Synthèse de modèles de plantes et reconstructions de baies à partir d'images

Jérôme Guénard

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *04/10/2013* **par :**

Jérôme Guénard

Synthèse de modèles de plantes
et reconstructions de baies à partir d'images

JURY

Thierry CHATEAU	Professeur d'Université	Rapporteur
Marc NEVEU	Professeur d'Université	Rapporteur
Frédéric BOUDON	Chargé de recherche	Examinateur
Raphaëlle CHAINE	Professeur d'Université	Examinatrice
Michel DEVY	Directeur de recherche CNRS	Examinateur
Vincent CHARVILLAT	Professeur d'Université	Directeur de thèse
Pierre GURDJOS	Ingénieur de recherche CNRS	Co-encadrant
Géraldine MORIN	Maître de conférence	Co-encadrante

École doctorale et spécialité :
MITT : Image, Information, Hypermedia
Unité de Recherche :
IRIT (5505)
Directeur(s) de Thèse :
Vincent CHARVILLAT et *Géraldine MORIN*
Rapporteurs :
Thierry CHATEAU et *Marc NEVEU*

Titre :

Synthèse de modèles de plantes et reconstructions de baies à partir d'images

Résumé :

Les plantes sont des éléments essentiels du monde qui nous entoure. Ainsi, si l'on veut créer des environnements virtuels qui soient à la fois agréables et réalistes, un effort doit être fait pour modéliser les plantes. Malgré les immenses progrès en vision par ordinateur pour reconstruire des objets de plus en plus compliqués, les plantes restent difficiles à reconstruire à cause de la complexité de leur topologie. Cette thèse se divise en deux grandes parties. La première partie s'intéresse à la modélisation de plantes, biologiquement réalistes, à partir d'une seule image. Nous générons un modèle de plante respectant les contraintes biologiques de son espèce et tel que sa projection soit la plus fidèle possible à l'image. La première étape consiste à extraire de l'image le squelette de la plante. Dans la plupart de nos images, aucune branche n'est visible et les images peuvent être de qualité moyenne. Notre première contribution consiste en une méthode de squelettisation basée sur les champs de vecteurs. Le squelette est extrait suite à un partitionnement non déterministe du feuillage de l'image assurant son réalisme. Dans un deuxième temps, la plante est modélisée en 3D. Notre deuxième contribution est la création de modèles pour différents types de plantes, basée sur les L-systèmes. Puis, un processus d'analyse-par-synthèse permet de choisir le modèle 3D final : plusieurs propositions de squelette sont générées et un processus bayésien permet d'extraire le modèle maximisant le critère a posteriori. Le terme d'attache aux données (vraisemblance) mesure la similarité entre l'image et la reprojection du modèle, la probabilité a priori mesure le réalisme du modèle. Après avoir généré des modèles de plantes, des modèles de fruits doivent être créés. Ayant travaillé principalement sur les pieds de vigne, nous avons développé une méthode pour reconstruire une grappe de raisin à partir d'au moins deux vues. Chaque baie est assimilée à un ellipsoïde de révolution. La méthode obtenue peut être plus généralement adaptée à tout type de fruits assimilables à une quadrique de révolution. La seconde partie de cette thèse s'intéresse à la reconstruction de quadriques de révolution à partir d'une ou plusieurs vues. La reconstruction de quadriques et, en général, la reconstruction de surfaces 3D est un problème très ancien en vision par ordinateur qui a donné lieu à de nombreux travaux. Nous rappelons les notions nécessaires de géométrie projective des quadriques, et de vision par ordinateur puis, nous présentons un état de l'art sur les méthodes existantes sur la reconstruction de

surfaces quadratiques. Nous détaillons un premier algorithme permettant de retrouver les images des foyers principaux d'une quadrique de révolution à partir d'une vue « calibrée », c'est-à-dire pour laquelle les paramètres intrinsèques de la caméra sont connus. Puis, nous détaillons comment utiliser ce résultat pour reconstruire, à partir d'un schéma de triangulation linéaire, tout type de quadriques de révolution à partir d'au moins deux vues. Enfin, nous montrons comment il est possible de retrouver la pose 3D d'une quadrique de révolution dont on connaît les paramètres à partir d'un seul contour occultant. Nous évaluons les performances de nos méthodes et montrons quelques applications possibles.

Title :

Analysis and 3D reconstruction of natural objects from images

Abstract :

Plants are essential elements of our world. Thus, 3D plant models are necessary to create realistic virtual environments. Mature computer vision techniques allow the reconstruction of 3D objects from images. However, due to the complexity of the topology of plants, dedicated methods for generating 3D plant models must be devised. This thesis is divided into two parts. The first part focuses on the modeling of biologically realistic plants from a single image. We propose to generate a 3D model of a plant, using an analysis-by-synthesis method considering both a priori information of the plant species and a single image. First, a dedicated 2D skeletonisation algorithm generates possible branching structures from the foliage segmentation. Then, we built a 3D generative model based on a parametric model of branching systems taking into account botanical knowledge. The resulting skeleton follows the hierarchical organisation of natural branching structures. Varying parameter values of the generative model (main branching structure of the plant and foliage), we produce a series of candidate models. A Bayesian model optimizes a posterior criterion which is composed of a likelihood function which measures the similarity between the image and the reprojected 3D model and a prior probability measuring the realism of the model. After modeling plant models branching systems and foliage, we propose to model the fruits. As we mainly worked on vines, we propose a method for reconstructing a vine grape from at least two views. Each bay is considered to be an ellipsoid of revolution. The resulting method can be adapted to any type of fruits with a shape similar to a quadric of revolution. The second part of this thesis focuses on the reconstruction of quadrics of revolution from one or several views. Reconstruction of quadrics, and in general, 3D surface reconstruction is a very classical problem in computer vision. First, we recall the necessary background in projective geometry quadrics and computer vision and present existing methods for the reconstruction of quadrics or more generally quadratic surfaces. A first algorithm identifies the images of the principal foci of a quadric of revolution from a "calibrated" view (that is, the intrinsic parameters of the camera are given). Then we show how to use this result to reconstruct, from a linear triangulation scheme, any type of quadrics of revolution from at least two views. Finally, we show that we can derive the 3D pose of a given quadric of revolution from a single occlu-

ding contour. We evaluate the performance of our methods and show some possible applications.

Remerciements

Avant tout, merci à toute l'équipe VORTEX de m'avoir accueilli pendant ces 5 ans et demi. Tout particulièrement à mes encadrants Vincent, Pierre et Géraldine pour tout leur soutien et leurs conseils tout au long de ma thèse. Je vais essayer d'être exhaustif en citant tout le monde avec le risque d'oublier certaines personnes. Je remercie donc tous les membres de l'équipe que j'ai vu défiler : les permanents parmi lesquels Romulus, Bernard, Jean, Jean-Denis, Sylvie, Simone, Zouheir mais aussi tous les thésards et jeunes ingénieurs ; ceux qui ont du me supporter dans leur bureau (ce qui n'a pas toujours dû être facile) Sébastien, Lilian, Viorica, Pauline, Benjamin, Viet, Yvain, Marie-Anne et tous les autres Benoît, Alex, Axel, Rabih, Bastien, Phuong, Nicolas, Benjamin, Christophe, Olivier... Un immense MERCI aux Sylvies qui font que tout marche nickel au sein de l'IRIT-ENSEEIHT. Également merci à tous les membres de mon jury pour avoir accepté d'évaluer mes travaux de recherche.

Cette thèse n'aurait pas pu se dérouler sans argent pour la financer. Je remercie donc l'entreprise SODIMEL et son PDG Michel qui m'a permis de visiter des contrées lointaines pour l'acquisition de photos de grappes. Merci également à tous les membres du projet VINNEO ; en particulier le directeur de l'IFV Eric et Carole qui se sont énormément investis dans le montage puis dans le bon déroulement de ce projet ainsi que Jean de la cave de Fronton, toujours présent pour m'aider lors des expérimentations dans les parcelles de vignes.

Je remercie toute ma famille pour leur soutien : mes parents, mes frères Julien et Guillaume et tous les autres. En particulier mes parents qui se sont déplacés pour ma soutenance et qui ont fait que mon pot de thèse était une vraie réussite.

Un dernier paragraphe pour remercier tous mes amis qui m'ont permis de me divertir en dehors du travail dans toutes sortes d'activités : randos, escalade, voyages, camping, slack-line, jeux, baignade, apéros, soirées... Tout d'abord les golbutes de Dijon et de façon plus générale tous les anciens de la période avant Toulouse. Merci à mes colocs Pauline et Vivian et à tous mes potes de Toulouse parmis lesquels, par ordre aléatoire et non d'importance à mes yeux (principalement par coloc pour être sûr de n'oublier personne) : Ed, Marie, Loulou, Nico, Rémi, Diya, Thomas, Mathieu, JB, GP, Debur, Xav, Edouard, Agathe, Lambert, Émeline, Charles, Benoît, Élo, Germain...

Enfin merci à tous ceux que j'ai pu oublier. Je m'en excuse pleinement.

Table des matières

0.1 Notations

Dans la suite de ce document, toute matrice est représentée avec la notation : $\mathtt{M}$, tout vecteur avec la notation : $\mathbf{v}$ et tout scalaire avec la notation : s. $\mathtt{M}^T$ représente la matrice transposée de $\mathtt{M}$, $\mathtt{M}^{-1}$, son inverse et $\mathtt{M}^{-T}$, la transposée de son inverse. De la même manière, $\mathbf{v}^T$ représente le vecteur transposé de $\mathbf{v}$. Le symbole $\triangleq$ signifie « est égal par définition » alors que le symbole $\propto$ signifie « est égal à un facteur d'échelle près ». Le vecteur $\mathbf{0_i^T}$ représente le vecteur composé de i zéros (0...0). La matrice $\mathtt{I}_n$ est la matrice identité de taille $n \times n$. Ces notations sont les mêmes que celles utilisées dans [Hartley 2003].

Chapitre 1

Introduction et contexte

Sommaire

1.1 Contexte général

Les plantes sont des éléments essentiels du monde qui nous entoure. Ainsi, si on veut créer des environnements virtuels qui soient à la fois agréables et réalistes, un effort doit être fait pour modéliser les plantes. Ces environnements peuvent être utilisés aussi bien dans les jeux vidéos que dans le cinéma d'animation, pour créer des visites virtuelles d'environnements ou pour visualiser des modélisations de plantes avec son mobile comme cela se fait de plus en plus dans les jardins botaniques par exemple.

Dans cette optique, l'utilisation des ordinateurs, des téléphones... dont les capacités de calcul sont de plus en plus grandes permet de faire du rendu de scènes avec des détails de plus en plus fins. Cependant, malgré les immenses progrès en vision par ordinateur pour reconstruire des objets de plus en plus compliqués, les plantes restent difficiles à reconstruire à cause de la complexité de leur topologie. En effet, contrairement aux scènes urbaines composées de bâtiments qui peuvent être modélisés par un ensemble de plans et où l'on retrouve des objets très autosimilaires comme les fenêtres, les volets, les lampadaires, les panneaux de signalisation et bien d'autres encore, chaque plante est unique avec un squelette qui lui est propre et est composée de différents organes, souvent complexes, comme les boutons, les feuilles, les fleurs ou les fruits. On peut évidemment considérer que les feuilles ou les fleurs sont toutes autosimilaires également mais, numériser une plante réelle via une procédure de scannage 3D s'avère néanmoins compliqué et demande un stockage de données qui peut devenir énorme si l'on souhaite être le plus proche possible de la réalité. En effet, les plantes sont des objets très complexes et il peut exister des zones occultées. De plus, une fois les données 3D acquises, il faut extraire la topologie de la plante, ce qui est loin d'être évident. Un compromis entre réalisme et efficacité devient alors indispensable si l'on souhaite générer des scènes au sein desquelles on pourra naviguer aisément à l'intérieur comme l'expliquent Boudon *et al.* [Boudon 2006] dans leur article proposant une vue d'ensemble des articles sur la modélisation et le rendu de plantes.

La Figure 1.1, page 5 montre différents branchages d'arbres où l'on a ajouté plus ou moins d'irrégularités pour améliorer le réalisme. Le modèle de gauche est très simple, très rapide à générer et ne requiert que peu d'espace de stockage. En revanche il n'est pas très réaliste. Les arbres suivants sont de plus en plus réalistes mais s'il est de plus en plus coûteux de les générer.

La qualité du réalisme souhaitée peut varier en fonction de l'application ou en

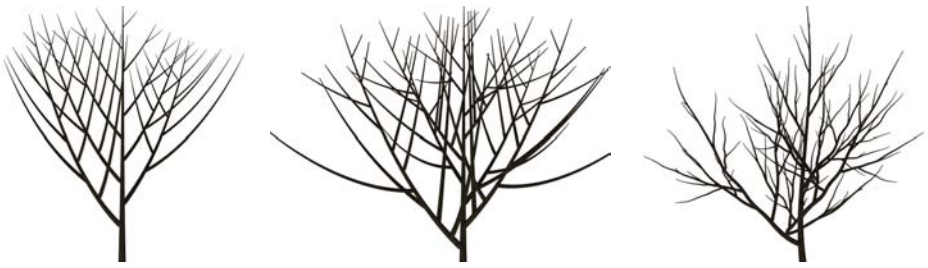

FIGURE 1.1 – Exemples de modélisations de branchages d'arbres. A gauche, le branchage ressemble à une fractale avec de grandes autosimilarités. Puis, de gauche à droite, on ajoute de l'irrégularité afin d'améliorer le réalisme tout en augmentant la taille de stockage nécessaire.

fonction de la distance aux plantes. En effet, il est inutile de modéliser les nervures des feuilles d'un arbre se situant à plusieurs centaines de mètres. Des modèles multi-échelles s'adaptant en fonction de la position de la plante dans la scène peuvent devenir nécessaire pour limiter le temps de calcul.

La modélisation de plantes fournit également des informations importantes sur son développement et peut, à ce titre, intéresser les biologistes ou les cultivateurs pour adapter leurs traitements. Cette thèse s'inscrit dans ce cadre et plus explicitement dans le domaine de la viticulture. Nous nous intéressons à la modélisation d'objets naturels à partir d'une ou plusieurs images et en particulier d'images de pieds de vignes. Cependant, nous avons la volonté de proposer des méthodes applicables à une plus grande variété d'objets naturels et à plusieurs échelles (échelle d'une parcelle ou échelle d'une rangée dans le cas de plantes cultivées, échelle d'une plante, échelle d'un fruit). Nous commençons par un état de l'art complet sur les différentes techniques existant aujourd'hui pour modéliser des plantes. Leurs avantages et leurs limites sont étudiés en fonction de nos besoins, issus des applications liées aux projets finançant cette thèse. Dans un deuxième temps, nous exposons plus précisément dans quel cadre s'inscrit cette thèse et nous développons le plan de ce manuscrit.

1.2 Modélisation de plantes

Nous dressons ici un état de l'art sur la modélisation de plantes. Dans un premier temps, nous nous intéressons aux articles utilisant les grammaires formelles pour générer des plantes. Ces articles visent à obtenir une modélisation réaliste et, éventuellement, à modéliser le processus de développement de la plante. Les mé-

thodes proposées permettent de générer des plantes respectant certaines règles sans pour autant modéliser une plante précise existant dans la nature. Dans un deuxième temps, nous détaillons des articles, plus récents, reconstruisant une plante existante à un moment donné de son développement à partir d'une ou plusieurs images de celle-ci.

1.2.1 Génération de plantes à partir de grammaires formelles

Nous avons vu qu'il n'est pas envisageable de créer un modèle exact d'une plante avec des détails très précis. La représentation numérique d'une plante doit rester relativement compacte pour être stockable et manipulable. Il devient essentiel de rechercher les redondances visuelles présentes au sein des plantes afin de regrouper leurs éléments semblables en vue d'optimiser la taille du modèle final. Ainsi, pour une espèce donnée, un nombre limité de modèles de feuilles peuvent être créés et réutilisés en divers endroits de la plantes ou de l'arbre. La même chose peut être effectuée pour les boutons ou les branches.

Un pionnier en modélisation de plantes est Lindenmayer qui a introduit et généralisé l'utilisation des *L-systèmes* dans [Lindenmayer 1968] (voir section 3.1.1.1). Pour régler le problème de complexité de la plante, des méthodes procédurales ont été développées, rendant possible de recréer l'architecture d'une plante très complexe à partir d'un petit nombre de règles de base simples en utilisant ces mêmes L-systèmes [Prusinkiewicz 1990]. En effet, les plantes respectent certaines règles qui peuvent être utilisées via des grammaires formelles pour modéliser un grand nombre de variétés de plantes différentes comme dans [Deussen 2005, Weber 1995].

Cependant, une application récursive trop stricte de ces règles de base, qui consistent à créer des sous-structures (branchettes) à partir de branches, conduit à la génération de plantes avec des structures autosimilaires pouvant ressembler à des fractales. Il est donc important de développer des approches probabilistes pour permettre plus de réalisme [Prusinkiewicz 1990, de Reffye 1988], en s'appuyant sur des études botaniques [Chaubert-Pereira 2010]. Pour cela, une librairie de symboles est utilisée en faisant varier les positions des éléments, leur orientation ou leur dimension. D'autres approches proposent de modéliser des plantes en faisant entrer en compétition différents organes de la plante devant croître dans un même espace [Palubicki 2009]. Pour cela, l'utilisateur spécifie un volume puis, les branches poussent dans ce volume grâce à un processus génératif. Les règles de compétition peuvent permettre différents types de structure mais un contrôle automatique reste

compliqué. Les résultats sont alors esthétiquement très réalistes comme le montre la Figure 1.2, page 7.

FIGURE 1.2 – Exemples de générations de forêts avec la méthode de Palubicki *et al.* [Palubicki 2009] (images extraites de l'article).

Mais toutes ces méthodes génèrent des plantes sans chercher à modéliser une instance particulière d'une plante à partir d'images. On peut cependant ici s'inspirer de la manière dont ces méthodes utilisent les connaissances a priori disponibles. Dans notre cas, on cherche à modéliser une plante que nous avons préalablement filmée ou photographiée. La suite de l'état de l'art se concentre donc sur les méthodes basées images.

1.2.2 Génération de plantes à partir d'images

Alors que les méthodes procédurales sont capables de représenter la plante à différents niveaux de son évolution, la modélisation à partir d'images cherche à fournir une modélisation instantanée de la plante à un moment donné.

Méthode pionnière

Un des premiers travaux sur ce sujet, celui de Shlyakhter *et al.* [Shlyakhter 2001], reconstruit la forme générale d'un arbre à partir de silhouettes déduites des images. Un volume est calculé comme l'intersection d'un ensemble de cônes de sommets les centres optiques des caméras et passant par ces silhouettes. Le squelette, extrait à partir d'une méthode d'axe médian appliqué sur le volume trouvé, est utilisé comme structure de branches principales puis, des branchettes et des feuilles sont ajoutées en utilisant des L-systèmes. Ce branchage ne semble cependant pas très réaliste car les branches ne sont pas toujours suffisamment lisses à cause de l'utilisation de l'axe

médian, ce qui donne parfois un effet peu naturel comme on peut le voir dans la Figure 1.3, page 8.

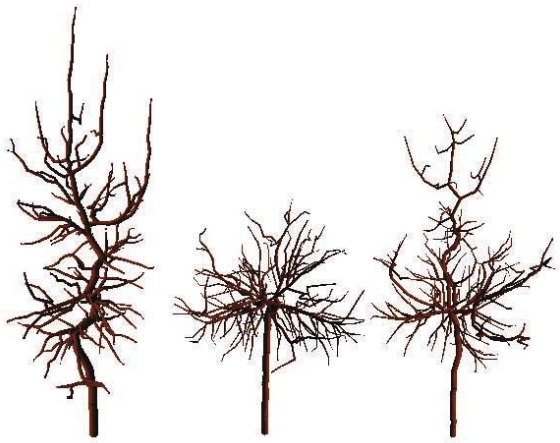

FIGURE 1.3 – Exemples de générations d'arbres avec la méthode de Shlyakhter *et al.* [Shlyakhter 2001] (images extraites de l'article).

Méthodes utilisant la reconstruction de points 3D

Quan *et al.* [Quan 2006, Quan 2007] et Tan *et al.* [Tan 2007] utilisent également plusieurs images pour reconstruire un modèle 3D d'une plante. Pour éviter les problèmes d'appariement, ils se servent de vues proches les unes des autres (plus de 20 images dans leurs exemples). Puis, ils construisent un nuage quasi-dense de points 3D (grâce à la méthode de Lhuillier *et al.* [Lhuillier 2005]) en calculant les paramètres des caméras (Hartley *et al.* [Hartley 2000] et Faugeras *et al.* [Faugeras 2001]). Pour les petites plantes, Quan *et al.* [Quan 2006, Quan 2007] remarquent que les feuilles sont souvent plus essentielles que le branchage. Ils segmentent donc le nuage de points 3D par feuilles en utilisant des informations 2D et 3D puis, ils ajustent un modèle de feuille sur chacun de ces segments. Ils ajoutent enfin les tiges. Cette méthode est présentée dans la Figure 1.4, page 9.

En revanche pour les arbres, Tan *et al.* [Tan 2007] reconstruisent d'abord les branches visibles. Les branches occultées sont alors supposées semblables à ces branches visibles qui sont donc utilisées comme modèles pour faire pousser, de manière récursive, les branches jusqu'à atteindre un volume englobant les feuilles. Celles-ci sont ajoutées à la fin. Cette méthode est présentée dans la Figure 1.5, page 9.

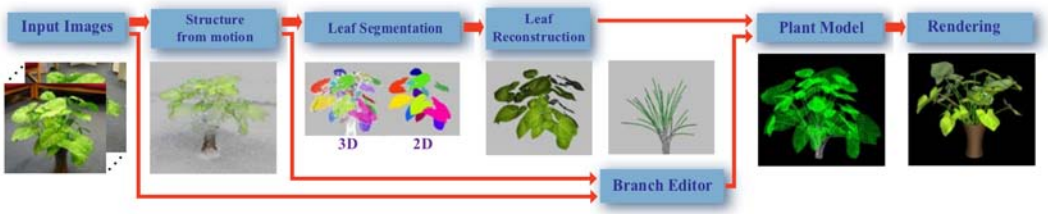

FIGURE 1.4 – Résumé de la méthode de Quan *et al.* [Quan 2006] (images extraites de l'article). Une première étape de *structure from motion* permet de retrouver un nuage de points 3D qui sont ensuite segmentés pour identifier les différentes feuilles. Un modèle 3D de feuille est adapté sur chaque segment et les branches sont ensuite rajoutées.

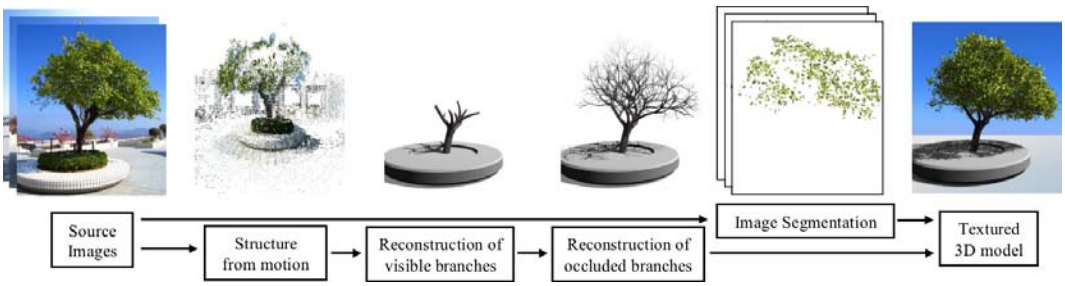

FIGURE 1.5 – Résumé de la méthode de Tan *et al.* [Tan 2007] (images extraites de l'article). Une première étape de *structure from motion* permet de retrouver un nuage de points 3D, utilisé pour la reconstruction des branches visibles. Puis, les branches occultées sont générées en utilisant les branches déjà reconstruites. Les feuilles sont ajoutées à la fin.

Méthodes utilisant plusieurs images

Reche-Martinez *et al.* [Reche-Martinez 2004] se basent quant à eux sur les silhouettes dans plusieurs images et calculent, pour chaque pixel, une opacité qui détermine la projection de plusieurs éléments de l'arbre (feuille, branche, fond...). Ils obtiennent des rendus d'arbres pouvant être insérés dans des scènes virtuelles. La méthode de Neubert *et al.* [Neubert 2007] est détaillée dans la Figure 1.6, page 10. À partir d'images originales (a), ils extraient une carte de densité (b) ainsi qu'un graphe d'attracteurs. Ils calculent ensuite un volume englobant la plante sous forme de voxels à partir d'images prises de différents points de vue (c). Ce volume est rempli de particules se déplaçant en fonction d'un champ de directions calculé à partir des graphes d'attracteurs. Les chemins de ces particules sont utilisés comme système de branchage (d). Un résultat de branchage (e) et d'arbre avec feuilles (f) est

montré dans la Figure 1.6, page 10.

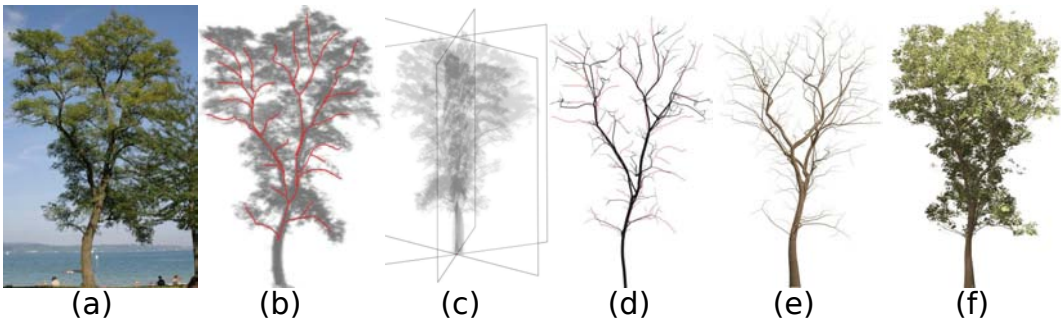

FIGURE 1.6 – Résumé de la méthode de Neubert *et al.* [Neubert 2007] (images extraites de l'article).

Wang *et al.* [Wang 2006] modélisent différents types d'arbre en utilisant, pour chaque espèce, des images d'arbres réels dont ils extraient des homogénéités botaniques. Le but de cette approche n'est cependant pas de reconstruire une instance spécifique d'un arbre dont on fournit l'image mais de générer un arbre qui respecte toutes les propriétés biologiques que respectent les arbres de son espèce. Pour modéliser un objet, Talton *et al.* [Talton 2011] proposent de contrôler des grammaires formelles en les faisant évoluer de manière à ce qu'elles remplissent un volume en 3D donné en entrée. Pour cela, ils utilisent des chaînes de Markov. Les résultats sont esthétiquement très convaincants (Figure 1.7, page 11). On peut tout de même critiquer le fait qu'ils cherchent juste à remplir une zone en utilisant les grammaires données, sans se soucier du réalisme de la plante, ce qui rend les modèles parfois trop denses localement et nécessite une bonne segmentation ou un croquis réaliste en pré-traitement.

Enfin, Li *et al.* [Li 2011] présentent une approche probabiliste permettant la reconstruction d'un arbre dynamique à partir d'une vidéo en commençant par extraire un squelette 2D grâce à la technique de Diener *et al.* [Diener 2009] et en retrouvant dans un premier temps un arbre statique. Le suivi de points dans les vidéos permet de retrouver les variations des angles des branches du modèle 3D lorsque l'arbre bouge. En effet, pour une même branche, on peut retrouver l'angle grâce aux mouvements des projections de ces points 3D dans l'image en considérant le fait que la distance entre les points 3D doit être invariante. Enfin, ces auteurs sont capables de générer plusieurs arbres de la même espèce pour générer une forêt entière à partir de quelques arbres reconstruits (un exemple est montré dans la Fi-

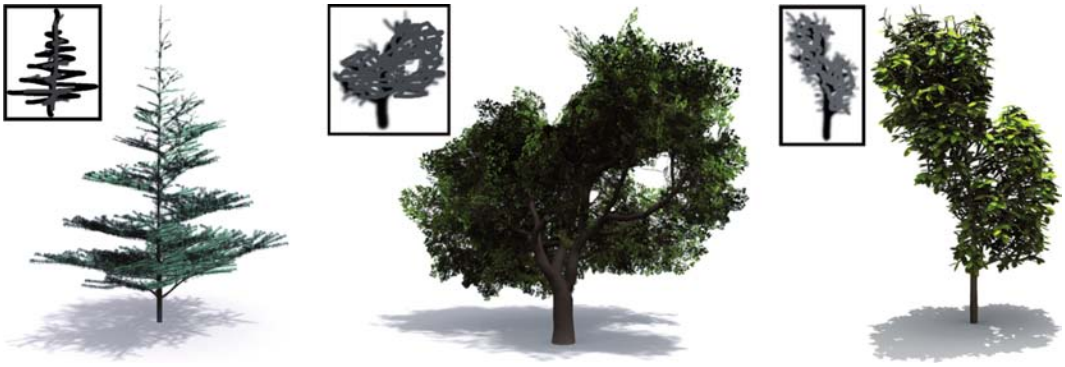

FIGURE 1.7 – Résultats de la méthode de Talton *et al.* [Talton 2011] (images extraites de l'article).

gure 1.8, page 11). Pour toutes ces méthodes, l'information de base est donnée par les images et lorsque les modèles de base de la plante sont suffisamment riches, les résultats obtenus peuvent être très esthétiques sans garantir pour autant d'être parfaitement représentatifs des espèces considérés dans le sens où les instances générées ne respectent pas nécessairement certaines propriétés biologiques.

FIGURE 1.8 – Exemple généré avec la méthode de Li *et al.* [Li 2011]. À gauche l'image originale, au milieu le modèle 3D et à droite, d'autres instances de la même espèce (images extraites de l'article).

Toutes ces méthodes utilisent plusieurs images d'une plante pour la reconstruire afin d'avoir un maximum d'informations et d'être le plus fidèle possible à la réalité contrairement à notre souhait de n'utiliser qu'une seule image.

Méthodes utilisant des croquis données par l'utilisateur

L'approche pionnière est celle de Okabe *et al.* [Okabe 2006] qui proposent à un utilisateur de générer rapidement et facilement un modèle 3D en dessinant à la main des croquis à partir d'opérations d'éditions de base. Par exemple, l'utilisateur peut

spécifier des branches ou des feuilles. Pour distribuer les branches en 3D, celles-ci sont projetées sur le sol et un algorithme permet que les angles formés par chaque couple de projections de branches soient le plus large possible (Figure 1.9, page 12). Nous nous inspirons de cette méthode dans la section 2.6.

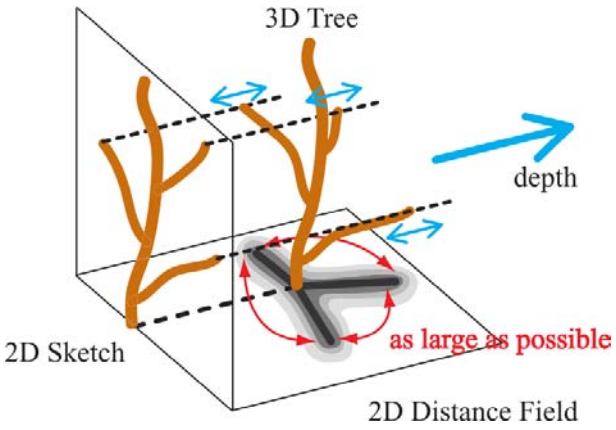

FIGURE 1.9 – Illustration de la méthode de Okabe *et al.* [Okabe 2006] pour retrouver des branches en 3D (image extraite de l'article).

Liu *et al.* [Liu 2010] proposent également de retrouver un modèle de plante possible à partir de croquis dessinés par un utilisateur par dessus des images. Ils en calculent un nuage de points 3D à partir duquel ils construisent une structure de branchage 3D. Wither *et al.* [Wither 2009] partent du constat que faire un croquis pour donner l'allure du feuillage à plusieurs échelles est plus rapide et plus intuitif pour l'utilisateur que de chercher à dessiner les branches une par une. L'utilisateur ne dessine, sous forme de traits, qu'un croquis de l'arbre à différentes échelles. Il n'affine ses dessins à l'échelle inférieur que pour une seule sous-partie de l'arbre. Les positions des branches sont ensuite déduites grâce à des connaissances botaniques fournies à l'algorithme pour donner un modèle 3D de plante possible, parfaitement ajusté au croquis de l'utilisateur. Enfin, Runions *et al.* [Runions 2007] utilisent la compétition des branches dans l'espace comme facteur dominant pour modéliser la forme de leurs arbres. L'utilisateur peut alors spécifier un volume dans lequel les branches peuvent croître. L'arbre pousse de manière itérative. À chaque étape, l'algorithme évalue l'espace disponible dans lequel peut croître une branche et calcule sa direction optimale.

Ces méthodes, bien que donnant des résultats très satisfaisants, ne sont pas automatiques car l'utilisateur doit intervenir pour dessiner un croquis représentant l'allure générale du branchage de la plante. Nous souhaitons en revanche un traite-

ment de l'image de la plante qui soit totalement automatisé.

Méthodes n'utilisant qu'une seule image

Certains travaux comme celui de Tan *et al.* [Tan 2008] n'utilisent qu'une image de la plante dont ils extraient le squelette. Néanmoins une intervention de l'utilisateur pour segmenter le feuillage ou pour dessiner le tronc et les branches visibles est requise. Une représentation 3D du branchage est alors déduite à partir des branches visibles qu'ils utilisent pour construire un alphabet de sous-branchages et à partir de modèles existants dans une base de données. Un exemple de résultat est montré dans la Figure 1.10, page 13.

FIGURE 1.10 – Résultat de la méthode de Tan *et al.* [Tan 2008] (images extraites de l'article).

Une autre méthode qui peut être citée est celle de Zeng *et al.* [Zeng 2006]. Ces auteurs retrouvent un graphe représentant la topologie du branchage à partir d'une image d'arbre sans feuilles. Ils ajoutent ensuite des informations de profondeur aux branches en supposant une projection orthogonale afin de retrouver une structure en 3D (Figure 1.11, page 13). Nous nous inspirons également de cette méthode dans la section 2.6.

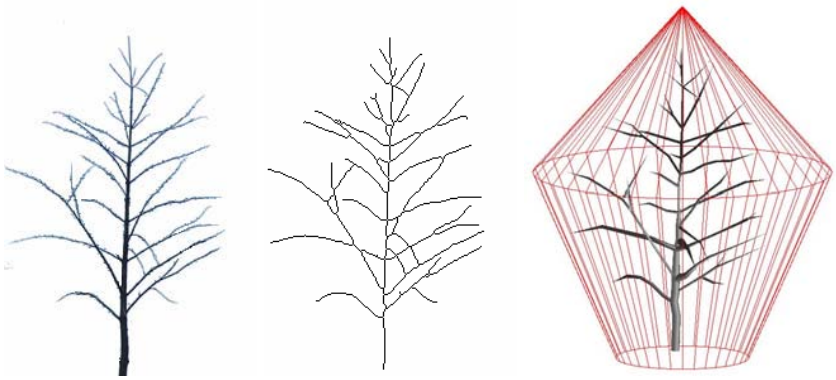

FIGURE 1.11 – Illustration de la méthode de Zeng *et al.* [Zeng 2006] pour retrouver des branches en 3D (images extraites de l'article).

Ces dernières méthodes ne sont pas toujours automatiques et elles utilisent la plupart du temps les branches visibles pour effectuer un apprentissage afin de reconstruire les branches occultées. Nous verrons dans nos applications que souvent, aucune branche n'est visible sur nos images et nous ne disposons que de la forme du feuillage pour retrouver le squelette du branchage.

Toutes les méthodes présentées ci-dessus travaillent au niveau de la plante en général. Nous allons maintenant changer d'échelle et parler des méthodes travaillant au niveau du fruit.

1.3 Modélisation de fruits

Dans la plupart des travaux, la modélisation de fruits est motivée par l'estimation du volume. Ainsi, Iqbal *et al.* [Iqbal 2011] estiment le volume de fruits possédant un axe de symétrie comme les pommes à partir du contour de la projection du fruit dans une image. Ils déterminent si cette forme est plutôt sphérique, ellipsoïdale ou parabolique afin d'estimer le volume en utilisant un modèle adapté. Omid *et al.* [Omid 2010] évaluent le volume d'agrumes grâce à deux vues perpendiculaires du fruit. Le fruit est découpé en différents troncs elliptiques et son volume est calculé comme la somme de ces sous-parties comme montré dans la Figure 1.12, page 14. Grâce à des tests, ils montrent que leurs résultats sont similaires aux volumes calculés en plongeant les fruits dans de l'eau et en regardant les volumes déplacés. De plus, ils montrent que la densité des agrumes est proche de 1 et donc que le volume et la masse sont les mêmes.

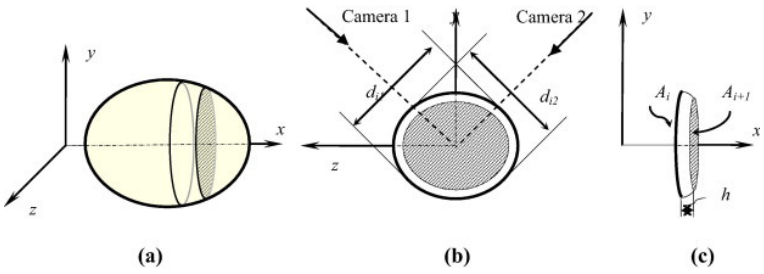

FIGURE 1.12 – Illustration de la méthode de Omid *et al.* [Omid 2010] (images extraites de l'article).

Mebatsion *et al.* [Mebatsion 2011] développent une nouvelle procédure de modélisation de fruits semi-symétriques en utilisant leurs profils longitudinaux et transversaux. La première étape consiste à extraire les contours grâce aux séries de Fou-

rier dans une image du fruit coupé en deux. Les coupes longitudinales ne suffisent pas pour la plupart des fruits à cause de l'absence d'axe de symétrie. En combinant les coupes longitudinales des demis fruits et les coupes transversales du fruit en entier, un modèle complet du fruit est généré (voir Figure 1.13, page 15).

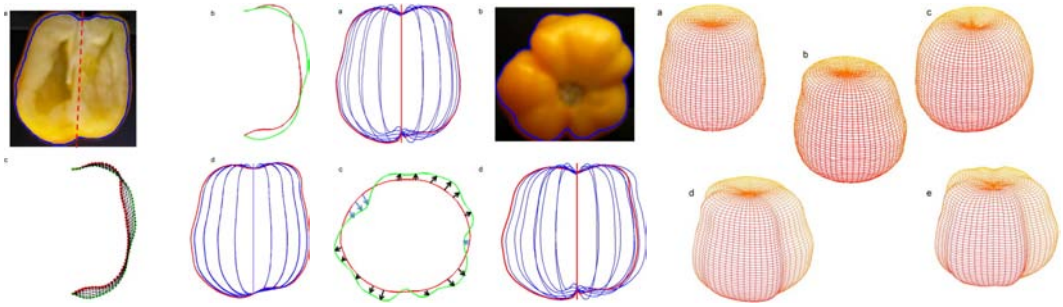

FIGURE 1.13 – Illustration de la méthode de Mebatsion *et al.* [Mebatsion 2011] (images extraites de l'article).

Enfin, Huang *et al.* [Huang 2013] proposent une approche procédurale basée sur des grammaires pour modéliser des fruits en forme de grappes. Un modèle de grappe est généré en utilisant les L-systèmes grâce à des connaissances sur le développement d'une grappe. Puis, un outil interactif permet à l'utilisateur de modéliser une grappe en s'inspirant par exemple d'une photo comme on peut le voir sur des exemples dans la Figure 1.14, page 15.

FIGURE 1.14 – Résultats de la méthode de Huang *et al.* [Huang 2013]. Pour chaque paire d'images, la photo se trouve sur la gauche et la modélisation sur la droite (images extraites de l'article).

L'objectif premier pour nous a été la modélisation de grappes de raisin. Nous nous sommes aperçus que l'on pouvait approximer chaque grain par un ellipsoïde et même plus précisément un ellipsoïde de révolution. Nous avons donc développé des méthodes de reconstructions de quadriques de révolution à partir d'une ou plusieurs images.

1.4 Cadre et plan de la thèse

Cette thèse s'inscrit dans le cadre du projet VINNEO (section 4.2) d'une part et d'une collaboration avec l'entreprise SODIMEL (section 6.8.1) d'autre part. Ces deux projets ont pour but l'extraction de caractéristiques de plantes ou de fruits à partir d'images dans le cadre de la viticulture. Dans les deux cas, nous décidons de modéliser en 3D les éléments que l'on cherche à inspecter visuellement et nous généralisons les méthodes développées pour des cas différents de ceux de la viticulture.

Dans le cas des plantes, nous cherchons à développer des méthodes de reconstruction n'utilisant qu'une seule image. Pour le cas des vignes, nous verrons que nous partons d'une vidéo mais le pied de vigne n'étant filmé que d'un seul côté, seule une image sera utilisée car les informations apportées par d'autres images sont alors redondantes. Ce problème est fortement sous-contraint d'autant que les images que nous utilisons lors de nos applications peuvent être de qualité moyenne car acquises par une caméra placée sur un tracteur circulant dans les rangées de vignes. De plus, le recul n'étant pas suffisant entre deux rangées, l'axe optique de la caméra n'est pas perpendiculaire aux vignes et une rectification métrique doit être effectuée (nous verrons par la suite que, à cause des techniques de palissage utilisées, les branches sont supposées être dans un même plan et nous souhaitons donc une image parallèle au plan où se trouvent les branches). Pour terminer, les branches sont parfois toutes invisibles dans l'image à l'exception du tronc. Par conséquent, les méthodes utilisant les branches visibles pour créer un alphabet de base de sous-branchages ne peuvent s'appliquer ici. Pour palier à ces problèmes, l'utilisation de connaissances a priori est très utile et même nécessaire pour retrouver des modélisations de plantes réalistes, c'est-à-dire à la fois dont la projection est similaire aux images mais aussi respectant les propriétés biologiques de cette plante. En effet, via la partie modélisation, nous pouvons obliger la plante à suivre telle ou telle structure tout en collant au mieux à l'image. Pour le cas des vignes, un très grand nombre de

travaux dans le domaine botanique ou agronomique peuvent être trouvés comme par exemple [Lebon 2004] ou [Louarn 2005]. Nous avons également acquis un grand nombre de connaissances sur les vignes grâce à des experts dans ce domaine avec qui nous avons travaillé. Nous avons par exemple appris le nombre de rameaux moyen par vigne ou la forme globale du pied de vigne (connus grâce au système de taille utilisé). Pour utiliser tous ces a priori en plus des images, nous nous inspirons d'un grand nombre d'articles présentés précédemment. Tout cela est développé en détail dans la partie I divisée en trois chapitres qui sont exposés dans l'introduction de la partie I.

Dans le cas des fruits, nous proposons une nouvelle méthode de reconstruction de quadriques de révolution détaillée dans la partie II. En effet, beaucoup de fruits peuvent être assimilés à des formes quadratiques et en particulier à un ensemble d'ellipsoïdes de révolution dans le cas des grappes de raisin. Les méthodes développées sont utilisables pour reconstruire tout type d'objets de forme quadratique. Nous montrons que au moins deux vues sont nécessaires pour reconstruire une quadrique de révolution seulement à partir de ses contours dans les images alors qu'une quadrique dont on connaît déjà les paramètres peut être reconstruite avec le contour dans une seule vue. Cette partie est très différente de la précédente et propose comme contributions des méthodes de résolution algébrique au problème de la reconstruction de quadriques en utilisant la géométrie projective. Nous analysons les résultats de nos méthodes et les appliquons à quelques exemples réels.

Nous commençons donc par la première partie de cette thèse en s'intéressant au problème de la modélisation de plantes à partir d'une image.

Première partie

Reconstruction à l'échelle de la plante

Introduction de la partie I

Le but de cette partie est la modélisation de plantes, biologiquement réalistes, à partir d'une image. Ainsi, la plante modélisée doit se projeter de manière la plus fidèle possible à l'image tout en respectant un certain nombre de contraintes biologiques. Nous proposons ici une méthode en plusieurs étapes permettant d'obtenir une modélisation de plante la « meilleure possible ».

La première étape consiste à extraire de l'image le squelette de la plante. Nous commençons par détailler notre méthode sur des pieds de vignes où toutes les branches sont supposées être dans un même plan en raison de la taille et du palissage utilisés pour ce type de culture. Cependant, dans la plupart de nos images, aucune branche n'est visible et les images peuvent également être de qualité moyenne car souvent acquises à partir d'une caméra fixée sur un engin motorisé (voir section 4.1.1). Nous considérons ici la segmentation du feuillage effectuée. Notre première contribution consiste donc en une méthode de *squelettisation* assurant de trouver un *squelette* respectant toutes les propriétés biologiques de la plante à l'intérieur d'un feuillage ne laissant apparaître que peu ou pas d'informations sur les branches. Cette étape est développée dans le premier chapitre de cette partie. À la fin de ce chapitre, nous montrons comment il est possible d'adapter cette méthode à d'autres types de plantes et en particulier à des plantes dont le squelette est non plus en 2D comme supposé pour les pieds de vignes mais en 3D.

