher la

détection du motif en *L*. Des transitions de luminance dues au bruit peuvent provoquer de fausses alarmes dans la carte de contours, ces pixels peuvent, s'ils sont nombreux, s'agglomérer en courtes lignes. Si ces lignes apparaissent à proximité du motif en *L*, elles peuvent interférer avec celui-ci lors de la restauration de ligne et empêcher sa détection.

En fonction de la période spatiale, un nombre variable de pixels sont échantillonnés pour établir la valeur des cellules. Si la période spatiale et faible, un niveau de bruit important peut provoquer des erreurs dans l'estimation de la valeur d'une cellule.

IV.B.4.d. Hétérogénéité du blanc

Nous appelons *gradient de luminance du blanc* g_e la pente moyenne de la luminance du blanc le long de l'image, mesurée en dehors des zones saturées. Nous appelons *écart de luminance* d_L l'amplitude de la variation de la luminance du blanc ramenée à la largeur du symbole : $d_L = T \times g_e$.

Pour mesurer g_e, nous calculons la luminance moyenne sur une fenêtre de taille 16×16 prises dans du blanc aux quatre coins de l'image. g_e est estimé par la plus grande pente calculée entre ces quatre mesures de blanc prises deux à deux.

La recherche de contour utilisée est basée sur des discontinuités locales de la luminance. Si le blanc est hétérogène autour du symbole, cela n'affecte pas la détection ou la localisation tant que le contraste reste suffisant sur toute la surface du symbole.

S'il y a une forte hétérogénéité du blanc, des erreurs de lecture peuvent se produire. Ce sera le cas si deux cellules de valeur différentes ont des luminances semblables, ou bien si deux cellules de valeur identiques ont des luminances de part et d'autre du seuil.

IV.B.4.e. Distorsions projectives

Les distorsions projectives font qu'un symbole apparaît déformé sur l'image s'il n'est pas présenté face à la caméra.

Nous appelons *angle d'incidence*, noté *i*, l'angle que fait l'axe optique de la caméra avec le plan du support du symbole (figure IV-8). En augmentant l'angle d'incidence, nous augmentons la distorsion projective. Cet angle est mesuré lors de la prise de vue. La caméra étant fixe, nous faisons varier l'orientation du support.

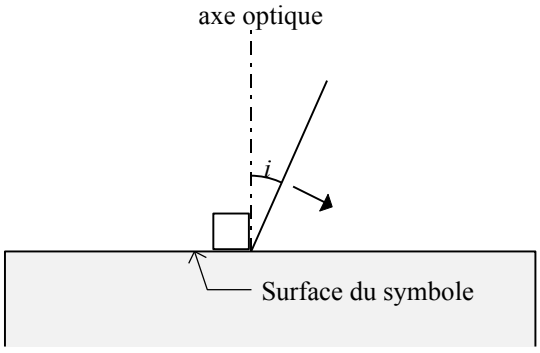


FIG. IV-8 - *angle d'incidence de la prise de vue*

Les distorsions projectives se traduisent par une déformation du symbole carré en un quadrilatère quelconque. Si la forme du quadrilatère est trop différente du parallélogramme bâti sur le motif en *L*, le coin d'horloge ne sera probablement pas détecté, causant une erreur de localisation.

IV.B.4.f. Distorsions optiques

Si les distorsions optiques sont importantes au point de courber les contours du motif en *L*, le motif risque de ne pas être détecté (en fonction du seuil de rectitude utilisé).

Indépendamment de ce problème, la lecture ne fonctionne plus dès que la flèche de la courbure est de l'ordre de la période spatiale. Dans ce cas, la grille d'échantillonnage n'est plus alignée avec celle, courbe, de la matrice du symbole.

IV.B.4.g. Focalisation

Nous définissons la défocalisation f_l comme étant l'épaisseur moyenne des contours entre cellules. Pour une image nette, cette valeur vaut 1.

Pour estimer cette valeur, nous définissons un intervalle de luminances appelé *zone de transition*. Nous avons fixé cet intervalle à $[Lm+¼\Delta L;Lm+¾\Delta L]$ (cet intervalle a comme largeur la moitié de l'amplitude des luminances du symbole et est centré sur leurs valeurs moyennes).

Dans un profil pris sur le signal d'horloge, f_l est estimé par le nombre de pixels ayant une luminance dans la zone de transition divisé par le nombre de transition (f+1).

Une défocalisation de l'image provoque les phénomènes suivants.

Les contours qui devraient être étroits sont répartis sur plusieurs pixels, et la transition de luminance sur le pixel est atténuée. Certains contours ne sont pas détectés.

Les caractéristiques de textures sont modifiées : réduction du nombre de contours, modification de leurs propriétés, modification de la répartition des luminances.

La détection de coin est aussi rendue plus incertaine. Elle peut ne pas détecter le coin d'horloge, ou le détecter avec une erreur de localisation.

Le flou peut donc entraîner un échec de la détection de lignes ou de la validation de texture.

IV.B.4.h. Objets perturbateurs

Une tache sur l'image (présente sur le capteur ou sur le support du symbole) peut provoquer une erreur de lecture sur une cellule. Mais sur la zone d'horloge, c'est l'identification de la fréquence qui peut échouer.

Si la zone vide n'est pas respectée (tache ou objets proches du motif en *L*), alors des lignes peuvent apparaître à proximité ou au contact du motif en *L* et empêcher sa détection.

IV.B.5. Limites de la méthode

Nous présentons ci-après divers tests réalisés afin de valider la méthode et déterminer ses limites. Nous utilisons des images de symboles, pour lesquels nous faisons varier un ou plusieurs des éléments vus ci-dessus (taille, période, netteté...). Pour chaque image, le résultat obtenu est soit un échec (symbole non détecté ou non lu), soit un succès, avec sur tous les cas présentés une erreur de lecture de moins de 1,5 % (sur cinq images uniquement).

Nous indiquons pour chaque série de test les résultats de lecture obtenus pour un certain nombre de conditions que nous allons d'abord quantifier ; nous interprétons les résultats au § IV.IV.B.IV.

IV.B.5.a. Variation de la période spatiale (symbole isolé, fréquence 21)

Dans les images IV-9A à IV-9I, le symbole S_{31}, de fréquence propre f=21, est seul dans l'image et bien contrasté. Nous faisons varier la taille et la période spatiale du symbole en faisant varier la distance de la caméra au symbole.

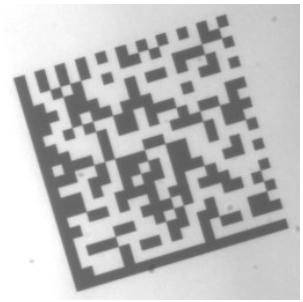


FIG. IV-9A - *t=9,1 px, T=191 px*

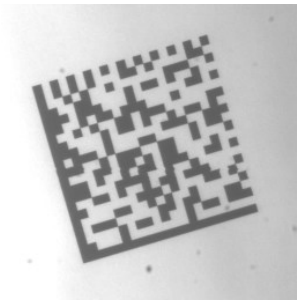


FIG. IV-9B - *t=7,5 px, T=158 px*

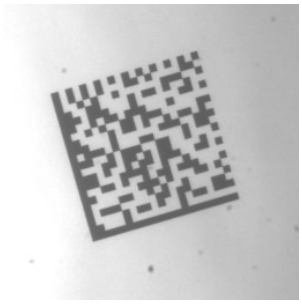


FIG. IV-9C - *t=6,3 px, T=132 px*

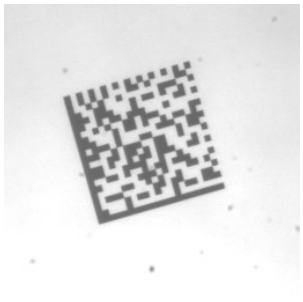


FIG. IV-9D - *t=5,4 px, T=113 px*

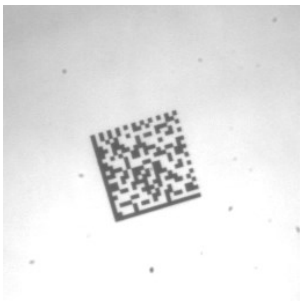


FIG. IV-9E - *t=3,7 px, T=78 px*

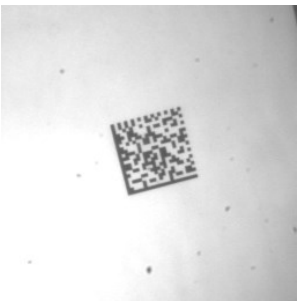


FIG. IV-9F - *t=3,0 px, T=63 px*

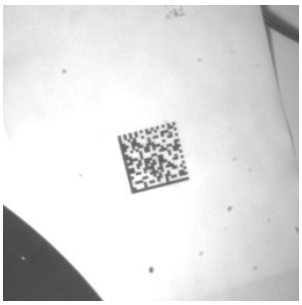


FIG. IV-9G - *t=2,1 px, T=52 px*

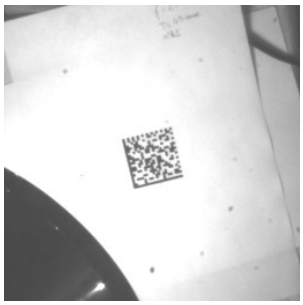


FIG. IV-9H - *t=2,5 px, T=46 px*

FIG. IV-9I - *t=0,86 px, T=40 px*

FIG. IV-9 - *images du symbole S_{31} de fréquence propre f=21 à période spatiale t et taille T variable (série BAN31a21)*

Le symbole est trouvé et lu pour les images IV-9A à IV-9F. La détection échoue pour les images IV-9G à IV-9I.

IV.B.5.b. Variation de la période spatiale (symbole non isolé, fréquence 21)

Dans les images IV-10A à IV-10I, le symbole S_{31}, de fréquence propre f=21, est placé dans une scène contenant d'autres éléments.

FIG. IV-10A - *t=7,5 px, T=158 px*

FIG. IV-10B - *t=6,4 px, T=134 px (image ban31q21)*

FIG. IV-10C - *t=5,5 px, T=116 px*

FIG. IV-10D - *t=4,8 px, T=101 px*

FIG. IV-10E - *t=3,9 px, T=82 px*

FIG. IV-10F - *t=3,0 px, T=63 px*

FIG. IV-10G - *t=2,52 px, T=53 px*

FIG. IV-10H - *t=2,19 px, T=46 px*

FIG. IV-10I - *t=0,95 px, T=20 px*

FIG. IV-10 - *images du symbole S_{31}, non isolé, de fréquence propre f=21 à période spatiale t et taille T variable (série BAN31r21)*

Le symbole est trouvé et lu pour les images IV-10A à IV-10F. La détection échoue pour les images IV-10G à IV-10I.

IV.B.5.c. Variation de la période spatiale (symbole non isolé, fréquence 43)

Dans les images IV-11A à IV-11F, le symbole S_{32}, de fréquence propre f=43, est placé dans une scène contenant d'autres éléments.

FIG. IV-11A - *t=4,0 px, T=172 px*

FIG. IV-11B - *t=3,1 px, T=133 px*

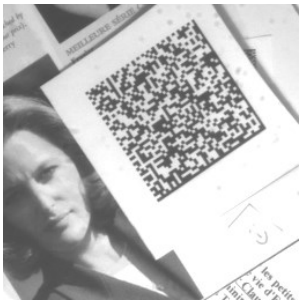


FIG. IV-11C - *t=2,7 px, T=115 px*

FIG. IV-11D - *t=2,4 px, T=102 px*

FIG. IV-11E - *t=1,8 px, T=79 px*

FIG. IV-11F - *t=1,5 px, T=63 px*

FIG. IV-11 - *images du symbole S_{32} , non isolé, de fréquence propre f=43 à période spatiale t et taille T variable (BAN32a43)*

Le symbole est trouvé et lu pour les images IV-11A à IV-11B. La détection échoue pour les images IV-11C à IV-11F.

IV.B.5.d. Variation du contraste

Dans les images IV-12A à IV-12E, le symbole S_{33}, de fréquence propre f=11, est isolé. La période spatiale et la taille sont constantes à t=13,1 et T=144. Le contraste diminue (du plus contrasté au moins contrasté). Les courbes de la figure IV-13 montrent comment nous avons fait varier contraste et luminance moyenne pour obtenir les images tests. Nous avons gardé un rapport signal-sur-bruit constant.

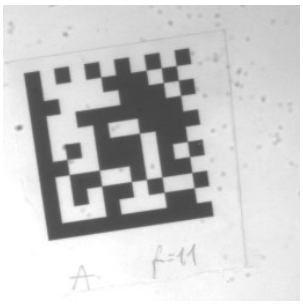


FIG. IV-12A - *c=71 %, $\overline{L}$ =200*

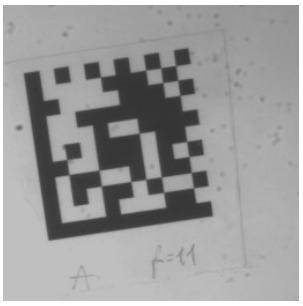


FIG. IV-12B - *c=49 %, $\overline{L}$ =150*

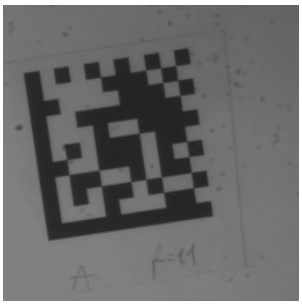


FIG. IV-12C - *c=32 %, $\overline{L}$ =100*

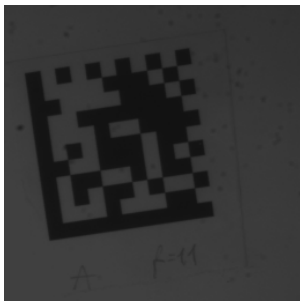


FIG. IV-12D - *c=18 %, $\overline{L}$ =50*

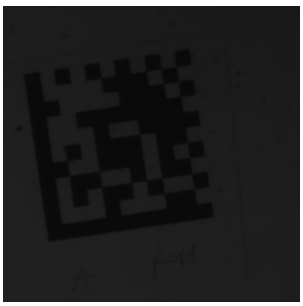


FIG. IV-12E - c=9 %, $\overline{L}$ *=25*

FIG. IV-12 - *images du symbole S_{33} de fréquence propre f=11, de période spatiale t=13,1 et taille T=144, avec contraste variable (série HAN33a11)*

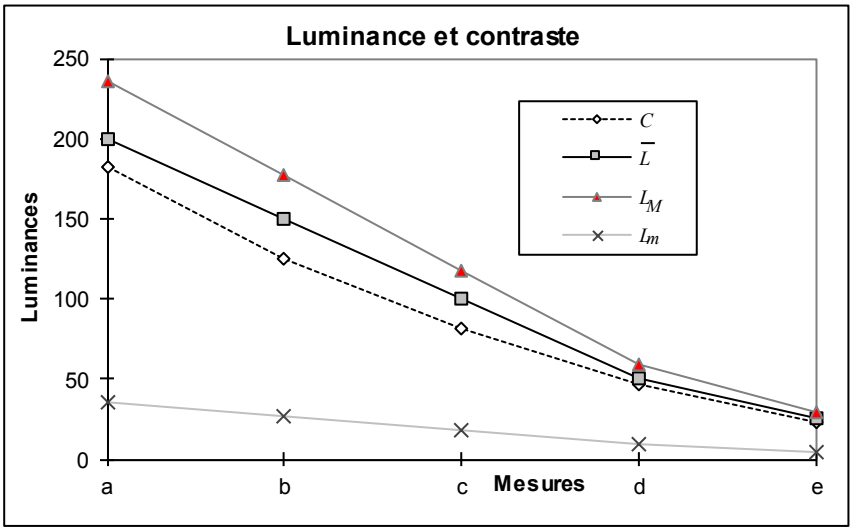


FIG. IV-13 - *variation des contraste et luminance*

Le symbole est trouvé et lu pour les images IV-12A à IV-12D. La détection échoue pour l'image IV-12E.

