Mémoire présenté à
L'Univerité du Québec à Trois-Rivières

Comme exigence partielle
De la maîtrise en mathématiques et informatique appliquées

Par
Evans Girard

Détection automatique de semences de résineux pour l'évaluation en temps-réel de l'efficacité d'un semoir

Juin 2015

Remerciements

J'aimerais d'abord remercier mon directeur de recherche, le professeur François Meunier, de m'avoir laisser l'opportunité de compléter ma maîtrise sous sa direction. J'ai pu découvrir une personne intéressante, flexible et, surtout, une personne qui prend la peine d'écouter et d'analyser avec constructivisme tout commentaire. Je ne saurais remercier, également, mon co-directeur de recherche, le professeur Mhamed Mesfioui, qui est lui aussi une personne de très grande qualité. J'ai essayé de tirer profits de son savoir et de tous ces judicieux conseils. Je remercie aussi Conrad Drolet et tous les travailleurs de la pépinière de Berthier. Un remerciement s'impose aussi à l'organisme Mitacs pour leur bourse. Ces deux années passées à la maîtrise furent synonymes de changements majeurs et d'obstacles qui furent facilement traversées grâce à mes amis que je tiens également à remercier. Ceux-ci savent qui ils sont. Finalement, je suis très reconnaissant envers l'Université du Québec à Trois-Rivières et plus précisément au département de Mathématiques et informatique, entre autres, pour leurs bourses d'études, mais aussi pour la qualité des professeurs et du personnel. Je n'y ait rencontré que des gens bien.

Sommaire

La vision par ordinateur est un domaine en pleine effervescence qui est utilisée dans une variété de disciplines. Par exemple, le multimédia, l'industrie automobile, la sécurité sous plusieurs formes ainsi que l'agriculture pour ne nommer que ces derniers. Ce mémoire à pour mission, d'une part, d'explorer de quelle façon il est possible d'utiliser cette branche de l'informatique dans certains procédés liés à la foresterie. Plus précisément, aider à la détection d'erreurs lors de l'ensemencement de graines de résineux à l'aide d'un planteur pneumatique déjà existant d'une pépinière afin de reboiser certaines régions du Québec. D'autre part, certaines techniques probabilistes et statistiques seront utilisées dans le cadre du contrôle statistiques des procédés. Tout cela en vue d'optimiser la qualité de la production.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

Bonne semence fait bon grain,
et bons arbres portent bon fruit.

proverbe populaire

Ce chapitre à pour but de situer le lecteur tant sur le contexte que sur la problématique rencontrée à la pépinière de Berthier. Une présentation sommaire du projet sera donnée afin de pouvoir apprécier les chapitres ultérieurs qui décortiquent plus en détail ce dernier. De plus, l'objectif de ce mémoire sera donné.

1.1 Mise en contexte

La pépinière forestière de Berthier [10] (voir la Figure 1.1), qui fut la première pépinère de la province, fut fondée en 1908 par M. Gustave Clodomir Piché [1].

Un programme scientifique, en lien avec les forêts, fut mis en place à ses débuts. De plus, une école technique pour les gardes forestiers fut mise sur

1. Il faut noter qu'il fut le premier ingénieur forestier canadien français, entre autre. Pour sa biographie, voir :`http://faculty.marianopolis.edu/c.belanger/quebechistory/encyclopedia/GustaveClodimirPiche-HistoireduQuebec.htm`

FIGURE 1.1 – Vue aérienne de la pépinère de Berthier

place. À ces balbutiements la pépinière avait pour mission de reboiser les terrains sablonneux de sa région. Aujourd'hui, cette mission s'est élargie afin de reboiser de feuillus nobles le Québec. Il faut signaler que c'est la seule productrice (publique) de la province.

FIGURE 1.2 – Gustave Clodomir Piché (1879-1956), ingénieur forestier. À noter qu'il étudia aux universités Yale et Laval et fut très impliqué dans le domaine de la foresterie, à l'époque. Il fut membre de plusieurs sociétés importantes et fut décoré du Mérite Agricole de France.

Plus techniquement, la pépinière c'est :

- Jusqu'à 200 employés.
- Production moyenne de 4,5 millions de plants de fortes dimensions.
- Superficie de 155 ha.
- Les essences :
  - **Résineux en récipients :** épinette blanche, épinette de Norvège, épinette rouge, épinette noire, pin blanc et pin rouge.
  - **Feuillus en récipients :** bouleau jaune, bouleau à papier, chêne rouge, érable rouge, érable à sucre, frêne de Pennsylvanie et frêne

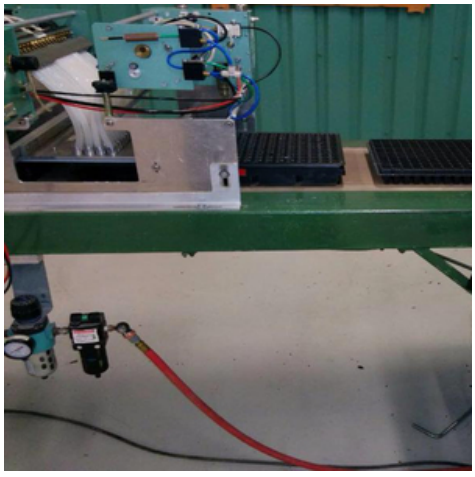
d'Amérique.

  - **Feuillus à racines nues :** caryer cordiforme, chêne rouge, chêne à gros fruits, cerisier tardif, érable à sucre, frêne d'Amérique, noyer noir et peupliers.
- Les plants feuillus sont produits pour toutes les régions.

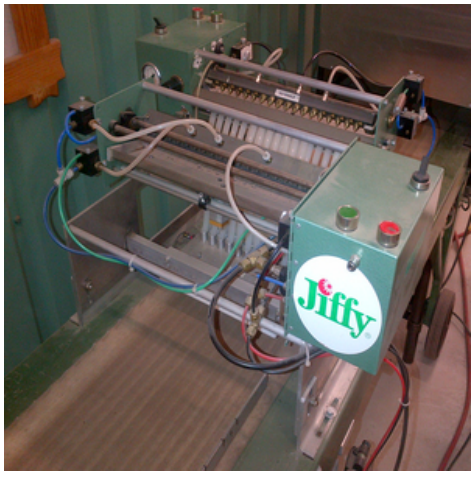
1.1.1 Description sommaire de la production

Il ne s'agit pas, ici, de décrire tous les processus de la production de la pépinière, mais bien de la sous-étape en lien avec le projet. C'est l'étape où les semences sont injectées dans un plateau comme celui de la Figure 1.3a de 12×24 cellules. Les plateaux, initialement vides de graine, défilent sur un convoyeur (voir Figure 1.3b). Un planteur pneumatique tel qu'apparaissant à la Figure 1.3c situé sur ce dernier sème tout-à-tour chaque cellule des plateaux. Pour qu'éventuellement les graines arrivent à maturité, la condition de base est qu'un récipient (cellule) doit ne contenir qu'une et une seule graine. Ainsi, un plant risquerait de ne pas voir le jour si un récipient ne possédait aucune ou plusieurs graines. Dès lors, il devient important de corriger, en cours de production, les situations où la condition de base n'est pas respectée.

(a) Un plateau typique



(b) Vue d'ensemble



(c) Le planteur

FIGURE 1.3 – Le semoir et ses composantes

1.1.1.1 Problématique

À l'heure actuelle, les opérateurs tentent de détecter les erreurs en inspectant du regard les plateaux qui défilent. Une fois, une erreur détectée elle est aussitôt corrigée par un autre opérateur. La Figure 1.4 illustre ce processus.Il doit y avoir plusieurs préposés à l'assurance qualité pour déceler et corriger les plateaux fautifs. De plus, il n'est pas déraisonnable de penser qu'il devient vite fastidieux (concentration, posture, vue,...) de faire manuellement de telles opérations. Ainsi, il devient intéressant de construire un prototype qui puisse automatiser ces tâches peu avenantes et répétitives.

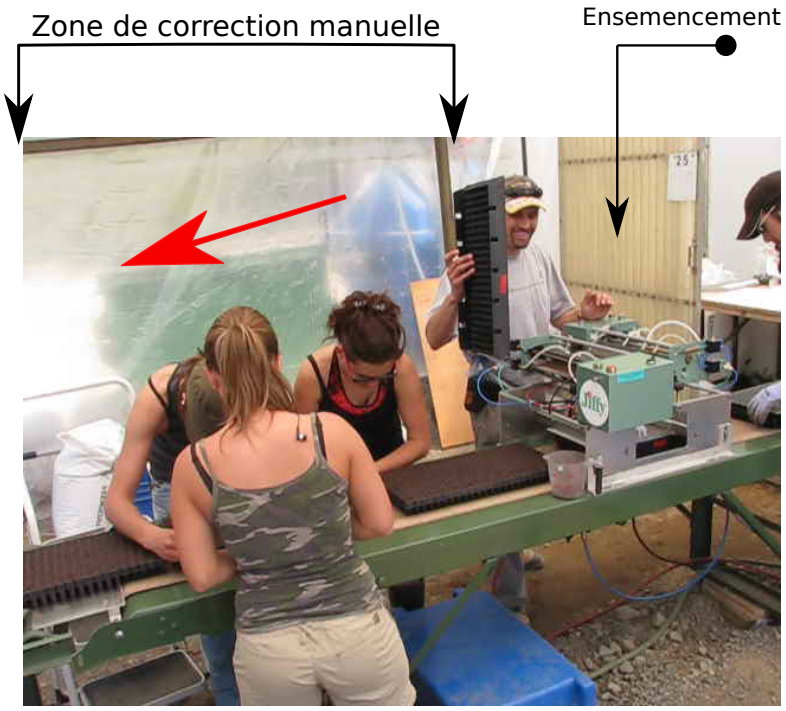


FIGURE 1.4 – Processus d'assurance qualité manuelle : les préposés à l'assurance qualité détectent les erreurs par la vue.

1.2 Projet et objectif du mémoire

1.2.1 Projet

Pour éviter de mobiliser plusieurs employés à la correction, un prototype fut imaginé (Figure 1.5) : Un système de vision qui analyse les plateaux un à la suite de l'autre et qui indique, en temps-réel, les cellules fautives, s'il y a lieu. Ce premier prototype est greffé au convoyeur traditionnel. Le système de vision est constitué d'un :

- caisson métallique avec une caméra proche infrarouge intégrée,
- un ordinateur qui permet d'analyser les images,
- un écran destiné au principal opérateur,
- un projecteur servant à projeter l'image des plateaux arrivés.

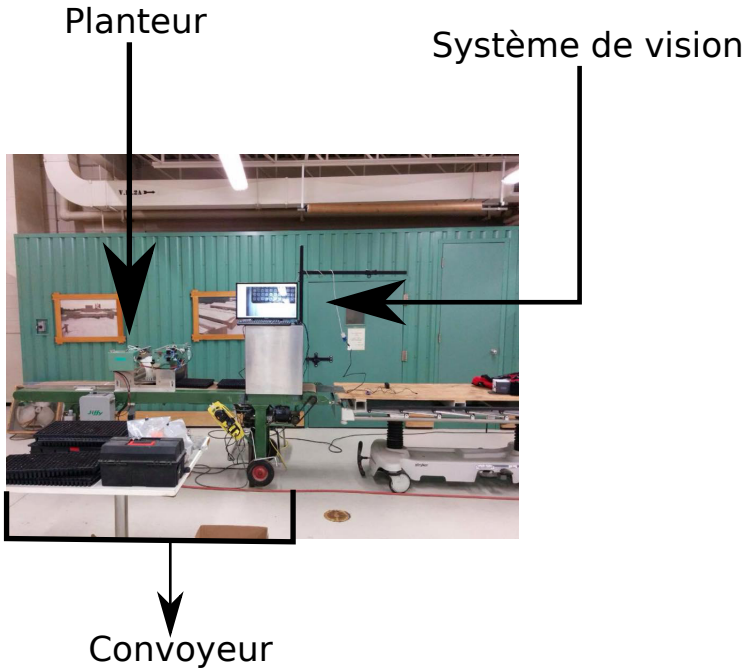


FIGURE 1.5 – Prototype du projet

Ainsi, dans les grandes lignes, le déroulement du processus d'automatisation de la phase d'assurance qualité est le suivant : le planteur passé, le plateau entre dans un caisson métallique et la caméra proche infrarouge capture les images du plateau défilant. Les images se font traiter par différents algorithmes permettant de segmenter chaque cellule des images d'un plateau pour, ensuite, détecter la présence ou l'absence de semence. Les résultats partiaux sont affichés à l'écran de l'opérateur principal. Entre autres, le nombre de cellules correctes et incorrectes est affiché en temps-réel informant celui-ci rapidement sur le fonctionnement du planteur. Un fois le plateau sorti du caisson et totalement analysé, un projecteur projette sur chaque cellule de ce dernier une couleur. Un code de trois couleurs fut choisi afin de permettre l'identification des trois conditions possibles :

- **I** : cellule sans graine (en rouge).
- **II** : cellule avec une graine (en bleu).
- **III** : cellule avec plus d'une graine (en jaune).

Ainsi, l'opérateur affecté à la correction peut rapidement identifier et corriger la situation. Voici un flot de contrôle simplifié du système à la Figure 1.6 :

1.2.2 Objectif du mémoire

L'objectif central du mémoire est de décrire dans le détail de quelle façon fonctionne le prototype du projet décrit précédemment. C'est-à-dire, donner ou rappeler, dans un premier temps, les notions de bases liées au traitement d'images et à un sous-ensemble de la vision par ordinateur dans le cadre du projet. C'est le but du chapitre 2. S'en suit, par la suite, l'intégration de ces notions à la description du fonctionnement du prototype. C'est la mission du chapitre 3. Par la suite, il s'agira de donner, au chapitre 4, les résultats de terrain et d'en déduire la validité. Ensuite, suivra une conclusion.

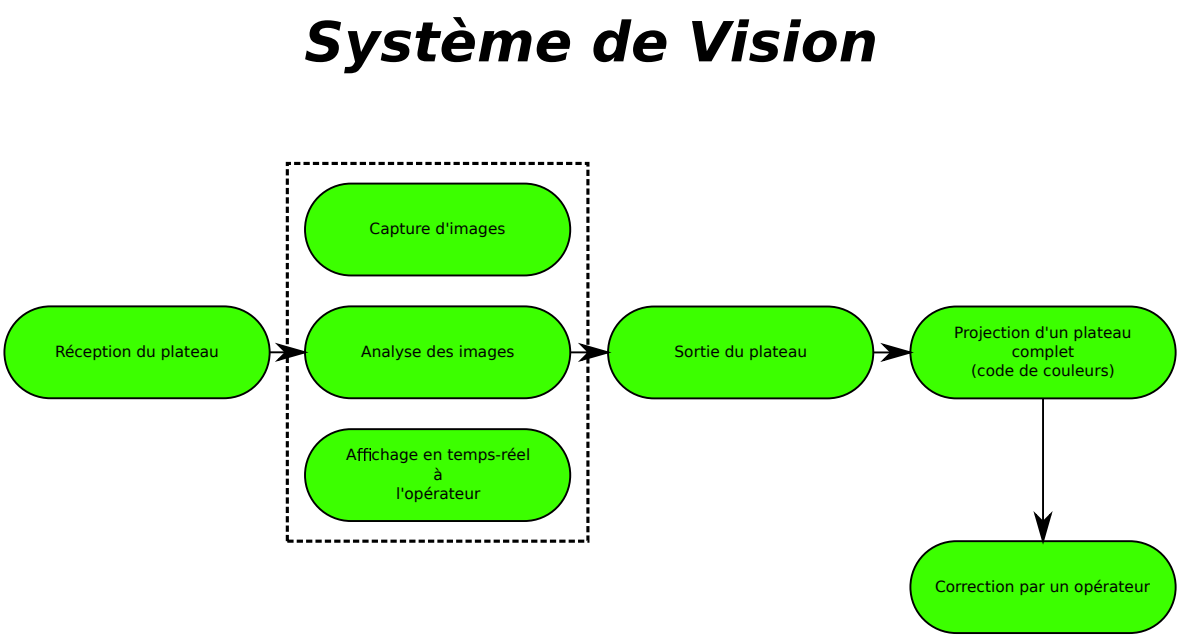


FIGURE 1.6 – Flot de contrôle

CHAPITRE 2

REVUE DE LITTÉRATURE

La science n'a pas de patrie.

Louis Pasteur

2.1 Introduction

Ce chapitre à pour mandat d'introduire ou de rappeler certaines notions au lecteur afin de permettre une meilleure compréhension ultérieure du projet. Il est entendu qu'il s'agit d'un amalgame de sujets divers pouvant avoir ou ne pas avoir de lien entre eux. Finalement, l'accent est mis sur la brièveté plutôt que sur le détail.

2.2 Éléments d'optique géométrique et physique

Le système de vision, utilisé dans le système de détection automatique de semences de résineux, est constitué d'une caméra captant dans les proches infrarouges. Dès lors, il devient impératif de connaître quelques concepts d'optique afin de bien circonscrire les problèmes éventuels liés au rayonnement dans le traitement d'image. À l'inverse, certaines caractéristiques propres à

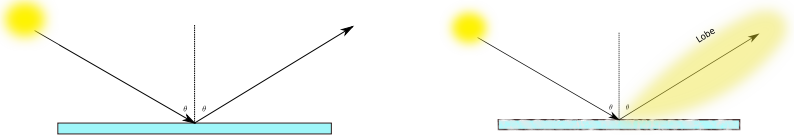
l'interaction des infrarouges avec la matière organique peuvent être utilisées à bon escient.

2.2.1 Réflexions

Il existe, en physique, deux grands types de réflexions : **spéculaire** et **diffuse**. Dans la réalité, la réflexion est un hybride de ces deux types. La réflexion, d'un point de vue théorique, découle du principe de Fermat [26]. Il ne sera question que de réflexion spéculaire ici. Il devient essentiel de bien comprendre de quelle façon se manifeste cette réflexion dans les images captées par le système de vision. En effet, si pour un objet donné avec deux parties similaires dans leurs natures/propriétés, mais orientées autrement dans l'espace reflètent différemment, elles peuvent occasionner des problèmes lors du traitement et de l'analyse des images. Ce problème sera traité ultérieurement.