Dans un deuxième temps, cette plante doit être modélisée en 3D grâce à un modeleur adéquat comme expliqué dans la section 3.1. Notre deuxième contribution est la création de modèles pour différents types de plantes, utilisant les L-systèmes. Nous commençons par expliquer ce qu'est un L-système puis, le choix du modeleur utilisé et enfin la manière dont nous avons décidé de modéliser les plantes. Tout ceci est détaillé dans les premières sections du chapitre 3.

La fin de ce même chapitre explique le processus d'analyse-par-synthèse en commençant par un bref état de l'art sur ce sujet. La méthode de squelettisation est mise à contribution un certain nombre de fois en faisant varier différents paramètres propres à la plante comme le nombre de rameaux pour un pied de vigne. Puis un critère de sélection permet de choisir la modélisation de la plante la meilleure possible au sens où sa projection est la plus fidèle possible à l'image d'origine.

Enfin, le dernier chapitre de cette partie I montre quelques applications possibles

à cette méthode ainsi que les applications liées spécifiquement au projet VINNEO (section 4.2). Nous développons ici le pré-traitement des images dans le cas des pieds de vigne pour obtenir une image où le feuillage est segmenté automatiquement.

Chapitre 2

Squelettisation

Sommaire

Nous cherchons ici à retrouver une structure de branchage d'une plante, biologiquement réaliste, à partir de sa forme apparente. Pour cela, nous disposons en entrée d'une forme binaire correspondant à la projection du volume d'une plante. Le but de la méthode est de proposer un squelette possible qui, une fois modélisé et après ajout des feuilles se projette de manière à coïncider au mieux avec la forme binaire de départ. Nous supposons en effet dans un premier temps que la segmentation du feuillage est donnée. Nous expliquons plus tard, dans la section 4.1 comment il est possible de segmenter automatiquement le feuillage dans des images acquises de manière similaire aux nôtres.

Nous avons commencé à travailler sur des images de pieds de vignes dans le cadre du projet finançant en grande partie cette thèse. Nous avons donc demandé de dessiner la structure du branchage sur des images de pieds de vignes à des experts en viticulture (Eric Serrano, Directeur Régional de l'Institut Français de la Vigne et du Vin sud-ouest et ingénieur viticole et Jean Hemmi, Sous-Directeur du groupe Vinovalie, responsable de la cave de Fronton et ingénieur également). On fait l'hypothèse que toute la structure de la plante se trouve dans un même plan. En effet, les branches sont attachées à des fils de fer parallèles par les vignerons. Quelques exemples de ces tracés experts de branchages peuvent être vus dans la Figure 2.1, page 24.

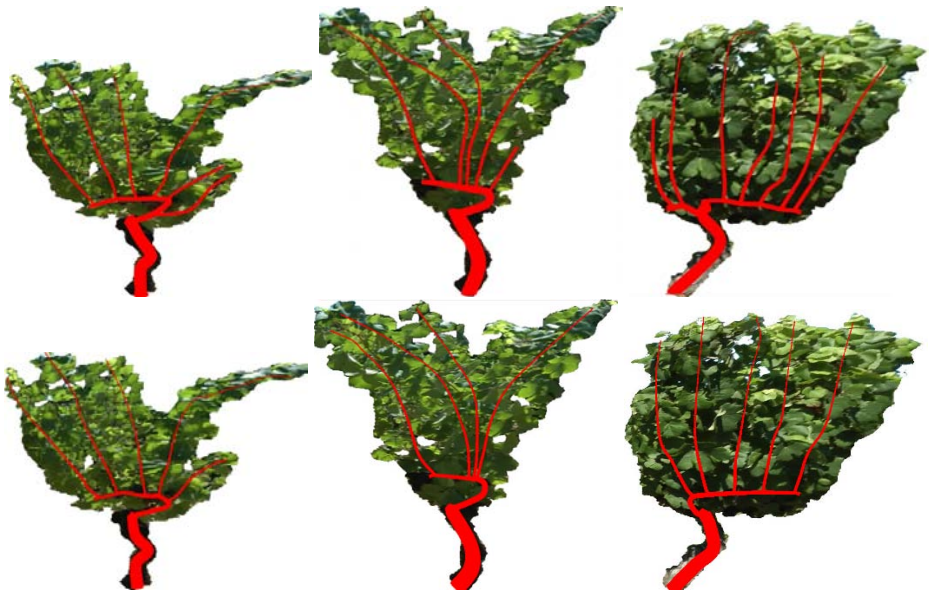

FIGURE 2.1 – Tracés des deux experts sur trois images de pieds de vignes.

On remarque que les experts, à partir de l'image de feuillage d'un pied de vigne,

arrivent à retrouver le branchage principal sans que celui-ci ne soit visible. En fait, leurs connaissances sur ce type de plantes ainsi que sur les techniques de palissage et de taille leur permettent de « deviner » où se trouvent les branches en analysant la forme du contour du branchage ou la position et l'orientation des feuilles. Bien sûr, les experts peuvent se tromper. Néanmoins, les deux experts retrouvent souvent les mêmes branches pour un même pied de vigne, ce qui montre l'influence de la structure du branchage sur les images. Dans tous les cas, on est assuré qu'il s'agisse au moins d'un branchage cohérent et donc biologiquement réaliste grâce aux connaissances des experts sur le sujet.

Nous partons donc du postulat que si des experts peuvent retrouver un squelette de branches biologiquement réaliste à partir de l'image d'un pied de vigne et à partir de connaissances sur ce type de plante, on peut espérer trouver une méthode pour parvenir au même résultat automatiquement afin de retrouver des structures de branches similaires à celles tracées par les experts. Nous rappelons ici que la difficulté principale réside dans le fait qu'aucune branche n'est visible mis à part le tronc ce qui nous empêche d'utiliser des méthodes comme [Tan 2008] ou [Zeng 2006]. Pour extraire une structure dans une forme dont on ne voit aucun élément, l'utilisation de connaissances a priori est indispensable. Nous nous sommes naturellement intéressés à la squelettisation d'une forme binaire et nous avons cherché à développer une méthode telle qu'il soit possible de prendre en compte des connaissances a priori correspondant à des contraintes biologiques.

Dans le cas des vignes, notre méthode de squelettisation est capable de respecter la structure principale des squelettes déduite des techniques de taille et de palissage (de la même manière, ces connaissances ont été utilisées par les experts lorsqu'ils ont dessiné le branchage sur les images des pieds de vigne). Nous avons ensuite adapté cette méthode pour d'autres types de plantes.

2.1 État de l'art

La squelettisation est utilisée en analyse de formes dans le but de réduire une forme en un ensemble de courbes, appelé squelette, centré dans la forme d'origine. La squelettisation conserve les propriétés topologiques de la forme d'origine et, selon la méthode employée, les propriétés géométriques. La plupart du temps, le principe marche pour le cas continu mais en pratique, et plus particulièrement en traitement d'images comme notre cas d'usage, les algorithmes sont décrits dans le cas discret.

On s'intéresse donc ici au cas discret car nous travaillons sur des images qui sont donc pixelisées. On peut diviser les différentes méthodes en cinq sous-ensembles :

– Les méthodes d'*axes médians* dont le premier article a été proposé par Blum *et al.* [Blum 1967]. Dans cet article, on nomme squelette l'ensemble des courbes définies comme le lieu des points correspondant aux centres des cercles tangents en au moins deux points au contour de la forme. Cependant, cette méthode est très sensible au bruit et aux petits trous dans la forme comme on peut le voir dans la Figure 2.2, page 26 (surtout dans le cas de feuillage où les feuilles peuvent laisser un petit trou qui peut se trouver juste à côté de la branche).

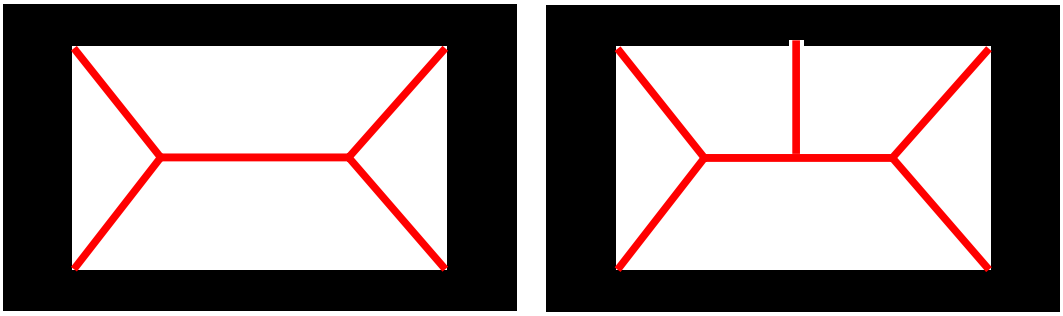

FIGURE 2.2 – Illustration de la non robustesse de l'axe médian au bruit. À gauche, on peut voir l'axe médian d'une forme binaire représentant un rectangle. À droite, le rectangle est légèrement bruité sur le haut. On voit qu'une petite modification sur le masque binaire modifie complètement le squelette trouvé.

Beaucoup d'algorithmes partent de cette méthode pour extraire une première base de squelette puis élague celui-ci selon certains critères. Dans le cas discret, pour avoir un squelette d'un pixel (ou un voxel en 3D) d'épaisseur, un choix doit être fait et la condition que doit respecter l'axe médian n'est donc plus automatiquement vérifiée. En effet, pour une forme binaire représentant un rectangle de dix pixels de largeur par exemple, le squelette peut se trouver sur la ligne 5 ou sur la ligne 6. L'autre possibilité est de garder un squelette avec deux pixels d'épaisseur.

– Les méthodes d'*amincissement topologique*. Elles consistent à retirer, de façon itérative, les points sur le contour de l'objet (la forme en 2D) jusqu'à obtenir un squelette d'un pixel d'épaisseur (d'un voxel en 3D). L'exemple souvent utilisé pour l'expliquer est le suivant : en supposant que la forme est composée d'herbes sèches, on met le feu à tous les contours en même temps. Si le feu se

propage à la même vitesse, l'endroit où se réunit les flammes correspond au squelette. On peut citer Yang *et al.* [Yang 2008] qui proposent un algorithme rapide en deux étapes, invariant aux rotations, dans le cas discret. La première étape est une méthode itérative permettant de retirer les pixels sur les contours de la forme jusqu'à obtenir un squelette de deux pixels d'épaisseur maximum. La deuxième étape assure aux branches du squelette de n'avoir qu'un pixel d'épaisseur tout en préservant la connectivité.

– Les méthodes de *détection de crêtes basées sur un champ de distances*. Pour chaque point à l'intérieur de l'objet, on calcule sa distance au bord de l'objet le plus proche. Une distance autre que la distance euclidienne peut être utilisée. Ces méthodes cherchent à retrouver le squelette à l'intérieur de ce champ de distances. Pour cela, les auteurs commencent par trouver des points candidats correspondant à des maximum locaux qui sont ensuite connectés (après suppression de certains si besoin). Telea *et al.* [Telea 2003] commencent par localiser des voxels centrés par rapport à trois directions orthogonales grâce à une méthode de squelettisation 2D. Puis, à partir de ce premier squelette, un nouveau critère permet l'extraction de nouvelles courbes centrées. Des étapes d'amincissement et de reconnexion permettent ensuite de trouver le squelette final.

– Les *méthodes géométriques*. L'approche la plus souvent utilisée consiste à partir de l'axe médian puis à extraire le squelette en élaguant certains segments. D'autres approches utilisent les lignes de niveaux correspondant aux courbes (plus exactement polygones pour le cas discret) telles que tous les points se trouvent à une même distance du bord le plus proche de la forme. Ces approches peuvent également être utilisées dans le cas continu. Le squelette est alors calculé en retrouvant les points où s'intersectent les lignes de niveaux et en les connectant. Par exemple, Bai *et al.* [Bai 2007] partent d'un squelette comme l'axe médian qui comporte souvent trop de composantes. Ils élaguent celui-ci grâce à une méthode basée sur la partition de contours de la forme binaire initiale en différentes cellules. Le squelette doit ensuite ne comporter qu'une seule composante par cellule. Pour partitionner la forme binaire, ils cherchent les points du contour de la forme binaire les plus caractéristiques de cette forme et pour cela, ils montrent que l'utilisation de l'algorithme DCE (*Discrete Curve Evolution*) introduit par Latecki *et al.* [Latecki 1999] donne les meilleurs résultats. Cet algorithme simplifie une forme binaire en un poly-

gone dont le nombre de sommets est choisi par l'utilisateur. Ces sommets sont extraits sur les contours de telle sorte que le polygone soit le plus représentatif de la forme binaire.

– Les *méthodes de champ de fonctions*. Les méthodes de cette classe calculent un potentiel pour chaque point à l'intérieur de l'objet qui est fonction des potentiels de l'ensemble des points du bord de l'objet. Les points avec les plus forts potentiels sont alors extraits comme étant des points du squelette. Ils sont ensuite reliés en suivant des chemins passant par des points avec les meilleurs potentiels possibles. On peut citer ici la méthode de Cornea *et al.* [Cornea 2005].

La Figure 2.3, page 28 montre des exemples de résultats pour quelques unes des méthodes citées ci-dessus.

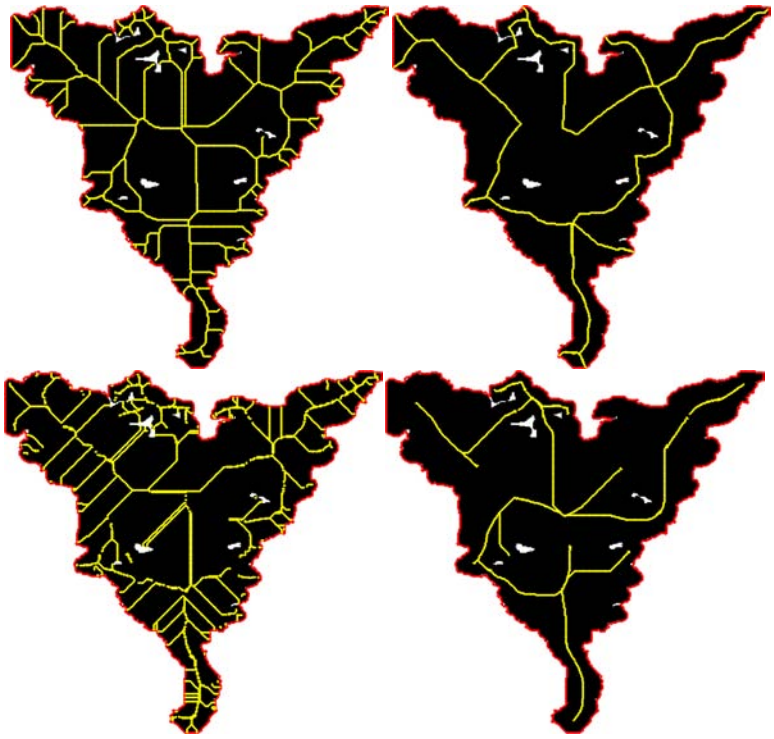

FIGURE 2.3 – Différentes méthodes de squelettisation utilisées sur l'image binaire d'un pied de vigne. En haut à gauche, la méthode d'axe médian [Blum 1967], en haut à droite, une méthode géométrique [Bai 2007], en bas à gauche, une méthode d'amincissement [Yang 2008] et en bas à droite, la méthode que l'on adapte [Cornea 2005]. Le squelette extrait est représenté en jaune alors que le contour de la forme binaire est en rouge.

Chacune de ces méthodes cherche à respecter certaines propriétés essentielles à un squelette. Cornea *et al.* [Cornea 2007] ont fait un état de l'art complet des différentes méthodes de squelettisation. Ils listent les propriétés souhaitables des algorithmes de squelettisation aussi bien pour le cas continu que pour le cas discret. Selon les méthodes utilisées, toutes ces propriétés ne peuvent pas être respectées. Cela est dû au fait que certaines propriétés sont contradictoires entre elles. Par exemple, dans le cas discret, il est possible qu'un squelette ne puisse pas être à la fois centré et fin (d'épaisseur égale à un). Dans la liste ci-dessous, une définition de chaque propriété est donnée dans le cas discret. Un squelette doit donc être :

- homotope (préservation de la topologie). Un squelette préserve la topologie d'un objet s'il a le même nombre de composantes connexes et au moins une boucle pour chaque tunnel ou chaque cavité de l'objet original.
- invariant sous transformation isométrique. Si O un objet et T une transformation isométrique, alors le squelette de l'objet transformé Sk(T(O)) doit être égal à la transformation du squelette de l'objet T(Sk(O)).
- de reconstruction correcte : la reconstruction d'un objet O à partir de son squelette doit être identique à O Rec(Sk(O))=O. Pour reconstruire l'objet, on considère l'union des boules inscrites de rayon maximal (par analogie au cercle inscrit en 2D) pour chaque point du squelette.
- fin : le squelette doit avoir 1 pixel d'épaisseur (1 voxel en 3D).
- centré : le squelette doit être centré, c'est-à-dire se trouver sur la surface médiane (axe médian en 2D).
- fiable : tout point appartenant à la surface de l'objet doit être visible d'au moins un point du squelette.
- tel qu'on peut détecter les jonctions et différencier les composantes « significatives ». Une composante significative d'un objet est une composante qui peut être distinguée visuellement du reste de l'objet (définition vague impliquant la perception humaine et donc très subjectif). Toutes ces composantes doivent avoir un correspondant un à un avec une composante du squelette.
- connexe : une forme binaire connexe doit avoir un squelette connexe, à savoir en un seul morceau (conséquence de homotope).
- robuste : le squelette d'un objet et celui de ce même objet bruité doivent être les mêmes.
- lisse : la variation de la direction de la tangente lorsque l'on se déplace le long du squelette ne doit pas être trop grande.

– hiérarchique : différents squelettes peuvent être trouvés hiérarchiquement à différentes complexités. Le squelette calculé à un certain niveau dans la hiérarchie doit contenir tous ceux calculés à des couches inférieures comme sous-ensembles. Les niveaux de détails sont alors de plus en plus fins.

Toutes ces propriétés ne sont pas nécessairement souhaitées en fonction des applications. De plus, il faut parfois faire des compromis comme par exemple pour la robustesse et la reconstruction correcte qui sont deux propriétés contradictoires. Si l'on souhaite un squelette qui soit robuste au bruit, sa reconstruction ne sera alors automatiquement plus correcte. Dans notre cas, nous souhaitons obtenir la structure du squelette d'un branchage de plante ou d'arbre. Nous nous basons sur les dessins d'experts visibles dans la Figure 2.1, page 24. Nous voulons donc un squelette fin mais pas nécessairement centré car les branches ne passent pas automatiquement au centre des formes binaires, en particulier lorsque plusieurs branches partagent le même espace. Le squelette doit être en un seul morceau et donc connexe ainsi que robuste car l'ajout d'une tâche correspondant à une feuille sur la forme binaire ne doit pas modifier l'ensemble du squelette. De plus, chaque branche doit être suffisamment lisse individuellement même s'il peut exister des cassures correspondant aux jointures entre les branches. On ne souhaite pas avoir une même branche composée de multiples cassures, ce qui ne serait pas réaliste.

La Table 2.1, page 30 reprend et complète le tableau décrit dans [Cornea 2007]. Elle détaille pour quelques méthodes ou familles de méthodes les propriétés respectées (O pour Oui et N pour Non). Les lettres en vert sont utilisées pour signifier que le respect ou non de la propriété est en accord avec nos besoins. En revanche, une lettre rouge signifie que nos besoins ne sont pas en adéquation avec le respect ou non de cette propriété.

	Axe médian	Amincissement		Distance	Géométrique		Champ de fonctions		Besoins
G : général		G	[Yang 2008]	G	G	[Bai 2007]	G	[Cornea 2005]	
Homotope	O	O	O		O	O	N	N	**N**
Invariant aux transf.	O		N	O		O	O	O	**N**
Reconstruction	O	N	N		N	N	N	N	**N**
Fin	O		N			O	O	O	**O**
Centré	O		O			O		N	**N**
Fiable	O		O			N		N	**N**
Détection de jonctions	O		O		O	O	O	O	**O**
Connexe	O	O	O			O		N	**O**
Robuste	N	N	N	N	N	N	O	O	**O**
Lisse	N		N			N	O	O	**O**
Hiérarchique	O	N	N			O	O	O	**O**

TABLE 2.1 – Résumé des propriétés voulues pour le calcul d'un squelette et leur respect ou non en utilisant certaines méthodes ou familles de méthodes.

2.2 Adaptation de la formule de Cornea

En regardant la Table 2.1, page 30, on remarque que la méthode la plus proche de nos besoins est [Cornea 2005]. En effet, la propriété de connexité peut être assurée d'une autre façon grâce aux connaissances sur le squelette que nous allons insérer dans notre méthode. Notre méthode n'est pas invariante aux transformations car les connaissances que l'on apporte contraignent la direction de certaines branches comme cela sera explicité plus tard. Si on retourne notre forme binaire, on ne trouvera donc pas le même squelette. Ce n'est pas gênant à partir du moment où on fournit l'image binaire de la plante dans le bon sens à notre algorithme. Inversement, les autres méthodes assurent de retrouver un squelette centré et non lisse. En effet, le fait de lisser la branche induit de ne plus respecter le caractère centré du squelette dans la plupart des cas.

Nous détaillons dans un premier temps les étapes de la méthode originale de Cornea puis nous l'adaptons pour notre cas d'usage. Nous précisons que dans la suite de ce chapitre, nous considérons le voisinage d'un pixel comme étant tous les pixels le touchant par une arête ou par un sommet. Par conséquent, chaque pixel a huit voisins. Pour chaque point intérieur $\mathbf{p_i}$ de l'image binaire $\mathcal{B}$, un vecteur force $\overrightarrow{\mathbf{f_i}}$ est calculé comme la moyenne pondérée des vecteurs normées aux points du contour :

$$\overrightarrow{\mathbf{f}_i} = \sum_{\mathbf{m_j} \in \Omega} \frac{1}{||\overrightarrow{\mathbf{m_j p_i}}||^2} \frac{\overrightarrow{\mathbf{m_j p_i}}}{||\overrightarrow{\mathbf{m_j p_i}}||} \tag{2.1}$$

où Ω contient tous les pixels contours $\mathbf{m_j}$ de $\mathcal{B}$ (points rouges du (a) de la Figure 2.4, page 32) correspondant à tous les pixels de la forme binaire ayant au moins un pixel voisin n'appartenant pas à la forme binaire. Ensuite, les points $\mathbf{p_i}$ dont la magnitude du vecteur tend vers 0 ($||\overrightarrow{\mathbf{f}_i}|| \to 0$), aussi appelés *points critiques*, sont extraits (points bleus du (b) de la Figure 2.4, page 32). Ils sont connectés entre eux en suivant la direction des vecteurs $\overrightarrow{\mathbf{f}_i}$ pixel après pixel (points verts du (c) de la Figure 2.4, page 32).

Un défaut majeur de cette méthode pour notre cas est sa non-robustesse aux trous internes dans la forme binaire, malgré sa relative robustesse au bruit en ce qui concerne les contours externes de la forme binaire. Un autre problème est l'unicité de branche dans une même zone là où nous voudrions placer plusieurs branches dans notre modèle. L'idée est de rendre cette méthode robuste aux trous et de permettre de placer plusieurs branches dans une même zone. En se basant sur des experts

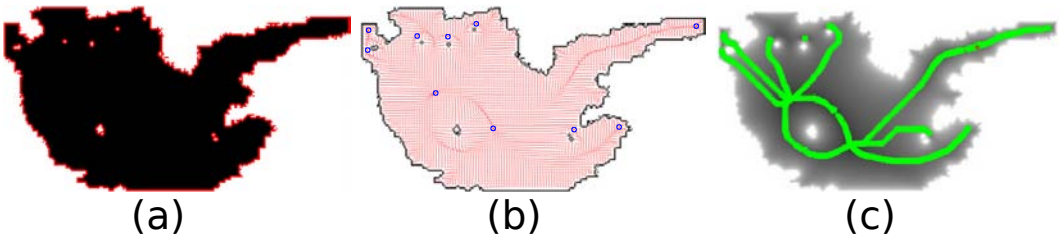

FIGURE 2.4 – Méthode originale de Cornea utilisée sur l'image binaire d'un pied de vigne (a). Les points du contour Ω sont représentés en rouge. (b) Le champ de vecteurs calculé grâce à l'équation (2.1) avec les points critiques représentés en bleu. (c) Le squelette extrait, représenté en vert.

botanistes, nous supposons que différentes branches de taille relativement similaire peuvent exister et partager une même zone dans l'image binaire de la plante. Voici une idée intuitive de notre stratégie : nous cherchons à partitionner la silhouette en différentes cellules en ajoutant des points contours dans la forme binaire comme montré dans la Figure 2.5, page 32.

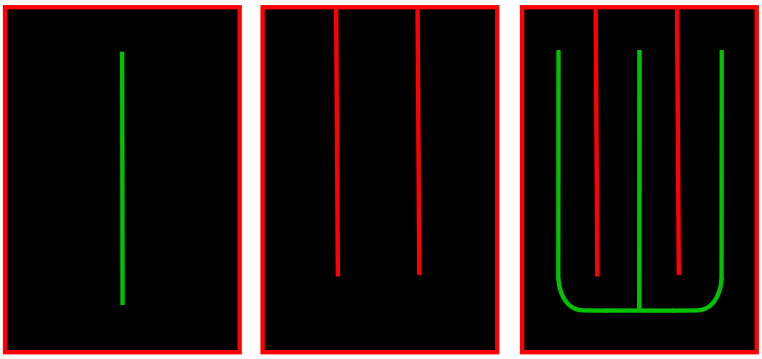

FIGURE 2.5 – À gauche, le squelette (en vert) extrait avec la méthode originale de Cornea. Au milieu, un partitionnement est généré (lignes rouges) pour contraindre le calcul du squelette à obtenir une structure plus ramifiée. À droite, le nouveau squelette est calculé.

Nous souhaitons une méthode non déterministe pouvant proposer plusieurs squelettes possibles pour chaque forme binaire d'une plante. L'idée sous-jacente est de sélectionner plus tard le meilleur squelette parmi plusieurs possibles. Les contours ne sont donc pas ajoutés de manière déterministe mais nous construisons une *carte de probabilités* $\mathcal{P}$ qui attribue, à chaque pixel intérieur $\mathbf{p_j}$ de $\mathcal{B}$, une probabilité $\mathcal{P}_j$ d'être un point contour. Ainsi, le calcul du champ de vecteurs s'effectue à l'aide de cette nouvelle formule :

$$\overrightarrow{\mathbf{f_i}} = \sum_{\mathbf{m_j} \in \Omega} \frac{1}{||\overrightarrow{\mathbf{m_j p_i}}||^2} \frac{\overrightarrow{\mathbf{m_j p_i}}}{||\overrightarrow{\mathbf{m_j p_i}}||} + \sum_{\substack{\mathbf{p_j} \in \mathcal{B} \setminus \Omega \\ j \neq i}} \frac{\mathcal{P}_j}{||\overrightarrow{\mathbf{p_j p_i}}||^2} \frac{\overrightarrow{\mathbf{p_j p_i}}}{||\overrightarrow{\mathbf{p_j p_i}}||} \tag{2.2}$$

Le vecteur force $\overrightarrow{\mathbf{f_i}}$ ne dépend plus uniquement des $\mathbf{m_j} \in \Omega$ mais de tous les pixels de $\mathcal{B}$ en fonction de leur probabilité $\mathcal{P}_j$.

Dans la Figure 2.5, page 32, nous souhaiterions obtenir une carte de probabilités où les probabilités d'être un point contour sont plus importantes autour des pixels rouges internes à la forme. Nous expliquons comment obtenir une telle carte dans la section suivante.

2.3 Calcul de la carte de probabilités

On détaille le calcul d'une carte de probabilité $\mathcal{P}$ pour un nombre de branches n fixé. Il s'agit d'effectuer $n-1$ découpes dans la forme binaire. Pour calculer $\mathcal{P}$, les trois étapes décrites ci-après sont nécessaires.

2.3.1 DCE : *Discrete Curve Evolution*

Nous remarquons dans les tracés d'experts que les composantes concaves et convexes les plus importantes sont très utiles pour détecter les branches. Nous souhaitons donc détecter les zones creuses de la forme qui pourraient être le signe d'une séparation entre deux branches. L'idée ici est de simplifier la forme binaire afin de ne conserver que ces composantes les plus importantes. Pour cela, nous utilisons l'algorithme DCE [Latecki 1999] qui permet de simplifier le contour Ω de la forme binaire en un polygone dont on aura préalablement donné le nombre de sommets, proportionnel au nombre n de branches voulues (typiquement, on a choisi dans notre méthode $3n$). L'algorithme DCE retire itérativement le sommet du polygone qui a la mesure de *pertinence* la plus faible. Cette mesure, détaillée dans l'article, dépend de l'angle de ce sommet, ainsi que des longueurs des segments qui l'entourent. Un exemple de cet algorithme peut être vu dans la Figure 2.6, page 34.

Grâce au polygone simplifié, on peut retrouver facilement les sommets dont l'angle interne à la forme est supérieur à π. Ces derniers ont plus de chance d'indiquer une séparation entre deux zones contenant deux branches distinctes. On calcule donc l'angle α_l correspondant à l'angle intérieur de chaque sommet du polygone. Ces angles vont nous permettre, dans la deuxième étape, de donner une probabilité

FIGURE 2.6 – À gauche, l'image originale d'un pied de vigne avec les points contours au feuillage représentés en rouge. Au milieu, le polygone calculé grâce à l'algorithme DCE avec 24 points puis avec 8 points à droite. Les sommets dont l'angle interne à la forme est supérieur à π sont représentés en bleu. Ils séparent plus certainement deux zones contenant deux branches distinctes que les autres sommets.

ρ_k d'indiquer une telle séparation à chaque point de l'ensemble de points $(\mathbf{c_k})_{,k=1..K}$ extrait uniformément le long du polygone.

2.3.2 Découpes

Les *découpes* sont des séparations qui vont permettre de partitionner notre forme binaire en différentes cellules (autant que de branches). Les pixels le long de ces découpes correspondent ensuite aux pixels ayant la probabilité la plus élevée d'être un point contour dans la carte $\mathcal{P}$. Les découpes sont effectuées en fonction :

- des connaissances biologiques que l'on a sur la plante considérée et des connaissances de la taille et du palissage utilisés dans le cas d'une plante cultivée,
- de la forme du feuillage.

Les sous-sections suivantes détaillent ces critères pour les différents cas traités.

2.3.2.1 Connaissances apportées pour le cas des vignes

Pour ce qui concerne le premier critère, dans le cas de la vigne, on sait que les branches de premier ordre (appelées aussi *rameaux*) sont fixées de manière verticale en partant d'une branche horizontale principale (appelée aussi *baguette*). Les découpes doivent alors respecter cette contrainte.

Pour ce qui concerne le deuxième critère, le polygone en sortie du DCE va nous aider à trouver itérativement les endroits les plus probables dans la forme où les découpes doivent être positionnées en fonction des angles α_l et des découpes trouvées aux itérations précédentes.

Les branches devant pousser de manière verticale, on place les points terminaux des découpes de manière quasi-uniforme en bas du polygone issu du DCE. Puis les

points initiaux des découpes sont choisis en haut du polygone, de manière à respecter la contrainte de verticalité des branches. Pour cela, on calcule, pour chaque point du polygone $\mathbf{c_k}$, une probabilité ρ_k d'être un de ces points initiaux tenant compte de deux critères :

- la proximité d'un angle convexe dans le polygone pour privilégier les découpes qui passent par ces angles : $\rho_k^1 \sim \sum_{l=1}^{N} \frac{1}{d(\mathbf{c_k},\mathbf{s_l})}(1 - \frac{\alpha_l}{2\pi})$,
- la distance le long du contour au plus proche des points de $\mathcal{H}$ contenant les points ayant déjà été choisis comme points initiaux d'une découpe afin de ne pas avoir de cellules trop petites : $\rho_k^2 \sim \min_{\mathbf{c}\in\mathcal{H}} d(\mathbf{c_k}, \mathbf{c})$.

ρ_k est proportionnel à $\phi(\rho_k^1, 1, \sigma) + \phi(\rho_k^2, 1, \sigma)$ avec $\phi(., 1, \sigma)$ représentant la fonction gaussienne centrée en 1 et d'écart-type σ (pour le cas de nos images de vignes, $\sigma = 0.4$ pour une résolution basse d'environ 100000 pixels par image).

Pour chaque point terminant une découpe, on lui choisit aléatoirement un point initial en fonction de ρ_k. Ce point peut alors être accepté ou refusé si par exemple les découpes se croisent ou si la découpe est telle que l'une des cellules a une aire trop faible par rapport à la taille totale de l'image binaire. On obtient un résultat comme celui que l'on peut voir à gauche de la Figure 2.7, page 36.

2.3.2.2 Connaissances apportées pour le cas des plantes monopodiales

Dans le cas d'une plante *monopodiale*, la croissance de la plante est assurée d'année en année par le bourgeon terminal (situé à l'extrémité de la tige et assurant la croissance en longueur, par opposition aux bourgeons latéraux responsables de la ramification). Ainsi, les branches d'ordre 1 poussent à partir d'un tronc vertical montant jusqu'en haut de la plante. Les points terminaux sont donc placés le long du tronc qui va du bas de la plante jusqu'en haut. Si l'on souhaite plus de branches en haut, on placera plus de points terminaux en haut. Puis les probabilités ρ_k sont calculées de manière analogue à ce qui est expliqué au-dessus. On associe itérativement un point terminal d'une découpe à un point choisi aléatoirement en fonction des ρ_k. Cette association est acceptée ou rejetée selon le respect ou non de certaines contraintes biologiques de la plante comme l'angle que forme la découpe avec le tronc. On remarque par exemple que pour un *liquidambar*, les branches du bas forment un angle d'environ 90° avec les tronc alors que celles du haut ne forment plus qu'un angle d'environ 60°. Un exemple est montré à droite de la Figure 2.7, page 36.

Afin d'adapter notre méthode au plus grand nombre de plantes, nous avons

ajouté à notre algorithme la possibilité de faire des découpes non plus droites mais quadratiques ou même cubiques. Cela permet d'obtenir des branches avec des formes plus ou moins complexes. Une tangente est alors imposée à la découpe en une ou deux de ses extrémités.

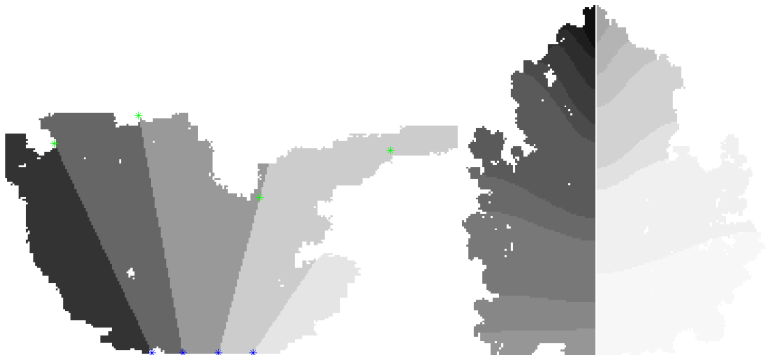

FIGURE 2.7 – Exemples de découpes linéaires sur la forme d'un pied de vigne avec $n = 5$ branches à gauche et de découpes cubiques sur la forme d'un liquidambar pour le cas monopodial avec n=17 branches à droite.

2.3.2.3 Connaissances apportées pour les autres cas

Cette méthode peut ensuite être adaptée à n'importe quel type de plante en ajustant le calcul des ρ_k et en modifiant l'acceptation ou le rejet d'une découpe de manière à ce que les connaissances biologiques sur la plante considérée soient respectées.

2.3.3 Probabilité dense sur la forme binaire

Pour terminer, une probabilité est donnée à chaque point de $\mathcal{B}$. Les point de Ω ont une probabilité 1 et les points intérieurs ont une probabilité inversement proportionnelle à leur distance à la plus proche découpe. Les découpes pouvant être droites, cela permet de donner plus de liberté à la forme des branches contrairement à si l'on attribuait simplement une probabilité 1 aux points se situant sur les découpes. On choisit l'utilisation d'une gaussienne et pour tout $\mathbf{p_i} \in \mathcal{B}$, on a :

$$\mathcal{P}_i = \begin{cases} 1 \text{ si } \mathbf{p_i} \in \Omega \\ \frac{1}{s\sqrt{2\pi}} \exp \frac{-\delta(\mathbf{p_i},\mathcal{D})}{2s^2} \text{ sinon} \end{cases} \tag{2.3}$$

où $\delta(\mathbf{p_i}, \mathcal{D})$ désigne la distance euclidienne de $\mathbf{p_i}$ à la plus proche découpe et s peut varier selon si on veut donner plus ou moins d'importance à la localité des

découpes. Ainsi, plus s est proche de 0, plus les découpes sont considérées comme un contour et les branches passent donc exactement au milieu des découpes avec le risque de paraître moins réalistes. Dans notre cas, on prend $s = 2$ pixels. On trouve alors une carte $\mathcal{P}$ comme montré dans la Figure 2.8, page 37.

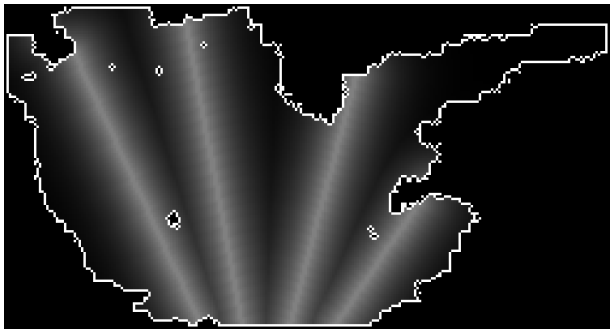

FIGURE 2.8 – Carte de probabilité $\mathcal{P}$ calculée pour un nombre de branches égal à 5.

Une fois la carte de probabilités calculée, on retrouve un champ de vecteurs $(\overrightarrow{\mathbf{f_i}})_i$ associé en tout point $(\mathbf{p_i})_i$ de $\mathcal{B}$ grâce à l'équation 2.2, comme montré dans la Figure 2.9, page 37.

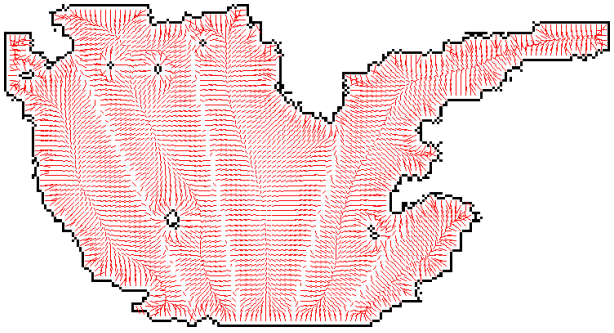

FIGURE 2.9 – Champ de vecteurs calculé avec notre méthode sur l'image binaire d'un pied de vigne pour $n = 5$ branches.

Nous sommes satisfaits des champs de vecteurs retrouvés car on voit que se dessinent à l'intérieur des positions potentielles de branches. Celles-ci tiennent alors compte du contour de la forme, des différentes découpes mais aussi des trous dans le feuillage comme on le souhaitait. Il s'agit maintenant de retrouver la géométrie des squelettes à partir de ces champs de vecteurs.

2.4 Extraction du squelette

Il s'agit maintenant d'extraire le squelette. Pour chaque *cellule* $p = 1..n$ (on peut voir les différentes cellules avec l'exemple d'un pied de vigne dans la Figure 2.7, page 36), on peut extraire un sous-squelette de manière analogue à celle de Cornea. Les points critiques correspondant aux points dont la norme du vecteur tend vers zéro sont extraits puis reliés entre eux en suivant la direction des vecteurs pixel par pixel. On obtient alors, pour chaque branche p, M_p points appelés *points attracteurs* et notés $a^p_{j,j=1..M_p}$ (points verts dans la Figure 2.10, page 38).

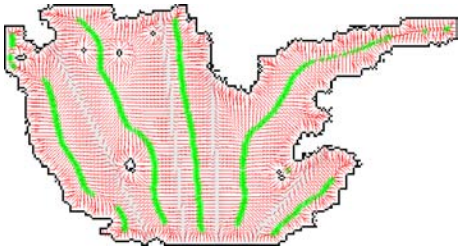

FIGURE 2.10 – Champ de vecteurs en rouge calculé pour $n = 5$ branches. Les points attracteurs, représentés en vert, sont extraits dans chacune des 5 différentes cellules de la partition.

Les connaissances biologiques sur la plante (voir sections 2.3.2.1, 2.3.2.2 et 2.3.2.3) permettent de construire un modèle paramétrique simple que l'on cherche à adapter aux points attracteurs. La Figure 2.11, page 39 montrent deux exemples de modèles de pieds de vignes pour deux modes de taille différents : le *guyot simple* et le *cordon de royat*. Cette étape assure d'obtenir des squelettes biologiquement réalistes.

Chaque $p$ième branche d'un squelette $\mathcal{S}$ est représentée par une courbe Catmull-Rom de degré 3 [Catmull 1974] paramétrée par des points de contrôle n^p_k. Cette courbe doit s'ajuster aux $a^p_{j,j=1..M_p}$. Pour cela, on extrait N points $s^p_{i,i=1..N}$ uniformément le long de chacune de ces courbes. On souhaite minimiser la distance entre chaque a^p_j et l'ensemble $\{s^p_{i,i=1..N}\}$ ainsi qu'entre chaque s^p_i et l'ensemble $\{a^p_{j,j=1..M_p}\}$. Le squelette final $\mathcal{S}_f$ est alors celui vérifiant :

$$\mathcal{S}_f = \arg\min_{\mathcal{S}}\Big(\sum_p \min_{n^p_k}\Big(\sum_{a^p_j} \min_{s^p_i} d(s^p_i, a^p_j) + \sum_{s^p_i} \min_{a^p_j} d(s^p_i, a^p_j)\Big)\Big). \tag{2.4}$$

Un exemple est illustré dans la Figure 2.12, page 40.

Nous sommes satisfaits des résultats obtenus car les branches sont bien réparties

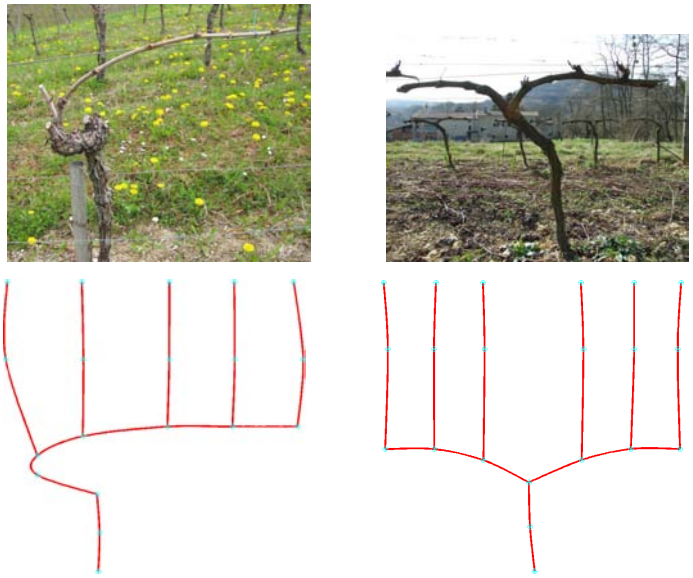

FIGURE 2.11 – À gauche un modèle de branchage de pied de vigne taillé en guyot simple. À droite, la taille utilisée est le cordon de royat. On peut voir des squelettes schématiques en dessous avec les noeuds qui les composent en cyan.

sur l'ensemble de la forme binaire, de manière cohérente d'une part avec le contour de la forme mais également avec les trous dans le feuillage.

2.5 Hiérarchisation

La méthode proposée ci-dessus permet, grâce à des découpes effectuées en fonction des connaissances sur la plante, d'obtenir un branchage de "*premier ordre*". Toutes ces premières branches partent du tronc ou de la baguette dans le cas des vignes. L'algorithme de squelettisation proposé peut être appliqué récursivement. Ainsi, chaque cellule est vue comme la forme binaire d'une nouvelle plante et la branche extraite lors de l'itération précédente devient la branche d'où partent de nouvelles sous branches d'ordre supérieur (Figure 2.13, page 41). Dans le but d'améliorer le réalisme de la plante, chaque cellule n'est plus définie par les découpes mais est une cellule de Voronoï. À chaque branche de l'ordre inférieur correspond sa cellule du diagramme de Voronoï [Aurenhammer 1991] généralisé à des courbes. Ainsi, chaque point de la forme binaire est associé à la cellule de la branche d'ordre inférieur la plus proche. Les cellules n'ont donc plus nécessairement de bords droits comme cela pouvait être le cas. On peut voir une illustration de ce calcul de branches de différents ordres pour le cas du liquidambar dans la Figure 2.13, page 41. Ensuite,

FIGURE 2.12 – Exemple d'un squelette calculé avec notre méthode sur l'image d'un pied de vigne avec $n = 5$ branches avec pour modèle utilisé celui de la taille en guyot simple.

pour déterminer les nouvelles sous-cellules, les points terminaux sont placés sur la courbe représentant la branche d'ordre inférieur et les points initiaux sont recherchés sur les bords de la cellule d'ordre inférieur. Pour sélectionner ces points qui génèrent les nouvelles sous-cellules, le même algorithme, explicité précédemment, est utilisé.

