# MÉMOIRE

présenté

à

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

Comme exigence partielle pour l'obtention du grade de

Maître ès Sciences Appliquées

(M.Sc.A.)

par

Louis Brassard, B.Ing.

---

# RECONNAISSANCE VISUELLE POUR UN ROBOT - CUEILLEUR DE TOMATES

---

Décembre 1990

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

# ***Mise en garde/Advice***

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

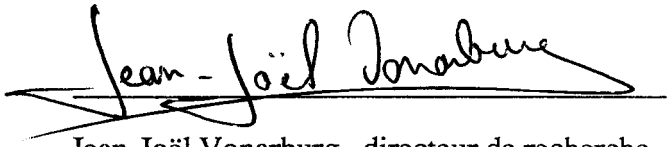
## RÉSUMÉ

Un robot-cueilleur de tomates, devant être utilisé dans une serre, doit pouvoir faire la récolte des tomates mûres et, pour ce faire, son système de vision doit lui communiquer la direction de ces dernières. La détermination, par le système de vision, de la distance des tomates n'est pas nécessaire si un senseur est placé au bout du bras du robot pour lui indiquer quand arrêter lorsqu'il se déplace en direction d'une tomate.

On propose un algorithme de localisation des tomates dans une image qui, dans un premier temps, localise le centre et le rayon de toutes les tomates dans une image couleur de la scène. On utilise à cette fin la transformation de Hough circulaire, c'est une technique qui permet de faire cette localisation sur la base de la forme circulaire des tomates. Cette technique est robuste à l'occlusion partielle d'une partie de la tomate par des feuilles ou d'autres tomates. Dans un deuxième temps, les tomates mûres qui sont rouges sont discriminées des autres qui sont vertes à partir de la couleur d'un certain nombre de points sélectionnés sur chacune.

Une grande partie de cette thèse est consacrée à résoudre le problème de reconnaissance de la couleur. Ce problème, relativement simple sous des conditions d'illumination idéales, est particulièrement compliqué sous les conditions d'illumination qui prévalent le jour dans les serres par temps ensoleillé. Nous faisons une analyse du problème qui fait intervenir un modèle de réflexion de la lumième par les objets et un modèle de la caméra, et nous analysons l'irréversibilité du procesus de mesure d'une caméra. Au chapitre V, nous faisons une analyse complète de la méthode de reconnaissance de la couleur de Wandell et Maloney. C'est une méthode qui, sous certaines conditions, permet d'estimer les conditions d'illumination et de compenser pour leurs effets. Cette méthode est nouvelle et n'a jamais été utilisée dans une application pratique car elle impose certaines prémisses difficiles à réaliser. Au chapitre VI on propose une solution à ces problèmes pour le cas de la reconnaissance de la couleur des tomates. Cette méthode est utilisée afin d'associer à chaque tomate un indice de couleur à partir duquel la tomate est classifiée comme étant soit mûre, soit non-mûre.


Louis Brassard, étudiant



Jean-Joël Vonarburg, directeur de recherche

# AVANT-PROPOS

Ce mémoire constitue l'aboutissement de trois années de recherche qui ont débutées en janvier 1987. Je voudrais d'abord remercier mon directeur de recherche, le professeur Jean-Joël Vonarburg, pour m'avoir donné l'opportunité d'entreprendre ce projet et pour m'avoir permis d'utiliser les facilités du laboratoire de robotique de l'université du Québec à Chicoutimi. Je voudrais également remercier les employés de la bibliothèque de l'UQAC pour l'aide précieuse qu'ils m'ont apportée pour la recherche bibliographique. Enfin je remercie mon épouse Astrid pour la correction du texte de ce mémoire et pour le support qu'elle m'a apporté.

En débutant cette maîtrise, ma formation était celle d'un ingénieur électrique spécialisé en automatique et ayant un peu d'expérience en informatique. Cette maîtrise fût pour moi l'occasion de m'introduire au domaine de la vision artificielle.

Comme c'est parfois le cas lorsque l'on débute un projet de recherche, les problèmes entrevus au départ ne sont pas nécessairement ceux rencontrés. Nous pensions que le principal problème auquel j'aurais à faire face serait celui de la détermination de la distance des tomates, or ce problème s'est avéré très secondaire. C'est plutôt le problème de la reconnaissance de la couleur des tomates qui a accaparé le plus mon attention. Mon seul regret est de ne pas avoir disposé d'un système de vision adéquat, ce qui m'aurait permis de pousser plus loin mon étude au niveau expérimental.

# TABLE DES MATIERES

# LISTE DES SYMBOLES

| | |
|---|---|
| $C[\lambda_1,\lambda_2]$ | Espace de Hilbert des fonctions réelles continues et dont le support est $[\lambda_1,\lambda_2]$. |
| da | Elément de surface infinitésimal d'un corps opaque. |
| DSEN | Densité spectrale d'éclairement normalisée. |
| DSRN | Densité spectrale de réflectance normalisée. |
| DBSR | Densité bidirectionnelle et spectrale de réflectance. |
| D(E) | Dimension du modèle linéaire d'éclairement. |
| D(R) | Dimension du modèle linéaire de réflectance. |
| $E(\lambda)$ | Densité spectrale d'éclairement . |
| $E(X;\lambda)$ | Densité spectrale d'éclairement du point X de la scène. |
| $\tilde{E}(X;\lambda)$ | Estimé de la densité spectrale d'éclairement du point X de la scène. |
| $\hat{E}(X;\lambda)$ | Estimé de la densité spectrale d'éclairement normalisée du point X de la scène. |
| $\hat{E}_p(\lambda)$ | Estimé de la densité spectrale d'éclairement normalisée sur la région $P$ de la scène. |
| $E$ | Modèle linéaire d'éclairement. |
| $\{ E_i \}_{i\leq D(E)}$ | Modèle linéaire d'éclairement. |
| $\mathbf{f}_c$ | Sensibilité spectrale d'une caméra idéale. |
| $\{ f_{ci} \}_{i\leq N}$ | Sensibilité spectrale d'une caméra idéale. |
| $G(\mathbf{f}_c)$ | Matrice de Gram associée à la famille $\mathbf{f}_c$ de vecteurs. |
| $I_c(X)$ | Mesure de la caméra au point X de l'image. |
| $\{ I_i \}_{i\leq N}$ | Mesure de la caméra au point X de l'image (ou dans la direction X de la scène). |
| $L(\lambda)$ | Densité spectrale de luminance. |
| $L_c(X;\lambda)$ | Densité spectrale de luminance dans la direction X relativement à l'axe optique de la caméra. |
| $L_i(\Theta_i,\Phi_i;\lambda)$ | Densité spectrale de luminance dans la direction $(\Theta_i,\Phi_i)$ relative à un système de coordonnées local à la surface d'un objet; cette fonction est appelée l'illumination ambiante. |
| N | Nombre de bandes spectrales (ou de senseurs) de la caméra. |
| $P$ | Région de la scène où la densité spectrale d'éclairement normalisée est constante. |
| $r(\lambda)$ | Densité spectrale de réflectance. |
| $\hat{r}(\lambda)$ | Densité spectrale de réflectance normalisée. |
| $r(X;\lambda)$ | Densité spectrale de réflectance du point X de la scène. |
| $\tilde{r}(X;\lambda)$ | Estimé de la densité spectrale de réflectance du point X de la scène. |
| $\hat{r}(X;\lambda)$ | Estimé de la densité spectrale de réflectance normalisée du point X de la scène, |

| | |
|---|---|
| $\Re$ | Ensemble des nombres réels. |
| $\Re^N$ | Ensemble des suites (ou familles) de N nombres réels. |
| $\Re^{N\times M}$ | Ensemble des matrices NxM de nombres réels. |
| $\mathbf{R}$ | Modèle linéaire de réflectance; c'est une famille de D(R) fonctions de $C[\lambda_1,\lambda_2]$. |
| $\{ R_i \}_{i\le D(R)}$ | Modèle linéaire de réflectance. |
| $X$ | Désigne selon les circonstances soit les coordonnées (x,y) d'un point ou d'un pixel de l'image, ou soit la direction $(\Theta_X,\Phi_X)$ (voir équations [27] et [28]) par rapport à l'axe optique d'un point de la scène. |
| $\varepsilon(X)$ | Coordonnées de $\tilde{E}(X;\lambda)$ relativement à la base $\mathbf{E}$ |
| $\hat{\varepsilon}(X)$ | Coordonnées de $\hat{E}(X;\lambda)$ relativement à la base $\mathbf{E}$ |
| $\hat{\varepsilon}_p$ | Coordonnées de $\hat{E}_p(\lambda)$ relativement à la base $\mathbf{E}$ |
| $\lambda$ | Longueur d'onde |
| $[\lambda_1,\lambda_2]$ | Spectre de la lumière visible, $[\lambda_1,\lambda_2]$ = [380 nm, 770 nm]. |
| $\rho(X)$ | Coordonnées de $\tilde{r}(X;\lambda)$ relativement à la base $\mathbf{R}$. |
| $\hat{\rho}(X)$ | Coordonnées de $\hat{r}(X;\lambda)$ relativement à la base $\mathbf{R}$. |
| $(\Theta_X,\Phi_X)$ | La coordonnée polaire et azimutale décrivant une direction par rapport à un système local de coordonnées. |

# LISTE DES FIGURES

# CONVENTIONS POUR LES NOTATIONS MATHÉMATIQUES

Notations vectorielles et matricielles

Un vecteur appartenant à un espace vectoriel de dimension fini sera noté par une lettre majuscule, ou une lettre de plus grande dimension, ou sera exprimé en termes de ses composantes et ce, de la manière suivante:

$\{ e_i \in G \}_{i \leq N}$ Famille de N éléments d'un espace vectoriel G.

$\{ e_i \}_{i \leq N}$ Famille de N éléments d'un espace vectoriel.

$$\{ e_i \}_{i \leq N} \equiv \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}$$

$$\{ e_i \}^T_{i \leq N} \equiv \begin{bmatrix} e_1 & \cdots & e_N \end{bmatrix}$$

$\left[ \{ e_i \}^T_{i \leq N} \right]$ Espace vectoriel généré par les vecteurs $e_i$ (voir théorème 4 de l'annexe 1).

Une matrice sera notée par une lettre majuscule, ou comme une famille indexée par deux indices et dont les éléments appartiennent à un espace vectoriel G.

$\underset{....}{\{ e_{ij} \in G \}}_{\substack{i \leq N \\ j \leq M}}$ Matrice à N lignes et M colonnes et dont les éléments $e_{ij}$ appartiennent à l'espace vectoriel G.

$$\underset{...}{\{ e_{ij} \}}_{\substack{i \leq N \\ j \leq N}} \equiv \begin{bmatrix} e_{11} & \cdots & e_{1M} \\ \vdots & \ddots & \vdots \\ e_{N1} & \cdots & e_{NM} \end{bmatrix}$$

$$\underset{...}{\{ e_{ij} \}}^T_{\substack{i \leq N \\ j \leq M}} \equiv \begin{bmatrix} e_{11} & \cdots & e_{N1} \\ \vdots & \ddots & \vdots \\ e_{1M} & & e_{NM} \end{bmatrix}$$

Produit scalaire et norme dans l'espace de Hilbert $C[\lambda_1,\lambda_2]$

$\langle f_1(\lambda) \mid f_2(\lambda) \rangle$ **Produit scalaire** des fonctions $f_1(\lambda)$ et $f_2(\lambda)$ de $C[\lambda_1,\lambda_2]$.

$$\langle f_1(\lambda) \mid f_2(\lambda) \rangle = \int_{\lambda_1}^{\lambda_2} f_1(\lambda)\ f_2(\lambda)\ d\lambda$$

$$\| f_1(\lambda) \|_1 = \left| \int_{\lambda_1}^{\lambda_2} f_1(\lambda)\ d\lambda \right|$$

Norme d'ordre 1 sur $\Re^N$

Soit $v = \{ v_i \}_{i \le N}$ un vecteur de $\Re^N$

$$\| v \|_1 = \left| \sum_{i=1}^{N} v_i \right|$$

Notation employée pour la mesure d'une caméra

$$\mathbf{f_c}\langle L_c(\lambda) \rangle = \{ \langle L_c(\lambda) \mid f_{ci}(\lambda) \rangle \}_{i \le N} \equiv \begin{matrix} \langle L_c(\lambda) \mid f_{c1}(\lambda) \rangle \\ \\ \langle L_c(\lambda) \mid f_{cN}(\lambda)_N \rangle \end{matrix}$$

où $\mathbf{f_c} = \{ f_{ci} \in C[\lambda_1,\lambda_2] \}_{i \le N}$ est la sensibilité spectrale de la caméra, et $L_c(\lambda) \in C[\lambda_1,\lambda_2]$ est la densité spectrale de luminance de la scène telle que vue par la caméra.

# INTRODUCTION

La mécanisation de la cueillette des tomates destinées à la transformation est déjà une réalité. Mais cette récolte mécanisée est aveugle et non sélective. En effet, tous les fruits sont récoltés simultanément et les plants sont détruits.

Les tomates cultivées dans les serres sont en général destinées au marché frais de consommation locale et, pour une qualité parfaite, elles doivent être récoltées seulement lorsqu'elles ont atteint leur maturité, caractérisée par la couleur rouge du fruit. La récolte est faite à la main et, pour les gros producteurs, elle nécessite l'embauche d'une main-d'oeuvre saisonnière considérable, engendrant des coûts appréciables.

La robotisation de la cueillette des tomates, ainsi que de certains autres types de cueilletttes fruitières, a fait l'objet de nombreuses recherches depuis une dizaine d'années. De ces recherches sont issus les prototypes expérimentaux suivants: un robot-cueilleur de tomates au Japon, un robot-cueilleur d'oranges aux États-Unis et un robot-cueilleur de pommes en France.

La réalisation d'un robot-cueilleur de fruits rencontre en général les problèmes suivants: un problème de déplacement du robot, un problème de préhension et d'évacuation des fruits, et un problème de reconnaissance visuelle pour localiser et possiblement trier les fruits. Les prototypes actuels sont encore assez loin d'avoir résolu ces problèmes à un degré tel que l'on pourrait envisager, à court terme, la commercialisation d'un robot-cueilleur de fruits.

Ce mémoire traite du problème de la reconnaissance visuelle pour un robot-cueilleur de tomates.

Au chapitre I, nous jetons un regard sur: la culture de la tomate, la faisabilité économique d'un robot-cueilleur de tomates et les principaux problèmes associés à sa réalisation. La section

1.4 fait une analyse qualitative des courbes de densité spectrale de réflectance des tomates à plusieurs stades de maturité.

Au chapitre II, nous regardons comment le tri des tomates est effectué sur les machines faisant la récolte mécanique, et nous faisons une revue des recherches les plus pertinentes sur la cueillette robotisée de fruits.

Au chapitre III, nous présentons une méthode de localisation des tomates dans une image basée sur la transformation de Hough circulaire. On présente ensuite deux algorithmes généraux permettant de localiser les tomates mûres dans une image. La reconnaissance de la couleur fait partie intégrante de chacun de ces deux algorithmes, parce qu'elle est absolument nécessaire pour discriminer les tomates mûres, qui sont rouges, des autres, qui sont vertes.

Par temps nuageux, ou la nuit lorsqu'on peut contrôler l'éclairage artificiellement, la reconnaissance de la couleur n'est pas un problème particulièrement difficile à résoudre, et plusieurs méthodes de reconnaissance de la couleur connues peuvent être mises à contribution. Mais par temps ensoleillé, la très grande majorité de ces méthodes ne sont pas applicables et c'est la raison pour laquelle les prototypes actuels de robots-cueilleurs de fruits ne peuvent fonctionner efficacement que le jour par temps nuageux et la nuit sous éclairage artificiel.

Le chapitre IV établit un cadre d'analyse dans lequel la problématique de la reconnaissance de la couleur dans des conditions d'illumination difficiles.peut être envisagée.

Au chapitre V, on fait une analyse détaillée de la méthode "computationnelle" de Wandell et Maloney. Cette méthode nous est apparue comme la plus appropriée car elle permet, sous certaines conditions, d'estimer les conditions d'éclairage et donc de compenser pour leurs variations.

Enfin au chapitre VI, on utilise la méthode de Wandell et Maloney afin d'assigner un indice de couleur à chaque tomate, et on classe ces dernières comme mûre ou non-mûre à partir de cet indice.

# Chapitre I

# APPROCHE AU PROBLÈME

## 1.1 LA TOMATE : DE LA CULTURE À LA RÉCOLTE

La tomate (*lycopersicon esculentum*) est l'objet d'une industrie alimentaire considérable. Elle est vendue sur le marché des fruits frais, mais elle est aussi vendue en conserve, en plus d'entrer dans la fabrication d'un grand nombre de produits alimentaires, tels que soupes, sauces (ketchup) et boissons.

À partir de la floraison, il faut de 40 à 60 jours (période de mûrissement) à la tomate pour être complètement mûre. Les fruits atteignent une taille définitive à peu près à la moitié de leur période de mûrissement et le reste du temps est occupé par une série d'étapes du développement qui constituent la maturation. Au début de la maturation, l'amidon s'accumule soit dans le fruit même, soit dans les parties avoisinantes. À l'approche de la maturité, la chlorophylle se dégrade graduellement, provoquant un blanchiment du fruit (stade vert-mûr). Au cours de la semaine suivante, la couleur rouge soutenue se développe ; durant cette transition, la chlorophylle restante est dégradée, des caraténoïdes (notamment le lycopène rouge) sont synthétisés, l'acidité diminue, l'amidon est transformé en sucres, les huiles essentielles et autres composés contribuant à la saveur sont synthétisés et les tissus deviennent plus mous.

Les tomates mûres sont fragiles et pourtant, pour une qualité parfaite, c'est à ce stade qu'elles devraient être cueillies. Les fruits récoltés au stade vert-mûr ont tendance à perdre leurs sucres et l'acide ascorbique; c'est une des raisons pour laquelle ils sont de moins bonne qualité que ceux ayant mûri sur la plante. Les tomates destinées aux marchés frais de consommation éloignés doivent toutefois être cueillies au stade vert-mûr ou au stade rose.

La plupart des tomates sont encore récoltées à la main mais la tendance à la récolte mécanique

s'affirme. En Californie notamment, toute la production pour la conserverie et la transformation est récoltée mécaniquement. La machine automotrice coupe la plante avec une lame qui passe légèrement au-dessous du niveau du sol, soulève et secoue la plante sur un système ascendant et incliné de baguettes horizontales, excentrées et mobiles, laisse tomber les plantes et les déchets sur le sol et pousse les fruits vers des tapis roulants en toile, disposés de chaque côté de la machine. Là, les tomates vertes ou abimées et les mottes de terre sont séparées, et les fruits triés sont envoyés vers les convoyeurs et chargés sur des camions-remorques. Il existe des variétés de plants de tomates adaptés à la récolte mécanique; la forme de ces plants facilite la récolte, le mûrissement des tomates se fait sur une plus courte période, les fruits supportent le traitement mécanique et démontrent une plus grande résistance au pourrissement sur pied.

Les tomates cultivées en serre sont destinées au marché frais et local de consommation. Les tâches associées à la culture en serre de la tomate peuvent être réparties en cinq périodes de temps [Coulon 86]: -avant plantation, -plantation, -entretien, -récolte (25% à 35% du total du temps), -et après récolte. Les conditions dans les serres n'étant pas favorables à la pollinisation des fleurs, une partie de l'entretien concerne cette pollinisation. La pollinisation est généralement effectuée à l'aide de dispositifs qui agitent les fleurs, stimulant la libération du pollen.

Pour les tomates cultivées dans les serres, le procédé de récolte mécanique exposé précédemment ne peut pas être utilisé, car on doit procéder à une récolte sélective et délicate des tomates mûres, et éviter d'endommager les plants et les autres tomates. Cette tâche est d'autant plus compliquée que les tomates poussent en bouquets et que les tomates mûres sont fragiles. Ces difficultés, et bien d'autres, expliquent pourquoi la cueillette des tomates cultivées en serre est encore faite à la main. Si l'on veut mécaniser cette cueillette, il faudra concevoir un robot-cueilleur de tomates possédant, entre autres, une capacité de reconnaissance visuelle des tomates. A ce stade-ci, il est justifié de se poser la question suivante: si un tel robot voyait le jour, aurait-il des chances d'être viable économiquement ? La section suivante tentera de répondre à cette question.

## 1.2 FAISABILITÉ ÉCONOMIQUE D'UN ROBOT-CUEILLEUR DE TOMATES

Le design d'un robot-cueilleur de tomates est un projet passionnant pour l'ingénieur, mais il ne serait qu'un exercice puéril si ce robot n'avait au départ aucune chance d'être économiquement viable. L'analyse qui est faite ici est simplifiée et très approximative, mais permet toutefois de fixer certaines idées. On cherchera à déterminer le coût maximum qu'un gros producteur de tomates en serre serait près à consacrer à la robotisation de la cueillette des tomates[1] . L'estimation qui est faite du coût maximum acceptable pour la robotisation (CMAR) permet seulement d'établir la possibilité d'une faisabilité économique et ne saurait en aucune façon la garantir.

Cette analyse s'inspire d'une étude réalisée en France [Coulon 86]: "Etude prospective technico-économique pour la robotisation de cueillette de tomates".

Le pourcentage de cueillette automatique n'intervient pas dans cette étude, mais on peut montrer que s'il est supérieur à 75%, il n'influence pas dans une large mesure la rentabilité économique du robot, mais s'il est inférieur, celle-ci est rapidement compromise, car alors l'utilisation d'un robot entraîne une utilisation inefficace de la main-d'oeuvre nécessaire à la cueillette des tomates ignorées par le robot.

On prend comme hypothèse que le volume de production de l'exploitation est suffisant pour occuper à plein temps au moins un robot durant toute la période de la récolte, et cela, sans entraîner une sous-utilisation de la main-d'oeuvre.

Afin de déterminer le CMAR, il faut évaluer dans un premier temps les économies

---

[1] Ce coût devrait couvrir:- l'achat d'un robot-cueilleur, - les coût suppémentaires nécessaires à l'aménagement de la serre, - et une marge de profit acceptable pour la manufacturier.

d'exploitation (E) que la robotisation de la cueillette permettrait à un producteur de réaliser.

$$E = [\text{Masse des tomates récoltées par le robot}] \times [\text{Coût en main-d'oeuvre pour cueillir 1 kg de tomate}] \quad [1]$$

$$E = [\, J_{récolte} \;\; h \;\; V_{robot} \,] \quad \frac{C_h}{V_{homme}} \quad [2]$$

où $J_{récolte}$ Nombre de jours que durent les récoltes dans une année.

$h$ Nombre d'heures par jour pendant lesquelles le robot peut récolter

$V_{robot}$ Vitesse de cueillette du robot en kg par heure.

$C_h$ Taux horaire payé aux travailleurs chargés de cueillir les tomates.

$V_{homme}$ Vitesse moyenne de cueillette d'un travailleur en kg par heure.

Les économies d'exploitation (E) sont reliées au CMAR, au coût annuel d'entretien du robot ($C_{entretien\text{-}robot}$) et au temps de retour sur l'investissement (T), par l'équation suivante:

$$CMAR = T \; (E - C_{entretien\text{-}robot}) \quad [3]$$

Si maintenant on remplace E dans l'équation [3] par sa valeur donnée par l'équation [2] alors on obtient l'équation suivante:

$$CMAR = T \quad [\, J_{récolte} \;\; h \;\; V_{robot} \;\; \frac{C_h}{V_{homme}} \;\; - \;\; C_{entretien\text{-}robot} \,] \quad [4]$$

Pour l'achat d'une machine agricole, un temps de retour d'investissement (T) de trois à cinq ans est généralement considéré comme raisonnable par les agronomes.

Si l'on suppose que:

$J_{récolte}$ = 225 jours = 5 x 30 jours (récolte du printemps) + 2,5 x 30 jours (récolte d'automne)

h = 12 heures de récolte par jour en moyenne.

$V_{robot}$ = 50 kg par heure (soit la moitié de la vitesse d'un cueilleur moyen)

$C_h$ = 7 $ / heure

$V_{homme}$ = 100 kg / heure

$C_{entretien\text{-}robot}$ = 1 000 $

T = 4 années

Et si l'on remplace ces valeurs dans l'équation [4], on obtient un CMAR de 33 800 $. Plus ce CMAR sera élevé et plus la faisabilité économique d'un robot-cueilleur de tomates sera assurée.

Un robot-cueilleur de tomates sera d'autant plus économique que les heures qu'il pourra travailler dans une journée seront nombreuses. Dans le dernier exemple, si le robot pouvait travailler 20 heures par jour au lieu de 12 heures et que le coût annuel d'entretien du robot était de 2 000 $ au lieu de 1 000 $, et si tous les autres paramètre étaient égaux, alors le CMAR deviendrait de 55 000 $ au lieu de 33 800 $. Il serait donc rentable pour un producteur d'investir une somme supplémentaire de 21 200 $ dans le système de vision du robot si cet investissement permettait au robot de travailler 8 heures de plus par jour.

À la lumière de ces chiffres, il apparaît donc nécessaire que le système de vision du robot-cueilleur de tomates lui permette de fonctionner de jour comme de nuit.

Il apparaît de plus probable qu'avec l'augmentation continuelle des coûts de la main-d'oeuvre et la diminution continuelle du coût des systèmes informatiques et électroniques, le jour où le prototype d'un robot-cueilleur sera suffisamment avancé sur le plan technique, il pourra être produit à un coût suffisamment en-dessous du CMAR et ainsi être viable économiquement.

## 1.3 DÉFINITION ÉLÉMENTAIRE D'UN ROBOT-CUEILLEUR DE TOMATES

On s'inspire ici de l'étude suivante [Coulon 86]: "Etude prospective technico-économique pour la robotisation de cueillette de tomates".

On peut définir quatre fonctions globales pour le robot: vision, préhension du fruit, évacuation (et tri ) du fruit et déplacement.

### Vision

Comme le robot ne doit cueillir que les tomates qui sont mûres, il doit pouvoir localiser dans l'espace ces tomates mûres, d'où une nécessité absolue d'un système de vision. Ce dernier doit pourvoir reconnaître la couleur des tomates. Du point de vue technique de vision, l'éclairage artificiel se contrôle mieux que la lumière naturelle qui est sujette à de grandes variations. C'est pourquoi la nuit est plus propice que le jour au fonctionnement du robot,.quoique la surveillance du robot la nuit pose certains problèmes. On doit viser à permettre au robot de fonctionner pendant le jour car, comme on l'a vu à la section 1.2, sa rentabilité est directement proportionnelle au nombre d'heures pendant lesquelles il peut travailler en moyenne dans une journée.

### Préhension

La préhension des fruits par le robot est délicate du fait que, actuellement, les tomates poussent en bouquets (un bouquet peut compter plus de 5 tomates) et comme tous les fruits d'un même bouquet n'arrivent tous à la maturité simultanément, il est difficile de cueillir une tomate sans endommager ses voisines.

Il est à prévoir que peu importe la sophistication du robot, certaines tomates seront hors de sa portée, ou rendues inacessibles par des nuisances, ou encore cachées complètement par le feuillage

et les tomates voisines. Il n'est donc pas question d'envisager un taux de cueillette automatique de 100%. On doit plutôt viser un taux de cueillette automatique de 75%, les fruits restants étant récoltés à la main.

Au niveau de la rapidité de cueillette du robot, on doit viser une vitesse de cueillette d'environ 50 kg de tomates par heure, soit approximativement trois tomates à la minute. La rentabilité du robot est directement proportionnelle à cette vitesse de cueillette.

## Évacuation

S'il était nécessaire d'effectuer un tri des tomates selon leurs qualités (forme, grosseur, présence de cicatrices ou dommages dûs aux insectes) , ce tri pourrait être fait lors de l'évacuation du fruit par un système de vision conçu à cette fin. Il faudrait alors prévoir des récipients différents pour chaque calibre.

## Déplacement

Le déplacement du bras de cueillette doit se faire sur un trajet fixe, ce qui facilite de beaucoup le guidage. Le bras peut, soit être déplacé au-dessus des rangées de tomates par un pont-roulant, soit être déplacé par un charriot auto-moteur, le guidage de ce dernier pouvant se faire par rail ou un fil enfoui dans le sol; le rail offre l'avantage de permettre l'alimentation électrique du charriot et du bras sans le recours à des batteries rechargeables. Le pont roulant offre l'avantage de la flexibilité dans la mobilité, mais surtout la possibilité de transporter un écran bloquant l'arrivée directe des rayons du soleil dans l'espace de travail du robot.

## Autres

Afin de rentabiliser au maximum un outil, il est souhaitable qu'il ait plus d'une fonction. On sait déjà remplacer automatiquement l'outil terminal d'un robot pour le faire changer de tâche. Il n'est donc pas utopique de réfléchir à l'aide complémentaire qu'un robot-cueilleur pourrait apporter au producteur. L'entretien de la plante arrive en tête des occupations. Les traitements sanitaires sont accessibles à un robot très simplifié. Le vibrage (pollinisation) est un travail fondamental et délicat qui occupe jusqu'à 20% du temps dans une période intensive. Certains producteurs envisagent l'automatisation de l'effeuillage.

## 1.4 DENSITÉ SPECTRALE DE RÉFLECTANCE DE LA TOMATE À DIFFÉRENTES PHASES DE MATURITÉ

La densité spectrale de réflectance est une propriété physique qui caractérise la réflexion de la lumière par les objets et que l'on mesure à l'aide d'un spectrophotomètre. En général, la densité spectrale de réflectance est présentée sous une forme normalisée pour qu'elle prenne une valeur entre 0 et 1 (ou 100), on appellera cette fonction normalisée: "la densité spectrale de réflectance normalisée" (DSRN). La possibilité de discriminer deux objects en fonction de leur couleur dépend essentiellement de la différence qui existe entre leurs DSRN respectives.

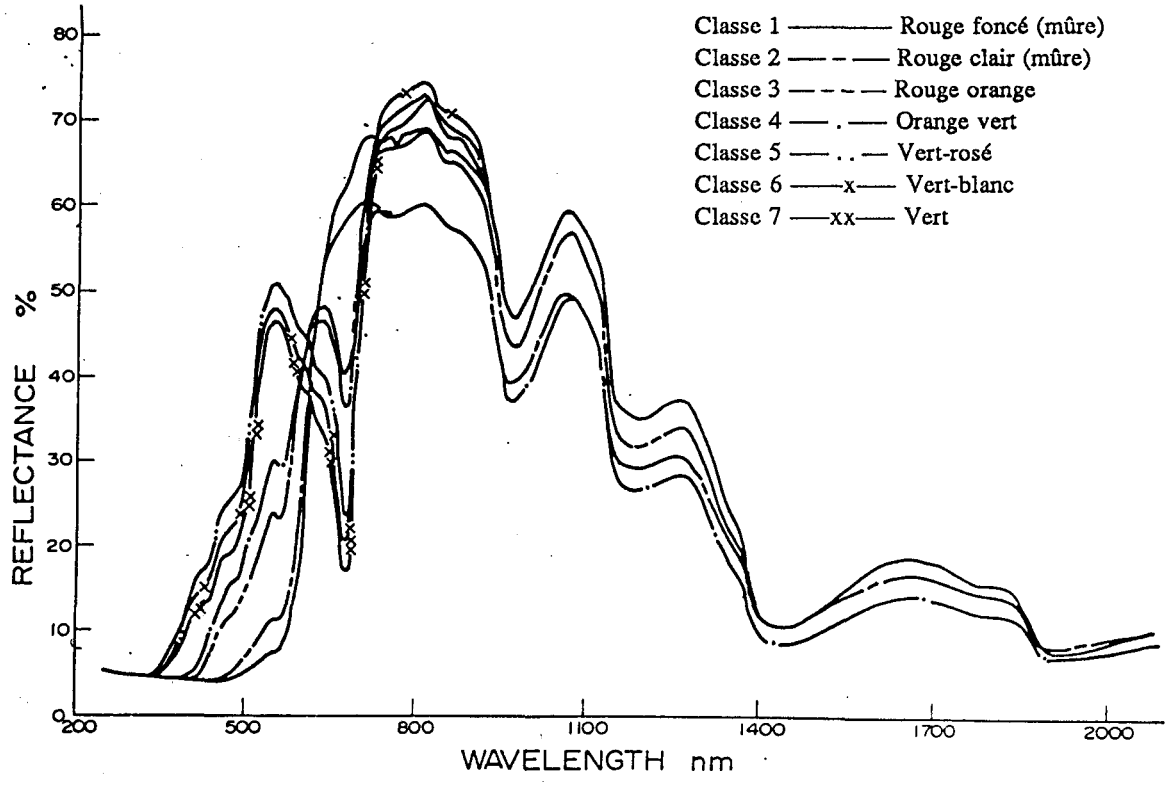



Figure 1. Densités spectrales de réflectance de la tomate à différents stades de maturité [Bittner 68]

[Bittner 68], [Goddard 70] et [Moini 78] ont mesuré les DSRN de plusieurs variétés de tomates et ce à plusieurs phases de maturité. Sur la base de ces données, nous allons faire ressortir les différences qualitatives qui existent entre la DSRN des tomates mûres (rouges) versus la DSRN

des tomates non-mûres (vertes).

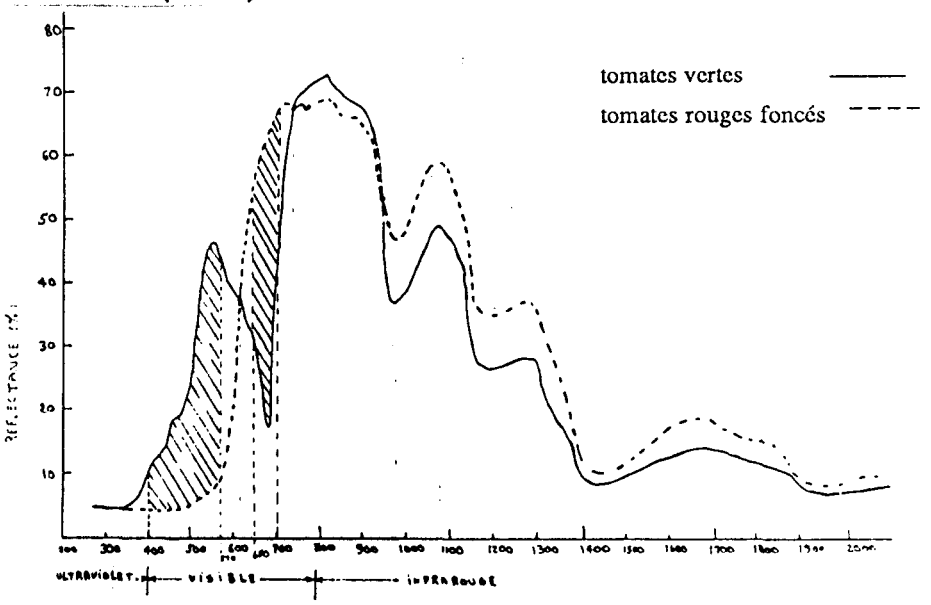



Figure 2. Densités spectrales de réflectance normalisées pour les tomates vertes et les tomates rouges

À la figure 2, apparaîssent seulement la courbe des tomates rouge foncé et celle des tomates vertes et les neuf régions spectrales suivante peuvent être identifiées :

- l'ultraviolet ( $\lambda$ <400 nm) Les DSRN ne sont pas différenciées dans cette région du spectre.
- **[400 nm, 570 nm]** Les DSRN sont beaucoup différenciées dans cette région du spectre.
- [570nm, 650 nm] Les DSRN sont peu différenciées dans cette région du spectre.
- **[650 nm, 700 nm]** Les DSRN sont beaucoup différenciées dans cette région du spectre.
- [700 nm, 970nm] Les DSRN sont peu différenciées dans cette région du spectre.
- **[970 nm, 1 100nm]** Les DSRN sont peu différenciées dans cette région du spectre.
- [1 100 nm, 1150 nm] Les DSRN sont très peu différenciées dans cette région du spectre.
- **[1 150 nm, 1 350 nm]** Les DSRN sont peu différenciées dans cette région du spectre.
- [350,∞[ Les DSRN ne sont pas différenciées dans cette région du spectre.

La diminution de la réflectance à 550 nm avec la maturité est probablement due à l'augmentation de la concentration en lycopène. L'augmentation à 675 nm avec la maturité est reliée à la diminution de la concentration en chlorophylle. Dans l'infrarouge (>800 nm), la concentration en eau dans la peau est principalement responsable du profil de réflectance. Les

bandes de l'eau sont à 970 nm, 1180 nm, 1451nm et 1940 nm.

Si l'on compare ces DSRN avec celles obtenues par [Moini 78] pour d'autres variétés de tomates avec d'autres appareils de mesure, on remarque qu'elles sont semblables dans l'ultraviolet et dans le visible [380 nm, 770 nm] mais qu'elles sont différentes dans l'infrarouge ($\lambda$>780 nm). Dans l'infrarouge, elles ont la même forme et la même amplitude sauf qu'elles sont inversées, c'est-à-dire que la courbe rouge est en bas de la courbe verte (Fig. 2). Est-ce que cela est dû aux variétés différentes de tomates, à des erreurs de mesure? Je formule l'hypothèse que les concentrations en lycopène et en chlorophylle en fonction de la maturité ne varient pas grandement d'une variété à l'autre, de sorte que dans le visible, le profil de réflectance est semblable pour toutes les variétés, tandis que la concentration en eau varie d'une variété à l'autre.

Ainsi, à la lumière de ces observations, il s'avère que la différentiation entre les DSRN des tomates vertes et celle des tomates rouges peut se faire exclusivement dans les deux bandes spectrales suivantes:

- [400 nm, 570 nm] avec un maximum à 550 nm
- [650 nm, 700 nm] avec un maximum à 675 nm.

Il aurait été intéressant d'obtenir la distribution spectrale de réflectance des feuilles mais, à défaut de cette information, il est raisonnable de supposer que celle-ci est semblable à celle des tomates vertes.

# Chapitre II

# REVUE DES MÉTHODES COMMERCIAL DE TRI DES TOMATES ET DES RECHERCHES EN CUEILLETTE DE FRUITS ROBOTISÉE

## 2.1 TRI DES TOMATES SELON LEUR COULEUR À PARTIR DE PHOTOSENSEURS

À la fin des années 1960 et au début des années 1970, on a pensé automatiser le tri sur les machines effectuant la récolte mécanique des tomates. On a mis au point des dispositifs qui font passer, l'une après l'autre, les tomates récoltées devant un ou deux photosenseurs; l'éclairage est diffus et maintenu le plus constant posssible. Les tomates sont classées selon leur couleur. On détermine ensuite un indice de couleur ($I_r$) à partir des mesures des photosenseurs. Cet indice de couleur est proportionnel à la proximité de la couleur de la tomate vis-à-vis de la couleur rouge. Finalement, chaque tomate est classée par la fonction de classification suivante:

$$\text{classe} = \begin{cases} \text{tomate mûre} & \text{si} \quad I_r \geq T \\ \text{autre (tomate non-mûre ou motte de terre)} & \text{si} \quad I_r < T \end{cases} \qquad [5]$$

où T est un seuil déterminé expérimentalement de façon à réduire au minimum les erreurs de classification . Voyons maintenant comment cet indice de couleur est déterminé.

<u>Détermination de l'indice de couleur à partir d'un seul photosenseur</u> [Power 53]

On utilise tout simplement le signal de sortie du photosenseur comme indice de couleur ($I_r$).