2.2.1.1 Réflexion spéculaire

Dans ce type de réflexion [24] un rayon réfléchi est le symétrique du rayon incident par rapport à la normale. Les surfaces qui occasionnent une réflexion spéculaire sont généralement des matériaux conducteurs (réflexion métallique) ou constituées de milieux transparents (réflexion vitreuse), une vitre par exemple. Schématiquement, il possible de voir sur la Figure 2.1a la réflexion spéculaire idéale sur une surface polie et sur la Figure 2.1b une réflexion spéculaire sur une surface rugueuse. Sur une telle surface, le rayon réfléchi est plutôt un lobe.



(a) Surface polie (b) Surface rugueuse

FIGURE 2.1 – Deux types de réflexions spéculaires.

Il faut remarquer aussi que certains matériaux tels que les plastiques

possèdent certaines caractéristiques optiques [17] comme : la brillance spéculaire (*Gloss*), transparence, clarté,...Par exemple, il est possible de voir sur la Figure 2.2, provenant du système de vision, de la brillance spéculaire importante sur un plastique [1].

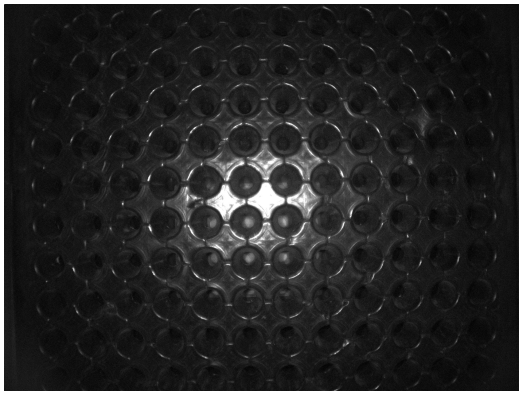


FIGURE 2.2 – Brillance spéculaire sur un couvercle de plastique

2.2.2 Infrarouge

L'infrarouge [11] est une région du spectre électromagnétique qui englobe approximativement les fréquences oscillants dans l'intervalle $f \in [3 \times 10^{11}, 4 \times 10^{14}]$ **Hz**. C'est Sir William Herschel (1738-1822) qui en 1800 en fit la découverte. Cette partie du spectre est située après la lumière rouge. La lumière visible a pour source microscopique les électrons de valences tandis que l'infrarouge, elle, provient des vibrations et rotations moléculaires. Il s'en suit que tout matériau absorbe et rayonne dans l'infrarouge. Le spectre infrarouge peut être grossièrement découpé en 4 sous-régions [2] (voir la Table 2.1) : les infrarouges proches (**IRP**), les infrarouges intermédiaires (**IRI**), les infrarouges lointains (**IRL**) et les infrarouges extrêmes (**IRE**). Typiquement,

1. Ce plastique est le même que celui des plateaux. C'est en fait, le couvercle d'un plateau nécessaire au planteur, mais qui est retiré avant l'entrée dans le système de vision.
2. Il n'existe pas, pour le moment, de consensus universellement accepté dans la littérature scientifique.

	$(\lambda_{\min}, \lambda_{\max}]$ **nm**
IRP	$(780, 3000]$
IRI	$(3000, 6000]$
IRL	$(6000, 15000]$
IRE	$(15000, 100000]$

TABLE 2.1 – Intervalles des longueurs d'onde

Finalement, ce type de rayonnement est non-destructif, disponible et très peu onéreux.

2.2.3 Proche infrarouge et matière organique

Il est reconnu [16] que beaucoup de bio molécules sont constituées de liens de types : C═O et/ou C═C, par exemple. Parallèlement, les **IRP** interagissent bien avec les harmoniques secondaires des vibrations de ces molécules au sens où les radiations sont bien absorbées. Ainsi, ce rayonnement pénètre profondément dans les semences, entre autre. Il en résulte que les **IRP** sont potentiellement très intéressantes pour investiguer et caractériser ce genre de matière. Sur l'image suivante (Figure 2.3) il est possible d'apprécier le fait que les graines sont naturellement misent en évidence.

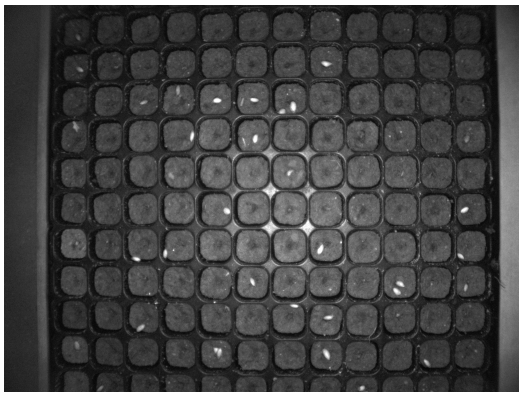


FIGURE 2.3 – Capture d'un plateau dans les **IRP**.

2.3 Semences

2.3.1 Essences

Une courte description des essences utilisées pour le projet sera donnée dans cette section.

2.3.1.1 Épinette noire

L'épinette noire possède une grande capacité d'adaptation. Elle couvre environ 20% de la superficie forestière de la province. Il est possible de retrouver de l'épinette noire du Yukon à Terre-Neuve, mais c'est en Ontario, au Québec et à Terre-Neuve qu'il est possible d'en retrouver le plus. L'arbre peut mesurer jusqu'à 20 [m] et vivre 175 à 200 ans. Finalement, le bois d'épinette noire est convoité. Pour une description complète voir [15].

2.3.1.2 Épinette rouge

Contrairement, à l'épinette noire, l'épinette rouge [5] est, en général, beaucoup plus vulnérable. De plus, elle ne fait pas partie de la forêt boréale, mais bien de la forêt feuillue et principalement mixte. Elle peut atteindre jusqu'à 26 [m] de hauteur et vivre 400 ans.

2.3.1.3 Pins rouge et blanc

Il est possible de retrouver ces espèces partout au Québec à l'exception du Nord-du-Québec [12]. La concentration est plus élevée en Abitibi-Témiscamingue et en Outaouais, par contre. Ils mesurent entre 20 et 30 [m] (le pin rouge est plus petit). et peuvent vivre des centaines d'années. Ce sont des essences nobles et sont très prisées par l'Industrie. Notamment, dans la construction. Il est possible de voir sur la Figure 2.4 la forme en deux dimensions des différentes semences.

(a) Épinette noire

(b) Épinette rouge

(c) Pin rouge

(d) Pin blanc

FIGURE 2.4 – Les différentes essences (même échelle).

2.4 Vision par ordinateur

2.4.1 Traitement d'image

Il existe une panoplie d'opérateurs en traitement d'image qui permettent d'améliorer l'aspect d'une ou des images. Ces opérateurs sont tantôt locaux, tantôt globaux. En ce qui concerne le projet, les images acquissent sont plusieurs fois modifiées afin d'extirper des informations utiles permettant de tirer des conclusions (par exemple, décider si un contour trouvé est une semence ou pas.). Dans un premier temps, le prétraitement sera abordé. Puis, des traitements de plus hauts niveaux seront décrits.

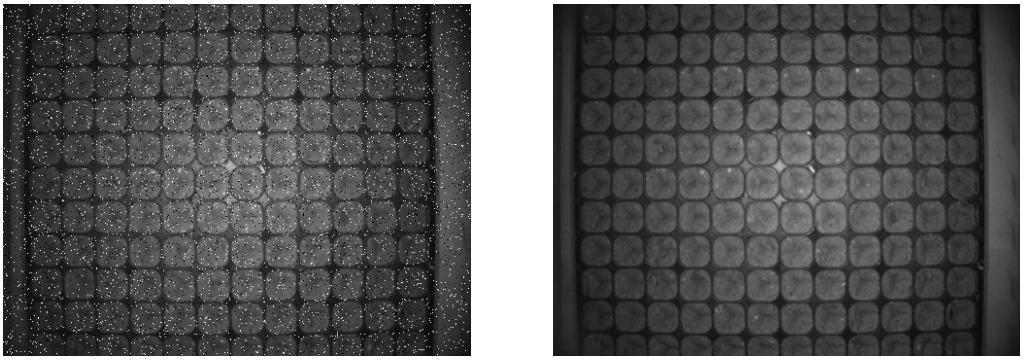
2.4.1.1 Prétraitement

Il faut signaler, d'emblée, que les images acquissent par la caméra du système sont en niveau de gris. Plus précisément, des images constituées de

1928x1448 pixels ayant chacun une valeur de niveau de gris dans l'intervalle $[0, 255]$.

2.4.1.1.1 Réduction du bruit

Les dispositifs qui captent des images sont imparfaits. Dès lors, les images acquissent sont bruitées. Il importe donc de l'éliminer autant que faire se peut tout en minimisant l'élimination de bons pixels. Un des filtres utilisé est le **filtre médian** qui élimine le bruit impulsionnel (poivre et sel). L'idée [6] est d'enlever les pixels ayant des valeurs extrêmes et donc improbables en les rendant plus proches des pixels voisins. L'effet de ce filtre peut être vu sur la Figure 2.5



(a) Bruit poivre et sel (b) Bruit diminué

FIGURE 2.5 – L'effet du filtre médian.

2.4.1.1.2 Seuillage binaire adaptatif

Plusieurs fonctions en traitement d'image (détection de contours, notamment.) nécessitent une image binaire. C'est-à-dire, dans le cas qui importe ici, un background noir et le ou les objets d'intérêts en blanc. Il existe une multitude de façons de faire, mais une façon intéressante est le seuillage binaire adaptatif. L'image est parcourue par une fenêtre carrée de $b \times b$ et un seuil $T \in [0, 255]$ est choisi. Le seuillage s'effectue donc localement dans chaque sous-région. Cette méthode est donc utile [2] lorsque les gradients d'illumination ou de réflexion sont élevés. Schématiquement, sur la Figure

2.6, le seuillage s'effectue dans le bloc, schématisé par le cadrage et non pas globalement.

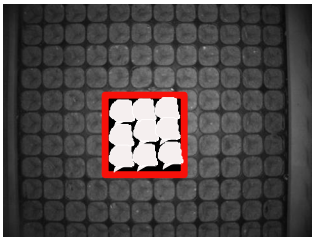


FIGURE 2.6 – Image d'un seuillage adaptatif.

2.4.1.1.3 Méthode d'Otsu

Tel qu'indiqué dans la section précédente, un seuil T doit être choisi afin de pouvoir faire un seuillage. Il est possible, par exemple, de fixer la valeur manuellement. Il semble plus intéressant, par contre, d'automatiser le choix de T. Pour ce faire, plusieurs méthodes existent, mais le choix, ici, s'arrête sur la méthode d'Otsu [7]. Sans surprise, l'objectif[3] est de de créer deux classes : l'objet recherché, et l'arrière-plan. La méthode d'Otsu repose sur l'histogramme des valeurs de couleurs en niveaux de gris où $\mathbb{P}(i)$ avec $i \in [0, 255]$ est la probabilité (approche fréquentielle) de tomber sur un pixel en niveau de gris d'intensité i. Le principe est de trouver le seuil $t \in \mathbb{N}$ qui :

- permette de minimiser la variance intra-classe

$$\sigma_w^2(t) = q_1(t)\,\sigma_1^2(t) + q_2(t)\,\sigma_2^2(t)$$

  – avec

$$q_1(t) = \sum_{i=0}^{t} \mathbb{P}(i)$$

3. La méthode, dans le cadre du projet, est appliquée avec le filtre adaptatif décrit plus haut.

– et

$$q_2(t) = \sum_{i=t+1}^{255} \mathbb{P}(i)$$

– aussi,

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} i\mathbb{P}(i)$$

– ainsi que,

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{255} i\mathbb{P}(i)$$

- ou maximiser la variance inter-classe

$$\sigma_b^2 = q_1(t)q_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2$$

S'ensuit un exemple de cette méthode à la Figure 2.7.

(a) Image originale (b) Image seuillée

FIGURE 2.7 – Méthode d'Otsu

2.4.1.2 Morphologie

La morphologie mathématique [29] est une branche des mathématiques et de l'informatique et trouve beaucoup d'applications en traitement de l'image, notamment. Actuellement, dans le cadre du projet, deux opérations de bases tirées de cette discipline sont utilisées dans la phase de prétraitement.

À savoir, l'ouverture et la fermeture. En effet, elles permettent d'éliminer des surfaces indésirables ou de lisser les contours des objets à analyser. Il sera donc question, dans cette section, de donner les définitions de bases afin de comprendre ces deux opérations.

L'image E est vue comme $E \subset \mathbb{Z}^2$. Elle est binaire (autrement dit, elle comporte deux valeurs de pixel. Par exemple, 0 et 255). L'élément structurant $B \subset E$ agit comme une sonde et peut revêtir différentes formes (croix, ellipse, carré,...). De manière intuitive, ce dernier peut-être décrit comme une configuration de pixels ayant une origine. Chaque pixel de l'image est sondé par l'élément structurant. Plus précisément, lorsque l'origine est alignée avec un pixel de l'image, l'intersection (au sens ensembliste) de ce dernier avec l'image définit la région ou le filtre morphologique sera appliqué. En définition, pour $x \in E$, l'ensemble B_x est l'ensemble B translaté par x et est noté comme suit

$$B_x = \{b + x | b \in B\}$$

. Soit $X \subset E$, alors sont définis les opérations non-linéaires :

- la dilatation :

$$\delta_B(X) := X \oplus B = \bigcup_{x \in X} B_x$$

- l'érosion :

$$\epsilon_B(X) := X \ominus B = \{x | B_x \subset X\}$$

En bref, l'érosion [1] remplace le pixel courant par la valeur minimum du pixel contenu dans le sous-ensemble et c'est l'inverse pour la dilatation. Puisque l'image est binaire, le pixel devient donc allumé ou éteint en fonction des pixels voisins visités par l'élément structurant. En termes d'arrière-plan (noir) et d'avant-plan (blanc) : avec l'érosion, si l'élément structurant à un pixel donné est en contact avec l'arrière-plan, alors le pixel s'éteint.

Inversement, pour la dilatation, si l'élément structurant, en contact avec un pixel d'arrière-plan, touche un pixel de l'avant-plan, alors ce dernier s'allume. Alors, en combinant ces opérations, il est alors possible d'enlever des objets parasites. En fait, sont définit les opérations suivantes :

- l'ouverture :

$$\gamma_B(X) := X \circ B = \delta_B(\epsilon_B(X))$$

- la fermeture :

$$\phi_B(X) := X \bullet B = \epsilon_B(\delta_B(X))$$

Ainsi, l'ouverture permet d'éliminer de petits contours blancs (en fonction de B) et la fermeture permet de remplir certains trous noires d'un objet. En effet, dans le contexte du projet, les petits contours blancs pourraient être, par exemple, des grenailles parfois présentes dans les cellules d'ensemencement. Tandis que le contour d'une graine aurait pu se détériorer lors du traitement d'où le remplissage. Ainsi, la Figure 2.8 témoigne, schématiquement, des résultats de ces deux opérateurs morphologiques :

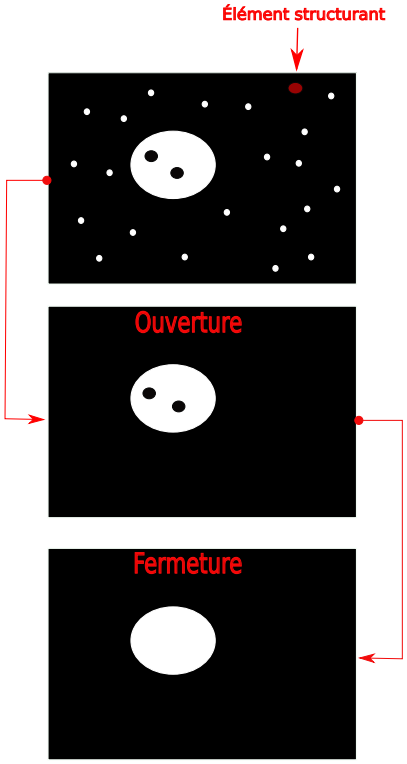


FIGURE 2.8 – Morphologie : l'ouverture et la fermeture.

2.4.1.3 Détection de contours

La détection de contours n'est pas simple et plusieurs approches peuvent être employées. Par exemple, le modèle de contour actif, introduit par Kass et Witkin [4], utilise une approche variationnelle. Il existe aussi des approches morphologiques. En ce qui concerne le projet, la fonction [4] qui détecte les contours d'une image binaire utilise un algorithme introduit par Suzuki et Abe dans l'article [23] : Topological Structural Analysis of Digitized Binary Images by Border Following. Cet algorithme inspecte chaque pixel et leur connectivité afin de le classer, grâce à un code (Freeman), comme étant un élément de contour ou pas. C'est façon de faire est beaucoup plus proche de la morphologie mathématique que les contours actifs.

4. Plus tard, la bibliothèque logicielle OpenCV sera introduite.

2.4.2 Analyse de contours

2.4.3 Formes et géométries des contours

Il est important de connaître, dans le cadre du projet, certaines caractéristiques géométriques des graines afin de pouvoir discriminer ces dernières d'objets géométriques différents mais détectés comme des graines par le système. Dès lors, en connaissant certains attributs de forme propres aux différentes essences il est possible d'écarter ces objets non voulus.

2.4.3.1 Moments statistiques

Il existe plusieurs façons d'aborder les moments statistiques. Du point de vue de la physique, par exemple ou encore une approche probabiliste (peut-être plus proche du traitement d'image) pourrait aussi faire l'affaire. Peu importe l'approche, les moments caractérisent une forme (la forme d'un corps rigide, la forme d'une densité de probabilité, la forme d'un contour dans une image,...). En ce qui concerne le projet, les moments sont utilisés afin d'extraire certaines caractéristiques du contour d'une semence donnée.

2.4.3.1.1 Définition mathématique

Sous l'angle des mathématiques pures [14], les moments sont les projections d'une fonction sur une base polynomiale au même titre que la transformée de Fourier sur une base harmonique. Donc, pour une fonction [5] f : $\mathfrak{D} \subset \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ Le terme général des moments M_{pq} avec $p, q \in \mathbb{N}$ en coordonnées cartésiennes est donné par

$$M_{pq} = \int\int_{\mathfrak{D}} p_{pq}(x, y)\, f(x, y)\, dx\, dy. \qquad \text{avec } p_{pq}(x, y) \in \mathbf{P}_{p+q}$$

Ainsi, selon le choix du type de polynôme de la base, $p_{pq}(x, y)$, l'information tirée d'un moment donné est différente. En effet, il est possible de

5. Il doit y avoir des restrictions (comme la continuité, entre autres) sur f qui ne seront pas données,ici. De plus, on considère le cas de dimension 2

choisir une base orthogonale (comme les polynômes de Legendre) ou complexe, par exemple. Dans le cadre du projet, la base utilisée est celle de la forme $p_{kj}(x,y) = x^k y^j$. Cela engendre les moments dits géométriques et qui sont probablement le cas le plus simple et classique.