2.6 Reconstruction d'un squelette en 3D

L'algorithme de squelettisation décrit ci-dessus ne permet de retrouver qu'un branchage 2D sur une forme binaire en 2D. Cela peut-être le cas pour certaines plantes comme par exemple les vignes qui sont cultivées sur des fils de fer et dont les branches principales sont donc tous dans un même plan.

Nous proposons ici une méthode permettant de retrouver une structure ramifiée en 3D en se basant sur les travaux de Zeng *et al.* [Zeng 2006] et Okabe *et al.* [Okabe 2006] pour les plantes de type monopodial. Le but est de déduire des informations de profondeur des branches dans le squelette en 2D afin d'avoir un rendu réaliste de la plante en 3D depuis les autres vues. Cependant, on doit veiller à préserver la forme de l'image obtenue après projection du modèle 3D dans une vue similaire à celle de l'image originale.

Dans un premier temps, on suppose que l'image de la plante que l'on analyse a été obtenue via une projection orthogonale. Cette hypothèse est réaliste si la caméra est positionnée suffisamment loin de la plante. On calcule ensuite l'enveloppe

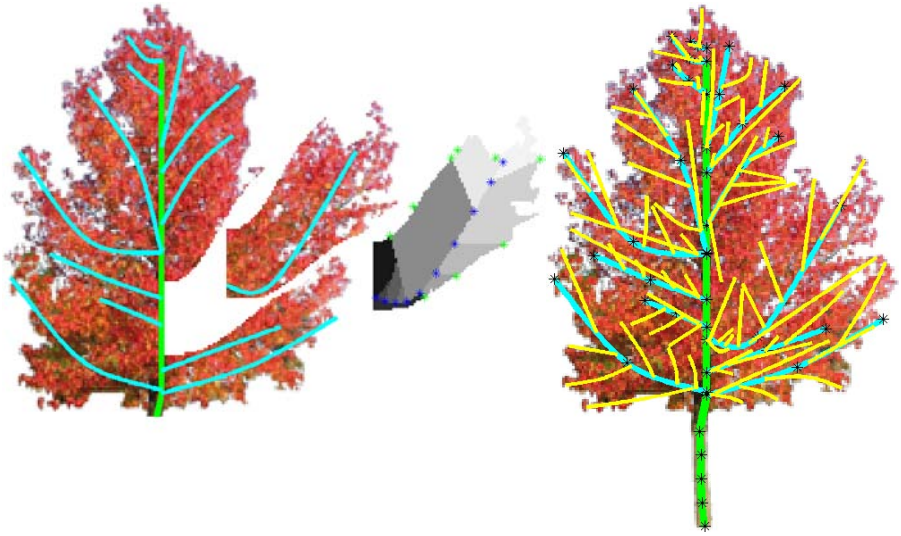

FIGURE 2.13 – À gauche, un calcul des branches du premier ordre. On applique ensuite l'algorithme de squelettisation sur les différentes cellules pour détecter de nouvelles sous-cellules. À droite, un système de branchage avec les branches du premier ordre en bleu et celles du second ordre en jaune.

convexe de l'image binaire. En chaque ligne de la forme binaire, on extrait le segment, intersection de l'image binaire avec la ligne. Chacun de ces segments est alors transformé en un cercle 3D. Son centre est le point 3D de coordonnées le centre du segment dans le repère image auquel on ajoute une coordonnée c et son rayon est la demie longueur du segment. c est choisi aléatoirement suivant une loi gaussienne centrée en 0 et d'écart-type le dixième de la longueur du segment. La normale à tous ces cercles est la direction du tronc. Le fait de ne pas prendre $c = 0$ comme valeur pour tous les cercles 3D permet de ne pas privilégier une direction par rapport à une autre. On souaite un résultat isotropique et on ne souhaite donc pas que tous les centres des cercles appartiennent à un même plan. On obtient alors un volume englobant le feuillage comme montré dans la Figure 2.14, page 42 pour le cas d'un liquidambar.

La profondeur de toutes les branches ne touchant pas l'enveloppe convexe 2D est modifiée de manière à ce que l'extrémité de la branche en 3D touche le volume englobant. La branche peut partir vers l'avant ou vers l'arrière. Le choix est fait de telle sorte que la somme des angles faits entre les différentes projections des branches (deux à deux) sur le sol soit maximale (voir la description de l'article de Okabe *et al.* [Okabe 2006] dans la section 1.2.2). Le problème avec cette méthode est que l'arbre en 3D semble réaliste lorsqu'il se projette dans la vue originale à

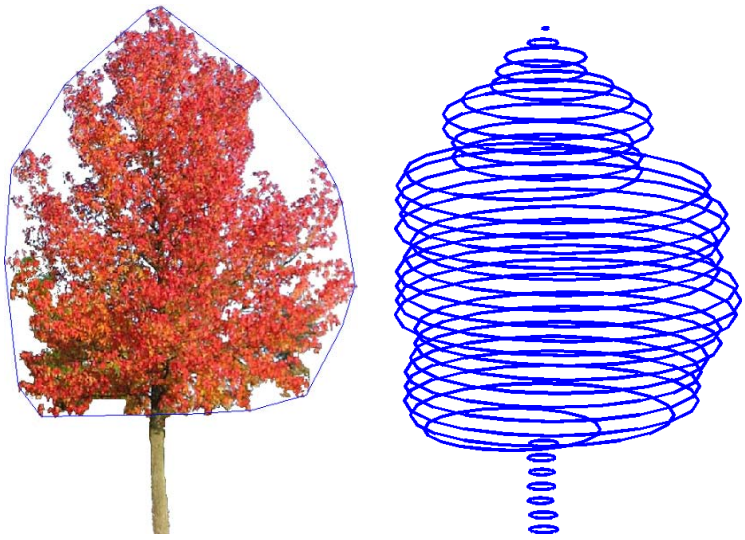

FIGURE 2.14 – À gauche, l'image originale d'un liquidambar avec son enveloppe convexe autour du feuillage. À droite, le volume englobant le feuillage en 3D.

l'image mais dans les autres vues, il peut sembler plat. On doit donc ajouter de nouvelles branches ne touchant pas l'enveloppe convexe afin que leur profondeur soit modifiée. L'arbre sera ainsi densifié et plus réaliste dans toutes les vues. Pour cela, on applique le même algorithme de squelettisation présenté ci-dessus à la forme binaire de la plante que l'on a préalablement érodée. On effectue cette opération avec plusieurs niveaux d'érosion afin de trouver des branches de différentes tailles comme on peut le voir dans la Figure 2.15, page 43.

Cette dernière étape permet d'obtenir un arbre 3D réaliste sous toutes les différentes vues. On peut voir un exemple pour le cas du liquidambar dans la Figure 2.16, page 43.

2.7 Algorithme

L'algorithme 1, page 44 détaille les différentes étapes de squelettisation. Cet algorithme est la première contribution de ce travail qui :

- généralise l'approche de Cornea (équation 2.2),
- utilise des modèles paramétriques pour la structure du branchage,
- traite les cas où toutes les branches principales sont dans un même plan mais aussi propose une généralisation pour les arbres 3D à partir d'une seule image.

Une fois cet algorithme utilisé sur l'image d'une plante, on obtient le squelette de cette plante, que ce soit en 2D dans le cas des vignes ou en 3D comme présenté

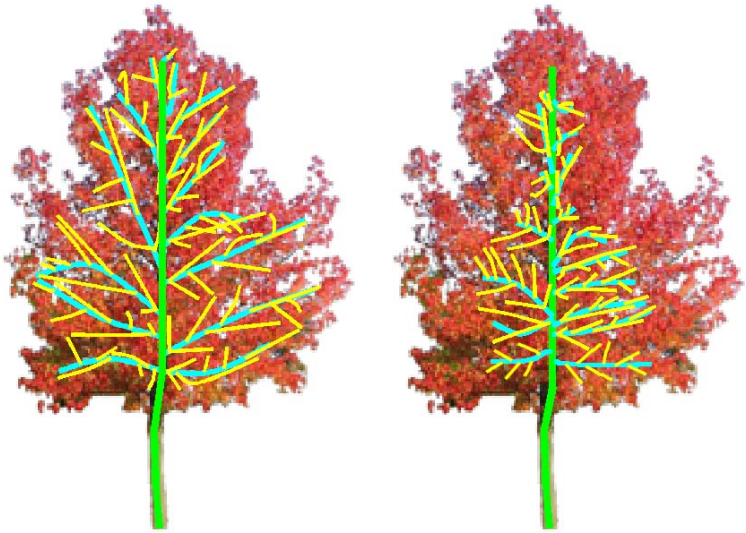

FIGURE 2.15 – À gauche, le squelette trouvé après avoir effectué une petite érosion à la forme binaire. À droite, le squelette extrait après avoir effectué une plus grande érosion.

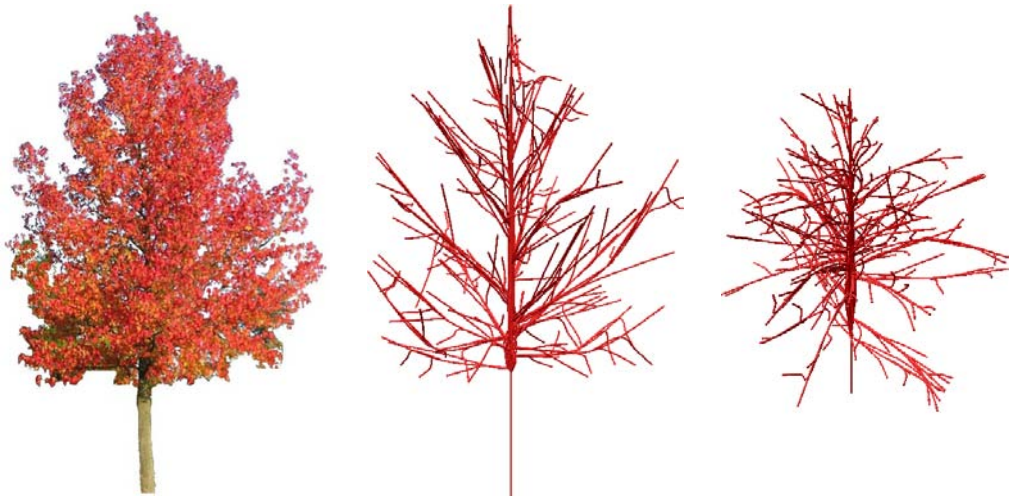

FIGURE 2.16 – À gauche, l'image originale d'un liquidambar. Au milieu, son squelette extrait avec notre méthode. À droite, le squelette vu d'une autre vue.

précédemment. Ce squelette contient seulement la topologie du squelette de la plante sous forme de courbes et non la modélisation du branchage de façon à avoir un rendu de branchage réaliste.

L'évaluation sur notre méthode d'extraction de squelettes s'effectuera de manière plus globale dans le prochain chapitre, après avoir modélisé les squelettes en 3D et après génération des feuillages. Pour générer le feuillage, nous avons besoin de quelques informations complémentaires en plus de la géométrie du squelette. Nous

Algorithme 1 *Entrées :* $\mathcal{B}$ la forme binaire d'une plante, n le nombre de branches que l'on souhaite trouver dans cette forme + des connaissances a priori sur les positions et les formes des branches. *Sortie :* Structure du branchage $\mathcal{S}_f$.

- Calculer le polygone pour simplifier le contour de $\mathcal{B}$ en utilisant l'algorithme DCE avec $3n$ sommets et en extraire uniformément le long les points $(\mathbf{c_k}, k=1..K)$.
- Placer les points initiaux aux découpes en fonction de la forme des branches voulue (voir section 2.3.2).
- Calculer les probabilités ρ_k pour chaque point $\mathbf{c_k}$ comme expliqué dans la partie 2.3.2.
- Tant que toutes les découpes n'ont pas été faites :
  Choisir un point initial à une découpe avec un des points $\mathbf{c_k}$ choisi en fonction des probabilités ρ_k et vérifier que la découpe satisfait les contraintes biologiques de la plante.
- Calculer la carte de probabilités $\mathcal{P}$.
- Calculer le champ de vecteur en utilisant l'équation 2.2.
- Extraire les points attracteurs dans le champ de vecteurs comme expliqué dans la section 2.4.
- Ajuster le squelette $\mathcal{S}_f$ en utilisant l'équation 2.4.
- Pour un squelette avec des branches d'ordres supérieurs voulues, appliquer de nouveau l'algorithme sur chaque cellule.
- Pour un squelette en 3D, ajouter des informations de profondeur aux branches comme détaillé dans la section 2.6.

calculons donc quelques attributs pour chacun des squelettes extraits.

2.8 Attributs du squelette extrait

Dans le prochain chapitre, on souhaite modéliser la plante en 3D à partir du squelette extrait. Pour cela, on calcule certains attributs. Le premier attribut est le rayon des branches qui est fonction de la distance au tronc et de l'ordre de la branche. Deuxièmement, on extrait une information d'épaisseur du feuillage en les différents noeuds qui composent ce squelette : en chaque noeud d'une courbe représentant une branche, on calcule la droite perpendiculaire à la courbe en ce noeud ; l'épaisseur est déterminée par la demi-distance du segment correspondant à l'intersection entre cette droite et la cellule dont a été extraite la courbe. Un indice de densité est également extrait en fonction du pourcentage de trous dans la forme binaire.

On peut alors tester la propriété de reconstruction du squelette en comparant la forme binaire originale à l'union de cercles de centres les points du squelette et de rayons les différentes épaisseurs du feuillage calculées (voir Figure 2.17, page 45). On

remarque alors un taux de recouvrement rarement au-dessus de 90% dans l'ensemble des cas testés. Ce pourcentage peut être amélioré en prenant en compte les trous dans le feuillage lors de la génération du feuillage en 3D explicité dans le prochain chapitre. De plus, les branches ne sont pas nécessairement centrées à l'intérieur de chaque cellule, la génération des feuilles avec des règles stochastiques va également permettre de potentiellement améliorer cette propriété de reconstruction.

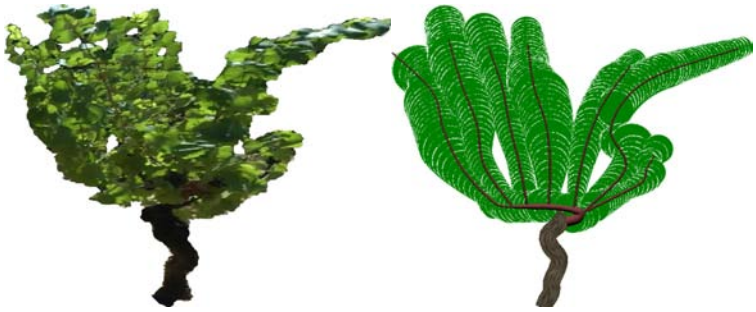

FIGURE 2.17 – À gauche une image originale d'un pied de vigne. À droite, une image composée de l'union des cercles centrés en les points d'un squelette extrait avec notre méthode et dont les rayons sont les épaisseurs du feuillage calculées.

Un dernier attribut en chaque noeud du branchage est la densité des feuilles. Celle-ci détermine la distance entre deux feuilles le long d'une branche. Plus l'indice de densité est élevé, plus cette distance est faible. Jusqu'à présent, nous n'utilisons que la forme binaire or nous avons remarqué que plus une zone est dense en feuilles, plus l'image est sombre relativement au reste de l'image. L'indice de densité attribué à chacun des noeuds est donc directement lié à la luminance dans l'image.

Dans le prochain chapitre, les entrées sont donc le squelette extrait dans ce chapitre ainsi que ses attributs. En sortie, nous trouvons un modèle 3D de la plante qui respecte au mieux l'image et les propriétés biologiques de la plante considérée.

Chapitre 3

Modélisation 3D de la plante

Sommaire

3.1 Modélisation de la plante

Nous avons montré dans la section précédente comment extraire un squelette d'une plante ainsi que tous les attributs définissant son feuillage. Ce squelette est non déterministe car certains paramètres sont des variables aléatoires suivant des densités de probabilité comme le choix des positions des découpes. Dans cette partie, nous nous intéressons à la modélisation de cette plante en utilisant les *L-systèmes* présentés dans la section 3.1.1.1. Nous montrons ensuite comment, par un procédé d'analyse-par-synthèse, nous pouvons choisir le modèle de la plante qui respecte au mieux le critère bayésien qui est développé dans ce chapitre.

Pour modéliser notre plante, nous avons fait le choix d'utiliser des L-systèmes. Les L-systèmes sont la représentation la plus classique pour la modélisation de plantes grâce à leurs nombreux avantages pour générer des objets ayant une géométrie et une topologie très complexes comme celles des plantes.

3.1.1 L-systèmes

3.1.1.1 Définition d'un L-système

Nous commençons par introduire la notion de *langage formel* qui est un langage composé d'un certain nombre de *mots*. Ces mots sont formés grâce à un *alphabet* qui est constitué d'un ensemble de symboles, généralement fini. Pour construire ces mots, on associe à ce langage une *grammaire formelle* qui définit les différentes règles de ce langage. Un *L-système* est une grammaire formelle qui permet, grâce à une étape d'interprétation géométrique supplémentaire, de modéliser le processus de développement d'un organisme. Aussi appelé système de Lindenmayer, le L-système a été inventé en 1968 par Aristid Lindenmayer. En 1990, Prusinkiewicz et Lindenmayer [Prusinkiewicz 1990] ont adapté les L-systèmes à la modélisation de plantes.

Pour cela, chaque partie de base d'une plante est représentée par un symbole w issu d'un alphabet $\mathcal{V}$ ou d'une constante issue de S. L'idée est de pouvoir construire cette plante en partant d'un objet initial simple $w_0 \in \mathcal{V}$ appelé axiome puis, grâce à des règles de production P, modéliser son développement afin de la reconstruire entièrement en plusieurs itérations. Une constante $s \in S$ ne changera pas d'état d'une itération à l'autre. Ce L-système est noté $\{\mathcal{V}, S, w_0, P\}$. Une représentation graphique de la plante peut alors être donnée via un interpréteur.

3.1.1.2 Exemple

On illustre, grâce à l'exemple de la suite de Fibonacci, ce que peut être un L-système. Dans ce cas, l'alphabet comporte deux symboles : $\mathcal{V} = \{A, B\}$. L'ensemble des constantes est vide : $S = \{\}$, l'axiome de départ est $w_0 = A$ et il y a deux règles de production : celle qui transforme A en B, que l'on note $(A \to B)$ et celle qui transforme B en AB notée $(B \to AB)$. La notation utilisée pour un tel L-système est la suivante :

$Fibonacci$

{

Alphabet :

$\mathcal{V} = \{A, B\}$

Axiome :

A

Règles de production :

$A \to B$

$B \to AB$

}

Il suffit maintenant de réécrire les symboles en se servant des règles de production à chaque nouvelle itération. Ainsi, pour les sept premières itérations, on a :

$n = 0$	A	1
$n = 1$	B	1
$n = 2$	AB	2
$n = 3$	BAB	3
$n = 4$	$ABBAB$	5
$n = 5$	$BABABBAB$	8
$n = 6$	$ABBABBABABBAB$	13
$n = 7$	$AABABAABABAABAABABAAB$	21
...	...	...

La dernière colonne indique le nombre de symboles à chaque génération. On remarque que l'on retrouve la suite de Fibonacci.

3.1.1.3 Interprétation

Les chaînes de symboles que l'on trouve après plusieurs itérations n'ont pas a priori d'interprétation géométrique. Cependant, si on leur donne une interprétation géométrique en 2D ou en 3D, cette chaîne de symbole correspondra à un objet qui

peut être aussi complexe qu'une plante. Ces règles d'interprétation peuvent par exemple être « $A \rightarrow$ dessiner un trait horizontal vers la droite de taille k ». Ces règles d'interprétation font partie de la *Turtle interpretation* qui vient de la tortue du langage de programmation Logo fonctionnant de manière similaire. On illustre cette interprétation graphique grâce à un exemple simple en 2D :

$Fractale$

{

Alphabet :

$\mathcal{V} = \{A, B, C, D\}$

Axiome :

A

Règles de production :

$A \rightarrow ABCA$

$B \rightarrow BDAB$

$C \rightarrow CADC$

$D \rightarrow DCBD$

Règles d'interprétation :

$A \rightarrow \_$

$B \rightarrow /$

$C \rightarrow \backslash$

$D \rightarrow |$

}

$n = 0$	A
$n = 1$	$ABCA$
$n = 2$	$ABCABDABCADCABCA$
$n = 3$	$ABCABDABCADCABCABDABDCBDABCABDAB$ $CADCABCADCBDCADCABCABDABCADCABCA$
$n = 4$	$ABCABDABCADCABCABDABDCBDABCABDAB$ $CADCABCADCBDCADCABCABDABCADCABCA$ ... $ABCABDABCADCABCABDABDCBDABCABDAB$ $CADCABCADCBDCADCABCABDABCADCABCA$
...	...

En remplaçant chaque symbole par son interprétation graphique correspondante, on obtient une suite de fractales convergeant vers le flocon de Koch (Figure

3.1, page 51). Toutes les règles explicitées ici sont appliquées de façon régulière et uniforme. Par conséquent, les objets générés sont trop réguliers pour ressembler de manière réaliste aux plantes que l'on trouve dans la nature.

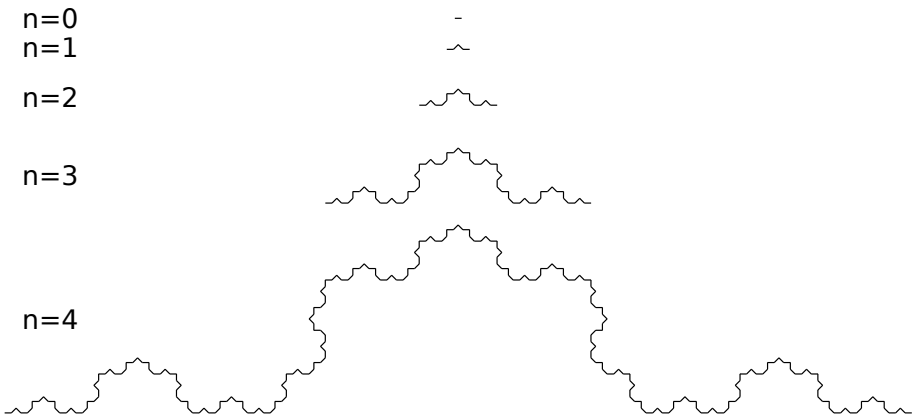

FIGURE 3.1 – Exemple d'interprétation d'un L-système pour dessiner le flocon de Koch.

3.1.1.4 Cas des plantes

Les règles d'interprétation peuvent être modifiées afin de modéliser des objets 3D et on peut les appliquer de manière plus stochastique. Une règle donnée à la tortue peut par exemple être « tourne par rapport à l'axe des z d'un angle qui est une variable aléatoire de valeurs comprises entre $\frac{\pi}{2}$ et π suivant une loi de probabilité (on peut choisir une loi uniforme ou donner plus de contraintes) ». Grâce à l'utilisation de ces L-systèmes, on peut donc modéliser tout type de plante comme montré dans la Figure 3.2, page 52. Dans ce cas, les règles d'interprétation sont des règles permettant la création de cylindres généralisés auxquels on peut appliquer une texture de branche. Lorsqu'on génère le feuillage, on peut aussi faire varier l'orientation des feuilles de manière aléatoire pour augmenter le réalisme.

Nous avons donc recherché un logiciel permettant d'utiliser ces L-systèmes pour modéliser des plantes : notre choix s'est porté sur L-py.

3.1.2 L-py

L-py [1] est un logiciel de modélisation utilisant les L-systèmes. Il est développé par l'équipe Virtual Plants du Cirad de Montpellier [2] et le langage de programma-

1. http ://openalea.gforge.inria.fr/wiki/doku.php ?id=packages :vplants :lpy :main
2. http ://www-sop.inria.fr/virtualplants/wiki/doku.php ?id=home

FIGURE 3.2 – Exemples de modélisations de différents branchages de plantes grâce aux L-systèmes. Grâce aux règles stochastiques utilisées ici pour la génération des branches, on obtient des résultats complexes et très réalistes car irréguliers.

tion utilisé est python. L'interface de ce logiciel est montrée dans la Figure 3.3, page 52.

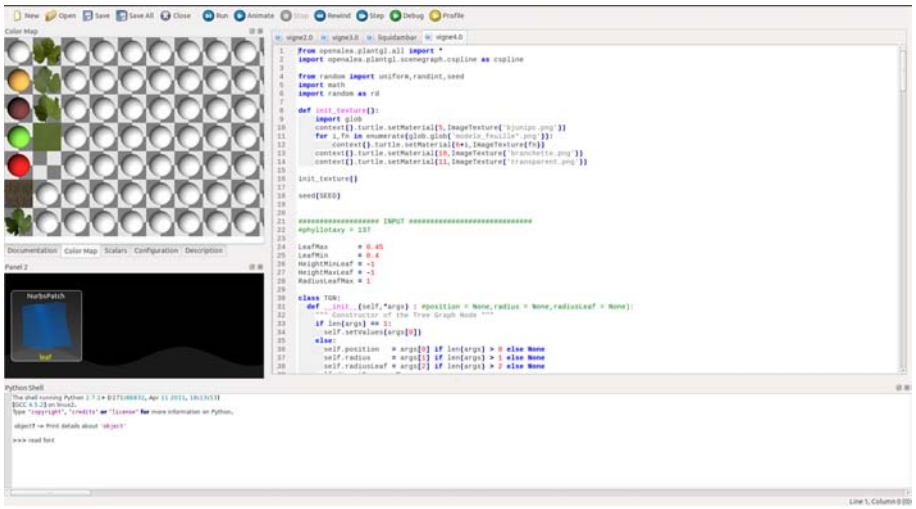

FIGURE 3.3 – Interface du logiciel L-Py.

L'utilisation des propriétés des langages de *programmation dynamique* améliore la modélisation des plantes grâce

- à une syntaxe simple permettant néanmoins des constructions de programmation haut-niveau,
- à une exécution du code simple, sans compilation,
- à la possibilité de réutiliser des modèles et de construire des modèles plus complexes.

Notre choix s'est donc porté sur ce logiciel et nous avons travaillé en collaboration avec les personnes l'ayant développé.

3.1.3 Modèle génératif

Une fois la structure du branchage extraite des images comme expliqué dans la section 2, une modélisation 3D de la plante doit être générée. Nous avons développé, grâce au logiciel L-py (section 3.3) un code qui est le plus générique possible afin de pouvoir modéliser un nombre de variétés de plantes le plus large possible. Ainsi, n'importe quelle structure de branchage peut être modélisée par notre programme en paramétrant correctement les différentes branches par des courbes à pôles. La modélisation peut être décomposée en deux parties : la génération du branchage puis l'ajout du feuillage dans un second temps.

Dans un but de réalisme et également pour éviter un trop grand nombre de paramètres, le modèle doit laisser suffisamment de libertés, principalement dans la génération des feuilles. Pour cela, le feuillage est généré de manière stochastique, c'est-à-dire que la position et l'orientation des feuilles sont choisies alétoirement par le modèle, en respectant bien sûr des contraintes qui assurent un aspect réaliste : une feuille tombe vers le bas et elle se trouve accrochée le long d'une branche. Ainsi, toutes les feuilles ne sont pas orientées dans le même sens, ce qui donne un effet réaliste à la modélisation finale.

3.1.3.1 Génération du branchage

Le branchage est généré de manière déterministe et récursive. Chaque branche est représentée par un cylindre généralisé le long d'une courbe à pôles. Celle-ci est une courbe B-spline avec un schéma d'interpolation local de degré 3 ([Piegl 1997]), paramétrée par un ensemble de noeuds 3D que l'on appelle les points de contrôle de la branche. Le nombre de points de contrôle est variable en fonction de la longueur et de la complexité de la branche. De chacun de ces noeuds peut partir une ou des nouvelles branches paramétrées de manière analogue. À chaque noeud est associé un rayon correspondant au rayon de la branche en ce point. Le rayon du cylindre généralisé est interpolé linéairement entre deux noeuds. Enfin, des textures extraites d'images réelles sont appliquées sur ces cylindres généralisés. Un exemple de modélisation de branchage est montré dans la Figure 3.4, page 54. Le code pour générer ce branchage est le suivant :

```
 [(0.000,0.000,0.000,4.000,0),(-0.750,10.000,0.000,4.000,0),(0.50,20.000,0.000,4.000,0),
[(-3.250,30.000,30.000,3.000,0),(-2.750,40.000,34.000,2.000,0),(-30.000,55.037,24.000,1.000,0)],
(0.309,33.693,10.916,3.313,0),
[(6.809,44.000,23.942,2.656,0.000),(12.763,88.536,30.001,1.292,0)],
(30.309,53.693,10.916,1.313,0)]
```

Chaque noeud comporte entre parenthèses les trois coordonnées en 3D, ainsi que le rayon de la branche et une cinquième valeur de densité explicitée dans la prochaine section. Pour indiquer qu'une nouvelle branche pousse à partir d'un noeud, il suffit d'ouvrir un crochet et de le refermer à la fin de cette nouvelle branche.

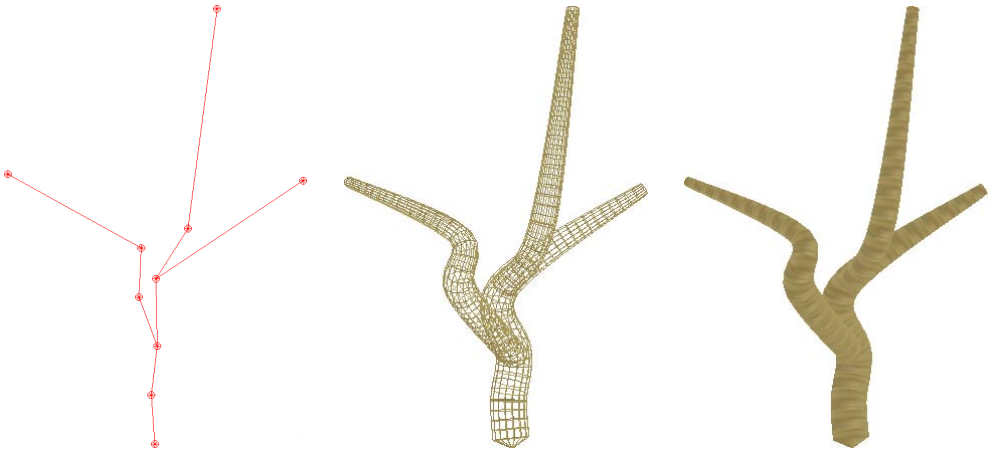

FIGURE 3.4 – À gauche, le squelette d'un branchage arbitraire avec les points de contrôle (ou noeuds) des courbes et les polygones de contrôle. Au milieu, les cylindres généralisés, modélisés en utilisant les rayons des branches attribués à chaque noeud. À droite, les textures de branches sont appliquées sur ce branchage.

Cette paramétrisation permet de représenter le branchage de n'importe quelle plante. En effet, tout arbre peut être discrétisé en noeuds 3D suffisamment proches les uns des autres en fonction de la qualité voulue. Ensuite, il suffit de supprimer un certain nombre de ces noeuds en trouvant un compromis entre la taille du fichier souhaitée pour modéliser l'arbre (donc le nombre de noeuds) et sa proximité du point de vue géométrique avec l'arbre d'origine. On montre la modélisation d'un branchage plus complexe que l'exemple présenté ci-dessus avec l'exemple du liquidambar dans la Figure 3.5, page 55.

3.1.3.2 Génération du feuillage

La première étape est la récupération de textures de feuilles à partir d'images réelles de la plante que l'on souhaite modéliser. Puis, à chaque noeud définissant la structure du branchage, on associe une nouvelle valeur correspondant au rayon du cylindre englobant les feuilles en ce point. Cette valeur est également interpolée linéairement entre deux noeuds. Enfin, un dernier indice de densité déterminant la fréquence des feuilles peut être attribué de manière locale ou globale à toute la plante. Ainsi, chaque point du branchage contient une liste d'attributs :

FIGURE 3.5 – À gauche, l'image originale d'un liquidambar. À droite, la modélisation de son squelette extrait avec notre méthode de squelettisation.

– le rayon de la branche r_b (explicité dans la section précédente),
– le rayon du feuillage r_f,
– la densité du feuillage d_f.

Il s'agit maintenant, à partir de ces attributs, de générer de façon aléatoire les feuilles dans le volume englobant le feuillage. d_f permet de déterminer la fréquence à laquelle on doit placer des feuilles sur la branche. Cette valeur correspond à la moyenne d'une loi gaussienne d'écart-type $\frac{d_f}{8}$. La distance entre la génération de deux feuilles sur une branche est tirée aléatoirement suivant cette loi gaussienne. Ainsi, plus d_f est faible, plus il y aura de feuilles sur la plante générée. Ensuite, afin que le volume soit totalement rempli de manière réaliste même lorsque r_f devient grand, une probabilité dépendant de r_f permet de générer de nouvelles branchettes à la place des feuilles, possédant elles-mêmes plusieurs feuilles. Ainsi, plus r_f est élevé, plus on a de chance de générer une branchette avec plusieurs feuilles à la place d'une seule feuille. On peut voir dans la Figure 3.6, page 56 un même squelette avec différents rayons des cylindres englobants et différents indices de densité. On obtient alors des plantes très différentes. Les valeurs des indices des rayons englobant les feuilles sont nulles sur le tronc ou parfois même sur les plus grosses branches dans le cas de certains arbres.

L'intérêt d'avoir un nombre minimal de paramètres est de limiter l'espace de recherche du modèle 3D en ayant un nombre restreint de paramètres à rechercher. Une fois notre modèle construit, il s'agit donc de sélectionner ces paramètres du

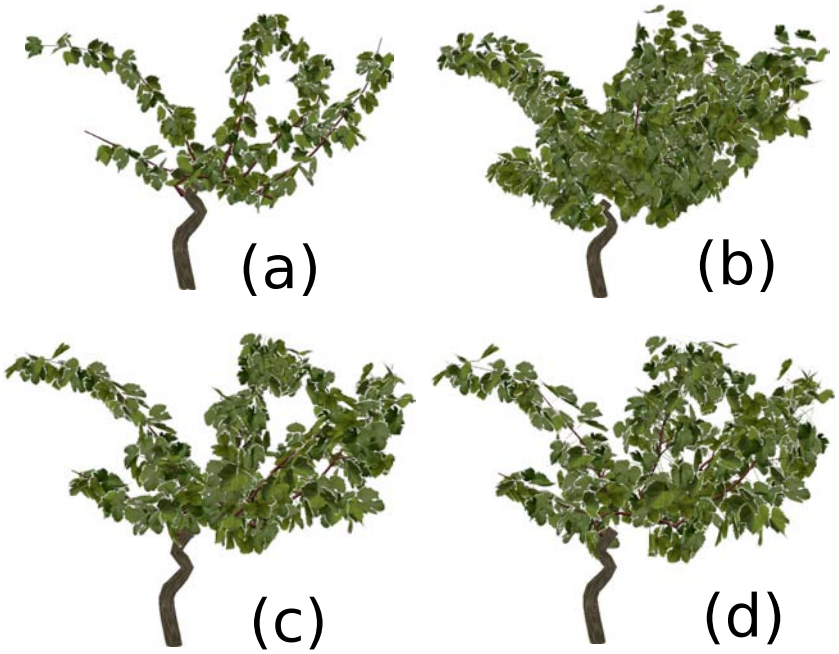

FIGURE 3.6 – Un même squelette de branchage modélisé en faisant varier le rayon englobant le feuillage r_f et l'indice de densité du feuillage d_f : r_f et d_f faibles (a), r_f et d_f plus élevés (b), r_f faible et d_f élevé (c) et r_f élevé et d_f faible (d).

modèle en fonction de l'image naturelle d'une plante que l'on souhaite reconstruire. Ce mélange de la synthèse d'une part avec l'analyse de l'image d'autre part nous a naturellement conduit à nous intéresser aux méthodes d'analyse-par-synthèse.

3.2 Analyse-par-synthèse

3.2.1 État de l'art sur l'analyse-par-synthèse

Le problème majeur en vision par ordinateur est de retrouver des propriétés de la scène photographiée à partir des propriétés de l'image acquise. C'est un problème inverse difficile dû au fait qu'il est très généralement fortement sous-contraint. En effet, certains objets similaires en 3D se projettent de façon très différente dans plusieurs images et inversement, des objets différents en 3D peuvent apparaître similaires dans des images. C'est ainsi que certains algorithmes donnant de très bons résultats sur des images de synthèse fonctionnent moins bien sur des images réelles plus complexes.

L'*analyse-par-synthèse* cherche à déterminer le scénario le plus probable selon lequel une image a été formée. Pour cela, des connaissances a priori sur la scène

(qui peuvent être modélisées sous forme d'*alphabets* ou de *bases d'apprentissage*) sont données à un modèle qui évalue et modélise la scène la plus probable. Une solution gloutonne serait de générer tous les scénarios possibles et décider le plus probable mais cela devient inenvisageable lorsque la taille du vocabulaire à partir duquel on construit cette modélisation est très grande. On distingue ici le langage *bas niveau*, proche du langage machine (directement interprétable par le processeur) du langage *haut niveau* qui nécessite donc plus d'intermédiaire entre ce qu'écrit le programmateur et le langage machine. On pourra également parler de processus bas ou haut niveau, de caractéristiques bas niveau comme les primitives telles que les pixels ou les courbes et enfin de caractéristiques haut niveau comme les visages, les piétons ou les voitures. En vision, le bas niveau s'apparente plus aux techniques du type détection de contours qui est une approche travaillant à un niveau très local avec des primitives très simples comme la plus simple de toutes : le pixel. En revanche, le haut niveau utilisera des primitives et des techniques plus complexes comme par exemple l'appel à des modèles pour la détection d'objets composés d'un grand nombre de pixels comme les visages. En observant que la vision bas niveau est souvent ambiguë mais rapide d'exécution alors que la vision haut niveau est rarement ambiguë mais beaucoup plus complexe et coûteuse, on peut se demander comment utiliser des processus bas niveau donnant rapidement accès aux bons modèles haut niveau ?

C'est la question que pose Yuille *et al.* dans [Yuille 2006]. Les auteurs font le rapprochement entre la vision par ordinateur et la vision humaine. Par exemple, regardons l'image très connue de la Figure 3.7, page 57.

FIGURE 3.7 – Image en noir et blanc extraite de [Gregory 1971].

Le cerveau humain ne voit au départ que des tâches noires sur du blanc et a beaucoup de mal à détecter la position du chien s'il n'a aucune information alors qu'il le détecte très rapidement à partir du moment où il sait qu'il doit chercher

un chien. Dans leur article, le but de ces auteurs est de retrouver sur une image le texte et les visages. Ils présentent un système où des caractéristiques déduites tout d'abord de processus bas niveau (donc rapides d'exécution), couplés à des a priori comme des règles de regroupement spatial, font des propositions sur la compréhension d'éléments de l'image au départ très simples, puis plus complexes (c'est ce qu'on appelle l'*approche ascendante*). Par exemple pour la détection et la reconnaissances de lettres dans l'image, Yuille *et al.* utilisent un détecteur de contours. Puis, ces contours sont regroupés en utilisant des propriétés de parallélisme ou de continuité, ce qui donne des premières informations sur la position ou l'identité de certaines lettres. On peut voir des propositions faites dans la deuxième image de la Figure 3.8, page 58 comme les positions où se trouvent potentiellement du texte ou des visages. On remarque que la détection d'un visage a été faite dans l'écorce d'un arbre. Ces propositions, qui limitent ainsi fortement l'espace de recherche (on sait qu'on ne recherchera pas de visages partout dans l'image mais juste aux endroits où des propositions ont été faites), sont ensuite validées par une *approche descendante* qui utilise les modèles adaptés pour vérifier la véracité ou non des propositions [Cavanagh 1999]. On voit les lettres et les visages détectés dans la quatrième image de la Figure 3.8, page 58. L'intérêt vient du fait que parfois, les propositions faites par des processus bas niveau sont suffisamment peu ambiguës pour être acceptées directement sans utiliser l'approche descendante, plus coûteuse en temps de calcul. Des indices de confiance sont calculés pour chaque proposition afin de déterminer si les étapes plus coûteuses sont nécessaires.

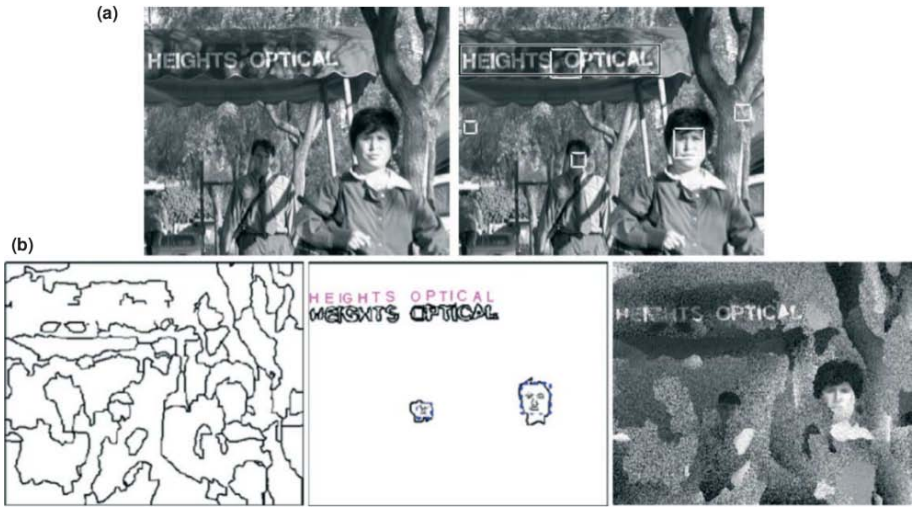

FIGURE 3.8 – Résultats de la méthode de Yuille *et al.* [Yuille 2006] (images extraites de l'article).

Un autre exemple similaire est l'article de Tu *et al.* [Tu 2003] dont le but est également de détecter les symboles, les visages et de segmenter les autres zones en fonction de leur couleur ou de leur texture. Dans cet article, la première phase nommée « *image parsing* » est définie comme la segmentation d'une image $\mathcal{I}$ en les différents motifs visuels que sont les différentes régions de couleur ou de texture similaires ou les objets (visages, textes). En sortie, la méthode donne un graphe hiérarchique $\mathcal{W}$ appelé « *parsing graph* » et qui détaille toutes les composantes de l'image. Les auteurs utilisent pour cela l'inférence bayésienne qui permet de déduire la probabilité d'un événement à partir des probabilités d'autres événements, déjà évaluées. Nous développerons un peu plus l'inférence bayésienne dans la prochaine section. La solution est le graphe $\mathcal{W}^*$ qui maximise la probabilité $p(\mathcal{W}^*|\mathcal{I}) = p(\mathcal{W}^*)p(\mathcal{I}|\mathcal{W}^*)$. Celle-ci combine une probabilité a priori $p(\mathcal{W})$ avec une fonction de vraisemblance $p(\mathcal{I}|\mathcal{W})$. Des modèles génératifs par *approche descendante* sont utilisés pour décrire comment les objets et les différents modèles de régions génèrent les intensités de l'image (et donc $\mathcal{I}$). Par exemple, le modèle génératif pour un symbole (lettre ou chiffre) est composé d'une limite extérieure, de 0, 1 ou 2 limites intérieures (par exemple le 0 et le 8 ont chacun une limite extérieure mais le 0 n'a qu'une limite intérieure alors que le 8 en a deux car il comporte deux trous dans sa forme) ainsi que d'une position et d'une transformation. Chacune des limites est une courbe modélisée par 25 points de contrôle. Il s'agit alors de calculer les paramètres de ces différents modèles qui représentent au mieux l'image. Par exemple, si un 3 se trouve dans l'image, il faudra trouver sa position ainsi que la transformation telle que la limite extérieure après transformation (pas de limites intérieures dans le cas du 3) s'ajuste le mieux aux contours du 3 dans l'image. Ils font des propositions par *approche ascendante* basées sur des caractéristiques *bas-niveau* pour restreindre l'espace de recherche de ces modèles en utilisant l'algorithme de DDMCMC (*Data Driven Markov Chain Monte Carlo*) [Tu 2002]. Dans l'article, à partir des détections du texte, des visages et des différentes régions segmentées, des images de synthèse sont ensuite recrées comme montrées dans la dernière colonne de la Figure 3.9, page 60.