Le photosenseur n'étant sensible que dans la bande spectrale suivante: [650 nm, 700 nm].

$$I_r = S \; R \qquad [6]$$

où $I_r$ Signal de sortie du photosenseur.

R Réflectance moyenne de la tomate dans la bande spectrale [650, 700].

S Constante qui est fonction de l'éclairage, de la sensibilité spectrale du photosenseur et de la distance du fruit au senseur.

La constante S est fonction de plusieurs facteurs et il est difficile de les maintenir constants d'une tomate à l'autre. Afin de réduire ce problème, on peut utiliser un autre indice de couleur requérant l'utilisation de deux photosenseurs.

Détermination de l'indice de couleur à partir de deux photosenseurs [Power 53]

Afin de minimiser les variations dues à l'éclairage et à la distance du photosenseur, on a pensé utiliser comme indice de couleur ($I_r$) le rapport des signaux de sortie de deux photosenseurs ayant des sensibilités spectrales différentes.

$$I_r = I_2 \; / \; I_1 \qquad [7]$$

où

$I_1$ Signal de sortie du photosenseur dont la sensibilté spectrale est maximale autour de 550 nm.

$I_2$ Signal de sortie du photosenseur dont la sensibilité spectrale est maximale autour de 675 nm.

La plus grande robustesse de cet indice de couleur aux conditions d'éclairage et à la distance entre la tomate et les photosenseurs s'explique par le fait que cet indice de couleur est, en première approximation, proportionnel au rapport $R_2/R_1$, où $R_2$ est la réflectance moyenne de la tomate dans la bande spectrale autour de 675 nm et $R_1$ est la réflectance moyenne de la tomate dans la bande

spectrale centrée autour de 550 nm.

Les longueurs d'ondes de sensibilité maximum les plus appropriées sont:

| Senseur 1 | Senseur 2 | Référence | T suggéré | Taux d'erreur |
|---|---|---|---|---|
| 540 nm | 680 nm | [Goddard 70] | ----- | ------- |
| 528 nm | 671 nm | [Heron 71] | 0,35 | ------- |
| 520 nm | 670 nm | [Heron 74] | 0,21 | 6.2% |

Les tomates peuvent être triées non seulement en regard de leur couleur mais aussi en regard de leur qualité. Les principaux facteurs de qualité sont la forme, la grosseur, la présence de cicatrices, la présence de dommages dus aux insectes. Pour plus de renseignements sur ces méthodes de tri des tomates selon la qualité, veuillez vous référer aux deux articles publiés par Sarkar et Wolfe en 1984 et à [Robin 82].

## 2.2 REVUE DES RECHERCHES EN CUEILLETTE DE FRUITS ROBOTISÉE

### 2.2.1 PROTOTYPE JAPONAIS D'UN ROBOT-CUEILLEUR DE TOMATES

Des chercheurs de la Faculté de génie agricole de l'université de Kyoto au Japon ont construit, en 1983, un prototype de robot-cueilleur de tomates. Ils ont publié un article [Kawamura 85] dont la première partie traite de l'aspect mécanique, tandis que la deuxième partie traite de l'aspect vision du robot. Les renseignements suivants ont été tirés de la seconde partie de cet article avec beaucoup de difficultés, l'article étant écrit en japonais.

Ce prototype de robot-cueilleur de tomates comprend un chariot automoteur supportant un bras manipulateur RINO, un micro-ordinateur pour la commande du bras et l'analyse de l'image, une caméra vidéo couleur, un circuit électronique de reconnaissance du rouge et un échantillonneur de trames ("image grabber" en anglais) permettant de digitaliser le signal (image rouge) à la sortie du circuit de reconnaissance du rouge.

Le système de vision localise les tomates rouges qui sont devant le robot afin de lui permettre de procéder à la cueillette. Voyons maintenant comment cette localisation est effectuée.

Une première image de la scène est prise par la caméra; cette image est analysée afin de déterminer les centroïdes de chaque tomate rouge de l'image. Cela permet de déterminer la direction dans l'espace par rapport à la caméra où sont situées ces tomates.

La distance des tomates est déterminée grâce à la technique de vision stéréo dite "K2D" [Itoh 84] qui consiste à prendre une seconde image de la même scène à partir d'un deuxième point de vue. Une deuxième caméra peut être utilisée à cette fin, mais dans ce cas-ci, c'est la même caméra qui est déplacée par une translation d'une distance L du chariot sur lequel est monté le bras et la caméra. Une deuxième image est alors prise puis analysée afin de déterminer à nouveau la position des centroïdes des tomates rouges. A chaque tomate localisée dans la première image est ensuite

associée celle qui lui correspond dans la seconde image. Enfin, la distance (D) de chaque tomate est calculée de la façon suivante :

$$D = \frac{d\ L}{x_{i1} - x_{i2}} \qquad [8]$$

où L Déplacement de la caméra.

d Distance focale de la lentille de la caméra.

$x_{i1}$ Coordonnée horizontale du centroïde de la tomate "i" dans la première image.

$x_{i2}$ Coordonnée horizontale du centroïde de la tomate "i" dans la deuxième image.

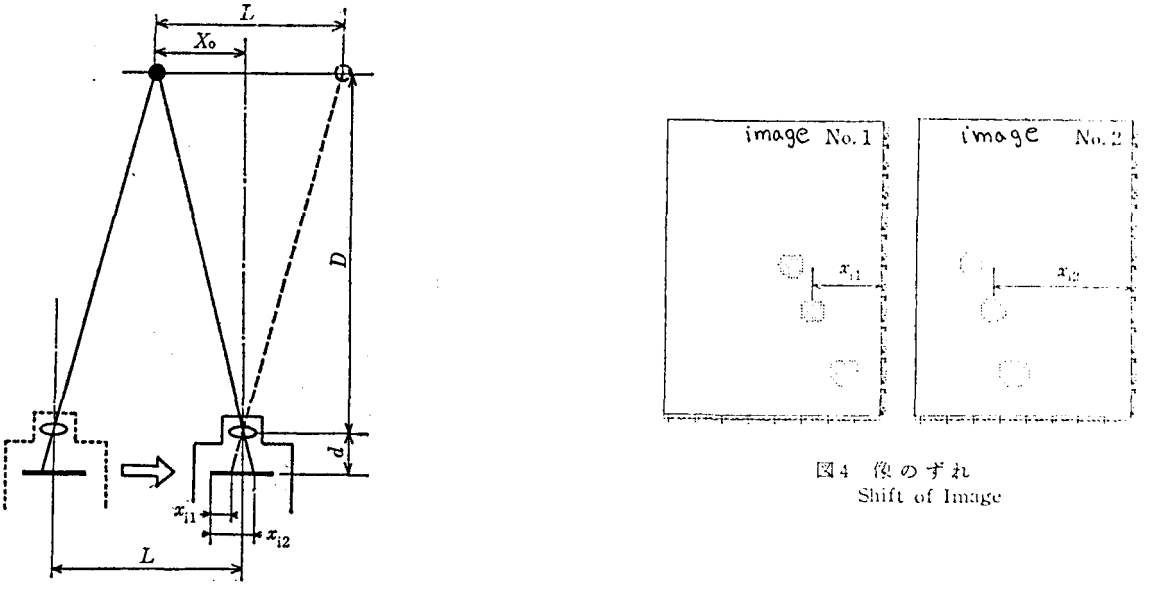



Figure 3. Technique de vision stéréoscopique K2D utilisée pour déterminer la distance des tomates [Kawamura 85].

Cette distance est déterminée avec une précision raisonnable pour des tomates situées jusqu'à 120 cm devant la caméra lorsque les conditions d'éclairage sont idéales.

Algorithme de localisation des tomates dans une image

## 1. Acquisition d'une image couleur.

La caméra utilisée est une caméra vidéo couleur qui fonctionne selon la technologie CCD et qui a une résolution de 485(V) X 344(H); l'objectif de la caméra a une distance focale de 10 mm et les signaux de sortie de la caméra respectent le standard NTSC.

## 2. Formation d'une image rouge analogique

Un circuit électronique analogique reçoit le signal vidéo composite de la caméra et en extrait deux signaux: le signal d'intensité (Y) et le signal du rouge (R). Ensuite, un deuxième circuit compare ces deux signaux et génère un signal dont l'amplitude est proportionnelle à la distance entre la couleur du pixel et la couleur rouge; on appellera ce signal "l'image rouge analogique". Aucun renseignement n'est fourni sur la métrique (distance) utilisée, mais son calcul doit être assez simple puisqu'il est réalisé par un circuit électronique analogique. Les performances de ce système de vision sont très diminuées lorsque les conditions d'éclairement ne sont pas idéales (une journée ensoleillée par exemple).

## 3. Digitalisation de l'image rouge analogique

Finalement l'image rouge analogique est digitalisée (8 bits) par un échantillonneur de trames et est mémorisée dans la mémoire du micro-ordinateur sous la forme d'une matrice de 256 par 256 qu'on appellera "l'image rouge" et que l'on notera $I_r(X)$.

## 4. Binarisation de l'image rouge

Une image binaire ( $I_b(X)$ ) est calculé à partir de l'image rouge de la façon suivante :

$$I_b(X) = \begin{cases} 1 & \text{si} \quad I_r(X) \leq T \\ 0 & \text{si} \quad I_r(X) > T \end{cases} \qquad [9]$$

Aucun détail n'est fourni sur la manière dont ce seuil T est déterminé, mais les méthodes à cet effet ne manquent pas dans la littérature: [Sahoo 88], [Ballard 82], [Pratt 78].

### 5. Filtrage de l'image binaire

Aucun détail n'est fourni sur l'algorithme de filtrage employé, mais cet algorithme de filtrage est probablement du même type que celui utilisé par le prototype américain (section 6 3).

### 6. Segmentation de l'image binaire en sous-régions contenant une seule tomate rouge

Aucun détail n'est donné sur l'algorithme utilisé, mais on indique que l'occlusion partielle des tomates par le feuillage, par les tiges ou par les autres tomates diminuait de beaucoup les performances de l'algorithme; il est donc probable qu'une méthode de segmentation très semblable à celle de la section 6.4 est utilisée.

### 7. Les coordonnées des centroïdes des tomates rouges sont calculées

Pour chaque sous-région (ou segment) de l'image binaire identifiée comme ne contenant qu'une seule tomate rouge, on calcule les coordonnées (X = (x,y) ) du centroïde de l'objet (ensemble des pixels ayant la valeur "1" et correspondant à une tomate rouge). Aucun détail n'est donné sur ce calcul mais une méthode du type de celles suggérées par [Horn 86] est probablement utilisée.

### 2.2.2 PROTOTYPE FRANÇAIS D'UN ROBOT-CUEILLEUR DE POMMES

Des chercheurs du C.E.M.A.G.R.E.F[2] de Montpellier en France ont construit un prototype de robot-cueilleur de pommes. À l'automne 1986, le quatrième prototype était à l'essai dans les vergers de la région de Montpellier. Les renseignements sur ces recherches ont été obtenus à partir des articles suivants: [Grand d'Esnon 85] et [Morice 86].

Le dernier de cette lignée de prototypes consiste en un chariot qu'on doit pour l'instant déplacer manuellement et sur lequel est installé un portique porteur à crémaillère permettant de déplacer verticalement un bras télescopique; le bras peut également faire une rotation dans le plan horizontal, lui conférant ainsi trois degrés de liberté. A l'extrémité du bras, il y a une pince de cueillette en caoutchouc munie d'un clapet mobile lui permettant de couper le pédoncule du fruit.

Seulement la direction du fruit, c'est-à-dire les coordonnées du centroïde du fruit dans une image, est déterminée par le système de vision. Contrairement au prototype japonais, la distance du fruit n'est pas déterminée par le système de vision du robot.

Pour la cueillette d'une pomme, le bras télescopique est orienté (mouvement de rotation du bras dans le plan horizontal) dans la direction du fruit et il s'allonge dans cette direction jusqu'au moment où les capteurs de proximité placés sur la pince de cueillette signalent la présence du fruit.

La cueillette du fruit opérée, le bras se recule au-dessus de la plate-forme, pour dégager le champ de vision et recommencer une nouvelle opération. La pince s'ouvre et le fruit est évacué par l'intermédiaire d'un décélérateur qui le conduit dans le palox. Ce dernier tourne sur lui-même pour assurer une bonne répartition des fruits;

Ce robot vise à récolter environ 70% des fruits, ceux inaccessibles pouvant être cueillis par

---

[2] Centre national du machinisme agricole, du génie rural, des eaux et forêts

l'équipe réduite, fixe, employée à l'année par l'arboriculteur. Le prototype prend actuellement quatre secondes pour récolter une pomme.

Un robot commercial serait constitué de quatre bras de cueillette, disposés par paire de chaque côté de la rangée fruitière et montés sur un chariot automoteur et autoguidé.

## Algorithme de localisation des pommes

### 1. Acquisition d'une image binaire

Sur le bras du robot, il y a une caméra monochrome à balayage (Reticon LC 110) fonctionnant selon la technologie CCD. La caméra possède un objectif grand angle en avant duquel est placé un filtre dont le maximum de sensibilité est autour de 650 nm .

La stratégie de reconnaissance de la couleur rouge qui est utilisée ici, repose sur l'hypothèse que, dans une certaine bande du spectre, la luminance des fruits mesusée par la caméra est supérieure à celle des feuilles. Dans ces conditions, si la caméra est munie d'un filtre qui n'est sensible qu'à l'intérieur de cette bande spectrale, alors l'intensité du signal vidéo à la sortie du senseur de la caméra sera maximal pour les pixels correspondant aux fruits. Par temps nuageux ou la nuit sous un éclairage artificiel diffus, cette hypothèse est en général vérifiée. Par temps ensoleillé, cependant, certaines feuilles pourront avoir une luminance quatre fois plus grande que celle des fruits placés à l'ombre.

À tous les 1/10 de seconde, la caméra mesure à l'aide de son senseur (256 photodiodes) une image[3] d'une bande horizontale de la scène. Notons par $I_r(x)$ (où $1 \leq x \leq 256$), le signal analogique à la sortie du senseur de la caméra. Puisque la caméra est pourvue d'un filtre sensible seulement

---

[3] Le terme "image" pour qualifier le signal unidimensionnel à la sortie du senseur n'est pas approprié, car une image est par définition un signal bidimensionnel.

dans la bande spectrale où la luminance des fruits est supposée être supérieure à celle des feuilles, la valeur en x du signal $I_r(x)$ sera d'autant plus grande que la couleur du pixel x sera proche du rouge.

A l'intérieur de la caméra, un circuit électronique convertit ensuite le signal $I_r(x)$, à la sortie du senseur, en un signal binaire que l'on notera $I_b(x)$. La binarisation de $I_r(x)$ est faite comme suit:

$$I_b(x) = \begin{cases} 1 & \text{si} \quad I_r(x) \geq T \\ 0 & \text{si} \quad I_r(x) < T \end{cases} \qquad [10]$$

Le seuil T est déterminé lors de la calibration de la caméra. Si la caméra est calibrée de façon appropriée alors les pixels de couleur rouge (couleur des pommes) auront la valeur logique "1" et les autres auront la valeur logique "0"

## 2. Détermination des coordonnées des centroïdes des fruits

Un fruit est localisé dans l'image binaire, lorsque plus de six "1" sont détectés. L'ordinateur calcule alors le centre de la tache visuelle qui correspond au fruit.

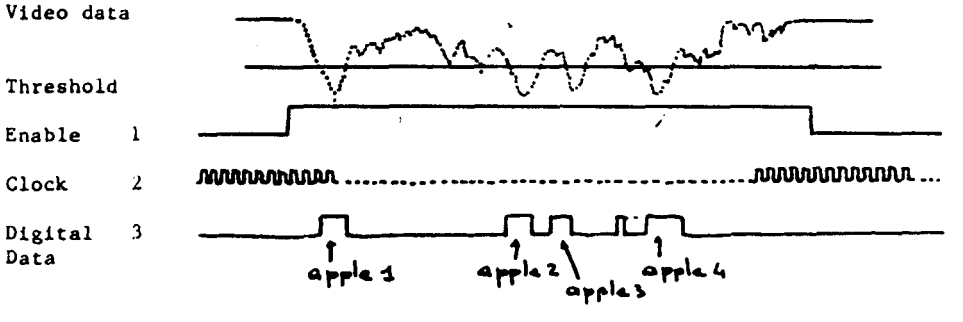



Figure 4. Signaux de la caméra utilisée sur le prototype français de robot-cueilleur de pommes

On remarque que les quatre premières étapes de l'algorithme de localisation des tomates mûres dans une image du prototype japonais sont toutes présentes, mais elles sont simplifiées et réalisées à l'intérieur même de la caméra. Les trois dernières étapes de l'algorithme du prototype japonais sont également présentes ici, mais sous une forme très simplifiée. Ainsi, le fait d'ignorer les groupes de moins de six pixels est une forme de filtrage binaire, localiser les groupes de six pixels et plus constitue l'étape de la segmentation et la dernière étape consiste à trouver le centre de chaque tache visuelle.

### 2.2.3 PROTOTYPE AMÉRICAIN D'UN ROBOT-CUEILLEUR D'ORANGES

Aux État-Unis, la société Martin Marietta a déposé un brevet de bras-cueilleur. C'est l'université de la Floride qui mène la recherche robotique alors que l'université de Purdue s'intéresse à la vision. Les informations recueillies sur ce robot l'ont été à partir de l'article suivant: [Tutle 83].

Le robot a été conçu pour la récolte de trois sortes de fruits: l'orange, le pamplemousse et le citron. Un système typique consiste en trois modules de cueillette robotisée autonome (Fig.5) , montés sur un chariot tiré par un tracteur.

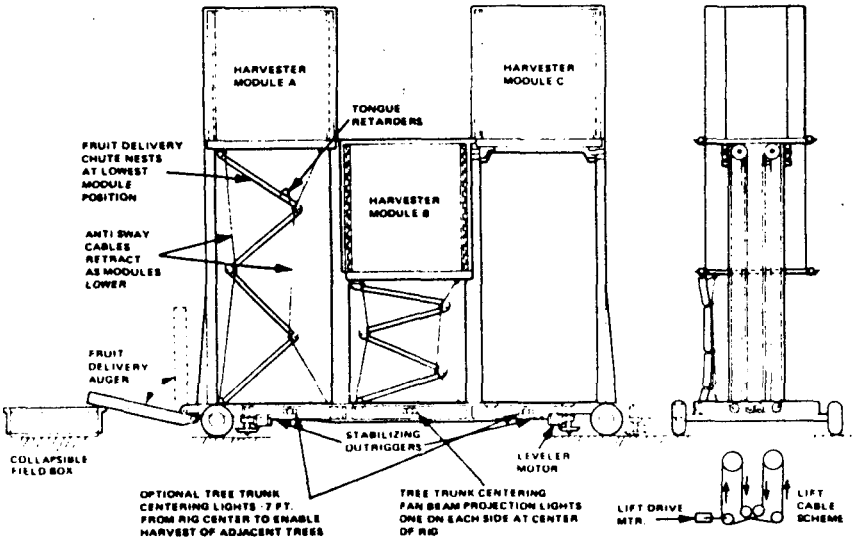



Figure 5. Trois modules de cueillette [Tutle 83]

Les trois modules (Fig. 5) peuvent travailler simultanément sur la moitié d'un arbre d'un coté de la rangée d'arbres, puis travailler sur la moitié d'un autre arbre dans la rangée adjacente et ce, sans avoir à repositionner le chariot. Cela suppose que la plantation des arbres dans les vergers ait été faite de façon à optimiser la récolte.

Chaque module (Fig. 6) comprend: un système de vision (une caméra munie de deux filtres interchangeables, un échantillonneur de trames, un système d'éclairage et un ordinateur) et un bras de cueillette avec son contrôleur et un ordinateur pour la supervision des opérations.

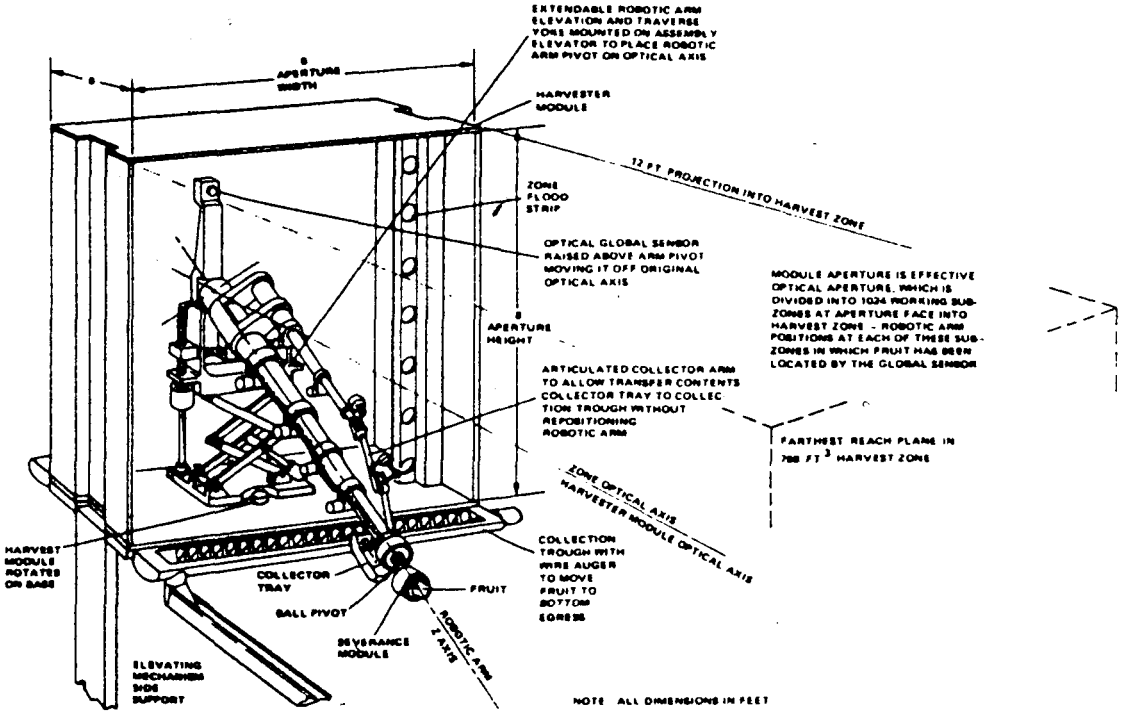



Figure 6. Module de cueillette avec son bras conçu par la société Martin Marietta [Tutle 83].

Le bras de cueillette (Fig. 6) est montée sur une base qui lui permet un mouvement de rotation dans le plan horizontal et qui lui permet également de monter et de descendre. Le bras de cueillette est télescopique et peut être incliné dans le plan vertical relativement à sa base. Cela confère au bras quatre degrés de liberté. La pince de cueillette (Fig. 7) est placée au bout du bras par une articulation sur pivot permettant d'orienter la pince dans toute les directions jusqu'à 15 degrés.

Un bras secondaire (Fig. 6) permet de collecter les fruits cueillis et de les déposer dans un mécanisme d'évacuation.

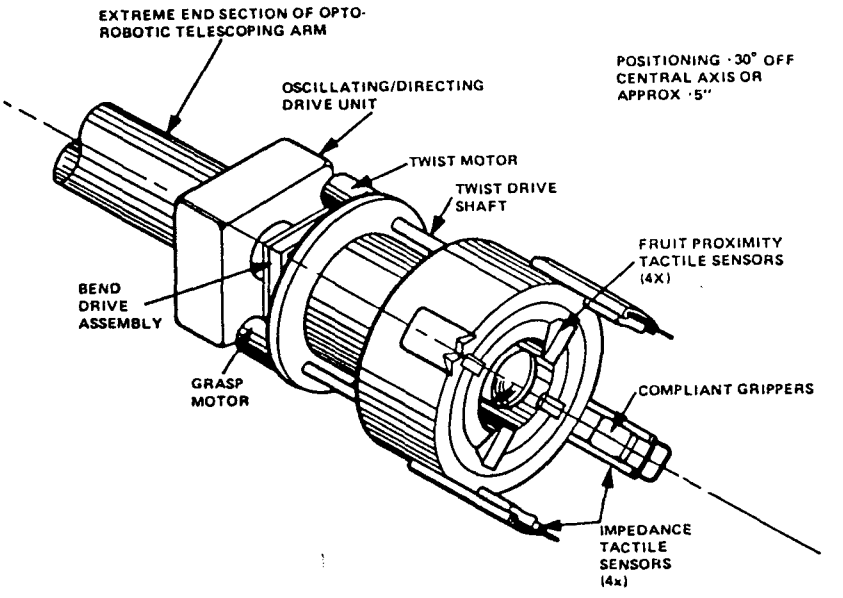



Figure 7. Pince du bras de cueillette de la société Martin Marietta [Tutle 83]

Une caméra CCD (256H x 256V) à grand angle placée au centre du module permet de déterminer approximativement la direction de tous les fruits devant le module de cueillette. Par la suite, le bras de cueillette est envoyé successivement dans chacune de ces directions à la vitesse de 4 mètres par seconde.

A l'intérieur de la pince de cueillette (Fig. 7), il y a une source lumineuse et quatre photodiodes placées en croix ("Martin Marietta Series LD230 quad detectors") qui sont munies d'un filtre passe-bande [600nm,700nm]. En fin de course, la pince exécute un mouvement d'oscillation dans toutes les directions afin de chercher le fruit. Ce dernier sera détecté lorsque les photodiodes recevront la lumière réfléchie par le fruit. Le bras cherchera à s'orienter afin que les 4 photodiodes reçoivent les mêmes intensités lumineuses, signe que le fruit est aligné avec l'axe de la pince de cueillette. Des senseurs tactiles permettent à la pince de cueillette d'exercer la pression désirée sur le fruit.

Algorithme de localisation des oranges dans une image

## 1. Acquisition et digitalisation de deux images et formation de "l'image orange"

Les densités spectrales de réflectance des feuilles et des oranges se différencient surtout dans les deux bandes spectrales suivantes: [600 nm, 700 nm] et [750 nm, 850 nm].

Pour l'orange (le fruit), le rapport des réflectances moyennes sur ces bandes spectrales est égal à cinq ($r_{o[600,700]}/r_{o[750,850]} = 5$) et pour les feuilles, il est égal à 0,25 ($r_{f[600,700]}/r_{f[750,850]} = 0{,}25$).

Deux images de la scène sont prises et digitalisées. Une première image, que l'on notera $I_A(X)$, est prise en plaçant devant la caméra un filtre qui n'est sensible que dans la bande spectrale [600 nm, 700 nm]. Une deuxième image, que l'on notera $I_B(X)$, est prise en plaçant devant la caméra un filtre qui n'est sensible que dans la bande spectrale [750 nm, 850 nm].

Lorsque l'éclairage est diffus, le rapport des intensités $I_A(X) / I_B(X)$ est, en première approximation, égale au rapport des réflectances moyennes sur les bandes spectrales A et B.

$$I_A(X) / I_B(X) = r_A(X) / r_B(X) \qquad [11]$$

où

$r_A(X)$ Réflectance moyenne sur l'intervalle [600 nm, 700 nm].de la surface de la scène au pixel X.

$r_B(X)$ Réflectance moyenne sur l'intervalle [750 nm, 850 nm].de la surface de la scène au pixel X.

On peut donc s'attendre à ce que ce rapport soit égal à 0,25 dans le cas des feuilles et à 5 dans le cas des oranges. On forme donc, à partir du rapport des images $I_A(X)$ et $I_B(X)$, une image appelée "l'image orange" que l'on notera par $I_O(X)$:

$$I_O(X) = \frac{I_A(X)}{I_B(X)} \qquad [12]$$

La nuit sous l'éclairage de lampes au tungsten (2500 $^{o}$K) placées de façon à éliminer les

ombres, l'équation [12] tient. Par temps ensoleillé cependant, certaines feuilles qui sont près de la caméra et dont l'angle par rapport au soleil favorise la réflexion directe de la lumière vers la caméra, pourront avoir un rapport $I_A(X)/I_B(X)$ quatre fois plus élevé que celui de certains fruits à l'ombre. C'est pourquoi ce prototype est limité à des opérations de nuit.

## 2. Binarisation de l'image orange

La binarisation de l'image orange est faite de la façon suivante:

$$I_b(X) = \begin{cases} 1 & \text{si} \quad I_O(X) \geq T \\ 0 & \text{si} \quad I_O(X) < T \end{cases} \qquad [13]$$

où le seuil T, si adéquatement choisi, permet de discrimiminer entre les fruits et le feuillage.

## 3. Filtrage de l'image binaire

L'image binaire est filtrée afin d'éliminer les pixels de valeur "1" qui sont isolés et qui correspondent probablement à du bruit.

## 4. Segmentation de l'image en régions contenant un seul fruit.

Aucun détail n'est donné sur l'algorithme utilisé. Apparemment, l'occlusion partielle des oranges par le feuillage, par les tiges ou par d'autres oranges diminuent de beaucoup les performances de l'algorithme. Ce qui nous incite à croire que la méthode de segmentation utilisée pourrait être semblable à celle présentée à la section 6.4.

## 5. Les coordonnées des centroïdes des oranges sont calculées

### 2.2.4 RECONNAISSANCE VISUELLE DE FRUITS: L'APPROCHE DE SITES ET DELWICHE

P.W. Sites & M.J. Delwiche ont publié l'article [Sites 85]:"Computer vision to locate fruit on a tree" . Leur étude portait sur le cas de la pomme et de la pêche. Voici l'algorithme de localisation des pommes dans une image qu'ils proposent:

#### 1. Acquisition d'une image

La caméra (TN2200 de Général Electric) fonctionne selon la technologie CID ("Charge Injected Device"), son objectif a une distance focale de 25 mm et son senseur a une résolution de 128x128 pixels. On a déterminé qu'une résolution de 25 $mm^2$/pixel était suffisante pour distinguer les fruits dans une image. Cela veut dire qu'avec cette camera une image peut couvrir une portion d'arbre de .64 m par .64 m. En effet :

$$25\ mm^2/\text{pixel} = \frac{640\ mm \times 640\ mm}{128x128\ \text{“pixels”}}$$

La nuit, un filtre passe-bande (#35-5115) est placé devant la caméra. Ce filtre a un maximum de transmittance de 80% et n'est sensible que dans la bande spectrale [610nm,690nm]. Le jour, c'est un filtre passe-bande (#35-4035) qui est placé devant la caméra. Ce filtre a un maximum de transmittance de 46% et n'est sensible que dans la bande spectrale [648.4nm, 691.6nm]. Deux lampes "flood" à 4800 $^{o}$K de 500W sont placés de chaque côté de l'arbre. Cet éclairage est même utilisé de jour, car il atténue les ombres et les variations de l'ensoleillement. Les filtres ont été choisis de façon à ce que l'insensité de l'image soit maximum aux pixels correspondant aux fruits. Évidemment, cela n'est pas toujours le cas, particulièrement le jour par temps ensoleillé.

#### 2. Digitalisation de l'image

L'image est digititalisée par un convertisseur analogue-numérique de 8 bits permettant 256 niveaux de gris.

### 3. Accentuation de l'intensité des pixels appartenant aux régions uniformes

Les régions de l'image correspondant aux fruits ont tendance à être plus uniformes que les autres. Afin d'aider la binarisation (discrimination des pixels correspondant aux fruits des autres), il est avantageux de mettre à profit cette propriété en augmentant l'intensité des pixels appartenant aux régions uniformes de l'image au détriment de l'intensité des autres pixels:

$$I_e(x,y) = 100\,[\,1 - |\,(I(x,y) - I_{moyen})/(\,I(x,y)\,|\,] \qquad [14]$$

où $I_e(x,y)$ est l'image résultante et $I_{moyen}$ est la moyenne des intensités de l'ensemble "8-voisins du pixel X"[4]

### 4. Binarisation de l'image.

$$I_b(X) = \begin{cases} 1 & \text{si} \quad I_e(X) \geq 10 \\ 0 & \text{si} \quad I_e(X) < 10 \end{cases} \qquad [15]$$

### 5. Filtrage de l'image binaire

$$I_{bf}(X) = \begin{cases} 1 & \text{si} \quad N_8(X) \geq 5 \\ I_b(X) & \text{si} \quad N_8(X) = 4 \\ 0 & \text{si} \quad N_8(X) \leq 3 \end{cases} \qquad [16]$$

où $N_8(X)$ est le nombre de pixels qui appartiennent à l'ensemble 8-voisins du pixel X et qui sont égals à "1".

### 6. Segmentation de l'image en sous-régions (segments) contenant un seul fruit

6.1 Segmentation de l'image en sous-régions contenant potentiellement un ou plusieurs fruits. Ce processus consiste à identifier les ensembles "connectés" [Nevatia 82], mutuellement exclusifs de pixels, pour lesquels $I_{bf}(X) = 1$. Chacun de ces ensembles de pixels correspondent à un object dans la scène.

6.2. Un vecteur de sept caractéristiques est déterminé pour chaque object; ces caractéristiques

---

[4] Ensemble 8-voisins du pixel X = {(x,y+1),(x+1,y+1),(x+1,y),(x+1,y-1),(x,y-1),(x-1,y-1),(x-1,y),(x-1,y+1)}

sont: le périmètre(P), la surface(S), le rapport de rondeur = $\pi$ S/P2, $\phi$1, $\phi$2, $\phi$3 et le rapport d'élongation (Max{$\mu_{02}$, $\mu_{20}$}/ Min{$\mu_{02}$,$\mu_{20}$})[5].

6.3. Classification de chaque objet dans l'une des catégories suivantes: fruit individuel, groupe de fruits, bruit. A cette fin plusieurs type de classificateurs peuvent être utilisés: linéaire par morceau, quadratique, polynomial, Baysien. L'auteur a utilisé un classificateur linéaire paramétrique [Fu 76] qui a réussi à classifier correctement 89.5% des fruits individuels, 95.2% des fruits en groupes et 90.5% du bruit.

6.4 Reconnaissance des fruits individuels pour chaque objet identifié comme groupe de fruits. Plusieurs méthodes existent [Riutala 68], [Arcilli 71], [Parrish 77].

## 7. Calcul des coordonnées des centroïdes des fruits

Cet algorithme de localisation des fruits dans une image fonctionne moins bien avec les pommes qu'avec les pêches, en raison du feuillage plus abondant des pommiers. La faute est imputable à la méthode de segmentation de l'image qui n'est pas très robuste à l'occlusion des fruits par les feuilles. De plus, lorsque les fruits sont groupés, et c'est presque toujours le cas pour la tomate, cet algorithme éprouve encore d'autres difficultés.

La méthode de segmentation de l'image de la section suivante permet de solutionner avantageusement ces problèmes.

---

[5] $\mu_{20}$ et $\mu_{02}$ sont les moments d'inertie centraux, respectivement en x et en y de chaque objet [Gonzalez 87] et $\phi$1,$\phi$2 $\phi$3 sont des moments, définis par [Hu 62], qui sont invariants en rotation, en translation et en échelle.

# Chapitre III

# RECONNAISSANCE VISUELLE DES TOMATES:

# APPROCHE GÉNÉRALE

## 3.1 LOCALISATION DES TOMATES DANS UNE IMAGE BINAIRE DE CONTOURS PAR LA TRANSFORMATION DE HOUGH CIRCULAIRE

Une image binaire de contours est une image obtenue par la transformation d'une image digitale (monospectrale ou multispectrale) par un opérateur binaire d'extraction d'éléments locaux de contour. La valeur associée à un pixel quelconque de l'image binaire de contours sera "1", si le contour d'un objet a été jugé présent sur ce pixel, et sera "0" dans le cas contraire. La notion de frontière des objets dans une image et des discontinuités de la luminance qu'elle provoque sont assez complexes et c'est pourquoi il existe plusieurs opérateurs binaires d'extraction d'éléments de contours différents [Martelli 76],[Persoon 76] et que la littérature à ce sujet est particulièrement abondante.

La transformation de Hough a été originellement conçue [Hough 62] afin de localiser des droites dans une image binaire de contours. Cette méthode a été reprise et améliorée par Duda et Hart (1972); Ballard (1980) généralisa la méthode à une courbe quelconque.pouvant être paramétrisée. Kimme et al (1975) ont proposé une façon de calculer efficacement la transformation de Hough circulaire. Cette brève revue de la littérature sur la transformation de Hough est très imcomplète.

Le résultat de la transformation d'une image binaire de contours par la transformation de Hough circulaire est un tableau à trois indices (ou à trois dimensions) appelé le tableau des accumulateurs. Le premier indice du tableau correspond au rayon d'un cercle et les deux derniers correspondent

aux coordonnées d'un point de l'image binaire de contours originale. Théoriquement, chaque maximum dans le tableau des accumulateurs correspond à un cercle dans l'image binaire originale de contours: les deux derniers indices de la position du maximum correspondent aux coordonnées du centre du cercle dans l'image binaire de contours, et le premier indice correspond à son rayon.

La transformation de Hough circulaire semble donc tout indiquée pour localiser les tomates dans une image binaire de contours et ce, même si les tomates n'ont pas un contour de forme parfaitement circulaire et même si très souvent une partie du contour de la tomate n'est pas visible, c'est-à-dire cachée par des feuilles, par d'autres tomates ou tout simplement dans l'ombre. Cette transformation permet de localiser les cercles, même si une ou plusieurs parties du contour du cercle sont absentes et même si le contour n'est pas parfaitement circulaire.

La première étape consiste à transformer l'image originale en une image binaire de contours via un opérateur binaire d'extraction d'éléments locaux de contour. Les programmes EDGE_1 à EDGE_4 ainsi que les programmes HISTO et THRESHOD de l'annexe III ont été utilisés pour obtenir ces images binaires de contours.

L'étape suivante consiste à transformer cette image binaire de contours via la transformation de Hough circulaire et à rechercher les maximums dans le tableau des accumulateurs.

Algorithme de localisation des tomates dans une image binaire de contours

1. Transformation de Hough circulaire (formation du tableau des accumulateurs).
2. Localisation des maximums dans le tableau des accumulateurs.

Les programmes HOUGH_1 à HOUGH_4 de l'annexe III suivent tous cet algorithme général. Ils diffèrent cependant dans leur stratégie de localisation des maximums dans le tableau des accumulateurs.

Le programme HOUGH_1 utilise une stratégie qui considère comme des maximums toutes les valeurs du tableau des accumulateurs qui sont supérieures à un certain seuil T. Ce seuil est constant pour un rayon donné (premier indice du tableau), mais varie en fonction de ce rayon de la façon suivante:

$$T = 21 + (R - 10)$$

où T = Seuil variable selon le rayon

R = Rayon en nombre de pixels du cercle; des rayons de 10 à 20 ont été utilisés.

Les programmes HOUGH_2 à HOUGH_4 ont tenté d'améliorer cette stratégie de localisation des maximums en éliminant la possibilité de cercles concentriques ou de cercles dont les centres sont très rappochés. Ces programmes fonctionnent bien avec des images binaires de contours générées par le programme TCERCLE (annexe III).

Nous nous sommes aperçus que les images digitales de tomates dont nous disposions étaient dilatées dans un rapport de 3 horizontalement à 4 verticalement, de sorte que le contour des tomates dans ces images était oval, empêchant l'algorithme ci-dessus de fonctionner. La raison de cette dilatation est que la caméra vidéo que nous utilisions suit le standard NTSC défini pour la télévision. Pour pouvoir utiliser la transformation circulaire de Hough il faut utiliser une caméra dont les pixels sont carrés (voir l'annexe II pour le choix d'une caméra).