2.4.3.1.2 Définition discrète

En considérant une image [21] $N \times M$ comme $P(n,m) \in \{0,1,\ldots,255\}$, alors les moments centralisés sont données par

$$\mu_{pq} = \sum_{n=1}^{N} \sum_{m=1}^{M} (n-\overline{n})^p (m-\overline{m})^q P(n,m)$$

Il faut noter que dans le cadre du projet il y a la restriction [6] suivante :$p+q \leq 3$. De plus, dans la définition générale c'est toute l'image qui est prise en compte, mais il est possible de considérer un contour comme étant un sous-ensemble de cette dernière. Il suffit alors, de calculer sur les points du contours seulement. Ainsi, il est possible, toujours pour une image binaire, de calculer l'aire (μ_{00}), le périmètre et autres propriétés grâce aux moments.

6. OpenCV ne va pas au-delà du troisième ordre. Dans un ordre d'idées, les calculs s'effectuent via la formule de Green et non celle présentée plus haut.

2.4.3.2 Facteurs de forme

Malheureusement, la détection de contours n'est jamais vraiment parfaite et donne lieu, parfois, à de faux négatifs : du bruit de contour. Il existe certains descripteurs de forme simples pouvant aider à discriminer les bons et les mauvais contours et ce, à l'aide de peu de paramètres. Il s'agit, avec l'aide des facteurs de forme, de reconnaître si un contour, dans le cas qui nous intéresse, est une graine de semences ou pas. Une introduction de certains termes s'impose.

- A : L'aire de la forme.
- P : Le périmètre de la forme.
- $D_{\max}$: Le plus grand des diamètres de la forme.
- $D_{\min}$: Le plus petit des diamètres de la forme.
- ConvexA : L'aire de l'enveloppe convexe.
- ConvexP : Le périmètre de l'enveloppe convexe.

À titre de rappel, une définition [22] mathématique de l'enveloppe convexe et la suivante :

Definition (Enveloppe Convexe). *L'enveloppe convexe d'un sous-ensemble S de $\mathbb{R}^n$ est l'intersection de tous les sous-ensembles convexes de $\mathbb{R}^n$ contenant S. L'enveloppe convexe de S est par conséquent le plus petit sous-ensemble convexe de $\mathbb{R}^n$ qui contienne S.*

Ou, plus intuitivement, tiré directement du Wiki :

≪ Dans un plan, l'enveloppe convexe peut être comparée à la région limitée par un élastique qui englobe tous les points qu'on relâche jusqu'à ce qu'il se contracte au maximum. L'idée serait la même dans l'espace avec un ballon qui se dégonflerait jusqu'à être en contact avec tous les points qui sont à la surface de l'enveloppe convexe. ≫

Il est possible d'apprécier le concept d'enveloppe convexe sur la Figure 2.9

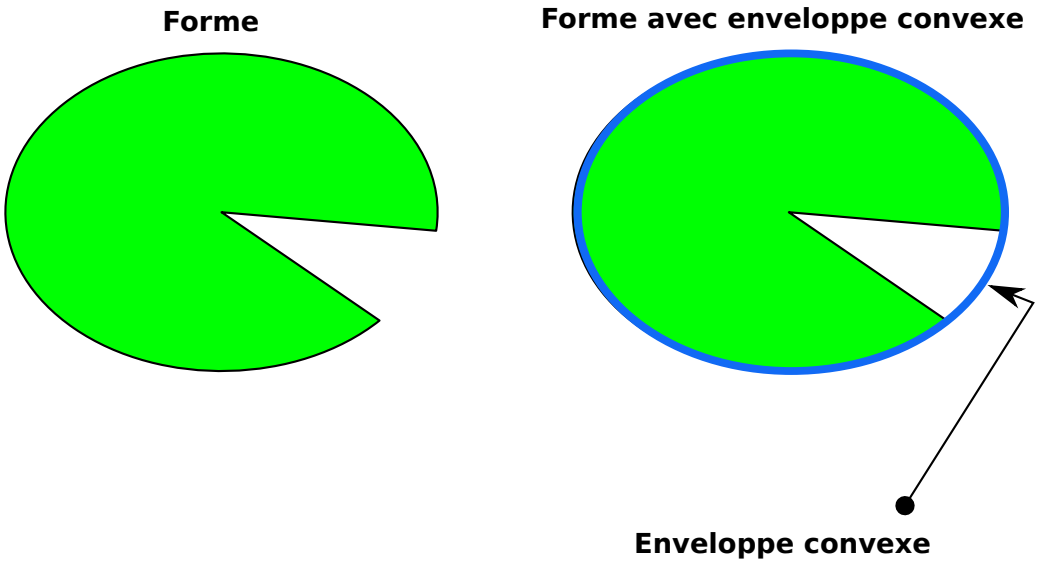


FIGURE 2.9 – Enveloppe convexe

Voici, au Tableau 2.2 , quelques facteurs [9] utiles [7] ainsi que leur définition et la valeur de ces derniers si la forme en question était une cercle ou un carré :

Mesures	Définition	Cercle	Carré
Facteur de forme	$\frac{4\pi \cdot A}{P^2}$	1	$\frac{\pi}{4}$
Aspect ratio	$\frac{D_{\max}}{D_{\min}}$	1	1
Solidité	$\frac{A}{\text{ConvexA}}$	1	1
Compacité	$\frac{\sqrt{\frac{4A}{\pi}}q}{D_{\max}}$	1	$\frac{\sqrt{2}}{\pi}$
Convexité	$\frac{\text{ConvexP}}{P}$	1	1

TABLE 2.2 – Différents facteurs de forme.

Manifestement, les facteurs de forme sont sans dimension, invariants sous rotation et sous changement d'échelle.

7. Il semble qu'il n'y ait pas consensus sur les noms utilisés.

2.5 Langages de programmation

Le choix d'un langage de programmation pour un projet donné est important. En effet, un choix peut reposer sur plusieurs critères [8] tels que la nature du langage, la portabilité, la stabilité, la pérennité, l'ouverture à d'autres langages, la sécurité, la popularité, les licences en jeu,... En ce qui concerne le projet, le C# fut choisi pour sa convivialité. Le langage Python, quant à lui, fut utilisé pour le prototypage logiciel rapide.

2.5.1 C#

Le C# est un langage [28] relativement récent apparu en 2001. Il est à paradigmes multiples (impératif,orienté objet, fonctionnel,...). Il fut crée par Microsoft et est sans doute le langage par excellence du **.NET**. Il est similaire à certains égards au langage JAVA. Il fut inspiré de plusieurs langages : C++, Eiffel, Pascal Objet, Haskell,... Finalement, un défaut objectif de ce langage est le manque de portabilité.

2.5.2 Python

Python [30] apparu en 1990. Son auteur est Guido van Rossum. À la grande différence du C# , Python n'est pas compilé, mais interprété [9]. À l'instar du C#, c'est un langage multi-paradigmes. Il est très prisé par la communauté scientifique. De plus, Python roule très bien sur différents systèmes tels que Unix, GNU-Linux, Windows, Mac OS, iOS, Android, par exemple. Ce langage est placé sous licence libre. Il est possible, par contre, d'affirmer qu'en général un programme Python est moins rapide dans son exécution qu'un programme C# équivalent, sauf exception.

8. Les critères sont directement tirés du site `http://www.commentcamarche.net/faq/3964-programmation-criteres-de-choix-d-un-langage-framework`

9. Il faut remarquer, par contre, qu'il est possible de compiler un programme Python.

2.5.3 Bibliothèques logicielles

Plusieurs bibliothèques sont utilisées. La principale est l'**OpenCV** [13] (Open Source Computer Vision Library) qui, comme son nom l'indique, est une librairie libre dédiée à la vision par ordinateur. Elle comporte plus de 2500 algorithmes dans le domaine. Une grande communauté la supporte et elle est utilisée autant par des compagnies (Google, Yahoo, Intel, Toyota...) que par des hobbyistes. Son support à diverses applications est très varié. Par exemple, détecter par vidéo surveillance des intrusions, des noyades, permettre à des robots de prendre des objets...L'OpenCV est compatible avec divers systèmes d'exploitations (Windows, Linux, Os X, ...) et est écrite en C++, mais il existe plusieurs bibliothèques d'enveloppes qui permettent de l'utiliser avec d'autres langages (Emgu CV, Python CV, Ruby CV, Perl CV). Finalement, une des forces d'OpenCV est son efficacité dans les applications temps-réels. Une autre bibliothèque intéressante [10] est l'extension Python : **Numpy** [27]. Cette dernière se spécialise dans le traitement et la manipulation de tableaux, matrices et fonctions mathématiques opérant sur ces dernières. Dans certains cas, la vitesse d'une fonction Numpy peut s'approcher de son équivalent C. Ainsi, par exemple, le bout de code suivant permet d'ouvrir une image nommée *image.jpg* et calculer son histogramme :

```
1 import numpy as np
2 import cv2
3
4 # lire image
5 img = cv2.imread('image.jpg',0)
6
7 hist,bins = np.histogram(img.flatten(),256,[0,256])
8
9 cdf = hist.cumsum()
```

10. Dans ce mémoire, les exemples, sauf avis contraire, sont donnés en langage Python et ce, même si le projet est écrit en C sharp. En effet, Python est intuitif et ce langage fut utilisé intensivement pour le prototypage.

2.6 Maîtrise statistique des procédés

Cette section sert à donner quelques outils afin d'évaluer statistiquement la qualité de la production. Il faut noter que la description se veut la plus simple possible quoique certains raccourcis peuvent être utilisés. Néanmoins, le tout est donné en se voulant correct du point de vue mathématique. Il existes plusieurs ouvrages spécialisés pour une description rigoureuse.

2.6.1 Modèle probabiliste pour une cellule

Il s'agit, dans un premier temps, de donner un modèle probabiliste le plus simple possible, quitte à le raffiner plus tard si nécessaire, qui compte le nombre de graines observées dans une cellule.

Le modèle probabiliste $(\Omega, \mathcal{P}(\Omega), \mathbb{P})$ de paramètres $p, q > 0$ et $p + q < 1$ est le suivant :

- $\Omega = \{\{0\}, \{1\}, \{2\}\}$
- $\mathbb{P}(\{0\}) = p$
- $\mathbb{P}(\{1\}) = q$
- $\mathbb{P}(\{2\}) = 1 - p - q$

Par exemple, l'événement $\{0\}$ signifie qu'il n'y a pas de graine, $\{1\}$ signifie qu'il y a une graine alors que l'événement $\{2\}$ témoigne qu'il y a deux graines. Il possible de désigner une variable aléatoire (v.a)

$$X : \Omega \to \{0, 1, 2\} : \textbf{le nombre de graines observées}.$$

dont la loi est donnée au Tableau 2.3

x	0	1	2
$\mathbb{P}(X = x)$	p	q	$1 - q - p$

TABLE 2.3 – Loi de X.

2.6.1.1 Estimation des paramètres : maximum de vraisemblance

Soit l'observation du nombre de graines de n cellules dont le résultats est $x_1, x_2, \ldots, x_n \in \{0, 1, 2\}$ alors [11], l'ajustement au modèle pour cette expérience est donc le triplet $(\Omega^n, \mathcal{P}(\Omega^n), \mathbb{P})$. Dès lors, l'événement associé à cet échantillon est $\{\{x_1\} \cap \{x_2\} \cap \ldots \cap \{x_n\}\}$. La supposition est que ces observations sont indépendantes. Ainsi, la probabilité des intersections des observations équivaut à multiplier les probabilités de chaque observations. Désignant $X := X(\{x_i\}) \in \{0, 1, 2\}$ $\forall i = 0, 1, \ldots, n$:

$$\mathbb{P}(\{\{X = x_1\} \cap \{X = x_2\} \cap \ldots \cap \{X = x_n\}\}) = \prod_{i=1}^{n} \mathbb{P}(X = x_i).$$

Soient :

- n_1 : le nombre de cellules n'ayant aucune graine.
- n_2 : le nombre de cellules ayant une graine.
- $n - (n_1 + n_2)$: le nombre de cellules ayant deux graines.

Alors, clairement, il y a un nombre n_1 de p, un nombre n_2 de q et un nombre $n - (n_1 + n_2)$ de $1 - q - p$ dans le produit. Donc,

$$\prod_{i=1}^{n} \mathbb{P}(X = x_i) = p^{n_1} q^{n_2} (1 - p - q)^{n - n_1 - n_2}.$$

Maintenant, les paramètres p, q doivent être estimés. Par le principe de maximum de vraisemblance [8], le meilleur choix à faire est celui dont les paramètres maximisent cette probabilité. La fonction suivante doit donc être maximisée (sous les contraintes données plus haut) :

$$L(x_1, x_2, \ldots, x_n \mid p, q) = p^{n_1} q^{n_2} (1 - p - q)^{n - n_1 - n_2}.$$

11. Une petite remarque s'impose afin d'être clair et consistant en terme de notation. Par exemple, $\{x_1\}$ représente l'événement : observation du nombre de graine dans la cellule étiquetée par un et $x_1 = 0$ ou 1 ou 2 désigne le nombre de graines dans la cellule.

Plus formellement,

$$\arg\max_{\mathcal{D}} \left(L\left(x_1 \ldots x_n : (p,q)\right)\right)$$

où $\mathcal{D} = \{(p,q) \mid p,q > 0 \text{ et } p+q < 1\}$

Cela est un problème d'optimisation non-linéaire avec contraintes. Étant donné que c'est une discipline spécialisée, aucune méthode formelle ne sera utilisée. L'heuristique à employer est la suivante : une grille est construite. C'est en quelque sorte un plan cartésien discret. C'est grille est une matrice telle que chaque élément soit une coordonnée. Puis, chaque élément est remplacé par la valeur de la fonction en ce point. Étant donné la contrainte, seuls les éléments supérieurs droits excluant la diagonale sont retenus. La valeur maximale est recherchée. Ensuite, la condition est évaluée. Si elle n'est pas rencontrée, l'élément est mis à zéro et une nouvelle recherche s'effectue et ce, jusqu'à ce que la condition soit rencontrée. La coordonnée trouvée est jugée comme étant la position à laquelle la fonction est maximale. Cette heuristique est grossièrement décrite dans le tableau suivant 1 :

Algorithm 1 Recherche du point maximisant

Require: $n_1, n_2, n \in \mathbb{N} \wedge (p,q) \in (0,1)$
Ensure: $L\left(p,q\right) = p^{n_1} q^{n_2} \left(1-p-q\right)^{n-n_1-n_2}$
  step $\leftarrow 1000$\{l'intervalle est subdivisé en 1000.\}
  $p, q \leftarrow \left(0, \frac{1}{\text{step}}\right] \times \ldots \times \left(\frac{\text{step}-1}{\text{step}}, 1\right)$
  $G = p \otimes q$\{matrice où les entrées sont les points du pavé.\}
  $L\left(G\right)$\{L est évaluée à tous les points du pavé.\}
  $L\left(G\right)_{ij} \leftarrow 0 \forall j \leq i$
  $(p,q) \leftarrow \max\{L\left(G\right)\}$
Ensure: $p+q < 1$\{sinon, mettre $L(q,p) = 0$ et recommencer\}

Au dernier chapitre, la méthode sera illustrée avec un exemple concret.

2.6.1.2 Estimation des paramètres : méthode des moments

L'idée générale est de tenter de faire coïncider les moments [25] d'un échantillon avec celui de la distribution théorique. Pour rappel, les moments

théoriques sont données par $\mathbb{E}\left(X^k\right) = \sum_{i=1}^{n} x_i^n p_i$ et ceux du n-échantillon $\frac{1}{n}\sum_{i=1}^{n} X_i^k$ pour $k \in \mathbb{N}$. Soit

$$\overline{x} = \sum_{i=1}^{n} x_i \quad \text{et} \quad t^2 = \sum_{i=1}^{n} x_i^2$$

les moments d'ordre 1 et 2 de l'échantillon. Il faut maintenant calculer les moments théoriques :

$$\begin{aligned} \mathbb{E}(X) &= x_1 \cdot p_1 + x_2 \cdot p_2 + x_3 \cdot p_3 \\ &= 0 \cdot p + 1 \cdot q + 2 \cdot (1 - q - p) \\ &= q + 2(1 - q - p) \end{aligned}$$

et

$$\begin{aligned} \mathbb{E}\left(X^2\right) &= x_1^2 \cdot p_1 + x_2^2 \cdot p_2 + x_3^2 \cdot p_3 \\ &= q + 4(1 - q - p) \end{aligned}$$

Le système qu'il faut résoudre sachant que $p, q \to \widehat{p}, \widehat{q}$ est donc :

$$\begin{cases} \overline{x} = \widehat{q} + 2(1 - \widehat{q} - \widehat{p}) \\ t^2 = \widehat{q} + 4(1 - \widehat{q} - \widehat{p}) \end{cases}$$

Simplement,

```
In [1]: from sympy import *

In [2]: q, p =symb
symbol   symbols

In [3]: q, p, x, t =symbols('q p x t')

In [4]: eq1 = q+2*(1-p-q)-x

```

```
In [5]: eq2 = q+4*(1-p-q)-t

In [6]: solve([eq1,eq2],q,p)
Out[6]: {p: t/2 - 3*x/2 + 1, q: -t + 2*x}
```

Donc, $\widehat{p} = \frac{t^2 - 3\overline{x} + 2}{2}$; $\widehat{q} = 2\overline{x} - t^2$. Il faudra toujours veiller à ce que les contraintes soient respectées. Voici un petit example théorique d'utilisation :

```
import numpy as np

In [2]: echantillon = [1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,2,2,1,1]

In [3]: moyenne =np.mean(echantillon)

In [4]: t_carre = np.mean(np.power(echantillon,2))

In [5]: p=(t_carre -3*moyenne+2)/2.0

In [6]: print p
0.142857142857

In [7]: q=2*moyenne-t_carre

In [8]: print q
0.761904761905

In [8]: p+q < 1
Out[8]: True

In [9]: 1-q-p
Out[9]: 0.095238095238095122
```

2.6.2 Chaîne de Markov

L'étude des chaînes de Markov est large et élaborée. Par contre, avec quelques définitions fondamentales de cette théorie, il sera possible d'estimer certaines occurrences qui pourraient surgir. Plus précisément, il sera possible de répondre, au moins partiellement, à une question du type : quelle est la probabilité que deux cellules adjacentes contiennent 2 graines chacune ? Finalement, une fois de plus, les notions seront données à titre de rappel et un petit sous-ensemble de la théorie est utilisée et ce, sans trop de formalité.