On peut encore citer la méthode de Nair *et al.* [Nair 2008]. Dans leur article, ces auteurs définissent l'analyse-par-synthèse de manière théorique comme l'idée d'expliquer des données (par exemple une image) par l'ensemble le plus compact de causes qui les génèrent. Ils distinguent alors le modèle génératif qui spécifie comment les causes produisent les données, du modèle descriptif qui retrouve les causes à

FIGURE 3.9 – Résultats de la méthode de Tu *et al.* [Tu 2003] (images extraites de l'article). De gauche à droite, on peut voir les images originales, les couches de régions, les différents objets détectés et les images resynthétisées.

partir des données. Le but de leur article est de retrouver les paramètres du modèle descriptif d'une image à partir du modèle génératif et à partir d'un apprentissage préalable sur une base d'images. De manière plus pratique, les auteurs travaillent sur des images d'yeux. Dans ce cas, les données sont les images et les causes les générant sont la forme des yeux et la position de la pupille. Le modèle génératif est une boîte noire, c'est-à-dire qu'après construction du modèle, l'algorithme n'utilise aucune connaissance sur les détails internes du modèle génératif. Ce modèle 2D d'oeil a donc plusieurs paramètres comme la direction de la pupille ou l'ouverture de l'oeil paramétrée sous forme de courbes. Il s'agit alors d'extraire les paramètres à partir d'une image et de les fournir à la boîte noire qui va générer une image de synthèse de l'oeil. Dans la Figure 3.10, page 61, on peut voir des résultats de cette méthode sur des images d'yeux. Ces auteurs soulignent deux avantages à leur méthode. D'une part, elle fournit un moyen général d'incorporer des connaissances a priori via le modèle génératif pour l'apprentissage sur une base de données. D'autre part, les modèles génératifs sont un moyen naturel d'exprimer des connaissances a priori complexes.

3.2.2 Notre approche : analyse-par-synthèse dans le cadre bayésien

Nous appliquons donc ce principe de l'analyse-par-synthèse à notre cas de modélisation de plantes dans le cadre du formalisme bayésien. Ce dernier consiste à baser

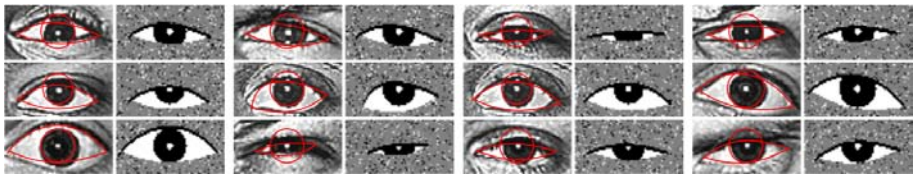

FIGURE 3.10 – Résultats de la méthode de Nair *et al.* [Nair 2008] (images extraites de l'article).

l'estimation des paramètres sur une distribution nommée *loi a posteriori*, qui mélange l'information portée par les données (via la *vraisemblance* ou terme d'*attache aux données*) avec une information *a priori*. Ainsi, différentes hypothèses sont faites par exemple sur le nombre de branches de premier ordre ou la densité foliaire. Dans notre méthode, l'approche ascendante consiste en une squelettisation qui part de l'image pour en extraire un squelette possible avec ses différents attributs. Cette approche travaille au départ sur des primitives bas niveau (comme le calcul de la carte de probabilités d'être un point contour qui raisonne au niveau pixellique) puis, monte à des niveaux plus complexes comme l'utilisation de modèles paramétriques pour extraire le squelette final). Les hypothèses sont finalement vérifiées et acceptées ou rejetées grâce au critère de sélection du modèle qui part du modèle généré en 3D et vérifie sa cohérence avec l'image (approche descendante).

3.2.3 Reconstruction de modèles virtuels

La première étape est de générer plusieurs modèles différents en fonction d'hypothèses faites sur la plante. Ces hypothèses sont faites à partir de statistiques connues sur la plante considérée. Notre modèle de plante est noté $\mathcal{M}$ et l'image $\mathcal{I}$. Il s'agit donc de maximiser la probabilité d'avoir un modèle $\mathcal{M}$ connaissant l'image originale $\mathcal{I}$, $p(\mathcal{M}|\mathcal{I})$ (loi a posteriori). Pour cela, en utilisant la formulation bayésienne, on décompose $p(\mathcal{M}|\mathcal{I})$ de la manière suivante :

$$p(\mathcal{M}|\mathcal{I}) \propto p(\mathcal{M})p(\mathcal{I}|\mathcal{M}) \tag{3.1}$$

$p(\mathcal{M})$ est la loi a priori calculée à partir de toutes les connaissances a priori que l'on a sur la plante et $p(\mathcal{I}|\mathcal{M})$ est une probabilité évaluant la différence entre la manière dont se reprojette le modèle $\mathcal{M}$ dans une image et l'image originale (terme d'attache aux données).

Cas des pieds de vignes

Notre algorithme ayant été développé à la base pour les pieds de vigne, nous

commençons par développer la méthode sur ces plantes. La transposition de connaissances a priori en une loi de probabilité sur les paramètres est loin d'être évidente. Dans le cas de pieds de vignes, nous avons récupéré un grand nombre d'informations de la part de nos experts. Sur les vignobles où nous sommes passés, nous connaissons le type de taille le plus fréquemment utilisé ou encore le nombre de rameaux moyen laissés sur chaque pied qui est modélisé, selon nos observations, par une loi normale de moyenne 5 et d'écart-type 2. De même, la densité des feuilles est paramétrée par un indice donné en entrée au modeleur L-Py et qui suit également une loi normale (la distance entre deux feuilles sur un rameau est assez stable). La squelettisation comporte une partie stochastique notamment dans le choix des découpes. Ainsi, on a vu que les découpes, passant proche d'un angle convexe au polygône simplifiant la forme du feuillage, ont une probabilité plus grande d'être générées. Cette probabilité doit également entrer en compte dans le calcul de notre loi a priori. Toutes ces connaissances nous permettent de calculer $p(\mathcal{M})$ comme un produit de probabilités (une pour chaque connaissance apportée qui définit un *paramètre* du modèle). Nous supposons donc que ces probabilités sont indépendantes les unes des autres. Cela pourrait être discuté mais nous n'avons pas remarqué de dépendance particulière entre les quatre termes de cette loi a priori :

- le modèle de taille,
- le nombre de rameaux,
- la densité des feuilles sur les pieds de vigne,
- la « pertinence » des découpes.

Autres cas

Dans les autres cas, nous n'avons pas d'informations directes. Nous avons donc observé différentes images pour chaque espèce afin de trouver un nombre de branches réaliste, des tailles et des densités de feuilles donnant de bons résultats visuels. Travaillant avec des experts dans le cadre viticole, il a été plus facile de trouver des informations plus scientifiques sur les pieds de vignes mais notre méthode peut fonctionner pour tous types de plantes en adaptant le choix des découpes (section 2.3.2) et la construction du squelette.

Par exemple dans le cas monopodial, les découpes respectent un certain angle avec le tronc. Toutes ces informations a priori ont été apportées grâce à une observation visuelle de différentes images d'arbres monopodiaux et sont donc empiriques mais elles prouvent qu'en apportant des connaissances réalistes, les modèles 3D générés sont satisfaisants.

3.2.4 Critère de sélection du modèle

Notre but est de reconstruire une plante biologiquement réaliste et dont la projection soit le plus proche possible de l'image originale. Le côté biologiquement réaliste est assuré en partie par la squelettisation et par la modélisation de la plante en 3D. En effet, l'utilisation de modèles paramétriques lors de la phase de squelettisation implique que les modèles de branchage aient la topologie souhaitée. De plus, la loi a priori $p(\mathcal{M})$ assure que les différents paramètres du modèle soient cohérents. La génération de plantes totalement aberrantes est donc éliminée. Le critère de sélection du meilleur modèle doit également assurer la similarité entre la projection de la plante modélisée et l'image d'origine grâce au terme d'attache aux données $p(\mathcal{I}|\mathcal{M})$ de l'équation 3.1. Pour cela, nous commençons par projeter chaque plante modélisée $\mathcal{M}_i$ avec un point de vue similaire à celui de l'image originale lors de l'acquisition. Travaillant sur une seule image et supposant que le point principal se trouve au centre de l'image, nous estimons une position théorique du point de vue qui se trouve sur un plan orthogonal au plan image et passant par le point principal. Nous obtenons ainsi une image pouvant être transformée en image binaire $\mathcal{I}_i$ (1 si il y a du feuillage ou des branches et 0 sinon). Cette image binaire est ensuite comparée à $\mathcal{B}$ (voir section 2.2) qui correspond à l'image binaire de l'image originale (1 si il y a du feuillage ou des branches et 0 pour le fond). En utilisant l'équation (3.1), le modèle $\mathcal{M}_{i_0}$ retenu est celui vérifiant :

$$\mathcal{M}_{i_0} = \arg\max_{\mathcal{M}_i} p(\mathcal{M}_i)(1 - \frac{\sum_j (\mathcal{I}_i(j) - \mathcal{B}(j))^2}{\#pixels}) \tag{3.2}$$

où $\mathcal{I}_i(j)$ (respectivement $\mathcal{B}(j)$) correspond au jième pixel de $\mathcal{I}_i$ (respectivement de $\mathcal{B}$) et $\#pixels$ correspond au nombre de pixels dans la boîte englobant les deux images.

On peut représenter les erreurs de reprojection du modèle sous forme d'une « *carte d'erreurs* » comme montré dans la Figure 3.11, page 64.

La Figure 3.12, page 64 montre les cartes d'erreurs de différents modèles 3D à partir desquelles on calcule la vraisemblance. En prenant en compte également le terme de la loi a priori comme on peut le voir dans l'équation 3.2, le modèle sélectionné est celui dont la carte d'erreurs est encadrée en rouge.

FIGURE 3.11 – À gauche, l'image originale d'un pied de vigne. À droite, la projection d'un modèle 3D reconstruit grâce à notre méthode. Au milieu la carte d'erreurs entre l'image de gauche et l'image de droite. Les pixels en gris correspondent aux pixels où les deux images ne se superposent pas.

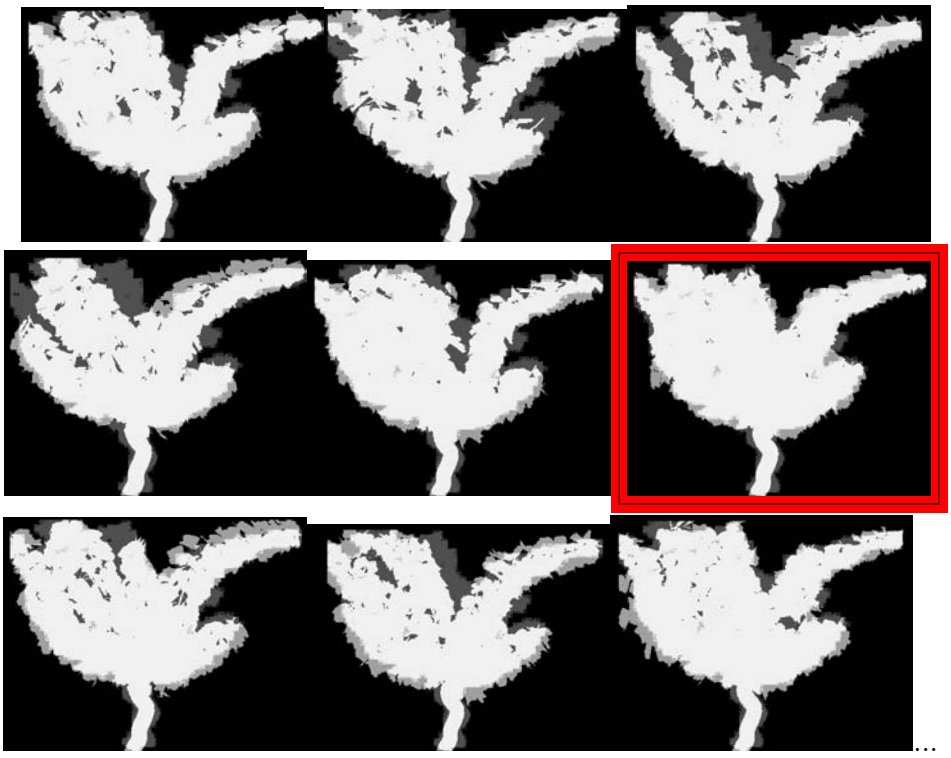

FIGURE 3.12 – Comparaisons entre les projections de différents modèles 3D et l'image originale. Le modèle sélectionné est celui dont la comparaison de sa projection avec l'image originale est encadrée en rouge.

3.2.5 Résultats et évaluations

Cette partie se décompose de la manière suivante : dans un premier temps, nous évaluons les modèles 3D d'arbres visuellement et en nous servant de notre critère d'erreur de reprojection $p(\mathcal{I}|\mathcal{M})$. Puis, nous évaluons de la même manière la modélisation des pieds de vignes. Dans ce cas, nous avons à notre disposition

des tracés d'experts. Nous évaluons par la suite chacun des critères du modèle : le nombre de rameaux, le choix du modèle de taille et la densité des feuilles pour terminer par une évaluation sur le temps de calcul nécessaire.

Résultats visuels dans le cas des arbres

Nous commençons à montrer quelques résultats visuels dans les Figures 3.13, page 65, 3.14, page 65, 3.15, page 66 et 3.16, page 66.

FIGURE 3.13 – Un exemple de modélisation dans le cas d'arbre monopodial. À gauche, une image originale d'un liquidambar. Au milieu, le modèle 3D obtenu par notre méthode avec le même point de vue que l'image originale et à droite avec une point de vue différent.

FIGURE 3.14 – Exemple d'un noyer. À gauche, l'image originale et à droite, le modèle 3D retrouvé grâce à notre méthode avec le même point de vue que celui dans l'image originale.

L'erreur du critère de reprojection est de 8.5% dans le cas du liquidambar et de 7.2% dans le cas du noyer. Nous rappelons que ce pourcentage correspond au

FIGURE 3.15 – Exemple d'un ginkgo. À gauche, l'image originale, au milieu le modèle 3D du branchage et à droite, le modèle 3D complet retrouvé grâce à notre méthode avec le même point de vue que celui dans l'image originale.

FIGURE 3.16 – Exemple d'un sapin. À gauche, l'image originale et à droite, le modèle 3D retrouvé grâce à notre méthode avec le même point de vue que celui dans l'image originale.

pourcentage de pixels dans l'image qui ne se superposent pas correctement entre les images originales et les images des modèles 3D reprojetés. Ces résultats semblent visuellement satisfaisants. Nous rappelons ici la difficulté de proposer une solution en 3D à partir d'une seule image 2D.

Résultats visuels dans le cas des vignes et intérêt d'un processus d'analyse-par-synthèse

Notre méthode a été testée sur la dizaine d'images soumises aux experts pour tracer les branchages. L'erreur du critère de reprojection à la fin du processus d'analyse-par-synthèse est de 6.9% en moyenne. Nous pouvons voir quelques résultats dans la Figure 3.17, page 67.

FIGURE 3.17 – Quelques exemples de modélisations de pieds de vignes. À gauche, des images originales et à droite des projections des modèles 3D obtenus par notre méthode avec le même point de vue que l'image originale.

Pour montrer l'intérêt d'utiliser un processus d'analyse-par-synthèse, nous avons tracé la courbe que l'on peut voir dans la Figure 3.18, page 68. Cette courbe représente, pour différentes tranches, le meilleur critère de reprojection diminuant en fonction du nombre de modèles 3D générés pour le deuxième exemple de pied de vigne vu dans la Figure 3.17, page 67.

On peut remarquer que plus on cherche à faire varier des paramètres comme le nombre de branches, la densité foliaire ou la distribution des feuilles, meilleur est le critère de reprojection et donc meilleur est le modèle 3D dans le sens où sa projec-

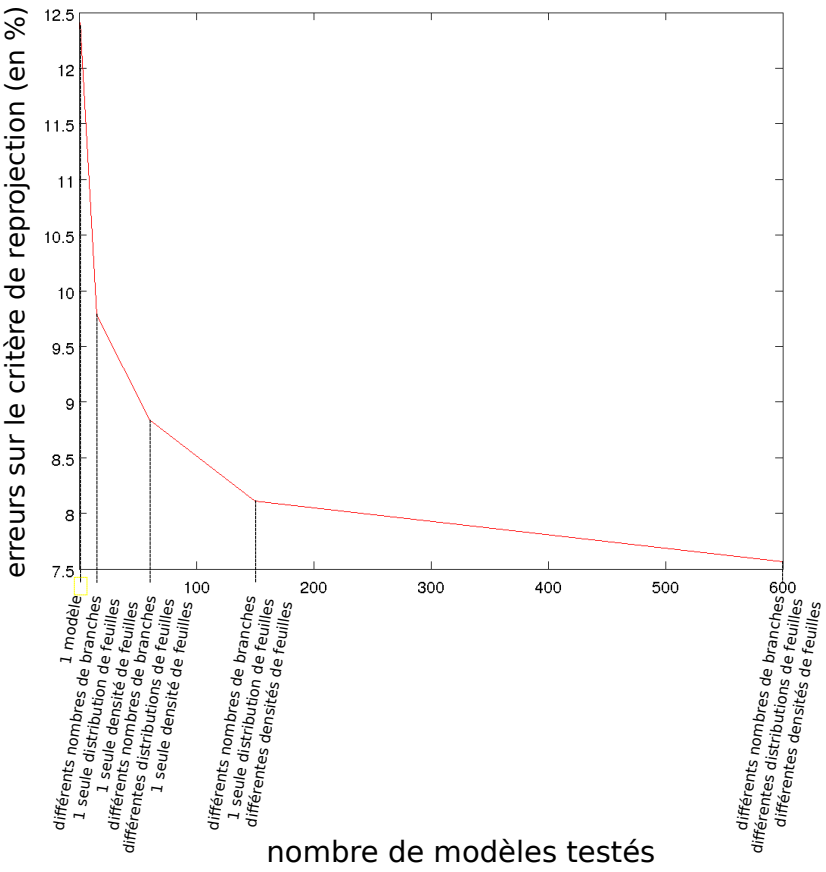

FIGURE 3.18 – Courbe représentant l'erreur de reprojection en fonction du nombre de modèles 3D générés et des différents paramètres testés.

tion est plus proche de l'image originale. Cette courbe descend très rapidement au début puis, de manière moins rapide à la fin. Bien évidemment, cette courbe sert à illustrer l'importance du processus de l'analyse-par-synthèse mais le modèle 3D sélectionné n'est pas nécessairement le dernier testé étant donné que tous les modèles sont testés indépendamment. Il ne s'agit pas d'un procédé itératif qui utilise les résultats de l'itération précédente pour affiner le modèle. Cependant, cette piste reste à explorer pour améliorer notre méthode. L'intérêt de ce processus par synthèse va être développé dans les sections suivantes. Nous montrons en particulier que plus on réduit l'espace de recherche dans lequel se trouve la solution (en apportant plus de connaissances a priori), plus la partie d'analyse-par-synthèse converge rapidement car moins de modèles 3D doivent être générés pour arriver au même résultat.

Tracés d'experts

Nous avons demandé à deux experts en viticulture (voir section 2) de tracer la structure du branchage sur une dizaine d'images de pieds de vignes. Ces dessins nous ont fourni des vérités terrain avec lesquelles nous avons pu comparer nos résultats.

Une possible analyse est la comparaison directe des squelettes trouvés par notre méthode avec ceux dessinés par les experts. Un exemple est montré dans la Figure 3.19, page 69.

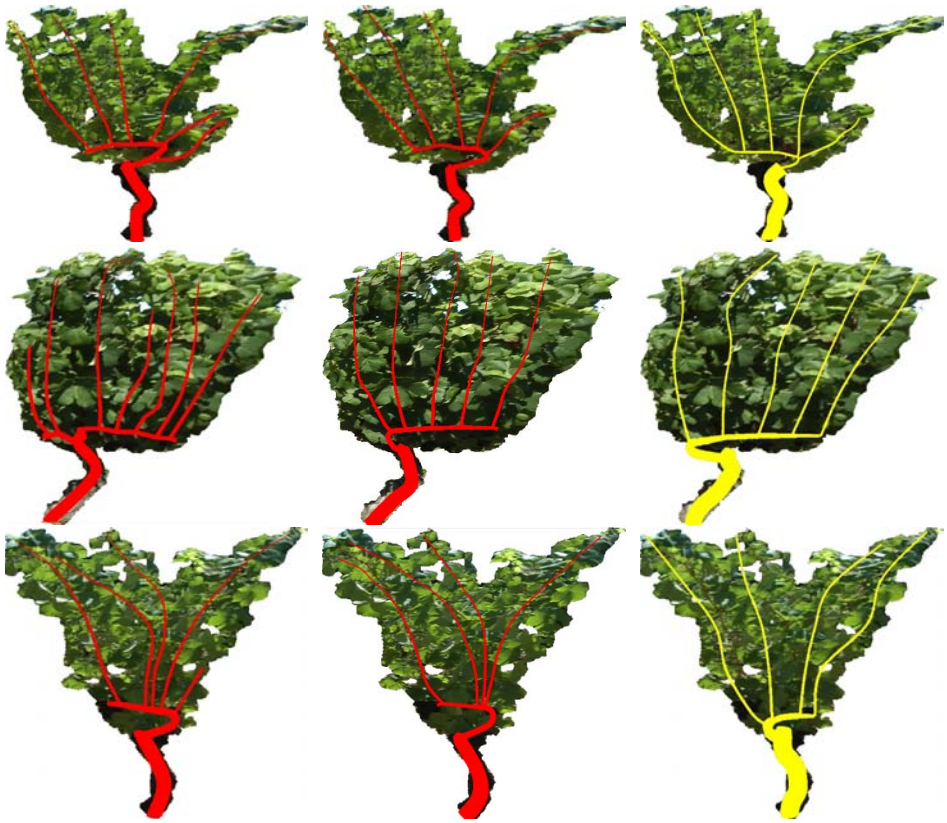

FIGURE 3.19 – Comparaison entre les tracés d'expert et notre méthode. À gauche et au milieu, on peut voir les tracés d'expert et à droite les structures de branchage trouvées par notre méthode. On remarque que pour le deuxième exemple, l'expert 1 trouve plus de branches que notre méthode. Cela est dû au fait que la forme est très compacte. Lorsque la forme n'est pas discriminante, notre méthode a tendance à remplir de branches uniformément. En revanche, l'expert 2 trouve le même nombre de branches que nous.

La bibliographie comparant des squelettes nous montre que la technique la plus efficace est la comparaison des graphes en utilisant une métrique adéquate comme expliqué par Sundar *et al.* [Sundar 2003] ou Brennecke *et al.* [Brennecke 2004]. Par exemple, on peut calculer une distance en associant un coût à chaque opération comme l'insertion ou la suppression d'un sommet et l'insertion ou la suppression

d'une arête. Cependant dans notre cas, la squelettisation est effectuée de telle manière que les graphes sont les mêmes (un tronc d'où part une baguette puis d'où poussent différents rameaux). Les seules différences peuvent être le nombre de rameaux ou le choix du modèle de taille. La comparaison de graphes dans notre cas n'est donc pas une bonne façon d'évaluer notre méthode.

Évaluation sur le nombre de rameaux

Nous avons remarqué que la génération d'un grand nombre de branches lors de l'utilisation de notre méthode de squelettisation pour un pied de vigne avec un feuillage relativement compact pouvait permettre de réduire l'erreur de reprojection. Par exemple, dans la Figure 3.20, page 70, il est facile de remplir la forme de feuilles en plaçant beaucoup de rameaux. C'est le défaut de beaucoup de méthodes qui cherchent simplement à avoir une forme remplie sans que le branchage interne soit biologiquement réaliste ou, dans le cas de plantes cultivées, conforme aux techniques de taille pratiquées. L'utilisation de nos connaissances a priori, données à notre modèle via la probabilité $p(\mathcal{M})$, s'avère alors très importante pour garder un nombre de branches raisonnable en pénalisant les squelettes avec un nombre de branches peu réaliste.

FIGURE 3.20 – Squelette trouvé par notre méthode avec un nombre de branches très élevé.

Nous avons maintenant voulu mesurer plus précisément l'influence de $p(\mathcal{M})$ sur l'erreur de reprojection et sur la rapidité pour trouver un modèle 3D satisfaisant, en particulier en ce qui concerne le nombre de rameaux. Nous avons choisi un pied de vigne ayant une forme complexe (troisième image de la Figure 3.23, page 74). Le nombre de rameaux trouvé par les experts est de 5. Puis, nous avons lancé notre méthode en obligeant dans un premier temps à ne modéliser que des pieds de vigne avec 5 rameaux en mettant $p(\mathcal{M}) = 0$ si ce n'était pas le cas. Puis, nous

avons recommencé cette expérience avec une probabilité de 14.3% d'avoir 4, 5 ou 6 rameaux et une probabilité de 9.5% d'avoir 3, 7, 8, 9, 10 ou 11 rameaux. Nous avons testé différentes distributions et différentes densités de feuilles et nous avons calculé l'erreur de reprojection pour chaque modèle 3D. Dans la Figure 3.21, page 71, nous avons tracé les histogrammes représentant le nombre de modèle 3D pour différentes tranches d'erreur de reprojection. À gauche, il n'y a que des modèles avec 5 rameaux et à droite, les modèles avec une probabilité différente d'avoir de 3 à 11 rameaux.

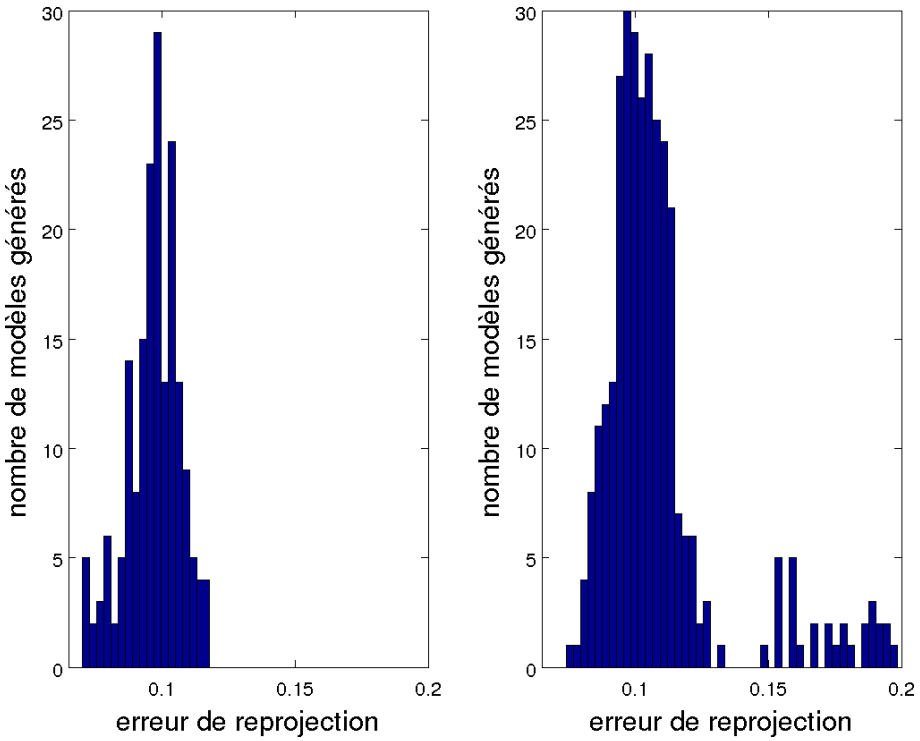

FIGURE 3.21 – Histogrammes représentant la répartition des erreurs de reprojection des modèles 3D. À gauche, tous les modèles générés possèdent 5 rameaux (le nombre de rameaux trouvé par les experts). À droite, les modèles générés peuvent avoir de 3 à 11 rameaux. L'image traitée est la troisième de la Figure 3.23, page 74.

On remarque alors que, si l'on force notre modèle, grâce à des connaissances a priori (en l'occurrence le nombre de rameaux sur le modèle que l'on cherche à reconstruire est de 5), alors les erreurs sont plus faibles et la méthode peut alors converger plus rapidement vers de bons résultats tout en testant moins de modèles. Les modèles avec un nombre aberrant de rameaux donne de très mauvais résultats. La Table 3.1, page 72 nous montre l'erreur de reprojection minimale et l'erreur de reprojection moyenne en fonction du nombre de rameaux générés.

Nombre de branches	3	4	5	6	7	8	9	10	11
Erreur de reprojection minimale	14.7	8.9	**7.4**	9.0	8.1	9.5	9.4	10.0	8.6
Erreur de reprojection moyenne	17.3	11.0	**9.4**	10.1	**9.4**	10.6	10.5	11.0	9.8

TABLE 3.1 – Table montrant les erreurs de reprojection minimales et moyennes des modèles 3D générés en fonction de leur nombre de rameaux. L'image traitée est la troisième de la Figure 3.23, page 74.

On voit bien qu'avec 3 rameaux, les résultats sont très mauvais, ce qui est logique vu la forme très complexe de l'image du pied de vigne traité. On voit également que l'on obtient les meilleurs résultats avec un nombre de rameaux égal à 5. En revanche, lorsque le nombre de rameaux augmente, l'erreur de reprojection tend également à augmenter mais pas très rapidement. En effet, les squelettes générés remplissent alors la forme du pied de vigne et donnent donc un résultat biologiquement peu réaliste même si l'erreur de reprojection augmente faiblement.

Ces exemples montrent l'importance de $p(\mathcal{M})$ qui permet d'obtenir des modèles de plantes réalistes grâce aux connaissances apportées. De plus, plus les connaissances apportées sont précises, plus la méthode trouvera des modèles proches de la réalité et plus le nombre de modèles testés pourra être réduit.

Nous avons alors évalué la pertinence de l'utilisation de $p(\mathcal{M})$ en comparant le nombre de rameaux trouvés avec notre méthode avec le nombre de rameaux trouvés par les experts comme vu dans la Table 3.2, page 72.

Numéro de l'image	1	2	3	4	5	6	7	8	9
Estimation du premier expert	3	6	5	5	6	5	3	7	7
Estimation du deuxième expert	2	6	4	5	4	4	2	5	4
Estimation avec notre méthode	2	5	4	5	6	6	3	5	6

TABLE 3.2 – Nombre de rameaux trouvés par les experts et par notre méthode pour quelques exemples de pieds de vignes.

Lorsque les experts sont quasiment d'accords entre eux, c'est-à-dire que la différence entre le nombre de rameaux qu'ils dessinent est inférieur ou égal à 1, notre méthode trouve également un nombre de rameaux égal ou de différence maximale égale à 1 avec celui de chacun des experts dans 92% des cas. De plus, pour chaque pied de vigne, le nombre de rameaux est égal ou de différence maximale égale à 1 avec au moins un des deux experts dans 100% des cas. En revanche, lorsque les experts ne sont pas d'accord, c'est-à-dire qu'il y a au moins deux rameaux d'écart

entre les deux dessins, notre méthode peut se tromper jusqu'à 2 rameaux maximum avec l'estimation d'un des deux experts. Cependant, dans ce cas, nous sommes à au plus un rameau d'erreur avec le deuxième expert. On peut supposer qu'il s'agit alors de cas difficiles vu que même les experts n'arrivent pas à se mettre d'accord, ce qui explique que notre méthode ne parvient pas à trouver le même nombre de rameaux.

Évaluation sur le choix du modèle de taille

Deux modèles de taille ayant été fournis à notre modèle (le guyot simple et le cordon de royat comme on peut le voir dans la Figure 2.11, page 39), il est intéressant de voir si le modèle de taille choisi est le bon. On peut voir dans la Figure 3.22, page 73 différentes manières possibles de rattacher les branches au tronc en utilisant différentes techniques de taille connues ou en inventant une nouvelle technique imaginée par nous et volontairement irréaliste (personne n'utilise une telle technique de taille). Sur cet exemple, les experts sont d'accord sur le fait que la technique utilisée est celle du guyot simple.

FIGURE 3.22 – En haut à gauche, on peut voir l'extraction de 5 rameaux et du tronc sur un exemple de pied de vigne. En haut à droite, les rameaux ont été attachés au tronc en utilisant une baguette via la technique de taille du guyot simple. En bas à gauche, la technique de taille du cordon de royat à été utilisée et en bas à droite, une technique « inventée », très peu réaliste, en rattachant directement les branches au tronc.

Pour évaluer le modèle de taille choisi par notre méthode, nous avons procédé à

des tests sur des images de pieds de vigne les plus différents possibles : un avec peu de rameaux, un en forme de V, un avec une silhouette très complexe et un dernier avec une forme très compacte comme montré dans la Figure 3.23, page 74.

FIGURE 3.23 – Images de pieds de vigne sur lesquelles on a évalué la capacité de notre méthode à choisir la bonne technique de taille.

Dans un premier temps, grâce à notre méthode de squelettisation, nous avons calculé cinq squelettes de branchage possible par image que nous avons reliés au tronc avec les trois techniques de taille présentées dans la Figure 3.22, page 73. Nous avons donc pu générer quinze modèles 3D avec trois densités de feuilles différentes soit 45 modèles 3D par image. La Table 3.3, page 75 montre les erreurs de reprojection moyennes et les erreurs de reprojection minimales pour chaque pied de vigne et pour chaque type de taille (ce qui correspond à la moyenne et au minimum des probabilités $p(\mathcal{I}|\mathcal{M})$).

On remarque que notre méthode privilégie la taille en guyot simple lorsque le pied n'est pas centré alors qu'elle préfère la taille en cordon de royat lorsque le pied est centré. En revanche, lorsque le pied à une forme en V, la taille "inventée" est privilégiée face aux deux autres. Les experts, eux, se basent surtout sur l'orientation des feuilles, ce que ne fait pas notre méthode qui se base seulement sur la forme du feuillage. Une fois encore, la connaissance a priori que l'on apporte via la probabilité $p(\mathcal{M})$ est très importante. On sait par exemple que dans le domaine de Fronton où la plupart de nos expérimentations ont été réalisées, plus de 8 pieds sur 10 sont taillés en utilisant la technique du guyot simple. En ajoutant cette connaissance, notre méthode retrouve la technique de taille estimée par les experts dans 77.8 %. C'est un résultat qui nous semble satisfaisant étant donné que l'on ne voit aucune branche sur nos images et que les systèmes de taille se ressemblent fortement.

Évaluation sur la densité des feuilles

Cette partie de notre évaluation se concentre sur l'influence du critère de densité des feuilles sur l'erreur de reprojection. Nous avons testé notre méthode sur deux pieds de vignes : un ayant une forme compacte et un autre ayant une silhouette plus

Numéro de l'image	1	5	6	8
Taille estimé par le premier expert	guyot	cordon	guyot	cordon
Taille estimé par le deuxième expert	guyot	guyot	guyot	guyot
moyenne des erreurs avec la taille en guyot simple	**10.3**	10.4	10.7	10.1
moyenne des erreurs avec la taille en cordon de royat	**10.3**	10.3	**9.4**	9.7
moyenne des erreurs avec la taille « inventée »	10.7	**10.1**	10.2	**9.3**
minimum des erreurs avec la taille en guyot simple	**8.5**	8.1	9.5	**7.2**
minimum des erreurs avec la taille en cordon de royat	8.8	6.9	**7.7**	8.3
minimum des erreurs avec la taille « inventée »	9.1	**6.8**	9.3	**7.2**

TABLE 3.3 – Table montrant l'influence du choix du modèle de taille utilisé dans le modèle 3D sur l'erreur de reprojection. La première ligne comporte les numéros des images traitées (vues dans la Figure 3.23, page 74) et la deuxième l'estimation de la technique de taille par nos deux experts (guyot ou royat, voir Figure 3.22, page 73). Les autres lignes montrent les erreurs moyennes et les erreur minimums de reprojection des différents modèles 3D générés comme expliqué dans le texte.

complexe avec plus de trous à l'intérieur. Pour chacune de ces images, nous avons alors lancé notre algorithme pour un même squelette trouvé avec différents indices de densité des feuilles. Les résultats sont montrés dans la Figure 3.24, page 75. L'indice de densité est représenté par la distance minimale entre deux feuilles sur le rameau. Plus cet indice est petit, plus la densité des feuilles sera donc importante.

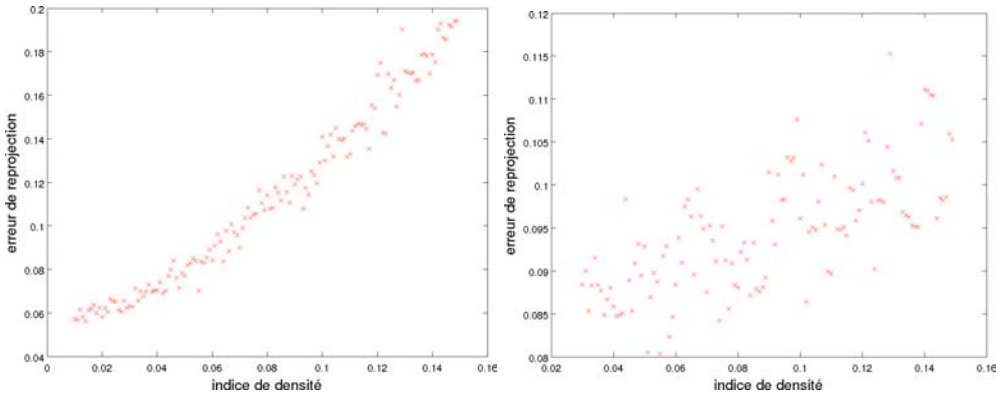

FIGURE 3.24 – Nuages de points représentant l'influence de la densité sur l'erreur de reprojection. À gauche, l'image traitée est la deuxième de la Figure 3.17, page 67. À droite, l'image traitée est la première de la Figure 3.17, page 67.

On remarque que selon l'image traitée, la densité des feuilles appliquée lors de la génération du modèle a plus ou moins d'importance. Lorsque la forme est compacte, plus la densité est élevée, plus l'erreur de reprojection diminue. Cependant, si la densité est trop importante, le modèle n'est plus réaliste car il y a trop de feuilles sur chaque branche. Dans ce cas, c'est encore une fois la probabilité $p(\mathcal{M})$ qui fait en sorte que la densité reste raisonnable. En revanche, lorsque la forme est plus complexe, l'influence de la densité des feuilles est moins marquée. Pour ces cas, l'erreur de reprojection a donc plus d'influence et notre méthode est donc plus fiable.

Évaluation sur le temps de calcul

Le premier traitement à effectuer est une partie de segmentation du feuillage. Celle-ci est détaillée dans la section 4.1. Elle comporte un suivi des points d'intérêt dans la séquence d'image puis, une segmentation basée couleur et profondeur et enfin, une rectification métrique. Elle a été programmée en C. Nous évaluons ici seulement son temps de calcul. L'algorithme de squelettisation 1, page 44 a été implanté en matlab. La partie analyse-par-synthèse comportant la génération du modèle 3D ainsi que la comparaison entre la projection du modèle 3D généré avec l'image ont été implantées en python (ce qui simplifie la tâche vu que L-Py est programmé en python). Les temps de calcul sont présentés dans la Table 3.4, page 76.

	Pour 1 modèle	
	Temps absolu	Pourcentage
Temps pour la partie segmentation	7 sec	15.6%
Temps pour la partie squelettisation	37 sec	76.5%
Temps pour la partie analyse-par-synthèse	4 sec	7.9%

TABLE 3.4 – Temps de calcul nécessaire à notre méthode pour chacune des différentes parties.

Dans un premier temps, on remarque que la partie squelettisation est la plus gourmande en temps de calcul, ce qui est logique vu que l'implantation a été faite en matlab. Cependant, la dernière colonne de cette Table montre le temps de calcul lorsque chaque partie de l'algorithme est appelé une fois. En réalité, pour retrouver un modèle 3D, la partie segmentation est appelée une seule fois, la partie squelettisation est appelée entre 1 et quelques dizaines de fois (ce nombre varie en fonction des connaissances plus ou moins importantes apportées préalablement) et la par-

tie analyse-par synthèse est appelée quelques dizaines de fois plus que la partie squelettisation (toujours en fonction des connaissances apportées).

Typiquement, pour les vignes testées, la partie squelettisation est appelée 15 fois (avec différents nombres de rameaux et les 2 tailles différentes) et pour chaque squelette généré, on teste 4 densités de feuilles et 10 distributions de feuilles différentes. On l'appelle donc 40x15=600 fois. La partie segmentation dure donc toujours 7 secondes soit 0.2% du temps total, la partie squelettisation dure 9 minutes et 21 secondes soit 19.4% du temps total et la partie analyse-par-synthèse dure 38 minutes et 48 secondes soit 80.4% du temps total. La partie la plus longue est donc bien la génération et la reprojection des modèles 3D. En effet, les modèles 3D possèdent parfois plusieurs centaines ou milliers d'éléments comme les feuilles représentées par des surfaces NURBS sur lesquelles on a plaqué une texture ou les branches qui sont représentées par des cylindres généralisés suivant des courbes à pôles et sur lesquels on a également plaqué une texture (voir section 3.1.3). Il pourrait être envisagé de modéliser beaucoup plus simplement les modèles 3D dans un premier temps car cela ne changerait que très peu la valeur de $p(\mathcal{I}|\mathcal{M})$ en approximant par exemple les feuilles par des cercles 3D de couleur unie. De plus, les branches peuvent également être générées beaucoup plus simplement. On verra en particulier dans le prochain chapitre que l'une des applications sert à estimer la densité foliaire. Dans ce cas précis, il n'est donc pas utile de modéliser des feuilles texturées, étape très couteuse en temps de calcul.

Une autre solution consiste à limiter l'espace de recherche et donc à réduire le nombre de tests en augmentant le nombre de connaissances a priori apportées. La Table 3.5, page 78 montre que sans connaissance a priori ($p(\mathcal{M})$ avec une forte entropie), le temps nécessaire pour avoir de bons résultats est très long car beaucoup de modèles doivent être testés. Si on diminue le nombre de tests, l'erreur de reprojection augmente. En revanche, si on apporte plus de connaissances, moins d'instances doivent être générées pour obtenir un résultat satisfaisant. Par exemple, si on sait que le nombre de rameaux est de 6, $p(\mathcal{M}) = 0$ pour tous les modèles dont le nombre de rameaux est différents de six. On ne teste donc pas ces modèles et le nombre de tests peut être divisé par un facteur de cinq.

Maintenant que nous avons évalué notre méthode, nous allons montré quelques applications dans lesquelles elles ont été utilisées au cours de ces années de thèse.

Analyse-par-synthèse	Nombre de tests	Erreur de reprojection	Temps
$p(\mathcal{M})$ avec une forte entropie	1	12.4%	4 sec
$p(\mathcal{M})$ avec une forte entropie	30	9.7%	2 min
$p(\mathcal{M})$ avec une forte entropie	120	8.8%	8 min
$p(\mathcal{M})$ avec une forte entropie	150	8.1%	10 min
$p(\mathcal{M})$ avec une forte entropie	600	6.9%	40 min
$p(\mathcal{M})$ avec une entropie faible	60	7.1%	4 min
Avec la connaissance sur la densité des feuilles	150	7.0%	10 min
Avec la connaissance sur le nombre de rameaux	120	6.9%	8 min
Avec les deux connaissances précédentes	30	7.0%	2 min

TABLE 3.5 – Temps de calcul de la partie analyse-par-synthèse et erreur de reprojection en fonction du nombre de tests effectués et des connaissances a priori apportées au modèle.

Chapitre 4

Applications

Sommaire

Nous allons présenter dans ce chapitre toutes les applications liées au projet VINNEO (section 4.2). Nous expliquons dans la section 4.1 comment nous segmentons le feuillage de nos images de vignes et, ensuite, nous détaillons les différentes expérimentations que nous avons faites durant cette thèse après rappel du contexte.

4.1 Cas des pieds de vignes

Lors de l'explication de notre algorithme sur la squelettisation dans la section 2, nous supposons que la segmentation du feuillage est connue. Nous montrons ici comment il est possible de segmenter nos images de pieds de vigne grâce aux connaissances sur le mouvement de la caméra lors de l'acquisition de nos images. Nous avons développé une méthode pour segmenter le feuillage en deux étapes : une première étape de segmentation basée sur la couleur et une autre basée sur la profondeur. À la suite de ces segmentations, une dernière étape de rectification métrique est nécessaire pour obtenir des images dans une vue fronto-parallèle au plan contenant le branchage principal des pieds de vignes. Nous rappelons ici l'hypothèse de planarité du feuillage et nous notons $\mathcal{Q}$ le plan approximant ce feuillage. En sortie, nous voulons une image binaire du feuillage du pied de vigne dans une vue fronto-parallèle au plan $\mathcal{Q}$.

4.1.1 Acquisitions des images

Les images que nous utilisons lors de nos expérimentations sont des vidéos acquises par un caméscope HD (1920x1080 pixels). Le mouvement, en caméra portée, a été le plus rectiligne possible, parallèle aux plans du sol et des rangs, en contraignant l'axe optique d'être lui aussi parallèle au plan du sol (voir Figure 4.1, page 81). La hauteur et l'orientation de la caméra peuvent donc être supposées constantes ; de plus, les paramètres intrinsèques de la caméra ont été pré-calculés, grâce à la procédure d'étalonnage décrite dans [Gurdjos 2005]. Dans un deuxième temps, ces caméras sont fixées sur un tracteur ou un quad (voir section 4.2.2). Dans ce cas, les hypothèses sur le mouvement et la direction de l'axe optique restent valides ; on prend soin de garder l'axe optique parallèle au sol.