Ce problème aurait pu être résolu de plusieurs façons. On peut modifier la transformé afin de localiser des ellipses dont les dimensions respectives des axes verticale et horizontale sont dans les rapport 3 à 4. Ou peut également transformormer l'image en une dont les pixels sont carrées.

Avec le recul, nous croyons que la stratégie pour localiser les maximums dans le tableau des accumulateurs devrait utiliser un seuil pour identifier les régions du tableau où ces maximums sont susceptibles d'être localisés, puis de rechercher les maximums dans chacune de ces régions, en utilisant une méthode s'apparentant aux méthodes numériques recherchant les maximums d'une

fonction à plusieurs variables, et enfin certains de ces maximums pourraient être éliminés à partir de considérations topologiques sur la scène telles que l'impossiblilité d'avoir des cercles concentriques.

Ayant pris connaissance des travaux de [Whittaker 84] qui démontre la possibilié d'utilisée cette méthode, nous avons interrompu nos recherches dans cette direction. Whittaker propose d'utiliser un seuil pour localiser les maximums dans le tableau des maximums. L'auteur a montré que la distribution des accumulateurs dans le tableau des accumulateurs suit une distribution exponentielle. Il est donc possible, selon l'auteur, de choisir ce seuil pour n'importe quel degré de confiance à partir de la moyenne des accumulateurs. Il suggère à cet effet un degré de confiance de 99%.

Nous croyons que cette méthode est la mieux indiquée, car elle est particulièrement robuste, de nature, à l'occlusion partielle des tomates par les feuilles ou les autres tomates.

## 3.2 LOCALISATION DES TOMATES MÛRES

Le système de vision du robot-cueilleur de tomates comprendra une caméra couleur, un échantillonneur de trames (carte digitalisante) et un ordinateur. L'échantillonneur de trames permet de digitaliser l'image à la sortie de la caméra et de conserver cette image digitale dans la mémoire de l'ordinateur.

Afin de pouvoir faire la cueillette des tomates mûres, un robot-cueilleur de tomates doit pouvoir déterminer les directions, par rapport à l'axe optique de la caméra, qui correspondent à ces tomates. Cela revient à déterminer les coordonnées des centroïdes des tomates mûres dans l'image de la scène.

La distance des tomates doit également être connue par le robot-cueilleur. Cette distance pourra soit être déterminée par un capteur placé au bout du bras du robot, comme c'est le cas sur les prototypes français et américain, soit être déterminée par le système de vision du robot comme c'est le cas sur le prototype japonais. Pour ce faire, le système de vision devra appliquer l'algorithme de localisation des tomates sur deux images de la même scène, prises à partir de positions différentes, et puis associer à chaque tomate localisée dans la première image celle qui lui correspond dans la deuxième image et déterminer sa distance par triangulation.

Afin de discriminer entre les tomates mûres qui sont rouges et les autres qui sont vertes, la reconnaissance de la couleur est essentielle. Nous croyons, par ailleurs, qu'un algorithme de localisation des tomates basé exclusivement sur la couleur, tel que ceux présentés aux sections 6.1 à 6.4, n'est pas satisfaisant, et ce, pour la raison suivante. Les tomates font souvent partie d'un bouquet qui peut en compter jusqu'à six, et les tomates d'un même bouquet ne mûrissent pas nécessairement en même temps. Il est donc important, pour éviter d'endommager les tomates vertes situées à proximité des tomates rouges qui doivent être cueillies, que le système de vision soit en mesure de localiser également ces tomates vertes. L'algorithme de localisation des tomates

doit nécessairement localiser ces tomates vertes sur la base de leur forme et, à cet effet, l un algorithme fondé sur la transformation de Hough circulaire semble tout indiqué.

On propose ici deux algorithmes de localisation des tomates dans une image. Le premier localise toutes les tomates de l'image à partir de leur forme pour ensuite déterminer à partir de leur couleur celles qui sont mûres. Le deuxième algorithme localise dans un premier temps seulement les tomates mûres et vérifie dans un deuxième temps s'il y a des tomates vertes à proximité des tomates mûres déjà localisées.

## ALGORITHME I

1. Acquisition et digitalisation d'une image couleur.

2. Calcul d'une image binaire de contours à partir de l'image digitale couleur à l'aide d'un opérateur binaire d'extraction d'élements locaux de contour.

3. Localisation des centroïdes et des rayons des tomates à partir de l'algorithme de la section 6.5.

4. Classification de chaque tomate comme mûre ou non-mûre à partir de la couleur d'un certain nombre de ses pixels.

## ALGORITHME II

1. Acquisition et digitalisation d'une image couleur.

2. Formation d'une image binaire rouge (pixels rouges prenant la valeur "1", les autres la valeur "0").

3. Filtrage de l'image binaire.

4. Segmentation de l'image en segments disjoints contenant une tomate mûre ou un groupe de tomates mûres.

5. Calcul pour chaque segment d'une sous-image binaire de contours à l'aide d'un opérateur binaire d'extraction d'éléments locaux de contour.

6. Localisation des centroïdes et des rayons des tomates pour chaque segment à partir de l'algorithme de la section 6.5.

La première étape de ces deux algorithmes est la même et est réalisée automatiquement par le système de vision. Pour les étapes 2 et 3 du premier algorithme et les étapes 5 et 6 du deuxième algorithme, veuillez vous référer à la section 6.5. L'étape 4 du deuxième algorithme pourraît être réalisée par les étapes 6.1 à 6.3 de l'algorithme de la section 6.4.

De ces deux algorithmes, nous avons retenu le premier pour sa simplicité et pour sa compatibilité avec la méthode de reconnaissance de la couleur de Wandell et Maloney décrite au chapitre V. Pour une description détaillée de l'étape 4 de l'algorithme I, veuillez référer au chapitre VI.

## ALGORITHME II

1. Acquisition et digitalisation d'une image couleur.

2. Formation d'une image binaire rouge (pixels rouges prenant la valeur "1", les autres la valeur "0").

3. Filtrage de l'image binaire.

4. Segmentation de l'image en segments disjoints contenant une tomate mûre ou un groupe de tomates mûres.

5. Calcul pour chaque segment d'une sous-image binaire de contours à l'aide d'un opérateur binaire d'extraction d'éléments locaux de contour.

6. Localisation des centroïdes et des rayons des tomates pour chaque segment à partir de l'algorithme de la section 6.5.

La première étape de ces deux algorithmes est la même et est réalisée automatiquement par le système de vision. Pour les étapes 2 et 3 du premier algorithme et les étapes 5 et 6 du deuxième algorithme, veuillez vous référer à la section 6.5. L'étape 4 du deuxième algorithme pourraît être réalisée par les étapes 6.1 à 6.3 de l'algorithme de la section 6.4.

De ces deux algorithmes, nous avons retenu le premier pour sa simplicité et pour sa compatibilité avec la méthode de reconnaissance de la couleur de Wandell et Maloney décrite au chapitre V. Pour une description détaillée de l'étape 4 de l'algorithme I, veuillez référer au chapitre VI.

# Chapitre IV

# LA PROBLÉMATIQUE DE LA RECONNAISSANCE DE LA COULEUR

## 4.1 PROBLÉMATIQUE GÉNÉRALE

Aucun des algorithmes de reconnaissance visuelle de fruits présentés à la section 2.2 ne permet d'obtenir des performances acceptables par temps ensoleillé, et la solution de n'utiliser le robot que la nuit ou par temps nuageux ne serait sûrement pas viable économiquement pour un système commercial.

La raison de ces mauvaises performances tient d'abord au fait qu'il est très difficile de reconnaître la couleur des objets dans les conditions d'éclairage qui existent dans les serres (ou les vergers) par temps ensoleillé. Voici quelques-unes de ces difficultés. L'intensité et la composition spectrale de la lumière en provenance du soleil varient au cours de la journée en fonction de l'ennuagement, de l'huminidité et de la position du soleil dans le ciel. L'intensité de cet éclairement varie beaucoup d'un endroit à l'autre de la scène, à cause de la présence d'obstacles créant de l'ombre ou de l'orientation des surfaces vis-à-vis le soleil. Et comme la surfaces des feuilles, des tiges et des tomates ne sont pas parfaitement mates, les rayons du soleil sont, pour certaines surfaces réfléchies vers la caméra à la manière d'un miroir, ce qui leur donne une apparence faussement blanchâtre.

La reconnaissance de la couleur des objets dans les conditions d'éclairage difficiles des serres fait partie intégrante des deux algorithmes de localisation des tomates dans une image proposés à la section 3.2. Une solution adéquate à ce problème est donc essentielle et le reste de ce mémoire sera consacré à cette question.

La couleur d'un objet est une perception d'une qualité propre à l'objet qui est reliée à la façon dont celui-ci réfléchit la lumière à l'intérieur du spectre visible. À la section 1.4, on a vu que la

densité spectrale de réflectance normalisée (DSRN), telle que mesurée par un spectrophotomètre, permet de distinguer les tomates mûres (tomates rouges) des tomates non-mûres (tomates vertes).

On adoptera donc la définition suivante: "*la couleur d'un objet est une estimation , faite à partir d'une image, de la densité spectrale de réflectance normalisée (DSRN) de la surface de l'objet* ." Par l'expression "reconnaître la couleur d'un objet " on signifiera "faire une estimation de la DSRN de l'objet.dans le spectre visible".

Nous désignerons la densité spectrale de réflectance d'un point X de la scène par $r(X;\lambda)$; l'estimé de la densité spectrale de réflectance sera désigné par $r(X;\lambda)$ et l'estimé de la densité spectrale de réflectance normalisé (DSRN) (c'est-à-dire la couleur du point X) sera désigné par $r(X;\lambda)$.

L'instrument idéal pour reconnaître la couleur d'un objet est un spectrophotomètre. Cependant, ce dernier nécessite que l'objet soit placé à l'intérieur de l'instrument et, pour la reconnaissance de la couleur des tomates, cette condition est évidemment impossible à remplir. Une alternative serait d'utiliser un "scanner LASER" à plusieurs fréquences dans le visible, jumelé à une caméra dotée d'un filtre ne laissant passer que la lumière polarisée du LASER. Un tel instrument n'est pas encore disponible.

Nous allons voir comment on peut reconnaître la couleur des objets dans des conditions d'éclairage difficiles à partir de la mesure d'une caméra couleur (c'est-à-dire à partir d'une image couleur). Mais avant de proposer une solution à ce problème, nous allons analyser la question de la réflexion de la lumière par les objets ainsi que la question de la formation d'une image par une caméra.

## 4.2 ÉMISSION ET RÉFLEXION DE LA LUMIÈRE

Les objets que l'on voit sont tous des sources lumineuses car, s'il en était autrement, il ne serait pas possible de les voir.

Les sources lumineurses sont:

- soit des sources actives qui ont un bilan d'énergie lumineuse positif, c'est-à-dire qu'elles émettent plus d'énergie lumineuse qu'elles n'en reçoivent; le soleil, le gaz ionisé des tubes fluorescents et le filament incandescent d'une ampoule électrique sont des exemples de sources actives;

- soit des sources passives, c'est-à-dire des sources ayant un bilan d'énergie lumineuse négatif; celles-ci sont de deux types:
    - les objets translucides tels que le verre; ces objets laissent traverser une partie de la lumière incidente en la réfractant, en absorbent une certaine partie et en réfléchissent une autre;
    - les objets opaques qui ne se laissent pas traverser par la lumière; ils réfléchissent une partie de la lumière incidente et absorbent le reste; les objets opaques forment en général la très grande majorité des objets d'une scène.

Figure 8. Densité spectrale de luminance dans la direction $(\Theta_e,\Phi_e)$ d'une source lumineuse infinitésimale [Horn 86]

Considérons un élément de surface infinitésimal d'une source lumineuse (Fig.8); l'émission par cette source infinitésimale des radiations lumineuses dans la direction $(\Theta_e,\Phi_e)$ est caractérisée par sa densité spectrale de luminance dans cette direction, et on notera cette quantité radiométrique

par: $L(\Theta_e,\Phi_e;\lambda)$.

Considérons d'autre part, une surface infinitésimale "da" d'un objet opaque (Fig.9); cette surface reçoit en général de la lumière en provenance de toutes les directions $(\Theta_i,\Phi_i)$ où $0 < \Theta_i < \pi/2$ et où $0 < \Phi_i < 2\pi$. A chaque direction incidente $(\Theta_i,\Phi_i)$ correspond une source lumineuse infinitésimale qui est sous-tendue par un angle solide $d\Omega_{(\Theta_i,\Phi_i)} = \sin\Theta_i\ d\Theta_i\ d\Phi_i$, qui a une densité spectrale de luminance que l'on notera $L_i(\Theta_i,\Phi_i;\lambda)$; on appellera cette dernière fonction **l'illumination ambiante** de "da".

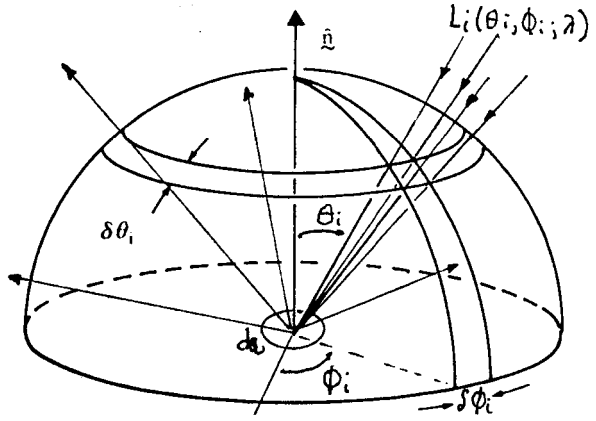



Figure 9. Réflexion de la lumière par un élément de surface infinitésimal d'un objet opaque.

La densité spectrale d'éclairement de "da", que nous noterons $E(\lambda)$, est la somme des densités spectrales d'éclairement $\delta E(\Theta_i,\Phi_i;\lambda)$ produites par la lumière en provenance de chacune des directions incidentes $(\Theta_i,\Phi_i)$. Les équations suivantes relient ces quantités entre elles:

$$\delta E(\Theta_i,\Phi_i;\lambda) = L_i(\Theta_i,\Phi_i;\lambda)\ d\Omega_{(\Theta_i,\Phi_i)} \cos\Theta_i$$

$$\delta E(\Theta_i,\Phi_i;\lambda) = L_i(\Theta_i,\Phi_i;\lambda) \sin\Theta_i \cos\Theta_i \; d\Phi_i \; d\Theta_i \qquad [17]$$

$$E(\lambda) = \int_0^{\pi/2} \int_0^{2\pi} L_i(\Theta_i,\Phi_i;\lambda) \sin\Theta_i \cos\Theta_i \; d\Phi_i \; d\Theta_i \qquad [18]$$

Les objets opaques ne réfléchissent pas tous la lumière de la même façon. Afin de caractériser la réflexion de la lumière par un élément infinitésimal de surface "da" d'un objet opaque (Fig.9), il faut connaître le rapport entre la densité spectrale de luminance de "da" dans la direction $(\Theta_e,\Phi_e)$ et la densité spectrale d'éclairement de "da" due à la lumière incidente en provenance de la direction $(\Theta_i,\Phi_i)$ et ce, pour tous les couples de directions $\{(\Theta_i,\Phi_i),(\Theta_e,\Phi_e)\}$ et pour toutes les longueurs d'onde du spectre visible, c'est-à-dire pour $\lambda \in [\lambda_1,\lambda_2]$:

$$R(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda) = \frac{\delta L(\Theta_e,\Phi_e;\lambda)}{\delta E(\Theta_i,\Phi_i;\lambda)} \qquad [19]$$

où

$R(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda)$ Distribution bidirectionnelle et spectrale de réflectance (DBSR) de l'élément de surface "da".

$\delta E(\Theta_i,\Phi_i;\lambda)$ Densité spectrale d'éclairement de "da" par la lumière en provenance de la direction $(\Theta_i,\Phi_i)$

$\delta L(\Theta_e,\Phi_e;\lambda)$ Densité spectrale de luminance de "da" dans la direction $(\Theta_e,\Phi_e)$ dûe à la réflexion de la lumière incidente en provenance de la direction $(\Theta_i,\Phi_i)$.

Si l'on connaît la DBSR d'un élément de surface infinitésimale "da" d'un objet opaque et que l'on connaît l'illumination ambiante de ce même élément de surface "da", alors on peut déterminer la densité spectrale de luminance de "da" dans toutes les directions $(\Theta_e,\Phi_e)$, à l'aide de l'équation suivante:

$$L(\Theta_e,\Phi_e;\lambda) = \int_0^{\pi/2} \int_0^{2\pi} R(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda)\, L_i(\Theta_i,\Phi_i;\lambda) \sin\Theta_i \cos\Theta_i \, d\Phi_i \, d\Theta_i \qquad [20]$$

Quoique la DBSR constitue un modèle de réflexion complet et général qui décrit le comportement d'un élément de surface "da" d'un objet opaque, vis-à-vis la réflexion de la lumière, c'est, pour le moins qu'on puisse dire, un modèle complexe et c'est pourquoi on doit se résoudre à utiliser des modèles de réflexion beaucoup plus simples tels que:

- le modèle de Cook et Torrance [Cook 1981],
- le modèle standard (ou bichromatique) [Shafer 85],[Tominaga 89],[Lee 86], [D'Zuma 86],
- le modèle de Horn qui est décrit à la section suivante.

## 4.3 LE MODÈLE DE RÉFLEXION DE HORN

La description qui est faite ici du modèle de réflexion de Horn est déduite des informations incomplètes données dans [Horn 86] sur ce modèle. Quoique l'idée du modèle est empruntée à Horn, la formulation qui en est faite et sa dérivation à partir de la DBSR est à notre connaissance originale.

Ce modèle prend comme hypothèse simplificatrice que le comportement de beaucoup d'objets opaques vis-à-vis la réflexion de la lumière oscille entre celui des deux cas extrêmes suivants:

- le réflecteur mat (Fig.10),
- et le miroir parfait (Fig.11).

### 4.3.1 LE RÉFLECTEUR MAT

Un élément de surface infinitésimal "da" d'un objet opaque est un réflecteur mat, s'il réfléchit la lumière incidente dans toutes les directions, de façon à ce que sa luminance soit la même dans toutes les direction et ce, pour n'importe qu'elle illumination ambiante. On dit que le réflecteur mat réfléchit la lumière de façon diffuse.

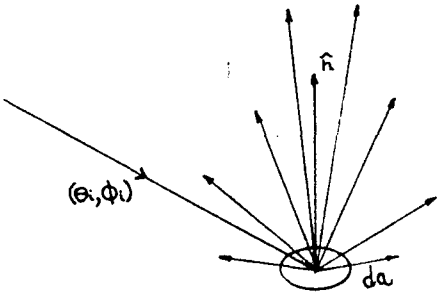



Figure 10. Réflexion de la lumière par un réflecteur mat

En termes mathématiques, un réflecteur mat est un élément infinitésimal de surface "da" dont la DBSR n'est pas fonction des directions incidentes et excitantes, mais seulement fonction de la longeur d'onde:

$$R_{mat}(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda) = r(\lambda) \qquad [21]$$

La DBSR d'un réflecteur mat, qui est notée par r ($\lambda$), est appelée la densité spectrale de réflectance (DSR). Si maintenant nous remplaçons dans l'équation [20], $R(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda)$ par r ($\lambda$), on obtient l'équation [22] où la densité spectrale de luminance de "da" est égale au produit de la densité spectrale d'éclairement de "da" avec la DSR de "da":

$$L_{mat}(\Theta_e,\Phi_e;\lambda) = \int_0^{\pi/2} \int_0^{2\pi} r(\lambda)\ L_i(\Theta_i,\Phi_i;\lambda) \sin\Theta_i \cos\Theta_i\ d\Phi_i\, d\Theta_i$$

$$L_{mat}(\Theta_e,\Phi_e;\lambda) = r(\lambda) \int_0^{\pi/2} \int_0^{2\pi} L_i(\Theta_i,\Phi_i;\lambda) \sin\Theta_i \cos\Theta_i\ d\Phi_i\, d\Theta_i$$

$$L_{mat}(\lambda) = r(\lambda)\ E(\lambda) \qquad [22]$$

Les remarques suivantes s'imposent:

On constate que la densité spectrale de luminance de "da" n'est pas fonction de la direction $(\Theta_e,\Phi_e)$. La luminance de "da" est donc la même dans toutes les directions. On dit d'une source lumineuse infinitésimale ayant cette propriété qu'elle suit la loi de Lambert.

La densité spectrale de luminance de "da" dans une direction $(\Theta_e,\Phi_e)$ n'est pas directement

fonction de la distribution des sources lumineuses autour de "da" (l'illumination ambiante), mais est plutôt fonction de la densité spectrale d'éclairement de "da".

Les surfaces couvertes par une poudre fine d'un matériel transparent tel que le sulfate de barium viennent très près de suivre le modèle de réflexion du réflecteur mat. Les surfaces, que l'on qualifie de mates dans le langage courant, peuvent être considérées en première approximation comme des réflecteurs mats.

### 4.3.2 LE MIROIR PARFAIT

Une surface infinitésimale "da" d'un objet opaque est un miroir parfait, si elle réfléchit toute la lumière visible en provenance de n'importe quelle direction incidente $(\Theta_i,\Phi_i)$ vers la direction $(\Theta_e = \Theta_i, \Phi_e = \Phi_i + \pi)$, ce qui correspond bien à l'idée qu'on se fait d'un miroir. On dit du miroir parfait qu'il réfléchit la lumière de façon directe.

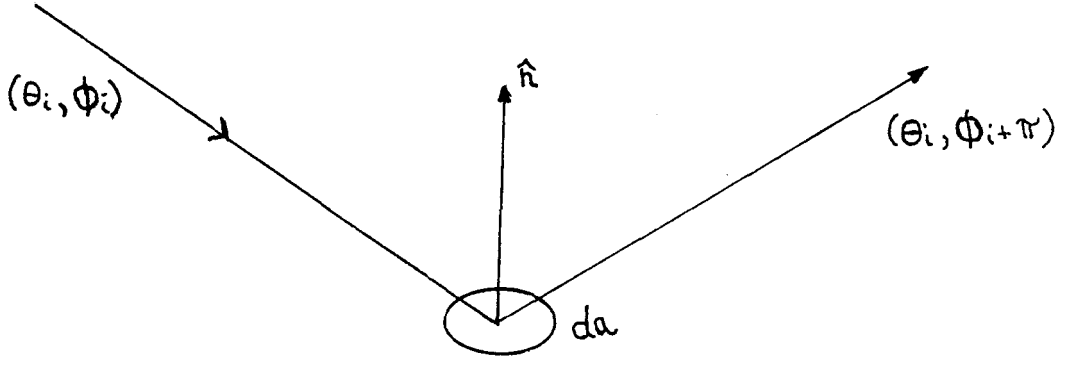



Figure 11. Réflexion de la lumière par un miroir parfait.

En termes plus mathématiques, un miroir parfait est une surface infinitésimale "da" dont la DBSR est de la forme suivante:

$$R_{miroir}(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda) = \frac{\delta(\Theta_e-\Theta_i)\,\delta(\Phi_e-\Phi_i-\pi)}{\sin\Theta_i\,\cos\Theta_i} \qquad [23]$$

Si l'on remplace, dans l'équation [20], $R(\Theta i,\Phi i;\Theta e,\Phi e;\lambda)$ par cette dernière expression alors:

$$L_{miroir}(\Theta_e,\Phi_e;\lambda) = \int_0^{\pi/2}\int_0^{2\pi} \frac{\delta(\Theta_e-\Theta_i)\,\delta(\Phi_e-\Phi_i-\pi)}{\sin\Theta i\,\cos\Theta i}\, L_i(\Theta i,\Phi i;\lambda)\sin\Theta_i\cos\Theta_i\, d\Phi_i\, d\Theta_i$$

$$L_{miroir}(\Theta_e,\Phi_e;\lambda) = \int_0^{\pi/2}\int_0^{2\pi} \delta(\Theta_e-\Theta_i)\,\delta(\Phi_e-\Phi_i-\pi)\, L_i(\Theta_i,\Phi_i;\lambda)\, d\Phi_i\, d\Theta_i$$

$$L_{miroir}(\Theta_e,\Phi_e;\lambda) = L_i(\Theta_i=\Theta_e,\Phi_i=\Phi_e-\pi;\lambda) \qquad [24]$$

### 4.3.3 LE RÉFLECTEUR SEMI-LUSTRÉ

Comme on l'a déjà dit, le modèle de réflexion de Horn prend comme hypothèse simplificatrice que beaucoup d'objets opaques réfléchissent la lumière, en partie de façon diffuse à la manière du réflecteur mat et en partie de façon directe à la manière du miroir parfait. Nous appelerons une surface infinitésimale "da", qui suit ce modèle de réflexion de Horn, un réflecteur semi-lustré.

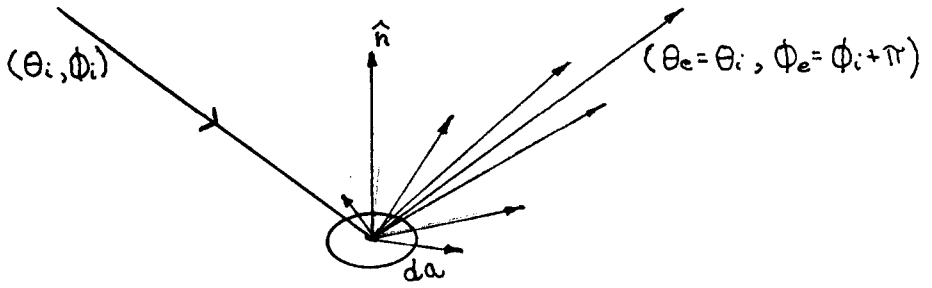



Figure 12. Réflexion de la lumière par un réflecteur semi-lustré

La DBSR d'un réflecteur semi-lustré sera de la forme suivante:

$$R_{S\text{-}L}(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda) = r(\lambda) + k \frac{\delta(\Theta_e-\Theta_i)\,\delta(\Phi_e-\Phi_i-\pi)}{\sin\Theta_i \cos\Theta_i} \qquad [25]$$

où $0 < k < 1$.

Si nous remplaçons dans l'équation [20] $R(\Theta_i,\Phi_i;\Theta_e,\Phi_e;\lambda)$ par cette dernière expression, on obtient après simplification l'expression suivante pour la densité spectrale de luminance du réflecteur semi-lustré:

$$L_{S\text{-}L}(\Theta_e,\Phi_e;\lambda) = r(\lambda)\,E(\lambda) + k\, L_i(\Theta_i=\Theta_e,\Phi_i=\Phi_e-\pi;\lambda) \qquad [26]$$

La densité spectrale de luminance d'un réflecteur semi-lustré a deux composantes: une composante mate et une composante miroir.

$$L_{S\text{-}L}(\Theta e,\Phi e;\lambda) = L_{mat}(\lambda) + L_{miroir}(\Theta e,\Phi e;\lambda)$$

où

$$L_{mat}(\lambda) = r(\lambda)\, E(\lambda)$$

$$L_{miroir}(\Theta_e,\Phi_e;\lambda) = k\ L_i(\Theta_i=\Theta_e,\Phi_i=\Phi_e-\pi) \qquad 0 < k < 1$$

Lors d'une journée ensoleillée à midi, si l'on regarde vers le sud, on peut remarquer que les surfaces, dont le vecteur normal est dans le plan médian entre la direction du soleil et celle de nos yeux, ont plus d'éclat (luminance plus grande) que les autres. On peut expliquer ce phénomène par la présence de la composante miroir de la luminance qui est, pour ces surfaces, égale à k fois celle du soleil.

## 4.4 MODÉLISATION DE LA FORMATION D'UNE IMAGE PAR UNE CAMÉRA

### 4.4.1 FORMATION D'UNE IMAGE OPTIQUE PAR UN SYSTEME OPTIQUE IDÉAL

L'existence des matériaux translucides, tels que le verre, qui réfractent et donc dévient la lumière qui les traverse, a permis la réalisation de systèmes optiques. Ces dispositifs permettent de faire converger en un point d'un plan (plan image) la lumière en provenance d'une source lumineuse infinitésimale de la scène, et qui est incidente à l'ouverture du système optique. Une lentille mince est un exemple d'un système optique simple.

Au lieu de considérer un système optique réel particulier, nous allons plutôt considérer un modèle simple qui représente assez bien un système optique de qualité; on appellera ce modèle simplifié: "un système optique idéal". Notre système optique idéal aura:

- une résolution infinie,
- une géométrie de convergence des rayons lumineux qui est une projection orthographique dans le plan image par rapport au point central de l'ouverture;
- une profondeur de champ infinie,
- aucune aberration chromatique,
- aucune diffraction,
- une transmissibilité spectrale $f_o(\lambda)$ qui n'est pas fonction de l'angle d'incidence de la lumière,
- une linéarité parfaite à l'intérieur d'une certaine plage dynamique.

On présente à la figure suivante les différentes parties du système optique idéal.

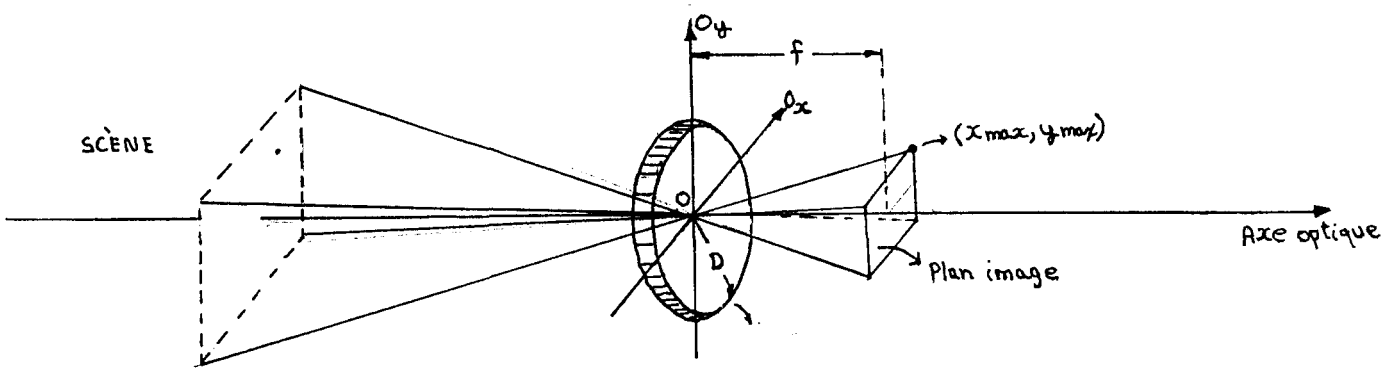
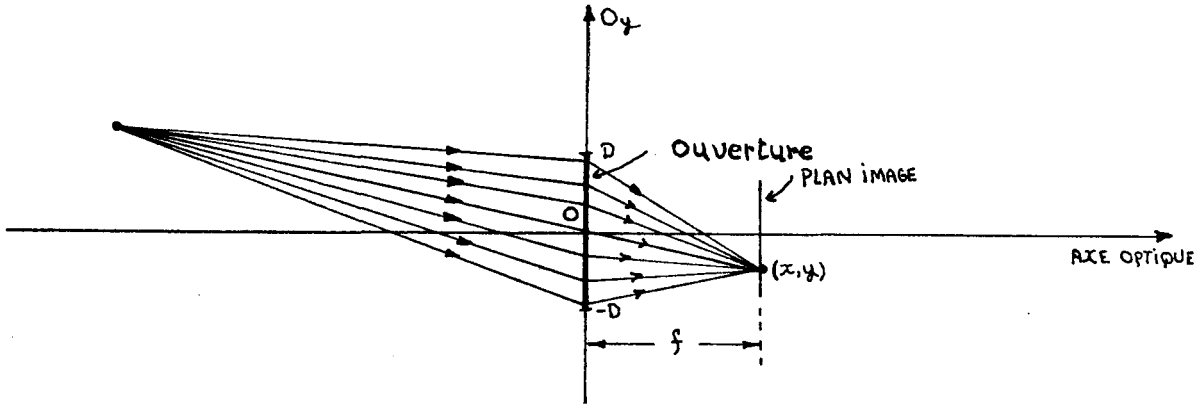



Figure 14. Description du système optique idéal.

Considérons (Fig.15) une source lumineuse infinitésimale située dans la direction de la droite OX passant par le point origine (centre de l'aperture) et le point (x,y,f) du plan image. En autant que $z < z_{min} < 0$ et que $|x| < x_{max}$ et que $|y| < y_{max}$ notre système optique idéal permettra de faire converger la lumière émise par cette source en direction de l'aperture vers le point (x,y,f).




Firure 15. Formation d'une image optique par un système optique idéal.

Ainsi la lumière qui éclaire le point X = (x,y) du plan optique provient de la direction $(\theta_X,\phi_X)$ où

$$\theta_X = \arctan \frac{\sqrt[2]{x^2 + y^2}}{f} \qquad [27]$$

$$\phi_X = \begin{cases} \arctan y/x & \text{si } x \geq 0 \\ \arctan y/x + \pi & \text{si } x < 0 \end{cases} \qquad [28]$$

Si les hypothèses ci-dessus sont vérifiées, on peut montrer [HORN 86], [LEE 90] que la densité spectrale d'éclairement en chaque point X du plan image du système optique idéal est reliée à la densité spectrale de luminance de la scène dans la direction OX, ou X, ou $(\theta_X,\phi_X)$ , de la façon suivante:

$$E(X;\lambda) = \pi/4 \ (D/f)^2 \ \cos^4 \theta_{(x,y)} \ f_o(\lambda) \ L_c(X;\lambda) \qquad [29]$$

où

| | |
|---|---|
| $E(X;\lambda)$ | Densité spectrale d'éclairement du point X = (x,y) du plan image. |
| $L_c(X;\lambda)$ | Densité spectrale de luminance dans la direction $(\theta_X,\phi_X)$ de la scène vue à partir du point O. |
| f | Distance entre le plan image et l'ouverture du système optique. |
| D | Diamètre de l'ouverture du système optique. |
| $f_o(\lambda)$ | Densité spectrale de transmittance du système optique. |

On appelle "image optique" la fonction $E(X;\lambda)$ qui associe à chaque point X du plan image la densité spectrale d'éclairement en ce point .

L'équation [29] indique qu'un système optique idéal est un dispositif permettant de former une image optique $E(X;\lambda)$ proportionnelle à l'illumination ambiante $L_c(X;\lambda)$ du point central de son ouverture.

### 4.4.2 MESURE D'UNE IMAGE OPTIQUE PAR UN RÉCEPTEUR D'IMAGES IDÉAL

Un récepteur de radiations lumineuses est un dispositif qui fait une mesure de la densité spectrale d'éclairement que reçoit une certaine surface réceptrice.

Il y a plusieurs types de récepteurs de radiations qui se différencient selon la nature physique de la mesure. Le récepteur photo-électrique traduit la réception de la lumière par un courant électrique, le récepteur photochimique par une réaction chimique (noircissement d'une plaque photographique), le récepteur thermique par un dégagement de chaleur.

Un récepteur de radiation effectuant une mesure scalaire sera appelé un récepteur monospectral et un récepteur de radiations effectuant une mesure vectorielle, c'est-à-dire N mesures scalaires, sera appelé un récepteur multispectral. Un récepteur multispectral peut être vu comme un ensemble de N récepteurs monospectraux partageant la même surface réceptrice.

Un récepteur monospectral idéal est caractérisé mathématiquement par une fonction $f(\lambda)$ que l'on appellera la "sensibilité spectrale du récepteur". Cette fonction est en général une fonction scalaire réelle, bornée et continue, et dont le support (domaine spectral) est borné. Puisque nous ne sommes intéressés qu'à la lumière visible, nous ne considérerons que les récepteurs dont le domaine spectral de sensibilité est limité au spectre visible, c'est-à-dire à l'intervalle suivant $[\lambda_1,\lambda_2]$.où $\lambda_1 = 380$ nm et où $\lambda_2 = 770$ nm.

Les fonctions de sensibilité spectrale, ainsi que les fonctions décrivant les densités spectrales d'éclairement en un point de la surface réceptrice d'un récepteur de radiation, seront considérées comme des vecteurs de l'espace de Hilbert $C[\lambda_1,\lambda_2]$.

La mesure $I \in \Re$ faite par un récepteur monospectral idéal, de sensibilité spectrale $f(\lambda)$ donnée, est définie comme étant:

$$I = 1/(t_2-t_1) \int_{t_1}^{t_2} 1/S \int_{S} \left[ \int_{\lambda_1}^{\lambda_2} E(X;\lambda;t)\, f(\lambda)\, d\lambda \right] dS\ dt \qquad [30]$$

si $\|E(X;\lambda;t)\|_{max} \in [0, L]$

où

| | |
|---|---|
| $[t_1,t_2]$ | Période de mesure |
| $[\lambda_1, \lambda_2]$ | Spectre visible = [380 nm, 770 nm] |
| [0, L] | Plage dynamique du récepteur de radiation, c'est-à-dire la plage d'intensités pour laquelle la mesure du récepteur est linéaire |
| S | Surface réceptrice du récepteur monospectral |
| X | Vecteur position sur la surface réceptrice |
| $E(X;\lambda;t)$ | Densité spectrale d'éclairement au temps t et au point X de la surface réceptrice |

Dans la mesure où la surface réceptrice du récepteur est suffisamment petite, que la période de mesure est suffisamment courte et que l'éclairement n'est pas trop grand, il est raisonnable de faire les hypothèses suivantes:

- la densité spectrale d'éclairement est constante dans le temps pendant la période de mesure,
- la densité spectrale d'éclairement ne varie pas sur S,
- et $\|E(X;\lambda;t)\|_{max} \in [0, L]$.

Comme, selon ces hypothèses, la fonction de densité spectrale d'éclairement est constante par rapport à X et à t, nous allons tout simplement la noter $E(\lambda)$ et on pourra simplifier l'équation [30] de la façon suivante:

$$I = \left[ \int_{\lambda_1}^{\lambda_2} E(\lambda)\, f(\lambda)\, d\lambda \right] \quad 1/(t_2-t_1) \int_{t_1}^{t_2} dt \quad 1/S \int_S dS$$

$$I = \int_{\lambda_1}^{\lambda_2} E(\lambda)\, f(\lambda)\, d\lambda$$

$$I = \langle E(\lambda) \mid f(\lambda) \rangle \qquad [31]$$

Ainsi la mesure du récepteur monospectral idéal correspond au produit scalaire dans l'espace de Hilbert $C[\lambda_1, \lambda_2]$ entre la fonction de densité spectrale d'éclairement sur la surface réceptrice et la fonction de sensibilité spectrale du récepteur.