IV.B.5.e. Variation du contraste à niveau de bruit constant

Dans les images IV-14A à IV-14I, le symbole S_{31}, de fréquence propre f=21, est placé dans une scène avec d'autres éléments. La période spatiale et la taille sont constantes à t=3,9 et T=81. Les courbes de la figure IV-15 montrent comment nous avons fait varier contraste et luminance pour obtenir les images tests (du plus clair au plus sombre, avec un contraste maximum entre les deux). L'amplitude du bruit reste constante, le rapport signal-sur-bruit évolue comme le contraste.

FIG. IV-14A - $\bar{l}$ *=93 %, c=28 %*

FIG. IV-14B - $\bar{l}$ *=89 %, c=40 %*

FIG. IV-14C - $\bar{l}$ *=84 %, c=52 %*

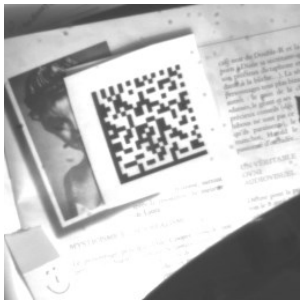


FIG. IV-14D - $\bar{l}$ *=75 %, c=68 %*

FIG. IV-14E - $\bar{l}$ *=68 %, c=76 %*

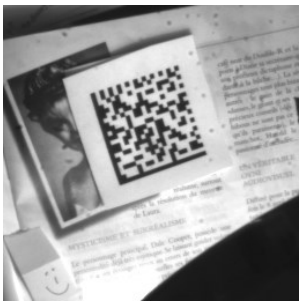


FIG. IV-14F - $\bar{l}$ *=61 %, c=84 %*

FIG. IV-14G - $\bar{l}$ *=48 %, c=79 %*

FIG. IV-14H - $\bar{l}$ *=39 %, c=64 %*

FIG. IV-14I - $\bar{l}$ *=25 %, c=40 %*

FIG. IV-14J - $\bar{l}$ *=20 %, c=32 %*

FIG. IV-14 - *images du symbole S_{31} de fréquence propre f=21, de période spatiale t=3,9 et taille T=81, avec luminance et contraste variables (série CAN31a21)*

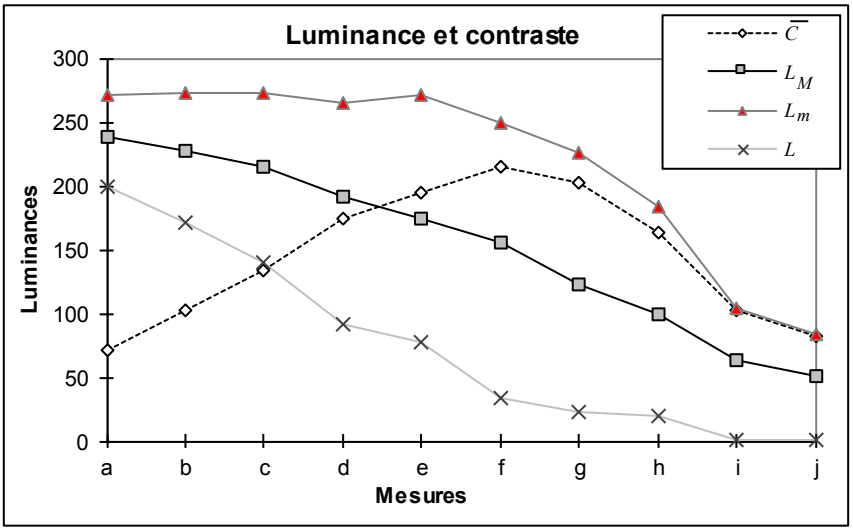


FIG. IV-15 - *variation des contraste et luminance*

Le symbole est trouvé et lu pour les images IV-14C à IV-14H. La détection échoue pour les images IV-14A, IV-14B, IV-14I et IV-14J.

IV.B.5.f. Variation du bruit à contraste constant

Dans les images IV-16A à IV-16H, le symbole S_{33}, de fréquence propre f=11, est isolé. La période spatiale et la taille sont constantes à t=11,5 et T=127 ; le rapport signal sur bruit diminue.

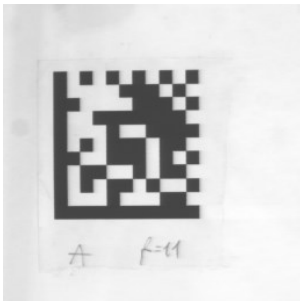


FIG. IV-16A - *s=1,16, b=26,0*

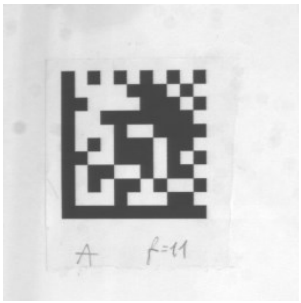


FIG. IV-16B - *s=1,26, b=22,5*

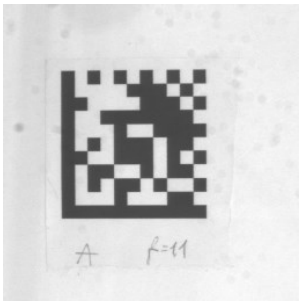


FIG. IV-16C - *s=1,37, b=20,0*

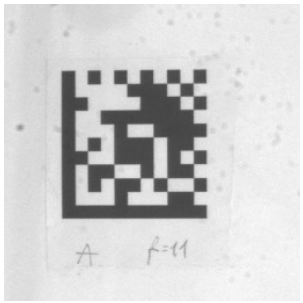


FIG. IV-16D - *s=1,48, b=17,5*

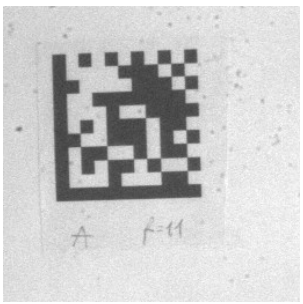


FIG. IV-16E - *s=2,34, b=10,5*

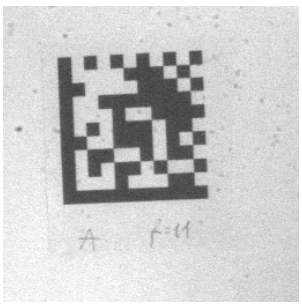


FIG. IV-16F - *s=2,79, b=8*

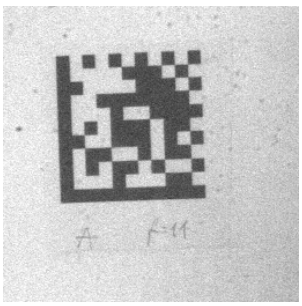


FIG. IV-16G - *s=3,99, b=5,5*

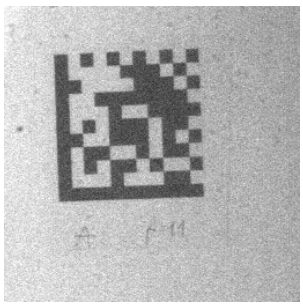


FIG. IV-16H - *s=5,08, b=4,1*

FIG. IV-16 - *images symbole S_{31} isolé, de fréquence propre f=11, de période spatiale t=11,5 et taille T=127, avec variation du bruit (série GAN33a11)*

Le symbole est trouvé et lu pour les images IV-16A à IV-16D. La détection échoue pour les images IV-16E à IV-16H.

IV.B.5.g. Variation de l'homogénéité de la luminance du blanc

Dans les images IV-17A à IV-17H, le symbole S_{33}, de fréquence propre f=11, est isolé. La période spatiale et la taille sont respectivement de t=10,2 et T=114 pour les images IV-17A à IV-17E et de t=15,1 et T=166 pour les images IV-17F à IV-17H L'éclairage est non homogène sur la scène, et présente un gradient dont la valeur augmente (excepté entre IV-17E et IV-17F).

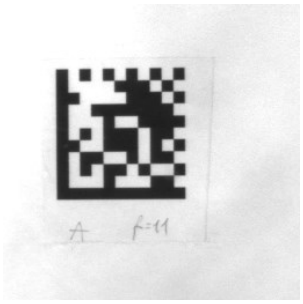


FIG. IV-17A - g_e=0,09 px^{-1}, d_L=10

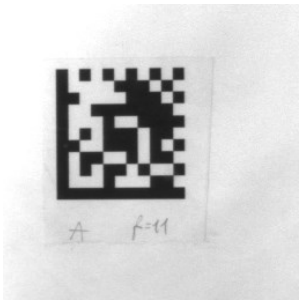


FIG. IV-17B - g_e=0,14 px^{-1}, d_L=16

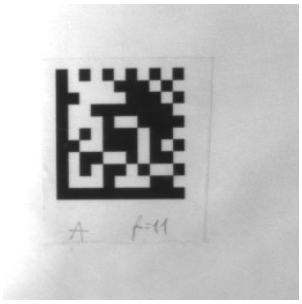


FIG. IV-17C - g_e=0,28 px^{-1}, d_L=32

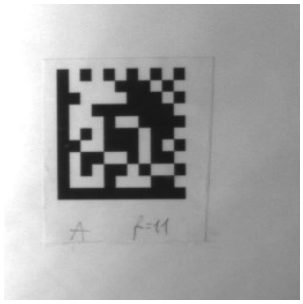


FIG. IV-17D - g_e=0,48 px^{-1}, d_L=55

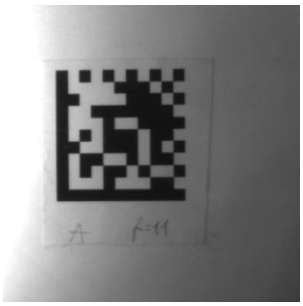


FIG. IV-17E - g_e=0,76 px^{-1}, d_L=87

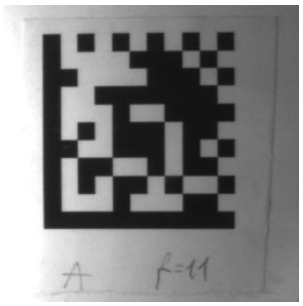


FIG. IV-17F - g_e=0,64 px^{-1}, d_L=73

FIG. IV-17G - g_e=0,71 px^{-1}, d_L=81

FIG. IV-17H - g_e=1,15 px^{-1}, d_L=131

FIG. IV-17 - *images du symbole S_{33} de fréquence propre f=11, avec variation du gradient de l'éclairage (série FAN33k11)*

Le symbole est trouvé et lu pour les images IV-17A à IV-17G. La détection échoue pour l'image IV-17H.

IV.B.5.h. Variation de la distorsion projective

Dans les images IV-18A à IV-18G, le symbole S_{31}, de fréquence propre f=21, est placé dans une scène avec d'autres éléments. La période spatiale et la taille sont constantes à t=3,8 et T=80.

FIG. IV-18A - *i=0°*

FIG. IV-18B - *i=3,6°*

FIG. IV-18C - *i=7,1°*

FIG. IV-18D - *i=10°*

FIG. IV-18E - *i=14°*

FIG. IV-18F - *i=25°*

FIG. IV-18G - *i=30°*

FIG. IV-18 - *images du symbole S_{31} de fréquence propre f=21, de période spatiale t=3,8 et taille T=80, avec distorsion projective variable (série EAN31a21)*

Le symbole est trouvé et lu pour les images IV-18A à IV-18F. La détection échoue pour l'image IV-18G.

IV.B.5.i. Variation de la focalisation

Dans les images IV-19A à IV-19E, le symbole S_{31}, de fréquence propre f=21, est placé dans une scène avec d'autres éléments. La période spatiale et la taille sont constantes à t=3,8 et T=80. La focalisation diminue (du plus net au plus flou).

FIG. IV-19A - *f_l=1,0 px*

FIG. IV-19B - *f_l=1,2 px*

FIG. IV-19C - *f_l=1,3 px*

FIG. IV-19D - *f_l=1,4 px*

FIG. IV-19E - f_l=2,6 px

FIG. IV-19 - *images du symbole S_{31} de fréquence propre f=21, de période spatiale t=3,8 et taille T=80, avec focalisation variable (série DAN31a21)*

Le symbole est trouvé et lu pour les images IV-19A à IV-19C. La détection échoue pour les images IV-19D et IV-19E.

IV.B.5.j. Interprétations des résultats

Les tests menés sur divers types d'image, avec différents types d'éléments perturbateurs mettent en évidence les conditions dans lesquelles fonctionne la méthode. Dans ces tests, nous avons fait varier isolément ou en groupes réduits les facteurs de bruit, contraste, homogénéité du blanc, focalisation, distorsions, période spatiale.

Nous allons étudier comment ces facteurs sont liés entre eux et l'efficacité de la méthode lorsqu'ils varient simultanément.

Nous voyons que dans de bonnes conditions d'acquisition d'image, la méthode est limitée aux périodes spatiales supérieures à 3 pixels. Si la période spatiale est plus faible, alors la distance entre les contours des cellules est inférieure à la taille du masque qui permet d'établir la carte de contours. La largeur de la zone d'horloge devient trop étroite pour pouvoir identifier la fréquence. Le contraste dans le symbole diminue, ce qui diminue le rapport signal-sur-bruit.

À faible période spatiale, la distance entre les pixels de contours du symbole diminue, ce qui augmente la valeur de critère de proximité dans l'analyse de texture. Ce critère est un critère d'exclusion qui repose sur le fait que dans un symbole, les contours sont relativement éloignés les uns des autres.

Des méthodes qui corrige l'hétérogénéité du blanc existent, en se basant sur une connaissance de la répartition du blanc, soit par calibration dans une phase préalable, soit par estimation du champ de luminance du blanc, soit par modélisation. Ces méthodes seraient assez délicates à mettre en œuvre, et relativement inutiles car nous avons vu que notre méthode est relativement peu sensible au contraste et à l'homogénéité du blanc. Elle est par contre assez sensible au bruit.

La détection de ligne est basée sur la discontinuité locale de la luminance. L'amplitude de ces discontinuités ne doit pas nécessairement être trop élevée ni constante sur l'ensemble du symbole. Ceci explique que la méthode soit peu sensible au contraste et à l'hétérogénéité de l'éclairage. L'amplitude des contours doit cependant être supérieure aux seuils utilisés lors du calcul des cartes de contours. Contrairement à la détection de lignes qui n'en utilise pas, l'analyse de texture se fait à partir d'une carte de contours seuillés.

La composition de ligne se base sur la direction des contours. Lorsque le rapport signal-sur-bruit dans l'image est faible, l'estimation de cette direction peut être faussée, et en-

traîner des ruptures dans les chaînes. Ces ruptures sont corrigées par l'étape de restauration, mais si le bruit est trop important, les lignes restent fragmentées, et la détection du symbole échoue.

Le contraste et le rapport signal-sur-bruit sont liés. Si on considère l'amplitude du bruit constante dans le système d'acquisition d'image, plus le contraste est faible et plus le rapport signal-sur-bruit est faible.

L'effet de la distorsion, optique ou projective, dépend de la période spatiale. Nous avons vu que ces distorsions n'entraînent des erreurs de localisation ou de lecture que si la distance entre le symbole supposé et le symbole dans l'image est de l'ordre de la période spatiale. Une déformation notable du symbole due à une distorsion aura peu d'effet si la période spatiale est élevée.