4.1.2 Segmentation basée couleurs

La première étape consiste à segmenter le feuillage par une approche bayésienne avec un apprentissage préalable en estimant une densité gaussienne pour la couleur

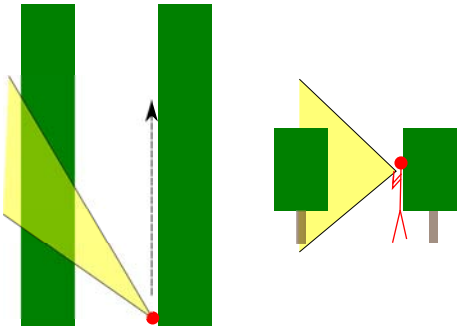

FIGURE 4.1 – À gauche, l'angle de vue de la caméra vu du dessus. À droite, l'angle de vue de la caméra vu du début de la rangée de vigne de côté. Les vignes sont représentées en vert, l'angle de la caméra en jaune, la direction de la caméra en noire et la personne tenant la caméra en rouge.

du ciel, une densité gaussienne pour la couleur du feuillage et une densité gaussienne pour la couleur du sol. À chaque pixel, on calcule la probabilité d'appartenir à chacune des classes. La probabilité la plus élevée désigne la classe du pixel. Pour améliorer cette approche simpliste, on peut ajouter des informations de textures et de position comme l'ont fait Hoiem *et al.* [Hoiem 2005] qui mélange des informations de couleurs, de textures et de positions pour segmenter leurs images. On sait par exemple que les pixels du feuillage se trouvent plutôt dans un triangle situé au milieu de l'image comme on peut le voir sur la Figure 4.2, page 82. Dans notre cas, l'ajout de la prise en compte de la texture n'améliore pas sensiblement les résultats car la couleur et la position des pixels suffisent à identifier le feuillage du ciel ou du sol.

4.1.3 Segmentation basée profondeur

Cette deuxième étape va permettre de segmenter le feuillage de la première rangée en filtrant le feuillage des rangées de l'arrière plan grâce à l'estimation de la profondeur de chaque pixel, effectuée grâce au suivi de points d'intérêt. Pour le suivi, nous utilisons l'algorithme *KLT* de Tomasi *et al.* [Tomasi 1991]. Dans nos expérimentations, les déplacements sont effectués à environ 5km/h avec une prise de vue à la fréquence de 50 trames par seconde. Dans ces conditions, nous suivons les points d'intérêt sur chaque trame et effectuons l'estimation de la profondeur toutes les 15 trames. 15 trames est un bon compromis d'après les tests que nous avons faits : d'une part, le nuage de points suivis reste suffisamment dense et d'autre part, les points suivis se sont suffisamment déplacés pour que l'estimation de la

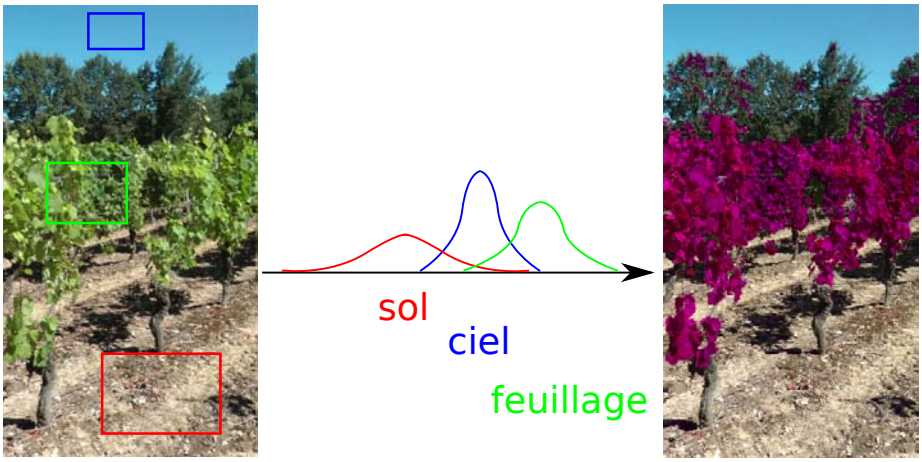

FIGURE 4.2 – À gauche, une image acquise dans une rangée de vignes. On donne un exemple de zones où l'on apprend les gaussiennes des différentes classes représentées schématiquement au milieu. À droite, la même image segmentée avec l'approche bayésienne basée sur les couleurs. Le feuillage est colorié en violet. On remarque que la couleur ne suffit pas à segmenter le feuillage de la vigne de la première rangée : le feuillage des rangées à l'arrière plan est aussi détecté.

profondeur soit correcte. Un point de l'image de départ est noté $\mathbf{p_{g_i}}$ et son correspondant 15 trames plus loin, $\mathbf{p_{d_i}}$. Puis, nous reconstruisons ces points suivis en 3D en se basant sur l'article de Zhao *et al.* [Zhao 2011]. Dans cet article, les auteurs supposent que la scène est composée de plusieurs plans. C'est aussi notre cas pour la très grande majorité de nos points car, une fois que l'on ne garde que les points suivis correspondant au feuillage grâce à la segmentation basée couleur, il ne reste que des points se trouvant sur des plans étant donnée l'hypothèse de planarité du feuillage des pieds de vignes. Les quelques points pouvant se trouver au fond n'affecteront pas le résultat. On sélectionne aléatoirement quatre points parmi les points suivis et on calcule l'homographie $\mathtt{G}$ entre ces quatre points et le plan image comme expliqué dans [Hartley 2003, p124] (une homographie 2D vers 2D ayant huit degrés de liberté, les deux coordonnées de quatre points suffisent pour l'estimer à condition que trois de ces points ne soient pas sur une même droite). Puis, à partir de cette homographie, on cherche toutes les paires de points $\{\mathbf{p_{g_j}}, \mathbf{p_{d_j}}\}_{,j\subset\mathcal{A}}$ vérifiant $\mathtt{G}\mathbf{p_{g_j}} \sim \mathbf{p_{d_j}} + \varepsilon$ où $\mathcal{A}$ est un sous-ensemble de $\{1, 2, ..., N\}$ si N est le nombre total de points suivis et ε est un vecteur de norme inférieure à un seuil fixé. Les paires de points vérifiant cette condition se trouvent alors sur un même plan (avec une petite marge d'erreur donnée par ε). Grâce à un procédé de type RANSAC qui va nous permettre d'assurer la robustesse et d'éliminer les données bruitées, on répète

cette opération un grand nombre de fois et on garde l'homographie correspondant à celle dont $\#\mathcal{A}$ est le plus grand. Une fois la matrice `G` retrouvée et étant donnée la matrice de calibrage `K` connue, il est possible d'extraire la matrice de rotation et la matrice de translation définissant le mouvement de la caméra entre les deux images en utilisant la décomposition de la matrice `G` proposée par [Triggs 1998]. Ainsi, on peut calculer les matrices de projection des deux images et donc reconstruire les points suivis en 3D par triangulation. Les images étant vues en perspective, on calcule une profondeur pour chacun de ces points en déterminant la distance qui les séparent du plan parallèle à celui des vignes et passant par le centre optique de la caméra. À partir de cette profondeur, on peut ne garder que les points suivis appartenant à la première rangée de vigne. À chaque pixel de l'image, on attribue la profondeur du point suivi le plus proche. Le nuage de points suivis doit donc être suffisamment dense. Un résultat est montré dans la Figure 4.3, page 83.

FIGURE 4.3 – À gauche, une image avec des points d'intérêt suivis. Les points en cyan sont les points les plus proches de la caméra. Les points bleus et rouges sont des points non souhaités, sélectionnés par la segmentation basée couleur qui sont ensuite filtrées par la segmentation basée profondeur. À droite, l'image finale segmentée avec la couleur et la profondeur. Le feuillage du rang de devant est colorié en violet.

Nous nous sommes intéressés à l'utilisation des superpixels lors de cette étape afin d'attribuer une profondeur à chaque pixel. Cependant, cette piste, détaillée dans l'annexe A, page 156, n'a pas abouti car les résultats n'étaient pas satisfaisants.

4.1.4 Rectification métrique

Lors de nos acquisitions, le recul dans les rangées de vigne n'est pas suffisant pour permettre de filmer les pieds de vignes de manière fronto-parallèle (c'est la raison pour laquelle sur l'image de gauche de la Figure 4.1, page 81, l'angle de la prise de vue n'est pas orthogonal au plan des vignes). De plus, l'utilisation d'une caméra avec un angle trop grand apporterait des distorsions que nous ne souhaitons pas. Le feuillage dans nos images est donc vu en perspective par rapport à $\mathcal{Q}$. Une rectification métrique est nécessaire. Elle consiste à transformer l'image grâce à une matrice de rectification notée $\mathtt{H}$ de manière à retrouver les angles et les rapports de longueur du plan $\mathcal{Q}$ et donc le parallélisme des images des droites de $\mathcal{Q}$ parallèles au sol. Étant donnés les paramètres intrinsèques de la caméra connus, on peut facilement calculer $\mathtt{H}$ comme expliqué dans [Hartley 2003, p50]. Pour cela, on commence par retrouver le point de fuite associé à la direction du mouvement de la caméra $\vec{\mathbf{v}}$. Le mouvement de la caméra étant une translation pure, ce point de fuite (noté $\mathbf{p_{f1}}$ dans la Figure 4.4, page 84) correspond à l'épipôle dans l'image et peut facilement être calculé à partir de la matrice fondamentale, elle-même calculée à partir des points d'intérêt suivis.

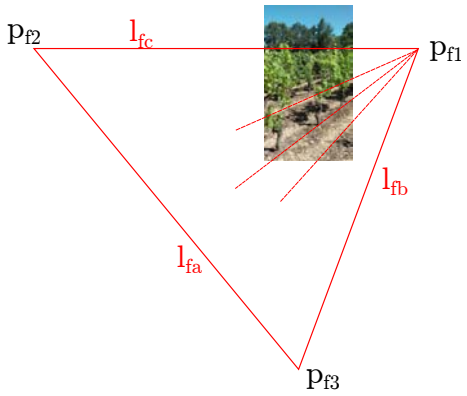

FIGURE 4.4 – Points de fuite et lignes de fuite sur une de nos images de vignes.

Ensuite, en supposant que la ligne d'horizon du sol se projette en une droite horizontale dans l'image $\mathbf{l_{fc}}$ (ce qui est vrai car l'axe optique est parallèle au sol), on retrouve les autres points de fuites associés aux deux directions orthogonales à $\vec{\mathbf{v}}$ et orthogonales entre elles. On utilise pour cela plusieurs fois la relation disant que si $\mathbf{l}$ est une ligne de fuite du plan contenant toutes les directions orthogonales à une direction $\vec{\mathbf{d}}$, alors $\mathbf{l} \sim \mathtt{K}^{-T}\mathtt{K}^{-1}\mathbf{pf}$ où $\mathbf{pf}$ est le point de fuite associé à la direction $\vec{\mathbf{d}}$. On peut donc retrouver dans l'ordre $\mathbf{p_{f3}}$, en déduire la droite $\mathbf{l_{fb}}$ qui passe par

$\mathbf{p_{f1}}$ et $\mathbf{p_{f3}}$ et finalement en déduire $\mathbf{p_{f2}}$. Ces trois points de fuite représentent les colonnes de `H` à des facteurs d'échelle près. On peut voir le résultat de l'image après rectification dans la Figure 4.5, page 85.

FIGURE 4.5 – Image rectifiée du pied de vigne vu dans la Figure 4.3, page 83. Le feuillage segmenté du premier rang est colorié en violet.

Cette rectification métrique détériore l'image. En effet, plus les branches sont éloignées de l'image, plus elles sont modifiées. Cependant, notre algorithme de squelettisation et de génération de modèle se base sur la silhouette du feuillage et par conséquent les résultats de notre algorithme restent très bons. Les connaissances a priori apportées permettent de pouvoir modéliser la plante à partir d'une image de faible qualité.

4.2 Le projet VINNEO

Le projet VINNEO est un projet FUI (Fond Unique Interministériel) qui regroupe une douzaine d'acteurs du sud-ouest de la France dans le domaine vitivinicole. Le projet VINNEO prévoit d'analyser les évolutions des goûts et des habitudes de consommation à l'échelle mondiale pour mieux adapter le profil sensoriel des vins de 4 cépages anciens du Sud-Ouest en faisant appel à des techniques d'élaboration innovantes. Le logo de ce projet est montré dans la Figure 4.6, page 86 et plus de détails peuvent être trouvés sur le site de l'INP de Toulouse [1].

Notre participation à ce projet concerne la caractérisation de parcelles de vigne à partir d'images prises au sol. Il s'agit d'identifier les différences qui peuvent exis-

1. http ://www.inp-toulouse.fr/fr/partenaires/innover-avec-l-inp-toulouse/toute-l-actualite–partenariats-et-transfert/l-inp-toulouse-impliquee-dans-vinneo.html

FIGURE 4.6 – Logo du projet VINNEO.

ter entre les feuillages de différentes parcelles (vigueur plus ou moins importante, homogénéité intra-parcellaire). En effet, il est important pour les caves coopératives de déterminer dans quelle gamme de vin seront affectées les grappes provenant de telle ou telle parcelle. Notre contribution majeure est le calcul de certains indices comme celui de la *SECV* (*Surface Exposé du Couvert Végétal*) qui est un indice important donnant de précieuses informations sur la qualité de la grappe de raisin. Il correspond à la somme des surfaces de toutes les feuilles d'un pied de vigne. Des études ont par exemple montré que pour faire mûrir correctement un kilogramme de grappes, le pied de vigne a besoin de 0.5 à 2 mètres carrés de feuilles selon les cépages, les conditions météorologiques ou le type de vin produit (vin de pays, grand cru...). Le rapport feuilles-fruits est calculé en divisant la SECV par le rendement par hectare. Il est donc utile pour les vignerons de connaître la SECV sur les différentes parcelles pour connaître le nombre de grappes à garder sur chaque pied de vigne. Aujourd'hui, cet indice est souvent estimé « à l'oeil ». Nous cherchons donc une manière automatique d'extraire cet indice à partir des images. De plus, cet indice permet de voir si une parcelle est homogène ou non, c'est-à-dire si la SECV de l'ensemble de ses pieds est stable, ce qui est important pour savoir si elle produira un vin de bonne qualité ou non. On peut ainsi caractériser une parcelle pour aider dans le choix de la cuve où seront pressées les grappes provenant de telle ou telle parcelle lors des vendanges.

4.2.1 Calcul de la surface foliaire exposée

Dans le cadre du projet VINNEO, les vignerons sont intéressés par le calcul de la SECV du pied de vigne. Cet indice, très important pour déterminer la vigueur d'une parcelle ou même son homogénéité comme expliqué précédemment, dépend de la largeur l et de la hauteur h du feuillage ainsi que de la discontinuité d qui correspond aux trous dans le feuillage :

$$SECV = \frac{2h + l}{e}(1 - d)$$

où e correspond à l'écartement entre les rangs.

Pour calculer cet indice, nous avons développé un logiciel qui prend en entrée une vidéo filmée à l'intérieur d'une rangée de vignes dans les conditions décrites dans la section 4.1.1. En sortie, ce logiciel est capable d'estimer la SECV tout au long de la rangée. Pour cela, la discontinuité d est estimée après segmentation et rectification métrique comme expliqué dans les sections précédentes. Le rapport du nombre de pixels appartenant au feuillage sur le nombre de pixels contenus dans le rectangle englobant le feuillage et correspondant à la taille des vignes donne le pourcentage de feuilles. On peut alors facilement déterminer la discontinuité comme étant 100% moins le pourcentage de feuilles. On affiche une courbe représentant le pourcentage de feuilles le long d'une rangée dans la Figure 4.7, page 88. On remarque pour cet exemple que la parcelle est très hétérogène car le pourcentage varie beaucoup. Cela indique également qu'il existe plusieurs pieds manquants dans cette parcelle. En ce qui concerne la hauteur h, elle est facilement estimée grâce à la reconstruction des points 3D détaillée dans la section 4.1.3. On connaît l'espacement entre pieds et entre rangs e qui nous permet de retrouver le facteur d'échelle. Pour la largeur l, on utilise celle donnée par le vigneron. En effet, lors de la taille, les machines utilisées sont réglées sur une certaine largeur. On peut également récupérer de la même manière la hauteur pour valider notre algorithme.

4.2.2 Expérimentations

Pour valider notre algorithme, des expérimentations ont été réalisées au cours des étés 2010, 2011 et 2012. Le but de ces expérimentations a été l'acquisition de vidéos à l'intérieur d'une rangée de vignes.

En 2010, le but a été d'acquérir les premières vidéos qui ont permis la création de notre logiciel. Les vidéos ont été réalisées à pieds dans 3 parcelles de vignes très différentes en ce qui concerne la vigueur des plantes afin de tester les différentes variabilités possibles auxquelles on peut être confronté. De plus, certaines rangées de vignes ont été filmées face au soleil ou à l'ombre.

Ensuite en 2011, nous avons acquis une base de données plus importante. Le but était de valider notre logiciel. De plus, on devait déterminer le nombre de trames nécessaires par seconde pour que le suivi de points s'effectue correctement ainsi que la vitesse maximale à laquelle filmer dans les parcelles. Les premiers essais sur engins

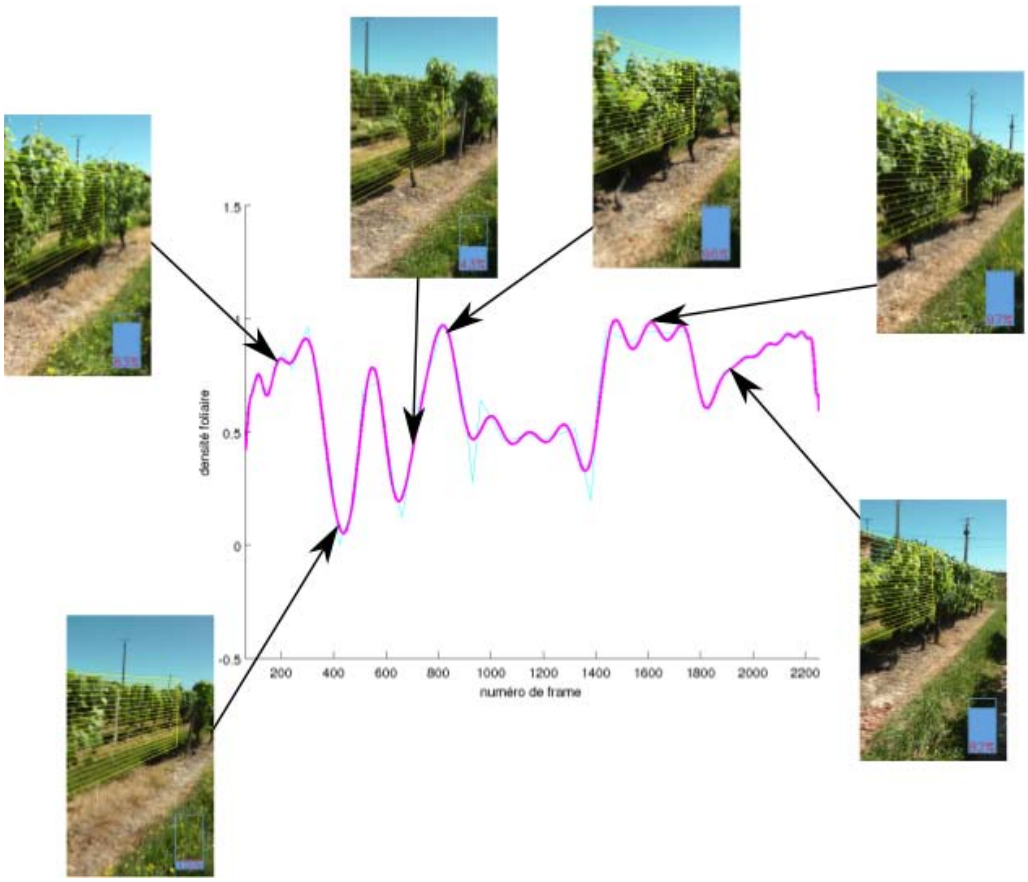

FIGURE 4.7 – Courbe représentant le pourcentage de feuillage le long d'une rangée de pieds de vignes. On remarque les quadrilatères sur chaque image indiquant la zone où a été calculé l'indice de discontinuité.

motorisés (quad et tracteur) ont été réalisés. Cependant, la caméra n'était pas fixée et devait encore être tenue à la main. Les expérimentations ont été effectuées sur une vingtaine de parcelles dépendant de la cave coopérative de Fronton. Les tests ont montré que la vitesse de 5km/h avec 50 trames par seconde convenait parfaitement.

Enfin en 2012, les premiers essais avec caméra fixée sur un tracteur ont été effectués. Le but était alors de déterminer si les vibrations dues au moteur ou au terrain affectaient la performance du logiciel. Les tests ont été effectués encore une fois à Fronton mais aussi dans le Gers dans le domaine de château de Mons. Deux nouvelles caméras ont été fixées sur le capot d'un tracteur comme montré dans la Figure 4.8, page 89. Le choix final s'est porté sur la caméra *Contour* permettant de filmer en haute résolution avec 30 trames par seconde. Par conséquent, en vue des tests effectués l'année précédente, le tracteur devait circuler moins rapidement.

L'indice le plus important à estimer étant celui de la discontinuité, nous avons

FIGURE 4.8 – Images du dispositif de prises de vues pour l'acquisition des vidéos dans les parcelles de vignes. Le boitier gris est la caméra *Imajbox* et sur sa droite, une caméra *Contour*, toutes deux prêtées par la société *Imajing*.

filmé deux fois la même rangée dans un sens puis dans l'autre. Le logiciel est censé donner les mêmes résultats de discontinuité sur les deux faces vu que les trous dans le feuillage doivent se voir des deux côtés. Nous avons donc tracé les deux courbes représentant le pourcentage de feuilles sur l'ensemble de la rangée (pour rappel, cela correspond à 1 moins la discontinuité) sur une face, puis sur l'autre. L'une des deux courbes a bien sûr été retournée pour qu'elles coïncident. Sur les quatre parcelles traitées l'erreur moyenne entre les paires de courbes a été de 3.89%. Un exemple de courbes est donné sur la Figure 4.9, page 90. On remarque bien que les trous dans le feuillage sont détectés dans les 2 sens même si parfois les trous apparaissent plus importants dans un sens que dans l'autre.

À terme, le but est d'équiper les tracteurs avec cet outil. En effet, les viticulteurs passent plusieurs fois par an dans les parcelles pour traiter ou tailler les vignes. Il serait alors intéressant d'utiliser ce temps passé dans les vignes pour ramener des images qui pourront ensuite donner un grand nombre d'informations sur les parcelles et éventuellement sur leur évolution au cours de la saison.

4.3 Modélisation à l'échelle de la parcelle

L'installation d'une caméra sur le tracteur permet donc de collecter une base de données géoréférencées qui, une fois traitées, permettent de fournir un grand nombre d'informations aux viticulteurs. La Figure 4.10, page 90 montre une capture d'écran du logiciel Google Earth sur lequel a été surimprimé le parcellaire de Fronton ainsi que les rangées de vignes filmées pendant la campagne 2011.

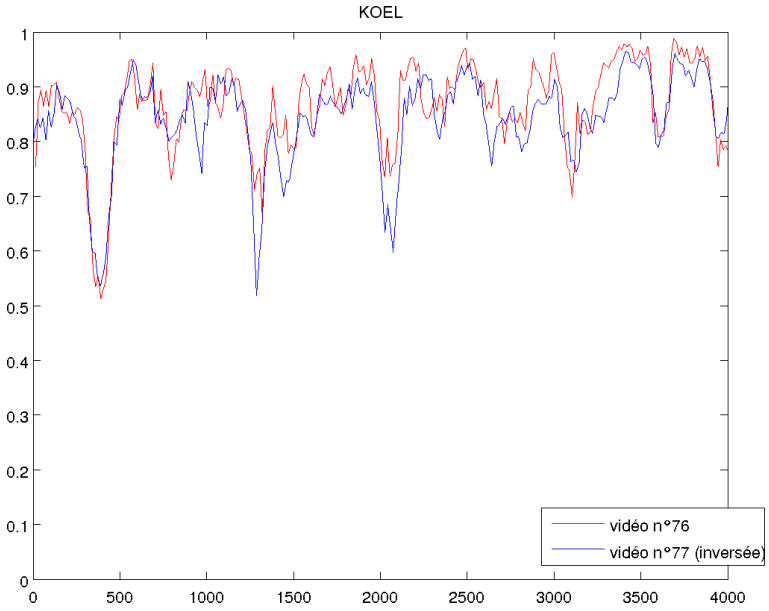

FIGURE 4.9 – Courbes représentant le pourcentage de feuilles (1 moins la discontinuité) sur une même rangée nommée KOEL. Le tracteur est passé dans les 2 sens. La courbe rouge montre les résultats de notre logiciel dans un premier sens (vidéo n˚76) et la courbe bleue montre les résultats sur la deuxième face (vidéo n˚77).

FIGURE 4.10 – Carte du parcellaire de Fronton surimprimé sur les photos de Google Earth. Chaque parcelle est délimité en rouge. Les rangées sur lesquelles nous sommes passés lors de la campagne 2011 sont représentées en jaune. En cliquant dessus, on peut avoir diverses informations comme la largeur de la vigne.

Le but final serait de mettre à jour une carte aussi souvent que possible afin de

connaître les parcelles ou même plus précisément les zones d'une parcelle où certains traitements doivent être effectués. Ces traitements pourraient varier en fonction des indices calculés comme par exemple la SECV qui donne une bonne indication de la vigueur de la vigne. Ces outils seront de plus en plus utiles car les viticulteurs ont des exploitations de plus en plus grandes en moyenne. Ils n'ont donc plus le temps de passer dans toutes leurs vignes eux-mêmes et sont donc obligés de recruter des salariés qu'ils doivent manager. L'intérêt pour eux est donc d'accéder directement à des informations depuis leur ordinateur sans avoir à se déplacer sur le terrain afin d'optimiser l'ensemble des tâches à effectuer.

Conclusion, limites et perspectives de la partie I

Dans cette première partie, les principales contributions apportées sont :

- une nouvelle méthode de squelettisation permettant d'extraire un branchage 3D réaliste à partir d'une seule image binaire d'une plante,
- la construction de modèles 3D paramétrables par un nombre réduit de paramètres dans le but de générer des plantes très facilement,
- une nouvelle approche d'analyse-par-synthèse permettant la sélection du meilleur modèle 3D parmi une multitude de modèles.

Ces travaux ont donné lieu à plusieurs articles :

- Reconstruction de modèles virtuels de vignes à partir d'images, *Jérôme Guénard, Géraldine Morin, Frédéric Boudon, Pierre Gurdjos, Vincent Charvillat* publié et accepté en présentation orale à **ORASIS** en 2011,
- Realistic Plant Modeling from Images based on Analysis-by-Synthesis, *Jérôme Guénard, Géraldine Morin, Frédéric Boudon, Vincent Charvillat* publié et accepté en présentation orale à **Curves and Surfaces** en 2012,
- Reconstructing Plants in 3D from a Single Image using Analysis-by-Synthesis, *Jérôme Guénard, Géraldine Morin, Frédéric Boudon, Vincent Charvillat* publié et accepté en présentation orale à **ISCV** en 2013.

La principale contribution de cette première partie consiste en notre méthode de squelettisation qui généralise la méthode de Cornea. Elle permet, à partir d'une forme binaire d'une plante, de retrouver une structure de branches possible. Cet algorithme peut être appelé de manière récursive dans le but d'avoir plusieurs niveaux de branchages. Dans le cas des vignes, nous avons pu montrer les performances de notre méthode qui retrouve des branches similaires à celles trouvées par des experts en viticulture.

Nous avons ensuite montré comment construire un modèle de synthèse 3D à partir de connaissances a priori sur une plante. Ce modèle est défini par divers paramètres comme la position des noeuds des différentes branches du squelette ou d'autres paramètres sur le feuillage comme la densité ou le volume englobant les feuilles. Un procédé d'analyse-par-synthèse sélectionne ensuite le meilleur modèle 3D grâce à un critère qui compare la projection des modèles 3D avec l'image originale

d'une plante. Cette combinaison de l'analyse et de la synthèse permet de coupler les connaissances biologiques d'une part avec les informations fournies par l'image afin d'obtenir un modèle 3D biologiquement réaliste qui s'ajuste au mieux à l'image d'entrée.

Plusieurs résultats visuels montrent les performances de nos méthodes. Nous avons testé notre algorithme sur des pieds de vignes et également sur des arbres monopodiaux qui sont composés d'un tronc vertical d'où partent toutes les branches de premier ordre. Nous avons validé nos résultats de modèles 3D de vigne grâce à des tracés d'experts faits sur les images originales. Un des plus grands intérêts de notre méthode est qu'elle fonctionne à partir d'une seule image qui peut être de résolution très basse et sur laquelle aucune branche n'est visible.

Dans le chapitre 4, nous avons montré des applications potentielles de ces travaux dans le cadre du projet VINNEO finançant en partie cette thèse. Ce projet s'intéresse à la culture des pieds de vigne. Les partenaires de ce projet étaient particulièrement intéressés par la segmentation « foliaire » des vignes afin d'en déterminer leur vigueur et l'homogénéité des parcelles. Nous avons donc développé une méthode de segmentation du feuillage des pieds de vigne à partir de vidéos acquises depuis un tracteur circulant dans les rangées. Cette méthode a été évaluée et plusieurs résultats ont été présentés. Vinovalie, qui regroupe les caves coopératives de Fronton, Cahors, Técou et Rabastens et qui a été à l'origine du projet VINNEO, est intéressé par les résultats que nous avons obtenus. Un projet permettant d'étudier la faisabilité de placer une caméra sur les tracteurs des vignerons, récupérer les données, les traiter en vue d'obtenir une base de données sur tout le parcellaire des caves coopératives de Vinovalie est envisagé.

Les limites de notre méthode de reconstruction se situent principalement dans le cas monopodial. En effet, nous inférons des informations 3D à partir d'une image 2D et donc même si les résultats sont très satisfaisants lorsque le modèle 3D est vu depuis une vue proche de celle lors de la prise de vue, ils peuvent être un peu moins pertinents lorsque le modèle 3D est vu de côté. En effet, le critère de sélection du modèle ne se base que sur cette vue originale. Le calcul de branches sur une forme érodée de la plante permet de reconstruire de nombreuses branches allant dans toutes les directions mais notre critère ne les prendra pas en compte. Il est envisagé d'inclure une loi dans la probabilité $p(\mathcal{M})$ afin de privilégier les arbres où les branches donnent un aspect le plus réaliste possible également dans les vues de

côté.

Plusieurs perspectives sont à l'étude. Tout d'abord, l'extension de notre méthode à l'ensemble des plantes. Nous avons présenté ici une méthode pour les plantes où toutes les branches principales se trouvent dans un même plan et les arbres monopodiaux où toutes les branches de premier ordre partent d'un tronc vertical. Pour cela, les découpes ne seraient plus des courbes en 2D mais des surfaces en 3D. Il faut commencer par trouver un volume englobant le feuillage à partir d'une image 2D. La méthode proposée pour le cas monopodial peut être utilisée. Elle pourrait également être améliorée en utilisant par exemple les ombres dans l'image du feuillage. Celles-ci peuvent nous donner des informations plus précises sur la forme 3D du volume englobant les feuilles en indiquant les zones où existent potentiellement un trou. On peut ensuite choisir par exemple des découpes en forme de cônes. Quelques tests ont déjà été effectués. Les premiers résultats semblent très encourageants et sont montrés dans la Figure 4.11, page 95.

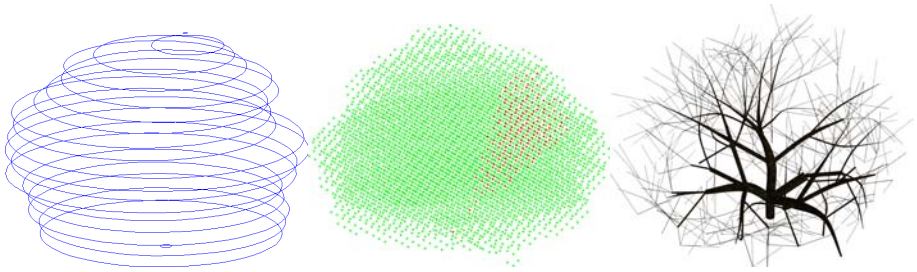

FIGURE 4.11 – À gauche, un volume englobant possible du noyer de la Figure 3.14, page 65. Au milieu, une découpe en forme de cônes qui pourraient être faite dans un nuage de voxels remplissant le volume englobant. À droite, une modélisation possible du branchage.

Plusieurs images pourraient également être utilisées en vue d'améliorer la forme du volume englobant le feuillage. Pour le cas monopodial, l'algorithme de squelettisation pourrait alors être appliqué sur les diverses vues pour obtenir un système de branchage plus réaliste avec toutes les vues. Cependant, un des intérêts de notre méthode est justement le fait qu'elle n'utilise qu'une seule image sans aucune branche visible. Dans le cas où l'on aurait plusieurs images disponibles, d'autres méthodes existent déjà comme celles détaillées dans la section 1.2.2.

Une des perspectives très intéressante serait de construire le modèle non plus à partir de connaissance a priori apportées par des biologistes mais automatiquement par apprentissage à partir d'une base de données de plusieurs centaines d'images. Il

serait également plus facile d'effectuer cet apprentissage directement sur des modèles 3D. En effet, apprendre sur des images est compliqué car cela revient à extraire des informations sur le squelette sans a priori, ce que nous faisons dans cette thèse mais avec a priori ! Il serait alors envisageable de travailler sur des images prises en hiver et donc sans feuilles pour effectuer cet apprentissage.

En ce qui concerne la partie analyse-par-synthèse, il pourrait être intéressant d'utiliser des algorithmes de type MCMC (*Markov Chain Monte Carlo*). Cependant, cette méthode, pour donner de bons résultats, implique qu'entre deux itérations, le modèle 3D puisse n'être modifié que localement. Ce qui n'est pas le cas pour les modèles 3D générés avec L-Py. En effet, l'orientation des feuilles étant aléatoire, même la petite modification d'un noeud d'une branche peut affecter l'ensemble du feuillage. C'est la raison pour laquelle nous avons fait le choix de générer une multitude de modèles 3D et de sélectionner celui répondant le mieux au critère. Nous pourrions modifier notre modèle de façon à utiliser les méthodes de type MCMC et ainsi, converger vers une solution, ce qui éviterait la génération d'un grand nombre de modèles 3D, très coûteux en temps de calcul. Cependant, la qualité du résultat final ne devrait pas être nécessairement améliorée, cette qualité dépendant essentiellement de la manière dont le modèle a été construit.

Deuxième partie

Reconstruction à l'échelle du fruit

Introduction de la partie II

Après avoir créé des modèles de plantes, nous cherchons à créer des modèles de fruits. Ayant travaillé sur les pieds de vigne au départ, nous avons développé une méthode pour reconstruire une grappe de raisin à partir d'au moins deux vues. Nous avons fait l'hypothèse que chaque baie était un *ellipsoïde de révolution*. La méthode obtenue peut être plus généralement adaptée à tout type de fruits assimilables à une *quadrique de révolution*.

Dans cette partie, nous allons donc nous intéresser à la reconstruction de quadriques de révolution à partir d'une ou plusieurs vues. La reconstruction de quadriques et, en général, la reconstruction de surfaces 3D est un problème très ancien en vision par ordinateur qui a donné lieu à de nombreux travaux. Nous commençons par rappeler quelques généralités sur la géométrie projective des quadriques puis sur la vision par ordinateur. Puis, dans la section 6.2, un petit état de l'art présente les méthodes déjà existantes s'intéressant au vaste sujet de la reconstruction de quadriques ou de manière plus générale de surfaces quadratiques. Nous détaillons dans la section 6.4 un premier algorithme permettant de retrouver les images des foyers principaux d'une quadrique de révolution à partir d'une vue « calibrée », c'est-à-dire dont on connaît les paramètres intrinsèques de la caméra, comme nous l'avions introduit dans [Gurdjos 2009]. Puis, nous détaillons comment utiliser ce résultat pour reconstruire, à partir d'un schéma de triangulation linéaire, tout type de quadriques de révolution à partir d'au moins deux vues dans la section 6.5. Enfin, toujours en utilisant le résultat de la section 6.4, nous montrons comment il est possible de retrouver la pose 3D d'une quadrique de révolution dont on connaît les paramètres à partir d'un seul contour occultant dans la section 6.6. Les dernières sections évaluent les performances de nos méthodes et montrent quelques applications possibles.

Chapitre 5

Quadriques et vision par ordinateur

Sommaire

5.1 Géométrie projective

Il est indéniable que c'est grâce à l'apport de la géométrie projective que la vision par ordinateur a atteint une maturité qui a permis la conception d'algorithmes fiables et performants permettant de résoudre ses principales tâches, notamment celles de reconstruction 3D. Le but des deux prochaines sections n'est pas de donner un cours complet sur la géométrie projective et la vision par ordinateur mais de définir certaines bases utilisées par la suite dans cette partie. De nombreux ouvrages très complets existent déjà et pour avoir des informations complémentaires, nous invitons le lecteur à se reporter aux ouvrages [Semple 1952] et [Hartley 2003].

5.2 Notions élémentaires

L'espace projectif. Soit $\mathbb{K}_{n+1}$ un espace vectoriel de dimension $n+1$. Soit la *relation d'équivalence* $\sim$ définie sur $\mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$ par

$$\forall \mathbf{X}, \mathbf{Y} \in \mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\} \qquad \mathbf{X} \sim \mathbf{Y} \Leftrightarrow \exists \lambda \in \mathbb{K}\backslash\{0\} \mid \mathbf{X} = \lambda \mathbf{Y}.$$

Dans ce qui suit, la relation d'équivalence $\sim$ est appelée ***égalité projective*** et sera généralisée aux matrices.

On définit l'application P qui à tout vecteur de $\mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$ lui associe sa classe d'équivalence modulo l'égalité projective $\sim$. L'ensemble $P(\mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\})$, c'est-à-dire l'ensemble quotient de $\mathbb{K}_{n+1}\backslash\{0\}$ suivant $\sim$, est appelé ***espace projectif*** déduit de $\mathbb{K}_{n+1}$ et est noté $P_n(\mathbb{K})$. Par définition,

$$dim(P_n(\mathbb{K})) = dim(\mathbb{K}^{n+1}) - 1 = n.$$

Si $\mathbb{K} = \mathbb{R}$, l'espace projectif est dit *réel* et si $\mathbb{K} = \mathbb{C}$, l'espace projectif est dit *complexe*.

Pour tout $\mathbf{X} \in \mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$, l'élément $P(\mathbf{X})$ est appelé ***point projectif*** de $P_n(\mathbb{K})$ et s'identifie à la droite vectorielle de $\mathbb{K}_{n+1}\backslash\{0\}$ de vecteur directeur $\mathbf{X}$; ainsi il est usuel de dire que $P_n(\mathbb{K})$ est l'ensemble des droites vectorielles de $\mathbb{K}_{n+1}$ auquel on aurait soustrait le vecteur $\mathbf{0}_{n+1}$. Tout sous-espace vectoriel $F \subset \mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$ de dimension $r+1$ engendre un ***sous-espace projectif*** $P(F) \subset P_n(\mathbb{K})$ de dimension r.

Dual d'un espace projectif. Le dual de l'espace projectif $P_n(\mathbb{K})$ est un espace projectif de même dimension, noté $P_n^*(\mathbb{K})$. La dualité vient de la relation bijective

existant entre tout espace vectoriel et son dual : à tout point projectif de cet nouvel espace est associé un hyperplan de $P_n(\mathbb{K})$, de même qu'à tout hyperplan de $P_n^*(\mathbb{K})$ est associé un point projectif de $P_n(\mathbb{K})$. Un hyperplan de $P_n(\mathbb{K})$ est donc un « point » de $P_n^*(\mathbb{K})$ et peut être ainsi représenté par un vecteur $\mathbf{h} \in \mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$. Il s'ensuit que les points de $P_n(\mathbb{K})$, de vecteur $\mathbf{X} \in \mathbb{K}_{n+1}\backslash\{\mathbf{0}_{n+1}\}$, appartenant à cet hyperplan vérifient :

$$\mathbf{h}^T\mathbf{X} = 0.$$

5.3 Quadriques

5.3.1 Quadriques projectives

Définition 1 *Toute forme quadratique $\mathcal{Q}$ non nulle sur $\mathbb{K}^{n+1}$ définit une **quadrique projective** $\tilde{\mathcal{Q}}$ de $P_n(\mathbb{K})$ qui est l'ensemble des points de $P_n(\mathbb{K})$, appelé **lieu de la quadrique**, dont les vecteurs $\mathbf{X} \in \mathbb{K}^{n+1}$ satisfont l'équation $\mathcal{Q}(\mathbf{X}) = 0$.*

La matrice non nulle, symétrique et d'ordre $n+1$

$$\mathtt{Q} = \begin{pmatrix} Q_{1,1} & \frac{1}{2}Q_{1,2} & \cdots & \frac{1}{2}Q_{1,n+1} \\ \frac{1}{2}Q_{1,2} & Q_{2,2} & \cdots & \frac{1}{2}Q_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}Q_{1,n+1} & \frac{1}{2}Q_{2,n+1} & \cdots & Q_{n+1,n+1} \end{pmatrix}$$

associée à la forme quadratique

$$\mathcal{Q}(\mathbf{X}) = \sum_{i=1}^{n+1}\sum_{j=i}^{n+1} Q_{i,j}x_ix_j = \mathbf{X}^T\mathtt{Q}\mathbf{X}$$

définit une quadrique projective $\tilde{\mathcal{Q}}$ dont le lieu a pour équation

$$\mathbf{X}^T\mathtt{Q}\mathbf{X} = 0.$$

*On appelle **matrice de la quadrique** $\tilde{\mathcal{Q}}$ toute matrice (non nulle, symétrique et d'ordre $n+1$) proportionnelle à $\mathtt{Q}$.*

5.3.1.1 Conjugaison et polarité relatives à une quadrique

Deux points $\tilde{X}$ et $\tilde{Y}$, de vecteurs $\mathbf{X}$ et $\mathbf{Y}$, sont ***conjugués relativement à une quadrique*** $\tilde{\mathcal{Q}}$ de matrice $\mathtt{Q}$ si et seulement si

$$\mathbf{X}^T \mathtt{Q} \mathbf{Y} = 0.$$

L'ensemble des points conjugués à $\tilde{X}$, relativement à $\tilde{\mathcal{Q}}$, est un hyperplan $\tilde{U}$ dont le vecteur $\mathbf{U}$ est donné par

$$\mathbf{U} \sim \mathtt{Q}\mathbf{X}. \tag{5.1}$$

Définition 2 *L'hyperplan $\tilde{U}$, de vecteur (5.1), formé par l'ensemble des points conjugués à un point $\tilde{X}$, relativement à $\tilde{\mathcal{Q}}$, est appelé* ***hyperplan polaire*** *de $\tilde{X}$, et le point $\tilde{X}$ est appelé* ***pôle*** *de $\tilde{U}$.*

5.3.1.2 Tangence à une quadrique

Soit $\tilde{\mathcal{Q}}$ une quadrique et soient $\tilde{X}$ et $\tilde{Y}$ deux points, non situés sur $\tilde{\mathcal{Q}}$. Soient $\tilde{A}$ et $\tilde{B}$ les deux points où la droite passant par $\tilde{X}$ et $\tilde{Y}$ intersecte $\tilde{\mathcal{Q}}$. Si $\tilde{A}$ et $\tilde{B}$ coïncident en un même point, alors la droite est dite ***droite tangente à la quadrique*** $\tilde{\mathcal{Q}}$ en ce point, et celui-ci est dit ***point de contact***.

On notera les points suivants.

- Le point de contact de la droite tangente à la quadrique $\tilde{\mathcal{Q}}$ passant par deux points $\tilde{X}$ et $\tilde{Y}$, non situés sur $\tilde{\mathcal{Q}}$, est conjugué à $\tilde{X}$ et $\tilde{Y}$, relativement à $\tilde{\mathcal{Q}}$.
- Pour $\tilde{X}$ (ou $\tilde{Y}$) fixé, l'ensemble des points de contact, relatif aux droites passant par $\tilde{X}$ (ou $\tilde{Y}$) qui sont tangentes à la quadrique $\tilde{\mathcal{Q}}$, est inclus dans le plan polaire de $\tilde{X}$ (ou $\tilde{Y}$).

Si $\tilde{A}$ appartient au lieu de la quadrique, alors l'ensemble des droites tangentes à $\tilde{\mathcal{Q}}$ au point $\tilde{A}$ forme un hyperplan. Ainsi, l'hyperplan formé par l'ensemble des droites tangentes à $\tilde{\mathcal{Q}}$ en un point de contact est appelé **hyper*plan tangent à la quadrique*** $\tilde{\mathcal{Q}}$ en ce point.