La mesure du récepteur multispectral idéal est un vecteur $\mathbf{I} = \{ I_i \in \Re \}_{i \leq N} \in \Re^N$ dont chaque composante $I_i$ est égale à la mesure $\langle E(\lambda) \mid f_i(\lambda) \rangle$ faite par le $i^e$ récepteur monospectral idéal de sensibilité spectrale $f_i(\lambda)$. La fonction de sensibilité spectrale du récepteur multispectral correspond à la famille de fonction suivante: $\mathbf{f} = \{ f_i \in C[\lambda_1, \lambda_2] \}_{i \leq N}$ et la mesure $\mathbf{I} \in \Re^N$ de ce récepteur multispectral idéal est:

$$\mathbf{I} = \mathbf{f}\langle E(\lambda) \rangle = \{ \langle E(\lambda) \mid f_i(\lambda) \rangle \}_{i \leq N} \qquad [32]$$

Nous sommes maintenant en mesure de définir le **récepteur d'images idéal.** Un récepteur d'images idéal est un dispositif permettant de mesurer la densité spectrale d'éclairement en chaque point X d'une surface appelée le plan image. Si l'on considère un élément de surface dS situé à

proximité d'un point X du plan image, alors la mesure en ce point du récepteur d'images idéal sera celle que donnerait un récepteur de radiations idéal dont la surface réceptrice serait dS et dont la fonction de sensibilité spectrale $\mathbf{f} = \{ f_i \in C[\lambda_1, \lambda_2] \}_{i \leq N}$. Un récepteur d'images idéal sera dit monospectral ou multispectral selon qu'il a une (N=1) ou plusieurs (N > 1) bandes de sensibilité spectrale.

La fonction de densité spectrale d'éclairement définie sur le plan image sera notée $E(X;\lambda)$ et sera également appelée l'image optique. Nous allons supposer que $E(X;\lambda)$ est constant pendant l'intervalle de temps de mesure et ce, en chaque point X du plan image. Nous allons également supposer que:

$\| E(X;\lambda;t) \|_{max} \in [0, L]$.

Dans ces conditions, la mesure $\mathbf{I(X)} \in \Re^N$, qui est faite en chaque point X du plan image par le récepteur d'images idéal de sensibilité spectrale $\mathbf{f} = \{ f_i \in C[\lambda_1, \lambda_2] \}_{i \leq N}$, sera reliée à $E(X;\lambda)$ et à $\mathbf{f}$ par l'équation suivante:

$$\mathbf{I(X)} = \mathbf{f}\langle E(X;\lambda) \rangle = \{ \langle E(X;\lambda) \mid f_i(\lambda) \rangle \}_{i \leq N} \qquad [33]$$

Dans une caméra, le plan image du récepteur d'images correspond au plan image du système optique.

### 4.4.3 MESURE DE L'ILLUMINATION AMBIANTE D'UN POINT PAR UNE CAMÉRA IDÉALE

Une caméra idéale est un dispositif comprenant un système optique idéal, un récepteur d'images idéal, un système de correction spatiale de la mesure du récepteur d'images, et enfin, un système de correction de la couleur (Fig.16).

Soit $L_c(X,\lambda)$ la densité spectrale de luminance de la scène vue à partir de la caméra (c'est-à-dire vue à partir du point central de l'ouverture du système optique de la caméra). Le système optique permet de former sur le plan optique de la caméra une image optique $E(X,\lambda)$ proportionnelle à $L_c(X,\lambda)$. Le récepteur d'images idéal permet ensuite de mesurer cette image optique. Cette mesure est ensuite corrigée pour compenser la variation de sensibilité du système optique en fonction de X. Cette correction est faite simplement par la multiplication de la mesure par le facteur $1/\cos^4\theta_{X)}$ Finalement, chacune des composantes $I_i(X)$ de la mesure est multipliée par un facteur $k_{ii}$. Nous verrons plus loin comment ces N facteurs $k_{ii}$ sont déterminés lors de la calibration de la caméra.

On parlera de caméra monochrome idéale ou de caméra couleur idéale selon que le récepteur d'images de la caméra est monospectral (N=1) ou multispectral.(N>1). Le signal de sortie d'une caméra sera appelé l'image de la scène. On parlera d'une image monochrome ou d'une image couleur selon que la caméra est monocrhrome ou couleur.

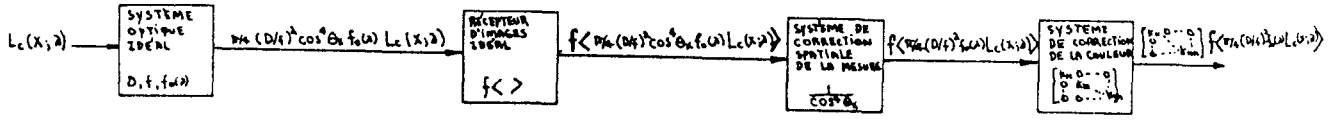



Figure 16. Les différents sous-systèmes constitutifs d'une caméra idéale.

La mesure $\mathbf{I_c}(X) \in \Re^N$, effectuée par une caméra idéale en chaque point X de la scène, est donnée par l'expression suivante:

$$\mathbf{I_c}(X) = \mathbf{K}\ 1/\cos^4\theta_X.\ \mathbf{f}\langle\ \pi/4\ (D/f)^2\ \cos^4\theta_X.\ f_o(\lambda)\ L_c(X,\lambda)\ \rangle \qquad [34]$$

où

**K** Matrice dont la diagonale est formée par les coefficients $k_{ii}$ et dont tous les autres éléments sont nuls

$\theta_X$. Cet angle est donné par l'équation [27]

**f** Sensibilité spectrale du récepteur d'images idéal

$f_o(\lambda)$ Densité spectrale de transmittance du système optique idéal de la caméra

D Diamètre de l'ouverture du système optique idéal de la caméra

f Distance entre le plan image et l'ouverture du système optique de la caméra

$L_c(X,\lambda)$Densité spectrale de la scène dans la direction X par rapport au point central de l'ouverture

L'équation [34] peut être simplifiée de la façon suivante:

$$\mathbf{I_c}(X) = \mathbf{K}\ 1/\cos^4\theta_{(X)}.\ \{\ \langle\ \pi/4\ (D/f)^2\ \cos^4\theta_{(X)}\ f_o(\lambda)\ L_c(X,\lambda)\ |\ f_i(\lambda)\ \rangle\ \}_{i\le N} \qquad [35]$$

$$\mathbf{I_c}(X) = \{\ \langle\ L_c(X,\lambda)\ |\ k_{ii}\ \pi/4\ (D/f)^2\ f_o(\lambda)f_i(\lambda)\ \rangle\ \}_{i\le N} \qquad [36]$$

Si maintenant nous définissons la sensibilité spectrale de la caméra idéale comme étant

$$\mathbf{f_c} = \{f_{ci}(\lambda)\}_{i\le N} = \{\ k_{ii}\ \pi/4\ (D/f)^2\ f_o(\lambda)\ f_i(\lambda)\ \}_{i\le N} \qquad [37]$$

on peut exprimer de façon très élégante la relation qui existe entre le signal d'entrée et le signal de sortie de la caméra idéale, c'est-à-dire entre la densité spectrale de luminance de la scène à l'entrée de la caméra, et l'image de la scène donnée par la caméra:

$$\mathbf{I_c}(X) = \mathbf{f_c}\langle L_c(X,\lambda)\rangle \qquad [38]$$

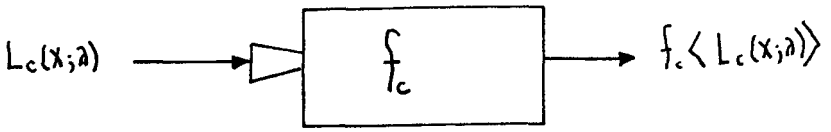



Figure 17. Signaux d'entrée et de sortie d'une caméra idéale.

N.B. *Les équations* [34],[35],[36] *et* [38] *sont toutes équivalentes.*

Disons maintenant quelques mots sur la manière dont les coefficients $k_{ii}$ de correction de la couleur sont fixés. La plupart des caméras ont un dispositif permettant de fixer ces coefficients.de la façon suivante: lorsque l'on pèse sur un bouton appelé "white balance", la caméra fixe les coefficients $k_{ii}$ de la façon suivante: $k_{ii} = 1/\langle L_C(0,0;\lambda) / \pi/4 \ (D/f)^2 f_o(\lambda) f_i(\lambda)\rangle$. La densité spectrale de luminance de la partie de la scène vers où la caméra pointe lorsque le bouton "white balance" est pressé,soit $L_C(0,0;\lambda)$, devient le blanc de référence de la caméra que l'on notera $B(\lambda)$. Tous les points X de la scènes tel que $L_C(X,\lambda) = B(\lambda)$ auront tous comme mesure $I_C(X) = I_N$.

## 4.5 IRRÉVERSIBILITÉ DU PROCESSUS DE MESURE D'UNE CAMÉRA IDÉALE

Comme on l'a vu à la section précédente, le processus de mesure par une caméra idéale de la densité spectrale de luminance d'une scène est exprimé par l'équation suivante:

$$\mathbf{I_c}(X) = \mathbf{f_c}\langle L_c(X;\lambda)\rangle = \{ \langle L_c(X;\lambda) \mid f_{ci}(\lambda) \rangle \}_{i \le N} \qquad [38]$$

Si ce processus de mesure était réversible il serait possible, connaissant $\mathbf{I_c}(X)$ et $\mathbf{f_c}$, de déterminer $L_c(X;\lambda)$. Ce processus de mesure serait réversible si et seulement si la transformation $\mathbf{f_c}\langle\,\rangle$: $C[\lambda_1, \lambda_2] \rightarrow \Re^N$ était réversible. Mais pour un entier N fini, la transformation $\mathbf{f_c}\langle\,\rangle$ ne peut pas être bijective, étant donné que l'espace $C[\lambda_1, \lambda_2]$ est de dimension infinie et que $\Re^N$ est de dimension N. Ainsi, le processus de mesure par une caméra idéale est toujours irréversible. En d'autres mots, il y a une perte inhérente d'informations spectrales lorsque l'on fait une mesure avec une caméra idéale.

On remarque que X n'intervient pas dans le processus de mesure et c'est pourquoi nous allons considérer la fonction $L_c(X;\lambda)$ comme un vecteur de l'espace $C[\lambda_1, \lambda_2]$.

Tâchons maintenant de déterminer quelle est l'information sur $L_c(X;\lambda)$ qui est perdue lors de la mesure et quelle est l'information sur $L_c(X;\lambda)$ qui est conservée.

Supposons que $\mathbf{f_c} = \{ f_{ci}(\lambda) \in C[\lambda_1, \lambda_2] \}_{i \le N}$ est une famille de fonctions linéairement indépendantes.

D'après les théorèmes 1 à 6 de l'annexe I, la famille de fonctions $\mathbf{f_c}$ détermine une décomposition orthogonale de l'espace $C[\lambda_1,\lambda_2]$ en deux sous-espaces supplémentaires orthogonaux de la façon suivante:

$$C[\lambda_1, \lambda_2] = [\mathbf{f_c}] \oplus \mathbf{f_c}^{\perp} \qquad [39]$$

où

$[f_c]$ Espace vectoriel généré par la famille $f_c$ (théorème 4)

$f_c^{\perp}$ Espace vectoriel orthogonal aux vecteurs de la famille $f_c$ (définition 3).

Et toujours d'après ces théorèmes, tout vecteur $L_c(X;\lambda)$ de $C[\lambda_1, \lambda_2]$ peut s'écrire de façon unique comme une somme:

$$L_c(X;\lambda) = L_{c_{[f_c]}}(X;\lambda) + L_{c_{f_c^{\perp}}}(X;\lambda) \qquad [40]$$

où $$L_{c_{[f_c]}}(X;\lambda) = \text{Proj}_{[f_c] \,//\, f_c^{\perp}}[\, L_c(X;\lambda)\,] \quad \in [f_c] \qquad [41]$$

$$L_{c_{f_c^{\perp}}}(X;\lambda) = \text{Proj}_{f_c \,//[f_c]^{\perp}}[\, L_c(X;\lambda)\,] \quad \in f_c^{\perp} \qquad [42]$$

Regardons à nouveau ce qui se passe lors de la mesure de $L_c(X;\lambda)$ par une caméra idéale de sensibilité spectrale $f_c$. Nous pouvons facilement vérifier que:

$$f_c \left\langle L_{c_{f_c^{\perp}}}(X;\lambda) \right\rangle = \mathbf{0}_N \quad \text{pour toutes les fonctions } L_{c_{f_c^{\perp}}}(X;\lambda) \in f_c^{\perp} \qquad [43]$$

et que:

$$I(X) = f_c \left\langle L_c(X;\lambda) \right\rangle \qquad [38]$$

$$I(X) = f_c \left\langle L_{c_{[f_c]}}(X;\lambda) + L_{c_{f_c^{\perp}}}(X;\lambda) \right\rangle$$

$$I(X) = f_c \left\langle L_{c_{[f_c]}}(X;\lambda) \right\rangle + f_c \left\langle L_{c_{f_c^{\perp}}}(X;\lambda) \right\rangle$$

$$I(X) = f_c \left\langle L_{c_{[f_c]}}(X;\lambda) \right\rangle + \mathbf{0}_N \qquad [44]$$

Les trois définitions suivantes sont une généralisation des définitions données à ces termes par [Wyszecky 82] dans le contexte de l'étude du système visuel humain.

– $L_{c_{[f_c]}}(X;\lambda)$ est appelée la **composante fondamentale** de $L_c(X;\lambda)$ relativement à la caméra idéale de sensibilité spectrale $f_c$.

– $L_{c_{f_c}^{\perp}}(X;\lambda)$ est appelée la **composante résiduelle** de $L_c(X;\lambda)$ relativement à la caméra idéale de sensibilité spectrale $f_c$.

– On dira de deux fonctions $L_c(X_1;\lambda)$ et $L_c(X_2;\lambda)$ qu'elles sont **métamères** relativement à la caméra idéale de sensibilité spectrale $f_c$ si et seulement si elles ont la même composante fondamentale relativement à cette caméra.

D'après l'équation [40], toute fonction $L_c(X;\lambda) \in C[\lambda_1, \lambda_2]$ est la somme de sa composante fondamentale et de sa composante résiduelle. Les équations [43] et [44] démontrent que la composante résiduelle de $L_c(X;\lambda)$ ne contribue en rien à la mesure $\mathbf{I}(X)$ de la caméra et c'est pourquoi on peut considérer que cette composante résiduelle est irrémédiablement perdue lors du processus de mesure de $L_c(X;\lambda)$.

La composante fondamentale de $L_c(X;\lambda)$ est conservée lors du processus de mesure et d'après les théorèmes 7 et 8 de l'annexe I, on peut la retrouver à partir de la mesure $\mathbf{I}(X)$ de la façon suivante:

$$L_{c_{[f_c]}}(X;\lambda) = f_c^{\,T}\, G^{-1}(f_c)\, \mathbf{I}(X) \qquad [45]$$

où

$G^{-1}(f_c) \in \Re^{NxN}$ Inverse de la matrice de Gram de la famille $f_c$.

$G^{-1}(f_c)\, \mathbf{I}(X) \in \Re^{N}$ Coordonnées du vecteur $L_{c_{[f_c]}}(X;\lambda)$ relatives à la base $f_c$ du sous-espace $[f_c]$

Il est facile de démontrer que l'ensemble des fonctions $L_c(X;\lambda)$ de $C[\lambda_1, \lambda_2]$, qui sont métamères relativement à la caméra idéale de sensibilité spectrale $f_c$, est la variété linéaire $V$ suivante:

$$V = L_{c\,[f_c]}(X;\lambda) + f_c^{\perp} \qquad [46]$$

et que la composante fondamentale de $L_c(X;\lambda)$, c'est-à-dire le vecteur $L_{c\,[f_c]}(X;\lambda)$ est le vecteur de norme minimale appartenant à cette variété linéaire.

Connaissant la mesure **I**(X) de la caméra idéale de sensibilité spectrale $f_c$, la densité spectrale de luminance de la scène dans la direction X appartiendra à la variété linéaire $V$, c'est-à-dire que:

$$L_c(X;\lambda) \in V = f_c^{T}\, G^{-1}(f_c)\, \mathbf{I}(X) + f_c^{\perp} \qquad [47]$$

## 4.6 RÉDUCTION DU PROBLÈME AU CAS DES SURFACES MATES

Prenant pour hypothèse que les feuilles et les tomates suivent le modèle de réflexion de Horn et qu'elles peuvent être considérées comme des réflecteurs semi-lustré, on peut démontrer que, pour un réflecteur semi-lustré, la densité spectrale de réflectance normalisée, telle que mesurée par un spectrophotomètre, est égale, à un facteur multiplicatif près, à la densité spectrale de réflectance $r(\lambda)$ telle que définie par le modèle de Horn. Dans ces conditions, le problème de la reconnaissance de la couleur des tomates se ramène à celui de l'estimation (à un facteur multiplicatif près) de la densité spectrale de réflectance $r(\lambda)$.

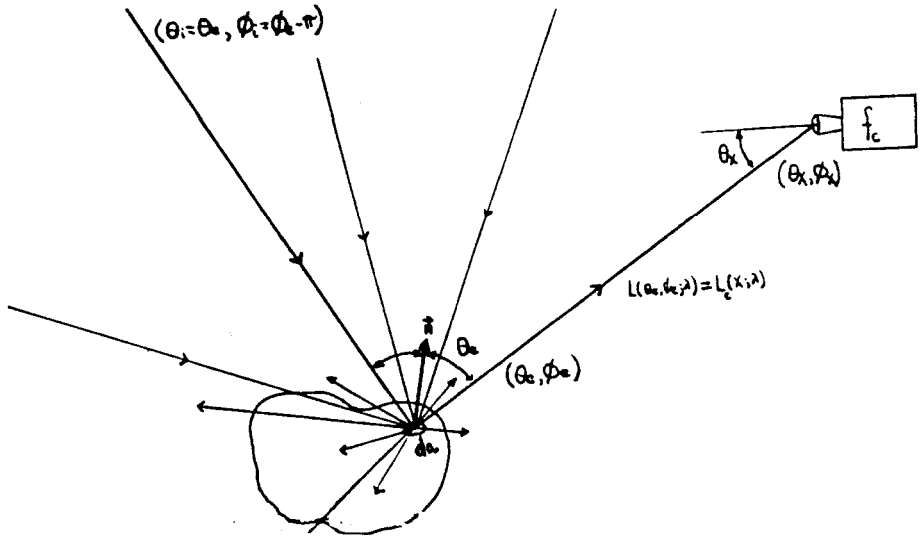



Figure 18. Élément de surface infinitésimale d'un réflecteur semi-lustré placé dans le champ de vision d'une caméra

Considérons (Fig.18) un élément de surface infinitésimale "da" d'un réflecteur semi-lustré placé dans le champ de vision d'une caméra couleur idéale (N>1). La caméra est située dans la direction $(\Theta_e,\Phi_e)$ par rapport à "da" et la densité spectrale de luminance de "da" en direction de la caméra sera désignée par $L(\Theta_e,\Phi_e;\lambda)$.

Rappelons ici l'équation [26] qui relie la densité spectrale de luminance d'un réflecteur semi-lustré à sa densité spectrale d'éclairement $E(\lambda)$ et à son illumination ambiante $L_i(\Theta_i,\Phi_i;\lambda)$:

$$L(\Theta_e,\Phi_e;\lambda) = r(\lambda)\,E(\lambda) + k\,L_i(\Theta_i=\Theta_e,\Phi_i=\Phi_e-\pi;\lambda) \qquad [26]$$

Supposons que la caméra est placée dans la direction $(\Theta_e,\Phi_e)$ par rapport au système de coordonnées local à la surface "da" et que "da" est située à la position X dans le système de coordonnées de la caméra. Dans ces conditions, la densité spectrale de luminance de la scène (telle que vue par la caméra) dans la direction de "da" ($L_c(X;\lambda)$) est par définition égale à la densité spectrale de luminance de "da" dans la direction de la caméra ($L(\Theta_e,\Phi_e;\lambda)$):

$$L_c(X;\lambda) = L(\Theta_e,\Phi_e;\lambda).$$

L'équation [26], par rapport à la caméra, peut maintenant être écrite de la façon suivante:

$$L_c(X;\lambda) = r(X;\lambda)\,E(X;\lambda) + k(X)\,L_i(X;\lambda) \qquad [48]$$

où

| | |
|---|---|
| $r(X;\lambda)$ | Densité spectrale de réflectance de la surface située dans la direction X |
| $E(X;\lambda)$ | Densité spectrale d'éclairement de la surface située dans la direction X |
| $k(X)$ | Coefficient qui caractérise le lustre de la surface située dans la direction X |
| $L_i(X;\lambda)$ | Densité spectrale de luminance dans la direction $(\Theta_i=\Theta_e,\Phi_i=\Phi_e-\pi)$ par rapport au système de coordonnées local à la surface de l'objet qui est situé dans la direction X. |

Si la sensibilité spectrale de la caméra est $\mathbf{f_c} = \{ f_{ci} \in C[\lambda_1, \lambda_2] \}_{i \le N}$, alors la mesure de la caméra au point X de l'image correspondant à ce réflecteur semi-lustré sera:

$$\mathbf{I_c}(X) = \mathbf{f_c}\langle L_c(X;\lambda)\rangle \qquad [38]$$

$$\mathbf{I_c}(X) = \mathbf{f_c}\langle r(X;\lambda)\,E(X;\lambda) + k(X)\,L_i(X;\lambda)\rangle$$

$$\mathbf{I_c}(X) = \mathbf{f_c}\langle r(X;\lambda)\,E(X;\lambda)\rangle + \mathbf{f_c}\langle k(X)\,L_i(X;\lambda)\rangle$$

$$\mathbf{I_c}(X) = \mathbf{I_c}(X)_{mat} + \mathbf{I_c}(X)_{miroir} \qquad [49]$$

où $\quad \mathbf{I_c(X)_{mat}} = \quad \mathbf{f_c}\langle\, r\,(X;\lambda)\, E(X;\lambda)\rangle$ [50]

$\mathbf{I_c(X)_{miroir}} = \quad \mathbf{f_c}\langle\, L_{miroir}(X;\lambda)\rangle$ [51]

Ainsi la mesure d'une caméra en un point X de la scène, correspondant à une réflecteur semi-lustré, peut être considérée comme étant formée de deux composantes: une composante mate et une composante miroir. Toute l'information concernant la densité spectrale de réflectance du point X de la scène est contenue dans la composante mate de l'image.

Pour les points mats de la scène, c'est-à-dire les points pour lesquels la composante miroir est négligeable par rapport à la composante mate, on a:

$$\mathbf{I_c(X)} = \mathbf{I_c(X)_{mat}} = \mathbf{f_c}\langle\, r\,(X;\lambda)\, E(X;\lambda)\rangle \quad [52]$$

Aux points X de la scène, pour lesquels la composante miroir de l'image n'est pas négligeable et pour lesquels la mesure $\mathbf{I_c(X)}$ est donnée par l'équation [49], l'estimation de $r\,(X;\lambda)$ est un sérieux problème. Pour ces points de l'image, la première phase de la reconnaissance de la couleur sera l'élimination de la composante miroir de l'image. [Shafer 85] a mis au point un algorithme d'élimination du lustre qui est basé sur le modèle de réflexion bichromatique (ou standard). Cependant ce modèle de réflection ne peut pas être appliquée dans le cas présent étant donné qu'il n'est valable que dans le cas où l'illumination ambiante a même composition spectrale (à un facteur près) dans toutes les directions incidentes. Dans une serre, par temps ensoleillée ce n'est pas le cas (l'illumination ambiante en provenance du ciel s'additionne en proportion différentes avec celle du soleil selon la direction). Il n'existe pas à notre connaissance d'algorithme d'élimination du lustre basé sur le modèle de réflexion de Horn.

Pour le cas de la reconnaissance de la couleur des tomates un algorithme de reconnaissance de la couleur, qui ne serait efficace qu'aux points mats de la scène, pourrait être acceptable à la condition qu'un nombre suffisant de points mats soit présent sur chaque tomate. Considérons un point mat X de la scène pour lequel la densité spectrale d'éclairement $E(X;\lambda)$ est connue, alors

l'estimation de la densité spectrale de réflectance $r(X;\lambda)$ en ce point de la scène revient à l'estimation de la densité spectrale de luminance $L_c(X;\lambda)$ de ce point. En effet , si l'on connaît $L_c(X;\lambda)$ et $E(X;\lambda)$, l'équation suivante nous donne $r(X;\lambda)$:

$$r(X;\lambda) = L_c(X;\lambda) / E(X;\lambda) \qquad [53]$$

Or le problème de l'estimation de $L_c(X;\lambda)$ à partir de la mesure $I_c(X)$ est relativement simple et [Healey 88] a proposé une méthode à cet effet.

Cependant les conditions d'éclairage qui prévalent le jour dans les serres sont très variables dans le temps et varient beaucoup sur la scène, de sorte qu'on ne connaît pas à priori la valeur de $E(X;\lambda)$ en chaque point X de la scène. Il existe un certain nombre de méthodes de reconnaissance de la couleur qui font une estimation de la densité spectrale d'éclairement sur la scène. Cette estimation demande un certain nombre de calculs, et c'est pourquoi ces méthodes sont souvent appelées des méthodes "computationnelles" de reconnaissance de la couleur. Nous croyons que ce type de méthode est tout à fait approprié à la reconnaissance de la couleur des tomates dans les conditions d'éclairage difficiles des serres. Nous avons considéré un certain nombre de ces méthodes "computationnelles" de reconnaissance de la couleur : [Land 63-86], [Buchsbaum 80], [Wandell 86] et il serait trop long de les décrire ici. De ces méthodes, c'est la méthode de Wandell et Maloney qui nous est apparue comme la plus apte à être appliquée au problème de la reconnaissance de la couleur des tomates. Le prochain chapitre sera donc consacré à l'analyse de cette méthode.

# Chapitre V

# LA MÉTHODE DE RECONNAISSANCE DE LA COULEUR DE WANDELL ET MALONEY

La méthode de reconnaissance de la couleur de Wandell et Maloney permet d'estimer la densité spectrale de réflectance normalisée (DSRN) en chaque point d'une scène, à partir d'une image couleur de la scène. Peu de méthodes de reconnaissance de la couleur sont efficaces dans les conditions d'éclairage variables des serres. Parmi celles-ci, la méthode de reconnaissance de Wandell et Maloney nous est apparue comme étant la plus prometteuse. Cette méthode est nouvelle et n'a jamais, à notre connaissance, été utilisée dans une application pratique. C'est une méthode complexe qui doit nécessairement faire l'objet d'une adaptation particulière au contexte pour lequel on entend l'utiliser. Ce chapitre fait une analyse complète de cette méthode, fait ressortir ses limitations ainsi que les possibilités de son adaptation à la reconnaissance de la couleur des tomates[6].

## 5.1 NORMALISATION DES DENSITÉS SPECTRALES DE RÉFLECTANCE ET DES DENSITÉS SPECTRALES D'ÉCLAIREMENT

Il y a plusieurs façons de normaliser les estimés des densités spectrales de réflectance et d'éclairement en un point X de la scène. Désignons respectivement ces quantités par $\tilde{r}(X;\lambda)$ et par $\tilde{E}(X;\lambda)$ et leurs valeurs normalisées par $\hat{r}(X;\lambda)$ (ou DSRN) et par $\hat{E}(X;\lambda)$ (ou DSEN). Nous

---

6 Il existe un logiciel écrit par ces chercheurs, appelé le "Stanford Color Analysis Package" et qui utilise cet algorithme de reconnaissance de la couleur. Ce logiciel est écrit en C et peut être exécuté sur un ordinateur dont le système d'exploitation est UNIX BSD 4.2.

avons normaliser ces quantités relativement à leur norme d'ordre 1[7] [Healey 88].

$$\hat{r}(X;\lambda) = \frac{\tilde{r}(X;\lambda)}{\|\tilde{r}(X;\lambda)\|_1} \quad [54]$$

$$\hat{E}(X;\lambda) = \frac{\tilde{E}(X;\lambda)}{\|\tilde{E}(X;\lambda)\|_1} \quad [55]$$

## 5.2 HYPOTHÈSES SUR LESQUELLES EST FONDÉE LA MÉTHODE

Cette méthode repose principalement sur les six hypothèses suivantes:

1. la caméra est idéale[8] et sa sensibilité spectrale $f_c = \{ f_{ci}(\lambda) \in C[\lambda_1, \lambda_2] \}_{i \leq N}$ est connue;

2. les surfaces des objets de la scène sont mates;

3. la densité spectrale de réflectance des surfaces de la scène ($r(X;\lambda)$) est représentée approximativement par un modèle linéaire: $R = \{ R_i(\lambda) \in C[\lambda_1, \lambda_2] \}_{i \leq D(R)}$ où $D(R) \leq N-1$;

4. la densité spectrale d'éclairement sur la scène ($E(X;\lambda)$) est représentée approximativement par un modèle linéaire: $E = \{ E_i(\lambda) \in C[\lambda_1, \lambda_2] \}_{i \leq D(E)}$ où $D(E) \leq N$;

5. il est possible de segmenter l'image en sous-régions (Fig. 19) comprenant au moin D(R) matériels différents (c'est-à-dire D(R) surfaces ayant des DSRN différentes);

6. la densité spectrale d'éclairement normalisée est constante sur chacune de ces sous-régions.

La première hypothèse ne pose pas de problèmes et n'impose aucune limitation sur la méthode

---

[7] D'autres façons de normaliser auraient pu être choisies et utilisées avec la méthode de Wandell et Maloney.

[8] Veuillez référer à la section 4.4 pour la description du modèle de cette caméra idéale.

car on peut toujours déterminer la sensibilité spectrale de la caméra.

La deuxième hypothèse suppose que le signal d'entrée de la caméra $L_c(X;\lambda)$ est de la forme suivante:

$$L_c(X;\lambda) = r(X;\lambda)\ E(X;\lambda) \qquad [22]$$

où

| | |
|---|---|
| $r(X;\lambda)$ | Densité spectrale de réflectance du point X de la scène |
| $E(X;\lambda)$ | Densité spectrale d'éclairement du point X de la scène. |

Or, comme on l'a vu aux sections 4.2 et 4.6, cette hypothèse n'est valide que pour les points mats de la scène. Il faut donc s'attendre à ce que cette méthode ne soit pas très précise aux points de la scène où la composante miroir de l'image est importante.

L'irréversibilité du processus de mesure par une caméra, discuté à la section 4.5, ne rend possible la détermination d'une solution à l'équation [52] que si les densités spectrales de réflectance et d'éclairement sont respectivement confinées aux sous-espaces $[R]$ et $[E]$ de $C[\lambda_1, \lambda_2]$. Cette idée d'utiliser des modèles linéaires de réflectance et d'éclairement a été utilisée pour la première fois par [Sällström 73] et reprise par la suite par [Brill 78], [Buchsbaum 80], [Maloney 85] et [Gershon 89].

Il n'est pas possible, en général, avec ces modèles linéaires de dimensions finies, de représenter parfaitement les densités spectrales d'éclairement $E(X;\lambda)$ et de réflectance $r(X;\lambda)$. On ne pourra obtenir avec ces modèles que des estimés de $E(X;\lambda)$ et de $r(X;\lambda)$ que nous désignerons respectivement par $\tilde{E}(X;\lambda)$ et par $\tilde{r}(X;\lambda)$. Les coordonnées de $\tilde{E}(X;\lambda)$ et de $\tilde{r}(X;\lambda)$ dans les bases $E$ et $R$ seront désignées respectivement par $\varepsilon(X)$ et $\rho(X)$. Ainsi on aura:

$$\tilde{E}(X;\lambda) = E^T \varepsilon(X) = \sum_{i=1}^{D(E)} \varepsilon_i(X)\ E_i(\lambda) \quad \text{où} \quad \varepsilon(X) = \{\ \varepsilon_i(X) \in \Re\}_{i \le D(E)} \in \Re^{D(E)} \qquad [56]$$

$$\tilde{r}(X;\lambda) = R^T \rho(X) = \sum_{i=1}^{D(R)} \rho_i(X)\, R_j(\lambda) \quad \text{où} \quad \rho(X) = \{\, \rho_i(X) \in \Re \}_{i \le D(R)} \in \Re^{D(R)} \qquad [57]$$

Nous allons supposer que les modèles linéaires de réflectance et d'éclairement sont orthogonaux et qu'ils ont été normalisés de façon que $||R_j(\lambda)||_1 = 1$ pour $1 \le j \le D(R)$ et que $||E_i(\lambda)||_1 = 1$ pour $1 \le i \le D(E)$; cette hypothèse n'entraîne aucune perte de généralité.

En termes des quantitées normalisées, les équations [56] et [57] deviennent:

$$\hat{E}(X;\lambda) = E^T \hat{\varepsilon}(X) \qquad [58]$$

$$\hat{r}(X;\lambda) = R^T \hat{\rho}(X) \qquad [59]$$

où $\hat{\rho}(X)$ et $\hat{\varepsilon}(X)$ sont définis de la façon suivante:

$$\hat{\rho}(X) = \frac{\rho(X)}{||\tilde{r}(X;\lambda)||_1} = \frac{\rho(X)}{||\rho(X)||_1} \qquad [60]$$

$$\hat{\varepsilon}(X) = \frac{\varepsilon(X)}{||\tilde{E}(X;\lambda)||_1} = \frac{\varepsilon(X)}{||\varepsilon(X)||_1} \qquad [61]$$

Ces deux définitions on pour conséquences que:

$$||\hat{\rho}(X)||_1 = 1 \qquad [62]$$

$$||\hat{\varepsilon}(X)||_1 = 1 \qquad [63]$$

Pour cette méthode particulière, il est nécessaire que le nombre N de bandes spectrales de la caméra soit supérieur à la dimension D(R) du modèle linéaire de réflectance $R$ et supérieur ou égal à la dimension D(E) du modèle linéaire d'éclairement $E$. Autrement dit, on doit avoir:

$$D(E) \le N$$

$$D(R) \le N - 1$$

Lorsque la densité spectrale d'éclairement $E(X;\lambda)$ varie considérablement sur la scène mais qu'elle varie plus lentement que $r(X;\lambda)$, alors on pourra séparer l'image (Fig.19) en sous-régions $P_i$ assez petites pour que l'approximation de considérer $E(X;\lambda)$ comme constant sur $P_i$ soit valable, mais aussi assez grandes pour que chacune contienne au moins D(R) matériels différents. Il n'est pas nécessaire que les sous-régions $P_i$ soient disjointes, elles peuvent se superposer les unes les autres.

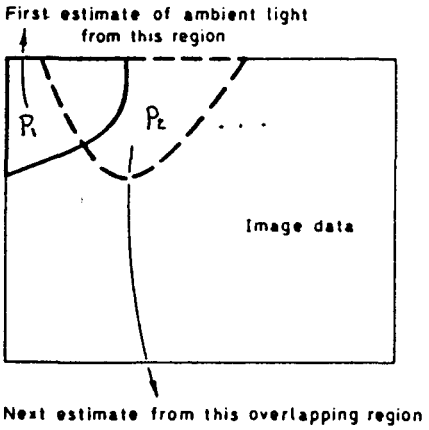


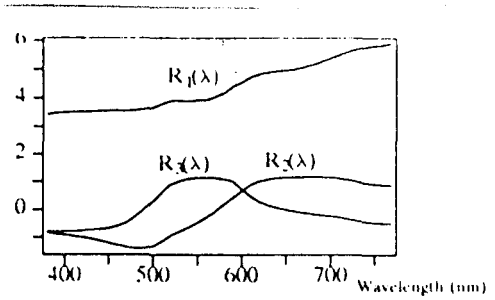

Figure 19. Séparation de l'image en sous-régions $P_i$

## 5.3 CHOIX DES MODÈLES LINÉAIRES DE RÉFLECTANCE ET D'ÉCLAIREMENT

Sur la base des densités spectrales de réflectance de 150 échantillons de couleur du livre des couleurs de Munsell, et en utilisant une méthode statistique connue sous le nom de "Principal Component Analysis", [Cohen 64] a calculé les quatre premiers vecteurs caractéristiques de cet échantillonnage et a montré que les trois premiers vecteurs constituaient un modèle linéaire de réflectance tenant compte de 99,2% de la "variance" cumulative de l'échantillonnage. Par ailleurs [Maloney 86] a montré que ces trois mêmes vecteurs (Fig.20) permettaient de reconstruire les densités spectrales de réflectance d'un échantillonnage de 370 substances naturelles [Krinov 47] avec une erreur de moins de 9,74%.

Le fait qu'un modèle linéaire de réflectance de dimension trois permette un bonne approximation sur un large échantillonnage permet de penser que, pour le cas qui nous intéresse, un modèle de réflectance de dimension deux soit suffisant. En effet, pour notre application, une scène typique ne devrait comprendre qu'un faible échantillonnage de matériels différents: tomates mûres (rouges), tomates vertes, feuilles vertes, tiges vertes. La détermination de ce modèle de réflectance de dimension deux n'a pas été faite dans le cadre de cette étude.

Judd, MacAdam et Wyszecky ont conduit une étude [Judd 64] similaire à celle menée par Cohen, mais sur les densités spectrales d'éclairement typiques pendant le jour à l'extérieur, et ce, à partir d'un échantillonnage de 622 densités spectrales d'éclairement. Les trois premiers vecteurs caractéristiques (Fig. 21) qu'ils ont calculés permettent d'estimer assez bien les conditions d'éclairage qui prévalent à l'extérieur le jour, et constituent ainsi un modèle linéaire d'éclairement valable de dimension trois qui pourrait être utilisé pour la reconnaissance de la couleur des tomates.




Figures 20. Les trois premiers vecteurs caractéristiques déterminés par Cohen à partir des densités spectrales de réflectance de 150 échantillons du livre des couleurs de Munsell [Gershon 89]

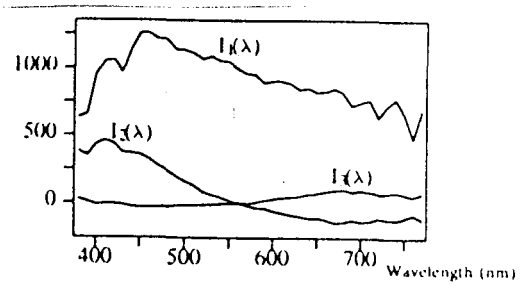



Figure 21.Les trois premiers vecteurs caractéristiques des densités spectrales d'éclairement typiques le jour déterminés par Judd , MacAdam et Wyszecki [Gershon 89]

Ainsi pour notre application il est raisonnonable de supposer qu'un modèle de réflection de dimension deux et qu'un modèle d'éclairement de dimension trois soient suffisamment précis, cela implique l'on doit choisir la caméra de sorte que:

$$D(R) = 2 \leq N\text{-}1$$

$$D(E) = 3 \leq N$$

Ainsi pour notre application et cette méthode de reconnaissancee de la couleur, une caméra de type RGB (N=3) pourrait être utilisée. Ce type de caméra est très répandu et leurs prix est relativement bas[9]. Une caméra ayant plus de trois types de senseurs permettrait d'utiliser des modèles de réflectance et d'éclairement de dimensions supérieures et par conséquent plus précis.

---

9 Veuillez référer à l'appendice II pour plus de détails sur le choix de la caméra.

Ces caméras ne sont pas présentement disponibles. Par ailleurs un modèle de réflection de dimension supérieure à deux n'entraînerait pas un gain en précision significatif tout en posant un problème supplémentaire pour la segmentation de l'image comme on le verra à la section 6.2.

## 5.4 ESTIMATION DE L'ILLUMINATION

Soit une partie $P$ de la scène pour laquelle la densité spectrale d'éclairement normalisée est constante, désignons celle-ci par $\hat{E}_P(\lambda)$. D'après l'équation [58] on a:

$$E_P(\lambda) = E^T \hat{\varepsilon}_P$$

Connaissant les mesures $I_c(X)$ de la caméra pour les points X de la scène appartenant à $P$, on va déterminer $\hat{\varepsilon}_P$.