2.6.2.1 Définitions

L'idée est d'appliquer le modèle probabiliste, vu plus-haut, non pas à une cellules, mais bien une suite de cellule. En fait, le plateau ne sera pas traité dans son ensemble, mais bien rangée par rangée pour des raisons qui seront données dans le chapitre consacré à la définition détaillée du projet [12]. En effet, chaque état $s \in \Omega$ d'une cellule est la réalisation de la v.a $X(s) = x$ décrite plus haut. Pour caractériser une suite de cellules d'une rangée fixée, il suffit d'indexer la suite de réalisations par $n \in \mathbb{N}$, par exemple. Par convention (établi par la direction de propagation du plateau), la cellule la plus à gauche est étiquetée par $n = 0$, la suivante par $n = 1$ et ainsi de suite. Ainsi, il est possible d'observer systématiquement toutes les cellules d'une rangée ou un sous-ensemble, par exemple, toutes les deux cellules. Cela est illustré à la Figure 2.10 :

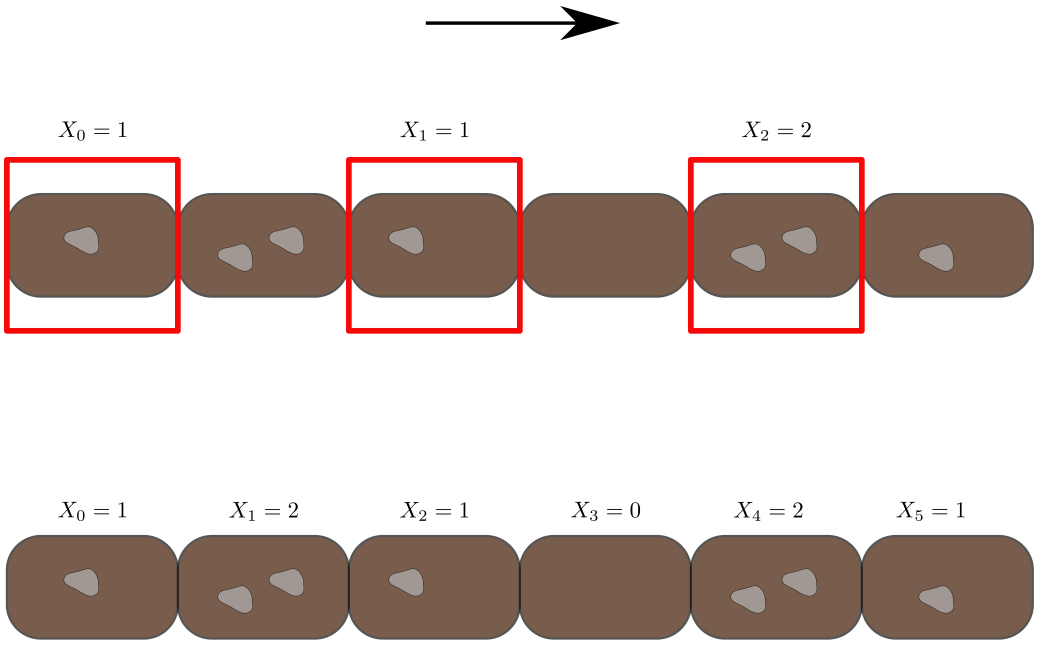


FIGURE 2.10 – Différentes façons d'observer des cellules

12. Il suffit de dire, ici, que le planteur est divisé en plusieurs sous-unités (blocs de cellules de 4 rangées de 12 cellules) dont chacune projette une ou des semences sur le plateau. Chaque sous-unité traite une rangée à la fois par saut de 4 cellules. Donc, une sous-unité peut ensemencer, pour une rangée donnée, une cellule, puis la quatrième plus loin, puis la huitième et ainsi de suite.

Donc, d'un point de vue mathématique [19], tout cela peut être considéré comme un processus stochastique $\{X_n \mid n \in \mathbb{N}\}$. L'hypothèse suivante est faite : pour tout n, la probabilité que le processus prenne une valeur quelconque à $n' > n$ ne dépend pas des valeurs prises par la processus avant n. Dit plus simplement, le processus est sans mémoire. Il est donc possible de considérer le processus comme étant markovien. Plus formellement, si X_n suit le modèle $\forall n$, alors

$$\mathbb{P}\left(\{X_{n+1} = x\} \mid \{X_1 = x_1\} \cap \ldots \cap \{X_n = x_n\}\right) = \mathbb{P}\left(\{X_{n+1} = x\} \mid \{X_n = x_n\}\right) = p_{n,n-1}$$

Il est donc possible de créer une matrice de transition stochastique $P = (p_{ij})$ $i, j = 0, 1, 2$ telle que $\dim(P) = |\Omega| \times |\Omega|$ qui décrive les différentes transitions d'une état à l'autre. Par exemple, p_{01} est la probabilité de passer d'une cellule au niveau n n'ayant aucune graine à une cellule en possédant une au niveau suivant $(n+1)$. Le graphe de la Figure 2.11 est équivalent à P :

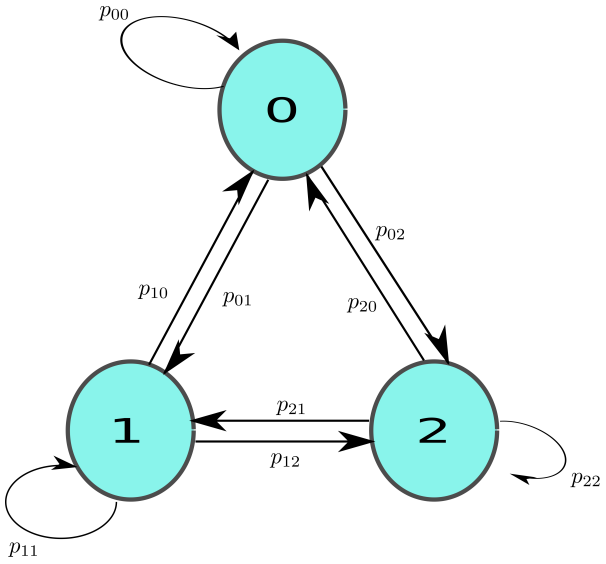


FIGURE 2.11 – Graphe des transitions

La matrice est donnée par :

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix}$$

Elle se décrit comme représentant l'ensemble des transitions possibles d'une cellule vers la suivante.Ici, les probabilités de transitions sont calculées par expérimentation. Il faut, par contre, que $p_{ij} \geq 0 \; \forall i, j$ et que $\sum_j p_{ij} = 1 \; \forall i$.

2.6.2.2 Classification des états

Si le planteur fait des erreurs, c'est donc dire qu'une case peut n'avoir aucune graine, voir une ou deux. Donc, en allant d'une case à l'autre, pour un planteur imparfait, il est possible d'observer n'importe quels des trois états. La chaîne de Markov est donc qualifiée d'*irréductible* [20]. Dans le cas où le planteur serait parfait, ce qualificatif ne tiendrait plus. Lorsqu'une chaîne est irréductible, ses états sont dits *récurrents*. C'est-à-dire, qu'il est certain que tous les états finiront par être observés à un moment ou un autre. Autrement, les états seraient qualifiés de transitoire. Cela ne risque pas d'arriver, car tout appareil physique est immanquablement imparfait. Finalement, une chaîne est homogène si l'état ne dépend pas de son emplacement.

2.6.2.3 Matrice de transition

Plus de détails s'impose concernant la matrice de transition. D'abord, pour calculer une probabilité de transition il faut conditionner. À savoir, $p_{ij} = \mathbb{P}\left(\{X_n = i\} \mid \{X_{n-1} = j\}\right)$. Donc,

$$p_{ij} = \frac{\mathbb{P}\left(\{X_n = j\} \cap \{X_{n-1} = i\}\right)}{\mathbb{P}\left(\{X_{n-1} = i\}\right)}$$

Aussi, il est possible de relier les différentes probabilités de transition en n étapes à celles à une étape. Par exemple, pouvoir répondre à la question : qu'elle la probabilité d'observer l'état 1 de la huitième cellule plus loin étant

donné que la cellule actuelle est dans l'état 0 ? En notation : $p_{01}^{(8)}$. Dès lors, les états transitoires au niveau n est donné par P^n ou :

$$p_{ij}^n = \mathbb{P}\left(\{X_n = j\} \mid \{X_0 = i\}\right)$$

ou encore, en vertu de Chapman–Kolmogorov, pour $0 < k < n$

$$p_{ij}^n = \sum_{s \in \Omega} p_{ik}^k p_{kj}^{n-k}$$

2.6.2.4 Utilisation des chaînes de Markov

Dans le cadre du projet, cet outil permettra, dans une première mesure, d'évaluer les probabilités de transition. Cela pourra donner une idée des diverses occurrences. En effet, un bon planteur devrait donner p_{11}^n grand pour tout n. Dans un deuxième temps, après une analyse statistique il sera possible de donner une estimation des problème pouvant survenir. Par exemple, en supposant que des données sont amassées et actualisées de plateau en plateau (raffinement des probabilités) et bien il sera possible d'estimer, comme exemple, la probabilité d'obtenir 2 cellules vides séparées par 4 cases et ainsi de suite.

2.6.3 Cartes de contrôle

Les cartes de contrôle sont des outils qui servent, comme leur nom l'indique, à contrôler et faire du monitoring d'un processus (industriel). Il est ainsi possible d'en retirer de l'information et d'en suivre l'évolution. Il existe plusieurs types de cartes, selon le contexte. Par exemple, pour les données continues (longueur, température, poids,...) les carte-$\overline{x}$, carte-$\overline{s}$ et carte-$\overline{R}$ sont utiles. Pour la détection de défectuosités (il y a ou il n'y a pas de défectuosités), les carte-p et carte-np sont intéressantes. Lorsque le nombre de défectuosités est important, ce sont les cartes c et u qui sont de misent. Toutes ces cartes sont utilisées lorsqu'il y a échantillonnage. Si un processus est évalué de façon

exhaustive, alors une carte de Shewhart individuelle [13] peut être utilisée. Finalement, dans ce qui va suivre, les formules des limites seront données sans démonstration. Elle sont basées sur des hypothèses statistiques de bases. Dans le cas des cartes c et u, elles sont formulées en considérant le processus comme étant de Poisson. L'heuristique utilisée est qu'un processus est en contrôle dans le cadre de $\pm 3\sigma$.

2.6.3.1 Cartes-c

Ici, **les échantillons sont de taille fixe**. Par exemple, 5 plateaux sont investigués à chaque heure à chaque jour de production. Donc, un échantillon, E, est constitué de 5 unités. La construction pour $k \in \mathbb{N} - \{0\}$ est la suivante [18] :

1. Pour chaque E_k, compter le nombre c de défectuosités dans chaque unité.
2. Puis, calculer la moyenne $\overline{c}$.
3. Ensuite, calculer la limite supérieure selon :$\mathrm{UCL}_c = \overline{c} + 3\sqrt{\overline{c}}$.
4. Finalement, calculer la limite inférieure selon :$\mathrm{LCL}_c = \overline{c} - 3\sqrt{\overline{c}}$.

Pour reprendre l'exemple précédent avec 1 unité équivalent 5 plateaux avec le Tableau 2.4 hypothétique suivant :

Échantillon	Unités	c
1	1	50
2	1	48
3	1	34
4	1	60
5	1	28

TABLE 2.4 – Échantillonnage.

Voici la carte correspondante sur la Figure 2.12 :

13. Traduit directement de : *Shewhart individuals control chart*

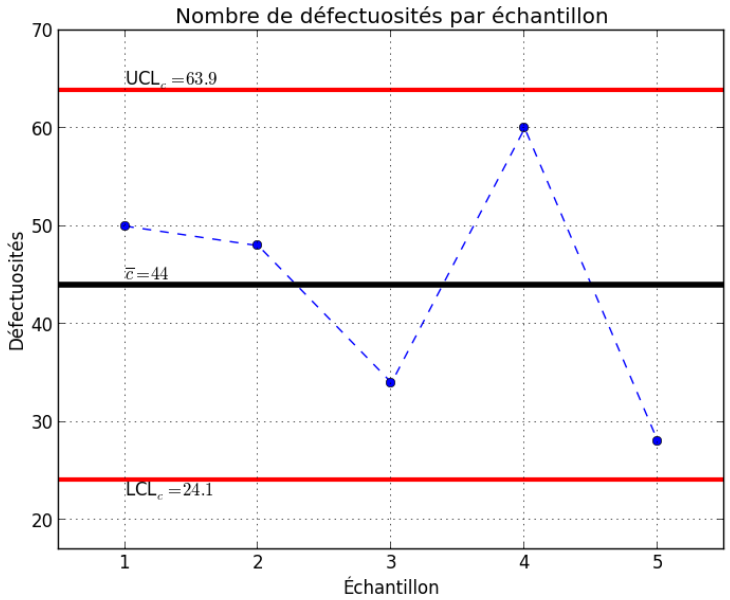


FIGURE 2.12 – Carte-c hypothétique.

2.6.3.2 Cartes-u

Comparativement aux cartes précédentes, les cartes-u tiennent compte d'échantillons de tailles différentes [3]. La démarche est celle-ci :

1. Pour chaque E_k ayant d défectuosités, compter $u = \frac{d}{|E_k|}$.
2. Puis, calculer la moyenne $\overline{u}$.
3. Ensuite, calculer la limite supérieure selon :$\text{UCL}_u = \overline{u} + 3\sqrt{\frac{\overline{u}}{|E_k|}}$.
4. Finalement, calculer la limite inférieure selon :$\text{LCL}_u = \overline{\overline{u}} - 3\sqrt{\frac{u}{|E_k|}}$.

Soit l'exemple suivant : à chaque heure de production des plateaux sont choisis et le nombre d'erreurs est compté. L'opérateur choisi un nombre arbitraire de plateaux selon sa disponibilité. Ici, l'unité correspond au nombre de plateaux choisis. Toujours hypothétiquement, selon le Tableau 2.5 :

Échantillon	Unités	Défectuosités	u	UCL_u	LCL_u
1	5	44	8,8	15,2	6.39
2	2	16	8	17,7	3.8
3	8	75	9.4	14,3	7.3
4	6	47	7.8	14,8	6.8
5	4	80	20	15,7	5.9

TABLE 2.5 – Échantillonnage à tailles variables.

La carte correspondante sur la Figure 2.13 :

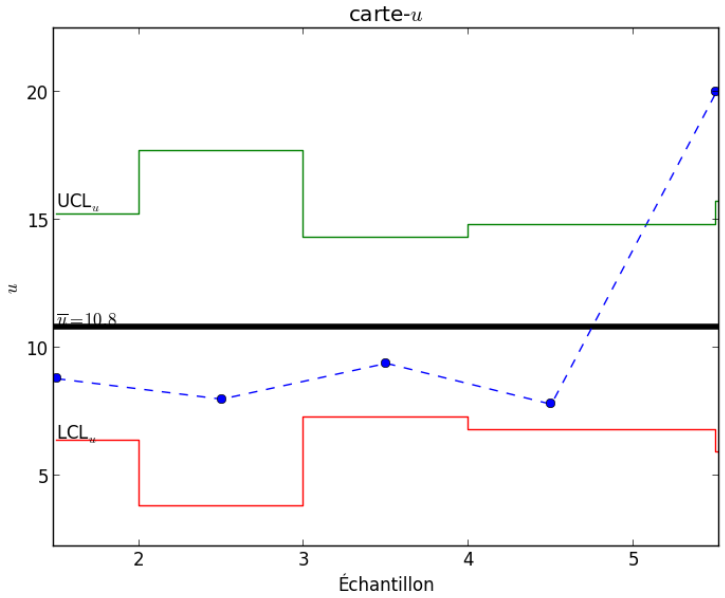


FIGURE 2.13 – Carte-u hypothétique.

2.6.3.2.1 Remarques

Il faut remarquer que ces cartes tiennent compte du nombre de défectuosités, mais elles ne permettent pas de discriminer les types possibles de défectuosités lorsqu'une erreur peut se manifester sous différentes formes. Par exemple, dans le cadre du projet, le fait qu'il n'y ait pas de graine dans une cellule ou plusieurs est en-soi une erreur. Par contre, le premier état est différent du

second. Ainsi, pour plus de raffinement, le monitoring doit se faire sous deux tableaux différents. C'est-à-dire, une carte pour chaque type. Finalement, dans le cadre du comptage du nombre de défectuosités **la limite inférieure pourrait très bien être omise ou égalée à 0**. En effet, quoique très peu probable, le fait de ne pas observer de défectuosités est en soi une bonne chose.

2.7 Conclusion partielle

Il fut possible, grâce à ce chapitre, de donner ou rappeler les bases en optique (réflexions), en traitement d'images (morphologie mathématique, moments statistiques, facteurs de formes,...) et en statistique qui permettront d'aborder les détails du système de vision au chapitre suivant. En effet, le prochain chapitre applique concrètement les notions abordées précédemment.

CHAPITRE 3

MÉTHODE EXPÉRIMENTALE

> Les machines un jour pourront
> résoudre tous les problèmes,
> mais jamais aucune d'entre elles
> ne pourra en poser un !
>
> Albert Einstein

3.1 Introduction

Ce chapitre est dédié au projet même. Il sera possible de voir en détail en quoi ce dernier consiste et comment les notions du chapitre précédent s'intègrent à celui-ci. Dans un premier temps, une vue d'ensemble sera donnée. Puis, l'ensemble du projet sera décortiqué en sous-ensembles détaillés permettant ainsi une compréhension complète du système.

3.2 Méthode de travail d'origine

À l'origine, avant l'implémentation du système de vision, l'espace de travail consistait en un convoyeur sur lequel est situé un planteur pneumatique. Les plateaux de cellules défilent tour-à-tour sur le convoyeur jusqu'au planteur (voir la Figure 3.1). Ce dernier injecte des semences à raison de 4 rangées de 12 cellules chacune à la fois jusqu'à complétion du plateau. Une fois le plateau rempli, un préposé à l'assurance qualité vérifie chaque cellule en s'assurant que chacune ne contienne qu'une et une seule semence. Si tel n'est pas le cas, une correction s'effectue (enlever ou ajouter des semences).