5.3.2 Quadrique duale

Définition 3 *Toute forme quadratique $\mathcal{Q}^*$ non nulle sur $(\mathbb{K}^{n+1})^*$ définit une quadrique projective $\tilde{\mathcal{Q}}^*$ de $P_n^*(\mathbb{K})$, dite* ***quadrique projective duale****, qui est l'ensemble des hyperplans de $P_n(\mathbb{K})$, appelé* ***enveloppe de la quadrique****, dont les vecteurs $\mathbf{U}$ satisfont l'équation $\mathcal{Q}^*(\mathbf{U}) = 0$.*

La matrice non nulle, symétrique et d'ordre $n+1$

$$\mathtt{Q}^* = \begin{pmatrix} Q^*_{1,1} & \frac{1}{2}Q^*_{1,2} & \cdots & \frac{1}{2}Q^*_{1,n+1} \\ \frac{1}{2}Q^*_{1,2} & Q^*_{2,2} & \cdots & \frac{1}{2}Q^*_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}Q^*_{1,n+1} & \frac{1}{2}Q^*_{2,n+1} & \cdots & Q^*_{n+1,n+1} \end{pmatrix},$$

associée à la forme quadratique sur $(\mathbb{K}^{n+1})^*$

$$\mathcal{Q}^*(\mathbf{U}) = \sum_{i=1}^{n+1}\sum_{j=i}^{n+1} Q^*_{i,j}u_iu_j = \mathbf{U}^T\mathtt{Q}^*\mathbf{U}.$$

définit une quadrique projective $\tilde{\mathcal{Q}}$ *dont l'enveloppe a pour équation*

$$\mathbf{U}^T\mathtt{Q}^*\mathbf{U} = 0.$$

On appelle ***matrice de la quadrique*** *de* $\tilde{\mathcal{Q}}^*$ *toute matrice (non nulle, symétrique et d'ordre* $n+1$*) proportionnelle à* $\mathtt{Q}^*$.

Si $\tilde{\mathcal{Q}}$ est une quadrique de $P_n(\mathbb{K})$, dont la matrice $\mathtt{Q}$ est de rang plein, alors l'ensemble des hyperplans tangents à $\tilde{\mathcal{Q}}$ définit l'enveloppe d'une quadrique projective $\tilde{\mathcal{Q}}^*$ de $P^*_n(\mathbb{K})$ dont la matrice $\mathtt{Q}^*$ vérifie

$$\mathtt{Q}^* \sim \mathtt{Q}^{-1}. \tag{5.2}$$

Ainsi, puisqu'on identifie les hyperplans de $P_n(\mathbb{K})$ aux points de $P^*_n(\mathbb{K})$, la quadrique $\mathcal{Q}^*$ de $P^*_n(\mathbb{K})$ identifiée à $\mathcal{Q}$ est la quadrique projective de matrice (5.2) et, pour cette raison, est parfois appelée ***quadrique duale de*** $\mathcal{Q}$.

5.3.3 Quadriques dégénérées des espaces projectifs de dimensions deux et trois

Nous nous plaçons maintenant dans le cadre des espaces projectifs de dimensions deux et trois même si les définitions 4 et 5 restent valides pour les espaces projectifs de toute dimension. Nous restreindrons le cadre de notre travail aux formes quadratiques à coefficients dans $\mathbb{K} = \mathbb{R}$ et ainsi *nous ne considérerons que les quadriques projectives de matrices réelles*. Dans le cas général, une quadrique a $\frac{n(n+3)}{2}$ degrés de liberté correspondant aux $\frac{(n+1)(n+2)}{2}$ éléments de sa matrice moins le facteur constant non nul :

– si $n = 2$, les quadriques (qui sont alors appelées ***coniques***) ont *cinq degrés de liberté* ;
– si $n = 3$, les quadriques ont *neuf degrés de liberté.*

Définition 4 *On appelle rang d'une quadrique projective le rang de sa matrice associée. Une quadrique de rang plein est dite quadrique propre.*

Définition 5 *Une quadrique projective dégénérée est une quadrique projective qui n'est pas de rang plein, c'est-à-dire. une quadrique projective dont la matrice a un rang strictement inférieur à $n+1$ dans un espace projectif de dimension n.*

5.3.3.1 Enveloppes des quadriques dégénérées lorsque $n \in \{2, 3\}$

Nous nous restreignons à l'interprétation géométrique des quadriques dégénérées de $P_3^*(\mathbb{K})$ dans la proposition 6 ; celle-ci ne se déduit pas trivialement par dualité de l'interprétation géométrique des quadriques dégénérées de $P_3(\mathbb{K})$.

Proposition 6

$n = 3$ *et* $\operatorname{rang} \tilde{\mathcal{Q}}^* = 3$: *l'enveloppe d'une quadrique projective $\tilde{\mathcal{Q}}^*$ de $P_3^*(\mathbb{K})$ de rang 3 est identifié à l'ensemble de faisceau de plans de $P_3(\mathbb{K})$ dont les axes sont des droites contenues dans un plan de $P_3(\mathbb{K})$ et tangentes à une quadrique non dégénérée $\tilde{\mathcal{A}}$ de $P_3(\mathbb{K})$. On dira que cet ensemble est l'enveloppe tridimensionnelle d'une conique projective. Si $\mathbf{p}$ est le vecteur de l'hyperplan et si $\mathtt{A}^*$ est la matrice de la quadrique duale $\tilde{\mathcal{A}}^*$ alors la matrice $\mathtt{Q}$ de $\tilde{\mathcal{Q}}^*$, de rang 3, admet la décomposition suivante :*

$$\mathtt{Q}^* \sim (\mathbf{p}^T \mathtt{A}^* \mathbf{p})\mathtt{A}^* - (\mathtt{A}^* \mathbf{p})(\mathtt{A}^* \mathbf{p})^T.$$

$n \in \{2, 3\}$ *et* $\operatorname{rang} \tilde{\mathcal{Q}}^* = 2$: *l'enveloppe d'une quadrique projective $\tilde{\mathcal{Q}}^*$ de $P_n^*(\mathbb{K})$ de rang 2 est formée par l'ensemble des droites de $P_n^*(\mathbb{K})$ passant par au moins un point d'une paire de points distincts de $P_n(\mathbb{K})$. Si $\mathbf{u}$ et $\mathbf{v}$ sont les vecteurs de ces points, alors la matrice $\mathtt{Q}^*$ de $\tilde{\mathcal{Q}}^*$, de rang 2, admet la décomposition suivante :*

$$\mathtt{Q}^* \sim \mathbf{u}\mathbf{v}^T + \mathbf{v}\mathbf{u}^T.$$

$n \in \{2, 3\}$ *et* $\operatorname{rang} \tilde{\mathcal{Q}}^* = 1$: *l'enveloppe d'une quadrique projective $\tilde{\mathcal{Q}}^*$ de $P_n^*(\mathbb{K})$ de rang 1 est formée par un point de $P_n^*(\mathbb{K})$. Si $\mathbf{u}$ est le vecteur de ce point alors*

la matrice $\mathtt{Q}^*$ *de* $\tilde{\mathcal{Q}}^*$*, de rang 1, admet la décomposition suivante :*

$$\mathtt{Q}^* \sim \mathbf{u}\mathbf{u}^T.$$

5.3.4 Transformation d'une quadrique projective

Soit $\mathtt{H}$ la matrice d'une homographie quelconque H de l'espace projectif. La transformation de toute quadrique projective de matrice $\mathtt{Q}$ par l'homographie H s'écrit

$$\mathtt{Q}' \sim \mathtt{H}^{-T}\mathtt{Q}\mathtt{H}^{-1}.$$

Dans l'espace projectif dual, la transformation par H de toute quadrique projective duale de matrice $\mathtt{Q}^*$ s'écrit

$$\mathtt{Q}^{*\prime} \sim \mathtt{H}\mathtt{Q}^*\mathtt{H}^T.$$

5.3.5 Signature d'une quadrique projective

Nous appelons ***invariant projectif*** toute quantité qui reste inchangée quelle que soit la représentation projective considérée de l'espace. L'unique invariant projectif des quadriques projectives à coefficients dans $\mathbb{C}$ est le rang de leur matrice. Puisque nous ne considérons que les quadriques à coefficients dans $\mathbb{R}$, il est possible de définir un autre invariant projectif appelé *signature de la quadrique*.

Définition 1 *La signature d'une quadrique projective de matrice* $\mathtt{Q}$ *(respectivement* $\mathtt{Q}^*$ *sous sa forme duale) est défini par* (ξ_1, ξ_2)*, où :*

$$\xi_1 = \max(\rho, \nu) \;\; et \;\; \xi_2 = \min(\rho, \nu),$$

avec ρ *et* ν *qui dénombrent respectivement les valeurs propres positives et les valeurs propres négatives de* $\mathtt{Q}$ *(respectivement* $\mathtt{Q}^*$*).*

Proposition 7 *La signature d'une quadrique projective à coefficients dans* $\mathbb{R}$ *est projectivement invariant.*

La démonstration de cette proposition est immédiate à établir à partir de la signature d'une forme quadratique et de la loi d'inertie de Sylvester [Golub 1983, p. 403]. On notera que

$$\xi_1 + \xi_2 = \operatorname{rank} \mathtt{Q},$$

ce qui confirme que le rang de $\mathtt{Q}^*$ est aussi invariant projectivement.

Définition 8 *On appelle quadrique virtuelle toute quadrique projective de signature* $(\xi_1, \xi_2) = (R, 0)$ *où* R *désigne le rang de la quadrique, c'est-à-dire dont la matrice réelle de rang* R *a* R *valeurs propres de même signe.*

Il est facile de montrer qu'une quadrique virtuelle ne contient pas de points réels (représentés par des vecteurs réels) mais uniquement des points complexes conjugués (représentés par des vecteurs complexes conjugués).

Plus généralement, la signature définie précédemment permet d'établir la classification suivante des quadriques projectives. Nous dirons que deux quadriques sont projectivement équivalentes s'il existe une homographie transformant l'une en l'autre et vice-versa.

Pour les quadriques projectives $\mathcal{Q}$ et $\mathcal{Q}^*$ de matrices réelles respectives $\mathtt{Q}$ et $\mathtt{Q}^* = \mathtt{Q}^{-1}$, on a :

$$(\xi_1, \xi_2) = \begin{cases} (4,0) : \mathcal{Q},\ \mathcal{Q}^* \text{ projectivement équivalentes à une sphère virtuelle} \\ (3,1) : \mathcal{Q},\ \mathcal{Q}^* \text{ projectivement équivalentes à une sphère réelle} \\ (2,2) : \mathcal{Q},\ \mathcal{Q}^* \text{ projectivement équivalentes à un hyperboloïde à une nappe} \end{cases} \tag{5.3}$$

On peut remarquer qu'il n'existe pas d'homographie qui transforme une sphère (virtuelle ou non) en un hyperboloïde à une nappe.

Pour les signatures d'une quadrique duale $\mathtt{Q}^*$ de rang 3 on a :

$$(\xi_1, \xi_2) = \begin{cases} (3,0) : \mathcal{Q},\ \mathcal{Q}^* \text{ projectivement équivalentes à un cercle virtuel} \\ (2,1) : \mathcal{Q},\ \mathcal{Q}^* \text{ projectivement équivalentes à un cercle réel} \end{cases} \tag{5.4}$$

Un cas remarquable de quadrique duale dégénérée de rang 3, bien connu des chercheurs en vision par ordinateur, est celui de la *quadrique absolue duale* [Hartley 2003, p. 84]) notée $\mathtt{Q}^*_\infty$ et introduit plus tard dans la section 5.5. Son enveloppe est le *cercle absolu* du plan à l'infini.

Pour les signatures d'une quadrique duale $\mathtt{Q}^*$ de rang 2 on a :

$$(\xi_1, \xi_2) = \begin{cases} (1,1) : \text{l'enveloppe de } \mathtt{Q}^* \text{ est une paire de points réels} \\ (2,0) : \text{l'enveloppe de } \mathtt{Q}^* \text{ est une paire de points complexes conjugués} \end{cases} \tag{5.5}$$

La Table 5.1, page 109 récapitule toutes les quadriques dégénérées de $P_3(\mathbb{R})$ avec leur nom et leur signature [Ladegaillerie 2003, p. 466]. La *dégénérescence* est

nom de la quadrique	signature
cône de base une conique propre réelle	$(2,1)$
cône imaginaire de sommet réel	$(3,0)$
cylindre à base elliptique réel	$(2,1)$
cylindre à base elliptique imaginaire	$(3,0)$
cylindre à base hyperbolique	$(2,1)$
paire de plans réels sécants	$(1,1)$
paire de plans imaginaires sécants	$(2,0)$
cylindre à base parabolique	$(2,0)$
paire de plans réels parallèles distincts	$(1,1)$
paire de plans imaginaires conjugués parallèles distincts	$(2,0)$
paire de plans confondus	$(1,0)$

TABLE 5.1 – Récapitulatif des quadriques dégénérées avec leur nom et leur signature.

invariante projectivement vu que le rang l'est.

5.4 Quadriques affines euclidiennes

5.4.1 Compléments de géométrie projective.

Nous donnons ici quelques notions complémentaires de géométrie projective.

5.4.1.1 Structure affine d'un espace projectif. Hyperplan de l'infini.

Nous rappelons que si F désigne un hyperplan vectoriel de $\mathbb{R}^{n+1}$, alors $P_n(\mathbb{R})\backslash P(F)$ possède une structure d'espace affine de dimension n ; $\mathcal{P}(F)$ est alors appelé ***hyperplan de l'infini*** de $P_n(\mathbb{R})$ et est noté H_∞. On dit que la donnée de l'hyperplan de l'infini munit l'espace projectif d'une structure affine car on montre que $P_n(\mathbb{R})\backslash P(F)$ a de facto une structure affine. Une ***représentation affine naturelle*** de l'espace projectif correspond au choix d'un système de coordonnées homogènes dans lequel l'hyperplan à l'infini a pour vecteur

$$\mathbf{H}_\infty = [0, ..., 0, 1]^T .$$

Dans une telle représentation, les points « affines" de l'espace projectif sont représentés par des vecteurs $\mathbf{X}$ tels que les composantes $X_1, X_2, ..., X_n$ sont des coordonnées cartésiennes et $X_{n+1} = 1$. Les « autres" points de l'espace projectif sont les points à l'infini avec $X_{n+1} = 0$.

5.4.1.2 Structure affine euclidienne d'un espace projectif. Cercle absolu.

La donnée du cercle absolu dans l'hyperplan à l'infini munit l'espace projectif affine d'une structure affine euclidienne. Le cercle absolu est un cercle virtuel et

son rayon est $\sqrt{-1}$; il ne contient aucun point réel et est uniquement composé de points conjugués complexes. C'est l'unique cercle de l'espace projectif qui est invariant par les similitudes de l'espace projectif, c'est-à-dire par des homographies dont les matrices sont de la forme

$$\mathtt{T} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}_n^T & 1 \end{bmatrix}$$

où s est un scalaire non nul, $\mathtt{R}$ une matrice orthogonale (représentant une rotation si $\det \mathtt{R} = 1$) et $\mathbf{t}$ un vecteur de translation.

Quadrique absolue duale. Le cercle absolu peut être défini dans l'espace projectif dual comme l'enveloppe d'une quadrique duale virtuelle, dégénérée de rang 3, généralement notée $\mathcal{Q}_\infty^*$. Cette quadrique est désignée par le terme quadrique absolue duale et la matrice de $\mathcal{Q}_\infty^*$ a la forme canonique suivante, pour toute représentation affine euclidienne de l'espace projectif

$$\mathtt{Q}_\infty^* = \begin{bmatrix} \mathtt{I}_n & \mathbf{0} \\ \mathbf{0}_n^T & 1 \end{bmatrix}.$$

L'invariance du cercle absolu par toute similitude de matrice $\mathtt{T}$ se traduit ici simplement par l'égalité :

$$\mathtt{T}\mathtt{Q}_\infty^*\mathtt{T}^T \sim \mathtt{Q}_\infty^*.$$

Le fait que $\mathbf{H}_\infty \in \ker \mathtt{Q}_\infty^*$ indique que $\mathtt{Q}_\infty^*$ « encode » simultanément les données de l'hyperplan de l'infini et du cercle absolu.

5.4.2 Quadriques affines

Dans la section précédente, nous avons introduit une théorie générale des quadriques projectives. Cette théorie peut se spécialiser dès lors que l'espace projectif possède une structure affine, c'est-à-dire lorsque l'hyperplan à l'infini peut-être identifié. Nous rappelons que si F désigne un hyperplan vectoriel de $\mathbb{R}^{n+1}$, alors $P_n(\mathbb{R}) \setminus \mathcal{P}(F)$ possède une structure d'espace affine de dimension n ; $\mathcal{P}(F)$ est alors appelé hyperplan de l'infini et noté $\mathcal{H}_\infty$. Nous nous plaçons sous cette hypothèse dans ce qui suit, toujours dans le cadre des espaces projectifs de dimensions deux et trois.

Une ***quadrique affine*** de $P_n(\mathbb{K})$ est une quadrique propre $\tilde{\mathcal{Q}}_\mathcal{A}$ de $P_n(\mathbb{K})$ dont le lieu restreint à l'ensemble des points affines de $P_n(\mathbb{K})$, c'est-à-dire $\notin \mathcal{H}_\infty$, n'est

pas vide. Par dualité, en identifiant les hyperplans de $P_n(\mathbb{K})$ aux points de $P_n^*(\mathbb{K})$, la quadrique duale $\mathcal{Q}_\mathcal{A}^*$ de $P_n^*(\mathbb{K})$ identifiée à $\mathcal{Q}_\mathcal{A}$ sera appelée ***quadrique affine duale***. Les quadriques affines héritent de nouvelles caractéristiques et propriétés, dites affines, et peuvent être distinguées à partir d'une certaine classification, étudiant leur intersection avec le plan de l'infini $\mathcal{H}_\infty$. Il est à noter que les quadriques affines héritent des propriétés des quadriques projectives énoncées précédemment.

Définition 9 *Soit $\tilde{\mathcal{Q}}_\mathcal{A}$ une quadrique affine propre de $P_n(\mathbb{K})$. Le **centre** de $\tilde{\mathcal{Q}}_\mathcal{A}$ est le pôle de l'hyperplan de l'infini $\mathcal{H}_\infty$ par rapport à $\tilde{\mathcal{Q}}_\mathcal{A}$, c'est-à-dire est le point de $P_n(\mathbb{K})$ dont le vecteur vérifie :*

$$\mathbf{C} \sim \mathtt{Q}^* \mathbf{h}_\infty \tag{5.6}$$

*où $\mathtt{Q}^*$ est la matrice de la quadrique duale de $\tilde{\mathcal{Q}}_\mathcal{A}$ et $\mathbf{h}_\infty$ est le vecteur de $\mathcal{H}_\infty$. La quadrique est dite **centrale** si son centre est un point affine, c'est-à-dire n'appartenant pas à $\mathcal{H}_\infty$.*

Il est possible d'établir la classification suivante des quadriques affines en étudiant leurs intersections avec l'hyperplan de l'infini (voir [Semple 1952, p. 282-283]). En supposant que l'hyperplan à l'infini $\mathcal{H}_\infty$ est représenté sous forme « naturelle » par le vecteur

$$\mathbf{h}_\infty = [0, \dots, 0, 1]^T \in \mathbb{K}^{n+1}$$

et que tout point affine est représenté par un vecteur $\mathbf{X} = [x, y, \dots, 1]^T \in \mathbb{K}^{n+1}$, où $(x, y, \dots)$ désigne les coordonnées cartésiennes du point, relativement à un repère choisi de manière adéquate, la Table 5.2 établit une classification des quadriques affines à partir de leur équation sous forme canonique [Ladegaillerie 2003, p. 466].

Il est à noter que, dans cette représentation, le centre de la quadrique affine propre $\tilde{\mathcal{Q}}_\mathcal{A}$ de matrice $\mathtt{Q}$ a pour vecteur

$$\mathbf{C} = \mathtt{Q}^* \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

c'est-à-dire correspond à la dernière ligne de l'inverse de $\mathtt{Q}$.

nom de la quadrique	équation canonique
ellipsoïde réel	$a^2x^2 + b^2y^2 + c^2z^2 = 1$
ellipsoïde imaginaire	$a^2x^2 + b^2y^2 + c^2z^2 = -1$
hyperboloïde à une nappe	$a^2x^2 + b^2y^2 - c^2z^2 = 1$
hyperboloïde à deux nappes	$a^2x^2 + b^2y^2 - c^2z^2 = -1$
cône réel	$a^2x^2 + b^2y^2 - c^2z^2 = 0$
cône imaginaire	$a^2x^2 + b^2y^2 + c^2z^2 = 0$
paraboloïde elliptique	$a^2x^2 + b^2y^2 + cz = 0$
paraboloïde hyperbolique	$a^2x^2 - b^2y^2 + cz = 0$
cylindre elliptique réel	$a^2x^2 + b^2y^2 = 1$
cylindre elliptique imaginaire	$a^2x^2 + b^2y^2 = -1$
cylindre hyperbolique	$a^2x^2 - b^2y^2 = 1$
plans sécants réels	$a^2x^2 - b^2y^2 = 0$
plans sécants imaginaires	$a^2x^2 + b^2y^2 = 0$
cylindre parabolique	$a^2x^2 + by^2 = 0$
plans parallèles réels	$a^2x^2 = 1$
plans parallèles imaginaires	$a^2x^2 = -1$
plans confondus	$a^2x^2 = 0$

TABLE 5.2 – Équations canoniques des quadriques affines.

5.4.3 Quadriques euclidiennes

La notion de quadrique affine peut être encore spécialisée en identifiant dans l'hyperplan de l'infini le cercle absolu. La donnée de ce cercle (ajoutée à celle de l'hyperplan à l'infini) munit l'espace projectif d'une structure affine euclidienne. Dans l'espace projectif de dimension trois, le cercle absolu est un cercle virtuel, parfois appelé conique absolue. Dans l'espace projectif de dimension deux, ce cercle est dégénéré et se réduit à une paire de points complexes conjugués, formée par les deux points circulaires du plan projectif.

La notion de foyers d'une quadrique joue un rôle primordial dans la classification des quadriques euclidiennes. Dans le plan projectif euclidien, le foyer d'une conique euclidienne peut être défini (tel que nous les connaissons) comme le point d'intersection de deux droites tangentes à la conique, chacune passant par un des deux points circulaires du plan. Une conique a quatre foyers formés par deux paires de points, l'une réelle et l'autre complexe conjuguée. La paire de foyers réels engendre le grand axe de la conique alors que la paire de foyers complexes conjugués engendre le petit axe de la conique. Dans l'espace projectif de dimension trois, la notion de foyer se généralise par la définition suivante [Salmon 1882, p. 126].

Définition 10 *Un point est un* ***foyer d'une quadrique*** *s'il est le point d'inter-*

section de deux droites, tangentes simultanément à la quadrique et au cercle absolu, telles que, pour chacune d'elles, le plan tangent à la quadrique la contenant soit aussi tangent au cercle absolu.

Les quadriques euclidiennes n'ont pas un nombre fini de foyers mais une infinité appartenant en général à trois coniques dites focales, situées dans des plans orthogonaux.

Quadriques de révolution. Les quadriques de révolution (que l'on désignera par la suite par QdR) sont une spécialisation des quadriques euclidiennes. Rappelons qu'une QdR est une quadrique obtenue par la révolution d'une conique autour d'un de ses axes de symétrie. Un point important dans notre travail est qu'une QdR comporte quatre foyers remarquables appelé ***foyers principaux***, qui sont les foyers de la conique de révolution.

Dans le cas où l'axe de révolution est celui portant les deux foyers principaux réels de la conique, la QdR engendrée pourra être un *ellipsoïde de révolution réel prolate*, un *ellipsoïde de révolution imaginaire oblate*, un *hyperboloïde de révolution à deux nappes* ou un *paraboloïde de révolution* (dans ce dernier cas, un des deux foyers principaux réel se trouve à l'infini). Une quadrique appartenant à ce premier groupe de QdR est nommée dans la suite QdR_1. En revanche, dans le cas où l'axe est celui portant les foyers principaux complexes conjugués (condition nécessaire et suffisante pour que l'axe soit une droite réelle), la QdR engendrée pourra être un *ellipsoïde de révolution réel oblate*, un *ellipsoïde de révolution imaginaire prolate* ou un *hyperboloïde de révolution à une nappe*. Une quadrique appartenant à ce second groupe de QdR est nommée QdR_2. On peut voir les QdR_1 sur la Figure 5.1, page 114 et les QdR_2 sur la Figure 5.2, page 114 avec les coniques qui les ont engendrées (en rouge) et leur axe de symétrie autour desquels on a effectué les révolutions.

La Table 5.3, page 114 récapitule l'ensemble des QdR non dégénérées existantes avec leur nom, leur signature et leur type tel qu'on l'a préalablement défini (QdR_1 ou QdR_2).

5.5 Vision par ordinateur

Nous allons rappeler brièvement comment se forme une image grâce à la géométrie projective détaillée ci-dessus et comment il est possible de retrouver des informations 3D sur une scène à partir de son image.

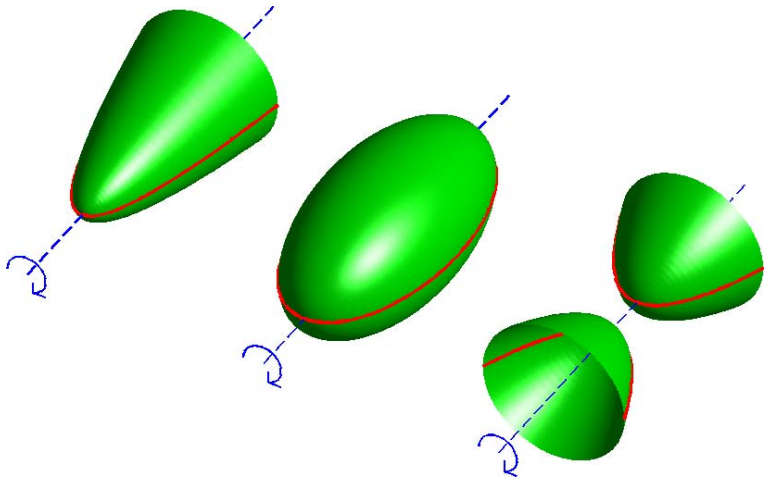

FIGURE 5.1 – Exemples de quadriques de révolution appartenant à la catégorie QdR_1. De gauche à droite, on peut voir un paraboloïde de révolution, un ellipsoïde de révolution réel prolate et un hyperboloïde de révolution à deux nappes.

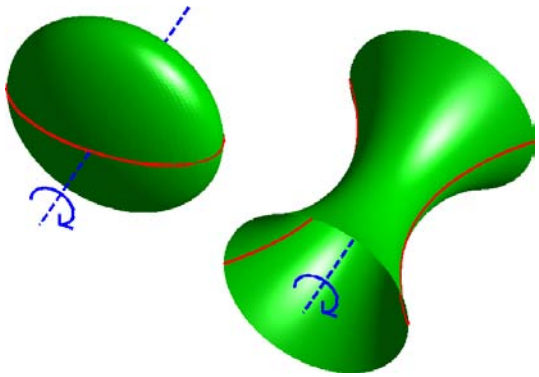

FIGURE 5.2 – Exemples de quadriques de révolution appartenant à la catégorie QdR_2. De gauche à droite, on peut voir un ellipsoïde de révolution réel oblate et un hyperboloïde de révolution à une nappe.

nom de la quadrique	**signature**	**révolution d'une conique autour**	**type**
ellipsoïde de révolution réel prolate ellipsoïde de révolution réel oblate	$(3,1)$	de son grand axe de son petit axe	QdR_1 QdR_2
ellipsoïde de révolution imaginaire prolate ellipsoïde de révolution imaginaire oblate	$(4,0)$	de son grand axe de son petit axe	QdR_2 QdR_1
hyperboloïde de révolution à une nappe	$(2,2)$		QdR_2
hyperboloïde de révolution à deux nappes	$(3,1)$		QdR_1
paraboloïde de révolution	$(3,1)$		QdR_1

TABLE 5.3 – Récapitulatif des quadriques de révolution avec leur nom, leur signature, la représentation de la manière dont elles sont engendrées et leur type (QdR_1 ou QdR_2).

5.5.1 Formation d'une image

Le modèle de prise de vue considéré ici est le *modèle sténopé* ou *trou d'aiguille*, c'est-à-dire tel que tous les éléments de la scène se projettent sur le plan du capteur, appelé *plan image* et noté $\mathcal{I}_f$, via des rayons qui passent tous par le centre optique de la caméra, noté $\mathbf{O}$. Le plan passant par $\mathbf{O}$ et perpendiculaire à l'axe optique de la caméra est appelé le *plan principal* $\mathcal{I}_0$. Ce dernier est parallèle au plan de l'image. La distance entre ces deux plans correspond à la *distance focale* utilisée lors de la prise de vue et notée f. L'intersection entre l'axe optique est $\mathcal{I}_f$ s'appelle le point principal et est noté $\mathbf{p_p} = (x_0, y_0, 1)^T$. On distingue maintenant trois repères différents :

- le *repère scène* $\mathcal{R}_S$ qui est un repère 3D quelconque de la scène.
- le *repère observateur* $\mathcal{R}_O$ qui est un repère 3D ayant pour centre $\mathbf{O}$ le centre optique de la caméra et dont l'axe des z correspond avec l'axe optique de la caméra.
- le *repère image* $\mathcal{R}_I$ qui est le repère 2D de l'image.

On appelle *projection centrale* de centre $\mathbf{O}$ sur le plan image $\mathcal{I}_f$ l'application qui à un point de vecteur $\mathbf{X} = (x, y, z, 1)^T$ de la scène associe un pixel de vecteur $\mathbf{p} = (u, v, 1)^T$ correspondant à l'intersection entre la droite $(\mathbf{OX})$ et le plan image $\mathcal{I}_f$. L'*équation de projection centrale*, transformant les coordonnées cartésiennes augmentées d'un point 3D, exprimées dans le repère repère $\mathcal{R}_S$, en coordonnées pixelliques augmentées de son image, exprimées dans le repère $\mathcal{R}_I$ s'écrit projectivement, via le repère $\mathcal{R}_O$

$$\mathbf{p} \sim \mathtt{P}\mathbf{X} = \mathtt{K}[\mathtt{R}|\mathbf{t}]\mathbf{X}$$

où $\mathtt{P}$ est la *matrice de projection*, $\mathtt{R}$ et $\mathbf{t}$ correspondent à la matrice de rotation et le vecteur de translation faisant passer du repère $\mathcal{R}_S$ au repère $\mathcal{R}_O$ (appelés *paramètres extrinsèques* de la caméra) et $\mathtt{K}$ est la *matrice de calibrage* contenant les *paramètres intrinsèques* de la caméra et permet de faire passer du repère $\mathcal{R}_O$ au repère $\mathcal{R}_I$ en coordonnées pixelliques. Cette matrice s'écrit :

$$\mathtt{K} = \begin{pmatrix} f & s & x_0 \\ 0 & \alpha f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

où α et s sont des paramètres permettant de corriger le fait que la projection se fait parfois de telle sorte que les pixels ne sont pas exactement des carrés.

L'ensemble de la formation d'une image présentée ici est résumée dans la Figure 5.3, page 116.

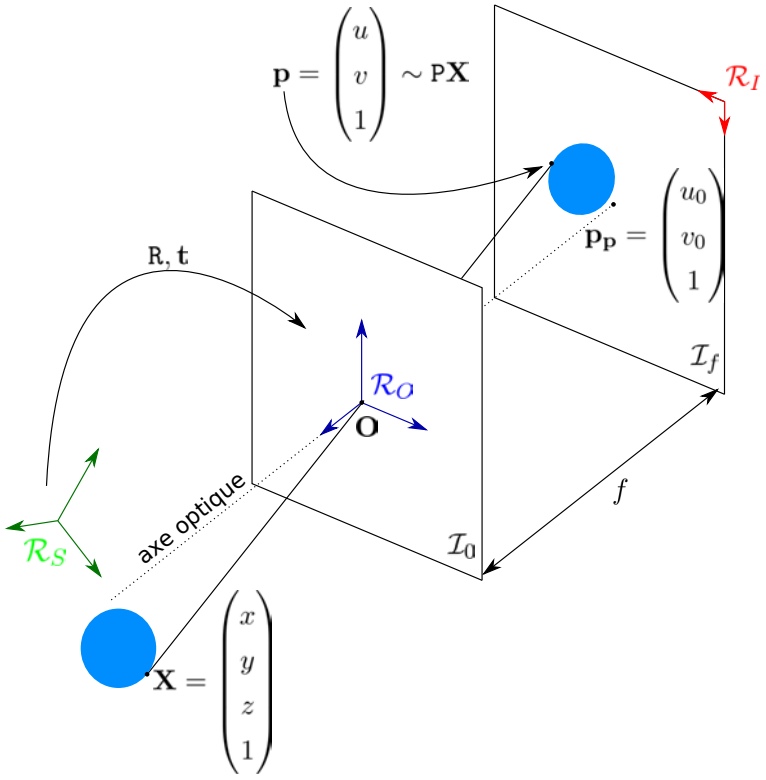

FIGURE 5.3 – Formation d'une image (voir texte).

5.5.2 Calibrage d'une caméra

L'étape permettant de retrouver les paramètres intrinsèques et les paramètres extrinsèques de la caméra s'appelle le *calibrage*. Le problème du calibrage intrinsèque d'une caméra est équivalent à celui de l'estimation de l'image ω_∞ du cercle absolu Ω_∞, situé sur le plan de l'infini, dont la matrice duale $\boldsymbol{\omega}^*$ admet la factorisation de Cholesky suivante

$$\boldsymbol{\omega}^* = \mathtt{K}\mathtt{K}^\top$$

où $\mathtt{K}$ désigne la matrice (triangulaire supérieure) de calibrage.

Le dual du cercle absolu est une quadrique duale de rang 3, notée Q_∞. Dans le cadre projectif, cette quadrique peut être vue comme l'enveloppe de plans tangents au cercle Ω_∞. La Figure 5.4, page 117 représente Ω_∞ et Q_∞.

De nombreuses techniques permettent de calibrer en utilisant Ω_∞ et Q_∞ que ce

soit à partir d'éléments de la scène comme des images de sphère [Zhang 2007], des surfaces de révolution [Wong 2003] ou des plans [Sturm 1999, Gurdjos 2005] pour n'en citer que quelques uns. L'utilisation de mires, qui peuvent être des éléments 1D, 2D ou 3D et dont on connaît parfaitement la géométrie ainsi que la manière dont elles se projettent dans l'image, permet également de retrouver les paramètres intrinsèques et les paramètres extrinsèques. Les positions de chaque caméra par rapport à ces mires peuvent en effet être calculées en fonction des projections de ces dernières dans les images. Ce manuscrit ne traite pas du calibrage mais dans la suite, nous supposons que nous travaillons dans des *vues calibrées*, c'est-à-dire que nous connaissons la matrice de calibrage $\mathtt{K}$.

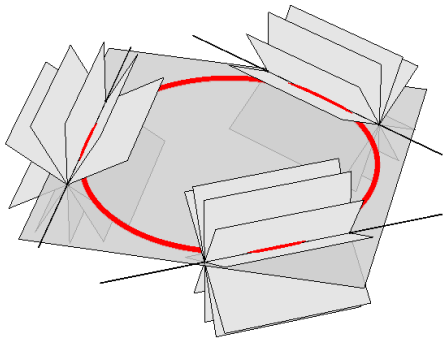

FIGURE 5.4 – La conique absolue est représentée en rouge (Q_∞), le plan à l'infini est représenté par le plan gris contenant la conique absolue et la quadrique absolue (Ω_∞) est représentée par l'ensemble des plans gris.

5.5.3 Triangulation pour la reconstruction 3D à partir de plusieurs images

Le paradigme de base pour reconstruire des éléments d'une scène en 3D à partir de plusieurs images peut se décomposer en 3 étapes. La première étape consiste à calibrer les caméras utilisées en retrouvant les paramètres intrinsèques et les paramètres extrinsèques. Puis, il s'agit d'apparier les images des éléments que l'on souhaite reconstruire dans les différentes images. La dernière étape est l'étape de triangulation qui permet de retrouver la position 3D de l'élément à reconstruire en *rétroprojetant* ses images, appariées auparavant. La rétroprojection est l'étape inverse de la projection centrale. Par exemple, pour un point $\mathbf{p_1}$ dans une image $\mathcal{I}_1$, sa rétroprojection est la droite passant par $\mathbf{p_1}$ et le centre optique de la caméra utilisée pour acquérir $\mathcal{I}_1$, privée de ce même centre optique. Typiquement,

avec n images $\mathbf{p_{i,i=1..n}}$ d'un point 3D $\mathbf{X}$ dans n images différentes $\mathcal{I}_{i,i=1..n}$, on commence par retrouver les n matrices de projection $\mathtt{P}_{i,i=1..n}$. Puis, on résout le système d'équations :

$$\begin{cases} \mathtt{P_1}\mathbf{X} \sim \mathbf{p_1} \\ \ldots \\ \mathtt{P_n}\mathbf{X} \sim \mathbf{p_n} \end{cases} \tag{5.7}$$

pour retrouver le point 3D. Cette résolution peut s'effectuer par une simple SVD (*Singular Value Decomposition*) en réécrivant le problème sous la forme $\mathtt{A}\mathbf{X} = \mathbf{0}$ ([Hartley 2003, p. 312]).

5.5.4 Projection d'une quadrique projective

Soit $\mathtt{P}$ une matrice de projection perspective, l'image par $\mathtt{P}$ d'une quadrique de matrice associée $\mathtt{Q}$ est une quadrique de l'espace projectif $P_2(\mathbb{R})$ (aussi appelée conique comme vu précédemment) de matrice associée $\mathtt{C}$. Cette conique définit le *contour occultant* de la quadrique dans la vue considérée. La rétroprojection de $\mathtt{C}$ est un cône de matrice associée $\mathtt{Q_c} \sim \mathtt{P}^T\mathtt{C}\mathtt{P}$ tangent à la quadrique $\mathtt{Q}$ et de centre, le centre optique de la caméra. Sous la forme duale, si $\mathtt{C}$ est de rang plein, on a $\mathtt{C}^* \sim \mathtt{C}^{-1}$ et $\mathtt{C}^* \sim \mathtt{P}\mathtt{Q}^*\mathtt{P}^T$.

Chapitre 6

Reconstruction de quadriques de révolution

Sommaire

6.1 Introduction

Dans ce chapitre, nous allons montrer qu'il est possible de retrouver l'image de l'axe de révolution d'une QdR à partir de son contour occultant dans une vue calibrée. Nous nous servons de ce résultat pour reconstruire une QdR à partir d'au moins deux contours occultants dans deux vues calibrées et pour retrouver la position et l'orientation d'une QdR de paramètres connus à partir du contour occultant dans une seule vue calibrée. Nous terminons par quelques évaluations et quelques possibles applications. Avant de continuer, nous effectuons un rappel rapide de la notion de de quadrique de révolution.

6.2 Travaux précédents

En utilisant la géométrie projective, Cross *et al.* [Cross 1998] ont décrit un schéma de triangulation linéaire pour reconstruire une quadrique générale à partir de trois contours occultants. En effet, ils ont démontré qu'un contour occultant dans une vue apportait cinq degrés de liberté. Dans une deuxième vue, seuls trois degrés de liberté sont alors indépendants de la première vue. Avec seulement deux vues, ils ne retrouvent par conséquent qu'une famille linéaire de solutions car il manque un des neuf degrés de liberté que comporte une quadrique.

Nous nous sommes intéressés à la reconstruction de quadriques à partir de seulement deux vues. Pour cela, les contours occultants ne suffisent pas pour avoir suffisamment de degrés de liberté. Cependant, l'intérêt de n'utiliser que les contours occultants est d'éviter l'appariement de pixels qui peut s'avérer compliqué surtout en cas de surface lisse. Le seul moyen est de réduire le nombre de degrés de liberté. Pour cela, nous avons travaillé sur les quadriques de révolution qui ne comportent que sept degrés de liberté. L'inconvénient est alors de ne pas avoir une méthode permettant de reconstruire l'ensemble des quadriques. En revanche, nous avons remarqué que tous les travaux portant que la reconstruction de quadriques ne montrent que des reconstructions de quadriques de révolution dans leurs exemples. En effet, elles sont beaucoup plus courantes dans les objets qui nous entourent.

Parmi ces travaux, certains utilisent plusieurs vues comme Ma *et al.* [Ma 1996] qui reconstruisent des ellipsoïdes à partir de trois contours occultants donnés dans trois vues perspectives. Un des intérêts de cette méthode est le fait que les calculs sont tous linéaires et que l'appariement de points n'est pas nécessaire. On a vu que Cross *et al.* [Cross 1998] ont besoin de trois contours occultants pour re-

construire tout type de quadriques mais ils montrent également que deux contours occultants peuvent suffire si on leur associe un point apparié correspondant à la projection d'un point 3D de la quadrique. Shashua *et al.* [Shashua 1997] décrivent une méthode n'ayant besoin que d'un contour occultant avec cependant quatre points appariés alors que Rahmann [Rahmann 2003] se servent d'informations provenant de la lumière et de sa réflectance pour reconstruire une quadrique à partir de deux vues polarisées. Enfin, Gong *et al.* [Gong 2004] détaillent une méthode de programmation quadratique séquentiel pour reconstruire n'importe quelle surface quadratique à partir de plusieurs images. Dans tous ces travaux, plus de deux vues sont nécessaire ou alors, on a besoin d'informations additionnelles comme la réflectance de la lumière ou l'appariement de points qui peut devenir compliqué dans le cas de surfaces lisses.

D'autres articles s'intéressent à la reconstruction de quadriques plus spécifiques, ce qui évite l'utilisation d'informations complexes à obtenir. Ainsi, Wu *et al.* [Wu 2006] reconstruisent des cônes circulaires tronqués à partir d'au moins deux vues non calibrées. Wijewickrema *et al.* [Wijewickrema 2006] proposent un algorithme en deux étapes pour reconstruire des sphères en utilisant que deux contours occultants et Quan [Quan 1996] et Kumar *et al.* [Kumar 2009] reconstruisent des coniques (quadriques dégénérées de rang 3) à partir d'au moins deux vues.

Ferri *et al.* [Ferri 1993] ont montré qu'il était possible, si on connaissait les paramètres d'une quadriques, de retrouver sa position dans l'espace à partir d'une seule vue. Cependant, la solution proposée ne fonctionne bien que pour quelques cas de quadriques comme les cônes et les sphères mais, pour le cas général, il existe des problèmes de calculs qui deviennent vitre trop complexes à résoudre.

La Table 6.1, page 122 résume toutes ces méthodes ainsi que le type de quadriques reconstruites et les besoins pour y parvenir. Les deux dernières lignes représentent nos méthodes, développées dans le prochain chapitre. On voit bien qu'aucune des méthodes de l'état de l'art ne propose des méthodes pour reconstruire les mêmes types de quadriques (dans le cas présent, les quadriques de révolution) avec les seules informations que l'on souhaite utiliser (à savoir les contours occultants dans deux vues calibrées).

article	type de quadrique reconstruite	informations utilisées
[Cross 1998]	toutes quadriques projectives	3 contours occultants ou 2 contours occultants + 1 point apparié
[Ma 1996]	ellipsoïdes	trois contours occultants
[Shashua 1997]	toutes quadriques	1 contour occultant + 4 points appariés
[Rahmann 2003]	toutes quadriques	2 vues polarisées
[Gong 2004]	toute surface quadratique	plus de 2 vues
[Wu 2006]	cônes circulaires tronqués	2 contours occultants
[Wijewickrema 2006]	sphères	2 contours occultants
[Quan 1996]	coniques (quadriques dégénérées de rang 3)	plus de 2 vues
[Kumar 2009]	coniques (quadriques dégénérées de rang 3)	plus de 2 vues
[Ferri 1993]	quadriques de paramètres connus mais applicables que pour certaines quadriques dégénérées	1 contour occultant
Nous	tout type de quadriques de révolution	2 contours occultants
Nous	tout type de quadriques de révolution de paramètres connus	1 contour occultant

TABLE 6.1 – Récapitulatif des méthodes de l'état de l'art sur la reconstruction de quadriques avec les types de quadriques reconstruites pour chaque méthode ainsi que les besoins informations nécessaires pour y parvenir. En bas du tableau, on voit les types de quadriques que nous souhaitons reconstruire dans cette partie avec les informations que l'on souhaite utiliser.

6.3 Paramétrage dual d'une quadrique de révolution

Dans le cadre projectif, une quadrique de révolution est une quadrique dont la matrice s'écrit comme la combinaison linéaire des matrices de la quadrique duale à ses foyers principaux et de la quadrique absolue duale. Via cette décomposition, il est facile de montrer que les sections planes de la quadrique contenant son axe de révolution correspondent toutes à une même conique. C'est cette décomposition que nous allons utiliser en étudiant la famille de quadriques engendrée par l'ensemble des combinaisons linéaires susmentionnées.