Les mesures de la caméra sont reliées à la densité spectrale de luminance de la scène selon l'équation [38]:

$$I_c(X) = f_c\langle L_c(X,\lambda)\rangle \qquad [38]$$

D'après l'équation [22], [38], [55] et les hypothèses deux, trois et quatre, on aura:

$$I_c(X) = f_c\langle \tilde{r}(X;\lambda)\, \|\tilde{E}_P(\lambda)\|_1\, \hat{E}_P(\lambda)\rangle \qquad [64]$$

$$I_c(X) = f_c\langle \left[\sum_{j=1}^{D(R)} \rho_j(X)\, R_j(\lambda)\right] \|\tilde{E}_P(\lambda)\|_1\, \hat{E}_P(\lambda)\rangle \qquad [65]$$

$$I_c(X) = \|\tilde{E}_P(\lambda)\|_1 \sum_{j=1}^{D(R)} \rho_j(X)\; f_c\langle R_j(\lambda)\, \hat{E}_P(\lambda)\rangle \qquad [66]$$

$$I_c(X) = \|\tilde{E}_P(\lambda)\|_1\, \Lambda_P\, \rho(X) \qquad [67]$$

où

$$\Lambda_p = \{ f_c\langle R_j(\lambda)\ \hat{E}_p(\lambda)\rangle \in \Re^N \}^T_{j\le D(R)} \in \Re^{NxD(R)} \qquad [68]$$

La matrice $\Lambda_p$ est appelée la matrice d'éclairement et la valeur de cette matrice en un point ne dépend que de la densité spectrale d'éclairement normalisée en ce point. Cette matrice sera donc constante sur $P$. L'équation [67] implique que les variations de $I_c(X)$ sur $P$ dépendent seulement des variations de la densité spectrale de réflectance ($\rho(X)$) et d'un scalaire ($\|\tilde{E}_p(\lambda)\|_1$) représentant l'intensité de l'illumination en chaque point. Comme la matrice d'éclairement est constante sur $P$, l'équation [67] implique que, quelque soit la valeur que prendra $\rho(X)$, tous les vecteurs $I_c(X)$ appartiennent au sous-espace $[\Lambda_p]$ de dimension D(R) de $\Re^N$. Comme d'après l'hypothèse 3, $D(R) \le N-1$, le sous-espace $[\Lambda_p]$ est un hyperplan de $\Re^N$ et il existe nécessairement un vecteur $\pi_p = \{ \pi_{pi} \}_{i\le N} \in \Re^N$ tel que $\pi_p$ est perpendiculaire au sous-espace $[\Lambda_p]$ et tel que $\pi_{p1}$[10] $= 1$. Ce vecteur $\pi_p$ est unique dans le cas où $D(R) = N-1$.

Afin de déterminer $\hat{\varepsilon}_p$, il faut dans un premier temps déterminer $\{\pi_{pi}\}_{2\le i\le N}$ et pour cela il nous faut au moins D(R) équations indépendantes. Pour ce faire on choisit m points $X_i \in P$ ($m \ge D(R)$) et on forme la matrice suivante:

$$\Delta = \{ I_c^T(X_i) \in \Re^{1xN} \}_{i\le m} \in \Re^{mxN}$$

Le choix de ces points doit faire en sorte que $[\Delta] = [\Lambda_p]$, ou que le sous-espace $[\Delta]$ soit de dimension D(R). Comme $\rho(X)$ est un caractéristique propre à un matériel donné, à partir de l'équation [67] on peut voir le nombre de valeurs différentes que prendra $\rho(X)$ sur $P$ va être égal au nombre de matériels différents sur $P$. Ainsi la dimension du sous-espace $[\Delta]$ sera égale au nombre de matériels différents sur $P$. Il faut donc qu'il y ait D(R) matériel sur $P$ et que la matrice $\Delta$ soit formée à partir de points dont au moins un appartienne à chaque matériel. C'est la raison pour laquelle l'hypothèse six de la section 5.2 est nécessaire.

---

[10] La valeur assignée à $\pi_{p1}$ est arbitraire, seulement la direction du vecteur $\pi_p$ est importante pour la suite.

Une fois la matrice $\Delta$ formée, on pourra déterminer le vecteur $\pi_p$ en solutionnant le système d'équations suivant:

$$\pi_1 = 1$$
$$\Delta\ \pi_p = 0_m \qquad [69]$$

Comme en général, il y a du bruit sur les mesures formant $\Delta$ , ce système d'équations n'a pas de solution exacte. Cependant, on peut déterminer une solution approximative pour $\pi_p$ et plusieurs méthodes peuvent être employées à cet effet.

Une fois le vecteur $\pi_p$ déterminé, on procède à la détermination de $\hat{\varepsilon}_p$ en solutionnant un système de n équations (avec $n \geq D(E)$ ) entre $\pi_p$ et $\hat{\varepsilon}_p$. D'après les équations [65],[58] et [61] :

$$I_c(X) = \rho(X)^T \left\{ f_c\langle R_i(\lambda)E_j(\lambda) \rangle \right\}_{\substack{i\leq D(R) \\ j\leq D(E)}} \|\tilde{E}_p(\lambda)\|_1\ \hat{\varepsilon}_p \qquad [70]$$

$$I_c(X) = \|\tilde{E}_p(\lambda)\|_1\ \Lambda_\rho\ \hat{\varepsilon}_p \qquad [71]$$

où

$$\Lambda_\rho = \rho(X)^T \left\{ f_c\langle R_i(\lambda)E_j(\lambda) \rangle \right\}_{\substack{i\leq D(R) \\ j\leq D(E)}} \in \Re^{D(R)xD(E)} \qquad [72]$$

Il est possible de choisir au préalable une famille de n vecteurs $\{\rho^k\}_{k\leq n}$ et et de calculer pour chaque vecteur de cette famille la matrice $\Lambda_{\rho\kappa}$ en remplaçant dans l'équation [72]:

$$\Lambda_{\rho\kappa} = \rho^{kT} \left\{ f_c\langle R_j(\lambda)E_j(\lambda) \rangle \right\}_{\substack{i\leq D(R) \\ j\leq D(E)}} \qquad [73]$$

Comme $\pi_p$ est par définition perpendiculaire à toutes les mesures $I_c(X)$, il se doit d'être perpendiculaire à $\Lambda_{\rho\kappa}\ \hat{\varepsilon}_p$ (équation [71]], de sorte que l'on a::

$$\pi_p^T\ \Lambda_{\rho\kappa}\ \hat{\varepsilon}_p = 0 \quad \text{pour} \quad 1 \leq k \leq n$$

d'où le système de n équations suivant:

$$\begin{aligned} \|\hat{\varepsilon}(X)\|_1 &= 1 \\ \Pi^T \hat{\varepsilon}_p &= \mathbf{0}_n \end{aligned} \qquad [74]$$

où

$$\Pi = \{\pi_p{}^T \ \Lambda_{p\kappa}\}_{k\leq n} \qquad [75]$$

Plusieurs méthodes permettent de calculer une solution approximative pour $\varepsilon_p$ à partir de ce système d'équations.

## 5.5 DÉTERMINATION DE LA DSRN EN CHAQUE POINT DE L'IMAGE

L'estimé de la DSRN en un point X donné est complètement déterminé par $\hat{\rho}(X)$ (équation [59]). Voyons maintenant comment on peut déterminer $\hat{\rho}(X)$. Supposons que l'image a été segmentée en sous-régions $P_l$ et que $\hat{\varepsilon}_{p_l}$ a été calculé pour chacune de ces sous-régions.

On doit d'abord calculer la matrice d'éclairement pour chacune de ces sous-régions en utilisant l'équation [68]; nous désignerons la valeur que prendra cette matrice en chaque point par $\Lambda_{p(X)}$. On doit ensuite calculer la matrice pseudo-inverse de la matrice d'éclairement pour chacune de ces sous-régions; nous désignerons la valeur que prendra cette matrice en chaque point par $\Lambda_{p(X)}{}^{-1*}$.

L'équation [67] peut être écrite en terme de $\hat{\rho}(X)$ de la façon suivante:

$$\mathbf{I_c}(X) = \|\tilde{E}_p(\lambda)\|_1 \ \|\tilde{r}(X,\lambda)\|_1 \ \Lambda_{p(X)}{}^{-1*} \ \hat{\rho}(X) \qquad [76]$$

On connaît $\mathbf{I_c}(X)$ et $\Lambda_{p(X)}{}^{-1}$ mais on ne connaît pas le facteur $\|\tilde{E}_p(\lambda)\|_1 \ \|\tilde{r}(X,\lambda)\|_1$. Cependant, l'on sait (équation [62]) que la norme d'ordre 1 de $\hat{\rho}(X)$ est égale à 1 de sorte que:

$$\hat{\rho}(X) = \frac{\Lambda_{p(X)}{}^{-1*} \ \mathbf{I_c}(X)}{\|\Lambda_{p(X)}{}^{-1*} \ \mathbf{I_c}(X)\|_1} \qquad [77]$$

# Chapitre VI

# CLASSIFICATION DES TOMATES SELON LEUR COULEUR

Au chapitre 3 (section 3.2) on a présenté deux algorithmes généraux de localisation des tomates mûres dans une image. Le premier[11] de ces algorithmes propose de déterminer la position des centroïdes et les rayons de toutes les tomates présentent dans l'image dans un premier temps et de déterminer celles qui sont mûres à partir de leur couleur dans un deuxième temps. À la section 3.1 on a vu que la transformation de Hough circulaire peut être utilisée pour réaliser la première étape de l'algorithme. Dans ce chapitre, nous verrons comment la méthode de reconnaissance de la couleur de Wandell et Maloney discutée au chapitre V peut être utilisée pour calculer un indice de couleur pour chaque tomate localisée dans l'image. Finalement nous verrons comment on peut classifier chacune des tomates à partir de son indice de couleur.

## 6.1 L'INDICE DE COULEUR

À la section 5.3 on a vu que, pour cette application particulière, nous devons utilser:

- une caméra de type RGB (N=3),
- un modèle linéaire de réflectance orthonormal approprié de dimension 2 (D(R) = 2),
- et un modèle linéaire d'éclairement de Judd et Wyszecky de dimension 3 (D(E) = 3).

Cette méthode déterminera, pour certains pixels X, les coordonnées ($\hat{\rho}(X)$) de la DSRN relativement au modèle de réflectance utilisé . Dans le cas présent où le modèle de réflectance est

---

[11] Le deuxième algorithme serait très difficilement utilisable avec la méthode de reconnaissance de la couleur de Wandell et Maloney et, pour cette raison, ne sera pas retenu.

de dimension 2, le vecteur $\rho(X)$ aura 2 composantes:

$$\hat{\rho}(X) = ( \hat{\rho}_1(X), \hat{\rho}_2(X) )^T$$

D'après l'équation [62] ce vecteur a une norme d'ordre 1 égale à un. On peut donc déterminer $\hat{\rho}_2(X)$ à partir de $\hat{\rho}_1(X)$. Comme $\hat{\rho}_2(X)$ n'apporte aucune information sur la couleur qui ne soit déjà présente dans $\hat{\rho}_1(X)$, on ne va utiliser que $\hat{\rho}_1(X)$ et l'appeler l'indice de couleur du pixel X.

## 6.2 CALCUL DE L'INDICE DE COULEUR D'UNE TOMATE

La figure 22 présente une image pour laquelle la position des centroïdes des tomates ainsi que leur rayon ont été déterminés par la méthode basée sur la transformation de Hough circulaire présentée à la section 3.1. Il y a peut-être une feuille ou une tige qui cache certaines parties de ces tomates, mais cette méthode ne permet pas de le vérifier. Dans le cas des tomates groupées qui se recouvrent partiellement, on ne sait pas lesquelles cachent partiellement l'autre.

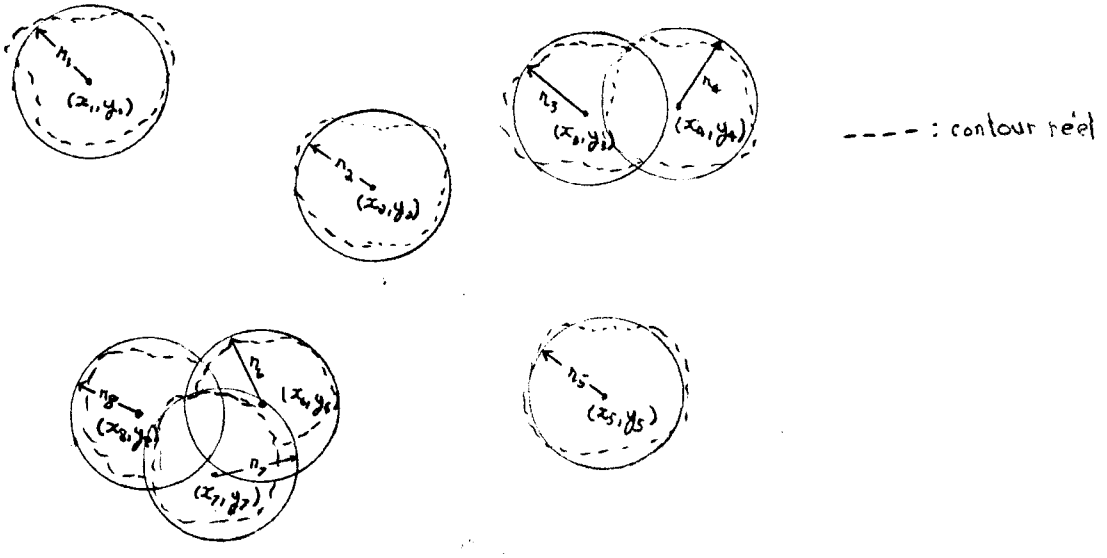



Figure 22. Positions des centoïdes et des rayons des tomates localisées dans une image.

Pour déterminer la couleur d'une tomate on n'a pas besoin de connaître l'indice de couleur

de chacun de ses pixels mais seulement celui de quelques uns car la couleur des tomates est très uniforme, et ce, à tous les stades de la maturité [Heron 71].

Pour déterminer la couleur d'une tomate, on propose de calculer l'indice de couleur d'un petit nombre de pixels sur chaque tomate et de classifier chaque tomate sur la base de l'indice de couleur moyen de ces pixels.

Pour calculer l'indice de couleur d'un pixel donné avec la méthode de Wandell et Maloney on doit d'abord estimer la DSEN sur la sous-région à laquelle appartient ce pixel, et ensuite utiliser l'équation[77] pour calculer cet indice de couleur. Pour pouvoir estimer la DSEN on doit choisir cette région de façon à ce qu'elle contienne au moins deux matériels ayant des DSRN différents (hypothèse 5 de la section 5.2) et de façon à ce que la DSEN puisse être considérée comme constante (hypothèse 6 de la section 5.2). Nous allons maintenant décrire une stratégie de sélection (pour chaque tomate) de la région la plus appropriée pour faire l'estimation de la DSEN.

Cette région devra contenir une partie de la tomate et une partie de l'extérieur de la tomate afin d'avoir au moins deux matériels ayant des DSRN différentes. Dans le cas d'une tomate verte sur un fond uniforme de la même couleur, cette condition ne sera pas remplie. On verra plus loins comment on peut traiter ce cas particulier. Cette région ne devra comprendre qu'un seul côté de la tomate pour minimiser les variations de la DSEN. La partie supérieure de la tomate ne devra pas faire partie de cette région car la couleur de l'endroit où la tomate est reliée à la plante est différente. De plus la composante miroir de la luminance (sections 3.2 et 4.6) sur cette partie supérieure devrait être beaucoup plus importante qu'ailleurs, et dans le cas où elle ne serait pas négligeable il serait impossible d'estimer la DSEN avec la méthode de la section 5.4 (hypothèse 2 de la section 5.2). Dans le cas d'une tomate faisant partie d'un groupe, cette région ne devrait pas être située sur un côté où il y a occlusion partielle avec une autre tomate du groupe; on évite ainsi le problème de savoir laquelle cache l'autre en plus de maximiser les chances d'avoir au moins deux matériels différents. En effet si les deux tomates étaient de la même couleur on n'aurait qu'un seul matériel dans la région.

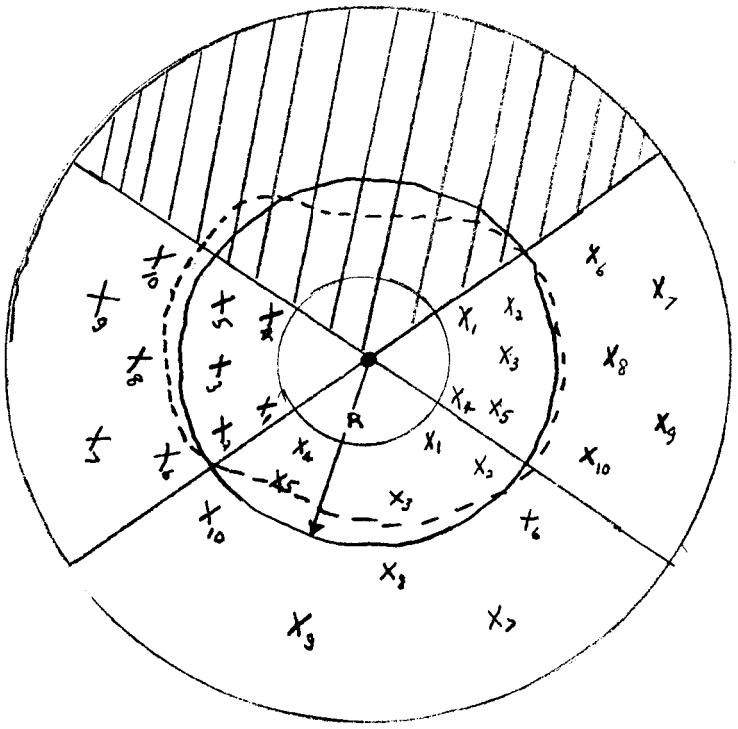


Figure 23. Trois régions où l'on peut faire l'estimation de la DSEN dans le cas d'une tomate isolée

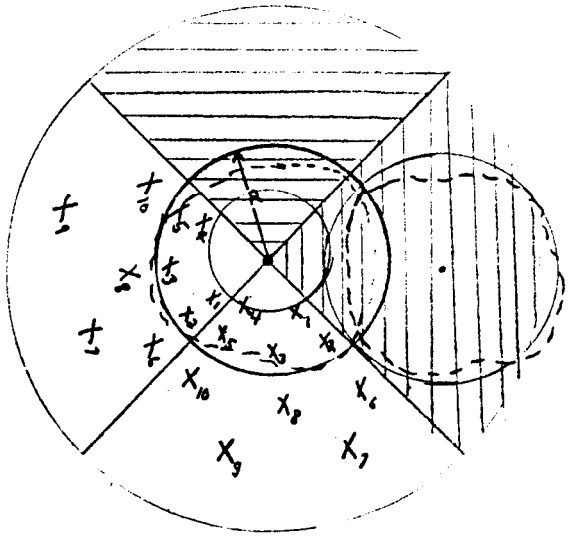


Figure 24. Deux régions où l'on peut faire l'estimation de la DSEN dans le cas d'une tomate faisant partie d'un groupe

À la figure 23, on présente une tomate isolée où sont indiquées trois régions satisfaisant aux conditions mentionnées plus haut. À la figure 24, on présente une tomate faisant partie d'un groupe où sont indiquées deux régions satisfaisant à toutes les conditions mentionnées. On peut, soit choisir de façon arbitraire une de ces régions, soit choisir la région qui satisfait le plus au critère (conséquence de l'équation [67] lorsque D(R) = 2) suivant: "les mesures de la caméra doivent être confinéee dans un plan". Afin de quantifier la validité de cette hypothèse pour chacune des région on propose la méthode suivante.

Choisir 10 mesures de la caméra sur chaque région, les cinq premières étant sur la tomate et les cinq dernières étant à l'extérieur $\{ \mathbf{I_c}^T(X_i) \in \Re^{1x3} \}_{i\leq 10}$, la position des pixels est indiquée aux figures 24 et 25. De cette famille de mesures on ne conservera que les mesures qui sont à l'intérieur de la plage linéaire de la caméra : $[\mathbf{I_{cMIN}}, \mathbf{I_{cMAX}}]$. On élimine donc les pixels pour lesquels $\mathbf{I_{cMIN}} \leq \mathbf{I_c}(X_i) \leq \mathbf{I_{cMAX}}$. Après l'élimination de ces pixels on obtient la matrice suivante:

$$\Delta = \{ \mathbf{I_c}^T(X_i) \}_{i\leq U\leq 10} \in \Re^{Ux3}$$

On détermine ensuite les valeurs propres de la matrice $\Delta^T\Delta \in \Re^{3x3}$; cette matrice doit avoir 3 valeurs propres distinctes: $\{ \sigma_i \}_{i\leq 3}$ (les valeurs propres sont indicées par ordre décroissant).

Les mesures formant $\Delta$ seront d'autant plus comprises dans un plan que le rapport suivant sera proche de 1[Golub 83]:

$$\frac{\sigma_1+\sigma_2}{\sum_{i=1}^{3}\sigma_i}$$

On choisira la région pour laquelle ce rapport est le plus grand.

Pour le cas rare où la couleur de la tomate est presque la même que celle de son environnement (cette situation ne devrait se produire que dans le cas d'une tomate verte), on ne pourra pas calculer l'indice de couleur de la tomate dans ce cas car la détermination de l'illumination ambiante par la méthode de la section 5.4 devient alors impossible. Mais ce cas peut être détecté. En effet, les mesures de la caméra sur la région devraient alors être comprises sur une droite [équation 67]. Le degré pour lequel cette approximation s'avère vraie est quantifié par le rapport suivant:

$$\frac{\sigma_1}{\sum_{i=1}^{3}\sigma_i}$$

Plus ce rapport sera proche de 1 et plus les mesures seront sur une droite.

Dans le cas où l'on juge que ce rapport est suffisamment proche de 1, on peut conclure que la tomate est verte et n'est pas mûre. Dans le cas où ce rapport est suffisamment éloigné de un, on procède à l'estimation de la DSEN à partir de la matrice $\Delta$ en suivant la méthode de la section 5.4.

À la section 5.4 on détermine dans un premier temps le vecteur $\pi_p$ en solutionnant le système d'équation [69]. Dans le cas présent on peut utiliser la méthode alternative suivante:

$$\pi_p = \mathbf{u}(\sigma_1) \times \mathbf{u}(\sigma_2) \qquad [78]$$

où

$\mathbf{u}(\sigma_1)$ vecteur propre de la matrice $\Delta^T\Delta$ associé à la valeur propre $\sigma_1$,

$\mathbf{u}(\sigma_2)$ vecteur propre de la matrice $\Delta^T\Delta$ associé à la valeur propre $\sigma_2$,

$\times$ produit vectoriel dans $\Re^3$.

Soit $\{ \mathbf{I}_c(X_i) \}_{i\leq 5}$ les mesures de $\Delta$ appartenant à la tomate, on peut maintenant calculer à partir de l'équation [77] les indices de couleur pour chacun de ces pixels: $\{ \hat{\rho}_1(X_i) \}_{i\leq 5}$. On élimine ensuite l'indice de couleur minimum et l'indice de couleur maximum[12] et l'on fait la moyenne de ceux restant; nous appellerons cette moyenne l'indice de couleur de la tomate et on le désignera par $\hat{\rho}_{1T}$.

---

[12] L'élimination de ces valeurs extrêmes réduit le bruit.

## 6.3 CLASSIFICATION DES TOMATES À PARTIR DE LEUR INDICE DE COULEUR

Il nous faut d'abord disposer d'un large échantillonnage de tomates mûres pour lesquelles on connaît les indices de couleur $\{ \rho_{1i} \}_{i \leq V}$. On peut alors calculer l'indice de couleur moyen ($\rho_{1M}$) de cet échantillonnage:

$$\hat{\rho}_{1M} = \frac{1}{V} \sum_{i=1}^{V} \hat{\rho}_{1i} \qquad [79]$$

On détermine ensuite le rayon maximum (T) de cet échantillonnage que l'on définit comme:

$$T = \max_{i \leq M} \{ | \hat{\rho}_{1i} - \hat{\rho}_{1M} | \}_i \qquad [80]$$

Connaissant l'indice de couleur moyen d'un échantillonnage représentatif de tomates mûres ainsi que le rayon maximum de cet échantillonnage, on peut définir une fonction binaire de classification des couleurs qui prend la valeur "1" lorsque que l'indice de couleur est celui d'une tomate mûre et prend la valeur "0" dans le cas contraire:

$$I_b(\hat{\rho}_{1T}) = \begin{cases} 1 & \text{si } | \rho_{1M} - \rho_{1T} | \leq T \\ 0 & \text{si } | \rho_{1M} - \rho_{1T} | > T \end{cases} \qquad [81]$$

# CONCLUSION

Malgré les nombreuses recherches qui ont été faites, aucun prototype de robot-cueilleur de fruits n'est assez avancé présentement sur le plan technique pour être commercialisé. L'étude faite au chapitre I sur la faisabilité économique d'un robot-cueilleur de tomates révèle que les économies qu'engendreraient un tel robot sont potentiellement assez grandes pour laisser présager avec optimisme que sa fabrication serait économiquement rentable. On mentionne, entre autres, que le pourcentage de cueillette doit être supérieur à 70%, ce qui demande du système de vision un taux d'efficacité minimum de près de 90%. Selon cette étude, les économies engendrées par un tel robot sont directement proportionnelles au temps pendant lequel il peut fonctionner en moyenne dans une journée. Le système de vision d'un tel robot se doit donc d'être efficace dans les conditions d'illumination extrêmes qui existent dans les serres le jour.

L'étude qualitative réalisée à la section 1.4 sur les variations du spectre de réflectance des tomates en fonction de leur maturité confirme la possiblilité de discriminer les tomates mûres sur la base de leur spectre de réflectance normalisé et ce, tout particulièrement sur les intervalles suivants: [400 nm, 570 nm ] et [650 nm, 675 nm].

Il n'est pas interdit de réfléchir aux autres fonctions qu'un robot-cueilleur de tomates pourrait effectuer pour le compte du producteur, mais ceci n'entrait pas dans le cadre de cette thèse.

Localiser une tomate (ou un fruit) dans l'espace, pour permettre à un robot de la cueillir demande, que l'on connaisse la direction de la tomate relativement à l'axe optique de la caméra et demande que l'on connaisse également sa distance dans cette direction. La direction est déterminée par un algorithme de localisation du centroïde de la tomate dans une image. La distance peut être déterminée par une technique de vision stéréoscopique mais nous croyons que l'approche adoptée

par les prototypes français et américain, qui consistent à utiliser un bras télescopique au bout duquel un senseur permet de détecteur la présence du fruit lorsque le bras a parcouru la bonne distance, est plus simple, plus rapide et en bout de ligne préférable.

Le bras de cueillette, conçu par la société Martin Marietta (section 2.2.3) pour la cueillette des oranges, nous semble approprié à la cueillette des tomates, mais à la condition qu'il soit réduit de taille. Les photosenseurs et la source de lumière placés au bout de ce bras, en plus de permettre une localisation précise du fruit en fin de course, permettraient de faire une double vérification de la couleur de la tomate avant de la cueillir. A cette fin, les techniques de tri des tomates de la section 2.1 pourrait être mises à profit.

Le prototype français de robot-cueilleur de pommes (section 2.2.2) utilise, pour la préhension des pommes, une pince munie d'un clapet mobile servant à couper le pédoncule du fruit; cette approche est intéressante et pourrait être utilisée par un robot-cueilleur de tomates.

L'utilisation d'un écran faisant obstacle aux rayons en provenance directe du soleil dans l'espace de travail du robot et d'un système d'éclairage favorisant un éclairage diffus et plus uniforme, faciliteraient la tâche du système de vision. Si le bras de cueillette était monté sur un pont roulant, il serait alors très facile de déplacer cet écran. Il y aurait probablement moyen d'utiliser cet écran même si le bras de cueillette était monté sur un chariot se déplaçant sur rails ou directement sur le sol.

La caméra devra être une caméra couleur de type RGB ou une caméra ayant trois senseurs. Il n'est pas nécessaire pour cette application que la caméra soit une caméra vidéo, c'est-à-dire une caméra qui envoie 60 demi-images par secondes. Une caméra digitalisant une image à la fois sur commande serait suffisante. Une caméra vidéo couleur dont les signaux de sortie suivent le standard NTSC ne serait pas appropriée, car les signaux de chrominance pour ce type de caméra sont trop filtrés, engendrant une perte importante d'information couleur. Finalement, l'image digitalisée devra avoir des pixels de forme carrée, excluant encore une fois l'utilisation d'une

caméra vidéo suivant le standard NTSC. L'annexe II discute en détail cette question du choix d'une caméra.

Les algorithmes de localisation des tomates dans une image qui sont utilisés par les trois prototypes de robot-cueilleur (sections 2.2.1 à 2.2.3) sont très similaires. Ils diffèrent légèrement par leur méthode de reconnaissance de la couleur et par leur méthode de segmentation de l'image.

Les méthodes de reconnaissance de la couleur présentées dans les sections 2.2.1 à 2.2.4 sont très rudimentaires. Deux de ces méthodes (sections 2.2.2 et 2.2.4) se contentent de l'information d'une caméra monochrome munie d'un filtre approprié. Le prototype de robot-cueilleur japonais (section 2.2.1) utilise une caméra couleur, mais dont l'information couleur est dégradée par le standard NTSC et utilise un circuit électronique qui génère une image rouge (selon une métrique simpliste), cette dernière étant par la suite binarisée. Quand à la stratégie du prototype américain qui d'ailleurs est semblable à celle utilisée sur les machines de tri des tomates, elle souffre elle aussi du même problème de faible robustesse face aux conditions d'illumination. Nous avons nous-même utilisé une méthode de reconnaissance de la couleur similaire à celles utilisées par les prototypes américains et japonais (voir annexe III). Ne disposant que d'une carte digitalisante qui ignore les signaux de chrominance de la caméra, nous avons pensé prendre deux images de la scène: la première sans filtre et la deuxième avec un filtre rouge placé devant la caméra. Cette méthode s'est avérée efficace dans la mesure seulement où l'illumination était suffisamment uniforme et diffuse. Nous n'avons pas cru bon de mentionner cette méthode dans le corps de ce mémoire étant donné qu'elle ne donne pas lieu à des résultants concluants.

À la section 3.1 on présente une méthode de segmentation d'une image qui permet de localiser les centroïdes des tomates ainsi que leur rayon respectif. Cette méthode consiste à former une image binaire de contours et à transformer cette image par la transformation de Hough circulaire, pour ensuite localiser les maximums dans le tableau des accumulateurs. Cette approche est robuste vis-à-vis l'occlusion partielle d'une partie de la tomate, en plus d'offrir l'avantage de n'être pas limitée à la localisation des tomates rouges seulement. Cela est important du fait que les tomates

poussent en bouquets et que l'on ne peut ignorer les tomates vertes qui sont à proximité des tomates que l'on veut cueillir. Nous avons expérimenté cette approche (voir programmes EDGE_1 À EDGE_4, HISTO, THRESHOD, HOUGH_1 à HOUGH_4 de l'annexe III). Les tests effectués sur des images binaires de contours contenant des cercles parfaits (produis par le programme TCERCLE) ont été concluants. Mais nous avons interrompu prématurément notre quête d'un algorithme de recherche des maximums lorsque nous avons pris connaissance d'une recherche similaire [Whittaker 84].

Nous avons proposé deux algorithmes de localisation des tomates dans une image (section 3.2). Le premier consiste à localiser toutes les tomates de l'image à partir de leur forme pour ensuite déterminer celles qui sont mûres en tenant compte de la couleur d'un certain nombre de ses pixels. Le deuxième algorithme procède de façon inverse; seulement les tomates mûres sont localisées dans un premier temps et ensuite toutes les tomates à proximité de ces dernières sont localisées à partir de leur forme. Ce deuxième algorithme serait difficilement utilisable avec la méthode de reconnaissance de la couleur de Wandell et Maloney. L'algorithme I est plus simple et s'intègre bien avec la méthode Wandell et Maloney, et c'est donc celui que l'on préconise.

La deuxième partie de la thèse (chapitres IV à VI) a été entièrement consacrée à faire une analyse de la question de la reconnaissance de la couleur des tomates. Nous avons cherché une méthode qui soit robuste aux contitions d'illumination variables des serres, afin de permettre à un robot-cueilleur de fonctionner indépendamment de ces conditions, même si les mesures de la caméra sont corrompues par celles-ci.

Le chapitre IV est consacré à l'analyse de la physique de la réflexion de la lumière par les objets, de la physique de la formation d'une image par une caméra ainsi que sur l'information couleur irrémédiablement perdue par une caméra.

À la section 4.2 on constate que la réflexion de la lumière par les objets opaques pouvait être parfaitement décrite par une fonction appelée "la densité bidirectionnelle spectrale de réflectance"

(DBSR). Cette fonction comporte tellement de paramètres (cinq avec la longeur d'onde) qui varient avec la géométrie, qu'elle n'est à toute fin pratique pas utilisable comme modèle de réflexion.

À la section 4.3, nous décrivons le modèle de réflexion de Horn qui modélise la surface d'un objet opaque comme un réflecteur semi-lustré, c'est-à-dire comme une surface qui a un comportement vis-à-vis la réflexion de la lumière, qui est un compromis entre celui du miroir parfait et celui du réflecteur mat. À la section 4.4 on procède à la modélisation d'une caméra où on introduit les concepts suivants: un système optique idéal, un récepteur d'images idéal, une caméra idéale.

À la section 4.5, on fait une analyse que nous croyons originale, à certains égards, de l'irréversibilité du processus de mesure d'une caméra. On définit les concepts de composantes principales et résiduelles ainsi que le concept de métamère. Cette analyse forme la base justifiant l'utilisation des modèles linéaires de réflectance et d'éclairement qui sont utilisés entre autres par la méthode de reconnaissance de la couleur de Wandell et Maloney.

À la section 4.6, on montre la difficulté d'utiliser le modèle du réflecteur semi-lustré et les conséquences d'utiliser le modèle du réflecteur mat utilisé, entre autres, par la méthode de Wandell et Maloney. On a d'abord montré à partir du concept de la décomposition d'une image en une image mate et une image miroir qu'il était absolument nécessaire d'éliminer la composante miroir dans le cas où cette dernière n'était pas négligeable. On a ensuite montré que la reconnaissance de la couleur dans une image mate était très dépendante de l'illumination de la scène. Lorsque ces conditions d'illumination sont variables sur la scène (le cas qui nous intéresse), alors il devient impossible d'utiliser les méthodes usuelles de reconnaissance de la couleur qui prennent pour hypothèse un éclairement constant. Les méthodes "computationnelles" de reconnaissance de la couleur sont alors tout à fait appropriées. A ce propos, les expériences faites par Land [Land 64] sont convaincantes. La méthode "computationnelle" qui nous sembles la plus appropriée à notre application est la méthode de Wandell et Maloney.

On a consacré le chapitre V à l'analyse de la méthode de reconnaissance de la couleur de Wandell et Maloney. Cette analyse a été faite de façon différente de celles présentées dans [Wandell 87] et dans [Maloney 86]. Nous avons par exemple représenté les spectres par des fonctions réelles et continues au lieu de les représenter de façon approximative comme des suites finies de nombres. La méthode de Wandell et Maloney suppose que l'image est mate, et que les densités spectrales d'éclairement et de réflectance peuvent être représentées par des modèles linéaires de dimensions finies. Elle suppose également que la caméra utilisée ait un nombre de senseurs supérieur à la dimensision du modèle linéaire de réflectance. Elle suppose enfin que chaque région de l'image contienne au moins autant de matériels différents que la dimension du modèle linéaire de réflectance et que la densité spectrale d'éclairement normalisée soit constante sur chaque région. Si toutes ces conditions sont remplies, alors les variations des mesures de la caméra sur ces régions seront confinées à un sous-espace de même dimension que le modèle linéaire de réflectance. On identifie ce sous-espace à partir d'un certain nombre de points sur une région, on peut ensuite calculer un estimé de la densité spectrale d'éclairement normalisée sur la région en question. Finalement une fois que la densité spectrale de réflectance normalisée est déterminée pour une région donnée, on peut calculer la DSRN pour chaque pixel de cette région en multipliant la mesure de la caméra par la matrice pseudo-inverse de la matrice d'éclairement.

À la section 5.2, nous avons jugé que, pour le cas de la reconnaissance de la couleur des tomates un modèle linéaire de réflectance de dimension 2 serait approprié, mais nous n'avons pas déterminé ce modèle. Nous avons proposé d'utiliser comme modèle linéaire d'éclairement celui proposé par [Judd 64] qui est de dimension trois. La faible dimension de ces modèles permet l'utilisation d'ne caméra de type RGB pour cette application.

Le chapitre VI décrit comment la méthode de reconnaissance de la couleur de Wandell et Maloney est utilisée pour assigner un indice de couleur à chaque tomate, et comment cet indice de couleur est utilisé pour classer chaque tomate comme étant mûre ou non-mûre.

La méthode de reconnaissance de la couleur de Wandell et Maloney requière plusieurs prémisses qui ont empêché jusqu'ici la méthode d'être utilisée dans une application pratique. La difficulté principale est la nécessité de segmenter l'image en régions sur lesquelles la densité spectrale d'éclairement normalisée soit constante et contenant autant de matériels différents que la dimension du modèle linéaire de réflectance. À la section 6.2, nous avons proposé une méthode pour résoudre ce problème dans le cas particulier de la reconnaissance de la couleur des tomates. Le fait d'utiliser un modèle de réflectance de dimension 2 seulement a grandement facilité la résolution de ce problème. On présente également une modification à la méthode de Wandell et Maloney dans la façon de calculer le vecteur perpendiculaire aux mesures de la caméra.

L'approche à la reconnaissance visuelle des tomates que nous proposons demande beaucoup de calculs, mais nous croyons que les nouveaux ordinateurs de traitement d'images à architecture parallèle sont (ou seront) assez rapides pour que ces calculs se fassent dans des temps raisonnables.

Une étude expérimentale de cette approche pour la reconnaissance visuelle des tomates devra être réalisée afin d'élucider les nombeux "si" inclus dans ce mémoire.

# BIBLIOGRAPHIE

## BIBLIOGRAPHIE

ARCELLI, C., S. LEVIALDI [1971] "Picture processing and overlapping blobs", *IEEE Transaction on Comput.*, C-20, pp. 1111-1115, september.

BADIQUÉ, E., N. Ohyama, T. Honda, J. Tsujiuchi [1988] "Color image correlation for spatial/spectral recognition and increased selectivity", *Optics communications*, Vol. 68, No. 2, 15 September 1988, pp. 91-97.

BALLARD, D.H. [1981] "Generalizing the Hogh transform to detect arbitrary shapes", *Pattern Recognition letters*, Vol. 2, 1981, pp. 111-122.

BALLARD, D.H., & C.M. BROWN [1982], *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982, 523 p.

BALLARD D.H., C.M. BROWN [1985] "Vision", *Byte*, April, 1985, pp. 245-261.

BERRY, D.T. [1987] "Colour recognition using spectral signatures", *Pattern Recognition Letters*, Vol. 6, No. 1, June 1987, pp. 69-75.

BIEDERMAN, I. [1985] "Human Image Understanding: Recent Research and Theory.", *Computer Vision, Graphics and Image Processing*, Vol. 32, No. 1, pp. 29-73.