Les optiques utilisées durant les mesures n'ont pas entraîné de déformations notables, quelle que soit la distance au symbole. L'effet des distorsions optiques est négligeable dans le cadre de notre application.

Nous avons pu par contre mettre en évidence les distorsions projectives et leur effet sur le processus.

L'expérience indique qu'un angle de 25° sur un symbole de période spatiale de 3,8 px, proche de la limite minimale, permet encore de trouver et de lire le symbole. Cependant, la distance entre le symbole supposé et le symbole dans l'image ne dépend pas uniquement de l'angle d'incidence, mais aussi de la façon dont les axes du symbole se situent par rapport à l'axe optique. En particulier, un axe du symbole orthogonal à l'axe optique subit peu de distorsion projective.

La méthode est relativement sensible au flou qui cause des erreurs de détection, de localisation et d'analyse de texture. En ce qui concerne la détection, les effets du flou dépendent du contraste. Le flou a pour effet de réduire la détection des contours. Si le contraste est faible, les contours ont déjà une amplitude faible, et les effets du flou sont aggravés.

Nous avons donc pu cerner les conditions qui permettent à la méthode de fonctionner. L'utilisation de la méthode est prévue sur un dispositif qui lit les codes présentés devant l'objectif. Il est prévu que ce dispositif soit muni de sa propre source lumineuse, afin de ne pas être dépendant de l'éclairage ambiant.

Nous avons donc la possibilité de choisir l'optique et le type d'éclairage adapté aux conditions de fonctionnement de la méthode. En choisissant un fort éclairage, même hétérogène, le bruit dans l'image est réduit, ce qui met l'application dans le champ de fonctionnement de la méthode.

IV.C.Temps d'exécution

IV.C.1. Mesures détaillées de durées

Le programme a été codé selon les principes suivants :

- Lisibilité : le fonctionnement des procédures doit se comprendre en lisant le programme, afin de permettre une maintenance et une reprise par d'autres programmeurs. Un code lisible est en outre plus facilement vérifiable et donc potentiellement moins porteur d'erreurs.

- Rapidité d'écriture : on privilégie l'utilisation de procédures déjà écrites et donc déjà testées et validées. Ceci est facilité par une programmation modulaire où les calculs des différentes étapes sont séparés les uns des autres.

- Robustesse vis-à-vis des erreurs : vérification à l'entrée des procédures que les paramètres reçus sont conformes aux spécifications (pointeurs et diviseurs non nuls, index de tableau dans les intervalles autorisés…), afin de détecter au plus tôt les erreurs du programme. Ceci permet à la fois de localiser les erreurs dans le processus et d'éviter par propagation d'erreurs non-récupérables le blocage du programme, voire du système d'exploitation.

Ces principes, utiles en phase de développement de logiciel, ne sont pas toujours compatibles avec une optimisation en quantité de calculs (et donc en vitesse). Des simplifications et des optimisations du code sont donc possibles afin d'obtenir une réduction sensible de la quantité de calculs.

Nous avons mesuré les temps d'exécution des différentes étapes du processus. Ces mesures permettent de constater quelles sont les étapes les plus limitatives en terme de vitesse d'exécution.

La figure IV-20 représente les durées d'exécution de chaque procédure et sous procédure du programme. Elles sont identifiées par des abréviations entres parenthèses. Les durées ont été obtenues avec l'image de la figure IV-10B (image ban31q21) et la version du logiciel d'octobre 1997.

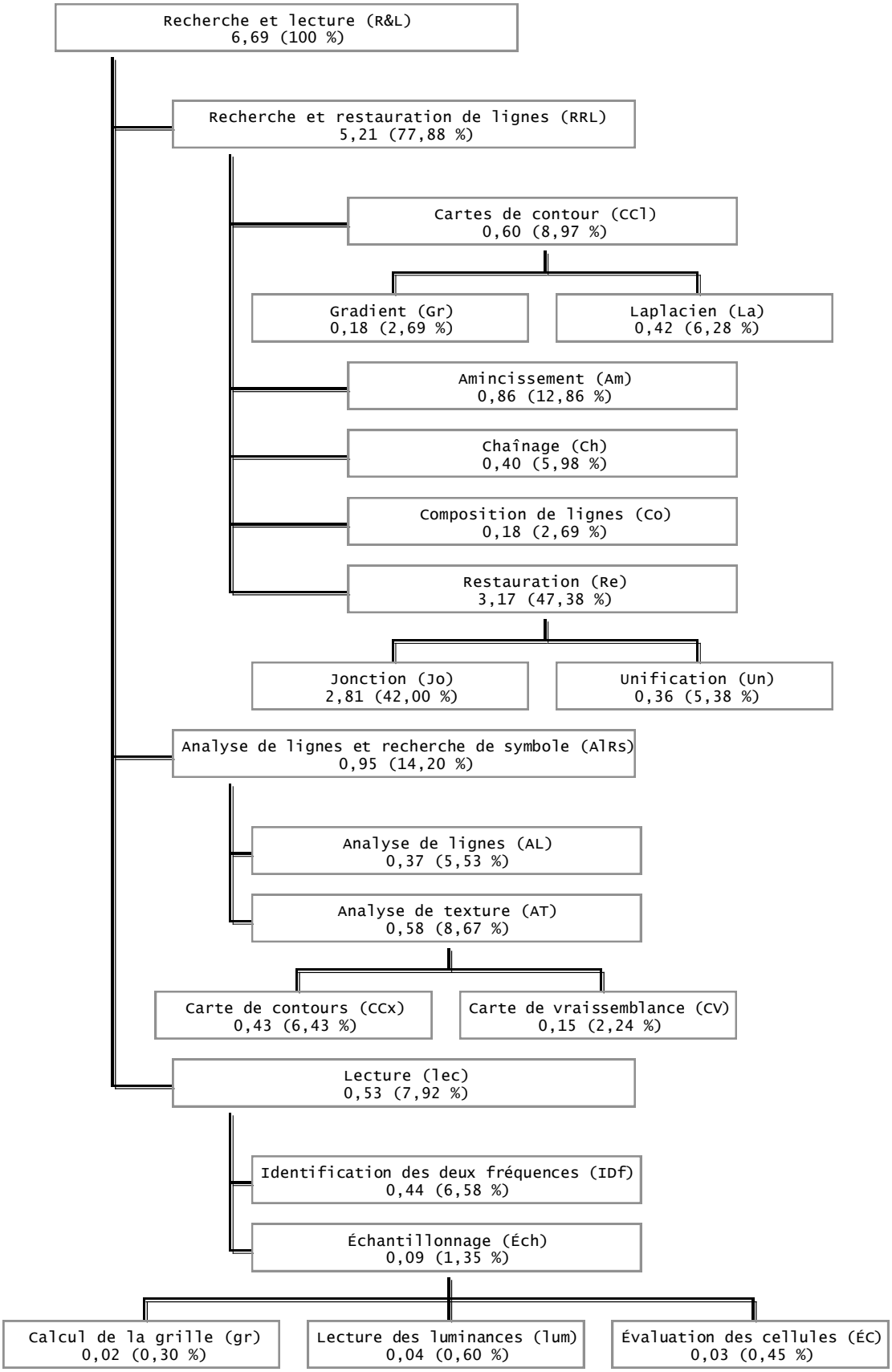


FIG. IV-20 - *durée des procédures et sous procédures (en secondes)*

La figure IV-21 représente la répartition des durées d'exécution. Les secteurs séparés du reste représentent l'ensemble des procédures de recherche et de restauration de ligne (RRL, qui fait 77,88 % de la durée totale).

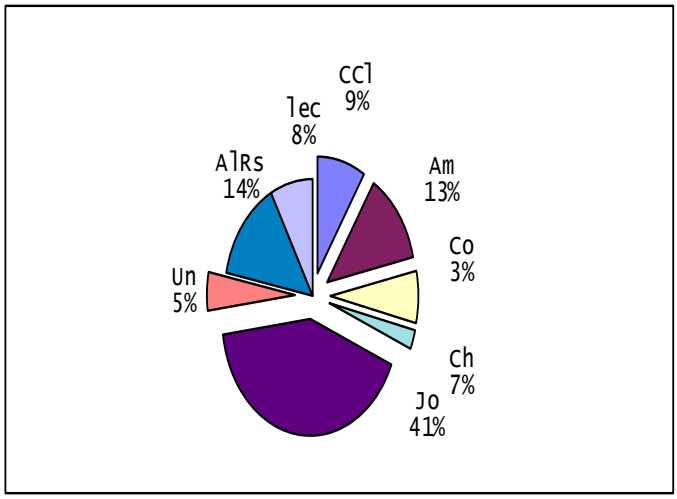


FIG. IV-21 - *répartition des durées des procédures et sous procédures*

Nous pouvons voir en première approche que la plupart du temps de calcul est consommée par la recherche et la restauration de lignes (RRL), et en particulier cette dernière (Re) qui dure près de la moitié du temps total. Le traitement des lignes (Co et Re) dure 3,72 s (56 %).

Viennent ensuite les traitements de contours qui représentent globalement (recherche de lignes et analyse de texture : CCl, Am, Ch, CCx) 2,29 s (34 %). La lecture (lec) prend 0,53 s (8 %) et les cartes de texture (CV) 0,15 s (2 %).

Cependant, ces comparaisons doivent être faites avec prudence, car les différentes procédures ont des durées variables en fonction du type d'image et de symbole lu. Nous allons étudier quels sont les facteurs qui déterminent la quantité de calculs.

IV.C.2. Complexité de la méthode

La complexité est la façon dont la quantité de calculs varie en fonction des données à traiter. Le temps d'exécution est directement lié à la quantité de calculs. Nous avons identifié les paramètres qui influent de manière notable sur la quantité de calculs.

- La taille, en pixels, du support de l'image (la matrice de pixels). Soit N cette valeur. Si la matrice n'est pas carrée, on considère N comme la racine carrée du nombre de pixels total de la matrice. De ce paramètre dépend la surface de la matrice : N^2.

- La densité de contours en dehors du symbole, c_e, qui représente le nombre de pixels de contour par unité de surface. On en déduit la quantité de contours $C_e=c_eN^2$.

- La densité de lignes en dehors du symbole, l_e, qui représente le nombre de lignes par unité de surface. On en déduit la quantité de lignes $L_e=l_eN^2$.

- La fréquence propre du symbole f.

- Le rapport de taille entre le symbole et l'image $K_T=T/N$ ($K_T\in]0;1[$).

c_e, et l_e, dépendent de la texture de l'environnement, et nous les considérerons comme des constantes liées au domaine d'utilisation de l'application.

Lorsqu'un symbole est présent dans l'image, il contient une certaine quantité de contours et de lignes que nous allons évaluer.

La quantité de contours C_s dans un symbole varie en fonction de l'agencement (aléatoire) des cellules, et on l'évalue par une fraction statistiquement constante (K_C) du nombre de contours du quadrillage de la matrice, c'est à dire la somme des longueurs des lignes et colonnes du quadrillage de sa matrice : $C_s=K_C2T(f+1)=2K_CNK_T(f+1)$.

La quantité de lignes L_s dans un symbole varie et est évaluée de la même manière par une fraction K_L du nombre de segments du quadrillage de la matrice : $L_s=K_L2f(f+1)$.

La quantité totale de contours et de lignes se déterminent par la somme de la quantité de contour dans la surface occupée par l'environnement (l'image sans le symbole, d'aire N^2-T^2) et des contours du symbole. Nous procédons de la même façon pour les lignes.

$$C = c_e(N^2-T^2)+C_s = c_e(N^2-T^2)+2K_CT(f+1) = c_eN^2(1-K_T^2)+2K_CNK_T(f+1) \text{ (IV-3)}$$

$$L = l_e(N^2-T^2)+L_s = l_e(N^2-T^2)+K_L2f(f+1) = l_eN^2(1-K_T^2)+K_L2f(f+1) \text{ (IV-4)}$$

Nous allons faire varier les paramètres N et f individuellement. Pour estimer la croissance de la quantité de calcul, nous étudions les équivalences de cette fonction lorsque ce paramètre tend vers ses valeurs limites ($N\to+\infty, f\to+\infty$).

Nous rappelons que deux fonctions $f(x)$ et $g(x)$ sont équivalentes lorsque x tend vers une limite L (finie ou non) si et seulement si $\frac{f(x)}{g(x)} \xrightarrow{x\to L} 1$.

Nous notons par le signe ~ deux fonctions équivalentes entre elles à une constante multiplicative près.

$$\left(f(x) \sim g(x)\right)_{x\to L} \Leftrightarrow \left\{ \exists K_L \in \mathbb{R}; K_L \frac{f(x)}{g(x)} \xrightarrow{x\to L} 1 \right\} \text{ (IV-5)}$$

Des équations IV-3 et IV-4 nous pouvons déduire :

$$\left(C(N) \sim N^2\right)_{N\to+\infty}, \left(L(N) \sim N^2\right)_{N\to+\infty}, \left(C(f) \sim f\right)_{f\to+\infty}, \left(L(f) \sim f^2\right)_{f\to+\infty} \qquad \text{(IV-6)}$$

Les procédures n'effectuant pas les mêmes tâches, les quantités de calculs à effectuer ne dépendent pas des mêmes facteurs et ne croissent pas de la même manière. Nous allons passer en revue les différents types de procédure du programme et indiquer la façon dont le temps de calcul croît en fonction de ces facteurs.

IV.C.2.a. Recherche et restauration de lignes

Les quantités de calcul des procédures détection de contours par convolution croissent de manière proportionnelle au nombre de pixels N^2. Les procédures de recherche de lignes à partir des cartes de contours croissent de manière proportionnelle au nombre de contours C.

Les fonctions de restauration et analyse de lignes cherchant à apparier des lignes entre elles, la quantité de calcul est proportionnelle au carré du nombre de lignes (L^2).

Étant donné les équations IV-6, nous pouvons écrire $\left(L(N)^2 \sim N^4\right)_{N\to+\infty}$. Nous pouvons en déduire que cette croissance est dominante par rapport à N^2 ou C.

Si N ou f croît, la quantité de calculs de la recherche et la restauration de ligne croît donc comme respectivement N^4 et f^4.

IV.C.2.b. Analyse de texture

En supposant que la taille du texel t_x reste constante, nous appelons n la taille de l'image en texels ($n = N / t_x$).

La quantité de calcul de l'évaluation d'une carte de texture est proportionnelle au nombre de texels n^2. La texture n'est évaluée que pour l'intérieur du symbole. Soit q_x la quantité de calculs à effectuer ; nous avons $\left(q_x(N) \sim K_T{}^2 n^2 \sim K_T{}^2 N^2\right)_{\substack{N\to+\infty \\ K_T\to+\infty}}$.

Les analyses de textures se font à plusieurs échelles, décalées d'un facteur 2. À chaque échelle, nous doublons la taille du texel équivalent. le nombre de changement d'échelles est limité par la taille du symbole T.

L'échelle la plus petite possible H est telle que la taille du texel équivalent atteint celle du symbole. Soit n_H le niveau d'échelle correspondant ($H=2^{-n_H}$). Le texel équivalent est de taille $2^{n_H}\ t_x$.

$$2^{n_H}\ t_x = T \Leftrightarrow n_H = \mathrm{Log}_2(T/t_x) = \mathrm{Log}_2(K_T N/t_x) = \mathrm{Log}_2(K_T n) \quad \text{(IV-7)}$$

Le nombre d'échelles utilisées n_{ech} est entier et vaut, en comptant l'échelle 1 : $\mathrm{E}(\mathrm{Log}_2(K_T n))+1$.