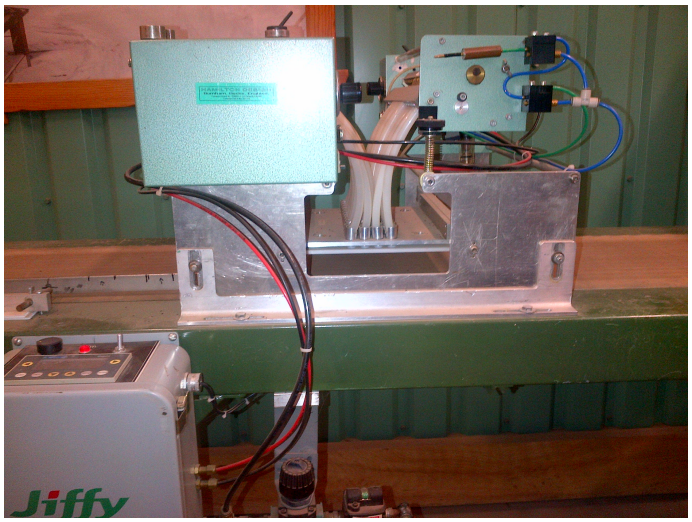
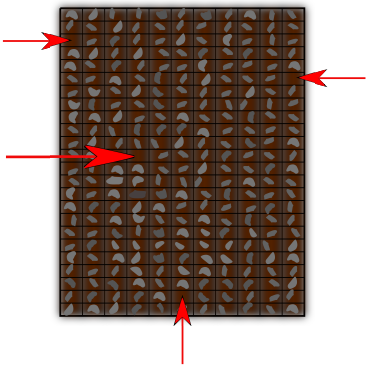


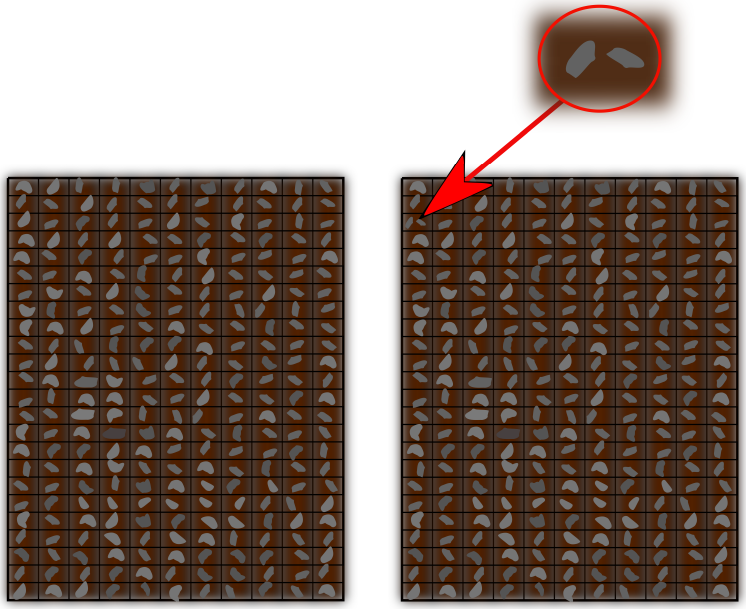
FIGURE 3.1 – Vue latéral du planteur pneumatique.

3.3 Erreurs à corriger

Le planteur étant imparfait, il peut injecter tantôt aucune, tantôt trop de semence dans une cellule donnée. Ce qui, tel que d'écrit dans l'introduction, pose problème. Ainsi, il est possible de classer l'état d'une cellule en trois grands cas d'espèces schématisés sur la Figure 3.2 :



(a) Cas **I** : sans graine



(b) Cas **II** : une graine

(c) Cas **III** : plus d'une graine

FIGURE 3.2 – Trois cas possibles de l'état d'une cellule

3.4 Objectif du système de vision

Le but principal du système est de détecter les erreurs et de diriger rapidement et facilement l'opérateur afin de corriger la situation. Donc, il permet, en théorie, de rendre le travail plus aisé en sollicitant moins les yeux et permettant une concentration moins soutenue chez les opérateurs à l'assurance qualité. Finalement, un seul opérateur à l'assurance qualité peut suffire à corriger les plateaux pris en défaut. Économiquement, le système a ainsi le potentiel d'être intéressant en redirigeant des gens compétents vers d'autres tâches plus utiles.

3.5 Système de capture

Cette section sert à décrire le système physique dans son ensemble. C'est-à-dire, les grandes pièces d'équipement qui le constituent tout en explicitant leur rôle respectif. Il peut être utile de rappeler que le système se greffe au convoyeur/planteur.

3.5.1 Emplacement du système de vision

Il peut s'avérer intéressant de glisser quelques mots au sujet de l'emplacement ou plutôt des emplacements du système de détection. Ce dernier est en fait mobile, car il est situé dans une remorque (voir Figure 3.3). Il est donc possible de déplacer ce dernier aux endroits appropriés durant les semences et ce, à l'abri des intempéries.

(a) Vue extérieure de la remorque

(b) Vue intérieure de la remorque

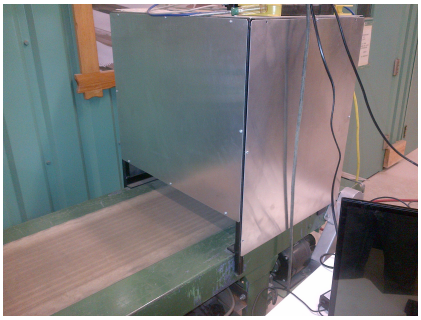
FIGURE 3.3 – Le système et la remorque.

3.5.2 Équipement

C'est ici que sont décrit les principaux éléments du système.

3.5.2.1 Caisson métallique

Le caisson métallique est situé un peu après le planteur pneumatique au bout du convoyeur. Les plateaux entrent graduellement dans ce dernier et sont analysés par la système en temps-réel. Le caisson permet d'éliminer les sources de lumière extérieures et contient la caméra. Il est possible de voir se dernier à la Figure 3.4



(a) Le caisson métallique

(b) Un plateau situé dans le caisson

FIGURE 3.4 – Le caisson métallique.

3.5.2.2 Caméra

La caméra (*grasshopper de Point Grey*) capte dans le proche infrarouge et est connectée à l'ordinateur via un fil USB 3.0. De plus, à sa lentille est fixé un anneau rempli de diodes électroluminescentes qui éclairent le plateau. Voici la caméra à la Figure 3.5. Les spécifications sont :

Modèle : Grasshopper3 GS3-U32854M .
Vendeur : Point Grey Research.
Résolution : 1928×1448.
Senseur : Sony ICX657AL (1/1.8" 1928×1448 CCD).

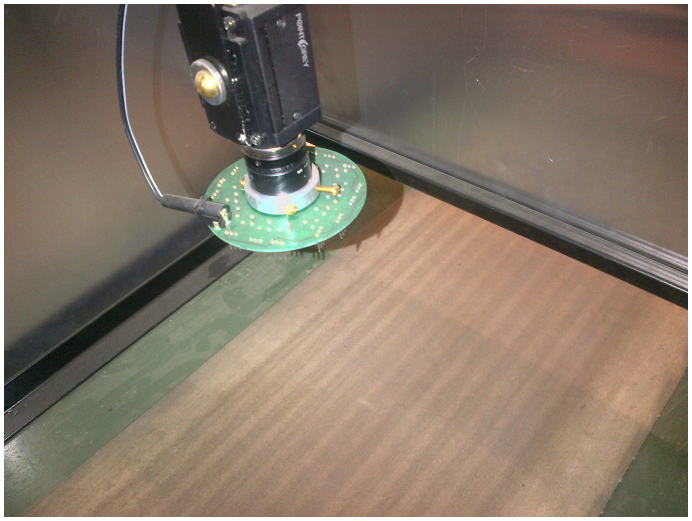


FIGURE 3.5 – Vue de haut de la caméra

3.5.2.3 Projecteur

Une fois un plateau analysé, chaque cellule est attribuée un code de couleur selon l'état de chacune (par exemple, rouge ⇒ cas **I** : absence de graine). Donc, lorsque le plateau sort du caisson métallique, le projecteur (Figure 3.7) projette sur chaque cellule le code de couleurs déterminé et permet, ainsi, de rapidement faire savoir au préposé à l'assurance qualité qu'elle action entreprendre.

FIGURE 3.6 – Le projecteur.

Au moment de cet écrit les codes de couleurs sont les suivants :

- Bleu ⇒ 1 graine.
- Rouge ⇒ 0 graine.
- jaune ⇒ > 2 graines.

Voici un exemple schématisé, sur la Figure 3.7 , du projecteur projetant les résultats finaux de l'analyse d'un plateau.

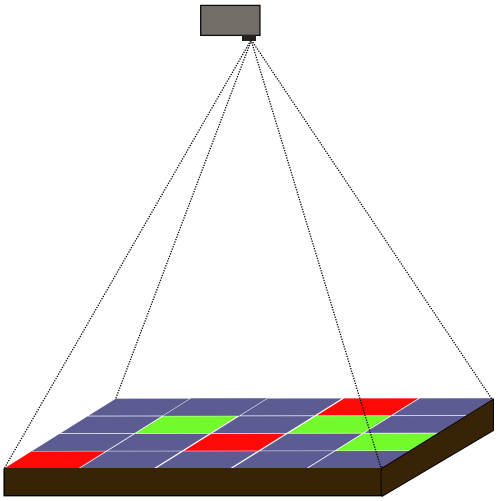


FIGURE 3.7 – Projection de l'analyse d'un plateau.

3.5.2.4 Ordinateur

L'ordinateur (équipé avec un écran tactile) est conventionnel, mais suffisamment rapide pour permettre les calculs en temps-réel. Plus spécifiquement :

Processeur : Intel(R) Core(TM) i5-4570 cpu @ 3.20 [GHz].
Mémoire RAM : 16.0 [Go].
Type de système : système d'exploitation 64 bits , processeur x64.
Système d'exploitation : Windows 8.1 .

3.5.3 Organigramme du processus de détection

Il peut-être utile maintenant de rappeler de quelle façon se déroule l'ensemencement et la détection sur le convoyeur.

Quelques plateaux vides de graines (de deux à trois) sont déposés sur une des extrémités du convoyeur. Il faut noter qu'un couvercle troué au niveau de chaque cellule (voir Figure 2.2) est mis sur ces derniers. Celui-ci permet aux graines de tomber, autant que faire se peut, au milieu de

chaque cellule. Ensuite, le plateau passe sous le planteur qui injecte des semences à raison de 4 rangées à la fois (donc, le planteur s'actionne 6 fois par plateau). Plus précisément, le convoyeur s'arrête une fois que les 4 premières rangées du plateau sont alignées avec le planteur. Le tapis du convoyeur se remet en branle, puis s'arrête de nouveau lorsque les 4 rangées suivantes sont alignées avec le planteur et ainsi de suite. Il y a donc un court laps de temps durant lequel les plateaux restent immobiles. Une fois que le plateau est complètement passé sous le planteur, ce dernier fait son entrée dans le caisson métallique. Le système détecte ensuite le plateau (cela sera détaillé plus loin) et traite l'image des 4 premières rangées de cellules lorsque le tapis s'arrête. Il en est de même pour les 4 prochaines rangées et ce, jusqu'à ce que les 24 rangées soient traitées. Les sous-images sont concaténées afin de former une image finale reconstituée du plateau complet. Pour finir, le plateau sort du caisson métallique et est déposé sous le projeteur. L'analyse est ainsi projetée sous forme de code de couleur sur le plateau : chaque cellule du plateau est teintée de la couleur témoignant de son état (vide de graine ou possédant une seule graine ou ayant plus d'une graine). Le schéma de la Figure 3.8 donne une idée générale de la durée de vie du plateau sur le convoyeur :

3.6 Programmation

Cette partie détaille la façon dont fonctionne le programme de détection. Évidemment, il ne s'agit pas de décrire ligne par ligne le code utilisé, mais plutôt de donner les étapes principales et d'illustrer ces dernières au moyen d'exemples concrets.

3.6.1 Critères sélectifs

Tel qu'il sera décrit plus loin, plusieurs critères doivent être utilisés afin de déterminer si un contour détecté dans l'image d'une section de plateau de cellules est bel et bien celui d'une graine de résineux. En effet, une surabondance de contours sont détectés (bruit et objets autres que des graines) et

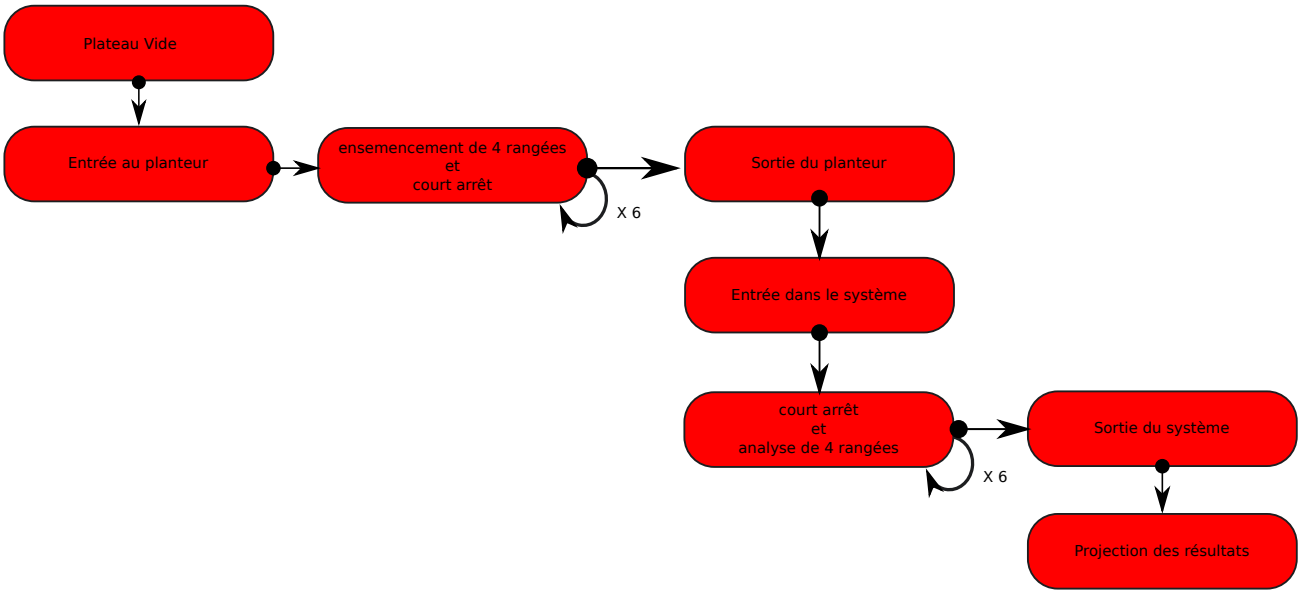


FIGURE 3.8 – Schéma : arrêt synchronisé sur les blocs de 4X12 cellules des plateaux sous le planteur.

un filtrage s'impose. Les principaux critères calculés à partir des contours sont : l'aire, le périmètre, facteur de forme, la solidité et l'excentricité. Il apparaît important de trouver des seuils permettant de n'extirper que les bons contours, du moins, dans une proportion acceptable. Une des difficultés de trouver un seuil correct est le fait qu'un graine peux atterrir de différentes manières dans une cellule, car en fait, la cellule possède une cavité. Ainsi, l'aire perçue peut varier pour une même graine. Il ne peut en être autrement, car la graine physique est de dimensions 3, alors que la caméra capture une image de dimension 2. L'angle sous lequel tombe une graine est, manifestement, aléatoire. Quoiqu'en toute vraisemblance, il est à priori plus probable qu'une graine tombe au milieu de la cellule à cause du couvercle utilisé de pair avec le planteur(voir Figure 3.9).

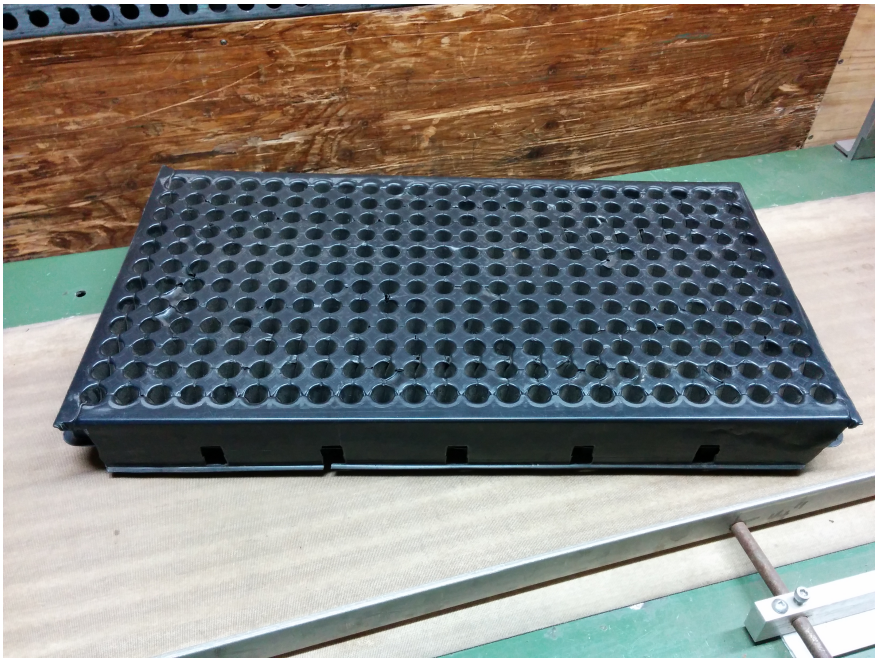


FIGURE 3.9 – Couvercle utilisé avec le planteur.

En effet, ce dernier est déposé sur un plateau avant de passer sous le planteur. Le couvercle sert à canaliser, diriger la trajectoire de la graine qui sort du planteur. Cela est dû au fait que la surface de la cellule vue du planteur est réduite au trou du couvercle qui est, manifestement, plus petit que la surface de la cellule. Le schéma présenté sur la Figure 3.10 illustre la problématique des mesures observées par le système comparativement aux mesures réelles dans le monde 3D.