6.3.1 Faisceau tangentiel des quadriques de révolution homofocales

Nous désignerons par le terme *faisceau de quadriques* toute famille de quadriques représentées par combinaisons linéaires des matrices de deux quadriques, dites *quadriques de base*. Cette définition s'applique aussi dans l'espace projectif dual ; nous sommes dans ce cas et utiliserons aussi bien le terme *faisceau de quadriques duales* que le terme consacré *faisceau tangentiel de quadriques* [Berger 1990, p. 232].

Deux quadriques sont *homofocales* si elles ont les mêmes foyers principaux. Ainsi, la famille des QdR qui sont homofocales relativement à deux foyers principaux F et

G fixés peut-être définie, sous forme duale, par un faisceau tangentiel de quadriques ayant comme quadriques de base la quadrique absolue duale et la quadrique duale à la paire de points (F, G). Ce résultat projectif s'énonce analytiquement ainsi :

Théorème 11 *Soient* $\mathbf{F}$ *et* $\mathbf{G}$ *les vecteurs de deux points 3D* F *et* G*, pouvant être réels ou complexes conjugués. Le faisceau tangentiel de QdR homofocales ayant* F *et* G *comme foyers principaux peut être représenté, sous forme duale, par la famille de matrices symétriques d'ordre 4 :*

$$\mathtt{Q}^*(u) = u\mathtt{X}^* + (1-u)\mathtt{Q}^*_\infty, \tag{6.1}$$

où $u \in [0,1]$*,* $\mathtt{Q}^*_\infty$ *est la matrice de la quadrique absolue duale et*

$$\mathtt{X}^* = \mathbf{F}\mathbf{G}^\top + \mathbf{G}\mathbf{F}^\top, \tag{6.2}$$

est la matrice de rang 2 *représentant la quadrique duale à la paire* (F, G)*.*

Nous omettons la démonstration de ce théorème qui est un résultat assez connu de la géométrie projective.

Nous effectuerons souvent le changement de variable $x = \frac{u-1}{u}$ pour réécrire (6.1) sous la forme

$$\mathtt{Q}^*(x) = \mathtt{X}^* - x\mathtt{Q}^*_\infty \tag{6.3}$$

avec la convention $x = \infty$ lorsque $u = 0$. La nouvelle variable x est alors appelée *paramètre projectif* de la quadrique. Notre idée-clé est de décomposer la matrice $\mathtt{Q}$ associée à la QdR que l'on souhaite reconstruire sous la forme duale suivante :

$$\mathtt{Q}^* = \mathtt{X}^* - x_0\mathtt{Q}^*_\infty, \tag{6.4}$$

où $x_0 \in \mathbb{R}^*$ désigne son paramètre projectif.

Le faisceau tangentiel défini par (6.3) comporte *quatre* membres dégénérés, dont les matrices $\mathtt{Q}^*(\lambda_i)$, $i \in \{1..4\}$, sont singulières et dont les paramètres projectifs λ_i sont les racines de $P(\lambda) = \det(\mathtt{X}^* - \lambda\mathtt{Q}^*_\infty)$, qui est un polynôme de degré 4 en λ. On peut montrer que $P(\lambda)$ admet une racine double, à savoir la valeur 0 telle que $\mathtt{Q}^*(0) = \mathtt{X}^*$, ainsi que deux racines simples : la valeur ∞ telle que $\mathtt{Q}^*(\infty) = \mathtt{Q}^*_\infty$, et une dernière valeur que nous noterons $\lambda_\circ$. Ce paramètre $\lambda_\circ$ est celui d'une quadrique duale dégénérée de rang 3 associée à un cercle, réel pour les QdR1 et virtuel pour les QdR2, qui est perpendiculaire à l'axe de révolution et dont le centre se situe sur

cet axe. Ce résultat est mis en évidence par exemple dans [Gurdjos et al., ICCV09].

6.3.2 Minimalité du paramétrage

Un résultat bien connu [Hartley 2003, p. 84] est que, dans n'importe quelle représentation euclidienne de l'espace projectif, la matrice de la quadrique absolue duale $\mathtt{Q}_\infty^*$ a la forme canonique suivante :

$$\mathtt{Q}_\infty^* = \begin{bmatrix} \mathtt{I}_3 & \mathbf{0} \\ \mathbf{0}_3^T & 1 \end{bmatrix}$$

La conséquence immédiate est que la matrice d'une QdR sous la forme matricielle (6.4) est paramétrée par sept inconnues puisque $\mathtt{X}^*$ est une matrice d'ordre 4 et de rang 2 représentant une paire de points 3D ayant six degrés de liberté. Cela est cohérent avec les paramétrages existants n'utilisant pas la forme duale d'une QdR, comme par exemple avec ceux donnés dans [Goldman 1983].

6.3.3 Paramétrage dual de l'image d'une quadrique de révolution

Les propriétés projectives garantissent que l'image par projection centrale d'un faisceau tangentiel de quadriques est un faisceau tangentiel de coniques dont les membres sont les images des quadriques du faisceau. Ainsi, on peut déduire du théorème 11 le corollaire qui suit.

Corollaire 12 *Soient* $\mathbf{f}$ *et* $\mathbf{g}$ *deux vecteurs représentant les images* f *et* g *des foyers principaux* F *et* G*. L'image d'un faisceau tangentiel de QdR homofocales ayant* F *et* G *comme foyers principaux, représenté par la famille de matrices (6.1), est le faisceau tangentiel de coniques représenté par la famille de matrices symétriques d'ordre 3 suivantes :*

$$\mathtt{C}^*(y) = \mathtt{Y}^* - y\boldsymbol{\omega}^*, \tag{6.5}$$

où y *est un scalaire* $\in \mathbb{R} \cup \{\infty\}$*, la matrice*

$$\mathtt{Y}^* = \mathbf{f}\mathbf{g}^\top + \mathbf{g}\mathbf{f}^\top, \tag{6.6}$$

représente la conique duale aux points f *et* g*, et la matrice*

$$\boldsymbol{\omega}^* = \mathtt{K}\mathtt{K}^\top$$

représente l'image duale de la conique absolue et se décompose par factorisation de Cholesky à partir de la matrice (triangulaire supérieure) de calibrage $\mathtt{K}$.

Démonstration. Soit $\mathtt{Q}^*(x_0)$ la matrice d'une QdR ayant F et G comme foyers principaux, soit $\mathtt{P}$ une matrice de projection associée à une caméra et soit $\mathtt{C}^*$ la matrice de l'image duale de la QdR par $\mathtt{P}$. Étant donnée la décomposition (6.4), l'équation de la projection de la QdR par $\mathtt{P}$ s'écrit

$$\begin{aligned}
\mathtt{C}^* &\sim \mathtt{P}\,(\mathtt{X}^* - x_0\mathtt{Q}_\infty^*)\,\mathtt{P}^T \\
&= \mathtt{P}(\mathbf{F}\mathbf{G}^\top + \mathbf{G}\mathbf{F}^\top)\mathtt{P}^T - x_0\mathtt{P}\mathtt{Q}_\infty^*\mathtt{P}^T \\
&= \left(\underbrace{(\mathtt{P}\mathbf{F})}_{s_\mathrm{f}\mathbf{f}}\underbrace{(\mathtt{P}\mathbf{G})^\top}_{s_\mathrm{g}\mathbf{g}^\top} + \underbrace{(\mathtt{P}\mathbf{F})}_{s_\mathrm{g}\mathbf{g}}\underbrace{(\mathtt{P}\mathbf{G})^\top}_{s_\mathrm{f}\mathbf{f}^\top} \right) - x_0\boldsymbol{\omega}^* \text{ avec } s_\mathrm{f}, s_\mathrm{g} \in \mathbb{R}^* \\
&\sim (\mathbf{f}\mathbf{g}^\top + \mathbf{g}\mathbf{f}^\top) - y_0\boldsymbol{\omega}^* \qquad\qquad \text{avec } y_0 = \frac{x_0}{s_\mathrm{f}s_\mathrm{g}}
\end{aligned}$$

Pour toute quadrique de matrice $\mathtt{Q}^*(x_0)$, il existe donc y_o tel que $\mathtt{C}^* = \mathtt{Y}^* - y_0\boldsymbol{\omega}^*$. ■

6.4 Calcul des images des foyers principaux d'une quadrique de révolution

Supposons maintenant que la caméra soit calibrée, l'image de la conique absolue est alors connue puisque sa matrice $\boldsymbol{\omega}$ (ou $\boldsymbol{\omega}^*$) ne dépend que de la matrice de calibrage $\mathtt{K}$. De plus, si le contour occultant de la QdR considérée est supposée visible, alors on peut raisonnablement supposer que la matrice $\mathtt{C}^*$ soit aussi connue. On peut alors montrer que, sous ces hypothèses, il est possible de déterminer les vecteurs $\mathbf{f}$ et $\mathbf{g}$ représentant les images des foyers principaux de la QdR. Nous faisons remarquer ici que les images des foyers d'une quadrique ne coïncident pas avec les foyers de la conique, image de la quadrique.

Avant d'établir ce résultat dans la prochaine section, remarquons tout d'abord que tout faisceau de quadriques/coniques peut-être engendré par n'importe quelle paire de quadriques/coniques du faisceau. Ainsi, l'image du faisceau tangentiel de QdR homofocales ayant F et G comme foyers principaux, représenté par (6.5), peut être aussi engendré à partir de la paire de coniques de base formée par le contour occultant de la QdR sous forme duale, représenté par $\mathtt{C}^*$, et l'image de la conique absolue, donnée par $\boldsymbol{\omega}^*$. Par la suite, on substituera à la représentation (6.5), la

représentation suivante :

$$\mathtt{A}^*(z) = \mathtt{C}^* - z\boldsymbol{\omega}^*, \tag{6.7}$$

où z est un scalaire $\in \mathbb{R} \cup \{\infty\}$.

Le résultat principal que nous avons établi est qu'il est possible de déduire les images des foyers principaux de la QdR à partir de la donnée du contour occultant d'une QdR obtenue à partir d'une caméra calibrée. Analytiquement, il exprime le fait que si les matrices $\mathtt{C}^*$ et $\boldsymbol{\omega}^* = \mathtt{K}\mathtt{K}^\top$ sont connues, alors il est possible d'identifier de manière unique la matrice $\mathtt{Y}^*$ de la conique duale aux images des foyers principaux, définie en (6.6), parmi les membres dégénérés du faisceau tangentiel representé en (6.7). Ce résultat généralise la contribution décrite dans [Gurdjos 2009], vis-à-vis de laquelle seul le cas des quadriques de révolution de type QdR1 avait été traité.

Afin d'établir ce résultat principal, nous avons besoin d'introduire plusieurs résultats intermédiaires.

Proposition 13 *Le faisceau tangentiel d'images de QdR homofocales contient trois membres dégénérés qui correspondent à trois coniques duales à trois paires distinctes de points. Deux de ces paires sont conjuguées complexes et une paire est réelle et ainsi, deux de ces coniques duales ont pour signature (2,0) et une a pour signature (1,1).*

Démonstration. On considère la représentation (6.7) du faisceau. Ce faisceau comporte des membres dégénérés dont les matrices singulières sont $\mathtt{A}^*(\lambda_i) = \mathtt{C}^* - \lambda_i\boldsymbol{\omega}^*$ et dont le paramètre projectif λ_i est une valeur propre généralisée de la paire de matrices $(\mathtt{C}^*, \boldsymbol{\omega}^*)$ c'est-à-dire une racine du polynôme $p(\lambda) = \det(\mathtt{C}^* - \lambda\boldsymbol{\omega}^*)$. Ce polynôme est de degré 3 en λ : le nombre de membres dégénérés est donc trois.

Pour calculer formellement les paramètres projectifs des trois membres dégénérés, c'est-à-dire les trois valeurs propres généralisées de $(\mathtt{C}^*, \boldsymbol{\omega}^*)$, on peut choisir une certaine représentation projective du plan induite par une homographie choisie adéquatement. On notera tout d'abord que les valeurs propres généralisées de $(\mathtt{C}^*, \boldsymbol{\omega}^*)$ sont aussi les valeurs propres « ordinaires » du produit de matrices $\mathtt{C}^*\boldsymbol{\omega}$, sachant que $\boldsymbol{\omega} = (\boldsymbol{\omega}^*)^{-1}$. En effet, les valeurs propres généralisées de deux matrices $\mathtt{A}$ et $\mathtt{B}$ correspondent aux valeurs propres de la matrice $\mathtt{A}\mathtt{B}^{-1}$. Nous savons également que si $\mathtt{H}$ désigne la matrice d'ordre 3 d'une homographie plane alors les matrices $\mathtt{C}^*\boldsymbol{\omega}$ et

$\mathtt{H}\mathtt{C}^*\boldsymbol{\omega}\mathtt{H}^{-1}$ sont similaires dans le sens où elles ont les mêmes valeurs propres. Ainsi,

$$\begin{aligned}\text{eig}(\mathtt{C}^*, \boldsymbol{\omega}^*) &= \text{eig}(\mathtt{C}^*\boldsymbol{\omega}) = \text{eig}(\mathtt{H}\mathtt{C}^*\boldsymbol{\omega}\mathtt{H}^{-1}) \\ &= \text{eig}(\mathtt{H}\mathtt{C}^*(\mathtt{H}^\top\mathtt{H}^{-\top})\boldsymbol{\omega}\mathtt{H}^{-1}) \\ &= \text{eig}(\mathtt{H}\mathtt{C}^*\mathtt{H}^\top, \mathtt{H}\boldsymbol{\omega}^*\mathtt{H}^\top).\end{aligned}$$

Sans perte de généralité, supposons que cette homographie est telle que :

$$\mathtt{H}\mathtt{C}^*\mathtt{H}^T = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \triangleq \mathtt{D} \text{ et } \mathtt{H}\boldsymbol{\omega}^*\mathtt{H}^T = \mathtt{I}, \tag{6.8}$$

où $a, b, c \in \mathbb{R} \setminus \{0\}$ sont triés par ordre décroissant. Une telle homographie existe de façon non unique et découle, par exemple, de la décomposition spectrale de la matrice

$$\bar{\mathtt{C}}^* = \mathtt{K}^{-1}\mathtt{C}^*\mathtt{K}^{-\top}$$

représentant le contour occultant « calibré » sous sa forme duale. En supposant cette décomposition spectrale de la forme

$$\mathtt{V}\mathtt{D}\mathtt{V}^\top = \bar{\mathtt{C}}^*,$$

où $\mathtt{V}$ est une matrice orthogonale d'ordre 3 et $\mathtt{D}$ est la matrice diagonale d'ordre 3 ayant les valeurs propres de $\bar{\mathtt{C}}^*$ comme éléments diagonaux, alors il s'ensuit que la matrice de l'homographie se décompose en

$$\mathtt{H} = \mathtt{V}^\top\mathtt{K}^{-1}.$$

Les paramètres projectifs λ_i des trois matrices singulières $\mathtt{C}^* - \lambda_i\boldsymbol{\omega}^*$ correspondent alors aux valeurs propres généralisées de $(\mathtt{D}, \mathtt{H}\boldsymbol{\omega}^*\mathtt{H}^T)$, et donc aux valeurs propres ordinaires de $\mathtt{D}$ puisque il est facile de vérifier que $\mathtt{H}\boldsymbol{\omega}^*\mathtt{H}^T = \mathtt{I}$. Ces valeurs propres sont bien sûr les valeurs a, b, c sachant que nécessairement

$$a > 0 \text{ et } c < 0 \tag{6.9}$$

puisque l'image de la QdR n'est pas une conique virtuelle et donc sa signature est nécessairement $(2, 1)$ ce qui implique que les trois valeurs propres de sa matrice ne peuvent jamais être toutes de même signe (voir 5.4). On peut alors déterminer les

matrices $\mathtt{Y}^{*i} = \mathtt{C}^* - \lambda_i \boldsymbol{\omega}^*$ et déduire leur signature à partir des matrices $\hat{\mathtt{Y}}^{*i} = \mathtt{D} - \lambda_i \mathtt{I}$:

$$\hat{\mathtt{Y}}^{*1} = \begin{pmatrix} a-c & 0 & 0 \\ 0 & b-c & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathrm{eig}(\hat{\mathtt{Y}}^{*1}) = \begin{Bmatrix} a-c \\ b-c \\ 0 \end{Bmatrix}, \tag{6.10}$$

$$(\xi_1^1, \xi_2^1) = (2, 0)\,,$$

$$\hat{\mathtt{Y}}^{*2} = \begin{pmatrix} a-b & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & c-b \end{pmatrix}, \quad \mathrm{eig}(\hat{\mathtt{Y}}^{*2}) = \begin{Bmatrix} a-b \\ c-b \\ 0 \end{Bmatrix}, \tag{6.11}$$

$$(\xi_1^2, \xi_2^2) = (1, 1)\,,$$

$$\hat{\mathtt{Y}}^{*3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & b-a & 0 \\ 0 & 0 & c-a \end{pmatrix}, \quad \mathrm{eig}(\hat{\mathtt{Y}}^{*3}) = \begin{Bmatrix} b-a \\ c-a \\ 0 \end{Bmatrix}, \tag{6.12}$$

$$(\xi_1^3, \xi_2^3) = (2, 0)\,.$$

On s'aperçoit que toutes ces matrices représentent des coniques duales à une paire de points : réels dans un cas (ici pour $i = 2$) et conjugués complexes dans les deux autres (ici pour $i \in \{1, 3\}$). ■

On considère à nouveau la droite portée par les images des deux foyers principaux, qui est l'image de l'axe de révolution de la QdR. Cette droite doit couper le contour occultant de la QdR en deux points qui sont nécessairement réels dans le cas des QdR_1 et dans le cas des ellipsoïdes de révolution réels oblates. En revanche elle ne coupe pas le contour occultant de la QdR en des points réels dans le cas des hyperboloïdes de révolution à une nappe (voir Figure 6.1, page 131). Qu'en est-il des droites portées par les paires de points associées aux coniques duales représentées par (6.10-6.12) ? Le résultat suivant répond à cette question.

Proposition 14

- *La droite (réelle) portée par les images de la paire de points réels, représentée sous forme duale par la conique de signature* $(1, 1)$*, coupe toujours le contour occultant de la QdR en deux points réels.*
- *Parmi les deux droites (réelles) portées par les images des deux paires de points complexes conjugués, représentées sous forme duale par les deux coniques de signature* $(2, 0)$*, une seule coupe le contour occultant de la QdR en deux points réels, l'autre le coupant en deux points complexes conjugués.*

Démonstration. Nous établissons la démonstration de ce résultat dans la représentation projective induite par l'homographie représentée par $\mathtt{H}$, en rappelant qu'elle reste valide dans n'importe quelle autre représentation projective du plan (et donc en particulier dans le repère pixellique), puisqu'elle est fondée sur la notion d'incidence qui est une notion projectivement invariante. Nous désignons par la suite « droite de signature s » la droite portée par la paire de points représentée sous forme duale par une conique de signature s. Cette démonstration se décline en quatre points.

– Une droite réelle, de vecteur $\mathbf{d}$, coupe la conique, de matrice $\mathtt{C}$, en une paire de points dont la conique duale a pour matrice [1]

$$\mathtt{A}^* = [\mathbf{d}\,]_\times \mathtt{C}\,[\mathbf{d}]_\times\,,$$

où $[\mathbf{d}]_\times$ est la matrice antisymétrique

$$[\mathbf{d}]_\times = \begin{bmatrix} 0 & -d_3 & d_2 \\ d_3 & 0 & -d_1 \\ -d_2 & d_1 & 0 \end{bmatrix}$$

avec d_1, d_2 et d_3 les composantes de $[\mathbf{d}]_\times$.
Cette paire est réelle si $sig(\mathtt{A}^*) = (1,1)$ et conjuguée complexe si $sig(\mathtt{A}^*) = (2,0)$.

– Les vecteurs des droites, portée par les images des trois paires de points réels représentées sous forme duale par les trois matrices $\hat{\mathtt{Y}}^{*1}, \hat{\mathtt{Y}}^{*2}, \hat{\mathtt{Y}}^{*3}$, sont respectivement $\mathbf{d}_1 = [0,0,1]^T, \mathbf{d}_2 = [0,1,0]^T, \mathbf{d}_3 = [1,0,0]^T$ du fait de la contrainte algébrique $\mathbf{d}_i \in \ker \hat{\mathtt{Y}}^{*i}$.

– La droite de signature $(1,1)$ coupe le contour occultant de la QdR en une paire de points représentée sous forme duale par les matrices $\mathtt{A}_2^* = [\mathbf{d}_2\,]_\times \mathtt{C}\,[\mathbf{d}_2]_\times$. On peut facilement établir que les valeurs propres de cette matrice sont respectivement $(-a, -c)$. De (6.9), on déduit immédiatement que la signature de $\mathtt{A}_2^*$ est $(1,1)$ et donc que cette droite coupe toujours le contour occultant de la QdR en deux points réels.

– Les deux droites de signature $(2,0)$ coupent le contour occultant de la QdR en deux paires de points représentées sous forme duale par les matrices $\mathtt{A}_1^* = [\mathbf{d}_1\,]_\times \mathtt{C}\,[\mathbf{d}_1]_\times$ et $\mathtt{A}_3^* = [\mathbf{d}_3\,]_\times \mathtt{C}\,[\mathbf{d}_3]_\times$. On peut facilement établir que les valeurs

1. La preuve de cette décomposition est décrite par exemple dans [Gurdjos 2009].

propres de ces matrices sont respectivement $(-a, -b)$ et $(-b, -c)$. De (6.9), on déduit immédiatement que les signatures de ces matrices ne peuvent être les mêmes (une est nécessairement $(2, 0)$ et l'autre $(1, 1)$) et donc qu'une seule des droites coupe le contour occultant de la QdR en deux points réels.

■

Proposition 15 *La droite (réelle) portée par les images des foyers principaux d'une QdR réelle coupe toujours le contour occultant de la QdR en deux points réels hormis dans le cas d'un hyperboloïde de révolution à une nappe où elle coupe le contour occultant en deux points complexes conjugués. Parmi les autres membres dégénérés du faisceau tangentiel représenté par (6.7), il n'en existe pas de même signature qui ait cette propriété.*

Démonstration. Nous supposons que les caméras se trouvent à l'extérieur des QdR. Cette hypothèse est très importante surtout dans le cas des hyperboloïdes de révolution à une nappe. Étudions la signature de la conique duale à l'image de la paire des foyers principaux. Le résultat a établi que la conique duale à l'image de la paire des foyers principaux est nécessairement un membre dégénéré du faisceau tangentiel représenté par (6.7) et que, parmi les trois membres dégénérés du faisceau, deux ont pour signature $(2, 0)$ et un seul a pour signature $(1, 0)$. On peut voir les différents cas d'images de QdR dans la Figure 6.1, page 131 (les cas d'ellipsoïdes virtuels sont omis).

- Dans le cas des QdR_1, les images des foyers principaux de la QdR_1 sont réels et donc la conique duale à ces foyers est de signature $(1, 1)$, cf. (5.5). Il existe donc un *unique candidat* à une telle conique dont la signature est $(1, 1)$.
- Dans le cas des QdR_2, les images des foyers principaux de la QdR_1 sont complexes conjugués et donc la conique duale à ces foyers est de signature $(2, 0)$, cf. (5.5). Il existe *deux candidats possibles* à une telle conique dont la signature est $(2, 0)$. Par contre, comme indiqué par le résultat de la proposition 14, une seule des deux droites portées par ces deux paires de points coupe le contour occultant de la QdR en deux points qui sont nécessairement réels. Dans le cas des ellipsoïdes de révolution réels oblates, on choisira la droite coupant le contour occultant de la QdR en deux points réels et dans le cas des hyperboloïdes de révolution à une nappe, on choisira la droite ne coupant pas le contour occultant.

Ainsi, nous venons de définir un critère de sélection opérationnel à la fois pour les QdR1 et les QdR2, à partir des signatures. Ce critère est implémenté dans

l'algorithme 2. ■

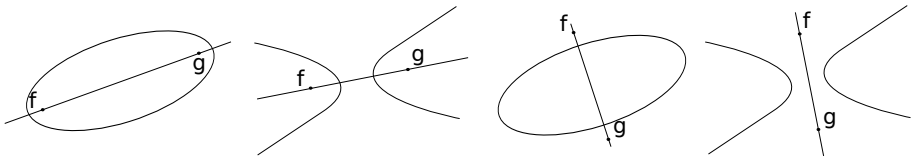

FIGURE 6.1 – Images des différentes types de QdR, de leurs deux foyers et de leur axe de révolution. De gauche à droite, l'ellipsoïde de révolution réel prolate, l'hyperboloïde de révolution à deux nappes, l'ellipsoïde de révolution réel oblate et l'hyperboloïde de révolution à une nappe.

Nous sommes maintenant en mesure d'énoncer la méthode permettant de retrouver l'image de la conique dual aux foyers principaux. *Il s'agit de rechercher la droite portée par une paire de points, duale à une des trois coniques du faisceau tangentiel, qui coupe le contour occultant en deux points réels ou deux points complexes conjugués.*

Pour une QdR1, cette droite est associée à l'unique conique duale qui a pour signature $(1, 1)$.

Pour une QdR2, cette enveloppe est :

- *la conique duale de signature* $(2, 0)$ *qui coupe le contour occultant en deux points réels dans le cas d'un ellipsoïde réel oblate,*
- *la conique duale de signature* $(2, 0)$ *qui ne coupe pas le contour occultant en deux points réels dans le cas d'un hyperboloïde de révolution à une nappe.*

L'algorithme 2 détaille toutes les étapes de la méthode proposée précédemment.

6.5 Reconstruction d'une quadrique de révolution à partir d'au moins deux vues

Nous allons maintenant décrire un schéma linéaire de triangulation permettant la reconstruction d'une QdR à partir d'au moins deux images de celle-ci dans deux vues calibrées. Nous faisons l'hypothèse que nous disposons des matrices de projection $\mathtt{P}_i$ exprimées relativement à une représentation euclidienne arbitraire de l'espace projectif 3D lié à la scène. Nous rappelons que la sous-matrice d'ordre 3 $\bar{\mathtt{P}}_i$, obtenue à partir de $\mathtt{P}_i$ en supprimant sa quatrième colonne et normalisée de telle façon que sa dernière ligne soit de norme unitaire, admet une décomposition unique $\bar{\mathtt{P}}_i = \mathtt{K}_i\mathtt{R}_i$, où $\mathtt{K}_i$ est la matrice de calibrage et $\mathtt{R}_i$ est la matrice orthogonale d'ordre 3 représentant la composante rotationnelle de la pose de la caméra. Cette décomposition

Algorithme 2

– *Entrées.* $\mathtt{C}$: matrice du contour occultant d'une QdR ; $\mathtt{K}$: matrice de calibrage ; type de la QdR : QdR_1 ou QdR_2 (dans ce dernier cas, préciser ellipsoïde de révolution réel oblate ou hyperboloïde de révolution à une nappe).
– *Sortie.* $\mathbf{r}$: vecteur de l'image de l'axe de révolution de la QdR ; (optionnel) $\mathtt{Y}^*$: matrice de la conique duale aux images des foyers principaux de la QdR.

1. $\bar{\mathtt{C}}^* = \mathtt{K}^{-1}\mathtt{C}^*\mathtt{K}^{-\top}$
2. Calculer les valeurs propres et vecteurs propres $\{\lambda_i, \mathbf{V}_i\}_{i=1..3}$ de $\bar{\mathtt{C}}^*$
3. **si** (la QdR est une QdR_1)
   Déterminer $i_0 \in \{1..3\}$ tel que la signature de $\bar{\mathtt{C}}^* - \lambda_{i_0}\mathtt{I}$ soit $(1,1)$
   sinon (la QdR est une QdR_2)
   si la QdR_2 est un ellipsoïde de révolution réel oblate
   Déterminer $i_0 \in \{1..3\}$ tel que
   – la signature de $\bar{\mathtt{C}}^* - \lambda_{i_0}\mathtt{I}$ soit $(2,0)$ et
   – la signature de $[\mathbf{d}_{i_0}]_\times \bar{\mathtt{C}} [\mathbf{d}_{i_0}]_\times$ soit $(1,1)$ où $\mathbf{d}_{i_0} = \mathtt{K}\mathbf{V}_{i_0}$ et $\bar{\mathtt{C}} = (\bar{\mathtt{C}}^*)^{-1}$.
   sinon (la QdR_2 est un hyperboloïde de révolution à une nappe)
   Déterminer $i_0 \in \{1..3\}$ tel que
   – la signature de $\bar{\mathtt{C}}^* - \lambda_{i_0}\mathtt{I}$ soit $(2,0)$ et
   – la signature de $[\mathbf{d}_{i_0}]_\times \bar{\mathtt{C}} [\mathbf{d}_{i_0}]_\times$ soit $(2,0)$ où $\mathbf{d}_{i_0} = \mathtt{K}\mathbf{V}_{i_0}$ et $\bar{\mathtt{C}} = (\bar{\mathtt{C}}^*)^{-1}$.
4. L'image de l'axe de révolution est $\mathbf{r} = \mathtt{K}\mathbf{V}_{i_0}$

peut-être calculée à partir d'une décomposition QR [Golub 1996]. Par conséquent, nous faisons implicitement l'hypothèse d'avoir aussi à notre disposition les matrices de calibrage $\mathtt{K}_i$.

Nous définissons deux types de contraintes. Le premier type est de nature projective et exprime le fait que la rétroprojection d'une droite tangente au contour occultant est un plan tangent à la QdR. Le second type est de nature euclidienne et exprime le fait que la rétroprojection de l'image de l'axe de révolution, calculée par l'algorithme 2, est un plan qui contient les deux foyers principaux de la QdR. Il est à noter que disposer des images des foyers principaux est une condition suffisante mais pas nécessaire. En effet, il nous suffit de disposer de l'image de la droite passant par ces images, c'est-à-dire de l'image de l'axe de révolution, ce qui est une hypothèse moins forte.

6.5.1 Contrainte sur le contour occultant de la QdR.

On va utiliser le fait que la rétroprojection d'un droite tangente au contour occultant d'une quadrique est un plan tangent à la quadrique. Pour transformer

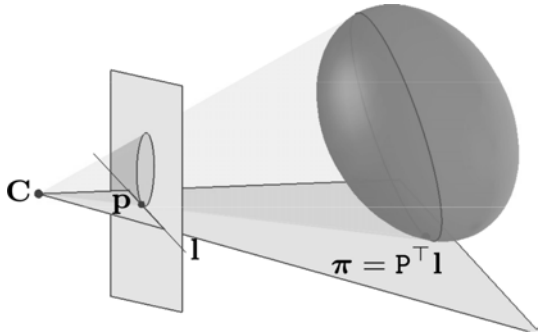

FIGURE 6.2 – La rétroprojection d'une droite tangente à un contour occultant d'une QdR est un plan tangent à cette QdR.

ce fait en une contrainte linéaire sur la quadrique, on rappelle que si $\mathtt{C}^*$ et $\mathtt{Q}^*$ sont les matrices respectivement du contour occultant et de cette quadrique, les deux considérées sous leur forme duale, alors la projection centrale de la quadrique s'écrit : $\mathtt{C}^* \sim \mathtt{P}\mathtt{Q}^*\mathtt{P}^T$, où $\mathtt{P}$ désigne la matrice de projection associée à la caméra. Il suffit alors d'utiliser le fait que la rétroprojection d'une droite de vecteur $\mathbf{l} \in \mathbb{R}^3$ est le plan 3D représenté par le vecteur $\boldsymbol{\pi} = \mathtt{P}^T\mathbf{l} \in \mathbb{R}^4$, dans le repère projectif 3D considéré. Ainsi, si cette droite est tangente au contour occultant, c'est-à-dire si $\mathbf{l}^T\mathtt{C}^*\mathbf{l} = 0$, alors on a

$$0 = \mathbf{l}^T\mathtt{P}\mathtt{Q}^*\mathtt{P}^T\mathbf{l} = \boldsymbol{\pi}^T\mathtt{Q}^*\boldsymbol{\pi}$$

Pour calculer $\mathbf{l}$ on utilisera la relation de polarité entre point d'une conique et droite tangente à la conique occultant, alors $\mathbf{l} = (\mathtt{C}^*)^{-1}\mathbf{p}$. On en déduit le résultat suivant.

Proposition 16 *Soit $\mathtt{C}^*$ la matrice duale du contour occultant d'une QdR et soit $\mathtt{P}$ la matrice de projection de la caméra. Alors de $\pi = \mathtt{P}^T\mathbf{l}$ représentant le plan correspondant à la rétroprojection de la droite $\mathbf{l}$, en utilisant le paramétrage (6.4) de la QdR, on déduit la contrainte linéaire suivante :*

$$\pi^\top\mathtt{X}^*\pi - x_0(\pi^\top\mathtt{Q}_\infty^*\pi) = 0. \tag{6.13}$$

Contrainte sur l'image de l'axe de révolution de la QdR.

Proposition 17 *Si $\mathtt{Y}^*$ est la matrice d'ordre 3 et de rang 2 représentant la conique duale aux images des foyers principaux d'une QdR, définie en (6.6), alors tout vecteur $\mathbf{r}$ vérifiant $\mathtt{Y}^*\mathbf{r} = \mathbf{0}$, représente l'image de l'axe de révolution de la QdR, et*

la contrainte linéaire suivante est satisfaite :

$$\mathtt{X}^*\varphi = \mathbf{0}_4 \tag{6.14}$$

où $\varphi = \mathtt{P}^T\mathbf{r}$ *représente le plan issu de la rétroprojection de l'axe par* $\mathtt{P}$.

Ce résultat découle naturellement du fait que, d'une part (i) $\ker \mathtt{Y}^*$ (de dimension 1) est la droite vectorielle qui représente l'image de l'axe de révolution de la QdR et d'autre part (ii), le plan de rétroprojection de l'image de l'axe de révolution contient les deux foyers principaux, comme cela est illustré sur la Figure 6.3, page 134, et donc peut être représenté par un vecteur de $\ker \mathtt{X}^*$ (de dimension 2).

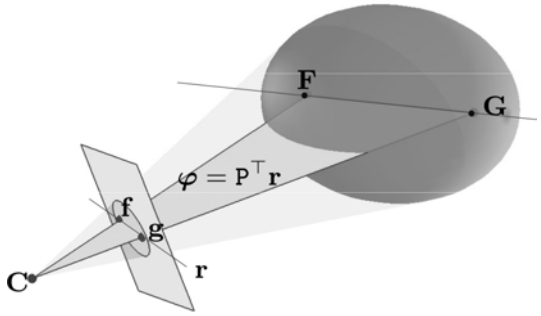

FIGURE 6.3 – La rétroprojection de l'image de l'axe de révolution d'une QdR est un plan contenant cet axe de révolution.

6.5.2 Le système d'équations pour deux vues

On suppose que la QdR est vue par deux caméras calibrées et que ses contours occultants sont représentés par $\mathtt{C}^i$ dans chaque vue $i \in \{1, 2\}$. On a vu que, sous ses hypothèses, on dispose pour chaque vue i de la conique duale aux images des foyers principaux, représentée par $\mathtt{Y}^{*i}$.

Conformément au paramétrage (6.4), l'ensemble des inconnues du problème inclut x_0 plus les 10 éléments la matrice représentant la quadrique duale à la paire de foyers principaux

$$\mathtt{X}^* = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_5 & x_6 & x_7 \\ x_3 & x_6 & x_8 & x_9 \\ x_4 & x_7 & x_9 & x_{10} \end{pmatrix} \tag{6.15}$$

Nous savons que $\mathtt{X}^*$ a seulement 6 degrés de liberté et nous expliquerons plus tard pourquoi, dans le cas de deux vues, la matrice solution est nécessairement de rang

2 et a le nombre de degrés de liberté désiré.

Soient $\mathbf{l}^i_j$, $j = 1..2$, les vecteurs de deux droites tangentes au contour occultant représenté par $\mathtt{C}^i$ et $\mathbf{r}^i \in \ker \mathtt{Y}^{*i}$ le vecteur de l'image de l'axe de révolution. Pour chaque vue, on a deux équations de type (6.13) et une équation de type (6.14), qui nous fournissent dix équations linéaires indépendantes sur les éléments de $\mathtt{X}^*$.

En linéarisant le problème, on peut réécrire les équations (6.13 et 6.14) sous forme matricielle :

$$\begin{pmatrix} \mathtt{A}^1 \\ \mathtt{A}^2 \end{pmatrix} \mathbf{X} = \mathbf{0}_{10}, \tag{6.16}$$

où le vecteur

$$\mathbf{X} = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10})^\top, \tag{6.17}$$

correspond à la « vectorisation » de la matrice $\mathtt{X}^*$ et où

$$\mathtt{A}^i = \begin{bmatrix} (a^i_1)^2+(b^i_1)^2+(c^i_1)^2 & (a^i_2)^2+(b^i_2)^2+(c^i_2)^2 & 0 & 0 & 0 \\ (a^i_1)^2 & (a^i_2)^2 & 0 & 0 & \varphi^i_1 \\ 2a^i_1b^i_1 & 2a^i_2b^i_2 & 0 & \varphi^i_1 & \varphi^i_2 \\ 2a^i_1c^i_1 & 2a^i_2c^i_2 & \varphi^i_1 & 0 & \varphi^i_3 \\ 2a^i_1d^i_1 & 2a^i_2d^i_2 & 0 & 0 & \varphi^i_4 \\ (b^i_1)^2 & (b^i_2)^2 & 0 & \varphi^i_2 & 0 \\ 2b^i_1c^i_1 & 2b^i_2c^i_2 & \varphi^i_2 & \varphi^i_3 & 0 \\ 2b^i_1d^i_1 & 2b^i_2d^i_2 & 0 & \varphi^i_4 & 0 \\ (c^i_1)^2 & (c^i_2)^2 & \varphi^i_3 & 0 & 0 \\ 2c^i_1d^i_1 & 2c^i_2d^i_2 & \varphi^i_4 & 0 & 0 \\ (d^i_1)^2 & (d^i_2)^2 & 0 & 0 & 0 \end{bmatrix}^T, \tag{6.18}$$

$\mathtt{A}^i$ est la matrice 5×11 associée à la vue i, en utilisant pour notation

$$\boldsymbol{\pi}^i_j = (a^i_j, b^i_j, c^i_j, d^i_j)^\top.$$

Les deux premières lignes de $\mathtt{A}^i$ correspondent aux contraintes (6.13), en substituant $\boldsymbol{\pi}^i_j$ par $\boldsymbol{\pi}$ avec $j = 1..2$; les trois dernières lignes de $\mathtt{A}^i$ correspondent aux contraintes (6.14) en substituant $\boldsymbol{\varphi}^i = \mathtt{P}^{i\top}\mathbf{r}^i$ à $\boldsymbol{\varphi}$. Puisque nous disposons de deux vues $i \in \{1, 2\}$ la matrice $\begin{pmatrix} \mathtt{A}^1 \\ \mathtt{A}^2 \end{pmatrix}$ est de taille 10×11.

Nous obtenons donc un système linéaire de 10 équations homogènes qui est

donc bien posé dans le sens où rank $\begin{pmatrix} \mathtt{A}^1 \\ \mathtt{A}^2 \end{pmatrix} = 10$. En considérant, sans perte de généralité, la contrainte $\|\mathbf{X}\|^2 = 1$ pour éviter la solution triviale $\mathbf{0}_{11}$, il est possible d'obtenir une solution exacte de ce problème. Le fait que deux contraintes (6.14) soient imposées exactement nous assurent que rank $\mathtt{X}^* = 2$.

6.5.3 Cas où le nombre de vues est supérieur à 2

En pratique, lorsque $n \geq 2$ vues calibrées sont disponibles, le système à résoudre déduit de (6.16) s'écrit $\mathtt{D}\mathbf{X} \approx \mathbf{0}_{5n}$, où $\mathtt{D}$ est une matrice $5n \times 11$ contenant tous les blocs $\mathtt{A}^i$, $i = 1..n$, et l'opérateur '$\approx$' exprime le fait que les données contenues dans la matrice $\mathtt{A}^i$ sont généralement perturbées par le bruit et donc qu'il n'existe pas en général de solution exacte. On résout alors un problème au sens des moindres carrés $\min_{\mathbf{X}} \|\mathtt{D}\mathbf{X}\|^2$ sous la contrainte $\|\mathbf{X}\|^2 = 1$. La solution peut alors être calculée comme le vecteur singulier de la matrice $\mathtt{D}$ associée à la plus petite valeur singulière [Hartley 2003, p. 593]. Il est à noter que, dans ce cas, la contrainte rank $\mathtt{X}^* = 2$ n'est pas prise en compte et qu'elle doit être imposée a posteriori.

6.5.4 Algorithme

L'algorithme 3 détaille les différentes étapes de la construction de la matrice $\mathtt{D}$.

Algorithme 3 *Entrées* $\{\mathtt{P}^i\}$: matrices de projection des vues $i = 1..n, n \geq 2$; $\{\mathtt{C}^i\}$: matrices des contours occultants. *Sortie* $\mathtt{Q}^*$: matrice de la QdR duale.

Pour chaque vue i

1. Choisir deux points distincts $\mathbf{p}^i_j$ de $\mathtt{C}^i$, avec $j \in \{1..2\}$.
2. Calculer $\mathbf{l}^i_j = \mathtt{C}^i\mathbf{p}^i_j$ les droites tangentes à $\{\mathtt{C}^i\}$ en $\mathbf{p}^i_j$.
3. Calculer $\boldsymbol{\pi}^i_j = \mathtt{P}^{i\top}\mathbf{l}^i_j$ les plans correspondant aux rétroprojections des $\mathbf{l}^i_j$.
4. Utiliser l'algorithme 2 pour retrouver $\mathbf{r}^i$, l'image de l'axe de révolution.
5. Calculer $\boldsymbol{\varphi}^i = \mathtt{P}^{i\top}\mathbf{r}^i$ les plans correspondant aux rétroprojections des $\mathbf{r}^i$.
6. Construire les blocs $\mathtt{A}^i$, comme défini dans (6.18), puis construire $\mathtt{D}$.
7. Chercher $\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \|\mathtt{D}\mathbf{X}\|^2$ sous contrainte $\|\mathbf{X}\|^2 = 1$.
8. Construire $\hat{\mathtt{X}}^*_o$ comme étant le vecteur contenant les 10 derniers éléments de $\hat{\mathbf{X}}$ et notons x_0 le premier élément de $\hat{\mathbf{X}}$.
9. Chercher $\hat{\mathtt{X}}^* = \arg\min_{\mathtt{M}} \left\|\hat{\mathtt{X}}^*_o - \mathtt{M}\right\|^2_F$ sous contrainte $rang(\mathtt{M}) = 2$.
10. Construire $\mathtt{Q}^* = \hat{\mathtt{X}}^* - x_0\mathtt{Q}^*_\infty$

6.6 Reconstruction d'une quadrique de révolution de paramètres connus à partir d'une seule image

Le problème que nous abordons ici est celui du calcul de la pose d'une QdR à partir d'un seul contour occultant. La seule hypothèse que nous faisons vis-à-vis de la QdR est que nous connaissons les petits et grands axes a et b de la conique de révolution qui l'engendre.

Dans l'espace affine euclidien de dimension 3, plongé naturellement dans l'espace projectif de même dimension, on considère un repère (S') ayant pour l'origine O le centre de la QdR et d'axe Oz, l'axe de révolution de la QdR (Figure 6.4, page 138). Dans ce nouveau repère, puisque la QdR a pour axes a et b, les vecteurs des deux foyers sont de la forme 0 $\mathbf{F}' = (0, 0, d, 1)^T$ et $\mathbf{G}' = (0, 0, -d, 1)^T$ avec :

- $d = \sqrt{b^2 - a^2}$ si $\mathtt{Q}$ représente la matrice d'un ellipsoïde de révolution réel (prolate si $a < b$ et oblate si $b < a$),
- $d = \sqrt{a^2 - b^2}$ si $\mathtt{Q}$ représente la matrice d'un ellipsoïde de révolution imaginaire (prolate si $a < b$ et oblate si $b < a$),
- $d = a\sqrt{1 + \frac{b^2}{a^2}}$ si $\mathtt{Q}$ représente la matrice d'un hyperboloïde de révolution à deux nappes,
- $d = a\sqrt{-1 - \frac{b^2}{a^2}}$ si $\mathtt{Q}$ représente la matrice d'un hyperboloïde de révolution à une nappe.

Remarque : d peut comporter une partie imaginaire dans les cas des QdR_2. De plus, le cas du paraboloïde de révolution n'est pas considéré ici car l'un de ses foyers se situe à l'infini.

Concernant la caméra, nous faisons la seule hypothèse de connaître la matrice de calibrage $\mathtt{K}$. La matrice de projection est de la forme $\mathtt{P}_i = \mathtt{K}[\mathtt{R}|\mathbf{t}]$ exprimée relativement au repère (S) où $\mathbf{r}_3$, la troisième colonne de $\mathtt{R}$, représente le vecteur directeur unitaire de l'axe de révolution et $\mathbf{t}$ a pour composantes les coordonnées cartésiennes du centre O de la QdR. À partir du contour occultant d'une QdR dans une image, on a vu comment retrouver la paire d'images $(\mathbf{f}, \mathbf{g})$ des foyers principaux $(\mathbf{F}, \mathbf{G})$ de la QdR.