Par exemple, pour $T=K_T N=128$ px, $t_x=8$ px, $K_T n = \dfrac{T}{t_x} = \dfrac{128}{8} = 16$, Les analyses possibles le sont avec des texels équivalents de taille 8 px, 16 px, 32 px, 64 px et 128 px. Cela fait donc 5 changements d'échelles ($\mathrm{Log}_2(16)+1$).

À chaque changement d'échelle, la taille du texel équivalent double, ce qui divise par quatre le nombre de texels équivalents du symbole, et donc le temps de calcul. Le temps de calcul cumulé est donc la somme des termes d'une suite géométrique de raison ¼, dont le nombre de terme est connu. Soit Q_x la quantité totale de calculs :

$$Qx = \sum_{e_{ch}=0}^{n_{ech}-1} \frac{q_x}{4^{e_{ch}}} = q_x \frac{1-¼^{n_{ech}}}{1-¼} = ⅔\, q_x \left(1 - ¼^{\mathrm{E}\left(\mathrm{Log}_2\left(K_T N / T_x\right)\right)+1}\right) \quad \text{(IV-8)}$$

$$Q_x(N) \sim q_x(N) \sim K_T N^2 \quad \text{(IV-9)}$$

Si N croît, la quantité de calcul sur cette partie du programme croît comme N^2.

IV.C.2.c. Lecture

La quantité de calculs de l'identification des fréquences est proportionnelle au nombre de profils à analyser (T) et à la longueur de chaque profil (T) soit à $T^2=N^2K_T^2$.

La quantité de calculs de l'échantillonnage des luminances est proportionnelle à la surface du symbole $T^2=N^2K_T^2$.

La quantité de calculs de l'évaluation de la grille d'échantillonnage et des valeurs des cellules est proportionnelle au nombre de cellules f^2.

Si N ou f croissent, la quantité de calcul sur cette partie du programme croît donc respectivement comme N^2 ou f^2.

IV.C.2.d. Ensemble du programme

Nous avons déterminé la quantité de calcul que nécessite l'algorithme à partir de données liées à l'image et au symbole.

En mesurant et en étudiant les durées individuelles des procédures, nous avons pu déterminer les coefficients de proportionnalité entre les durées de traitement et les paramètres dont elles dépendent (N, f…).

Nous pouvons donc, pour des images aux caractéristiques connues, prévoir les durées d'exécution et comparer avec les durées réelles mesurées lors de l'exécution du programme.

Nous avons fait cette comparaison pour la série d'images BAN31a21 et BAN31r21 (figure IV-9 et IV-10).

Dans la série BAN31r21, le paramètre K_T seul varie. Nous utilisons les images IV-10A à IV-10D qui ont le même environnement, dont les densités de contours et densité de ligne sont celles de l'image ban31q21 utilisée comme référence (IV-10B). Les courbes obtenues sont présentées figure IV-22.

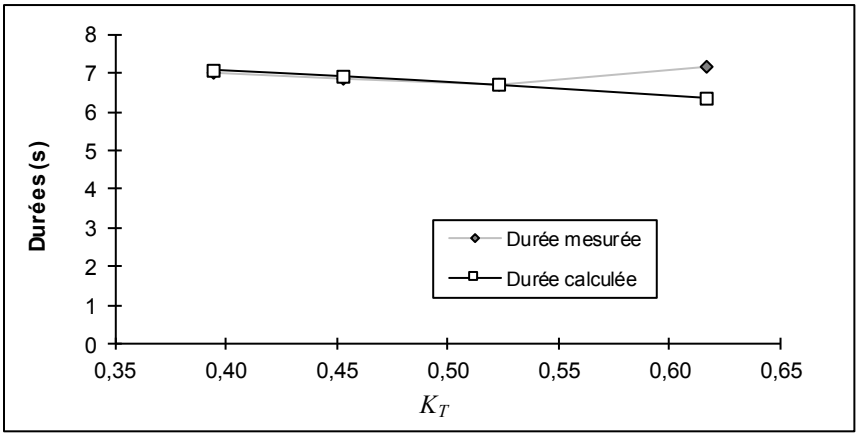


FIG. IV-22 - *comparaison des durées d'exécution : théorique et réelle (série BAN31r21)*

Dans la série BAN31a21, le symbole est isolé, et nous avons donc utilisé les paramètres adéquats de densité de contours et de densité de ligne (l_e de valeur zéro et c_e de valeur faible, de l'ordre du dixième de sa valeur dans la série BAN31r21). Les courbes obtenues sont présentées figure IV-23.

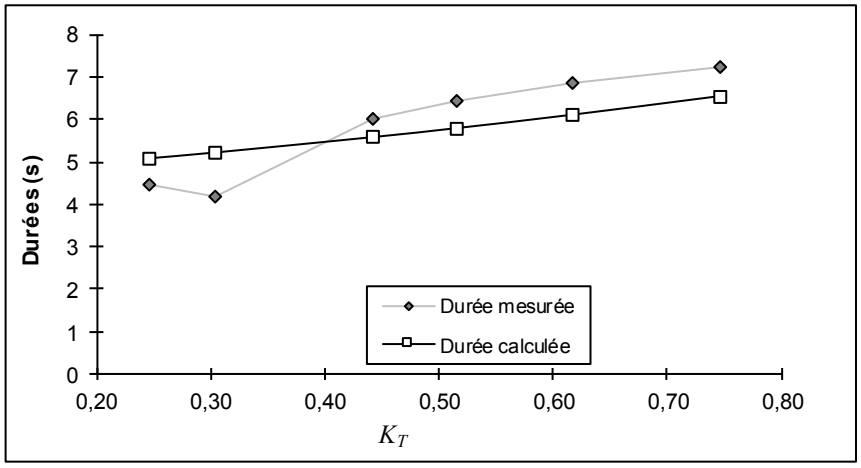


FIG. IV-23 - *comparaison des durées d'exécution : théorique et réelle (série BAN31a21)*

Les figures IV-22 et IV-23 indiquent que les modèles de complexité sont tout à fait satisfaisants. Si les durées mesurées et les durées estimées montrent parfois une petite différence, les tendances sont par contre identiques.

IV.C.3. Mesures de durées sur d'autres machines

Les mesures de durées présentées jusqu'ici ont été faites avec un processeur Pentium 100 MHz.

Nous avons effectué des mesures de durées sur diverses images avec un processeur MMX 200, et avons obtenu un gain en temps de calcul de facteur 1,37 à 2,15. Nous obtenions en effet en 3 à 5 secondes, ce que nous obtenions en 5 à 7 secondes avec le Pentium 100 MHz.

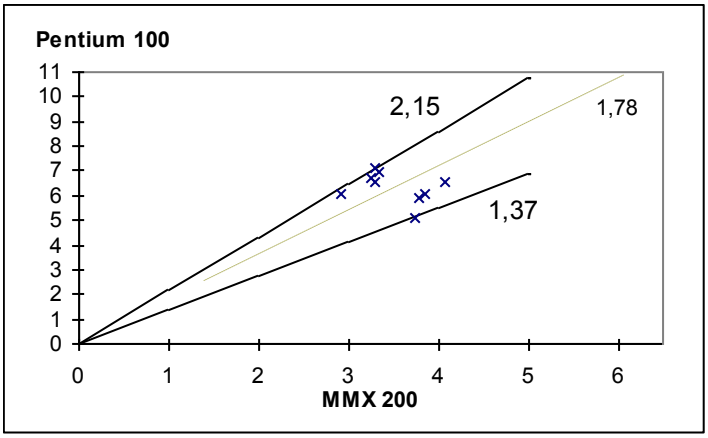


FIG. IV-24 - *comparaison des durées d'exécution :Pentium 100 MHz et MMX 200*

La figure IV-24 présente les mesures obtenues avec les deux processeurs, et les facteurs de gains (extrêmes et moyen).

IV.C.4. Extrapolation en fonction de la taille des images

Nous pouvons utiliser les résultats de cette étude de durées pour déterminer les durées d'exécution lorsque la taille du support de l'image (N) augmente.

En prenant les mêmes symboles, environnements et processeur que pour l'expérience de l'image ban31q21 (IV-10B), nous pouvons, pour f et K_T constants évaluer la durée d'exécution en fonction de N. La courbe est présentée figure IV-25.

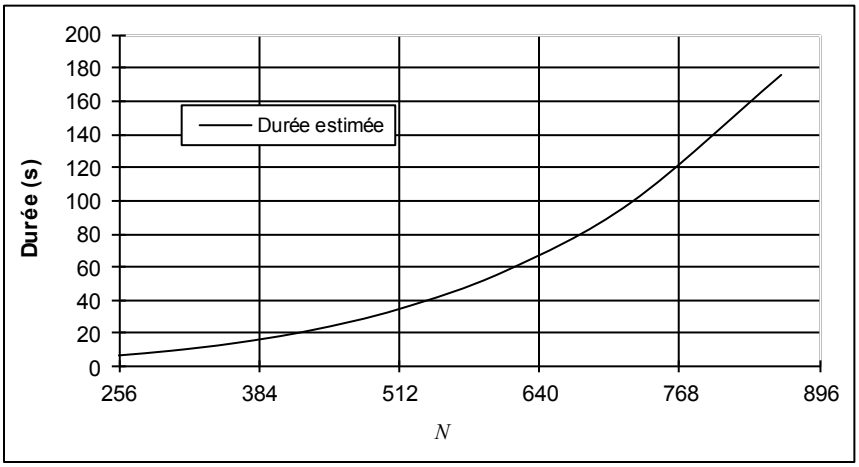


FIG. IV-25 - *estimation de la durée d'exécution en fonction de N (N>256)*

Une extrapolation au-delà donne une croissance bicarrée (pour $N \rightarrow +\infty$).

En effet, le traitement des lignes, qui représente la plus grande part du calcul croît comme N^4 ou f^4, c'est à dire respectivement le carré de la surface de l'image ou le carré du nombre de cellules du symbole. Cette estimation peut-être considérée comme la borne supérieure de la croissance de la quantité de calculs par rapport à la taille de l'image.

Viennent ensuite les traitements de contours et la lecture qui croissent comme la surface de l'image, et les calculs de texture qui croissent comme N^2 et K_T (indépendants de f).

Si on utilise des opérateurs câblés pour réaliser certaines étapes élémentaires telles que la détection de contour, on pourra réduire la durée d'exécution totale. Si nous nous situons dans un cas extrême où nous avons une accélération d'un facteur 10^3 sur ces parties de l'algorithme (`Gr`, `La`, `CCx`), nous pouvons faire une estimation de temps d'exécution pour le programme implémenté de cette manière (dite *estimation câblée*) et la comparer avec les durées établies précédemment (dites *estimation non câblée*). La figure IV-26 présente cette comparaison et indique la réduction (différence des durées) et le gain (différence des durées rapportée à l'estimation câblée) obtenus.

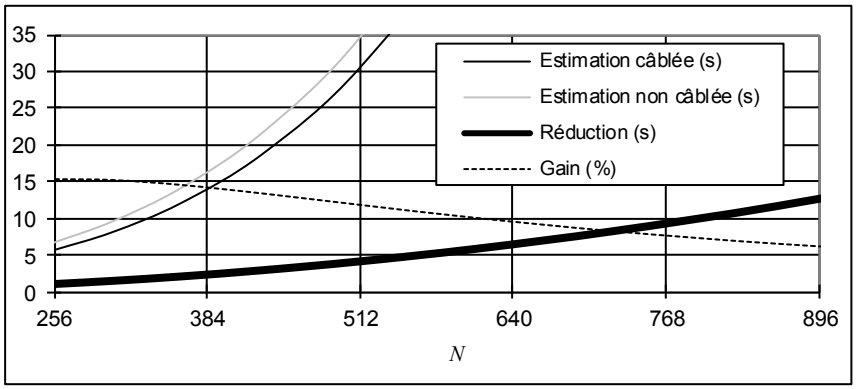


FIG. IV-26 - *estimations câblées et non câblées de la durée d'exécution (avec N croissant)*

Les réductions obtenues sont en proportion moins importantes lorsque N augmente. Ceci est dû au fait que ces réductions affectent les procédures de l'algorithme qui croissent en N^2, alors que la croissance globale est en N^4.

Si nous désirons faire fonctionner la méthode pour de grandes valeurs de N, le câblage des opérateurs élémentaires apportera un faible gain en temps d'exécution. C'est la partie du programme dédiée à la restauration de lignes qui consomme la plus grande part de calcul et qu'il faudra donc optimiser.

IV.D.Conclusion

Après avoir présenté l'environnement de programmation, nous avons vu la façon dont les paramètres internes avaient été choisis, par groupe ou isolément, chacun ayant une importance plus ou moins grande dans la qualité du résultat.

La décomposition de la méthode en étapes critiques permet de mieux comprendre comment les erreurs apparaissent et se propagent. Les caractéristiques de l'image qui vont déterminer le champ d'application de la méthode sont la période spatiale, le contraste, le bruit, l'hétérogénéité du blanc, les distorsions projectives, la netteté.

Ces caractéristiques se combinent, et interviennent à divers niveaux de la méthode avec plus ou moins d'effets sur les résultats.

Les différentes étapes de la méthode consomment du temps de calcul de manière très différentes. L'étape de détection de ligne, dont en particulier la restauration de ligne, représente une grande partie du temps de calcul. C'est aussi la partie dont la complexité indique qu'elle est la plus sensible à la taille des images (en pixels).

Nous avons pu voir cependant que, si la quantité de calculs est inhérente à la méthode et à son implémentation, la durée de traitement dépend aussi de la vitesse de calcul, qui peut-être réduite avec le développement technologique et l'arrivée de processeurs plus performants.

Conclusion

Pendant cette étude, une partie du temps a été consacrée à une recherche bibliographique, et une partie au développement d'algorithmes et à leur programmation. De nombreux articles de revues et de congrès ont été analysés, avec comme objectif de trouver et de comprendre les méthodes en usage pour la solution de problèmes similaires. La base bibliographique réalisée a permis d'offrir diverses approches possibles.

Les méthodes qui semblaient pouvoir apporter quelque chose, en particulier si cet intérêt était confirmé par plusieurs auteurs, ont été programmées afin d'étudier leur efficacité. La confrontation des méthodes entre elles et avec les résultats qu'elles délivraient a permis peu à peu de constituer une méthode cohérente et complète combinant plusieurs techniques : détection de contour par le laplacien, localisation de contour par le gradient, validation de détection par analyse de texture.

Ces techniques ont pu couvrir tous les problèmes majeurs qui se sont posés lors de l'établissement des algorithmes de recherche et lecture de symbole.

La méthode adoptée suit dans son ensemble le schéma suivant :

- Détection de contours et assemblage de ces contours en chaînes puis en lignes.
- Recherche de couples de lignes ayant les propriétés du motif en *L* du symbole en utilisant les propriétés géométriques et les propriétés de texture de la zone délimitée.
- Analyse du bord du symbole pour identifier sa fréquence propre.
- Échantillonnage du symbole selon sa fréquence propre le long d'une grille établie à partir des zones d'horloge.

Il est difficile de concevoir une méthode capable de rechercher et de lire les symboles quelles que soient les conditions de présentation (incidence, période spatiale…) ou de qualité d'image (netteté, contraste…), mais il est important de pouvoir déterminer les limites de la méthode par rapport à ces conditions.

À l'intérieur de ces limites, la méthode fonctionne correctement et permet de trouver et lire les symboles dans l'image, avec parfois une erreur de lecture de l'ordre inférieure à 5 %. Lorsqu'une limite est atteinte, elle l'est par l'une ou l'autre partie du programme, entraînant en général l'échec du processus.