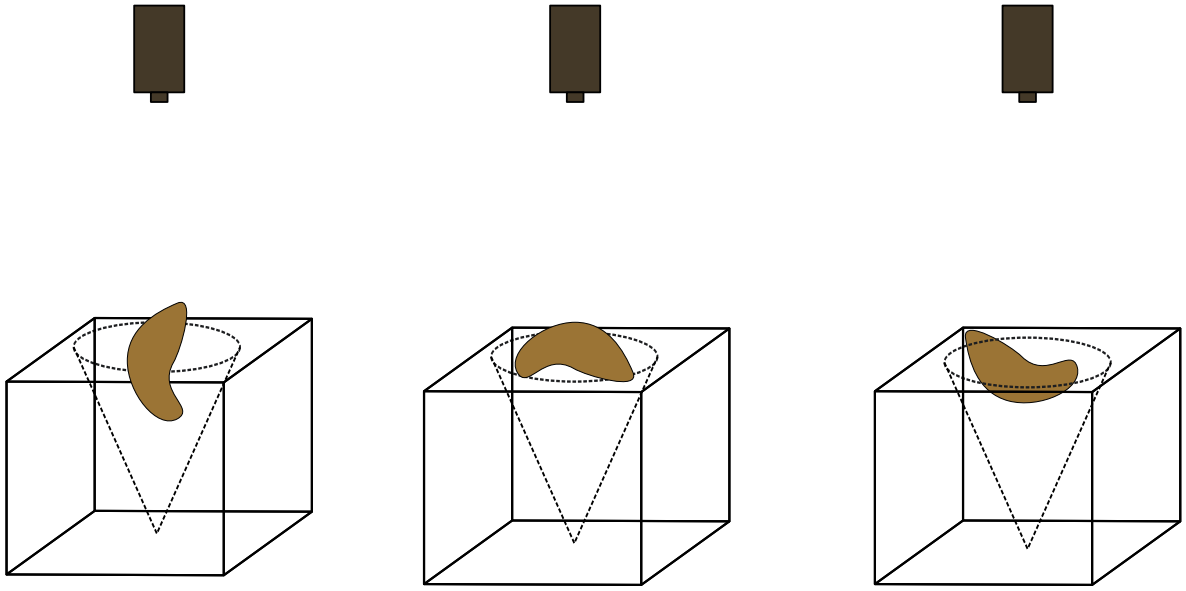


FIGURE 3.10 – Une même semence, une même cellule, des aires différentes.

Ainsi, plusieurs mesures sont prises afin d'en trouver des valeurs acceptables pouvant aider à déduire la validité d'un contour. En ce moment, les statistiques sont calculées en découpant manuellement les contours voulus sur des images capturées par le système. De là, il est certain que le contour trouvé est le bon. Donc, dans un dossier constitué d'images de semences, les contours et les statistiques associées sont amassées.

```
1
2  ...
3
4  # on boucle sur toutes les images
5
6  for image in imageNames:
7
8      currentImg = cv2.imread(image,0)
9
10     # Filtre Gaussien + methode d'Otsu
11
12     blurImg = cv2.GaussianBlur(currentImg,(5,5),0)
13     ret, otsuImg = cv2.threshold(blurImg,0,255,
14                                  cv2.THRESH_BINARY+cv2.THRESH_OTSU)
15     # Morphologie
```

```
16
17      kernel = np.ones((2,2),np.uint8)
18      otsuImg = cv2.morphologyEx(otsuImg, cv2.MORPH_OPEN, kernel)
19      otsuImg = cv2.morphologyEx(otsuImg, cv2.MORPH_CLOSE, kernel)
20
21      # Detection du contour de la graine
22
23      contours, hierarchy = cv2.findContours(otsuImg,
24                  cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
25
26      if len(contours) == 1 :
27
28      # Les moments
29          # aire
30
31          area = cv2.contourArea(contours[0])
32
33          # perimetre
34
35          perimeter = cv2.arcLength(contours[0],True)
36
37          # facteur de forme
38
39          circularity =  4* math.pi *(area/(perimeter*perimeter))
40
41          # solidite
42
43          hull = cv2.convexHull(contours[0])
44          hullArea = cv2.contourArea(hull)
45          solidity = float(area)/hullArea
46
47          # excentricite
48
49          (x,y) = cv2.fitEllipse(contours[0])[0]
50          center = (int(x),int(y))
51          (axes) = cv2.fitEllipse(contours[0])[1]
52          majorAxis = max(axes)*2
53          minorAxis = min(axes)*2
54          eccentricity = np.sqrt(1-(minorAxis/majorAxis)**2)
55
56  ...
```

Par exemple, une image typique utilisée est montrée à la Figure 3.11.

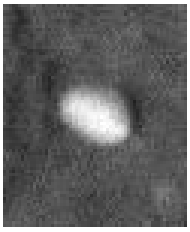


FIGURE 3.11 – Image du contour de la graine.

Les critères calculés sont données au Tableau 3.1 :

Périmètre	Aire	Facteur de forme	Solidité	Excentricité
60.9	243.5	0.826	0.978	0.790

TABLE 3.1 – Critères calculés.

3.6.2 Fonctionnement du système de vision

Il est maintenant temps de décrire comment la détection s'effectue. Dans un autre ordre d'idées, sans définition formelle, il est possible d'admettre qu'un flux vidéo est une collection d'images chacune étiquetées dans le temps. Une notation plausible [1] pourrait être : $\mathcal{V}(x, y, t)$ en considérant, évidemment, que la vidéo est en niveaux de gris (un seul canal). Ainsi, une image analysée au temps t_0 par le système est une instance de la vidéo $\mathcal{V}(x, y, t_0) = I_{t_0}(x, y) \in \mathbb{M}_{1928\times1448}(\mathbb{R})$ qui n'est rien d'autre qu'une matrice rectangulaire 1928×1448 du point de vue interne de l'ordinateur.

Cela établit, il est dorénavant permis de donner le flot de contrôle de la détection.

Détection de plateau

La caméra capte en continu, par contre, il n'y a pas de plateau nécessairement visible à tout moment. Lorsqu'il y a absence de plateau c'est la courroie du convoyeur qui est visible comme le montre la Figure 3.12.

1. Aucun critère mathématique ne sera donné ici sur la nature de la fonction. Elle sera assumée $\mathcal{V} : \mathbb{R}^3 \to \mathbb{R}$, par commodité sachant qu'il s'agit plutôt de nombres dans $\mathbb{N}$.

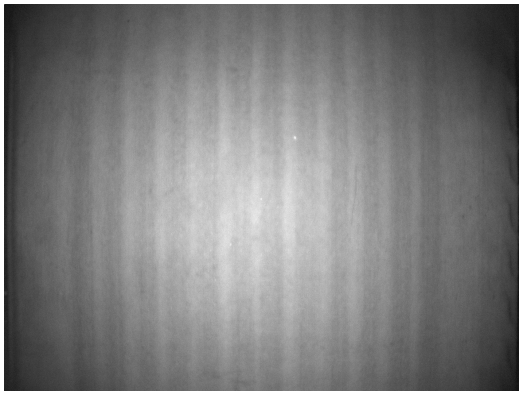


FIGURE 3.12 – La courroie du convoyeur.

La courroie possède la même texture sur tout son long. Dès lors, afin de déterminer si il y a présence de plateau ou non le système calcule la différence $\Delta := |I_t - I_{\text{stat}}|$ ayant I_{stat} représentant l'image statique de la courroie . Typiquement, le début de plateau apparaît ainsi à la caméra (Figure 3.13).

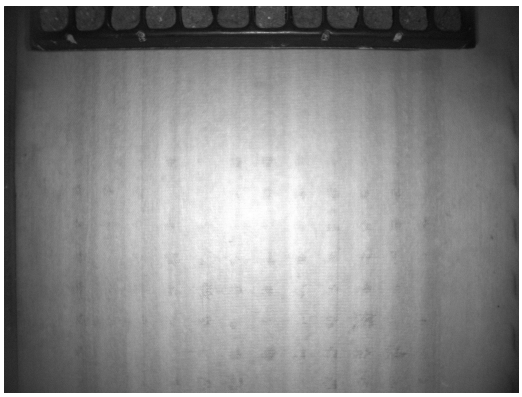


FIGURE 3.13 – Un début de plateau.

Si $\Delta \approx 0$ (matrice nulle, image noire), le système considère qu'il n'y a pas de plateau et ne fait, par conséquent, aucune analyse. Par contre, si $\Delta \neq 0$, alors [2] (voir Figure 3.14) il y a présence du plateau avec une forte probabilité.

2. Dans la pratique, le critère est plutôt $\Delta >$ seuil à cause du bruit et de la représentation interne des nombres.

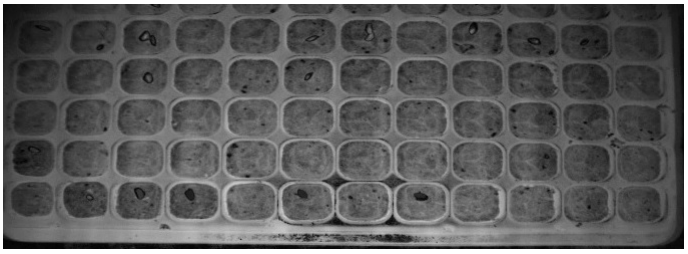


FIGURE 3.14 – $\Delta = |I_t - I_{\text{stat}}| \neq 0$.

Par exemple, l'histogramme $h(p)$ associé à la différence illustrée à la Figure 3.14 est donné à la Figure 3.15.

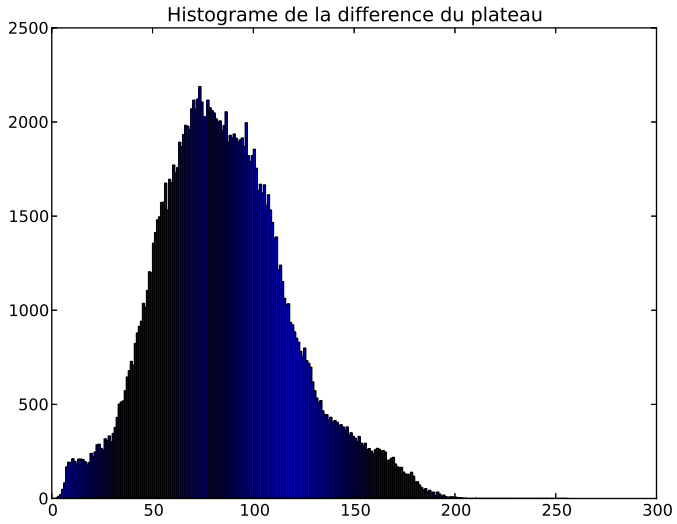


FIGURE 3.15 – Histogramme de Δ.

Les images de différences sont seuillées et les contours sont recherchés. Le plus grand contour est, avec un haut niveau de confiance, le plateau lui-même. Plus précisément, les bordures de celui-ci. Un contour est irrégulier. Il existe une fonction OpenCV qui prend en paramètre un contour (liste de points) et qui retourne le plus petit rectangle contenant tous les points et retourne les coordonnées du coin supérieur gauche ainsi que la largeur et hauteur du rectangle englobant $x, y, w, h \in \mathbb{N}$. À la Figure 3.16, il est possible de voir le contour irrégulier du plateau ainsi que son rectangle englobant.

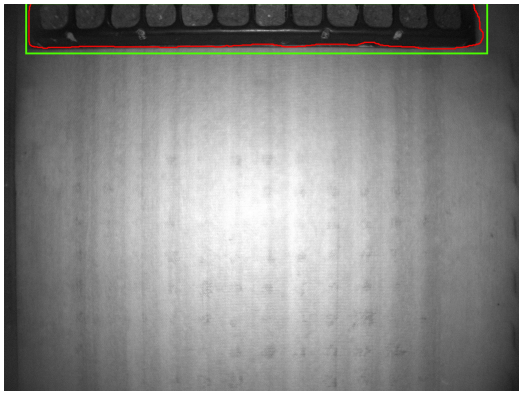


FIGURE 3.16 – Le contour du plateau et le rectangle englobant.

Tel que mentionné ultérieurement, la plateau est analysé 4 rangées à la fois. Il doit donc y avoir au moins 4 rangées visibles. Cette condition est atteinte assurément lorsque le plateau est au moins aussi haut que 4 rangées de haut soit ($H[px]$). En ayant les coordonnées de la boite englobante il devient donc facile de savoir quand le plateau rempli la condition précédente. L'analyse n'est effectuée que lorsque le convoyeur devient immobile temporairement. Il s'agit donc de détecter le temps de repos τ (typiquement, $\tau \approx 1[s]$). Pour ce faire, la tactique choisie est encore de faire une différence non pas avec une image de référence, mais bien avec deux images adjacentes dans le temps. Un peu plus formellement, $\Delta_\tau := |I_{t_{i+1}} - I_{t_i}| \quad \text{ayant } t_i < t_{i+1} \quad \forall i \in \mathbb{N}$. Alors, à quelques pixels bruités près, le temps de repos s'observe lorsque $\Delta_\tau \approx 0$. C'est à ce moment que peut s'effectuer le début d'analyse. Schématiquement, la première phase de détection peut se résumer ainsi (voir Figure 3.17) :

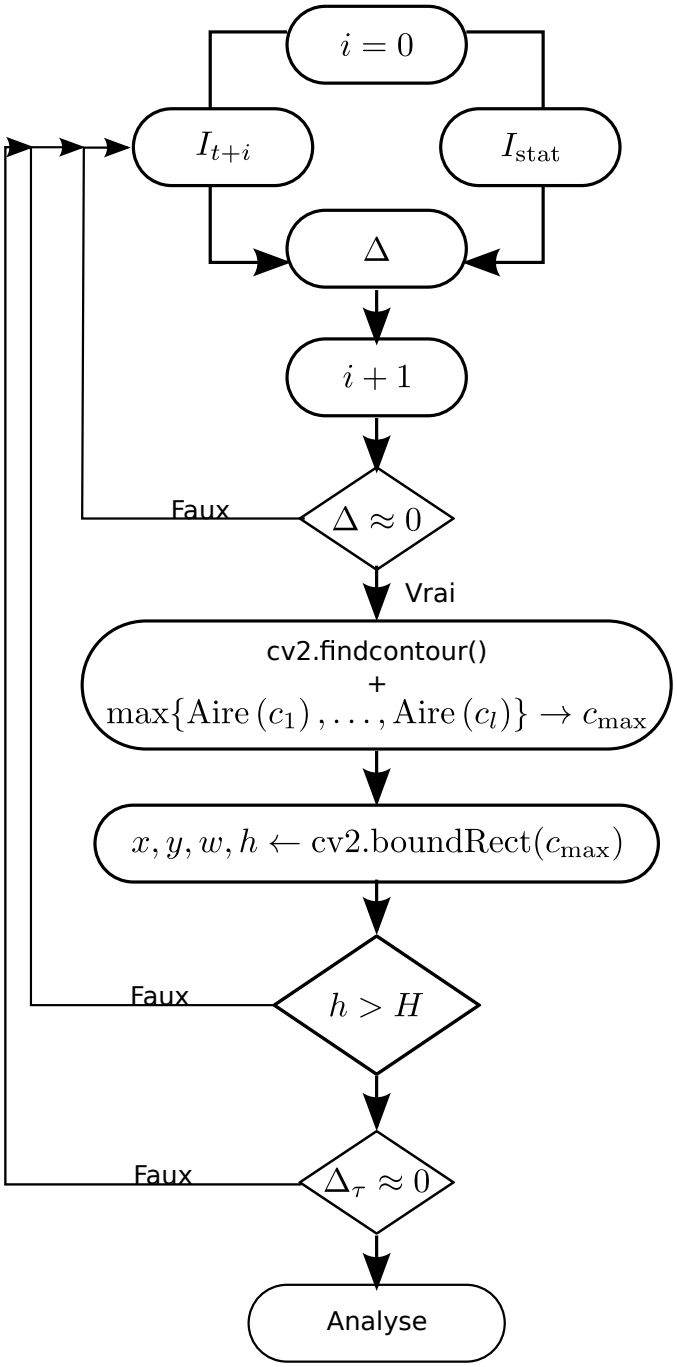


FIGURE 3.17 – Organigramme de programmation simplifié.

Analyse de plateau

En première instance, ici, une partie du plateau est donc visible à la caméra et l'image est au beau fixe (c'est le temps de repos τ). Une grille 4×12 est juxtaposée sur le plateau et chacune des 48 cases de cette dernière s'aligne sur une et une seule cellule du plateau. Cela constitue donc 48 **ROI** (*region(s) of interest*) qui serviront à investiguer l'état de la cellule correspondante. Pour alléger les notations, la grille virtuelle G peut se définir simplement comme $G = \{(n, m) \mid n \in \{0, 1, \ldots, 11\} \text{ et } m \in \{0, 1, \ldots, 3\}\}$ si l'origine du référentiel coïncide avec celui de la grille. Autrement, dit, le point $(0, 0)$ est le coin supérieur gauche de la première région d'intérêt. À titre de rappel, cela ne reflète pas l'implémentation, mais est tout de même équivalent du point de vue mathématique. Donc, $\text{ROI}_j := \mathcal{R}_j \subset G$ sachant que $j \in \{0, 1, \ldots, 47\}$ correspond à une des 48 cases de la grille.

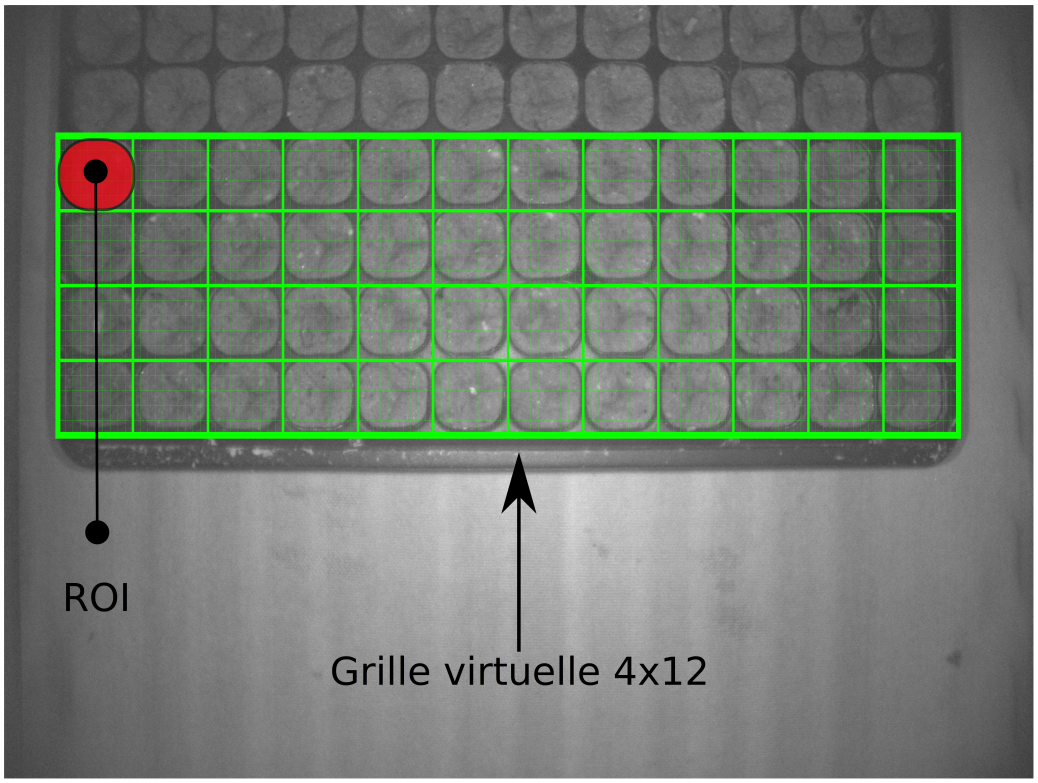


FIGURE 3.18 – La grille 4X12

Chaque cellule est traitée comme suit [3](traitement ROI) :

- Seuillage de la région $\mathbb{T}_s(\mathcal{R})$ avec [4] $s = \mu + 2\sigma$ ($s \equiv$ seuil)
- Détection de l'ensemble des contours.
- L'aire et la solidité de ces derniers sont testés avec les valeurs trouvées grâce aux statistiques : $A_{\min}, A_{\max}, s$. Typiquement, par exemple, si $s < 0.95$ le contour est rejeté.
- Le nombre n_c de contour(s) gardé(s) est calculé.