BIEMAN, L.H. [1988] "Three-dimensional machine vision", Photonics spectra, May 1988, pp. 81-92.

BITTNER, D.R. [1968], K.H. Norris, "Optical Properties of Selected Fruits vs Maturity", *Transactions of the ASAE*, Vol. 11, No 4, 1968, pp. 534-536.

BITTNER, D.R., K.Q. Stephenson [1968] "Reflectance and transmittance properties of tomatoes versus maturity", *American Society of Agricultural Engineering (ASAE) paper* No 68-329.

BOYER, K.L., A.J. VAYDA, A.C. KAK [1986] "Robotic Manipulation Experiments Using Structural Stereopsis for 3D Vision." *IEEE Expert*, Vol. 1, No. 3, pp. 73-94.

BRAINARD, D.H., B. A. Wandell [1986] "Analysis of the retinex theory of color vision", *Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1651-1661.

BUCHSBAUM, G. [1980], "A Spatial Processor Model for Object Colour Perception", *The Franklin Institute*, Vol. 310, No. 1, July 1980, pp.1-26.

BUNBACA, F. [1987] "Design and Implementation of a Colour Vision Model for Computer Vision Applications", *Computer Vision, Graphics, and Image Processing*, Vol. 39, No. 2, 1987, pp. 226-245.

BRILL, M. H.[1978] "A device performing illuminant-invariant assessment of chromatic relations", *J. Theor. Biol.*, Vol. 71, pp. 473-478, 1978.

CHIEN, Y.P., & K.S. FU [1974] "A Decision Function Method for Boundary Detection.", *Computer Graphics and Image Processing,* Vol. 3, No. 3, pp. 125-140.

COFFIN W.L., A.M. SKELTON, H.A. JACKSON [1988], "Light levels in multispan greenhouse models", *Canadian Agricultural Engineering*, Vol. 30, No. 1, 1988, pp. 143-149.

COHEN, J.B. [1964] "Dependency of the spectral reflectance curves of the Munsell color chips", *Psychon. Sci.,* 1964, Vol. 1, pp.369-370.

COHEN, J. B [1988] "Color and Color Mixture: Scalar and Vector Fundamentals", *Color research and application*, Vol. 13, No. 1, February 1988, pp.5-23.

CORBY, N.R. Jr. [1988] "Precision High-Speed range sensor and processor", *Sensor review,* Vol. 8, No. 3, July 1988, pp.155-160.

COULON, G. [1986]. "Etude prospective technico-économique pour la robotisation de cueillette de tomates." *Agrotique 86, Automatismes et robots en agriculture, Bordeaux, France, conference paper.* , 10 p.

CUADRADO J.L., C.Y. CUADRADO [1986] "AI computer vision.", *Byte,* January 1986, pp. 237-258.

DANIELSSON, P.-E., B. Kruse [1979] "Distance Checking Algorithms", *Computer Graphics and Image Processing*, No. 11, pp. 349-376.

DAVIES, E.R.[1984[ "A glance at image analysis.", *CME* , December 1984, pp. 32-35.

DAVIES, E.R. [1987] "A modified Hough scheme for general circle location",*Pattern Recognition Letters,* Vol. 7, No. 1, January 1987, pp. 37-43.

DUDA, R.O, & P.E. HART [1972] "Use of the Hough Transformation To Detect Lines and Curves in Pictures.", *Graphics and Image Processing*, Vol. 15, No. 1 , pp. 11-15, January.

DUNBAR P. [1986] "Machine Vision : An examination of what's new in vision hardware.", *Byte,* January 1986, pp. 161-172.

DUTCHER, J. [1986] "Machine vision, robotics: looking labor problems straight in the eye.", *Farm Computer News,* Vol. 2, No. 3, pp. 30-32, May.

D'ZMURA, M., P. Lennie [1986], "Mechanisms of color constancy", *Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1662-1672.

EDWARDS, I. [1989] "The Nomenclature of Radiometry And Photometry", *Lasers & Optronics*, August 1989, pp. 37-42.

FAUGERAS, O., [1988] "Les machines de vision", *La recherche*, Vol. 19, No. 204, Novembre 1988, pp 1334-1346.

FU, K.S., W.D. Keidel, and H. Wolter (editor) [1976] Digital pattern recognition: communicatio and cybernetics 10. Springer-Verlag, New York.

GERSHON, R., A.D. Jepson, and J. K. Tsotsos [1986], "Ambient illumination and the determination of material changes", *Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1700-1707.

GERSHON, R. , A.D. JEPSON [1989] "The computatio of color constant descriptors in chromatic images", *Color Research and Application*, Vol. 14, No. 6, December 1989.

GODDARD W.B, M. O'BRIEN C. LORENZEN, D.W. WILLIAMS [1970] "Development of criteria for mechanization of grading processing tomatoes",*American Society of Agricultural Engineering (ASAE) paper* No 70-389.

GOLUB, G.H., C.F. van Loan [1983], *Matrix Computations*; John Hopkins University Press, Baltimore, Md., 1983.

GONZALEZ, R.C., P. WINTZ [1987] *Digital Image Processing,* Addison-Wesley Publishing Co., Reading, Massachusetts, 1987, 503 p.

GROB B.[1984], *Basic Television and Video Systems,* McGraw-Hill, New York, 1984, 464 p.

GRAND D'ESNON, A. [1985] "Robotic harvesting of apples", *Proceedings of the Agri-Mation 1. ASAE Publication* 1-85, pp. 210-214.

HARALICK, R.M. [1986] "Computer Vision Theory: The Lack Thereof",*Computer Vision , Graphics and Image Processing,* Vol. 36, No. 2-3, 1986, 372-386.

HEALEY G., T.O. Binford [1988] "A Color Metric for Computer Vision",*The IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* June 1988, pp. 10-17.

HERON,J.R.,K.H. Kromer, G.L. Zachariah [1971] "Variation of tomato reflectance properties in maturity evaluation",*American Society of Agricultural Engineering (ASAE) paper* No 71-329.

HERON, J.R. , G.L. Zachariah [1974] "Automatic Sorting of Processing Tomatoes", *Transaction of the ASAE,* Vol. 17, No. 5, 1974, pp. 987-992.

HORN, B.K.P. [1974] "Determining lightness from an image" *Computer Graphics and Image Processing*, Vol. 3, pp. 277-299.

HORN, B.K.P. [1982] "Exact Reproduction of Colored Images", *Computer Vision, Graphics, and image processing*, Vol. 26, No. 2, pp. 135-167.

HORN B.K.P., & K. IKEUCHI [1984] "The Mechanical Manipulation of Randomly Oriented Parts.", *Scientific America,* Août 1984, pp. 100-111.

HORN, B.K.P [1986] *Robot Vision,* The MIT Press, Cambridge, Massachusetts, 1986, 509 p.

HOUGH, P.V.C. [1962] "Method and means for recognising complex patterns", US Patent 3069654.

HU, M.K. [1962]. "Visual Pattern Recognition by Moment Invariants", *IRE Trans. Info. Theory*, Vol. IT-8, pp. 179-187.

HURLBERT, A. [1986] "Formal connection between lightness algorithms",*Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1684-1693.

IMBERT, M. [1983] "La neurobiologie de l'image.", *La Recherche*, No. 144, mai 1983, special: La révolution des images, pp. 600-613.

ITOH H., [1984] "Distance Measuring Method Using Only Simple Vision Constructed for Moving Robots.", *Systems Computers Controls*, Vol. 15, No. 3, pp. 10-18, March.

JUDD, D.B., D.L. Mac Adam, G.W. Wyszecki [1964],"Spectral distribution of typical daylight as a function of correlated color temparature", *Journal of Optical Society of America*, Vol. 54, 1964, pp.1031-1040.

KAWAMURA N.,K. NAMIKAWA, T. FUJIURA, M. URA [1985] (Part 1 & 2) "Study on Agricultural Robot (Part 2)" *Faculty of Agirculture, Kyoto University, kyoto, Japan*, 6 p., (Ecrit en japonais).

KEELER, R. [1985] "Vision Guides Machines in Automated Manufacturing.", *Electronic Packaging & Production*, Vol. 25, No. 5, pp. 58-65.

KIJNER, S. [1983] "Le traitement d'images I-: échantillonnage, codage, restauration d'une image.", *Micro-Systemes*, Novembre 1983, pp. 115-135.

KIJNER, S. [1983] " Le traitement d'images II-: extraction de l'information et reconnaissance des formes.", *Micro-Systemes*, Décembre 1983, pp. 125-141.

KIMME C., D. BALLARD, J. SKLANSKY, [1975] "Finding Circles by an Array of Accumulators.", *Communications of the ACM*, Vol. 18, No. 2, February.

KLINKER, G.J. [1989] "A Physical approach to color image understanding", Ph.D. Thesis, Supervisor: Steven Shafer, source: Dissertation Abstracts , Vol. 9, No. 11, may 1989, pp. 4904-B

KRANZLER, G.A. [1985] "Applying Digital Image Processing In Agriculture." *Agricultural Engineering*, Vol. 66, No. 3, pp. 11-13, March.

KRINOV, E. L. [1947] "Spectral reflectance properties of natural formations", Technical Translation TT-439, Natioal Research Council of Canada, 1947.

LAND, E. H. [1974] "The retinex theory of colour vision", Proc. R. Instn. Gr. Br., Vol. 47, pp. 23-58, 1974.

LAND,.E. H. [1977]. "The Retinex Theory of Color Vision",*Scientific America*, December 1977, pp.108-128.

LAND, E. H. [1983] "Recent advances in retinex theory and some implications for cortical computations: Color vision and the natural image", *Proceeding of the National Academy of Science of USA*, Vol. 80, August 1983, pp.5163-5169.

LAND, E. H. [1986] "An alternative technique for the computation of the designator in the retinex theory of color vision", *Proceeding of the National Academy of Science of USA*, Vol. 83, May 1986, pp.3078-3080.

LAND, E. H. [1986] "Recent Advances in retinex theory", *Vision Research*, Vol. 26, No. 1, pp.7-21.

LEE, H.-C. [1986] "Method for computing the scene-illuminant chromaticity from specular highlights", *Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1694-1699.

LLOYD, S.A. [1985] "A Dynamic Programming Algorithm for Binocular Stereo Vision." *The GEC Journal of Research,* Vol. 3, No. 1, pp. 18-24.

LUH J.Y.S. [1983] "Industrial Robot that moves around unexpected obstacles with a minimum distance."

MALONEY, L. T., B.A. Wandell[1986] "Color constancy: a method for recovering surface spectral reflectance",*Journal of the Optical Society of America A*, Vol. 3, No.1, Janvier 1986, pp 29-33.

MALONEY, L. T. [1986] "Evaluation of linear models of surface spectral reflectance with small numbers of parameters", *Journal of the Optical Society of America A*, Vol. 3, No. 10, October 1986, pp.1673-1683.

MANIMALETHU, A. [1983] "Agricultural Robot Application.", *Society of Manufacture Engineer,* MS83-361, 12 p.

MARTELLI, A. [1972] "Edge Detection Using Heuristic Search Methods.", *Computer Graphisc and Image Processing,* Vol. 1, No. 2, pp. 169-182.

MARTELLI, A. [1976] "An Application of Heuristic Search Methods to Edge and Contour Detection.", *Graphics and Image Processing,* Vol. 19, No. 2, pp. 73-83.

MARTIN, E. [1988] "Choosing CCDs for machine vision applications", *Lasers & Optronics*, June 1988, pp.65-69.

MASLAND, R. [1987] "L'architecture fonctionnelle de la rétine.", *Pour la Science*, Février 1987, pp. 95-104.

MEYER Y., S. JAFFARD, O. RIOUL [1987] "L'analyse par ondelettes.", *Pour la Science,* Septembre 1987, pp. 28-36.

MOINI S., M. O'BRIEN [1978] "Tomato Color Measurement versus Maturity",*Transaction of the ASAE,* Vol. 21, No. 4, 1978, pp. 797-800.

MOINI S., M. O'BRIEN, P. CHEN [1980] "Spectral Properties of Mold and Defects of Processing tomatoes", *Transaction of the ASAE*, Vol. 23, No. 5, 1980, pp. 1062-1064.

MÖLLER, K.D [1988] *Optics* , University Science Books, Mill Valley, California, 1988, 400p

MONTGOMERY Geoffrey [1988] "Color perception : Seeing with the brain", *Discover*, December 1988, pp. 52-59.

MORICE G. [1986] "La robotique ou la faillite", *Science & Vie*, No. 830, Novembre 1986, pp. 109-113.

NEVIATIA, R. [1982] *Machine perception*, Prentice-Hall Inc., Englewood Cliffs, NJ.

NISHIHARA, H.K. [1984] "Practical real-time imaging stereo matcher.", *Optical Engineering,* Vol. 23, No. 5, pp. 536-545, October.

OHTA, Y., T. Kanade, T. Sakai [1980] "Color Information for Region Segmentation", *Computer Graphics and Image Processing*, No. 13, pp. 224-241.

PARKKINEN, J.P.S., J. Hallikainen, T. Jaaskelainen [1989], "Characteristic spectra of Munsell colors", *Journal of the Optical Society of America A*, Vol. 6, No. 2, February 1989, pp.318-322.

PARRISH, E.A.Jr.[1977], A.K. Goksel, "Pictorial Pattern Recongnition Applied to Fruit Harvesting", *Transactions of the ASAE*, Vol. 20, No 5, 1977, pp. 822-827.

PEJSA, J. H., A.K. Orrock [1984], "Intelligent Robot Systems: Potential Agricultural Applications", *First International Conf. on Robotics and Intelligent Machines in Agriculture (ASAE publication)*, No 4, 1984, pp. 104-111.

PEJSA, J. H. [1984] "Agricultural robots and economic issues." *Agric. Engr*,64(11), pp. 15-16.

PELLICANO, P. N.E. [1986] "Detection of specularities in colour images using local operators", *Graphics Interface 86 Vision Interface 86*, pp. 370-374.

PERSOON, E. [1976] "A New Edge Detection Algorithm and Its Applications in Picture Processing.", *Computer Graphics and Image Processing*, Vol. 5, No. 4, pp. 425-446.

POGGIO, T. [1984] "Vision humaine et vision par ordinateur.", *Pour la Science*, Juin 1984, pp. 48-58.

POGGIO, T., C. KOCH, [1987] "La perception visuelle du mouvement.", *Pour la Science*, Juillet 1987, pp. 26-31.

POWERS, J. B., J. T. GUNN, and F. C. JACOB [1953] "Electronic color sorting of fruits and vegetables", *Agricultural Engineering*, Vol. 34, No. 3, pp. 149.

PRATT W.K.[1978], Digital Image Processing, *John Wiley & Sons*, New York, 1978, 750 p.

RAO A.R., JAIN R. [1988], "Knowledge Representation an Control in Computer Vision Systems.", *IEEE Expert*, Vol. 3, No. 1, pp. 64-79.

REGTIEN, P.P.L. [1986] "Sensors for applications in robotics", *Sensors and Actuators*, Vol. 10, no. 3-4, pp. 195-218, August.

RIUTALA, M.W., C. C. HSU [1868] "A feature detection program for patterns with overlapping cells", *IEEE Trans. on Sys. Sci. & Cyb.*, SSC-4, March.

ROBIN, RUMPLER [1982] "Méthodes et moyens micro-informatique pour l'automatisation du tri des fruits selon masse, maturité, couleur", Thèse D. ing., INSA Lyon, 1982.

RUEB, K.D., A.K.C. Wong [1988], "Knowledge-based visual part identification and location in a robot workcell",*International Journal of machine tools Manufacturing*, Vol. 28, No. 3, pp. 235-249, 1988.

RUOCCO, S.R. [1987] "Derivation of a computer model for the front end of a robot 3-dimensional vision sensor", *IEEE Proceeding*, Vol. 134, No. 6 , December 1987, pp.339-344.

SAHOO, P.K., S. Soltani, and A.K.C. Wong [1988] "A survey of Thresholding Techniques", *Computer Vision, Graphics, and Image Processing*, Vol. 41, No. 2, 1988, pp. 233-260.

SALLSTROM, P. [1973] "Colour and physics; Some remarks concerning the physical aspects of human colour vision," Inst. phys. Rep. 73-109, 1973

SARKAR, N., R.R. Wolfe [1984] "Feature extraction techniques for sorting tomatoes for computer vision", *American Society of Agricultural Engineering (ASAE)* paper No 84-6018

SARKAR, N. , R.R. Wolfe [1984], "Computer Vision based System for quality separation of fresh market tomatoes", *ASAE Paper*, No. 84-6539.

SITES, P.W., M. J. Delwiche [1985] "Computer Vision to locate fruit on a tree", American Society of Agricultural Engineers, Paper No. 85-3039

SHAFER, Steven A. [1985], "Using Color to Separate Reflection Components", *Color Research and application*, Vol. 10, No. 4, Winter 1985, pp.210-218.

STILES,W.S., G.W. Wysecki [1962]. "Counting Metameric Object Colors", *Journal of the Optical Society of America*, Vol.52, March 1962, pp.313-328.

STILES, W.S., G.W. Wysecki, N. Ohta [1977]."Counting metameric object-color stimuli using frequency-limited spectral reflectance functions", *Journal of the Optical Society of America,* Vol 67, No.6, June 1977, pp.779-785.

STINE,Wm. Wren , J.W. Sparrow [1989] "Influence Theory for Color Constancy Models", *Color research and application*, Vol. 14, No. 2, April 1989, pp.86-95.

SHARPE,Landsay T.[1987] "A Landslide for Colour Science?", *Color research and application*, Vol. 12, No. 2, April 1987, pp.81-84.

TOMINAGA, S. [1984] "A mapping method for computer color vision", *Proceedings 7th International Conference on Pattern Recongnition*, pp. 650-652.

TOMINAGA, S. [1987] "Expansion of color images using three perceptual attributes", *Pattern Recognition Letters*, Vol. 6, No. 1, June 1987, pp.77-85.

TOMINAGA Shoji, Brian a. Wandell [1989] "Standard surface-reflectance model and illuminant estimation.", *Journal of the Optical Society of America A*, Vol. 6, No. 4, April 1989, pp. 576-584.

TREISMAN, A. [1987] "L'identification des objets visuels.", *Pour la Science*, Janvier 1987, pp. 51-60.

TSAI, T. Y [1987] "A Versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses", *IEEE Journal of robotics and automation*, Vol. RA-3, No. 4, August 1987, pp. 323-

TUTLE, E. G. [1984] "Image Controlled robotics in agricultural environments", *First International Conf. on Robotics and Intelligent Machines in Agriculture (ASAE publication)*, No 4, 1984, pp. 84-95

UEDA, M., F. Matsuda, S. Sako [1980] "Color Sensing System for an industrial robot", *Proc. 10th Int. Symp. Industrial Robots*, Milan, mars 1980, pp. 153-162.

WADHWA, S., J. Browne [1988] "Analysis of collision avoidance in multirobot cells using petri nets", *Robotsysteme*, Vol.4, No.2,1988,pp.107-115.

WANDELL B.A. [1987] "The Synthesis and Analysis of Color Image",*IEEE Transaction on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, No. 1, January 1987, pp. 2-13.

WEISS L.E., A.C. SANDERSON, C.P. NEUMAN [1987] "Dynamic Sensor-Based Control of Robots with Visual Feedback.", *IEEE Journal of robotics and automation,* Vol. RA-3, No. 5, pp. 404-416, October.

WHITTAKER, A. D. [1984], G.E. Miles, O.R. Mitchell, I.D. Gaultney, "Fruit Location in a Partially Occluded Image", *ASAE Paper*, No 84-5511.

WIEJAK, J.S. & H. BUXTON & B.F. BUXTON [1985] "Convolution with Separable Masks for Early Image Processing", *Computer Vision, Graphics, and Image Processing,* Vol. 32, No. 3, pp. 279-290.

WYSZECKI, G. & W.S. STILES [1982] Color Science: Concepts and Methods, Quantitative Data and Formulae, *John Wiley & Son*, New York, 1982, pp. 950

ZUECH, N. [1988] "Vision moves in on manufacturing",*Photonics spectra*, April 1988,pp.151-152.

# ANNEXE I

# QUELQUES THÉOREMES D'ALGEBRE LINÉAIRE

# Annexe I

## QUELQUES THÉOREMES D'ALGEBRE LINÉAIRE

Cette annexe contient une collection de définitions et de théorèmes auxquels on fait référence au chapitre XII du mémoire. Les démonstrations de ces théorèmes ont été omises mais elles peuvent facilement être retrouvées dans la plupart des livres qui traitent d'algèbre linéaire.

### DÉFINITION 1

Soit E un espace vectoriel et F et G deux sous-espaces vectoriel de E. On dit que F et G sont deux sous-espaces supplémentaires de E, et l'on notera $E = F \oplus G$, si et seulement si $\forall z \in E$, il existe une paire unique ($z_F \in F, z_G \in G$) telle que $z = z_F + z_G$. On appellera $z_F$ la projection de z sur F parallèlement à G et on appellera $z_G$ la projection de z sur G parallèlement à F.

### THÉOREME 1

Soit E un espace vectoriel et F et G deux sous-espaces supplémentaires de E, la fonction qui associe à tout vecteur z de E sa projection sur F parallèlement à G,que l'on notera $Proj_{F//G}[z]$ et la fonction qui associe à tout vecteur z de E sa projection sur G parallèlement à F, que l'on notera $Proj_{G//F}[z]$. L'opérateur linéaire $Proj_{F//G} : E \dashrightarrow F$ est appelé le projecteur sur F parallèlement à G et l'opérateur linéaire $Proj_{G//F} : E \dashrightarrow G$ est appelé le projecteur sur G parallèlement à F. Ces opérateurs linéaires ont les propriétés suivantes:

- $I = Proj_{F//G} + Proj_{G//F}$ où $I$ est l'opérateur unitaire sur E.

- $Proj_{F//G} = Proj_{F//G} \circ Proj_{F//G}$ et $Proj_{G//F} = Proj_{G//F} \circ Proj_{F//G}$

THÉOREME 2

Soit E un espace vectoriel sur le corps des réels $\Re$ et soit $B = \{ b_i \in E \}_{i \leq N}$ une base de E, alors pour tout vecteur $z \in E$, il existe une famille unique de scalaires appelés les coordonnées de z relativement à la base B, que l'on notera $\alpha = \{ \alpha_i \in \Re \}_{i \leq N}$ ,et telles que $z = B^T \alpha$.

DÉFINITION 2

Soit E un espace vectoriel euclidien sur $\Re$ , deux vecteurs $x,y \in E$ sont dits orthogonaux si et seulement si $\langle x \mid y \rangle = 0$.

DÉFINITION 3

Soit E un espace vectoriel euclidien[1] sur $\Re$ et F un sous-ensemble de E, l'ensemble des vecteurs de E qui sont orthogonaux à tous les vecteurs de F est appelé le complément orthogonal de F et il est noté $F^{\perp}$.

THÉOREME 3

Soit E un espace de Hilbert[2] sur $\Re$ et F un sous-ensemble de E, l'ensemble $F^{\perp}$ est un sous-espace fermé de E.

---

[1]Rappelons qu'un espace vectoriel euclidien est un espace vectoriel sur lequel on a défini un produit scalaire $< / >$.
[2]Rappelons qu'un espace de Hilbert est un espace vectoriel euclidien qui est complet et séparable.

THÉOREME 4

Soit E un espace de Hilbert et $f = \{ f_i \in E \}_{i \leq N}$ une famille finie de vecteurs de E, l'ensemble des combinaisons linéaires des vecteurs appartenant à la famille $f$ est un sous-espace vectoriel fermé de E que l'on appelle l'espace généré par $f$ et que l'on note $[f]$. La famille $f$ sera une base de $[f]$ si et seulement si elle est une famille de vecteurs linéairement indépendants.

THÉOREME 5

Soit E un espace de Hilbert et $f$ une famille de vecteurs de E, $f^{\perp\perp} = [f]$ et $f^{\perp} = [f]^{\perp}$.

THÉOREME 6

Soit E un espace de Hilbert et F un sous-espace fermé de E, alors F et $F^{\perp}$ sont deux sous-espaces supplémentaires de E, c'est-à-dire que $E = F \oplus F^{\perp}$ , et $F = F^{\perp\perp}$.

DÉFINITION 4

Soit E un espace vectoriel euclidien et $f = \{ f_i \in E \}_{i \leq N}$ une famille finie de vecteurs de E, la matrice suivante: $G(f) = \{ \langle f_i \mid f_j \rangle \}_{\substack{i \leq N \\ j \leq N}}$ sera appelée la matrice de Gram associée à la famille $f$.

THÉOREME 7

Soit E un espace vectoriel euclidien et $f = \{ f_i \in E \}_{i \leq N}$ une famille finie de vecteurs de E, la matrice de Gram associée à la famille $f$ n'est pas singulière si et seulement si c'est une famille de vecteurs linéairement indépendants.

DÉFINITION 5

Soit E un espace vectoriel euclidien et $\mathbf{f} = \{ f_i \in E \}_{i \leq N}$ une famille finie de vecteurs de E, et x un vecteur quelconque de E, alors $\mathbf{f}\langle x \rangle = \{ \langle x \mid f_j \rangle \}_{i \leq N}$

THÉOREME 8

Soit E un espace de Hilbert sur $\Re$ et $\mathbf{f} = \{ f_i \in E \}_{i \leq N}$ une famille finie de vecteurs linéairement indépendants de E, alors le projecteur sur $[\mathbf{f}]$ parallèlement à $\mathbf{f}^{\perp}$ est donné par l'expression suivante:

$$\forall z \in E,\ \mathrm{Proj}_{F//G}[z] = \mathbf{f}^{T}\, G(\mathbf{f})^{-1}\, \mathbf{f}\langle x \rangle \quad \in [\mathbf{f}]$$

où $G(\mathbf{f})^{-1}\, \mathbf{f}\langle x \rangle \in \Re^N$ représente les coordonnées du vecteur $\mathrm{Proj}_{F//G}[z]$ de $[\mathbf{f}]$ relativement à la base $\mathbf{f}$ .

# ANNEXE II

# LE CHOIX

# D'UNE CAMÉRA

# Annexe II

## LE CHOIX D'UNE CAMÉRA

Cette annexe vise à mettre en relief certaines des caractéristiques qu'il faudra considérer lors du choix d'une caméra pour un robot-cueilleur de tomates.

Le marché de la télévision est beaucoup plus important pour les manufacturiers de caméras que celui de la vision artificielle de sorte que la plupart des caméras sur le marché répondent aux besoins de la télévision plutôt qu'à ceux de la vision artificielle, quoique parfois ces derniers besoins peuvent être conciliés aux premiers.

Une caméra couleur possède en général trois senseurs qui sont respectivement sensibles dans les régions spectrales du rouge, du vert et du bleu et on désignera les signaux à la sortie de ces senseurs par les lettres R, G, et B respectivement. Une des caractéristiques les plus importantes d'une caméra est le protocole (ou le standard) de communication auquel les signaux à la sortie de la caméra se conforment. Les caméras vidéos destinées aux consommateurs ordinaires suivent en général l'un des protocoles de communication suivants: NTSC en amérique du nord, SCIC, PAL, ou SECAM dans les autres parties du monde. Ces standards ont été originellement conçus en fonction de la transmission herztienne des signaux de télévision et ont , à l'exception de SECAM, une largeur de bande totale de 6 Mhz. Pour être conforme au standard NTSC, les trois signaux R,G,B doivent être transformés dans un autre système colorimétrique (le système YIQ ) où le signal Y transporte l'information de luminance et où les deux signaux de chrominances I et Q transportent l'information couleur proprement dite. Ensuite ces trois signaux (YIQ) ainsi que les signaux de synchronisation sont modulés et sommés pour former le signal vidéo composite présent à la sortie de la caméra. Or, pour respecter cette contrainte de 6 Mhz de largeur de bande totale pour le signal vidéo composite et les contraintes propres au visionnement des images sur une

télévision, la plus grande partie de cette largeur de bande est réservée au signal d'intensité (Y) et les signaux de chrominances doivent être transmis sur une largeur de bande de seulement 1,8 Mhz. L'acuité de l'information couleur présente dans le signal vidéo composite du standand NTSC est donc réduite. Comme la reconnaissance de la couleur est très importante pour la localisation des tomates dans une image, une caméra vidéo couleur respectant le standard NTCS ne saurait être un choix judicieux. Une autre raison pour ne pas utiliser une caméra respectant le standard NTSC est que ce standard exige que la forme des "pixels" d'une image soit dans un rapport de 3 verticalement à 4 horizontalement. Nous avons proposé, dans les algorithmes de localisation des tomates (chapitre VII) de localiser les cercles via la transformation de Hough circulaire; cette approche ne fonctionnera évidemment que dans la mesure où les cercles demeurent des cercles. Il faut donc utiliser, pour notre application, une caméra dont les "pixels" sont carrés.

Il existe un type de caméra vidéo appelé "caméra RGB" qui transmet directement à sa sortie les trois signaux R,G,B sur trois lignes séparées. Les besoins du marché pour ce type de caméra étaient dans le passé réduits aux studios de télévision, mais ces dernières années ont vu apparaître des caméras RGB réalisées à partir de senseurs CCD qui ont, entre autres, l'avantage d'être peu dispendieuses. Une caméra RGB est le choix approprié pour notre application.

Pour notre application, il n'est pas nécessaire d'utiliser une caméra vidéo, c'est-à-dire une caméra qui prend trente images par seconde, une caméra envoyant de manière asyncrhone une image sur commande de l'ordinateur hôte ferait tout aussi bien l'affaire. Ce type de caméra est référé sous le nom de "monoshot camera". Un des avantages de ce type de caméra est que le temps de mesure (ou d'intégration) peut être varié sous la commande de l'ordinateur. Cela offre l'avantage de pouvoir varier la sensibilité spectrale moyenne de la caméra en fonction des conditions d'illumination.

Un autre facteur important dans le choix d'une caméra est sa linéarité. En effet, le modèle de la caméra qui est utilisé par l'algorithme de reconnaissance de la couleur suppose que la caméra soit linéaire. La plage dynamique de linéarité des senseurs devra être la plus grande possible ainsi que

le ratio signal-bruit. Afin de réduire le bruit, il serait avantageux (mais pas nécessaire) que la digitalisation des signaux en provenance des senseurs se fasse à l'intérieur même de la caméra.

L'objectif de la caméra devra avoir un auto-focus.

Les technologies CCD, MOS ou CID s'imposent de plus en plus aujourd'hui pour la fabrication des senseurs. La résolution de ces senseurs s'améliore de plus en plus et leur coût diminue sans cesse. Leurs principaux avantages, vis-à-vis la technologie Vidicom traditionnelle, sont une grande précision dans le positionnement des "pixels", la miniaturisation, la robustesse aux chocs, leur linéarité, la possibilité de les utiliser dans des environnements où il y a des champs électromagnétiques intenses, et leurs coûts réduits.

Le lecteur intéressé à en savoir plus sur les caractéristiques importantes qu'il faut considérer lors du choix d'une caméra pour une application en vision artificielle, peut référer aux articles suivants: [Martin 88] et [Dunbar 86].

# ANNEXE III

# PROGRAMMES DE TRAITEMENT D'IMAGES

# Annexe III

## PROGRAMMES DE TRAITEMENT D'IMAGES

Les programmes présentés dans cette annexe ont été écrits de l'automne 87 au printemps 88, et visaient à vérifier un certain algorithme de reconnaissance de la couleur rouge et à vérifier la possibilité d'utiliser la transformation de Hough circulaire afin de faire la localisation des tomates dans une image.

Liste des programmes



Tous ces programmes ont été écrits en C (MS-C version 4) et doivent être compilés et "linkés" en utilisant le modèle "LARGE" des librairies, car les donnés occupent plus de 64K. La plupart de ces programmes nécessitent la présence de la carte digitalisante.

Le système de vision comprend:

- un micro-ordinateur IBM-PC,

- une caméra couleur de type Vidicom dont les signaux de sortie (signal vidéo composite) suivent le standard NTSC,

- une carte digitalisante (ou échantillonneur de trames) modèle DT2802 manufacturé par la compagnie Data Translation Inc, qui est connectée sur le bus de l'IBM-PC et sert d'interface entre la caméra et l'ordinateur. Cette carte ne permet de digitaliser que le signal de luminance (Y) du signal vidéo composite et ce, à la vitesse de 30 images par seconde. La carte digitalisante ignore donc les signaux de chrominances en provenance de la caméra, de sorte qu'avec cette carte la caméra couleur se comporte à tout fin pratique comme une caméra monochrome. L'image digitale contient 256 "pixels" par ligne et 240 lignes. La carte permet une digitalisation sur 6 bits (64 niveaux de quantisation). La carte contient 64K de mémoire afin de mémoriser chaque image digitalisée et cette mémoire correspond au segment de la mémoire de l'IBM-PC commençant à l'adresse A0000000 (notation hexadécimale ou en base 16). La carte possède un interface graphique qui permet d'afficher l'image présente dans la mémoire de la carte sur un moniteur RGB. Comme nous utilisions une télévision ordinaire noir et blanc, nous avons interfacé le signal G à la sortie de la carte digitalisante via un modulateur RF dont la sortie est connectée à la prise d'antenne de la TV (voir fig. 23).

Un programme de support de la carte permet de contrôler de façon interactive les fonctions de la carte (digitalisation d'une image sur commande ou digitalisation en continue de 30 images/sec, etc), ainsi que de permettre de faire certaines opérations simples sur ces images (sauver une image dans un fichier, lire un fichier image et mettre cette image dans la mémoire de la carte, additionner un "offset" à l'image , etc...).

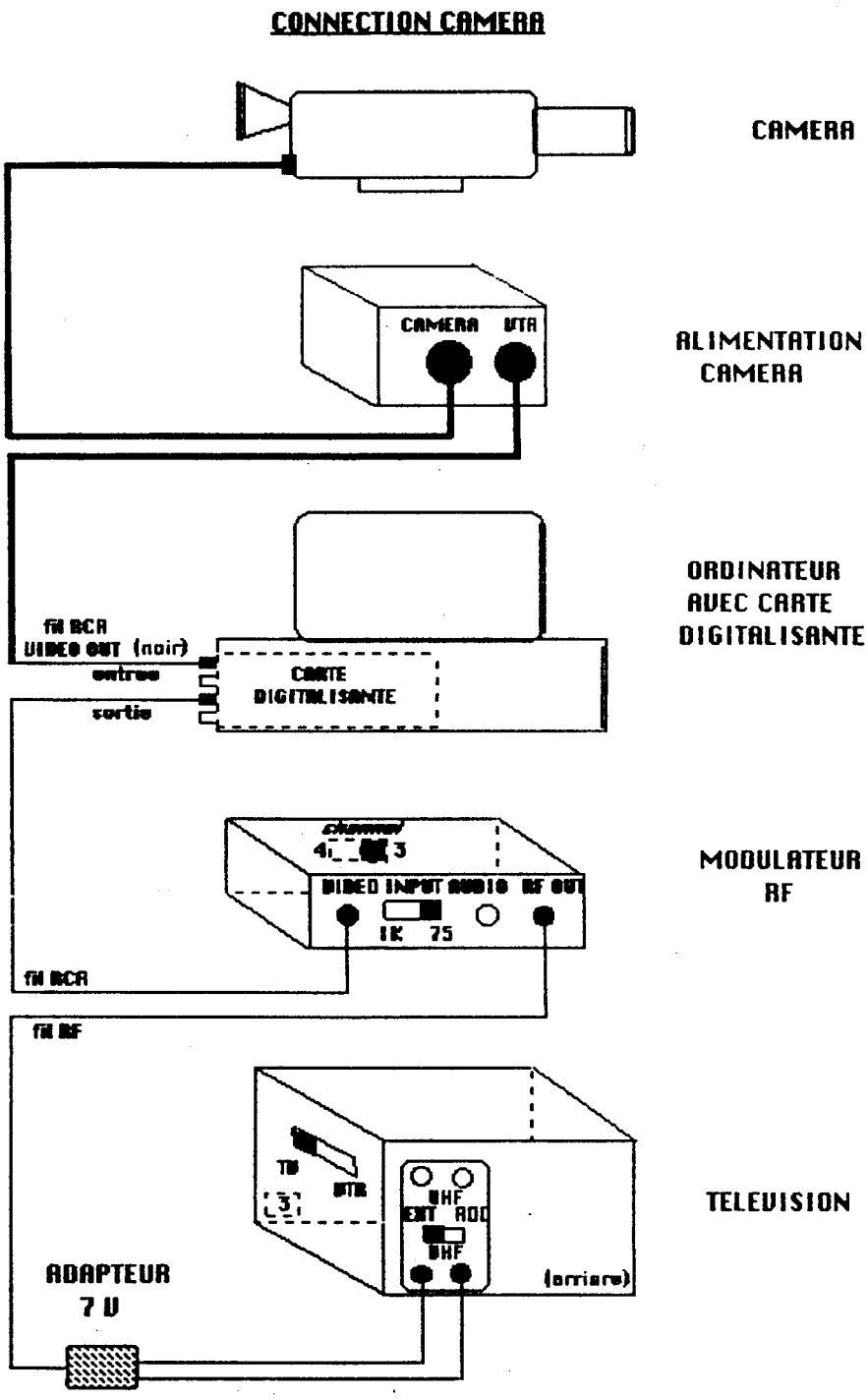



Figure 25. Connection à la carte digitalisante (modèle DT2802) d'une caméra vidéo et d'un moniteur TV noir et blanc

Programmes utilisés pour la formation d'une image binaire rouge:

COLOR_B, COLOR_D, HISTO, THRESHOD

À l'automne 87 nous n'avions pas encore pris connaissance des recherches en reconnaissance visuelle de fruits dont nous faisons la description au chapitre VI de ce mémoire. Nous cherchions un moyen d'obtenir une image binaire rouge, c'est-à-dire une image dont la valeur des "pixels" vaut 63 lorsque leurs couleur est rouge et 0 lorsque leur couleur est différente du rouge. Comme nous ne disposions pas avec cette carte digitalisante de la possibilité de digitaliser les signaux de chrominances de la caméra, nous avons pensé faire l'acquisition de deux images de la scène, l'une non-filtrée et l'autre filtrée à l'aide d'un filtre rouge (simple platique transparent teinté que nous nous sommes procuré dans un studio de photograhie). Avant de faire l'acquisition de la deuxième image, nous placions le filtre rouge devant l'objectif de la caméra. L'idée que nous avions trouvé alors pour obtenir cette image binaire rouge était de soustraire ces deux images et de binariser l'image résultante de la façon suivante:

$$I_d(X) = I_f(X) - I(X) \qquad [84]$$

où $I(X)$ Valeur de l'image non-filtrée au "pixel" X

$I_f(X)$ Valeur de l'image filtrée au "pixel" X

$$I_b(X) = \begin{cases} 63 & \text{si} \quad I_d(X) \geq T \\ 0 & \text{si} \quad I_d(X) < T \end{cases} \qquad [85]$$

où T est un seuil que l'on détermine expérimentalement à partir de l'histogramme de $I_d(X)$.

Le programme COLOR_B.C permet de lire deux fichiers images dont le premier est supposé contenir une image non-filtrée et le deuxième est supposé contenir l'image filtrée correspondante. Le programme demande à l'utilisateur le seuil T qu'il doit utiliser pour binariser l'image selon l'équation [85]. Finalement le programme conserve dans un fichier cette image binaire rouge.