Nous pouvons mesurer partiellement les limites de la méthode en observant ses étapes critiques, et éventuellement les modifier en intervenant sur une de ces étapes. Intervenir sur cette étape signifie modifier ses paramètres internes (seuils, coefficients de critères) ou quelques détails du principe de fonctionnement. En général, intervenir de cette façon peut permettre de repousser cette limite aux dépens d'une autre. Par exemple, augmenter la résistance au bruit d'un détecteur de contours en modifiant son seuil de détection le rend moins sensible aux faibles intensités.

La compréhension des limites et du point du processus où elles sont atteintes est importante si l'on veut envisager de les repousser. Lorsqu'une limite est atteinte pour des raisons différentes simultanément par plusieurs composantes de la méthode, il devient difficile de la repousser, car il faut intervenir sur chacun des composants.

Nous avons notamment observé les effets de la focalisation, du bruit, du contraste, des distorsions, de la période spatiale et de l'hétérogénéité du blanc.

- Focalisation : les étapes de détection de lignes, de coins et d'analyse de texture sont basées sur les contours. Il est donc nécessaire de ne pas avoir une image trop floue.
- Bruit : une image bruitée provoque du bruit sur la carte de contours et l'ensemble de lignes. Jusqu'à un certain niveau, le bruit est sans effet dans les étapes ultérieures. Le paramètre qui limite la méthode est le rapport signal-sur-bruit, qui dépend du contraste et de l'amplitude du bruit.
- Contraste : si l'image est peu contrastée, le rapport signal-sur-bruit sera plus faible, ce qui peut entraîner les problèmes de détection de lignes évoqués plus haut. Un faible contraste peut aussi entraîner des erreurs de lecture.
- Distorsion : les distorsions peuvent déformer un symbole au point qu'il ne soit pas détecté ou mal localisé, mais aussi déformer sa structure matricielle et la rendre non lisible.
- Période spatiale faible : pour plusieurs raisons (taille des texels, masques de contours utilisés, méthode d'identification de fréquence, détection de coins…), la méthode ne trouve et ne lit que les symboles de l'image dont la période spatiale est au-dessus de 3 pixels.
- Hétérogénéité du blanc : en fonction de la nature de la source lumineuse et des positions relatives de cette source, du support du symbole et du capteur, le blanc de l'image présente une certaine hétérogénéité. Cette hétérogénéité n'est pas gênante pour la recherche de symbole, mais peut provoquer des erreurs de lecture.

Les points critiques de la méthode sont les éléments de l'algorithme qui atteignent les limites de la méthode les premiers. Ces éléments doivent être surveillés lors de l'étude du processus, et éventuellement modifiés. Dans notre méthode, les deux points critiques sont les suivants :

- La détection de lignes qui conditionne la robustesse de la méthode vis-à-vis du bruit et de la focalisation. Il semble difficile de repousser significativement les limites sur ce point. Des améliorations futures consisteraient éventuellement à réaliser une recherche de contours suivant plusieurs détecteurs de sensibilité différente et de trouver un moyen de construire les lignes à partir des réponses conjuguées de ces deux détecteurs, en conservant la sensibilité de l'un et la robustesse de l'autre.
- L'identification de la fréquence. C'est un problème complexe si l'on veut tenir compte de tous les cas : localisation peu précise, irrégularité de la zone d'horloge suite à une distorsion, bruit provoquant des transitions parasites. Du développement peut encore être fait sur ce sujet. Cependant, les méthodes que l'on pourra développer seront, plus que pour le reste, dépendantes des spécifications définitives du code Data Matrix, notamment de la présence éventuelle de zones d'horloge à l'intérieur du symbole, de l'existence de symboles non carrés ou de fréquences paires.

Les principales causes d'échec peuvent cependant être supprimées par des solutions autres que logicielles, liées au dispositif de lecture et à son usage.

L'utilisation d'un objectif autofocus peut être envisagée, afin de garantir d'avoir toujours une image nette. Cette netteté peut aussi être obtenue si l'on augmente la profondeur de champ, ce qui peut-être réalisé par la réduction de l'ouverture du diaphragme. En contrepartie, ceci entraîne un rapport signal-sur-bruit plus faible, et réduit la fréquence spatiale limite (notons que la FTM d'un objectif est optimale pour une ouverture moyenne).

Le rapport signal-sur-bruit à son tour peut-être augmenté si nous augmentons l'éclairement du support du symbole. Il est prévu que le dispositif de lecture dispose de sa propre source lumineuse, dont la disposition et la puissance peuvent être choisies pour obtenir un éclairement homogène et un bon contraste dans la profondeur de champ de l'objectif. Ces éléments définissent un intervalle de distance entre le lecteur et le symbole pour lequel le dispositif de lecture fonctionne correctement.

Tout ceci suppose que le symbole est lui même correctement imprimé sur son support : à la fois net et contrasté. Le contraste intrinsèque du symbole sur son support dépend de la qualité de l'impression (quantité d'encre), mais aussi de la couleur du support.

Le logiciel réalisé permet d'exécuter l'ensemble du processus de recherche puis la lecture de symbole. Afin de pouvoir analyser le traitement dans les cas limites cités ci-dessus, l'utilisateur peut faire exécuter le processus pas à pas, en suivant chaque étape de la méthode. Il peut analyser les résultats intermédiaires et modifier les paramètres internes.

Ceci est possible car le logiciel a été conçu de manière modulaire, rendant les étapes indépendantes. Cette modularité permet au programmeur de modifier l'enchaînement des tâches, de remplacer des composantes, d'assurer une maintenance et une mise à jour plus sûre et plus rapide. La contrepartie est que le programme peut difficilement être optimisé de manière globale, en particulier du point de vue de la rapidité d'exécution. Les optimisations du code au niveau local n'ont pas été privilégiées non plus, car elles exigent une réécriture du code sous une forme moins lisible, ce qui est une source d'erreurs de programmation.

Cette étude a permis de déterminer comment mettre en œuvre un dispositif de recherche et lecture de symbole Data Matrix, à partir des diverses techniques connues de traitement d'image. Cependant, toutes les méthodes possibles n'ont pas été envisagées, ou n'ont pas été complètement exploitées. Parmi les méthodes susceptibles de compléter l'étude présente, on peut évoquer le seuillage d'image selon la luminance, qui permettrait, lorsque le contraste est suffisant, de la même façon que la méthode par la texture, d'apporter des informations sur la détection et localisation de symbole lorsque les autres méthodes échouent, notamment dans des conditions de flou ou de bruit au-delà des limites de notre méthode.

Mais la suite logique de cette étude est de transposer le logiciel dans un dispositif de lecture dédié à ce processus, après avoir éventuellement reporté un certain nombre de charges de calcul vers des couches câblées à la sortie du système d'acquisition d'image (opérateurs de gradient et laplacien notamment).

Bibliographie

[AIMT95] *Uniform symbology specification-Data Matrix - Draft*, AIM USA Technical Specification, AIM®USA Technology Group, TC-TSC/96-006, 12 décembre 1995.

[BALL81] D. H. BALLARD, « Generalizing the Hough transform to detect arbitrary shapes », *Pattern Recognition*, Vol. 13, n°2, 1981, pp. 111-122.

[BASU94] Mitra BASU, « Gaussian derivative model for edge enhancement », *Pattern Recognition*, Vol. 27, n°11, 1994, pp. 1451-1461.

[BENQ93] L. BENQUET, « Extraction automatique des aéroports dans les images du satellite SPOT », 14e colloque GRETSI sur le traitement du signal et des images, Juan-les-Pins, 13-16 septembre 1993, pp. 795-798.

[BENT94] M. Hafed BENTEFTIFA et Ludwik KURZ, « Feature detection via linear contrast techniques », *Pattern Recognition*, Vol. 26, n°10, 1993, pp. 1487-1497.

[BERZ84] Valdis BERZINS, « Accuracy of laplacian edge detectors », *CVGIP*, Vol. 27, 1984, pp. 195-210.

[BOUR93] E. BOURENNANE, M PAINDAVOINE et F. TRUCHETET, « Amélioration du filtre de Canny Deriche pour la détection de contours sous forme de rampe », *Traitement du signal*, Vol. 10, n°4, octobre 1993, pp. 297-310.

[BURN86] J. Brian BURNS, Allen. R. HANSOM et Edward M. RISEMAN, « Extracting straight lines », *IEEE TPAMI*, Vol. 8, n°4, juillet 1986, pp. 425-455.

[CANN94] John CANNING, « A minimum description length model for recognizing objects with variable appearances (the VAPOR model) », *IEEE TPAMI*, Vol. 16, n°10, octobre 1994, pp. 1032-1036.

[CANN86] John F. CANNY, « A computational approach to edge detection », *IEEE TPAMI*, Vol. 8, n°6, 1986, pp. 679-698.

[CASA92] A. CASALS, J. AMAT et A. GRAU, « Texture parametrization method for image segmentation », Second European Conference on Computer Vision, Santa Margherita Ligure, Italie, 1992, pp. 160-164.

[CHAN91] Donald G. CHANDLER, *Bar code reader,* European Patent 91302778.5, 1991.

[CHEN87] J. S. CHEN, A. HUERTAS et G. MEDIONI, « Fast convolution with laplacian-of-gaussian masks », *IEEE TPAMI*, Vol. 9, n°4, juillet 1987, pp. 584-590.

[CHEN91] M. CHEN, D. LEE et T. PAVLIDIS, « Residual analysis for feature detection », *IEEE TPAMI*, Vol. 13, n°1, janvier 1991, pp. 30-40.

[CHEN79] Patrick C. CHEN et Theodosios PAVLIDIS, « Segmentation by texture using a co-occurence matrix and a split-and-merge algorithm », *CGIP*, Vol. 10, 1979, pp. 172-182.

[CHER93] Hocine CHERIFI, Marc-Aurèle NJONKOU FANKAM et Richard LECORDIER, « Sur la détection optimale des contours par filtrage linéaire », 14e colloque GRETSI sur le traitement du signal et des images, Juan-les-Pins, 13-16 septembre 1993, pp. 683-686.

[CHUA93] E-Ren CHUANG et David SHER, « χ^2 test for feature detection », *Pattern Recognition*, Vol. 26, n°11, 1993, pp. 1673-1681.

[COCQ95] J.-P. COCQUEREZ et S. PHILIPP, *Analyse d'image : filtrage et segmentation, Masson, 1995.*

[DAVI75] Larry S. DAVIS, « A survey of edge detection techniques », *CGIP*, Vol. 4, 1975, pp. 248-270.

[DEFÉ91] Ireneusz DEFÉE et Yrjö NEUVO, « Median-based zero-crossing edge detectors for closely spaced edges », *CVGIP : graphical models and image processing*, Vol. 53, n°2, mars 1991, pp. 196-203.

[DELE96] Sébastien DELESTAING, *Détection et localisation de codes bi-dimensionnels par analyse de texture*, rapport DEA-SIC, GTTSI (LEN7), ENSEEIHT-INPT, Toulouse, 9 septembre 1996

[DERI87] Rachid DERICHE, « Détection optimale de contour avec une mise en oeuvre récursive », GRETSI-Nice, juin 1987, pp. 483-486.

[DEVI93] F. DEVILLARD, G. BOUVIER et A. CHEHIKIAN, « Algorithme d'approximation polygonale dédié à la robotique mobile », 14e colloque GRETSI sur le traitement du signal et des images, Juan-les-Pins, 13-16 septembre 1993, pp. 695-698.

[DUDA72] Richard O. DUDA et Peter E. HART, « Use of the Hough Transformation to detect lines and curves in pictures », *Communication of the ACM*, *CGIP*, Vol. 15, n°1, janvier 1972, pp. 11-15.

[FLEC92] Margaret M. FLECK, « Some defects in finite-difference edge finders », *IEEE TPAMI*, Vol. 14, n°3, mars 1992, pp. 337-345.

[FORE94] Gianluca FORESTI, Vittorio MURINO, Carlo S. REGAZZONI et Gianni VERNAZZA, « Grouping of rectilinear segments by the labeled Hough transform », *CVGIP : Image Understanding*, Vol. 58, n°3, novembre 1994, pp. 22-42.

[GUPT93] Arbind K. GUPTA, Santanu CHAUDHURY et Guturu PARTHASARATHY, « A new approach for aggregating edge points into line segments », *Pattern Recognition*, Vol. 26, n°7, 1993, pp. 1069-1086.

[HARA84] Robert. M. HARALICK, « Digital step edges from zero crossing of second directional derivatives », *IEEE TPAMI*, Vol. 6, n°1, janvier 1984, pp. 58-68.

[HARM93] Craig K. HARMON, « Cures for the common code », IDSystems, avril 1993, pp. 24-28.

[HIGG94] William E. HIGGINS et Chaoming HSU, « Edge detection using two-dimensional local structure information », *Pattern Recognition*, Vol. 27, n°2, 1994, pp. 277-294.

[HUER86] Andres HUERTAS et Gerard MEDIONI, « Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks », *IEEE TPAMI*, Vol. 8, n°5, septembre 1986, pp. 651-664.

[HUNT88] Douglas J. HUNT, Loren W. NOLTE et W. Howard RUEDGER, « Performance of the Hough transform and its relationship to statistical signal detection theory », *CVGIP*, Vol. 43, 1988, pp. 221-238.

[ILLI88] J. ILLINGWORTH et J. KITTLER « A survey on the Hough transform », *CVGIP*, Vol. 44, 1988, pp. 87-116.

[COCQ95] J.-P. COCQUEREZ et S. PHILIPP, *Analyse d'image : filtrage et segmentation, Masson, 1995.*

[KIRK95] David KIRK, Paul HECKBERT et Alan PAETH, *Graphics Gems I*, Andrew Glassner editor, «Graphics Gems vol. 1-5», ver. 1.0, 6 mars 1995, `ftp:// princeton. edu/ pub/ Graphics/ GraphicsGems/` et `http://www.geom.umn.edu/software/cglist/libs.html`.

[KITT86] K. KITTLER et J. ILLINGWORTH, « Minimum error thresholding », *Pattern Recognition*, Vol. 19, n°1, 1986, pp. 41-47.

[LAIN93] Andrew LAINE, Sergio SCHULER et V. GIRISH, « Orthogonal wavelet representations for recognizing complex annotations », *Machine vision and Applications*, Vol. 6, 1993, pp. 110-123.

[LASS96] Patricia LASSERRE, *Vision pour la robotique en environnement naturel*, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, thèse de doctorat n°2428, Toulouse, 26 septembre 1996.

[LEAV93] V. F. LEAVERS, « Which Hough transform ? », *CVGIP : Image Understanding*, Vol. 58, n°2, septembre 1993, pp. 250-264.

[LEEK95] Chung-Mong LEE et Atreyi KANKANHALLI, « Automatic extraction of characters in complex scene images », *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 9, n°1, 1995, pp. 67-82.

[LEES93] Jiann-Shu LEE, Yung-Nien SUN, Chin-Hsing CHEN et Ching-Tsorng TSAI, « Wavelet based corner detection », *Pattern Recognition*, Vol. 26, n°6, 1993, pp. 853-865.

[LIEC90] Wen-Nung LIE et Yung-Chang CHEN, « Robust line-drawing extraction for polyhedra using weighted polarized Hough transform », *Pattern Recognition*, Vol. 23, n°3/4, 1990, pp. 161-174.

[MANS87] Abdol-Reza MANSOURI, Alfred S. MALLOWANY et Martin D. LEVINE, « Line detection in digital pictures : a hypothesis prediction / verification paradigm », *CVGIP*, Vol. 40, 1987, pp. 95-114.