  - Si $n_c = 0$, alors le stade de la cellule est incorrect et le code rouge est associé.
  - Si $n_c = 1$, alors le stade de la cellule est correct et le code bleu est associé à celle-ci.
  - Si $n_c > 1$, alors le code jaune est associé.

Par la suite, l'image des 4 rangées est découpée et le nombre de cellules traitées y est affiché ainsi que le nombre de cellules fautives et adéquates. Puis, l'image est stockée en mémoire. Tout cela, est fait au premier temps de repos τ_1. Il y a donc 5 autres temps de repos pour un même plateau. Ainsi, Δ_τ est calculé et lorsque qu'un autre temps de repos est déterminé, les 4 rangées suivantes sont traitées et le processus recommence jusqu'à ce que tout le plateau soit traité. À chaque traitement l'image des 4 rangées du plateau est mémorisée pour être finalement concaténée à la fin pour former l'image finale. Voici une synthèse sur la Figure 3.19.

3. Pour alléger les notations, supposons un opérateur $\mathbb{T}_s : \mathcal{R} \to \mathcal{R}^B \in \{0, 1\}$ qui prend une région en niveau de gris et qui fait une région binaire selon un seuil s.

4. $\mu = \sum_{p \in \mathcal{R}} \frac{p}{\dim(\mathcal{R})}$ où $p \in [0, 255]$; $\sigma = \sum_{p \in \mathcal{R}} \frac{(p-\mu)^2}{\dim(\mathcal{R})}$

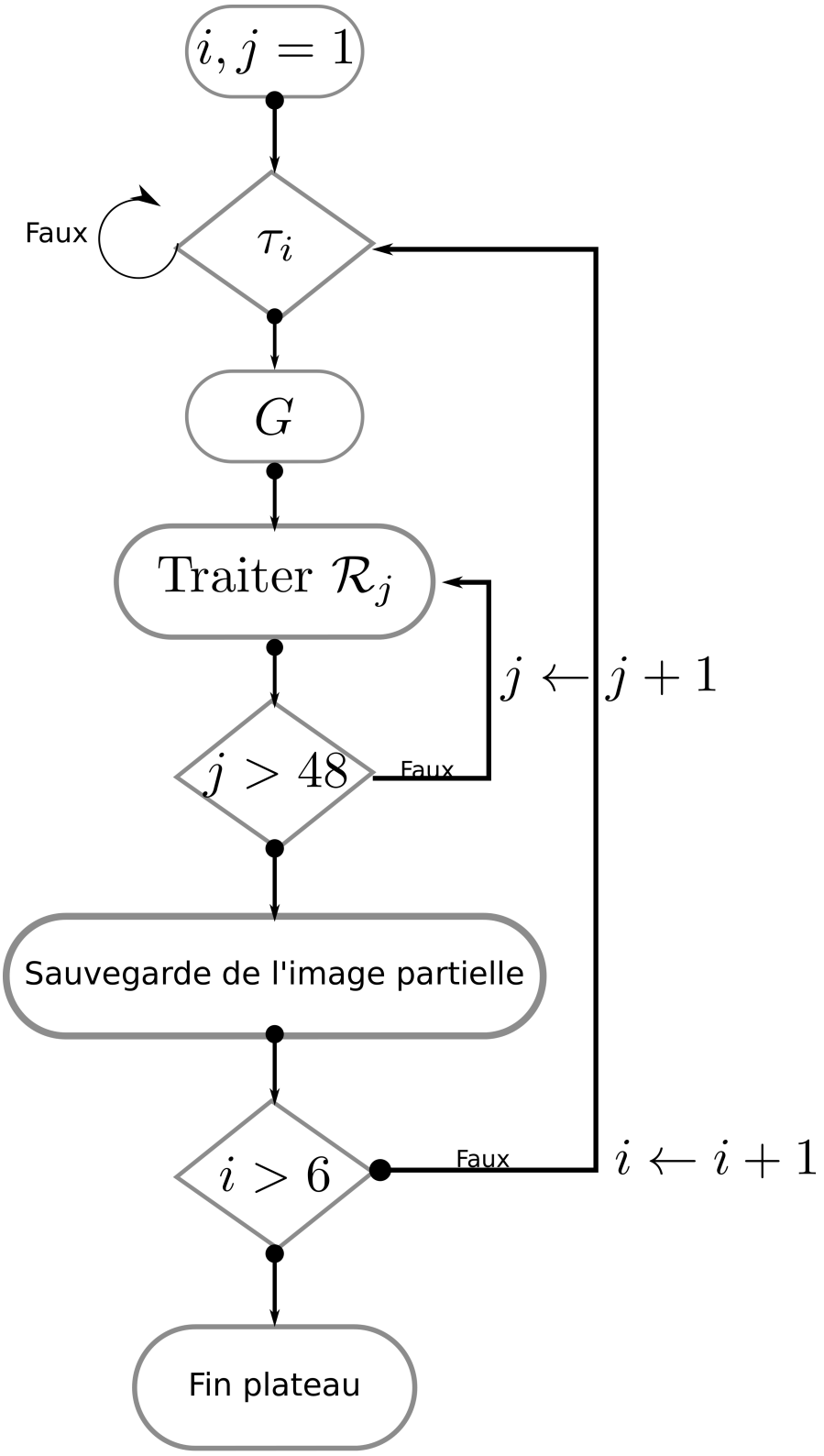
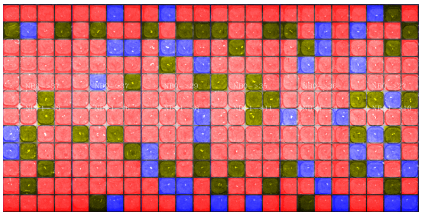


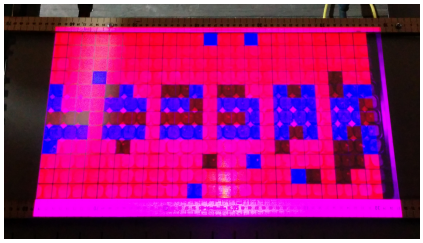
FIGURE 3.19 – Organigramme du traitement d'un plateau.

Projection de l'analyse

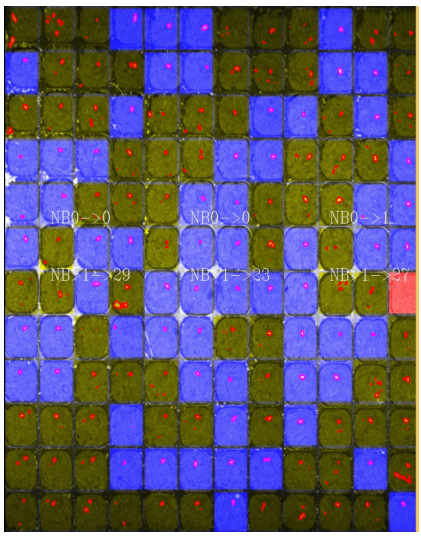
Un fois un plateau traité, les données sont enregistrées dans une petite base de données. Les 6 images partielles des cellules (avec code de couleur) sont combinées pour former une image complète et reconstituée du plateau qui s'affiche à l'écran. Cette dernière est projetée directement sur la table dédiée à l'assurance qualité suivant le caisson métallique. Le tout est illustré à la Figure 3.20 :



(a) Résultats affichés sur l'écran



(b) Projection du plateau analysé



(c) Résultats : graines en rouge

FIGURE 3.20 – Affichage des résultats

L'opérateur glisse le plateau de sorte que la projection coïncide avec le plateau réel. Il devient facile, par la suite, de corriger rapidement les cellules ayant un stade insatisfaisant.

3.7 Contrôle qualité

Tout ce qui attrait à la théorie sur le contrôle statistique sera appliqué dans le chapitre suivant. En effet, le modèle probabiliste et les cartes de contrôle seront confrontés aux données expérimentales. Au moment de cet écrit, aucune routine statistique n'a encore été écrite pour le système. Évidemment, cela se fera éventuellement afin de solidifier ce dernier.

3.8 Conclusion partielle

Ce chapitre a permis de détailler le projet sous plusieurs angles. La façon dont procèdent les travailleurs autour du convoyeur a été élaborée afin d'avoir une idée concrète de son utilisation. Le système de vision a été décrit ainsi que ces principaux composants. Puis, les principales étapes que subit un plateau ont été énumérées. Ensuite, la manière dont est analysée les plateaux du point de vue de la machine a été détaillée. Certaines problématiques ont également été explicitées. Finalement, les notions du chapitre antérieur ont pu être mises à profit soit de façon explicite ou implicite. Étant donné tout cela, il est raisonnable de penser que le lecteur a une bonne idée du projet.

CHAPITRE 4

RÉSULTATS ET DISCUSSION

4.1 Introduction

Cette section permet de mesurer la performance du système dans des conditions réelles d'utilisation. Il est important de noter que le projet est, au moment de cet écrit, à sa première phase. Donc, il apparaît que beaucoup d'ajustements sont nécessaires à sa future utilisation par le client. Le test utilisé sera décrit ainsi que les résultats qui en découle. Puis, une discussion sur les améliorations possibles sera faite.

4.2 Test

Le test en-soi est très simple et représentatif de l'utilisation réelle prévue. En effet, 4 plateaux semés furent envoyés au système de vision. Une fois analysés, les résultats furent enregistrés dans une base de données ($\mathcal{BD}$). Ensuite, les données stockées sont comparées aux données déterminées manuellement. Plus précisément, l'état des cellules de chaque plateau a été déterminé par un observateur (le nombre de graines dans chaque cellule a été compté). Ces valeurs seront considérées comme théoriques et justes. Ce jeu de données est, par la suite, comparé à celui engendré par le système. Finalement, le type

de semence utilisé est l'*épinette blanche* qui, faut-il le rappeler, n'est pas le type de semence le plus facile à détecter.

4.2.1 Résultats de la détection

Les critères de discrimination utilisé est indiqué dans la Table 4.1.

Épinette blanche		
s	A_{min} [px^2]	A_{max} [px^2]
$9.49999988 \times 10^{-1}$	50	100

TABLE 4.1 – Critères utilisés.

Les valeurs théoriques et expérimentales de la $\mathcal{BD}$ sont données (partiellement pour ne pas alourdir inutilement) dans la Table 4.2.

Valeurs théoriques		
celluleID	# graine(s)	PlateauID
1	0	**ID1**
2	0	**ID1**
3	0	**ID1**
4	0	**ID1**
5	5	**ID1**
...	...	...
1	1	ID2
2	1	ID2
3	2	ID2
4	1	ID2
5	2	ID2
...	...	...
1	1	**ID3**
2	2	**ID3**
3	1	**ID3**
4	1	**ID3**
5	1	**ID3**
...	...	...
1	0	ID4
2	0	ID4
3	1	ID4
4	0	ID4
5	1	ID4
...	...	...

Valeurs expérimentales		
celluleID	# graine(s)	PlateauID
1	1	**ID1**
2	3	**ID1**
3	1	**ID1**
4	0	**ID1**
5	1	**ID1**
...	...	...
1	1	ID2
2	1	ID2
3	1	ID2
4	1	ID2
5	3	ID2
...	...	...
1	1	**ID3**
2	0	**ID3**
3	1	**ID3**
4	0	**ID3**
5	0	**ID3**
...	...	...
1	1	ID4
2	2	ID4
3	1	ID4
4	0	ID4
5	3	ID4
...	...	...

TABLE 4.2 – Jeux de données.

Ainsi, globalement, le système est correct $\approx 30\%$ du temps.

Sous un angle différent, les erreurs peuvent être découplées en trois types tels que déjà discuté :

Type I : 0 graine détectée.
Type II : 1 graine détectée.
Type III : > 1 graines détectées.

Le classement apparaît sur la Figure 4.1

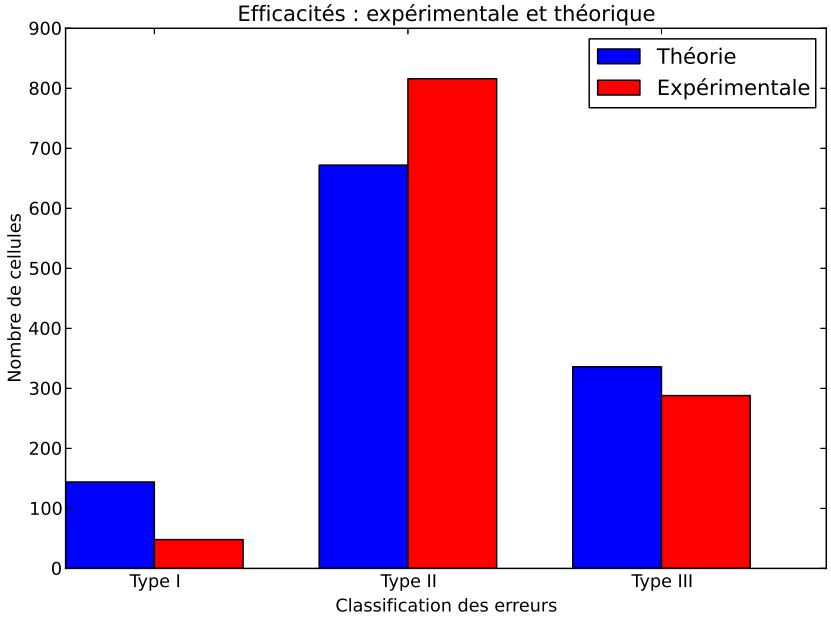


FIGURE 4.1 – Différence de classification entre l'observateur et le système

Pour calculer l'efficacité, il suffit de prendre le nombre de candidats dans chaque classe du jeu de donnée théoriques : n_I, n_{II}, n_{III}. Ensuite, la même chose est faite pour le jeu expérimental :$\widehat{n_I}, \widehat{n_{II}}, \widehat{n_{III}}$. Finalement, la simple formule est appliquée :

$$\text{eff} = \frac{|n_\chi - \widehat{n_\chi}|}{n_\chi} \cdot 100 \qquad [\%] \qquad \text{pour } \chi \in \{I, II, III\}$$

Les résultats sont visibles dans la Table 4.3

Efficacité de détection	
Type d'erreur	efficacité [%]
I	66
II	20
II	14

TABLE 4.3 – Performance selon la classe d'erreur

Il en résulte que les cellules ne présentant pas de graine sont mieux détectées.

4.2.2 Résultat du contrôle de qualité

Différents tests sont effectués afin de tester les différents outils statistiques.

4.2.2.1 Modèle probabiliste

L'échantillonnage se fait comme suit : des images de plateaux sont observées et chaque fois qu'une cellule ne représente pas d'ambiguïté, son état est noté et comptabilisé. En effet, même pour un humain, certains états son difficile à juger. Il y a eu 167 cellules d'observées. En somme, 17 ne contenait pas de graine, 18 en possédaient 2 et 137 en avait une. L'histogramme est donné sur la Figure 4.2 :

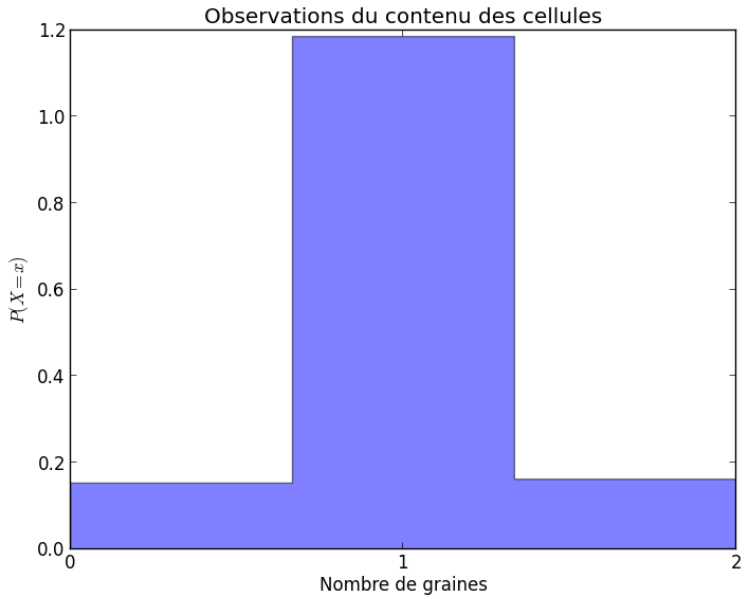


FIGURE 4.2 – Décompte expérimental du nombre de graines par cellule.