Le programme COLOR_D est identique à COLOR_B sauf qu'il ne procède pas à la binarisation de l'image; il conserve dans un fichier la différence entre les deux images (plus 20). Le programme HISTO peut être utilisé pour déterminer un seuil approprié pour la binarisation et le programme THRESHOD peut ensuite faire la binarisation selon l'équation [85].

Programmes pour segmenter l'image binaire rouge: TROUVE, SEGMENT

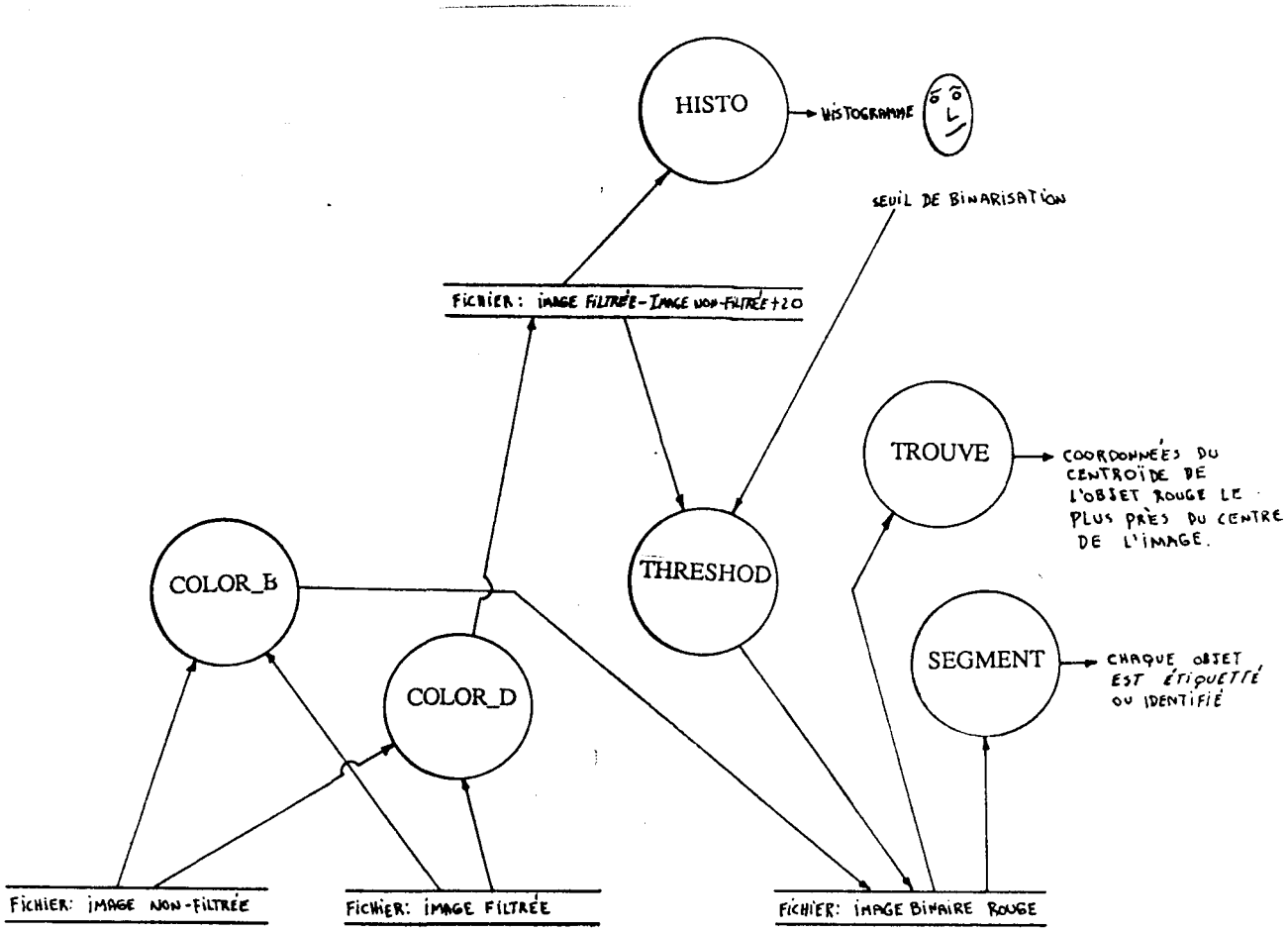



Figure 26. Formation et segmentation d'une image binaire rouge à l'aide des programmes: COLOR_B, COLOR_D, HISTO, THRESHOD, TROUVE, SEGMENT.

Programmes pour la formation d'une image binaire de contours: CONT_B, EDGE_1 à EDGE_4, HISTO, THRESHOD

Programmes pour la localisation des cercles dans une image binaire de contours: HOUG_1 àHOUG_4

Programmes utilitaires: VISION_1, VISION_2, TCERCLE, EFFACE, C_BIDON

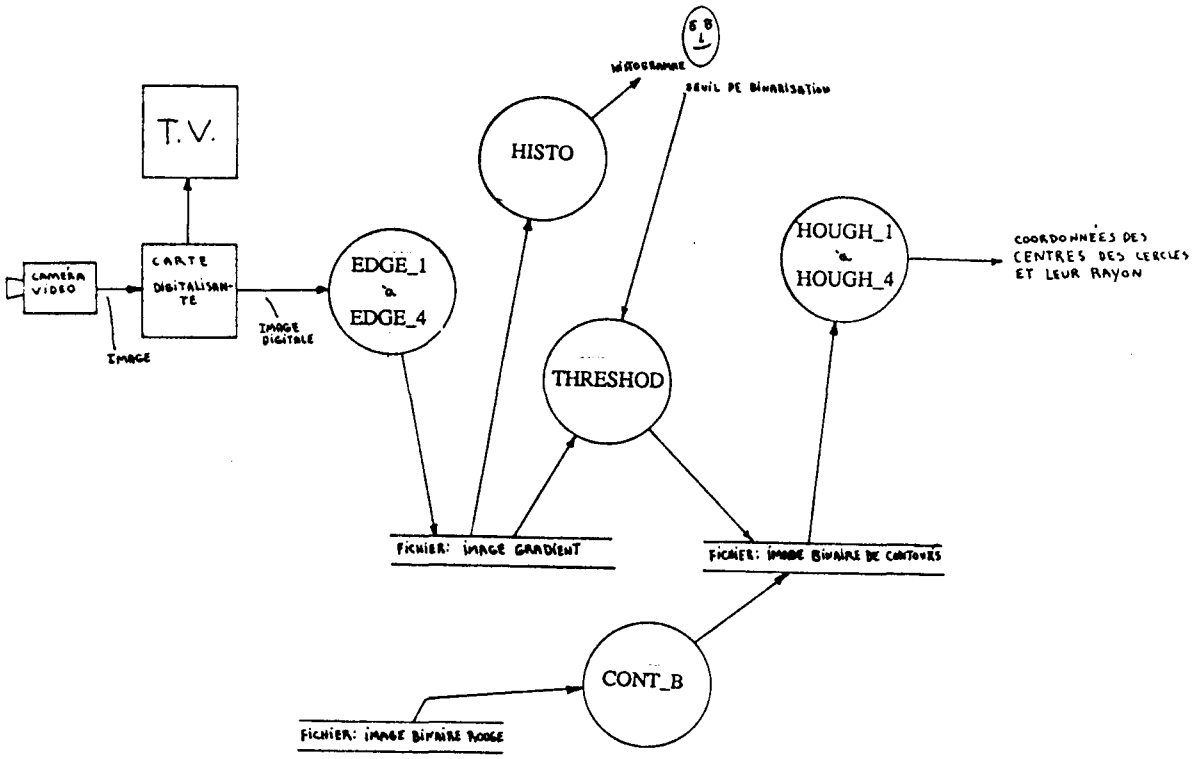



Figure 27. Formation d'une image binaire de contours et localisation des cercles dans celle-ci à l'aide de la transformation de Hough circulaire.

CODE SOURCES DE CES PROGRAMMES:

```
/*  ____________________________________________________________________

    Fichier:  COLOR_B.C

    Ce programme lit une image non filtre dans un fichier et affiche cette
    image a l'ecran via la carte digitalisante.
    Il lit ensuite une image filtre de la meme scene dans un fichier et
    affiche cette image a l'ecran via la carte digitalisante.
    L'image filtre est ensuite soustraite de l'image non filtre et l'image
    resultante est ensuite binarisee a partir d'un seuil ecrit au clavier
    par l'utilisateur.  Cette image binaire est ensuite affiche a l'ecran
    via la carte digitalisante et conservee dans un fichier. On appelle
    cette image binaire "l'image rouge".

    Voici une liste de fichiers contenant des images non filtre:
    TI1.IMG
    TI2.IMG
    TI3.IMG
    TI4.IMG
    TI5.IMG
    TG6.IMG
    LIVR.IMG
    IBM.IMG

    Et voici la liste correspondante des image filtres (filtre rouge):
    TI1_F.IMG
    TI2_F.IMG
    TI3_F.IMG
    TI4_F.IMG
    TI5_F.IMG
    TG6_F.IMG
    LIVR_F.IMG
    IBM_F.IMG

    ____________________________________________________________________*/

#define    IMAGE_SIZE    240 * 256

#include   <stdio.h>
#include   <malloc.h>
#include   <memory.h>
#include   <string.h>

main()
(

char file_name[25];
FILE *file_stream;
char *image,*image_init;
char *image_f,*image_f_init;
char *image_rouge,*image_rouge_init;
char *carte;
long int i;
int offset;
int reponse;

set();  /* fonction de la librairie LVISION */

image =  _fmalloc( IMAGE_SIZE * sizeof(char));
```

```
image_init = image;

image_f =  _fmalloc( IMAGE_SIZE * sizeof(char));
image_f_init = image_f;

image_rouge =  _fmalloc( IMAGE_SIZE * sizeof(char));
image_rouge_init = image_rouge;


/* Lecture de l'image non filtree */

printf("Entrez le nom du fichier contenant l'image non-filtree: ");
gets(file_name);
printf("\n");

if( (file_stream = fopen(file_name,"rb")) == NULL)
   {
   printf("Impossible d'ouvrir le fichier %s\n",file_name);
   exit();
   }

for(i=0; i<IMAGE_SIZE; i++)
  *(image++) = (char) getc(file_stream);

fclose(file_stream);

/* ------------------------------------------------ */


/*  Affichage de l'image non filtree */

printf("\n Affichage de l'image non filtree\n");

carte =  (char *) 0xA0000000;

memcpy(carte,image_init, IMAGE_SIZE);

/* ---------------------------------------- */


 /* Lecture de l'image filtree */

 printf("Entrez le nom du fichier contenant l'image filtree: " );
 gets(file_name);
 printf("\n");

if( (file_stream = fopen(file_name,"rb")) == NULL)
   {
   printf("Impossible d'ouvrir le fichier %s\n",file_name);
   exit();
   }

for(i=0; i<IMAGE_SIZE; i++)
  *(image_f++) = (char) getc(file_stream);

 fclose(file_stream);

 /* ------------------------------------------------ */
```

```
do
{

/*  Lecture de l'offset */

printf("Entree la valeur de l'augmentation minimum d'intensite: ");
scanf("%d", &offset);
fflush(stdin);
printf("\n");

/* ------------------------------------------*/


/* Comparaison des 2 images afin de trouver les points rouges */

printf("Comparaison des 2 images afin de trouver les points rouges\n");

image = image_init;
image_f = image_f_init;
image_rouge = image_rouge_init;

for(i=0; i< IMAGE_SIZE; i++)
*(image_rouge++) = *(image_f++)>((char)(*(image++) + offset)) ? 63 : 0;

/* ------------------------------------------------ */


/*  Affichage de l'image rouge */

carte =  (char *) 0xA0000000;

memcpy(carte, image_rouge_init, IMAGE_SIZE);

/* ---------------------------------------- */

printf("\nVoulez-vous essayer un autre offset?(Y ou N) : ");

} while( (reponse=getchar())=='Y'|| reponse == 'y' );

_ffree ( image_init);
_ffree ( image_f_init);
_ffree ( image_rouge_init);

}
```

```
/*  ------------------------------------------------------------------------

    Fichier:  COLOR_D.C

    Ce programme lit une image non-filtree dans un fichier et affiche cette
    image a l'ecran via la carte digitalisante.
    Il lit ensuite une image filtre de la meme scene dans un fichier et
    affiche cette image a l'ecran via la carte digitalisante.
    L'image filtre est ensuite soustraite de l'image non-filtre et l'image
    resultante est affiche et conserve dans un fichier.

    Voici une liste de fichiers contenant des images non-filtrees:
    TI1.IMG
    TI2.IMG
    TI3.IMG
    TI4.IMG
    TI5.IMG
    TG6.IMG
    LIVR.IMG
    IBM.IMG

    Et voici la liste correspondante des images filtrees (filtre rouge):
    TI1_F.IMG
    TI2_F.IMG
    TI3_F.IMG
    TI4_F.IMG
    TI5_F.IMG
    TG6_F.IMG
    LIVR_F.IMG
    IBM_F.IMG
    ------------------------------------------------------------------------*/

#define    IMAGE_SIZE    240 * 256

#include   <stdio.h>
#include   <malloc.h>
#include   <memory.h>
#include   <string.h>
#include   <vision.h>

main()
{

char *image,*image_init;
char *image_f,*image_f_init;
char *image_rouge,*image_rouge_init;
long int i;


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));

image_f_init =  _fmalloc( IMAGE_SIZE * sizeof(char));

image_rouge_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image non filtree */

printf("Image non_filtree\n");
```

```
lire_image(image_init);

affiche(image_init);


 /* Lecture de l'image filtree */

printf("Image filtree\n");

lire_image(image_f_init);

affiche(image_f_init);

/*  Calcul de l'image des differences d'intensites entre les deux images  */


image = image_init;
image_f = image_f_init;
image_rouge = image_rouge_init;

for(i=0; i< IMAGE_SIZE; i++)
*(image_rouge++) = *(image_f++) - *(image++) + 20;


/* Afficher  la difference d'intensite entre les deux images */

affiche(image_rouge_init);


/*  Sauver sur fichier les diffenrences  */

store_image(image_rouge_init);


_ffree ( image_init);
_ffree ( image_f_init);
_ffree ( image_rouge_init);

}
```

```
/*  ____________________________________________________________________

    Fichier:    HISTO.C

    Ce programme lit une image dans un fichier, affiche cette image a l'ecran
    via la carte digitalisante et fait un histogramme de cette image et
    imprime cette histogramme ainsi qu'un histogramme normalise sur 100.

    ____________________________________________________________________*/

#define   IMAGE_SIZE    61184


#include   <stdio.h>
#include   <malloc.h>
#include   <memory.h>
#include   <string.h>
#include   <vision.h>

main()

{

int j;
int indice;
long i;
long tableau_des_compteurs[64];
char *image_init, *image;
double facteur_normatif;
double tableau_normalise[64];
double nombre_de_points;


nombre_de_points = 60928.0;
facteur_normatif = 100.0 / nombre_de_points;

image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );

printf("Histogramme des intensites d'une image\n");


/* lire le fichier dont on veut faire l'histogramme */

lire_image(image_init);

affiche(image_init);


/*  Initialisation des compteurs a zero  */

for (j=0; j<64; j++)
   tableau_des_compteurs[j] = 0;


/*  Calcul de l'histogramme de l'image  */

image = image_init + 256;

for(i=256; i< IMAGE_SIZE; i++)
   (tableau_des_compteurs [(int)*(image++)])++;
```

```
/*  Normalisation de l'histogramme   */

for(j=0; j<64;  j++)
   tableau_normalise[j] = tableau_des_compteurs[j] * facteur_normatif;

/* Impression des resultats  */

for(i=0;i<16; i++)
   {
   for(j=0;j<4; j++)
        {
        indice = i + 16 * j;
        printf("%-3d %5ld %6.2f * ",indice,tableau_des_compteurs[indice],
        tableau_normalise[indice]);
        }
    printf("\n");
    }


_ffree(image_init);

}
```

```
/*  ____________________________________________________________________

    Fichier: THRESHOD.C

    Ce programme sert a binariser une image digitale (valeur de 0 a 63) selon
    la fonction de binarisation suivante:

                        0    si   image_digitale <   Threshold
    image_binaire =
                        1    si   image_digitale >ou= Threshold

    Le programme lit l'image digitale dans un fichier, affiche cette image
    a l'ecran via la carte digitalisante, lit la valeur du seuil (threshold)
    qu'on entre au clavier, calcule l'image binaire a partir de ce seuil,
    affiche cette image binaire a l'ecran via la carte digitalisante et
    conserve cette image binaire dans un fichier.

    ____________________________________________________________________*/

#define     IMAGE_SIZE      61440

#include    <stdio.h>
#include    <malloc.h>
#include    <vision.h>

main()

{

char *image, *image_init;
char *image_bin, *image_bin_init;
register int ecart;
register unsigned int i;
int threshold;


image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );
image_bin_init = _fmalloc( IMAGE_SIZE * sizeof(char) );


/* Lire le fichier */

lire_image(image_init);


/* Afficher l'image lue  */

affiche(image_init);


/* Lire le threshold  */

printf("Entrez le threshold: ");
scanf("%d", &threshold);
fflush(stdin);

ecart = threshold;
```

```
/*  Binarisation de l'image */

printf("Binarisation de l'image\n");


image = image_init;
image_bin = image_bin_init;


for(i=0; i<IMAGE_SIZE; i++)
  *(image_bin++) = *(image++)<ecart ?  0  :  63;


/*  Afficher l'image binaire  */

affiche(image_bin_init);


/*  Conserver l'image binaire dans un fichier  */

printf("Storage de l'image binaire\n");
store_image(image_bin_init);


_ffree(image_init);
_ffree(image_bin_init);

}
```

```
/*_____________________________________________________________________

  Fichier:  TROUVE.C

  Ce programme lit un fichier contenant une image binaire (rouge)
  Ce programme trouve l'objet rouge (intensite 63) le plus pres du centre de
  l'image binaire rouge. L'image est parcourue a partir du centre suivant une
  une spirale tournant dans le sens horaire.

  Un fois un point de l'objet trouver de cette facon on determine les
  quatre coordonnees d'un rectangle dont chacun de ses cotes touche le point
  extreme de l'objet et les coordonnees de ces points sont affichees a
  l'ecran.
_____________________________________________________________________*/

#define    ADRESSE_CARTE  (char *) 0xA0000000
#define    DELTA          8
#define    X_CENTRE       120
#define    Y_CENTRE       128

#include   <stdio.h>
#include   <vision.h>


main()

{

char *carte;
int x,y;
int x_min,x_max,y_min,y_max;
register int lign_vert_gauch, lign_vert_droit, lign_hor_haut, lign_hor_bas;

int spirale(int *,int *);
int l_v(int,int,int);
int l_h(int,int,int);

carte = ADRESSE_CARTE;

printf("Entrer le nom du fichier contenant l'image rouge\n");
lire_image(carte);

if(!spirale(&x,&y))
  {
  printf("Aucune tomate dans l'image\n") ;
  goto fin;
  }

printf("x initiale :  %d    y initiale : %d  \n",x,y);

x_min = x - DELTA;
x_max = x + DELTA;
y_min = y - DELTA;
y_max = y + DELTA;

do
  {

  lign_vert_gauch = l_v(y_min,x_min,x_max);
```

```
  if(!lign_vert_gauch)
     y_min -= DELTA;

  lign_vert_droit = l_v(y_max,x_min,x_max);

  if(!lign_vert_droit)
     y_max += DELTA;

  lign_hor_haut = l_h(x_min,y_min,y_max);

  if(!lign_hor_haut)
     x_min -= DELTA;

  lign_hor_bas = l_h(x_max,y_min,y_max);

  if(!lign_hor_bas)
     x_max += DELTA;

}while(!lign_vert_gauch || !lign_vert_droit || !lign_hor_haut || !lign_hor_bas )


/* Ajustement fin de la boite */

do
  {

  lign_vert_gauch = l_v(y_min,x_min,x_max);

  if(lign_vert_gauch)
     y_min++;

  lign_vert_droit = l_v(y_max,x_min,x_max);

  if(lign_vert_droit)
     y_max--;

  lign_hor_haut = l_h(x_min,y_min,y_max);

  if(lign_hor_haut)
     x_min++;

  lign_hor_bas = l_h(x_max,y_min,y_max);

  if(lign_hor_bas)
     x_max--;

}while(lign_vert_gauch || lign_vert_droit || lign_hor_haut || lign_hor_bas );

printf("x_min = %d     x_max = %d\n", x_min, x_max);
printf("y_min = %d     y_max = %d\n", y_min, y_max);

fin: ;

}

/*-------------------------------------------------------------------------*/

int l_h(x_ligne, y_min, y_max)
```

```
int x_ligne, y_min, y_max;

{

register int i;
register char *carte;

carte = ADRESSE_CARTE;
carte += (256 * x_ligne + y_min);

for(i=y_min; i<=y_max; i++)
  if(*(carte++)!= 0)
     return(0);

return(1);

}

/* ------------------------------------------------------------------------ */

int l_v(y_ligne, x_min, x_max)

int y_ligne, x_min, x_max;

{

register  int i;
register char *carte;

carte = ADRESSE_CARTE;
carte += (256 * x_min + y_ligne);

for(i=x_min; i<= x_max; i++)
  {
  if(*(carte)!=0)
     return(0);
  carte += 256;
  );

return(1);

}

/* ------------------------------------------------------------------------ */

int spirale(x,y)

int *x,*y;

{

char *carte,*carte_init;
int signe;
register int i,j;

carte = ADRESSE_CARTE;
carte_init = carte;
carte += (256*X_CENTRE + Y_CENTRE);

for(i=1,signe = -1; i<(240/DELTA) && *carte == 0;i++)
```

```
  {
  signe = 0 - signe;
  for(j=0; j<i && *carte == 0; j++)
     carte += signe*DELTA;
  for(j=0; j<i && *carte == 0; j++)
     carte += 256*DELTA;
  };

if(i==240)
  return(0);
else
  {
  *x = (int) (((long)(carte-carte_init)) / 256);
  *y = (int) (((long)(carte-carte_init)) - 256 * *x);
  return(1);
  };


}
```

```
/*  ____________________________________________________________________

    Fichier:  SEGMENT.C

    Ce programme sert a segmenter une image binaire en different objets separes
    selon un algorithme de segmentation appele "object colouring" et presente
    au chapitre 4 du livre "Computer Vision" de Ballard (1982).

    L'image binaire est lue dans un fichier.

    ____________________________________________________________________*/


#define     IMAGE_SIZE    240 * 256

#include    <stdio.h>
#include    <malloc.h>
#include    <memory.h>
#include    <string.h>

main()
{

char file_name[25];
FILE *file_stream;
char *image,*image_init;
char *image_f,*image_f_init;
char *image_rouge,*image_rouge_init;
char *carte;
long int i;
int offset;
int reponse;

set();  /* fonction de la librairie LVISION */

image =  _fmalloc( IMAGE_SIZE * sizeof(char));
image_init = image;

image_f =  _fmalloc( IMAGE_SIZE * sizeof(char));
image_f_init = image_f;

image_rouge =  _fmalloc( IMAGE_SIZE * sizeof(char));
image_rouge_init = image_rouge;


/* Lecture de l'image non filtree */

printf("Entrez le nom du fichier contenant l'image non-filtree: ");
gets(file_name);
printf("\n");

if( (file_stream = fopen(file_name,"rb")) == NULL)
   {
   printf("Impossible d'ouvrir le fichier %s\n",file_name);
   exit();
   }

for(i=0; i<IMAGE_SIZE; i++)
```

```
  *(image++) = (char) getc(file_stream);

fclose(file_stream);

/* ------------------------------------------------ */

/*  Affichage de l'image non filtree */

printf("\n Affichage de l'image non filtree\n");

carte =  (char *) 0xA0000000;

memcpy(carte,image_init, IMAGE_SIZE);

/* ---------------------------------------- */

 /* Lecture de l'image filtree */

 printf("Entrez le nom du fichier contenant l'image filtree: " );
 gets(file_name);
 printf("\n");

if( (file_stream = fopen(file_name,"rb")) == NULL)
   {
   printf("Impossible d'ouvrir le fichier %s\n",file_name);
   exit();
   }

for(i=0; i<IMAGE_SIZE; i++)
  *(image_f++) = (char) getc(file_stream);

 fclose(file_stream);

 /* ------------------------------------------------ */

do
{

/*  Lecture de l'offset */

printf("Entree la valeur de l'augmentation minimum d'intensite: ");
scanf("%d", &offset);
fflush(stdin);
printf("\n");

/* ----------------------------------------*/


/* Comparaison des 2 images afin de trouver les points rouges */

printf("Comparaison des 2 images afin de trouver les points rouges\n");

image = image_init;
image_f = image_f_init;
image_rouge = image_rouge_init;

for(i=0; i< IMAGE_SIZE; i++)
```

```
*(image_rouge++) = *(image_f++)>((char)(*(image++) + offset)) ? 63 : 0;

/* ------------------------------------------------ */

/*  Affichage de l'image rouge */

printf("Affichage de l'imge binaire\n");

carte =  (char *) 0xA0000000;

memcpy(carte, image_rouge_init, IMAGE_SIZE);

/* ---------------------------------------- */

/*  Segmentation de l'image  */

/* Creation d'image a resolution reduite par un facteur FR */
char sous_image[ROW][COLUMN];
char color_image[ROW][COLUMN];

/* initialiser color_image a zero */

memset(color_image, 0, SUB_IMAGE_SIZE);

/* Initialiser la sous_image  */

for(row=0; row < ROW ; row++)
  for(column=0; column < COLUMN; column++)
     sous_image[row][column]=*(image_rouge+256*FR*row+FR*column);

/* Algorihtme de segmentation local */

for(row=1; row< ROW; row++)
  for(column=1; column < COLUMN; column++)
     {
     if(sous_image[row][column]==63)
        {
        if(sous_image[row-1][column]==63 && sous_image[row][column-1]==0)
           color_image[row][column] = color_image[row-1][column];
        else if(sous_image[row][column-1]==63 && sous_image[row-1][column]==0)
           color_image[row][column] = color_image[row][column-1];
        else if(sous_image[row][column-1]==63 && sous_image[row-1][column]==1)
           {
           color_image[row][column] = color_image[row][column-1];
           if(color_image[row][column-1]>color_image[row-1][column])
              {
              relation[re_cnt][0] =  color_image[row][column-1];
              relation[re_cnt][1] =  color_image[row-1][column];
              }
           else
              {
              relation[re_cnt][0] =  color_image[row-1][column];
              relation[re_cnt][1] =  color_image[row][column-1];
              }
            re_cnt++;
            }
        else if(sous_image[row][column-1]==0 && sous_image[row-1][column]==0)
            image_color[row][column] = color++;
        else
```

```
/* ____________________________________________________________________

   fichier source: CONT_B.C

   Ce programme lit une image binaire dans un fichier et localise les
   contours dans cette image binaire.

   Le programme detecte un contour s'il y a une transition de l'intensite
   de 0 a 63 ou de 63 a 0 dans la direction x ou la direction y.  Le contour
   est place sur le "pixel" d'intensite 63.

   L'image binaire de contours resultante est ensuite affiche a l'ecran par
   la carte digitalisante et est conserve dans un fichier.

   ____________________________________________________________________ */

#define    IMAGE_SIZE   61440

#include   <stdio.h>
#include   <malloc.h>
#include   <vision.h>

main()
{

char *image,*image_init;
char *image_plus_un, *image_moins_un;
char *image_plus_256, *image_moins_256;
char *depart;
char *image_contour, *image_contour_init;
register int point;
register int i, j;


printf("Programme de calcul d'une image contour\n");


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));

image_contour_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image  */

lire_image(image_init);

/* Afficher l'image */

affiche(image_init);


/*  Recherche des contours  */
```

```
image_contour = image_contour_init;

for(i=0; i<256; i++)
  *(image_contour++) = 0;


depart = image_init + 256;

image = depart;
image_plus_un = depart+1;
image_moins_un = depart-1;
image_moins_256 = depart - 256;
image_plus_256 = depart + 256;


for(i=0; i<239; i++)
  {
  *(image_contour++) = 0;
  image++;
  image_moins_un++;
  image_plus_un++;
  image_moins_256++;
  image_plus_256++;


  for(j=1; j<255; j++)
     {
     point = *image++;
     if ( point == 63 )
        if( point != *image_moins_un )
           *image_contour = 63;
        else if( point != *image_plus_un )
           *image_contour = 63;
        else if( point != *image_moins_256 )
           *image_contour = 63;
        else if( point != *image_plus_256)
           *image_contour = 63;
        else
           *image_contour = 0;
     else
        *image_contour = 0;

     image_plus_un++;
     image_moins_un++;
     image_plus_256++;
     image_moins_256++;
     image_contour++;

     }

  *(image_contour++) = 0;
  image++;
  image_moins_un++;
  image_plus_un++;
  image_moins_256++;
  image_plus_256++;

  }

for(i=0; i<256; i++)
```

```
  *(image_contour++) = 0;

/*  Affichage de l'image contour  */
affiche(image_contour_init);

/*  Storage de l'image contour sur fichier  */
store_image(image_contour_init);

_ffree ( image_init);
_ffree ( image_contour_init);
}
```

```
/* ------------------------------------------------------------------------
    fichier source: EDGE_1.C

    Ce programme commande a la carte digitalisante de digitaliser
    (6 bits) une image en provenance de la camera.

    Le programme calcule ensuite une image gradient (de cette image
    en utilisant la methode numerique suivante:

    GRADx = F(i+1,j) - F(i,j)
    GRADy = F(i,j+1) - F(i,j)

    | GRAD |(i,j) =  SQRT ( GRADx * GRADx   +   GRADy * GRADy )
    ANGLE (i,j)   =  ARCTAN ( GRADy / GRADx )


    L'image gradient est affiche et ensuite conserve dans un fichier.

    ------------------------------------------------------------------------ */

#define    IMAGE_SIZE   (240 * 256)

#include   <stdio.h>
#include   <malloc.h>
#include   <memory.h>
#include   <string.h>
#include   <math.h>
#include   <vision.h>

main()
{

char *image,*image_init;
char *image_plus_un, *image_plus_256;
char *image_gradient, *image_gradient_init;
long i, j;
register int gradient_x, gradient_y;
int zero = 0;


printf("Programme de calcul d'une image gradient\n");


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));

image_gradient_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image  */

lire_image(image_init);


/* Afficher l'image */

affiche(image_init);
```

```
/* Calcul de l'image gradient  */

printf("Calcul de l'image gradient\n");

image = image_init;
image_plus_un = image_init + 1;
image_plus_256 = image_init + 256;
image_gradient = image_gradient_init;


for(i=0; i< 239; i++)
  for(j=0; j<256; j++)
     {
     gradient_x = *image_plus_256 - *image;

     if(j==255)
        gradient_y = 0;
     else
        gradient_y = *image_plus_un - *image;

     *image_gradient = (char) sqrt(
                       (double) (gradient_x*gradient_x+gradient_y*gradient_y) );

     image++;
     image_gradient++;
     image_plus_un++;
     image_plus_256++;

     }

for(i=0; i<256; i++)
   *(image_gradient++) =  (char) zero;


/*  Affichage de l'image gradient  */

affiche(image_gradient_init);


/*  Storage de l'image gradient sur fichier  */

store_image(image_gradient_init);


_ffree ( image_init);

_ffree ( image_gradient_init);

}
```

```
/* ------------------------------------------------------------------
    fichier source: EDGE_2.C

    Ce programme commande a la carte digitalisante de digitaliser
    (6 bits) une image en provenance de la camera.

    Le programme calcule ensuite une image gradient de cette image
    en utilisant la methode numerique suivante:

    GRADx = F(i+1,j) - F(i-1,j)
    GRADy = F(i,j+1) - F(i,j-1)

   | GRAD |(i,j)  =  SQRT ( GRADx * GRADx   +   GRADy * GRADy )
    ANGLE(i,j)    =  ARCTAN ( GRADy / GRADx )


    L'image gradient est affiche et ensuite conserve dans un fichier.

    ------------------------------------------------------------------ */


#define    IMAGE_SIZE   61440     /*  61440  = 240 * 256   */

#include   <stdio.h>
#include   <malloc.h>
#include   <vision.h>
#include   <math.h>

main()
{

char *image,*image_init;
char *image_plus_un, *image_plus_256;
char *image_moins_un, *image_moins_256;
char *image_gradient, *image_gradient_init;
long i, j;
register int gradient_x, gradient_y;
int zero = 0;


printf("Programme de calcul de l'image gradient (double)\n");


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


image_gradient_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image  */

lire_image(image_init);


/* Afficher l'image */

affiche(image_init);
```

```
/* Calcul de l'image gradient  */

printf("Calcul de l'image gradient\n");

image_plus_un = (image_init+256) + 1;
image_moins_un = (image_init+256) - 1;
image_plus_256 = (image_init+256) + 256;
image_moins_256 = image_init ;
image_gradient = image_gradient_init;

/*  On initialise arbitrairement la premiere ligne de l'image gradient a 0  */

for(j=0; j<256; j++)
  *(image_gradient++) = (char) zero;

for(i=1; i< 239; i++)
  for(j=0; j<256; j++)
     {
     gradient_x = *image_plus_256 - *image_moins_256;

     if(j<1 || j==255)
        gradient_y = 0;
     else
        gradient_y = *image_plus_un - *image_moins_un;

     *image_gradient = (char) sqrt(
                       (double) (gradient_x*gradient_x+gradient_y*gradient_y) );

     image_moins_un++;
     image_plus_un++;
     image_gradient++;
     image_plus_256++;
     image_moins_256++;

     }


for(i=0; i< 256; i++)
  *(image_gradient++) =  (char) zero;


/*  Afficher l'image gradient */

affiche(image_gradient_init);


/*  Sauver sur fichier l'image gradient  */

store_image(image_gradient_init);



_ffree ( image_init);

_ffree ( image_gradient_init);

}
```

```
/* ------------------------------------------------------------------
    fichier source: EDGE_3.C

    Ce programme commande a la carte digitalisante de digitaliser
    (6 bits) une image en provenance de la camera et l'affiche a l'ecran.

    Le programme calcule ensuite une image gradient (de cette image
    en utilisant la methode numerique suivante:

    GRADx = F(i+1,j) - F(i,j)
    GRADy = F(i,j+1) - F(i,j)

    | GRAD |(i,j) =  ABS(GRADx) +  ABS(GRADy)
    ANGLE(i,j)     =  ARCTAN ( GRADy / GRADx )

    L'image gradient est affiche et ensuite conserve dans un fichier.

    ------------------------------------------------------------------ */

#define    IMAGE_SIZE   (240 * 256)

#include   <stdio.h>
#include   <malloc.h>
#include   <memory.h>
#include   <string.h>
#include   <math.h>
#include   <vision.h>
#include   <stdlib.h>

main()
{

char *image,*image_init;
char *image_plus_un, *image_plus_256;
char *image_gradient, *image_gradient_init;
long i, j;
register int gradient_x, gradient_y;
int zero = 0;


printf("Programme de calcul d'une image gradient\n");


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));

image_gradient_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image  */

lire_image(image_init);


/* Afficher l'image */

affiche(image_init);
```

```
/* Calcul de l'image gradient  */

printf("Calcul de l'image gradient\n");

image = image_init;
image_plus_un = image_init + 1;
image_plus_256 = image_init + 256;
image_gradient = image_gradient_init;


for(i=0; i< 239; i++)
  for(j=0; j<256; j++)
      {
      gradient_x = *image_plus_256 - *image;

      if(j==255)
         gradient_y = 0;
      else
         gradient_y = *image_plus_un - *image;

      *image_gradient = (char)
                        (abs(gradient_x) + abs(gradient_y));

      image++;
      image_gradient++;
      image_plus_un++;
      image_plus_256++;

      }

for(i=0; i<256; i++)
   *(image_gradient++) =  (char) zero;


/*  Affichage de l'image gradient  */

affiche(image_gradient_init);


/*  Storage de l'image gradient sur fichier  */

store_image(image_gradient_init);


_ffree ( image_init);

_ffree ( image_gradient_init);

}
```

```
/* ------------------------------------------------------------------
    Fichier:  EDGE_4.C

    Ce programme commande a la carte digitalisante de digitaliser
    (6 bits) une image en provenance de la camera et l'affiche a l'ecran.

    Le programme calcule ensuite une image gradient (de cette image
    en utilisant la methode numerique suivante:

    GRADx = F(i+1,j) - F(i-1,j)
    GRADy = F(i,j+1) - F(i,j-1)

    | GRAD |(i,j) =  ABS(GRADx) +  ABS(GRADy)
    ANGLE(i,j)     =  ARCTAN ( GRADy / GRADx )

    L'image gradient est affiche et ensuite conserve dans un fichier.

    ------------------------------------------------------------------ */


#define    IMAGE_SIZE   61440     /*  61440  = 240 * 256   */

#include   <stdio.h>
#include   <malloc.h>
#include   <vision.h>
#include   <math.h>

main()
{

char *image,*image_init;
char *image_plus_un, *image_plus_256;
char *image_moins_un, *image_moins_256;
char *image_gradient, *image_gradient_init;
long i, j;
register int gradient_x, gradient_y;
int zero = 0;


printf("Programme de calcul de l'image gradient (double)\n");


image_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


image_gradient_init =  _fmalloc( IMAGE_SIZE * sizeof(char));


/* Lecture de l'image  */

lire_image(image_init);


/* Afficher l'image */

affiche(image_init);
```

```
/* Calcul de l'image gradient  */


printf("Calcul de l'image gradient\n");

image_plus_un = (image_init+256) + 1;
image_moins_un = (image_init+256) - 1;
image_plus_256 = (image_init+256) + 256;
image_moins_256 = image_init ;
image_gradient = image_gradient_init;

/*  On initialise arbitrairement la premiere ligne de l'image gradient a 0  */

for(j=0; j<256; j++)
  *(image_gradient++) = (char) zero;

for(i=1; i< 239; i++)
  for(j=0; j<256; j++)
     {
     gradient_x = *image_plus_256 - *image_moins_256;

     if(j<1 || j==255)
        gradient_y = 0;
     else
        gradient_y = *image_plus_un - *image_moins_un;

     *image_gradient = (char) ( abs(gradient_x)+abs(gradient_y) );


     image_moins_un++;
     image_plus_un++;
     image_gradient++;
     image_plus_256++;
     image_moins_256++;

     }


for(i=0; i< 256; i++)
  *(image_gradient++) =  (char) zero;


/*  Afficher l'image gradient */

affiche(image_gradient_init);


/*  Sauver sur fichier l'image gradient  */

store_image(image_gradient_init);



_ffree ( image_init);

_ffree ( image_gradient_init);

}
```

```
/*_____________________________________________________________________

    Fichier:  HOUGH1.C

    Ce progamme fait une transformation de Hough circulaire d'une image
    binaire de contours lu dans un fichier et procede ensuite a la localisation
    dans le tableau des accumulateurs des maximums.

    Seulement les cercles de rayons 10 a 20 sont consideres.  Le seuil
    en haut duquel une location du tableau des accumulateurs sera considere
    comme un maximum est variable en fonction du rayon : 21 + (rayon-10)

    Chacune de ces locations correspondant theoriquement a un cercle de
    l'image de coutours ayant un certain rayon et une certaine location
    pour son centre.  Les coordonnees de ces centres et la valeur de ces
    rayon sont affiches a l'ecran.

    Tous le processus de computation de tableau des accumulateurs est
    visualise sur l'ecran TV connecte a la carte digitalisante, parce que
    qu'on a deliberement fait les calcul des accumulateurs dans la memoire
    de la carte digitalisante.

    _____________________________________________________________________*/


#define     IMAGE_SIZE      61440
#define     ADRESSE_CARTE   (char *) 0xA0000000


#include    <stdio.h>
#include    <malloc.h>
#include    <memory.h>
#include    <vision.h>

int max[11][100][3];

main()

{

char *image_init, *image;
unsigned char *carte;
int i,j,k,l,r;
int zero = 0;
int threshold;
register int xr, yr;
void cercle(int, int, int);


image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );


/* Lecture du fichier de contours */

printf("Fichier de contours \n");

lire_image(image_init);
```

```
/*  Afficher le fichier de contours */

affiche(image_init);


/*  Transformation de Hough de l'image afin de trouver les centres et
    rayons des cercles   */

printf("Debut de la transformation de Hough\n");

for(r=0; r<=10; r++)
  {
  printf("r = %d\n", r+10);

  image = image_init + 5120;
  carte = ADRESSE_CARTE;

  /* Effacons le contenu de la carte */

  memset(carte, zero, IMAGE_SIZE * sizeof(char));

  carte += 5120;

  /* Trouvons les cercles de rayon r+10  */

  for(i=20; i<220; i++)
     {
     for(j=20; j<236; j++)
        {
        if( *image == 63 )
           for(k=1; k<=tab[r][0][0]; xr=256*tab[r][k][0], yr= tab[r][k][1], k++)
             {
             (*(carte+yr+xr))++;
             (*(carte+yr-xr))++;
             (*(carte-yr+xr))++;
             (*(carte-yr-xr))++;
             } ;
        image++;
        carte++;
        };

     carte += 40;
     image += 40;
     }

  /* Trouvons les maximums significatifs dans le tableau des accumulateurs */

  threshold =  r + 21;
  carte = ADRESSE_CARTE;
  k = 0;

  for(i=0; i<240; i++)
     for(j=0; j<256; j++)
        {
        if( *carte > threshold )
           {
           max[r][k][0] =  *carte;
           max[r][k][1]   =  i;
           max[r][k][2]   =  j;
           k++;
```

```
            }
         carte++;
         }

   for(l=0; l<k; l++)
      printf("      max = %d    (x = %d    y = %d)\n", max[r][l][0],
            max[r][l][1], max[r][l][2]);

 .}

/* Affichage des cercles sur l'image originale */

printf("Entrez le nom du fichier contenant l'image originale.\n");

lire_image(image_init);


affiche(image_init);

for (r=0; r<=20; r++)
   {
   l = 0;
   while( max[r][l][0] != 0 )
      {
      cercle(r, max[r][l][1], max[r][l][2]);
      l++;
      }
   );


} /* Fin du programme */
```

```
/*_____________________________________________________________________

    Fichier:  HOUGH_2.C

    Ce progamme fait une transformation de Hough circulaire d'une image
    binaire de contours lu dans un fichier et procede ensuite a la localisation
    dans le tableau des accumulateurs des maximums.

    Seulement les cercles de rayons 10 a 20 sont consideres.  Le seuil
    en haut duquel une location du tableau des accumulateurs sera considere
    comme un maximum est variable en fonction du rayon : 21 + (rayon-10)

    Chacune de ces locations correspondant theoriquement a un cercle de
    l'image de coutours ayant un certain rayon et une certaine location
    pour son centre.  Les coordonnees de ces centres et la valeur de ces
    rayon sont affiches a l'ecran.

    Ce programme se distingue du programme Houg_1.C par le fait qu'un
    certaine strategie d'elimination de certain maximums repondant au
    critere ci-haut est employe.  Cette strategie suppose qu'il n'y a
    pas de cercle concentrique de sorte que si plusieurs maximums sont
    trouves et qu'ils ont le meme position pour le centre, seulement le
    maximum est retenu.  Cette strategie aussi suppose que pour des
    objets pas tout a fait concentrique, tout maximum est accompagne de
    quatre maximum parasites. Ces maximums parasites sont elimines.

    Tous le processus de computation de tableau des accumulateurs est
    visualise sur l'ecran TV connecte a la carte digitalisante, parce que
    qu'on a deliberement fait les calcul des accumulateurs dans la memoire
    de la carte digitalisante.

    _____________________________________________________________________*/


#define     IMAGE_SIZE      61440
#define     ADRESSE_CARTE   (char *) 0xA0000000


#include    <stdio.h>
#include    <stdlib.h>
#include    <malloc.h>
#include    <memory.h>
#include    <vision.h>

int max[11][100][3];

main()

{

char *image_init, *image;
unsigned char *carte;
int i,j,k,l,r;
int zero = 0;
int threshold;
register int xr, yr;
void maximums(int);
```

```
image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );

/* Lecture du fichier de contours */

printf("Fichier de contours \n");

lire_image(image_init);

/*  Afficher le fichier de contours */

affiche(image_init);

/*  Transformation de Hough de l'image afin de trouver les centres et
    rayons des cercles   */

printf("Debut de la transformation de Hough\n");

for(r=0; r<=10; r++)
  {
  printf("r = %d\n", r+10);

  image = image_init + 5120;
  carte = ADRESSE_CARTE;

  /* Effacons le contenu de la carte */

  memset(carte, zero, IMAGE_SIZE * sizeof(char));

  carte += 5120;

  /* Trouvons les cercles de rayon r+10  */

  for(i=20; i<220; i++)
     {
     for(j=20; j<236; j++)
        {
        if( *image == 63 )
           for(k=1; k<=tab[r][0][0]; xr=256*tab[r][k][0], yr= tab[r][k][1], k++)
             {
             (*(carte+yr+xr))++;
             (*(carte+yr-xr))++;
             (*(carte-yr+xr))++;
             (*(carte-yr-xr))++;
             } ;
        image++;
        carte++;
        );

     carte += 40;
     image += 40;
     }

  maximums(r);

  for(l=0; max[r][l][0] > 0; l++)
     printf("      max = %d    (x = %d    y = %d)\n", max[r][l][0],
```

```
            max[r][l][1], max[r][l][2]);

  }
/* Elimination des cercles de rayon voisin et de meme centre  */


/* Affichage des cercles sur l'image originale */

printf("Entrez le nom du fichier contenant l'image originale.\n");

lire_image(image_init);

affiche(image_init);

for (r=0; r<=20; r++)
  {
  l = 0;
  while( max[r][l][0] != 0 )
     {
     cercle(r, max[r][l][1], max[r][l][2]);
     l++;
     }
  };


} /* Fin du programme */


/* ------------------------------------------------------------------*/

/* Trouvons les maximums significatifs */


int centre[50][3];
int nombre_max;
int nombre_max_abs;



void maximums(r)

int r;

{

unsigned char *carte;
register int threshold;
register int i,j;
void trouve_centre(int);

threshold = 3 * (r+10);
carte = ADRESSE_CARTE;
nombre_max = 0;

for(i=0; i< 240; i++)
  for(j=0; j<256; j++)
      {
```

```
      if( *carte > threshold )
         {
         max[r][nombre_max][0]    =  *carte;
         max[r][nombre_max][1]    =  i;
         max[r][nombre_max][2]    =  j;
         nombre_max++;
         }
         carte++;
      }

/* Elimination des 4 maximums relatifs autour de chaque maximum absolue */

nombre_max_abs = 0;

for (i=0; i<10; i++)
  {
  centre[i][0] = 0;
  centre[i][1] = 0;
  centre[i][2] = 0;
  };

trouve_centre(r);

for(i=0; i<nombre_max_abs; i++)
  {
  max[r][i][0] = centre[i][0];
  max[r][i][1] = centre[i][1];
  max[r][i][2] = centre[i][2];
  };

}

/*-------------------------------------------------------------------------*/

void trouve_centre(r)

int r;

{

int max_max;
int i;
int i_du_max;
int x,y;


for(i=0, max_max=0; i<nombre_max; i++)
  if( max_max < max[r][i][0])
     {
     max_max = max[r][i][0];
     i_du_max = i;
     };


if ( max_max == 0 )  return;

x = max[r][i_du_max][1];
y = max[r][i_du_max][2];
```

```
centre[nombre_max_abs][0] = max_max;
centre[nombre_max_abs][1] = x;
centre[nombre_max_abs][2]  = y;

for(i=0; i<nombre_max; i++)
  if(abs(max[r][i][1]-x) + abs(max[r][i][2]-y) <= 2)
        max[r][i][0] = 0;


nombre_max_abs++;
trouve_centre(r);


}
```

```
/*_____________________________________________________________________

    Fichier:  HOUGH_3.C

    Ce programme comporte que des modifications mineures au programme HOUGH_2.

    _____________________________________________________________________*/

#define    IMAGE_SIZE     61440
#define    ADRESSE_CARTE  (char *) 0xA0000000


#include   <stdio.h>
#include   <stdlib.h>
#include   <malloc.h>
#include   <memory.h>
#include   <vision.h>

int max[11][100][3];

main()

{

struct rond
  {
  int max;
  int r;
  int x;
  int y;
  };

struct rond tomate[50];
char *image_init, *image;
unsigned char *carte;
int i,j,k,l,r;
int zero = 0;
int threshold;
register int xr, yr;
int maximums(int);


image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );


/* Lecture du fichier de contours */

printf("Fichier de contours \n");

lire_image(image_init);


/*  Afficher le fichier de contours */

affiche(image_init);


/*  Transformation de Hough de l'image afin de trouver les centres et
    rayons des cercles   */
```

```
printf("Debut de la transformation de Hough\n");

for(r=0; r<=10; r++)
  {
  printf("r = %d\n", r+10);

  image = image_init + 5120;
  carte = ADRESSE_CARTE;

  /* Effacons le contenu de la carte */

  memset(carte, zero, IMAGE_SIZE * sizeof(char));

  carte += 5120;

  /* Trouvons les cercles de rayon r+10  */

  for(i=20; i<220; i++)
     {
     for(j=20; j<236; j++)
        {
        if( *image == 63 )
           for(k=1; k<=tab[r][0][0]; xr=256*tab[r][k][0], yr= tab[r][k][1], k++)
             {
             (*(carte+yr+xr))++;
             (*(carte+yr-xr))++;
             (*(carte-yr+xr))++;
             (*(carte-yr-xr))++;
             } ;
        image++;
        carte++;
        };

     carte += 40;
     image += 40;
     }

  max[r][99][0] = maximums(r);


  };

/* Elimination des cercles de rayon voisin et de meme centre  */

for(r=0;r<10;r++)
  for(i=0; i< max[r][99][0]; i++)
     for(j=0; j< max[r+1][99][0]; j++)
        if(abs(max[r+1][j][1]-max[r][i][1])+abs(max[r+1][j][2]-max[r][i][2])<=2)
           if( max[r][i][0] < max[r+1][j][0] )
               max[r][i][0] = 0;
           else
               max[r+1][j][0] = 0;

k = 0;

for(r=0; r<11; r++)
  for(i=0; i< max[r][99][0]; i++)
     if ( max[r][i][0] != 0 )
        {
```

```
        tomate[k].max = max[r][i][0];
        tomate[k].r   = r+10;
        tomate[k].x   = max[r][i][1];
        tomate[k].y   = max[r][i][2];
        k++;
        };

for(i=0; i<k; i++)
  printf("tomate : %d  --> max = %d  rayon = %d   ( x = %d  ,  y = %d ) \n",
  i, tomate[i].max, tomate[i].r, tomate[i].x, tomate[i].y);


/* Affichage des cercles sur l'image originale */

printf("Entrez le nom du fichier contenant l'image originale.\n");

lire_image(image_init);

affiche(image_init);

for (i=0; i<k; i++)
     cercle(tomate[i].r-10,tomate[i].x,tomate[i].y);


} /* Fin du programme */


/* ------------------------------------------------------------------------*/

/* Trouvons les maximums significatifs */


int centre[50][3];
int nombre_max;
int nombre_max_abs;



int maximums(r)

int r;

{

unsigned char *carte;
register int threshold;
register int i,j;
void trouve_centre(int);

threshold = 3 * (r+10);
carte = ADRESSE_CARTE;
nombre_max = 0;

for(i=0; i< 240; i++)
  for(j=0; j<256; j++)
      {
      if( *carte > threshold )
         {
         max[r][nombre_max][0]   =  *carte;
         max[r][nombre_max][1]   =  i;
```

```
            max[r][nombre_max][2]    =  j;
            nombre_max++;
            }
            carte++;
        }

/* Elimination des 4 maximums relatifs autour de chaque maximum absolue */

nombre_max_abs = 0;

for (i=0; i<10; i++)
  {
  centre[i][0] = 0;
  centre[i][1] = 0;
  centre[i][2] = 0;
  };

trouve_centre(r);

for(i=0; i<nombre_max_abs; i++)
  {
  max[r][i][0] = centre[i][0];
  max[r][i][1] = centre[i][1];
  max[r][i][2] = centre[i][2];
  };

return(nombre_max_abs);

}

/*-------------------------------------------------------------------------*/

void trouve_centre(r)

int r;

{

int max_max;
int i;
int i_du_max;
int x,y;


for(i=0, max_max=0; i<nombre_max; i++)
  if( max_max < max[r][i][0])
     {
     max_max = max[r][i][0];
     i_du_max = i;
     };


if ( max_max == 0 )  return;

x = max[r][i_du_max][1];
y = max[r][i_du_max][2];

centre[nombre_max_abs][0] = max_max;
centre[nombre_max_abs][1] = x;
```

```
centre[nombre_max_abs][2]  = y;

for(i=0; i<nombre_max; i++)
  if(abs(max[r][i][1]-x) + abs(max[r][i][2]-y) <= 2)
        max[r][i][0] = 0;


nombre_max_abs++;
trouve_centre(r);


}
```

```
/*_____________________________________________________________________

    Fichier:  HOUGH_4.C

    Ce programme comporte que des modifications mineures au programme HOUGH_3.

    _____________________________________________________________________*/

#define    IMAGE_SIZE     61440
#define    ADRESSE_CARTE  (char *) 0xA0000000


#include   <stdio.h>
#include   <stdlib.h>
#include   <malloc.h>
#include   <memory.h>
#include   <vision.h>

int max[11][100][3];

main()

{

struct rond
  {
  int max;
  int r;
  int x;
  int y;
  };

struct rond tomate[50];
char *image_init, *image;
unsigned char *carte;
int i,j,k,l,r;
int zero = 0;
int threshold;
register int xr, yr;
int maximums(int);


image_init = _fmalloc( IMAGE_SIZE * sizeof(char) );

/* Lecture du fichier de contours */

printf("Fichier de contours \n");

lire_image(image_init);

/*  Afficher le fichier de contours */

affiche(image_init);

/*  Transformation de Hough de l'image afin de trouver les centres et
```

```
    rayons des cercles   */

printf("Debut de la transformation de Hough\n");

for(r=0; r<=10; r++)
  {
  printf("r = %d\n", r+10);

  image = image_init + 5120;
  carte = ADRESSE_CARTE;

  /* Effacons le contenu de la carte */

  memset(carte, zero, IMAGE_SIZE * sizeof(char));

  carte += 5120;

  /* Trouvons les cercles de rayon r+10  */

  for(i=20; i<220; i++)
     {
     for(j=20; j<236; j++)
        {
        if( *image == 63 )
           for(k=1; k<=tab[r][0][0]; xr=256*tab[r][k][0], yr= tab[r][k][1], k++)
             {
             (*(carte+yr+xr))++;
             (*(carte+yr-xr))++;
             (*(carte-yr+xr))++;
             (*(carte-yr-xr))++;
             } ;
        image++;
        carte++;
        };

     carte += 40;
     image += 40;
     }

  max[r][99][0] = maximums(r);

  };

/* Elimination des cercles de rayon voisin et de meme centre  */

for(r=0;r<10;r++)
  for(i=0; i< max[r][99][0]; i++)
     for(j=0; j< max[r+1][99][0]; j++)
        if(abs(max[r+1][j][1]-max[r][i][1])+abs(max[r+1][j][2]-max[r][i][2])<=2)
           if( max[r][i][0] < max[r+1][j][0] )
               max[r][i][0] = 0;
           else
               max[r+1][j][0] = 0;

k = 0;

for(r=0; r<11; r++)
  for(i=0; i< max[r][99][0]; i++)
     if ( max[r][i][0] != 0 )
```

```
        (
        tomate[k].max = max[r][i][0];
        tomate[k].r   = r+10;
        tomate[k].x   = max[r][i][1];
        tomate[k].y   = max[r][i][2];
        k++;
        );

for(i=0; i<k; i++)
  printf("tomate : %d  --> max = %d  rayon = %d   ( x = %d  ,  y = %d ) \n",
  i, tomate[i].max, tomate[i].r, tomate[i].x, tomate[i].y);


/* Affichage des cercles sur l'image originale */

printf("Entrez le nom du fichier contenant l'image originale.\n");

lire_image(image_init);

affiche(image_init);

for (i=0; i<k; i++)
     cercle(tomate[i].r-10,tomate[i].x,tomate[i].y);


} /* Fin du programme */


/* ------------------------------------------------------------------*/

/* Trouvons les maximums significatifs */


int centre[50][3];
int nombre_max;
int nombre_max_abs;



int maximums(r)

int r;

{

unsigned char *carte;
register int threshold;
register int i,j;
void trouve_centre(int);

threshold = 2 * (r+10);
carte = ADRESSE_CARTE;
nombre_max = 0;

for(i=0; i< 240; i++)
  for(j=0; j<256; j++)
      {
      if( *carte > threshold )
         {
         max[r][nombre_max][0]   =  *carte;
```

```
            max[r][nombre_max][1]    =  i;
            max[r][nombre_max][2]    =  j;
            nombre_max++;
            }
            carte++;
        }

/* Elimination des 4 maximums relatifs autour de chaque maximum absolue */

nombre_max_abs = 0;

for (i=0; i<10; i++)
  {
  centre[i][0] = 0;
  centre[i][1] = 0;
  centre[i][2] = 0;
  };

trouve_centre(r);

for(i=0; i<nombre_max_abs; i++)
  {
  max[r][i][0] = centre[i][0];
  max[r][i][1] = centre[i][1];
  max[r][i][2] = centre[i][2];
  };

return(nombre_max_abs);

}

/*-------------------------------------------------------------------------*/

void trouve_centre(r)

int r;

{

int max_max;
int i;
int i_du_max;
int x,y;


for(i=0, max_max=0; i<nombre_max; i++)
  if( max_max < max[r][i][0])
     {
     max_max = max[r][i][0];
     i_du_max = i;
     };


if ( max_max == 0 )  return;

x = max[r][i_du_max][1];
y = max[r][i_du_max][2];

centre[nombre_max_abs][0] = max_max;
```

```
centre[nombre_max_abs][1] = x;
centre[nombre_max_abs][2]  = y;

for(i=0; i<nombre_max; i++)
  if(abs(max[r][i][1]-x) + abs(max[r][i][2]-y) <= 2)
        max[r][i][0] = 0;


nombre_max_abs++;
trouve_centre(r);


}
```