[MARC97] Baptiste MARCEL et Michel CATTOËN, « Edge and line detection in low level analysis », Third Workshop on Electronic Control and Measuring Systems, Université Paul Sabatier, Toulouse, 2-3 juin 1997, pp. 89-97

[MARR80] D. MARR et E. HILDRETH , « Theory of edge detection », *Proc. Royal Society of London*, 1980, pp.187-217.

[MEIS90] Amnon MEISELS et Doron MINTZ, « Symbolic reasoning in object extraction », *CVGIP*, Vol. 52, 1990, pp. 447-459.

[NALW86] Vishvjit S. NALWA et Thomas O. BINFORD, « On detecting edges », *IEEE TPAMI*, Vol. 8, n°6, novembre 1986, pp. 699-714.

[NALW87] Vishvjit S. NALWA et Eric PAUCHON, « Edgel aggregation and edge description », *CVGIP*, Vol. 40, 1987, pp. 79-94.

[NELS94] Randal C. NELSON, « Finding line segments by stick growing », *IEEE TPAMI*, Vol. 16, n°5, mai 1994, pp. 519-523.

[NEVA80] Ramakant V. NEVATIA et K. Ramesh BABU, « Linear feature extraction and description », *CGIP*, vol. 13, 1980, pp. 257-269.

[OLIV94] Jean-Christophe OLIVO, « Automatic threshold selection using the wavelet transform », *CVGIP : graphical models and image processing*, Vol. 56, n°3, mai 1994, pp. 205-218.

[ORTE89] Jean-José ORTEU, *Segmentation d'image en région par la méthode de séparation - fusion (*split and merge*)*, LAAS du CNRS, rapport technique n°89127, Toulouse, avril 1989.

[ORTE91] Jean-José ORTEU, *Application de la vision par ordinateur à l'automatisation de l'abattage dans les mines*, Université Paul Sabatier, thèse de doctorat n°1044, Toulouse, novembre 1991.

[PALP93] Nikhil R. PAL et Sankar K. PAL, « A review on image segmentation techniques », *Pattern Recognition*, Vol. 26, n°9, 1993, pp. 1277-1294.

[PALM93] P. L. PALMER, M. PETROU et J. KITTLER, « A Hough transform algorithm with a 2D hypothesis testing kernel », *CVGIP : Image Understanding*, Vol. 58, n°2, septembre 1993, pp. 221-234.

[PARK94] Deok J. PARK, Kwon M. NAM et Rae-Hong PARK, « Edge detection in noisy images based on the co-occurence matrix », *Pattern Recognition*, Vol. 27, n°6, 1994, pp. 765-775.

[PARK95] Deok J. PARK, Kwon M. NAM et Rae-Hong PARK, « Multiresolution edge detection techniques », *Pattern Recognition*, Vol. 28, n°2, 1995, pp. 1533-1537.

[PEIH94] Soo-Chang PEI et Ji-Hwei HORNG, « Corner point detection using nest moving average », *Pattern Recognition*, Vol. 27, n°11, 1994, pp. 1533-1537.

[PELI82] Tamar PELI et David MALAH, « A study of edge detection », *CGIP*, Vol. 20, 1982, pp. 1-21.

[RIYA97] RIYANTO, *Simulation, optimisation, et analyse de performances pour les systèmes industriels d'acquisition d'images*, Institut National Polytechnique de Toulouse, thèse de doctorat, Toulouse, 5 décembre 1997.

[ROSE74] Azriel ROSENFELD, « Digital straight line segments », *IEEE Transaction on Computers*, Vol. C-23, n°12, décembre 1974, pp. 1264-1269.

[ROSE71] Azriel ROSENFELD et Mark THURSTON, « Edge and curve detection for visual scene analysis », *IEEE Transaction on Computers*, Vol. C-20, n°5, mai 1971, pp. 562-569.

[ROTH93] Gerhard ROTH et Martin D. LEVINE, « Extracting geometric primitive », *CVGIP : Image Understanding*, Vol. 58, n°1, juillet. 1993, pp. 1-22.

[ROTH94] Gerhard ROTH et Martin D. LEVINE, « Geometric primitive extraction using a genetic algorithm », *IEEE TPAMI*, Vol. 16, n°9, septembre 1994, pp. 901-905.

[SARK91] Sudeep SARKAR et Kim L. BOYER, « On optimal IIR edge detection filters », *IEEE TPAMI*, Vol. 13, n°11, novembre 1991, pp. 1154-1171.

[SOTA89] G. E. SOTAK, JR et K. L. BOYER, « The Laplacian-Of-Gaussian Kernel : a formal analysis and design procedure for fast, accurate convolution and full-frame output », *CVGIP*, Vol. 48, 1989, pp. 147-189.

[SUET92] Paul SUETENS, Pascal FUA et Andrew J. HANSON, « Computational strategies for object recognition », *ACM Computing surveys*, Vol. 24, n°1, mars 1992, pp. 5-61.

[TANL93] C. L. TAN et S. K. K. LOH, « Efficient edge detection using hierarchical structures », *Pattern Recognition*, Vol. 26, n°1, 1993, pp. 127-135.

[TORR86] Vincent TORRE et Tomaso A. POGGIO, « On edge detection », *IEEE TPAMI*, Vol. 8, n°2, mars 1986, pp. 147-163.

[TRIV90] Mohan M. TRIVEDI et Chu Xin CHEN, « Object detection by step-wise analysis of spectral, spatial, and topographic features », *CVGIP*, Vol. 51, 1990, pp. 235-255.

[VAIL94] R. VAILLANT, C. MONROCQ et Y. LE CUN, « Original approach for localisation of objects in images », *IEE Proceedings on Vision and Image Signal Processing,* Vol. 141, n°4, août 1994, pp. 145-250.

[WALL84] Karin WALL et Per-Erik DANIELSSON, « A fast sequential method for polygonal approximation of digitized curves », *CVGIP*, Vol. 28, 1984, pp. 220-227.

[WANG94] Mao-Jiun J. WANG, Shiau-Chyi CHANG, Chih-Minh LIU et Wen-Yen WU, « A new edge detection method through template matching », *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 8, n°4, 1994, pp. 899-917.

[WHIT83] J. M. WHITE et G. D. ROHRER, « Image thresholding for optical character recognition and other applications requiring character image extraction », *IBM J. res develop.*, Vol. 27, n°4, juillet 1983, pp. 400-411.

[ZHOU89] Y. T. ZHOU, V. VENKATSEWAR et R. CHELLAPPA, « Edge detection and linear feature extraction using a 2-D random field model », *IEEE TPAMI*, Vol.11, n°9, janvier 1989, pp. 84-95.

[ZIOU93] Djemel ZIOU et Salvatore TABBONE, « Corner detection by a cost minimization approach », *Pattern Recognition*, Vol. 26, n°9, 1993, pp. 1305-1314.

Abréviations :

CVGIP	Computer Vision, Graphics, and Image Processing
CGIP	Computer Graphics and Image Processing
GRETSI	Groupe d'Étude et de Traitement du Signal et des Images
IEEE	Institute of Electrical and Electronics Engineers
TPAMI	Transaction on Pattern Analysis and Machine Intelligence

Annexes

Annexe A - Lexique

- **Antiparallèle**

Se dit de deux contours, ou lignes, parallèles, ayant des sens différents (le sens d'un contour est déterminé par convention à partir du sens de la transition de luminance).

- **Carte** (***map***)

Une carte est une fonction similaire à une image numérique contenant pour chaque pixel un certain nombre de propriétés. Une carte de contours contient l'intensité et/ou la direction des contours. Une carte de contours binaires contient l'absence ou la présence de contour (obtenue par seuil par exemple).

- **Chaîne**

Dans une image, une chaîne est un groupe connexe de pixels de contour.

- **Coin**

Dans une image, un coin est un point d'une chaîne où celle-ci change brusquement de direction, ou bien (selon les auteurs et les applications) est coupée par une autre chaîne.

- **Contour** (***edge***)

Un contour est un endroit où l'image présente une discontinuité importante en luminance. L'amplitude de cette discontinuité est l'amplitude du contour. Un pixel de contour est un pixel sur lequel l'image présente un contour.

- **Image**

Fonction de $\mathbb{R}^2$ dans $\mathbb{R}$, où l'ensemble de départ $\mathbb{R}^2$ représente la projection d'une scène, et l'ensemble d'arrivée $\mathbb{R}$ la luminance en chaque point de la projection. Cette fonction est en général restreinte à une surface du plan, généralement rectangulaire, appelée *support de l'image*. Lorsqu'une image est numérique, c'est une fonction de M×N dans D où M et N sont des intervalles de $\mathbb{N}$ et D un intervalle de $\mathbb{Q}$ (ensemble des rationnels).

- **Ligne**

Dans une image, une ligne est une chaîne où les pixels sont alignés.

- **Propriété gage de contour**

Propriétés que remplissent un point d'inflexion et son voisinage lorsqu'ils sont situés sur un contour.

- **Texel**

Groupe de pixels de l'image qui sert de base aux caractérisations de texture.

- **Texture**

La *texture* d'une image (ou d'une partie de l'image) est un ensemble de caractéristiques de l'image invariantes par translation d'une fenêtre d'observation de taille donnée. L'espace des textures est divisé en classes d'équivalence.

L'identification des textures se fait en utilisant des *critères* permettant d'attribuer à chaque zone telle ou telle classe de texture.

Annexe B - Modélisation de la distorsion projective pour la lecture de symbole

B.1. Présentation du modèle

Selon le modèle sténopé, les distorsions projectives se modélisent par la projection de l'espace (la scène) à travers un diaphragme ponctuel sur un plan (plan image). Soient P cette projection, P le plan de projection, O le diaphragme situé en dehors de P (O est le *centre de projection*). Les points du plan parallèle à P passant par O ne sont pas projetables, nous appelons P_O ce plan. Soient deux points de l'espace A et B hors de P_O et leur projetés A' et B'. On munit l'espace d'un repère orthonormé R_0 de centre O, de directions $\vec{e}_x$, $\vec{e}_y$ et $\vec{e}_z$ telles que $\vec{e}_z \perp P$. Soit z_0 l'abscisse sur $(O, \vec{e}_z)$ de P. On choisit $\vec{e}_z$ tel que $z_0=1$, et la direction de $\vec{e}_x$ telle que $\overrightarrow{A'B'}$ et $\vec{e}_y$ ne sont pas colinéaires (nous verrons la nécessité de cette contrainte plus loin).

Soit $M\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ un point de l'espace n'appartenant pas à P_O et $M'\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$ l'image de M dans la projection. Nous avons par définition :

$$M'\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathsf{P}\left(M\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right) \Leftrightarrow \left\{ M' \in (OM) \text{ et } M' \in P \right\} \quad \text{(B-1)}$$

$\overrightarrow{OM}$ et $\overrightarrow{OM'}$ sont colinéaires. Pour tout M, il existe un coefficient k tel $\overrightarrow{OM'} = k\,\overrightarrow{OM}$. Nous avons $M' \in P$, et donc $z' = z_0$. Ceci permet de calculer $k = {}^{z_0}\!/_{z}$ et de déterminer les coordonnées de M'.

$$\overrightarrow{OM}\begin{pmatrix} x \\ y \\ z \end{pmatrix},\ \overrightarrow{OM'}\begin{pmatrix} x' = x\,{}^{z_0}\!/_{z} \\ y' = y\,{}^{z_0}\!/_{z} \\ z' = z_0 \end{pmatrix},\ \text{et } k = {}^{z_0}\!/_{z} \quad \text{(B-2)}$$

B.2. Déformation d'un axe gradué

Afin d'étudier *in fine* la forme que prend un symbole lorsqu'il est projeté, on étudie la déformation que subit l'axe gradué $(A, \overrightarrow{AB})$.

A et B ont les coordonnées suivantes : $A\begin{pmatrix} x_A \\ y_A \\ z_A \end{pmatrix}$ et $B\begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix}$, on prend le point M hors de P_O et variable sur (AB). Soit w l'abscisse de M sur $(A, \overrightarrow{AB})$ (c'est à dire tel que $\overrightarrow{AM} = w\,\overrightarrow{AB}$). Pour cela, nous calculons les coordonnées de M' par rapport à celles de A' et de B'. Nous calculons les coordonnées suivantes (rappelons que $z_0=1$) :

$$A'\begin{pmatrix} x'_A = \frac{x_A}{z_A} \\ y'_A = \frac{y_A}{z_A} \\ z'_A = 1 \end{pmatrix},\ B'\begin{pmatrix} x'_B = \frac{x_B}{z_B} \\ y'_B = \frac{y_B}{z_B} \\ z'_B = 1 \end{pmatrix} \text{ et } \overrightarrow{A'B'}\begin{pmatrix} \frac{x_B}{z_B} - \frac{x_A}{z_A} \\ \frac{y_B}{z_B} - \frac{y_A}{z_A} \\ 0 \end{pmatrix} \text{ (B-3) ;}$$

$$M\begin{pmatrix} x = x_A + w(x_B - x_A) \\ y = y_A + w(y_B - y_A) \\ z = z_A + w(z_B - z_A) \end{pmatrix},\ k = \frac{1}{z_A + w(z_B - z_A)} \text{ et } M'\begin{pmatrix} x' = \dfrac{x_A + w(x_B - x_A)}{z_A + w(z_B - z_A)} \\ y' = \dfrac{y_A + w(y_B - y_A)}{z_A + w(z_B - z_A)} \\ z' = 1 \end{pmatrix} \text{ (B-4) ;}$$

$$A'M'\begin{pmatrix} \dfrac{x_A + w(x_B - x_A)}{z_A + w(z_B - z_A)} - \dfrac{x_A}{z_A} \\ \dfrac{y_A + w(y_B - y_A)}{z_A + w(z_B - z_A)} - \dfrac{y_A}{z_A} \\ 0 \end{pmatrix} \text{ (B-5)}$$

Nous pouvons écrire l'équation B-4 car M n'appartient pas à P_O (et donc $z \neq 0$ et $z_A + w(z_B - z_A) \neq 0$). Lors d'une projection comme celle-ci, il est évident que, puisque $M \in (AB)$, alors $M' \in (A'B')$, et de cette manière l'axe $(A, \overrightarrow{AB})$ a pour image l'axe $(A', \overrightarrow{A'B'})$. Nous allons cependant le démontrer en montrant qu'il existe un réel w' unique tel que $\overrightarrow{A'M'} = w' \overrightarrow{A'B'}$.