4.2.2.1.1 Méthode via maximum de vraisemblance

La méthode a été exposée dans la revue de la littérature. Voici quelques lignes de programme [1] qui permettent d'estimer p et q :

```
1  import numpy as np
2
3  n1= 17
4  n2= 132
5  n3= 18
6
7  step = 1000.0
8
9  """
10 Construction de la grille
11 """
12
13 p,q = np.meshgrid(np.linspace(0,1,step,endpoint=False),
14                    np.linspace(0,1,step,endpoint=False))
15
```

1. Le programme n'est en rien optimisé et n'est pas écrit avec rigueur. Cependant, il est court et permet une recherche de solution.

```
"""
Pour chaque point de la grille,
la fonction est appliquee
"""

L = np.power(p,n1)*np.power(q,n2)*np.power((1-p-q),n3)

"""
obtention du maximum
et verification des
contraintes. Tant qu
elles ne sont pas
remplies, la recherche
continue.
"""

pointMax = np.unravel_index(L.argmax(), L.shape)

while((pointMax[0]/step + pointMax[1]/step) > 1):
    L[pointMax[0]][pointMax[1]]=-1
    pointMax = np.unravel_index(L.argmax(), L.shape)

print "(q,p)=",pointMax
```

4.2.2.1.2 Méthode des moments

Très simplement, la moyenne est calculée ainsi que le carré des moyennes et les formules données dans la revue de littérature sont évaluées.

```
"""
echantillon est la liste des
observations :
echantillon =[1,2,1,1,0,1,...,1]
"""
In [59]: mu = np.mean(echantillon)


In [60]: p = (np.mean(np.power(echantillon,2))-3*mu+2)/2.0

In [61]: p
Out[61]: 0.10179640718562888

In [62]: q=2*mu-np.mean(np.power(echantillon,2))
```

```
15
16 In [63]: q
17 Out[63]: 0.7904191616766465
```

4.2.2.1.3 Comparaisons des méthodes

Voici un récapitulatif des résultats au Tableau 4.4 :

	f expérimentales	moments	max-vraisemblance
$\widehat{p}$	0.10	0.10	0.10
$\widehat{q}$	0.79	0.79	0.79
$(1-\widehat{p}-\widehat{p})$	0.10	0.10	0.10

TABLE 4.4 – Estimation des paramètres à l'aide de trois méthodes.

Il est remarquable que toutes les méthodes donnent le même résultat. Le modèle est donc satisfaisant dans le cadre de cette expérience.

4.2.2.2 Chaîne de Markov

Tel que mentionné plus haut, le planteur est subdivisé en plusieurs sous-unités. Chacune d'elles permet d'ensemencer une cellule. Pour une rangée fixée, il y a 4 sous-unités. Les flèches de la Figure 4.3 permettent d'illustrer le propos.

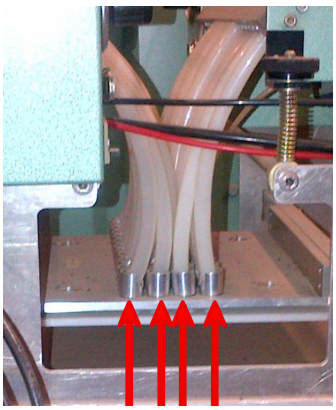


FIGURE 4.3 – Sous-unités du planteur.

Ainsi, lors du premier arrêt du plateau, l'élément pointé par la flèche gauche ensemence la quatrième cellule de la première rangée (par convention). Par la suite, le même élément, lors du deuxième arrêt, ensemence la huitième cellule et ainsi de suite jusqu'à la dernière. Soit c_i où $i = 1, 2, \ldots, 24$ le numéro, pour une rangée fixe, correspondant à une cellule. Donc, c_1 est la première cellule ; c_8 la huitième, par exemple. Le test, ici, consiste à choisir une rangée, puis une sous-unité. Le choix, purement aléatoire, consiste à l'élément du planteur qui ensemence les cellules $\mathcal{U}_1 = \{c_i \mid i = 1, 5, 9, \ldots, 17, 21\}$. Ensuite, l'objectif est de construire la matrice $\widehat{P}$ de transitions expérimentale. La probabilité $\widehat{p}_{00}$ se calcule comme suit : plusieurs plateaux sont analysés. La première rangée est regardée et toutes les cases vides (état 0) de $\mathcal{U}_1$ sont sélectionnées. Ensuite, la cellule suivante est observée. Si elle est également vide, alors elle est considérée comme un cas favorable. En effet, l'événement $\{\{X_{c_{i+4}} = 0\} \cap \{X_{c_i} = 0\}\}$ s'est réalisé. Tous ces événements sont additionnés. Puisque qu'il faut conditionner cette probabilité, alors, la somme des cas favorables est divisée par le nombre de cases vides faisant parties de l'échantillon. La façon de compter une telle probabilité est illustrée à la Figure 4.4

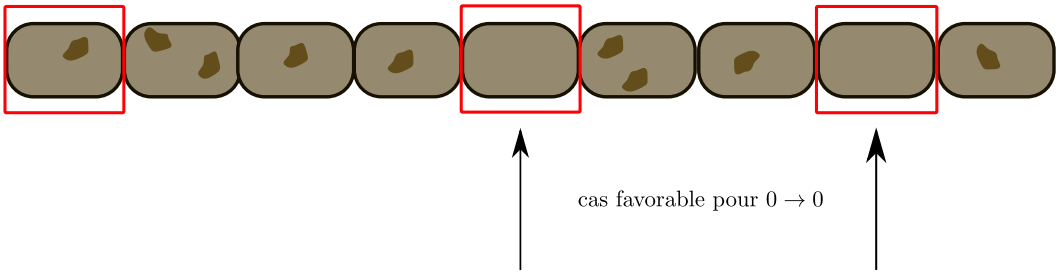


FIGURE 4.4 – Exemple de comptage.

Quatre plateaux furent analysés pour ce test. Les résultats figurent au Tableau 4.5

$\widehat{p}_{00}$	$\frac{1}{9}$
$\widehat{p}_{01}$	$\frac{7}{9}$
$\widehat{p}_{02}$	$\frac{1}{9}$
$\widehat{p}_{10}$	$\frac{6}{44}$
$\widehat{p}_{11}$	$\frac{34}{44}$
$\widehat{p}_{12}$	$\frac{4}{44}$
$\widehat{p}_{20}$	$\frac{4}{5}$
$\widehat{p}_{21}$	$\frac{1}{5}$
$\widehat{p}_{22}$	0

TABLE 4.5 – Probabilités expérimentales de transition.

Donc, la matrice est donnée par :

$$\widehat{P} = \begin{pmatrix} \frac{1}{9} & \frac{7}{9} & \frac{1}{9} \\ \frac{6}{44} & \frac{34}{44} & \frac{4}{44} \\ \frac{4}{5} & \frac{1}{5} & 0 \end{pmatrix}$$

Si $x \in \Omega^{|\Omega|}$, est l'état d'une cellule quelconque étiquetée par n ou plus exactement la loi en probabilité. Alors, il est possible d'écrire $x^{n+1} = x^n P$. Par exemple, supposons que $x = (0, 1, 0)$ représente la cellule c_5 de $\mathcal{U}_1$ ayant une graine. Il est donc possible d'estimer qu'elle sera l'état de la cellule c_{17} (4 étapes plus tard). Simplement,

```
form numpy import linalg as LA

In [13]: P = np.matrix(([1/9.0,7/9.0,1/9.0],[6/44.0,34/44.0,4/44.0],[4

In [14]: P
Out[14]:
[0.111111111111111, 0.777777777777778,  0.111111111111111]
[0.136363636363636, 0.772727272727273, 0.0909090909090909]
[               0.8,                0.2,
0]

In [50]: X.dot(LA.matrix_power(P,4))
Out[50]: array([ 0.18938143,  0.72382191,  0.08679666])

In [51]: X = np.array([0,1,0])
```

```
15
16 In [52]: X.dot(LA.matrix_power(P,4))
17 Out[52]: array([ 0.18938143,  0.72382191,  0.08679666])
```

Ainsi, il y a 18% de chance que la cellule, quatre étape plus tard, soit vide, 72% de chance qu'elle ait une graine et 8% de chance qu'elle en contienne 2. De cette modélisation, il est possible d'obtenir beaucoup d'informations afin de caractériser le planteur. De plus, après quelques transitions, la matrice semble se stabiliser.

```
1  In [15]: P**2
2  Out[15]:
3  [0.207295173961841, 0.709652076318743, 0.0830527497194164]
4  [0.193250688705234, 0.721349862258953, 0.0853994490358127]
5  [0.116161616161616, 0.776767676767677,  0.107070707070707]
6
7  In [16]: P**3
8  Out[16]:
9  [0.186245734562233,  0.72620764321101, 0.0875466222267569]
10 [0.188157748281715, 0.724792692767899, 0.0870495589503854]
11 [0.204486276910519, 0.711991633506785, 0.0835220895826956]
12
13 In [21]: P**5
14 Out[21]:
15 [0.189117526266612, 0.724022841995195, 0.0868596317381932]
16 [0.189182697662077,  0.72397292950754, 0.0868443728303828]
17 [0.189683952783647, 0.723586437231818, 0.0867296099845345]
18
19 In [22]: P**6
20 Out[22]:
21 [0.189231151409422,  0.72393553175331, 0.0868333168372675]
22 [ 0.18921937930102, 0.723944600094428, 0.0868360206045526]
23 [0.189130560545705, 0.724012859497664, 0.0868565799566312]
24
25 In [23]: P**7
26 Out[23]:
27 [0.189210818562474, 0.723951176879107, 0.0868380045584185]
28 [0.189212910156228, 0.723949568902751, 0.0868375209410209]
29 [0.189228796987741, 0.723937345421534, 0.0868338575907245]
30
31 # il en va aussi pour toutes les autres puissances!
```

Étant donné la convergence, la chaîne est irréductible. Tout cela constitue

quelques exemples pouvant caractériser le système.

4.2.2.3 Cartes de contrôle

Simplement, 5 plateaux ont été analysés par un observateur humain et les défectuosités ont été comptabilisées. C'est-à-dire, le nombre de cases ne présentant aucune ou plusieurs graines. Sur la Figure 4.5 , les 4 premiers plateaux sont en contrôle statistique. Les données sont peu nombreuses, mais les 3 derniers plateaux, quoiqu'en contrôle, semble avoir un nombre de défectuosités croissant. Quant au dernier plateau, le nombre de défectuosités est grand et la qualité n'est plus acceptable. En fait, ce comportement s'explique facilement dans le cadre de ce test bien précis : il ne restait que peu de semences dans le réservoir du planteur au moment où le dernier plateau, plus particulièrement, était ensemencé. Lorsqu'il y a peu de graines dans son réservoir, le planteur n'aspire que très peu de ces dernières. Il en résulte donc un grand nombre de cellules vides. L'implémentation de ce type de carte de contrôle est relativement simple quoiqu'elles puisse fournir de précieuses informations.

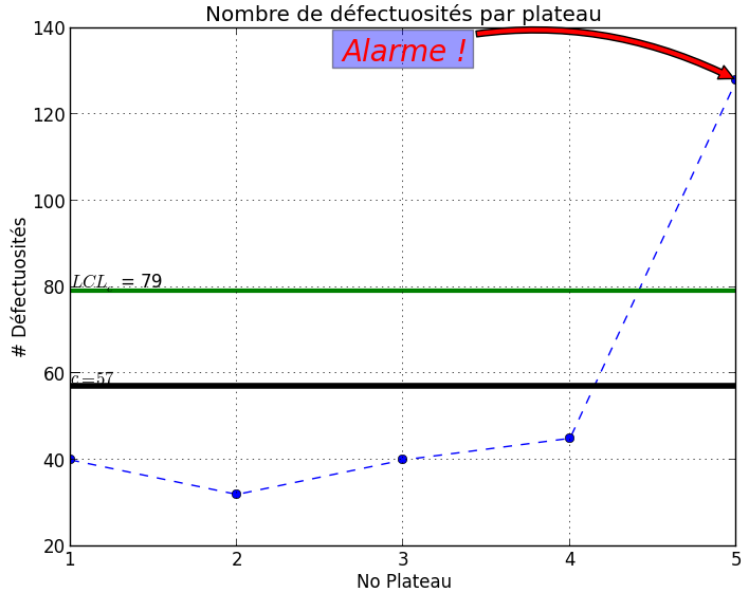


FIGURE 4.5 – Carte-c d'un échantillon réel

4.3 Ajouts potentiels de fonctionnalités

Tel que mentionné plus haut, le système n'en est qu'à ces balbutiements. Plusieurs améliorations peuvent et devront être abordées. L'obstacle majeur est sans aucun doute l'aspect financier. En effet, le matériel de vision (notamment, la ou les caméras) est onéreux.

4.3.1 Ajouts immédiats

Étant donné la faible efficacité du système avec des semences de petites dimensions (l'épinette noire, par exemple), il apparaît qu'une meilleure résolution est essentielle. Donc, une caméra de plus haute résolution est de mise.

4.3.2 Moyen-terme

Le système actuel peut et devrait être remodelé afin d'améliorer l'efficacité. Découper l'analyse d'un plateau en six parties comporte de nombreux problèmes (synchronisation, concaténation fidèle des sous-images en une image finale représentative, lourdeur des calculs en quasi temps réel,...). Ainsi, analyser un plateau en une étape semble être le chemin à suivre. De toute vraisemblance, le seul nouveau problème engendré comparativement à la méthode actuelle est un changement des manipulations effectuées par les opérateurs, car le système ne serait plus sur le convoyeur. Donc, la chaîne de montage ne pourrait plus être en continu et la vitesse d'ensemencement serait dépendante de la vitesse de traitement du nouveau système (mettre un plateau dans le système, puis analyser, puis sortir le plateau). Au lieu que le système s'adapte au processus d'ensemencement actuel, il serait peut-être mieux que le processus d'ensemencement s'adapte au système. Évidemment, il existe virtuellement une infinité de changements possibles, mais trois changements furent pensés récemment. Par exemple, installer deux caméras identiques de part et d'autre du convoyeur qui font la soustraction des images du plateau avant et après l'ensemencement. Malheureusement, deux caméras de qualité seraient nécessaires et il faudrait s'assurer que la

position du plateau soit identique dans les deux sous-systèmes. Un autre système pourrait être muni d'une caméra robot se déplaçant en $x - y$ au-dessus d'un plateau tout en l'analysant. Ainsi, la résolution n'aurait pas à être démesurée pour avoir des images de qualité, mais la vitesse d'exécution pourrait être faible. Finalement, un système très similaire à celui utilisé pourrait être envisagé. En effet, il s'agirait de placer une caméra d'une meilleure résolution qui capte tout le plateau en une seule capture. Le caisson ne serait pas placé sur le convoyeur ce qui aurait pour effet d'éviter les problèmes de synchronisation (attendre que le plateau entre, attendre un arrêt). Par contre, la continuité du processus d'ensemencement serait affectée.

4.4 Conclusion partielle

Ce chapitre avait pour but, d'une part, de mesurer la performance du système dans des conditions réelles. En second lieu, tous les outils statistiques furent également confrontés aux données expérimentales. Finalement, certaines recommandations et pistes de solutions pour un système plus performant furent données.

CHAPITRE 5

CONCLUSION

Tout d'abord, la mise en contexte du projet a été donnée. Puis, la théorie des principaux outils utilisés par le système de vision a été détaillée dans la revue de littérature. Le chapitre suivant, quant à lui, à permis de démontrer de quelle façon ces outils sont utilisés dans le cadre réel d'utilisation du système. Finalement, dans le chapitre dédié à l'expérimentation divers tests ont été menés afin de littéralement mettre à l'épreuve le système de vision. La conclusion est que ce dernier n'est fiable qu'à 30%. Le modèle probabiliste conçu permet de faire des prédictions quant aux erreurs d'ensemencement. De façon similaire, les chaînes de Markov permettent de prédire les diverses occurrences d'états d'une cellule à une autre permettant ainsi de détecter un défaut éventuel du planteur. Les cartes de contrôle créées permettent aussi le monitoring du système dans son ensemble ce qui permet d'évaluer la stabilité du système et lors d'erreurs d' identifier plus facilement les causes possibles. Il est bon de rappeler que le projet est encore au stade de prototype et que les changements futurs permettront assurément d'en faire un outil très utile dans le domaine forestier et même, possiblement, agricole.

BIBLIOGRAPHIE

[1] *OpenCV 2 Computer Vision Application Programming Cookbook*.

[2] Gary Bradsky and Adrian Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[3] R.F. Brewer. *Design of Experiments for Process Improvement and Quality Assurance*. Engineers in business series. Engineering & Management Press, 1996.

[4] Julio César Hernandez, José Mar'a Sierra, and André Seznec. The sac test : A new randomness test, with some applications to prng analysis. In *Proceedings of the International Conference Computational Science and Its Applications - ICCSA 2004, LNCS vol. 3043*, pages 960–967, April 2004.

[5] Patricia Raymond Daniel Dumais, Marcel Prévost. L'épinette rouge, une espèce à bien connaître... pour une sylviculture mieux adaptée ! ., 2007.

[6] E.R. Davies. Chapter 2 - images and imaging operations. In E.R. Davies, editor, *Computer and Machine Vision (Fourth Edition)*, pages 17 – 37. Academic Press, Boston, fourth edition edition, 2012.

[7] Stéphane Derrode. Méthode de binarisation d'otsu, 2012.

[8] Marc Diener. Probabilités élémentaires.

[9] Toby Breckon Eric Solomon. *Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab.* Wiley Online Library, UK, 2011.

[10] Faune et Parcs Québec Forêts. La pépinière de berthier, 2014.

[11] Eugene Hecht. *Optique.* Pearson Education France, Paris, quatrième edition, 2005.

[12] Partenariat innovation forêt. Les grands pins au québec un choix d'avenir, 2007.

[13] OpenCV Developers Team : itseez. About (official website).

[14] Tomáš Suk Jan Flusser and Barbara Zitová. Introduction to moments, 2012.

[15] Guy Lessard. L'épinette noire, une formidable capacité d'adaptation., 2010.

[16] Torbjörn Lestander. *Multivariate NIR Studies of Seed-Water Interaction in Scots Pine Seeds (Pinus sylvestris L.).* PhD thesis, Swedish University of Agricultural Sciences, 2003.

[17] Erik Lokensgard. *Industrial Plastics : theory and application.* Delmar, Cengage Learning, Clifton Park, NY, fifth edition, 2010.

[18] R.K. Morris and W.T. Ha. *The Book of Statistical Process Control.* Zontec Incorporated, 2002.

[19] James R. Norris. *Markov chains.* Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.

[20] Mike Robson. Les chaînes de markov.

[21] Jamie Shutler. Statistical moments, 2002.

[22] Didier Smets. Chapitre 12 ensembles convexes, polytopes et polyèdres.

[23] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1) :32–46, 1985.

[24] Richard Taillet. *Optique géométrique : Mémento.* de Boeck, Bruxelles, 2008.

[25] Samy Tindel. Statistiques : estimation ponctuelle.

[26] K. Thyagarajan Vasudevan Lakshminarayanan, Ajoy K. Ghatak. *Lagrangian Optics*. Kluwer Academic Pub, Norwell,Massachusetts, 2001.

[27] Wikipedia. Numpy.

[28] Wikipedia. C sharp (programming language), 2015.

[29] Wikipedia. Morphologie mathématique, 2015.

[30] Wikipedia. Python (langage), 2015.