}

```
/*-----------------------------------------------------------------*/
/* Fichier: VISION_1.C                                             */
/*                                                                 */
/* utilitaire qui permet une communication entre la carte          */
/* digitalisante et le SKY320 PC via un programme en C             */
/* modele large, et qui contient les principales fonctions de      */
/* la carte digitalisante                                          */
/*                                                                 */
/* signification des codes d'erreur                                */
/* retournes par les fonctions:                                    */
/* 0=pret a recevoir des commandes                                 */
/*   mais pas des donnees                                          */
/* 1=DT-2803 pas pret a recevoir des commandes                     */
/*   mais pret pour la transmission de donnees                     */
/* 2=demande d'ecriture lorsque DATA IN est plein                  */
/* 3=demande de lecture alors que DATA OUT n'est pas pret          */
/* 4=erreur dans les arguments passes                              */
/*                                                                 */
/*-----------------------------------------------------------------*/

#include "stdio.h"              /*APPEL DES FONCTIONS STANDARD    */
                                /*DU LANGAGE C                    */

/*-----------------------------------------------------------------*/
/* DEFINITION DES CODES D'OPERATIONS ET DES MOTS CLEFS POUR LA     */
/*                  CARTE DIGITALISANTE                            */
/*-----------------------------------------------------------------*/

#define DATAREG       736       /*ADRESSE DU REGISTRE DE DONNEE   */
#define CSREG         737       /*ADRESSE DU REGISTRE DE CONTROLE*/
                                /*ET D'ETAT                       */
#define SEG_GRAB      0xA000    /*ADRESSE DE RESIDENCE DE L'IMAGE*/
                                /*DANS LA CARTE DIGITALISANTE     */
#define DISP_ON       0x20      /*CODE DE LA FONCTION D'AFFICHAGE*/
                                /*DE L'IMAGE                      */
#define SYNC_EXT      0x13      /*CODE DE LA FONCTION DE          */
                                /*SYNCHRONISATION EXTERNE         */
#define SYNC_INT      0x12      /*CODE DE LA FONCTION DE          */
                                /*SYNCHRONISATION INTERNE         */
#define PASS_THR      0x23      /*CODE DE LA FONCTION D'AFFICHAGE*/
                                /*EN CONTINUE DE L'IMAGE          */
#define ACQ_FR        0x22      /*CODE DE LA FONCTION QUI PREND   */
                                /*UNE IMAGE ET LA PLACE DANS UN   */
                                /*BUFFER                          */
#define SEND_FR       0x24      /*CODE DE LA FONCTION POUR        */
                                /*ENVOYER UNE IMAGE SUR LE PORT   */
                                /*EXTERNE                         */
#define CATCH_FR      0x25      /*CODE DE LA FONCTION POUR        */
                                /*RECEVOIR UNE IMAGE DU PORT      */
                                /*EXTERNE                         */
#define S_OUT_LT      0x1D      /*CODE DE LA FONCTION DE          */
                                /*SELECTION DE LA PALETTE DE      */
                                /*SORTIE                          */
#define PL_CURS       0x15      /*ECRITURE DE LA POSITION DU      */
                                /*CURSEUR DANS LE REGISTRE DE     */
                                /*DONNEE                          */
#define CURS_OFF      0x17      /*DESACTIVE LA FONCTION CURSEUR   */
#define CURS_ON       0x16      /*ACTIVE LA FONCTION CURSEUR      */
#define TABLE 0                 /*CHOIX DE LA PALETTE D'INTENSITE*/
                                /*0 POUR LA SORTIE                */
```

```
static char status;                    /*LA VARIABLE STATUS EST DEFINIE */
                                       /*EN STATIC CHARACTER            */

/*-------------------------------------------------------------------*/
/* FONCTION QUI VERIFIE L'ETAT DU REGISTRE DE CONTROLE ET D'ETAT */
/*-------------------------------------------------------------------*/
int check()
{
status=inp(CSREG);                     /*LECTURE DE L'ETAT DU SYSTEME   */
if(!(~(status|0177)))                  /*VERIFIE SI LE BIT ERREUR EST   */
     erreur();                         /*ACTIVE SI OUI EXECUTE ERREUR   */
if(!(status&0004))                     /*VERIFIE SI LE BIT READY EST    */
     return(1);                        /*ACTIVE SI NON ACTIVE IMPRIME   */
                                       /*UN MESSAGE INDIQUANT QUE LA    */
                                       /*CARTE DIGITALISANTE N'EST PAS  */
                                       /*PRETE                          */
return(0);
}
/*-------------------------------------------------------------------*/
/*         FONCTION QUI LIT LE REGISTRE D'ERREUR                     */
/*-------------------------------------------------------------------*/
void erreur()
{
char read;                             /*READ, ERR1 ET ERR2 SONT        */
char err1,err2;                        /*DEFINIES COMME DES CHARACTERS  */
read=inp(DATAREG);                     /*LECTURE DU REGISTRE DE DONNEE  */
outp(CSREG,0004);                      /*ACTIVER LE BIT READY DU CSREG  */
err1=inp(DATAREG);                     /*LECTURE DU REGISTRE D'ERREUR   */
err2=inp(DATAREG);
printf("une erreur importante s'est produite,dt2803 p5-14\n");
printf("registre d'erreur: octet1=%o,octet2=%o/n",err1,err2);
exit(0);
}
/*-------------------------------------------------------------------*/
/*    FONCTION D'ECRITURE DES CODES D'OPERATIONS DU DT-2803           */
/*-------------------------------------------------------------------*/
int writec(com)
char com;                              /*VAR COM DEFINIE EN CHARACTER   */
{
unsigned i;                            /*DECLARE I EN ENTIER NON-SIGNE  */
/* BOUCLE QUI ATTEND QUE LE BIT READY DU CSREG SOIT EGALE A 1    */
for(i=0;(check()!=0)&&(i<0xFFFF);i++);
if (check())
        {
        printf("grabber pas pret!");
        return(1);
        }
/* VERIFICATION QUE LE BIT DATA IN FULL DU CSREG EST 0 (VIDE)    */
if (!(~(status|0375)))
        return(2);
outp(CSREG,com);                       /*ECRITURE DU CODE DE COMMANDE   */
return(0);
}

/*-------------------------------------------------------------------*/
/*   FONCTION QUI ECRIT UNE DONNEE DANS LE REGISTRE DE DONNEE        */
/*-------------------------------------------------------------------*/
int writed(data)
char data;                             /*DATA DEFINIE EN CHARACTER      */
```

```
{
unsigned i;                         /*I DEFINIE EN ENTIER NON-SIGNE  */
/* BOUCLE QUI ATTEND QUE LE BIT READY DU CSREG SOIT ZERO          */
for(i=0;(check()!=1)&&(i<0xFFFF);i++);
if (check()!=1)
        {
        printf("le grabber n'est pas en etat de recevoir data!\n");
        return(0);
        }
if (status&0002)                    /*VERIFIE SI LE BIT DATA IN FULL */
                                    /*EST NUL                        */
        {
/* BOUCLE QUI ATTEND QUE LE DATA IN FULL SOIT ACTIVE              */
        for (i=0;(status&0002)&&(i<30000);i++,check());
        if (i>=30000)
                return(2);
        }
outp(DATAREG,data);                 /*ECRIRE LA DONNEE DANS LE REGIS-*/
return(1);                          /*TRE DE DONNEE                  */
}

/*-------------------------------------------------------------------*/
/*   FONCTION QUI INITIALISE L'ETAT DE LA CARTE DIGITALISANTE        */
/*            C'EST LA PREMIERE FONCTION A APPELER                   */
/*-------------------------------------------------------------------*/
void set()
{
char read;                          /*READ DEFINIE EN CHARACTER      */
read=inp(DATAREG);                  /*LECTURE DU REGISTRE DE DONNEE  */
                                    /*POUR EN VIDER LE CONTENUE      */
check();                            /*VERIFICATION DU CSREG          */
}

/*-------------------------------------------------------------------*/
/*    FONCTION D'AFFICHAGE DE L'IMAGE SUR LE MONITEUR VIDEO          */
/*-------------------------------------------------------------------*/
int pass()
{
int erreur;                         /*ERREUR ET I SONT DEFINIES EN   */
int i;                              /*ENTIER                         */
if (erreur=writec(SYNC_EXT))        /*PERMET LA SYNCHRONISATION EXTE */
        return(erreur);
if (erreur=writec(DISP_ON))         /*PERMET L'AFFICHAGE SUR MONITEUR*/
        return(erreur);
return(check());
}



/*-------------------------------------------------------------------*/
/* FONCTION D'ACQUISITION D'UNE IMAGE DANS LA MEMOIRE DE LA CARTE*/
/*                      DIGITALISANTE                                */
/*-------------------------------------------------------------------*/
int acq()
{
int erreur;                         /*ERREUR DEFINIE EN ENTIER       */
if (erreur=writec(SYNC_EXT))        /*SYNCHRONISATION EXTERNE        */
        return(erreur);
if (erreur=writec(ACQ_FR))          /*PERMET L'ACQUISITION D'IMAGE   */
```

```
        return(erreur);
return(check());
}

/*-------------------------------------------------------------*/
/* FONCTION POUR ENVOYER LE CONTENU DE LA MEMOIRE DE LA CARTE  */
/*            DIGITALISANTE SUR LE PORT EXTERNE                */
/*-------------------------------------------------------------*/
int send()
{
int erreur;                     /*ERREUR DEFINIE EN ENTIER      */
if (erreur=writec(SYNC_INT))    /*SYNCHRONISATION INTERNE       */
        return(erreur);
if (erreur=writec(SEND_FR))     /*PERMET D'ENVOYER LE CONTENU DE */
        return(erreur);         /*LA MEMOIRE DE LA CARTE        */
return(check());                /*SUR LE PORT EXTERNE           */
}

/*-------------------------------------------------------------*/
/* FONCTION POUR RECEVOIR UNE IMAGE TRAITEE VIA LE PORT EXTERNE */
/*-------------------------------------------------------------*/
int catch()
{
int erreur;                     /*ERREUR DEFINIE EN ENTIER      */
if (erreur=writec(SYNC_INT))    /*SYNCHRONISATION INTERNE       */
        return(erreur);
if (erreur=writec(CATCH_FR))    /*PERMET DE RECEVOIR UNE IMAGE  */
        return(erreur);         /*EN PROVENANCE DU PORT EXTERNE */
return(check());
}

/*-------------------------------------------------------------*/
/* FONCTION QUI PREPARE LE SKY A RECEVOIR UNE IMAGE ET QUI     */
/* TRANSMET UNE IMAGE DE LA CARTE DIGITALISANTE VERS LE SKY    */
/*-------------------------------------------------------------*/
void sendgs()
{
int arg[20];                    /*DECLARE UN VECTEUR ARG        */
arg[0]=0;                       /*NOMBRE D'ARGUMENT A PASSER    */
                                /*A LA MEMOIRE PROGRAMME DU SKY */
prload("comib.320",arg);        /*CHARGE LE PROGRAMME DE        */
                                /*TRANSFERT DANS LE SKY ET MET  */
                                /*EN MARCHE LE TMS32010         */
send();                         /*EXECUTE LA FONCTION SEND      */
}

/*-------------------------------------------------------------*/
/* FONCTION QUI PREPARE LE SKY A TRANSMETTRE UNE IMAGE ET QUI  */
/* TRANSMET UNE IMAGE DU SKY VERS LA CARTE DIGITALISANTE       */
/*-------------------------------------------------------------*/
void sendsg()
{
int arg[20];                    /*DECLARE LE VECTEUR ARG        */
arg[0]=0;                       /*NOMBRE D'ARGUMENT A PASSER A LA*/
                                /*MEMOIRE PROGRAMME DU SKY      */
prload("comob.320",arg);        /*CHARGE LE PROGRAMME DE        */
                                /*TRANSFERT COMOB.320 DANS LE SKY*/
                                /*ET MET EN MARCHE LE TMS32010  */
catch();                        /*EXECUTE LA FONCTION CATCH     */
}
```

```
/*-------------------------------------------------------------*/
/* FONCTION QUI PERMET DE SELECTIONNER UNE DES 4 OUTPUTS LOOK UP */
/*                          TABLE                              */
/*-------------------------------------------------------------*/
int outs(table)
int table;                      /*TABLE DEFINIE EN ENTIER       */
{
int erreur;                     /*ERREUR DEFINIE EN ENTIER      */
char tablebyte;                 /*TABLEBYTE DEFINIE EN CHARACTER */
tablebyte=table;
if (erreur=writec(S_OUT_LT))    /*SELECTION DE LA OUTPUT L.U.T  */
        return(erreur);
if ((erreur=writed(tablebyte))!=1)/*ECRITURE DE LA L.U.T CHOISIE */
        return(erreur);         /*DANS LE REGISTRE DE DONNEE    */
return(check());
}
```

```
/*_____________________________________________________________________

    Fichier:  VISION_2.C

    Ce fichier contient le code source des routines suivantes:
    - lire_image
    - store_image
    - affiche

    _____________________________________________________________________*/

#define    IMAGE_SIZE   61440    /*    61440 = 240 * 256      */
#define    ADRESSE_CARTE  (char *) 0xA0000000

#include   <stdio.h>
#include   <process.h>
#include   <vision.h>
#include   <memory.h>
#include   <string.h>


/*  Fonction qui lit une image dans un fichier  */


void lire_image( pointeur_image)

char *pointeur_image;

{

char *image;
char nom_fichier[80];
int c;
long i;
FILE *file_stream;


image = pointeur_image;

printf("Entrez le nom du fichier image: ");
scanf("%s", nom_fichier);
fflush(stdin);

if( (file_stream=fopen(nom_fichier, "rb")) == NULL)
  {
  printf("Impossible d'ouvrir le fichier %s\n",nom_fichier);
  exit();
  }

for(i=0; i<IMAGE_SIZE; i++)
  {
  if( (c=getc(file_stream))==EOF )
     {
     if(ferror(file_stream))
        printf("Une erreur est survenue lors de la lecture du fichier %s\n",
        nom_fichier);
     else
        printf("Une fin de fichier a ete rencontre\n");
```

```
      exit();
      }

  *(image++) = (char) c;
  }

fclose(file_stream);

}

/* ------------------------------------------------------------------------ */

/* routine qui store une image dans un fichier */

void store_image(pointeur_image)

char *pointeur_image;

{

FILE *file_stream, *file_stream_bidon;
char *pointeur ;
char nom_fichier[80];
int c;
long i, j;


printf("Entrez le nom du fichier ou l'image traitee sera conservee: %s",
        nom_fichier);
scanf("%s", nom_fichier);
fflush(stdin);


if( (file_stream_bidon=fopen("/vtutor/bidon.img","rb"))==NULL )
  {
  printf("Impossible d'ouvrir le fichier /vtutor/bidon.img \n");
  exit();
  }

if( (file_stream=fopen(nom_fichier,"wb"))==NULL )
  {
  printf("Imposssible d'ouvrir le fichier %s\n", nom_fichier);
  exit();
  }


pointeur = pointeur_image;

for(i=0; i< IMAGE_SIZE; i++)
  putc( (int)*(pointeur++), file_stream);


while( (c=getc(file_stream_bidon)) != EOF )
    putc(c, file_stream);


fclose(file_stream);
fclose(file_stream_bidon);
```

```
}
/* ------------------------------------------------------------------------ */

/*    Fonction qui affiche une image      */

void affiche(pointeur_image)

char *pointeur_image;

{

char *carte;

set();

carte =  (char *) 0xA0000000;

memcpy(carte,pointeur_image, IMAGE_SIZE);

pass();

}

/* ---------------------------------------------------------------- */

/* ------------------------------------------------------------------------

    Fonction qui trace un cercle dans la memoire du grabber

    On doit inclure le fichier  tab.h et le fichier vision.h   */

void cercle(r,x,y)

int r,x,y;

{

int k;
register int xr, yr;
char *carte;
extern char tab[11][38][2];

carte = ADRESSE_CARTE;
carte = carte + 256*x + y;

for(k=1; k<=tab[r][0][0]; k++)
  {
  xr=256*tab[r][k][0];
  yr= tab[r][k][1];

  *(carte+yr+xr) = 63;
  *(carte+yr-xr) = 63;
  *(carte-yr+xr) = 63;
  *(carte-yr-xr) = 63;
  } ;
```

3

```
/*_____________________________________________________________________

    Fichier:  TCERCLE.C

    Programme servant a tracer des cercles dans la memoire du "grabber"

    _____________________________________________________________________*/

#define    ADRESSE_CARTE  (char *) 0xA0000000


#include   <stdio.h>
#include   <malloc.h>
#include   <conio.h>
#include   <memory.h>
#include   <vision.h>
#include   <tab.h>


main()

{

int r,x,y;
int c;


printf("Programme d'impression de cercles\n");

set();

boucle:

  printf("Voulez-vous terminer le programme? ( o ou n ) : ");
  if((c=getche()) == 'o') goto fin;
  printf("\n");
  printf("\nr = ");
  scanf("%d", &r);
  printf("    x = ");
  scanf("%d", &x);
  printf("    y = ");
  scanf("%d", &y);
  printf("\n");

  cercle(r,x,y);

  goto boucle;

fin:;

}
```

```
            printf("Erreur dans algorithme de segmentation\n");
        }

/* Assignation a chaque couleur de sa classe d'equivalence */

for(i=0; i<color; i++)
  classe[i] = i;

for(i=color; i < CLASS_SIZE; i++)
  classe[i] = 0;

for(i=1; i < color; i++)
  {
  color_minimum = classe[i];
  for(j=1; j< re_cnt; j++)
     if(relation[j][0]==i && relation[j][1]<color_minimum)
        color_minimum = relation[j][1];
  classe[color] = classe[ classe[color_minimum] ];
  }

/* Numerotation des classes   */

for(classe_no=1,i=0; i<color; i++)
  if(classe[i]==i)
     classe[i] = classe_no++;
  else
     classe[i] = classe[ classe[i] ];

/*  Assignation a chaque point de la sous_image de son numero de classe
    ou d'objet */

    for(row=0; row< ROW; row++)
     for(column=0; column< COLUMN; column++)
        image_color[row][column] = classe[ image_color[row][column] ];

/* Determination du rectangle delimittant chaque objet de l'image */




printf("\nVoulez-vous essayer un autre offset?(Y ou N) : ");

} while( (reponse=getchar())=='Y'|| reponse == 'y' );

_ffree ( image_init);
_ffree ( image_f_init);
_ffree ( image_rouge_init);

}
```

```
/*  ____________________________________________________________________

    Fichier:  EFFACE.C

   Petit programme utilitaire servant a initialiser a zero la memoire du
   "grabber"

    ____________________________________________________________________*/

#define    IMAGE_SIZE     61440
#define    ADRESSE_CARTE  (char *) 0xA0000000

#include   <memory.h>

main()

{
char *carte;
int zero = 0;

  carte = ADRESSE_CARTE;
  /* Effacons le contenu de la carte */

  memset(carte, zero, IMAGE_SIZE * sizeof(char));
}
```

```
/*  ____________________________________________________________________

    Fichier:  C_BIDON.C

   Ce programme sert a creer un fichier de donnees "BIDON.IMG"  qui est utilise
   par la fonction "store_image" de la librairie "LVISION2.LIB".

   Ce programme lit le fichier tg6.img dans le repertoire \vtutor mais tout
   autre fichier image aurait pu faire l'affaire.

   Le fichier cree "bidon.img" est place dans le repertoire \vtutor.    */

   ____________________________________________________________________*/

#define    IMAGE_SIZE   (240 * 256)

#include   <stdio.h>
#include   <process.h>

main()
{

FILE *file_stream, *file_stream_bidon;
long i, j;
int c;


printf("Creation du fichier BIDON.IMG\n");


/* Lecture de l'image  */


if( (file_stream = fopen("/vtutor/tg6.img","rb")) == NULL)
   {
   printf("Impossible d'ouvrir le fichier /vtutor/tg6.img\n");
   exit();
   }

if( (file_stream_bidon=fopen("/vtutor/bidon.img","wb")) == NULL)
  {
  printf("Impossible d'ouvrir le fichier /vtutor/bidon.img\n");
  exit();
  }


for(i=0; i< IMAGE_SIZE; i++)
  c = getc(file_stream);



while( (c=getc(file_stream)) != EOF )
    putc(c, file_stream_bidon);


fclose(file_stream);
fclose(file_stream_bidon);
}
```