Utilisons la première coordonnée de $\overrightarrow{A'M'}$ pour calculer w' s'il existe :

$$\frac{x_A + w(x_B - x_A)}{z_A + w(z_B - z_A)} - \frac{x_A}{z_A} = w'\left(\frac{x_B}{z_B} - \frac{x_A}{z_A}\right) \Leftrightarrow w' = \frac{\dfrac{x_A + w(x_B - x_A)}{z_A + w(z_B - z_A)} - \dfrac{x_A}{z_A}}{\dfrac{x_B}{z_B} - \dfrac{x_A}{z_A}} \text{ (B-6) ;}$$

$$
\begin{aligned}
w' &= \frac{\frac{x_A + w(x_B - x_A)}{z_A + w(z_B - z_A)} - \frac{x_A}{z_A}}{\frac{x_B}{z_B} - \frac{x_A}{z_A}} \\
&= \frac{x_A + w(x_B - x_A) - \frac{x_A}{z_A}\left(z_A + w(z_B - z_A)\right)}{\left(\frac{x_B}{z_B} - \frac{x_A}{z_A}\right)\left(z_A + w(z_B - z_A)\right)} \\
&= \frac{x_A z_A z_B + w z_A z_B (x_B - x_A) - x_A z_B \left(z_A + w(z_B - z_A)\right)}{\left(x_B z_A - x_A z_B\right)\left(z_A + w(z_B - z_A)\right)} \\
&= \frac{x_A z_A z_B + w z_A z_B x_B - w z_A z_B x_A - x_A z_B z_A - x_A z_B w(z_B - z_A)}{\left(x_B z_A - x_A z_B\right)\left(z_A + w(z_B - z_A)\right)} \\
&= \frac{w z_A z_B x_B - w z_A z_B x_A - x_A z_B w z_B + x_A z_B w z_A}{\left(x_B z_A - x_A z_B\right)\left(z_A + w(z_B - z_A)\right)} \\
&= \frac{w z_B \left(z_A x_B - x_A z_B\right)}{\left(x_B z_A - x_A z_B\right)\left(z_A + w(z_B - z_A)\right)} \\
&= \frac{w z_B}{z_A + w(z_B - z_A)}
\end{aligned}
$$

(B-7)

Nous pouvons écrire l'équation B-6 car $\overrightarrow{A'B'}$ et $\vec{e}_y$ ne sont pas colinéaires. La composante en $\vec{e}_x$ de ce vecteur ($\frac{x_B}{z_B} - \frac{x_A}{z_A}$) est donc non nulle. Le calcul de w' avec la seconde coordonnée de $\overrightarrow{A'M'}$ donnerait le même résultat. Nous avons donc bien $\overrightarrow{A'M'} = w' \overrightarrow{A'B'}$, et w' est l'abscisse de M' sur $(A', \overrightarrow{A'B'})$.

Dans le cas où (*AB*) est parallèle à *P*, nous avons une perspective cavalière, $z_A = z_B$ et $w' = w$. Il n'y a pas de distorsion sur l'axe $(A', \overrightarrow{A'B'})$.

Dans les autres cas, nous appelons $\boldsymbol{I}_{AB}$ le pôle de w' (l'abscisse du point d'intersection entre (*AB*) et P_O), et exprimons w' en fonction de $\boldsymbol{I}_{AB}$ et de w :

$$\boldsymbol{I}_{AB} = -\frac{z_A}{z_B - z_A}$$

$$w' = \frac{wz_B}{z_A + w(z_B - z_A)}$$

$$= \frac{w\dfrac{z_B}{z_B - z_A}}{\dfrac{z_A}{z_B - z_A} + w}$$

$$= w\frac{\boldsymbol{I}_{AB} - 1}{\boldsymbol{I}_{AB} - w}$$

(B-8)

Lorsque le plan de l'objet est incliné par rapport au plan image, les droites supports des bords du symbole convergent vers des points de fuite, et la taille des cellules dans l'image varie selon une fraction rationnelle dont on peut déterminer les coefficients par la détermination du point de fuite.

Nous voyons que les fonctions $w'(w)$ et $w'_w(\boldsymbol{I}_{AB})$ sont des fractions rationnelles, et que la limite de w' lorsque w tend vers l'infini est la valeur finie 1-$\boldsymbol{I}_{AB}$, qui est l'abscisse sur l'axe $(A', \overrightarrow{A'B'})$ du point de fuite de la droite (AB). On voit aussi que pour w donné, w' tend vers w lorsque l'on s'approche d'une projection cavalière (z_A-z_B tend vers zéro, $\boldsymbol{I}_{AB}$ tend vers l'infini).

$$\mathrm{w}'(w) \xrightarrow{w\to+\infty} 1 - \boldsymbol{I}_{AB} \text{ et } \mathrm{w}'_{\mathrm{w}}(\boldsymbol{I}_{AB}) \xrightarrow{z_A - z_B \to 0} w \quad \text{(B-9)}$$

B.3. Effet de la distorsion sténopé sur un symbole

Les deux droites supports des limites du symbole sont parallèles, et ont donc comme image deux droites convergeant vers leur point de fuite. L'image du symbole n'a donc pas la forme d'un parallélogramme, ni *a fortiori* d'un carré.

Par contre, la connaissance des quatre coins du symbole permet de localiser les points de fuite des droites supports, ce qui permet de reconstituer la grille support du symbole. Nous allons le montrer ci-dessous, et l'illustrer avec la figure B-1.

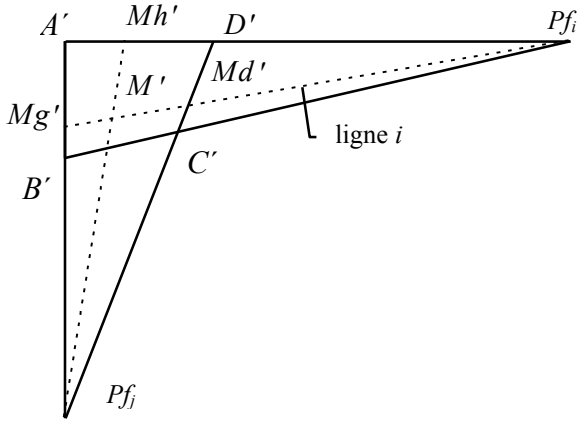


FIG. *B*-1 - *projection d'un quadrilatère. Le point M a pour coordonnées (2f/3,f/3)*

Soit *ABCD* le symbole à lire et $A'B'C'D'$ son image. Le symbole *ABCD* a une fréquence *f*, les coordonnées d'un point du symbole s'expriment sous la forme du couple (*numéro de ligne*, *numéro de colonne*), où une ligne est parallèle à (*AD*) et une colonne parallèle à (*AB*). On cherche M' l'image du point *M* de coordonnées (*i*,*j*) dans le symbole. Pour cela, on cherche par exemple la ligne *i* du symbole sur l'image.

Nous supposons que nous connaissons dans l'image la position du quadrilatère $A'B'C'D'$. Les intersections de $(A'B')$ et $(D'C')$ d'une part et de $(A'D')$ et $(B'C')$ d'autre part permettent de déterminer la position des points de fuite que nous appelons respectivement Pf_j et Pf_i.

D'après l'équation B-9, l'abscisse de Pf_j sur $(A', \overrightarrow{A'B'})$ vaut 1-$\boldsymbol{I_{AB}}$ et l'abscisse de Pf_i sur $(A', \overrightarrow{A'D'})$ vaut 1-$\boldsymbol{I_{AD}}$. La ligne *i* est parallèle aux bords du symbole et passe par le point $Mg(i,0)$ situé sur l'axe $(A, \overrightarrow{AB})$ à l'abscisse $w = i/f$. En utilisant la formule de l'équation B-8, connaissant $\boldsymbol{I_{AB}}$, on calcule *w'*, l'abscisse du point M_g' sur l'axe $(A', \overrightarrow{A'B'})$. La droite $(Pf_i\, Mg')$ est l'image de la ligne *i*.

$$\frac{\overline{A'Pf_j}}{\overline{A'B'}} = 1 - \boldsymbol{I_{AB}} \Leftrightarrow \boldsymbol{I_{AB}} = 1 - \frac{\overline{A'Pf_j}}{\overline{A'B'}} \quad \text{(B-10)}$$

$$\begin{aligned} \frac{\overline{A'Mg'}}{\overline{A'B'}} = i/f \frac{(\boldsymbol{I_{AB}} - 1)}{(\boldsymbol{I_{AB}} - i/f)} \quad &\Leftrightarrow \quad \overline{A'Mg'} = \overline{A'B'}\; i/f \frac{(\boldsymbol{I_{AB}} - 1)}{(\boldsymbol{I_{AB}} - i/f)} \\ &\Leftrightarrow \quad \overrightarrow{Mg'} = \overrightarrow{A'} + \overrightarrow{A'B'}\; i/f \frac{(\boldsymbol{I_{AB}} - 1)}{(\boldsymbol{I_{AB}} - i/f)} \end{aligned} \quad \text{(B-11)}$$

On appelle *Md* le point du symbole de coordonnées (*i*,*f*) et Md' son image. Md' se trouve à l'intersection de $(Pf_i Mg')$ et $(D'C')$ et est donc connu. Le point *M* est sur l'axe $(Mg, \overrightarrow{MgMd})$ à l'abscisse i/f et son image M' est sur l'axe $(Mg', \overrightarrow{Mg'Md'})$ à une abscisse que l'on peut calculer avec la formule de l'équation B-8, en utilisant $\boldsymbol{I}_{MgMd}$.

$$\frac{\overline{Mg'Pf_i}}{\overline{Mg'Md'}} = 1 - \boldsymbol{I_{MgMd}} \Leftrightarrow \boldsymbol{I_{MgMd}} = 1 - \frac{\overline{Mg'Pf_i}}{\overline{Mg'Md'}} \quad \text{(B-12)}$$

$$\begin{aligned} \frac{\overline{Mg'M'}}{\overline{Mg'Md'}} = i/f \frac{(\boldsymbol{I_{MgMd}} - 1)}{(\boldsymbol{I_{MgMd}} - i/f)} \quad &\Leftrightarrow \quad \overline{Mg'M'} = \overline{Mg'Md'}\; i/f \frac{(\boldsymbol{I_{MgMd}} - 1)}{(\boldsymbol{I_{MgMd}} - i/f)} \\ &\Leftrightarrow \quad \overrightarrow{M'} = \overrightarrow{Mg'} + \overrightarrow{Mg'Md'}\; i/f \frac{(\boldsymbol{I_{MgMd}} - 1)}{(\boldsymbol{I_{MgMd}} - i/f)} \end{aligned} \quad \text{(B-13)}$$

Pour le couple (*i*, *j*) donné de coordonnées du point *M*, nous avons donc pu déterminer la position de l'image M' de ce point par rapport aux coins du symbole image $A'B'C'D'$. Nous sommes donc en mesure, si la position du symbole est connue, de déterminer la grille d'échantillonnage à utiliser pour lire la valeur du symbole à partir de l'image.

Annexe C - Détermination des matrices de calcul du gradient et du laplacien

Nous avons vu dans le chapitre II quelques masques de convolution permettant de calculer le gradient et le laplacien. Ces masques sont représentés par des matrices. Nous allons étudier ici en détail la façon dont nous obtenons ces matrices.

C.1. Calculs des matrices

On note g l'image à analyser. g est défini sur une grille de points repérée par les indices (i, j).

Pour une plus grande clarté de l'exposé, on confondra la dérivée et son estimation.

On note ∇ l'opérateur de gradient et ∇^2 l'opérateur de laplacien.

$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \quad \text{(D-1)}$$

C.2. Gradient : dérivation du premier ordre

Soient f une fonction de $\mathbb{R}$, $\bar{f}$ la fonction f filtrée par un filtre passe-bas, ∂f et $\partial \bar{f}$ les dérivées de f et $\bar{f}$.

C.2.a. Dérivation à deux pixels

$$\partial f(i) = f(i+1) - f(i) \quad \text{(D-2)}$$

C.2.b. Dérivation à trois pixels

La dérivation à 3 pixels s'obtient en combinant une moyenne sur 2 pixels et la dérivation à 2 pixels.

$$\bar{f}(i) = f(i) + f(i+1) \quad \text{(D-3)}$$

$$\begin{aligned} \partial \bar{f}(i) &= \bar{f}(i+1) - \bar{f}(i) \\ &= f(i+1) + f(i+2) - [f(i) + f(i+1)] \\ &= f(i+2) - f(i) \end{aligned} \quad \text{(D-4)}$$

Si on décale la matrice de convolution pour la centrer autour du point de calcul, on obtient la matrice D-8 de la table D-1.

C.2.c. Matrices obtenues

Taille (pixels)	Matrice	Dérivation
1×2	$\begin{pmatrix}-1 & 1\end{pmatrix}$ ou $\begin{pmatrix}0 & -1 & 1\end{pmatrix}$ (D-5)	$\frac{\partial g}{\partial x} = g(i+1) - g(i)$ (D-6)
1×3	$\begin{pmatrix}-1 & 0 & 1\end{pmatrix}$ (D-7)	$\frac{\partial g}{\partial x} = g(i+1) - g(i-1)$ (D-8)

TABLE D-1 - *matrices de calcul du gradient*

C.3. Laplacien : dérivation du second ordre

C.3.a. Dérivation à deux pixels

À partir de l'équation D-2, nous écrivons :

$$\begin{aligned}\frac{\partial^2 g}{\partial x^2} &= \frac{\partial g}{\partial x}(i+1) - \frac{\partial g}{\partial x}(i) \\ &= g(i+2) - g(i+1) - [g(i+1) - g(i)] \quad \text{(D-9)} \\ &= g(i+2) - 2g(i+1) + g(i)\end{aligned}$$

Ceci peut s'exprimer par la matrice (1 -2 1). Le calcul de $\frac{\partial^2 g}{\partial y^2}$ mène à la matrice (1 -2 1)′. La somme de ces deux dérivations mène à la matrice suivante :

$$\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$$

C.3.b. Dérivation à trois pixels

À partir de l'équation D-8, nous écrivons :

$$\begin{aligned}\frac{\partial^2 g}{\partial x^2} &= \frac{\partial g}{\partial x}(i+1) - \frac{\partial g}{\partial x}(i-1) \\ &= g(i+2) - g(i) - [g(i) - g(i-1)] \quad \text{(D-10)} \\ &= g(i+2) - 2g(i) + g(i-1)\end{aligned}$$

Ceci peut s'exprimer par la matrice (1 0 -2 0 1). Le calcul de $\frac{\partial^2 g}{\partial y^2}$ mène à la matrice (1 0 -2 0 1)′. La somme de ces deux dérivations mène à la matrice suivante:

$$\begin{pmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{pmatrix}$$

C.3.c. Dérivation à deux pixels selon le voisinage à 8

Le laplacien au sens du voisinage à 8 se calcule de la manière suivante :

$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} + \frac{\partial^2 g}{\partial \delta_1{}^2} + \frac{\partial^2 g}{\partial \delta_2{}^2} \quad \text{(D-11)},$$

où $\frac{\partial^2 g}{\partial \delta_1{}^2}$ et $\frac{\partial^2 g}{\partial \delta_2{}^2}$ sont respectivement les dérivées de g dans les directions Δ_1 et Δ_2 que sont la première et la seconde diagonale. Nous notons $\frac{\partial^2 g}{\partial \Delta_1{}^2}$ et $\frac{\partial^2 g}{\partial \Delta_2{}^2}$ les dérivées similaires à $\frac{\partial^2 g}{\partial \delta_1{}^2}$ et $\frac{\partial^2 g}{\partial \delta_2{}^2}$, calculées selon le pas de dérivation pris sur la diagonale ($\sqrt{2}$ pixels au lieu de 1 pixel).

Nous utilisons les mêmes formules de dérivation :

$$g_{\Delta I} = g(i+1, j+1) - g(i, j) \quad \text{(D-12)}$$

$$\begin{aligned} \frac{\partial^2 g}{\partial \Delta_1{}^2} &= \frac{\partial g}{\partial \Delta_1}(i+1, j+1) - \frac{\partial g}{\partial \Delta_1}(i, j) \\ &= g(i+2, j+2) - g(i+1, j+1) - [g(i+1, j+1) - g(i, j)] \quad \text{(D-13)} \\ &= g(i+2, j+2) - 2g(i+1, j+1) + g(i, j)] \end{aligned}$$

Ceci peut s'exprimer par la matrice $\begin{pmatrix} & & 1 \\ & -2 & \\ 1 & & \end{pmatrix}$. Le calcul de $\frac{\partial^2 g}{\partial \Delta_2{}^2}$ mène à la matrice $\begin{pmatrix} 1 & & \\ & -2 & \\ & & 1 \end{pmatrix}$. La somme de ces deux dérivations mène à la matrice $\begin{pmatrix} 1 & & 1 \\ & -4 & \\ 1 & & 1 \end{pmatrix}$.

Le pas de cette dérivation vaut $\sqrt{2}$. On ramène les dérivations en diagonales à un pas de 1 pixel, on en fait la somme avec la dérivation droite et on obtient la matrice suivante :

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & & 1 \\ & -4 & \\ 1 & & 1 \end{pmatrix} + \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & -4-2\sqrt{2} & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \text{(D-14)}$$

C.3.d. Dérivation à trois pixels selon le voisinage à 8

$$\frac{\partial g}{\partial \Delta_1} = g(i+1, j+1) - g(i-1, j-1) \quad \text{(D-15)}$$

$$\begin{aligned} \frac{\partial^2 g}{\partial \Delta_1{}^2} &= \frac{\partial g}{\partial \Delta_1}(i+1, j+1) - \frac{\partial g}{\partial \Delta_1}(i-1, j-1) \\ &= g(i+2, j+2) - g(i, j) - [g(i, j) - g(i-2, j-2)] \quad \text{(D-16)} \\ &= g(i+2, j+2) - 2g(i, j) + g(i-2, j-2)] \end{aligned}$$

Ce qui peut s'exprimer par la matrice $\begin{pmatrix} & & 1 \\ & 0 & \\ 0 & & \\ 1 & & \end{pmatrix}$. Le calcul de $\frac{\partial g}{\partial \Delta_2}$ mène à la matrice $\begin{pmatrix} 1 & & & & \\ & 0 & & & \\ & & 2 & & \\ & & & 0 & \\ & & & & 1 \end{pmatrix}$. La somme de ces deux dérivations mène à la matrice $\begin{pmatrix} 1 & & & & 1 \\ & 0 & & 0 & \\ & & 4 & & \\ & 0 & & 0 & \\ 1 & & & & 1 \end{pmatrix}$.

Le pas de cette dérivation vaut √2. On ramène les dérivations en diagonales à un pas de 1 pixel, on en fait la somme avec la dérivation droite et on obtient la matrice D-27 :

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & & & & 1 \\ & 0 & & 0 & \\ & & 4 & & \\ & 0 & & 0 & \\ 1 & & & & 1 \end{pmatrix} + \begin{pmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & & 1 & & \frac{1}{\sqrt{2}} \\ & & 0 & & \\ 1 & 0 & -4-2\sqrt{2} & 0 & 1 \\ & & 0 & & \\ \frac{1}{\sqrt{2}} & & 1 & & \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 1 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -4-2\sqrt{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 1 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \text{(D-17)}$$

C.3.e. Dérivation à trois pixels selon le voisinage à 4 et avec moyenne

Le laplacien 5×5 obtenu équation D-17 n'est pas un laplacien précédé d'un filtrage de moyenne.

Si l'on fait précéder le laplacien 3×3 par un filtrage passe-bas 2×2 de type $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, nous obtenons un calcul de laplacien selon une matrice 4×4 comme démontré ci-dessous.

Soit g une fonction, $\bar{g}$ la fonction g filtrée, $\partial^2 g$ et $\partial^2 \bar{g}$ les laplaciens de g et $\bar{g}$.

$$\bar{g}(i,j) = g(i,j) + g(i+1,j) + g(i,j+1) + g(i+1,j+1) \quad \text{(D-18)}$$

$$\partial^2 g(i,j) = g(i+1,j) + g(i,j+1) + g(i-1,j) + g(i,j-1) - 4g(i,j) \quad \text{(D-19)}$$

$$\partial^2 \bar{g}(i,j) = \bar{g}(i+1,j) + \bar{g}(i,j+1) + \bar{g}(i-1,j) + \bar{g}(i,j-1) - 4\bar{g}(i,j) \quad \text{(D-20)}$$

Soient $M=(i,j)$, $e_i=(0,1)$, $e_j=(1,0)$, $M_d=M+e_j$, $M_h=M+e_j$, $M_{hd}=M+e_j+e_j$.

$$\bar{g}(i,j) = \bar{g}(M) = g(M) + g(M_d) + g(M_h) + g(M_{hd}) \quad \text{(D-21)}$$

$$\begin{aligned}
\partial^2 \bar{g}(M) &= \bar{g}(M+e_i) + \bar{g}(M+e_j) + \bar{g}(M-e_i) + \bar{g}(M-e_j) - 4\bar{g}(M) \\
&= \left\{\begin{matrix} g(M+e_i) \\ +g(M_d+e_i) \\ +g(M_h+e_i) \\ +g(M_{hd}+e_i) \end{matrix}\right\} + \left\{\begin{matrix} g(M+e_j) \\ +g(M_d+e_j) \\ +g(M_h+e_j) \\ +g(M_{hd}+e_j) \end{matrix}\right\} + \left\{\begin{matrix} g(M-e_i) \\ +g(M_d-e_i) \\ +g(M_h-e_i) \\ +g(M_{hd}-e_i) \end{matrix}\right\} + \left\{\begin{matrix} g(M-e_j) \\ +g(M_d-e_j) \\ +g(M_h-e_j) \\ +g(M_{hd}-e_j) \end{matrix}\right\} - 4\left\{\begin{matrix} g(M) \\ +g(M_d) \\ +g(M_h) \\ +g(M_{hd}) \end{matrix}\right\} \\
&= \left\{\begin{matrix} g(M+e_i)+g(M+e_j)+g(M-e_i)+g(M-e_j)-4g(M) \\ +g(M_d+e_i)+g(M_d+e_j)+g(M_d-e_i)+g(M_d-e_j)-4g(M_d) \\ +g(M_h+e_i)+g(M_h+e_j)+g(M_h-e_i)+g(M_h-e_j)-4g(M_h) \\ +g(M_{hd}+e_i)+g(M_{hd}+e_j)+g(M_{hd}-e_i)+g(M_{hd}-e_j)-4g(M_{hd}) \end{matrix}\right\} \\
&= \partial^2 g(M) + \partial^2 g(M_d) + \partial^2 g(M_h) + \partial^2 g(M_{hd}) \\
&= \overline{\partial^2 g}(M)
\end{aligned}$$

(D-22)

Nous constatons que nous pouvons permuter le filtrage passe-bas et le laplacien. La matrice résultante est :

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} & 1 & 1 & \\ 1 & -2 & -2 & 1 \\ 1 & -2 & -2 & 1 \\ & 1 & 1 & \end{pmatrix}$$

(D-23)

C.3.f. Matrices obtenues

taille (pixels)	Matrice (voisinage à 4)	Matrice (voisinage à 8)
3×3	$\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$ (D-24)	$\begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & -4-2\sqrt{2} & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix}$ (D-25)
5×5	$\begin{pmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{pmatrix}$ (D-26)	$\begin{pmatrix} \frac{1}{\sqrt{2}} & & 1 & & \frac{1}{\sqrt{2}} \\ & & 0 & & \\ 1 & 0 & -4-2\sqrt{2} & 0 & 1 \\ & & 0 & & \\ \frac{1}{\sqrt{2}} & & 1 & & \frac{1}{\sqrt{2}} \end{pmatrix}$ (D-27)

TABLE D-2 - *matrices de calcul du laplacien*

Annexe D - Distance et angle entre deux lignes

Dans l'étape de restauration de ligne (§ III.A.6), nous devons estimer si des lignes doivent être associées (jointes et/ou unifiées) en fonction de l'angle qu'elles font entre elles et des distances de leurs extrémités. Pour réaliser plus rapidement les tests de distance et d'angle, nous avons exploité certaines propriétés mathématiques.

Soient (L_1, L_2) le couple de lignes à étudier. Nous connaissons pour chaque ligne les deux coordonnées de ses extrémités. Soient B de coordonnées (x_B, y_B) l'origine de L_1 et A de coordonnées (x_A, y_A) la seconde extrémité de L_2. Soient x_1, y_1, x_2 et y_2 les coordonnées des lignes : $L_1(x_1, y_1)$ et L_2 (x_2, y_2).

D.1. Distance

La distance AB vaut :

$$AB^2 = (x_B - x_A)^2 + (y_B - y_A)^2$$
$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \quad \text{(D-1)}$$

Soit S_{Rd} le seuil de distance maximale pour une jonction. Rd est réalisé si :

$$AB \leq S_{Rd} \Leftrightarrow AB^2 \leq S_{Rd}^2 \quad \text{(D-2)}$$

Pour déterminer Rd, il n'est donc pas nécessaire de calculer AB. AB^2 suffit, ce qui économise du temps de calcul en évitant l'appel à la fonction *racine carrée*.

D.2. Angle

Soit α l'angle entre les deux lignes.

L'angle α est tel que :

$$\cos(\alpha) = \frac{L_1 . L_2}{|L_1 L_2|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{(x_1^2 + y_1^2)(x_2^2 + y_2^2)}} \text{ (produit scalaire)}$$
$$\sin(\alpha) = \frac{L_1 \wedge L_2}{|L_1 L_2|} = \frac{x_1 y_2 - x_2 y_1}{\sqrt{(x_1^2 + y_1^2)(x_2^2 + y_2^2)}} \text{ (produit vectoriel)} \quad \text{(D-3)}$$
$$\tan(\alpha) = \frac{\sin(\alpha)}{\cos(\alpha)} = \frac{L_1 \wedge L_2}{L_1 . L_2} = \frac{x_1 y_2 - x_2 y_1}{x_1 x_2 + y_1 y_2} \text{ (pour } \cos(\alpha) \neq 0,\ \alpha \neq 90°)$$

Nous allons utiliser notamment la valeur de tan(α) pour réaliser les tests R0 et R9. La figure D-1 illustre comment ceci est possible ; cette démonstration peut-être suivie sur cette figure.

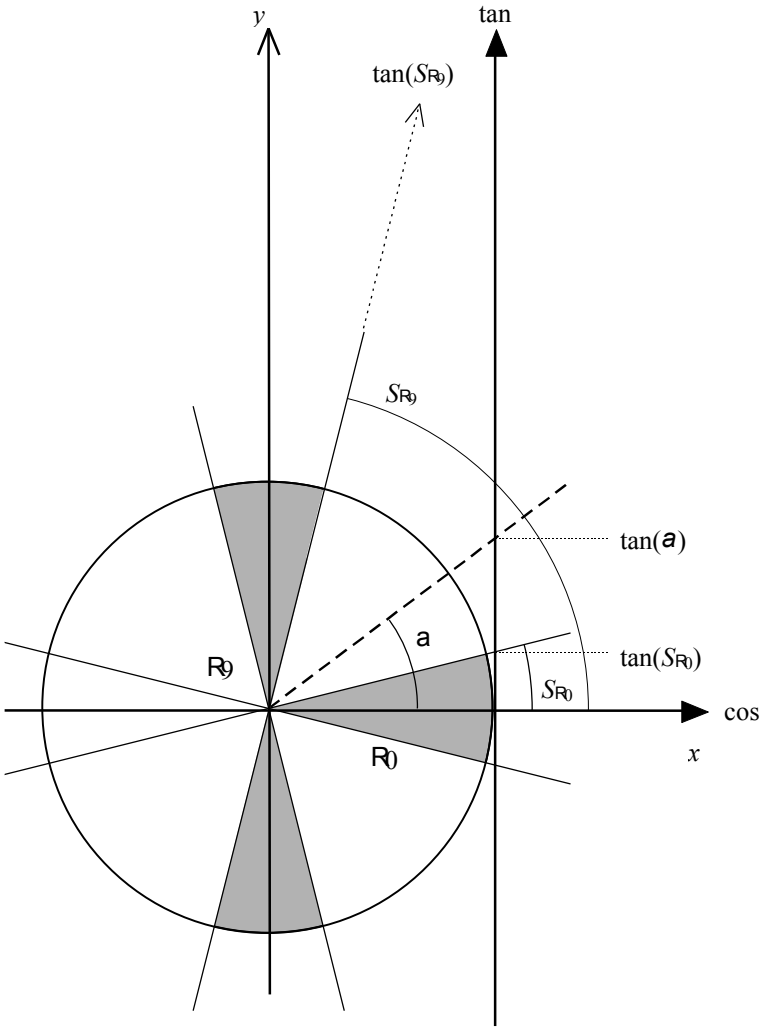


FIG. D-1 - *Appartenance aux zones des tests R0 et R9 en fonction de a*

Supposons $|a| \neq 90°$. Soit le seuil d'angle plat $S_{R0} \in]0° ; 90°[$.

R0 est réalisé si :

$$\begin{aligned} |\alpha| \le S_{P0} \quad &\Leftrightarrow \quad \left\{ |\tan(\alpha)| \le \tan(S_{P0}) \quad \text{et} \quad \cos(\alpha) \ge 0 \right\} \\ &\Leftrightarrow \quad \left\{ \left| \frac{x_1 y_2 - x_2 y_1}{x_1 x_2 + y_1 y_2} \right| \le \tan(S_{P0}) \quad \text{et} \quad x_1 x_2 + y_1 y_2 \ge 0 \right\} \\ &\Leftrightarrow \quad \left\{ |x_1 y_2 - x_2 y_1| \le |x_1 x_2 + y_1 y_2| \tan(S_{P0}) \quad \text{et} \quad x_1 x_2 + y_1 y_2 \ge 0 \right\} \\ &\Leftrightarrow \quad |x_1 y_2 - x_2 y_1| \le (x_1 x_2 + y_1 y_2) \tan(S_{P0}) \end{aligned} \qquad \text{(D-4)}$$

La dernière ligne de l'équation D-4 s'obtient lorsque l'on considère que si $x_1x_2+y_1y_2<0$, alors $(x_1x_2+y_1y_2)\tan(S_{R0})<0$ (car $\tan(S_{R0})>0$), et donc la proposition $|x_1y_2-x_2y_1| \le (x_1x_2+y_1y_2)\tan(S_{R0})$ est nécessairement fausse.

Si $|a|=90°$, alors $x_1x_2+y_1y_2=0$ (produit scalaire) et $|x_1x_2-y_1y_2|>0$ (produit vectoriel), et la proposition $|x_1y_2-x_2y_1| \le (x_1x_2+y_1y_2)\tan(S_{R0})$ est aussi nécessairement fausse, ce qui est cohérent puisque le test R0 valide des angles proches de 0°.

Supposons $|a| \neq 90°$. Soit le seuil d'angle droit $S_{R9} \in]0° ; 90°[$.

R9 est réalisé si :

$$\begin{aligned} 180°-S_{P9} \le |\alpha| \le S_{P9} \quad &\Leftrightarrow \quad |\tan(\alpha)| \ge \tan(S_{P9}) \\ &\Leftrightarrow \quad \left|\frac{x_1y_2 - x_2y_1}{x_1x_2 + y_1y_2}\right| \ge \tan(S_{P9}) \qquad \text{(D-5)} \\ &\Leftrightarrow \quad |x_1y_2 - x_2y_1| \ge |x_1x_2 + y_1y_2| \tan(S_{P9}) \end{aligned}$$

Si $|\boldsymbol{a}|=90°$, alors $x_1x_2+y_1y_2=0$ et la proposition $|x_1y_2-x_2y_1| \ge (x_1x_2+y_1y_2)\tan(S_{R9})$ est nécessairement vraie, ce qui est cohérent puisque le test R9 valide des angles proches de ±90°.

Les dernières lignes des équations D-4 et D-5 permettent donc de réaliser les tests R0 et R9, quel que soit $\boldsymbol{a}$ sans avoir à le calculer par trigonométrie inverse, ni même avoir à réaliser les calculs des longueurs $|L_1|$ et $|L_2|$ par la fonction *racine carrée*. Les valeurs de $\tan(S_{R0})$ et de $\tan(S_{R9})$ sont calculées une fois pour toutes.