Les équations de projection des foyers s'écrivent $\mathtt{P}\mathbf{F} = \lambda\mathbf{f}$ et $\mathtt{P}\mathbf{G} = \mu\mathbf{g}$ où λ et μ désignent deux scalaires. Les inconnues sont ici le vecteur $\mathbf{r}_3$ et le vecteur $\mathbf{t}$. On déduit le système d'équations suivant :

$$\begin{cases} d\mathtt{K}\mathbf{r_3} + \mathtt{K}\mathbf{t} = \lambda\mathbf{f} \\ -d\mathtt{K}\mathbf{r_3} + \mathtt{K}\mathbf{t} = \mu\mathbf{g} \end{cases} \Leftrightarrow \begin{cases} (d\mathtt{K}\mathbf{r_3} + \mathtt{K}\mathbf{t}) \wedge \mathbf{f} = 0 \\ (-d\mathtt{K}\mathbf{r_3} + \mathtt{K}\mathbf{t}) \wedge \mathbf{g} = 0 \end{cases} \tag{6.19}$$

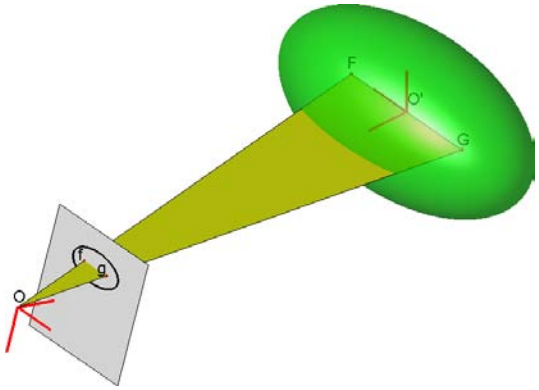

FIGURE 6.4 – Représentation d'une quadrique dans le repère caméra et dans le repère centré en son centre et d'axe Oz, son axe de révolution.

Ce système n'a que quatre équations linéaires indépendantes alors que le problème a cinq degrés de liberté. Supposons que l'on détermine la famille de solutions de ce système sous-déterminé par exemple par l'intermédiaire d'une combinaison linéaire de deux solutions de base $\mathbf{U}$ et $\mathbf{V}$. On peut maintenant se ramener à un problème à deux inconnues u et v avec

$$\begin{pmatrix} \mathbf{R_3} \\ \mathbf{t} \end{pmatrix} = \begin{pmatrix} u\mathbf{U} + v\mathbf{V} \end{pmatrix}.$$

où les coefficient u et v sont inconnus.

Pour retrouver les autres colonnes de la matrice de rotation, on posera $\mathbf{r_1} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \wedge \frac{\mathbf{v}}{\|\mathbf{v}\|}$ où $\mathbf{u}$ (respectivement $\mathbf{v}$) désigne les trois premières composantes de $\mathbf{U}$ (respectivement $\mathbf{V}$) et $\mathbf{r_2} = \mathbf{r_1} \wedge \mathbf{r_3}$. En effet, la quadrique étant de révolution, $\mathbf{r_1}$ et $\mathbf{r_2}$ doivent seulement être choisis orthogonaux à $\mathbf{r_3}$. La solution proposée ci-dessus convient parfaitement car $\mathbf{u}$ et $\mathbf{v}$ appartiennent au plan correspondant à la rétroprojection de la droite passant par $\mathbf{f}$ et $\mathbf{g}$ vu que les équations de 6.19 sont respectées. Ce plan contenant $\mathbf{r_3}$, les vecteurs $\mathbf{r_1}$ et $\mathbf{r_2}$ sont bien orthogonaux à $\mathbf{r_3}$.

6.6.1 Utilisation de deux nouvelles contraintes

Dans l'espace dual, on sait qu'un plan π tangent à une quadrique $\mathtt{Q}^*$ vérifie :

$$\pi^T \mathtt{Q}^* \pi = 0 \tag{6.20}$$

En prenant deux droites $\mathbf{l_1}$ et $\mathbf{l_2}$ tangentes à $\mathtt{C}$ dans l'image, on peut calculer les plans $\pi_\mathbf{1} = \mathtt{P'^T}\mathbf{l_1}$ et $\pi_\mathbf{2} = \mathtt{P'^T}\mathbf{l_2}$. Sachant que dans (S), la matrice de la quadrique

duale s'écrit :

$$\mathtt{Q}^* = \begin{pmatrix} \sigma_a a^2 & 0 & 0 & 0 \\ 0 & \sigma_a a^2 & 0 & 0 \\ 0 & 0 & \sigma_b b^2 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \tag{6.21}$$

avec :

- $\sigma_a = 1$ et $\sigma_b = 1$ avec $a < b$ si $\mathtt{Q}^*$ est la matrice duale d'un ellipsoïde de révolution réel prolate.
- $\sigma_a = 1$ et $\sigma_b = 1$ avec $a > b$ si $\mathtt{Q}^*$ est la matrice duale d'un ellipsoïde de révolution réel oblate.
- $\sigma_a = -1$ et $\sigma_b = -1$ avec $a < b$ si $\mathtt{Q}^*$ est la matrice duale d'un ellipsoïde de révolution imaginaire prolate.
- $\sigma_a = -1$ et $\sigma_b = -1$ avec $a > b$ si $\mathtt{Q}^*$ est la matrice duale d'un ellipsoïde de révolution imaginaire oblate.
- $\sigma_a = -1$ et $\sigma_b = 1$ si $\mathtt{Q}^*$ est la matrice duale d'un hyperboloïde de révolution à deux nappes.
- $\sigma_a = 1$ et $\sigma_b = -1$ si $\mathtt{Q}^*$ est la matrice duale d'un hyperboloïde de révolution à une nappe.

On obtient donc deux nouvelles équations quadratiques en u et v :

$$\begin{cases} \pi_1^T \mathtt{Q}^* \pi_2 = 0 \\ \pi_2^T \mathtt{Q}^* \pi_2 = 0 \end{cases} \tag{6.22}$$

que l'on peut écrire sous la forme :

$$\begin{cases} \begin{pmatrix} u & v & 1 \end{pmatrix} \mathtt{M^1} \begin{pmatrix} u & v & 1 \end{pmatrix}^T = 0 \\ \begin{pmatrix} u & v & 1 \end{pmatrix} \mathtt{M^2} \begin{pmatrix} u & v & 1 \end{pmatrix}^T = 0 \end{cases} \tag{6.23}$$

où :

$$\mathtt{M^i} = \begin{pmatrix} m_{11}^i & m_{12}^i & 0 \\ m_{12}^i & m_{22}^i & 0 \\ 0 & 0 & m_{33}^i \end{pmatrix} \tag{6.24}$$

Les valeurs des composantes des $\mathtt{M^i}$, pour $i = 1, 2$, sont :

$$
\begin{aligned}
m^i_{11} &= \sigma_a T_2^2 a^2 + \sigma_b S_6^2 b^2 - S_4^2 \\
m^i_{22} &= \sigma_a T_1^2 a^2 + \sigma_b S_7^2 b^2 - S_5^2 \\
m^i_{33} &= \sigma_a (S_2 l^i_1 + S_3 l^i_2 + S_1 l^i_3)^2 a^2 \\
m^i_{12} &= \sigma_a T_1 T_2 a^2 + \sigma_b S_6 S_7 b^2 - S_4 S_5
\end{aligned}
$$

en utilisant les notations suivantes :

$$
\begin{aligned}
S_1 &= (U_1 V_2 - U_2 V_1)/||\mathbf{u} \wedge \mathbf{v}|| \\
S_2 &= (U_2 V_3 - U_3 V_2)/||\mathbf{u} \wedge \mathbf{v}|| \\
S_3 &= (U_3 V_1 - U_1 V_3)/||\mathbf{u} \wedge \mathbf{v}|| \\
S_4 &= (U_4 l^i_1 + U_5 l^i_2 + U_6 l^i_3) \\
S_5 &= (V_4 l^i_1 + V_5 l^i_2 + V_6 l^i_3) \\
S_6 &= (U_1 l^i_1 + U_2 l^i_2 + U_3 l^i_3) \\
S_7 &= (V_1 l^i_1 + V_2 l^i_2 + V_3 l^i_3) \\
T_1 &= (S_3 V_3 - S_1 V_2) l^i_1 + (S_1 V_1 - S_2 V_3) l^i_2 + (S_2 V_2 - S_3 V_1) l^i_3 \\
T_2 &= (S_3 U_3 - S_1 U_2) l^i_1 + (S_1 U_1 - S_2 U_3) l^i_2 + (S_2 U_2 - S_3 U_1) l^i_3
\end{aligned}
$$

Ici, U_j (respectivement V_j et l^i_j) correspond à la $j^{ème}$ composante de $\mathbf{U}$ (respectivement de $\mathbf{V}$ et de $\mathbf{l^i}$).

Les solutions (u, v) sont les points se trouvant à la fois sur la conique associée à la matrice $\mathtt{M^1}$ mais aussi sur la conique associée à la matrice $\mathtt{M^2}$. Il suffit donc de trouver les intersections de ces deux coniques qui sont, en général, 4 points [de Almeida Barreto 2003, p. 55].

$\mathtt{M^1}$ et $\mathtt{M^2}$ n'ayant pas de termes linéaires en u et en v, les quatre solutions possibles sont de la forme $(u1, v1)$, $(u2, v2)$, $(-u1, -v1)$ et $(-u2, -v2)$ (Figure 6.5, page 141).

Dans un premier temps, sachant que la QdR se situe devant la caméra, deux solutions peuvent être écartées (celles vérifiant $uU_6 + vV_6 < 0$). Il reste donc deux possibilités. Le contour occultant seul ne permet pas de départager ces deux possibilités. Elles sont souvent très proches mais elles sont distinctes.

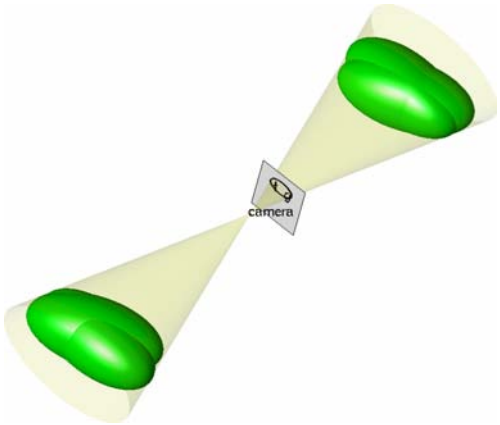

FIGURE 6.5 – Les 4 quadriques possibles de mêmes paramètres se projetant de manière identique dans l'image.

6.7 Étude des performances des méthodes proposées

Nous allons maintenant évaluer les performances de nos méthodes pas rapport aux autres méthodes déjà existantes. Pour cela, nous utilisons le logiciel *MATLAB* pour simuler des scènes avec des QdR. Puis nous les projetons dans des images. Nous ajoutons alors du bruit sur ces projections puis, nous tentons de reconstruire ces QdR afin de les comparer aux QdR originales.

6.7.1 Reconstruction de QdR à partir de plusieurs contours occultants

À notre connaissance, il n'existe pas de méthode générale permettant de reconstruire tout type de QdR à partir de seulement deux contours occultants. Nous faisons le choix de comparer notre méthode à celle de Cross [Cross 1998] sachant que leur méthode utilise au moins trois vues.

Dans un premier temps, nous évaluons la robustesse de notre méthode vis-à-vis du bruit. Pour cela, nous simulons 1000 configurations différentes. Pour chacune de ces configurations, nous plaçons aléatoirement trois caméras en faisant varier leur position et leur orientation tout en faisant attention à ce qu'elles restent à une distance relativement constante de la QdR et à ce qu'elles visent un point situé sur la surface d'une sphère centrée en l'origine et de rayon variable, proche du dixième de la distance entre le centre de la scène et la caméra. Puis, un ellipsoïde de révolution est généré avec son centre à l'intérieur de cette sphère. Les images exactes de cet ellipsoïde sont calculées. Ce sont des ellipses étant donné que les

conditions de la scène simulée sont telles que le plan à l'infini de chaque caméra ne peut pas intersecter l'ellipsoïde. Ces ellipses sont ensuite bruitées de la manière suivante : on discrétise chaque ellipse en 30 points auxquels on ajoute un bruit gaussien de moyenne 0 et de variance $\sigma = n\%$ du grand axe de l'ellipse. Une nouvelle ellipse est alors ajustée sur ces points bruités. On obtient trois nouvelles ellipses à partir desquelles on peut utiliser l'algorithme 3 pour retrouver une QdR. On peut alors calculer différents types d'erreurs entre cette QdR reconstruite et l'ellipsoïde original comme le pourcentage de différence entre les petits axes et les grands axes, la distance absolue entre les centres, l'angle entre les axes de révolution et le pourcentage de différence antre les volumes. Ces erreurs sont représentées sur la Figure 6.6, page 143 en utilisant notre méthode et celle de Cross avec trois vues. On peut remarquer que ces erreurs augmentent linéairement en fonction du bruit et que les résultats de notre méthode sont meilleurs que ceux de Cross.

Dans une deuxième expérience, nous regardons l'importance du nombre de vues. Pour cela, nous répétons la même expérience que précédemment mais en faisant varier le nombre de caméras de deux à huit et en gardant un bruit gaussien fixé avec $\sigma = 2\%$ du grand axe de l'ellipse. Les résultats présentés dans la Figure 6.7, page 144 représentent des courbes où chaque point est une erreur moyenne calculée à partir de cinq cents simulations différentes dans le but de lisser les courbes. Les résultats de Cross ne commencent qu'à partir de trois vues. Dû au nombre de plus en plus important de contraintes, les erreurs diminuent exponentiellement quand le nombre de vues augmente. Une fois encore, notre méthode est plus performante même si l'écart diminue lorsque le nombre de caméras augmente.

6.7.2 Reconstruction de QdR de paramètres connus à partir d'un seul contour occultant

Nous n'avons pas trouvé d'autres algorithmes permettant de retrouver la position et l'orientation d'une QdR générale à partir d'une seule image. Dans [Ferri 1993], la méthode de Ferry *et al.* implique quelques problèmes de calculs dans le cas général. Ici, nous nous contentons de montrer les performances de notre méthode en fonction du bruit en reprenant les mêmes conditions des expériences de la partie 6.7.1. Les QdR considérées sont une nouvelle fois les ellipsoïdes de révolution. Par rapport aux premières expérimentations, trois des erreurs calculées sont nécessairement nulles vu que l'on suppose connu les paramètres intrinsèques de la QdR. De plus, la distance entre le centre de la QdR originale et le centre de la QdR

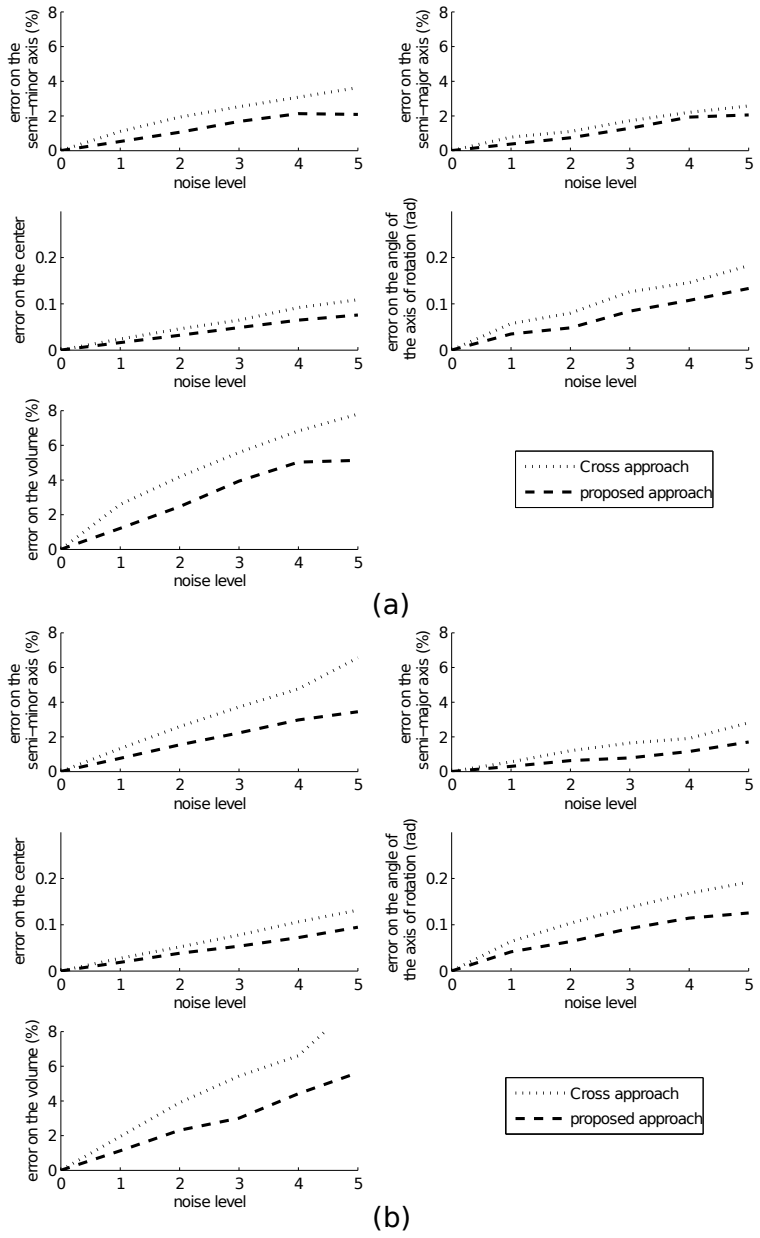

FIGURE 6.6 – Notre algorithme est comparé à celui de Cross (voir texte) pour évaluer la robustesse des méthodes par rapport au bruit. (a) Tests sur les ellispoïdes prolates ($\in \mathrm{QdR}_1$). (b) Tests sur les ellispoïdes oblates ($\in \mathrm{QdR}_2$).

reconstruite n'a plus vraiment de sens si on ne la compare pas à une autre méthode. Nous gardons donc l'erreur d'angle entre les axes de révolutions des deux QdR puis, nous calculons comme deuxième erreur la distance entre les projections des centres des deux QdR (originale et reconstruite) que nous divisons par le grand axe de l'image de la QdR originale. Ces erreurs sont montrés dans la Figure 6.8,

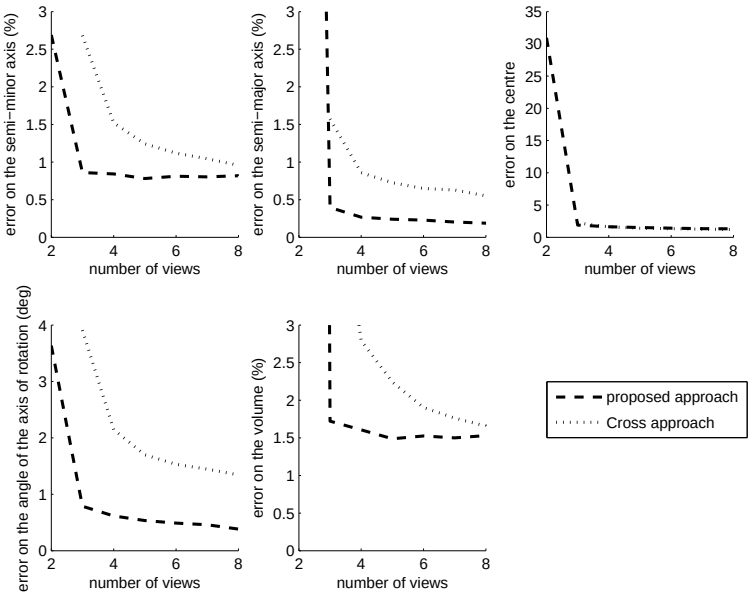

FIGURE 6.7 – Notre algorithme est comparé à celui de Cross (voir texte) pour regarder la performance des méthodes par rapport au nombre de caméras utilisées.

page 145. Pour chaque niveau de bruit, 500 simulations sont réalisées et on calcule la moyenne afin de lisser les courbes. Nous avons vu dans la partie 6.6 que notre méthode ne permet pas de retrouver une solution unique mais deux possibilités. Pour calculer les erreurs, étant donné qu'il y a aucun moyen de déterminer laquelle des deux possibilités est la bonne, nous choisissons de prendre la solution donnant l'erreur la plus petite. Bien que les deux possibilités donnent des QdR très proches l'une de l'autre et donc avec des erreurs similaires, nous avons choisi cette option pour montrer que parmi les possibilités, la meilleure a des erreurs très faibles.

On peut encore remarquer que ces erreurs augmentent linéairement lorsque le bruit augmente et restent relativement faibles même avec un bruit de 5% ($< 2\%$ pour les projections des centres et <0.17 radians pour les angles entre les axes de révolution).

6.8 Applications

6.8.1 Projet SODIMEL

L'intérêt que nous avons porté à la reconstruction de quadriques de révolution nous est venu dans le cadre d'un projet avec l'industriel SODIMEL. Cette entreprise, basée dans la région bordelaise vend du matériel de mesures en particulier dans le

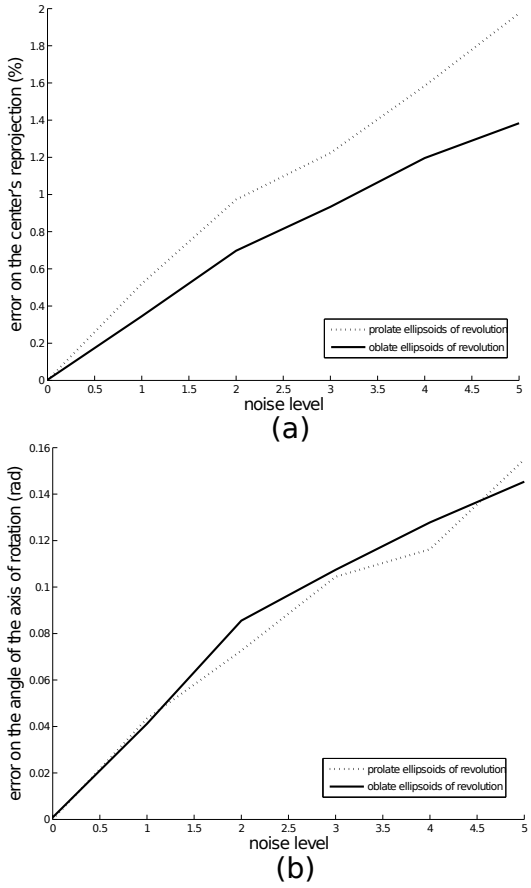

FIGURE 6.8 – Erreurs sur les reconstructions des QdR de paramètres connus à partir d'une seule image (voir texte). (a) Erreurs sur les projections des centres (%). (b) Erreurs sur l'angle entre les axes de révolution (rad).

domaine viti-vinicole. Nous avons développé pour SODIMEL un appareil de mesure capable d'estimer le volume d'une grappe de raisin à partir d'une vision binoculaire (Figure 6.9, page 146).

6.8.2 Modélisation d'une grappe de raisin à partir de deux images

Dans cette partie, nous décrivons les différentes étapes permettant la modélisation de la grappe de raisin à partir de deux images. Chaque baie de la grappe est assimilé à un ellipsoïde de révolution. Celui-ci n'étant pas intersecté par le plan de la caméra, son image est nécessairement une ellipse. Nous décrivons comment reconstruire les baies selon si elles sont visibles dans les deux vues, dans une seule vue ou dans aucune vue. Bien entendu, nous ne prétendons pas reconstruire quelque chose que l'on ne voit pas mais l'on va tout de même détailler une méthode per-

FIGURE 6.9 – Dispositif permettant d'acquérir deux images d'une grappe de raisin dans le but d'estimer son volume.

mettant d'obtenir une grappe de raisin réaliste et cohérente vis-à-vis de sa partie visible dans les images.

Dans un premier temps, les paramètres intrinsèques des deux caméras sont précalculés, grâce à la procédure de calibrage décrite dans [Gurdjos 2005] grâce à des mires composées de cercles concentriques.

6.8.2.1 Reconstruction d'une baie à partir de deux vues

Les matrices de projection étant connues, nous utilisons la méthode détaillée dans la section 6.5 pour reconstruire l'ensemble des baies visibles dans les deux images. Une phase de détection d'ellipses et d'appariements sont donc nécessaires.

En ce qui concerne la détection des ellipses, nous utilisons un algorithme très similaire à la transformée de Hough en ajoutant des contraintes liées à la direction des gradients dans l'image et en restreignant les dimensions des espaces de recherche (connaissant les tailles moyennes des baies et les conditions d'acquisition, on connaît la taille minimal et la taille maximal des ellipses à détecter). Ensuite, les ellipses sont appariées. Il est beaucoup plus facile d'apparier des ellipses que des points. En effet, dans le cas d'un point, la contrainte épipolaire permet de restreindre les appariements possibles à un point de l'image de gauche à toute une droite dans l'image de droite. En revanche, dans le cas d'une ellipse, seules quelques ellipses (en général 3 ou 4) de l'image de droite sont retenues comme appariements possibles avec une ellipse de l'image de gauche. On doit cependant déterminer quel est le bon appariement. Pour chaque possibilité, nous reconstruisons une quadrique et nous la reprojetons dans les images. Seule la quadrique minimisant l'erreur de reprojection

est conservée.

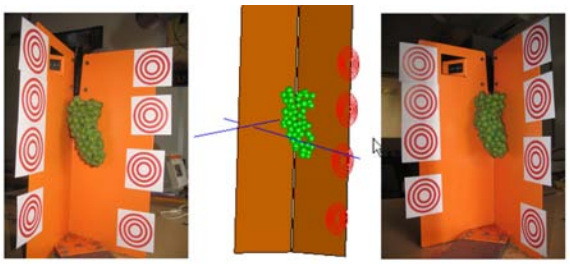

FIGURE 6.10 – Exemple de reconstructions de baies visibles sur les deux images. De part et d'autre, nous pouvons voir les images acquises par le dispositif de prises de vues et au centre la reconstruction des baies visibles ainsi que des plans orange. Les axes optiques des caméras sont représentés par des segments bleus.

6.8.2.2 Reconstruction d'une baie à partir d'une seule vue

Une fois un ensemble de quadriques de révolution reconstruits à partir de deux images, on peut faire un apprentissage sur les paramètres de ces quadriques. On a remarqué que ces paramètres sont tous très proches pour une même grappe. Cela va nous permettre de reconstruire les baies visibles sur une seule vue en les supposant de paramètres égaux à la moyenne des paramètres des quadriques de révolution déjà reconstruites. On utilise l'algorithme détaillé dans la section 6.6 tout en vérifiant cependant à chaque fois la cohérence de la quadrique de révolution reconstruite.

6.8.2.3 Ajout de nouvelles baies

Dans un dernier temps, on cherche à compléter la grappe jusqu'à présent reconstruite. En effet, il reste une partie invisible de la grappe que l'on peut seulement supposer similaire à la partie reconstruite dans le sens où ayant la même densité dans la disposition des baies. On ajoute donc de nouvelles baies une par une entre les baies reconstruites et le cadre orange visible sur la Figure 6.10, page 147. On vérifie à chaque fois si sa position respecte la densité des baies déjà reconstruites (écartement avec ses baies voisines). Si ce n'est pas le cas, on ne retient pas cette baie. Un exemple de résultat est montré sur le Figure 6.11, page 148.

6.8.3 Modélisation d'une quadrique de révolution à partir d'images

Nous avons également fait des expérimentations avec des QdR disposées sur un échiquier. Ce dernier nous sert à étalonner les caméra afin de déterminer les para-

FIGURE 6.11 – Reconstruction d'une grappe de raisin (au centre) à partir de deux vues (image de gauche et image de droite).

mètres intrinsèques et extrinsèques. Puis, nous utilisons l'algorithme 3 après avoir détouré les images des QdR manuellement. Le but ici est de tester notre algorithme de reconstruction et c'est la raison pour laquelle on suppose connus les contours occultants. Dans la Figure 6.12, page 148, les deux images originales se trouvent sur la gauche. Sur la droite, on voit la reprojection de la scène 3D reconstruite ainsi que la position des deux caméras dont on était prises les images de gauche. 26 baies de raisin et 3 tomates sont reconstruites en les supposant assimilables à des ellipsoïdes de révolution.

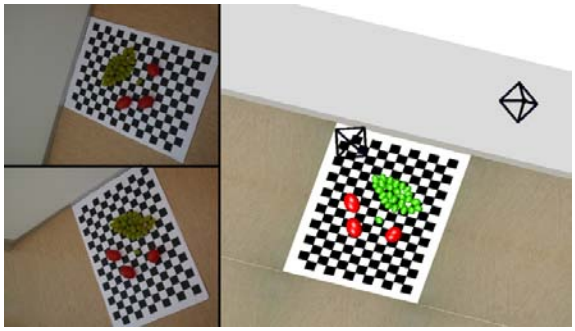

FIGURE 6.12 – Reconstruction de fruits vus comme des ellipsoïdes de révolution réels prolates.

Sur la Figure 6.13, page 149, divers objets ont été reconstruits à partir de deux images. Un échiquier est également utilisé pour étalonner les images. Une des images originales est représenté en haut alors qu'une image reprojetée de la scène reconstruite est affichée dessous. On peut remarquer que ces objets ont différentes formes de QdR : 1 ellipsoïde de révolution prolate, 3 ellipsoïdes de révolution oblates, 1 hyperboloïde de révolution à une nappe et 1 cylindre. On rappelle ici que la matrice

d'un cylindre est dégénérée, de rang 3 et que cette contrainte doit donc être imposée a posteriori.

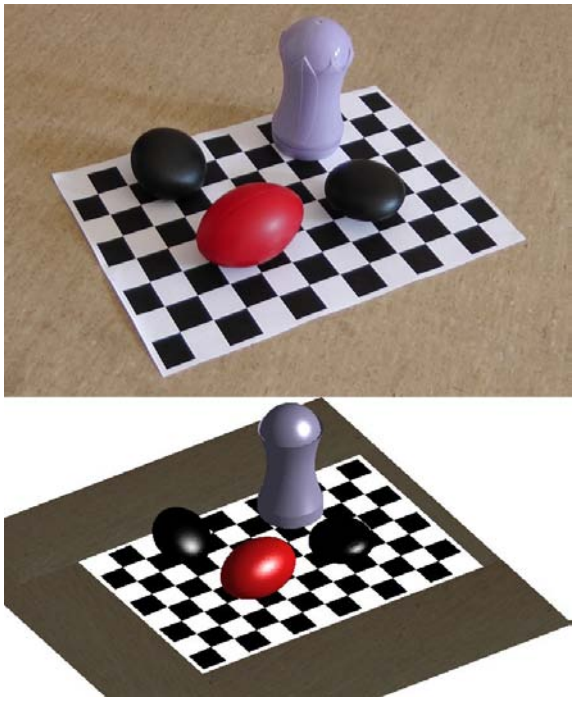

FIGURE 6.13 – Reconstruction de quadriques de révolution. On peut voir un ellipsoïde de révolution prolate, trois ellipsoïdes de révolution oblates, un hyperboloïde de révolution à une nappe et un cylindre.

Conclusion, limites et perspectives de la partie II

Dans cette deuxième partie, les principales contributions apportées sont :

- un nouvel algorithme permettant de retrouver les images des foyers principaux d'une quadrique de révolution à partir du contour occultant dans une vue calibrée,
- une nouvelle méthode de reconstruction de quadriques de révolutions à partir d'au moins deux vues calibrées,
- une nouvelle méthode permettant de retrouver la position et l'orientation d'une quadrique de révolution de paramètres connus à partir d'un seul contour occultant dans une vue calibrée.

Ces travaux ont donné lieu à deux articles :

- Multiple View Reconstruction of a Quadric of Revolution from its Occluding Contours, *Pierre Gurdjos, Vincent Charvillat, Géraldine Morin, Jérôme Guénard* publié et accepté en présentation orale à **ACCV** en 2009,
- De la Reconstruction de Quadriques de Révolution à partir d'Images à la Complémentation d'Objets Naturels, *Jérôme Guénard, Géraldine Morin, Pierre Gurdjos, Vincent Charvillat* publié et accepté en présentation orale à l'**AFIG** en 2009.

La première contribution de cet article consiste en une nouvel méthode permettant le calcul des images des foyers principaux d'une QdR à partir d'un contour occultant dans une seule vue calibrée et du type de la QdR (on doit en effet connaître si les foyers principaux de la QdR sont réels ou complexes et donc si la quadrique $\in$ QdR_1 ou QdR_2). Ce résultat (dont le détail est donné dans l'algorithme 2)est très important car il a permis le développement de nouveaux algorithmes permettant de reconstruire des QdR. En particulier, nous avons proposé un schéma de triangulation linéaire pour reconstruire la QdR à partir d'au moins deux contours occultants dans deux vues calibrées. Une autre méthode permettant le calcul de la position d'une QdR de paramètres connus à partir d'un seul contour occultant est présentée. Toutes ces méthodes ont été évaluées et les résultats sont très satisfaisant aussi bien sur des données de synthèse qu'à partir d'images réelles. Ces différents algorithmes permettent la reconstruction d'un grand nombre d'objets possédant des surfaces

quadratiques.

Une des limites de cette méthode est le fait de travailler sur des vues calibrées. Cela implique de calculer les paramètres intrinsèques et extrinsèques des caméras en amont. Il serait intéressant de chercher à savoir s'il est possible de calibrer la scène directement à partir des contours occultants des quadriques. L'avantage de la méthode de Cross *et al.* [Cross 1998] est qu'elle permet de reconstruire les quadriques projectivement, sans connaissance de la matrice de calibrage.

Une deuxième limite est de devoir donner le type de la QdR en entrée de nos algorithmes. Cela implique de devoir le détecter à partir des contours occultants. Bien qu'il est très facile de le détecter pour certaines QdR (un hyperboloïde à une nappe est automatiquement une QdR_2 car il possède des foyers principaux complexes alors qu'un hyperboloïde à deux nappes est automatiquement une QdR_2 car il possède des foyers principaux réels), il est parfois plus difficile de déterminer si un ellipsoïde de révolution est prolate ou oblate. Nos méthodes ne permettent pas, pour le moment, de détecter le type de la QdR mais il serait intéressant de savoir si c'est possible.

Nous n'avons pas présenté de reconstructions de surfaces plus complexes qui peuvent être décomposées en plusieurs morceaux assimilables à des surfaces quadratiques. Cela pourrait être fait dans de futurs travaux. Pour chaque morceau, il faut alors veillé à détecter des courbes dans les images suffisamment précisément pour reconstruire la QdR de manière correcte et surtout ne garder que la partie de la QdR qui convient. Pour cela, une méthode permettant, à partir des portions de contours occultants dans au moins deux vues 2D, de détecter la portion 3D de la QdR à conserver doit être développée, ce qui est loin d'être évident. Tous les morceaux doivent ensuite être assemblés en assurant la continuité en chaque jointure.

Conclusion générale

Chacune des deux parties de cette thèse comportant déjà sa propre conclusion ainsi que ses propres perspectives, il s'agit simplement dans ce dernier chapitre d'apporter une conclusion globale à cette thèse.

Nous avons donc présenté des méthodes permettant de modéliser une plante dans la première partie et un fruit dans la seconde. Plus particulièrement, les méthodes ont été développées dans le domaine de la viticulture et donc sur les vignes et les grappes de raisins. Nous avons démontré que ces méthodes pouvaient être adaptées à d'autres types de plantes. Une question qui pourrait se poser est pourquoi n'avoir pas modéliser de pieds de vigne avec des grappes de raisins dessus ? La réponse vient du fait que les échelles sont trop différentes. En effet, une photo d'un pied de vigne dans son intégralité ne permet pas de voir les grappes de raisin avec une résolution suffisamment élevée. Nous pourrions tout juste détecter l'emplacement de ces grappes et encore, lorsque celles-ci ne sont pas occultées par les feuilles et que leur couleur ou leur texture diffère suffisamment des branches ou de tout autre objet naturel de la scène. Bien sûr, un tel appareil de mesure passant dans les rangées de vignes et étant capable à la fois d'estimer des paramètres sur le feuillage comme l'homogénéité d'une parcelle ou la vigueur des pieds ainsi que des paramètres sur les grappes de raisin comme le volume total produit sur une parcelle ou la colorimétrie des grappes serait très intéressant pour le vigneron. Cependant, il semble encore compliqué aujourd'hui, avec seulement de l'optique, de pouvoir fournir de tels résultats. Cependant, les techniques évoluent de plus en plus vite. L'apparition d'appareils couplant l'optique à une information de profondeur comme celui présenté dans l'article de Izadi *et al.* [Izadi 2011] pourrait permettre d'apporter une connaissance supplémentaire 3D d'une part mais aussi de valider nos reconstructions. Les grappes pourraient alors être plus facilement détectées même sur des images de plus grande échelle grâce à leur forme géométrique particulière. Il est certain que dans le futur, les prix de tels appareils convergeront vers les prix des appareils optiques actuels et ils seront donc de plus en plus utilisés. Cependant, ces appareils ne permettront pas de voir ce qui se passe derrière l'objet photographié. Dans le cas d'une seule image comme c'était notre cas pour la modélisation d'arbres, l'utilisation de connaissances a priori pour compléter l'arbre dans les zones occultées restera le moyen le plus efficace. C'est pourquoi, coupler l'analyse avec la

synthèse devrait encore se développer dans les années à venir dans la plupart des domaines du traitement de l'image.

De plus, l'image étant utilisée de plus en plus, il serait intéressant de développer de nouvelles techniques qui permettraient une agriculture plus durable et plus écologique. En effet, la modélisation de plantes grâce à une caméra située sur des engins agricoles (comme on l'a vu dans la conclusion de la première partie) permettrait d'optimiser par exemple l'utilisation de produits vaporisés sur les plantes. Des études ont montré qu'une part importante des engrais ou des insecticides ne finissent pas sur les feuilles des plantes traitées mais dans la nature. De nouveaux appareils permettant de vaporiser la quantité adéquate aux endroits précis où se trouve le feuillage permettrait d'une part de rendre l'agriculture plus écologique et ferait faire des économies aux agriculteurs. De tels appareils pourraient également être utilisés pour la taille des vignes comme cela commence déjà à se faire avec le robot *VIN* de *Wall Ye*. Une nouvelle agriculture plus « intelligente » commence donc à se développer et de nombreux progrès restent encore à réaliser dans ce domaine.

Annexe A

Annexes

Sommaire

A.1 Utilisation des superpixels pour la segmentation

A.1.1 Problème

Lors de la phase de segmentation basée profondeur, nous rappelons que le but est alors, lorsqu'il y a des trous dans le feuillage de la première rangée de vignes, de segmenter ce feuillage du feuillage des rangées de derrière. Nous avons écrit « À chaque pixel de l'image, on lui attribue la profondeur du point suivi le plus proche ». Nous nous sommes auparavant demandé comment assigner une profondeur à chaque pixel de l'image. Nous avions d'abord pensé utiliser les superpixels. Les superpixels sont des primitives intermédiaires : elles sont plus complexes que les pixels mais moins complexes que des objets détectés par des méthodes bien souvent haut niveau et qui demandent donc un temps de calcul plus conséquent. Ils sont formés en rassemblant des pixels (généralement voisins) en utilisant des critères qui peuvent être spatiaux, colorimétriques ou en se basant sur les contours de l'image. Il était alors naturel de se demander s'il était judicieux d'attribuer la même profondeur à tout un superpixel s'il n'y avait qu'un seul point suivi à l'intérieur de celui-ci. Les tests que nous avons effectués sont présentés ici.

A.1.2 Expérimentations

Nous avons commencé par se constituer une base de vérité terrain en détourant à la main 3 images de pieds de vignes avec des conditions différentes (feuillage au soleil, à l'ombre, avec de l'herbe au sol, sans herbe). Nous pouvons voir des exemples de ce détourage manuel dans la Figure A.1, page 157.

Puis, nous avons testé la segmentation en superpixels avec les algorithmes de Mori [Mori 2005] et de Achanta *et al.* [Achanta 2010] qui font parties des méthodes les plus souvent utilisées. La méthode de Mori privilégie le critère colorimétrique aux dépens du critère spatial alors que la méthode de Achanta cherche à avoir des superpixels les plus compactes possibles. Pour chacun, nous avons essayé différents nombres de superpixels demandés par image. Les résultats sont montrés dans la Figure A.2, page 158 pour Mori et dans la Figure A.3, page 158 pour Achanta.

Puis, on a compté le nombre de superpixels ne contenant que des pixels de la classe « feuillage de la première rangée » ou ne contenant que des pixels de la classe « ne faisant pas partie du feuillage de la première rangée ». On a laissé un taux d'erreur de 5%, 10%, 15% ou 20%, c'est-à-dire qu'un superpixel est considéré comme faisant partie d'une classe s'il contient seulement 95%, 90%, 85% ou 80% de

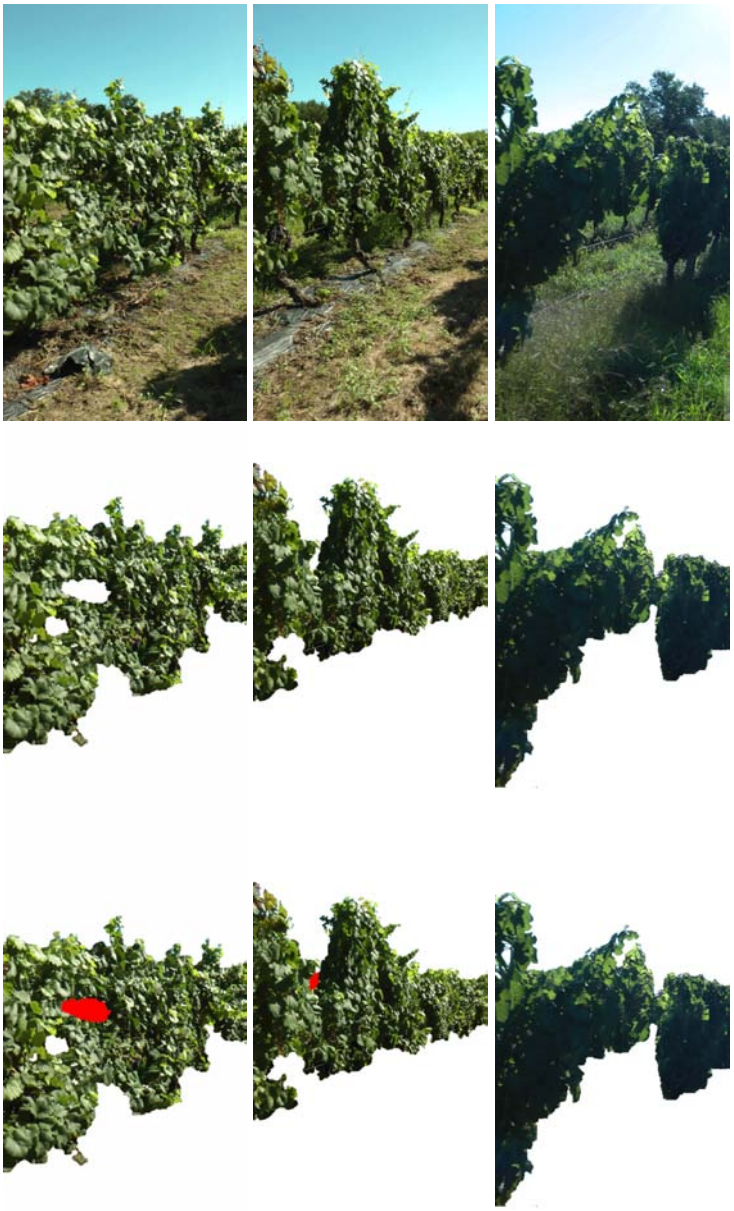

FIGURE A.1 – La première ligne représente des images extraites des vidéos acquises dans les vignes. La deuxième ligne représente seulement le feuillage des vignes de la première rangée, segmenté à la main. Les zones rouges de la troisième ligne représentent du feuillage des rangées de derrière.

pixels d'une même classe. On peut voir un exemple dans la Figure A.4, page 159 et les résultats sont référencés dans la Table A.1, page 160 et la Table A.2, page 160.

Comme on peut le remarquer, il n'y a pas de différences significatives entre les images ou entre les deux méthodes (si on compare pour un nombre de super-

FIGURE A.2 – Image de vignes sur laquelle on a appliqué l'algorithme de Mori avec 40, 206 et 1100 superpixels.

FIGURE A.3 – Image de vignes sur laquelle on a appliqué l'algorithme de Achanta avec 175, 458 et 1926 superpixels.

pixels similaires). Plus le nombre de superpixels est élevé, plus le pourcentage de superpixels n'appartenant qu'à une seule classe est important. Cependant, les superpixels sont alors de plus en plus petits et le risque d'avoir des superpixels sans points suivis à l'intérieur augmente. Par conséquent, on ne peut alors plus attribuer de profondeur à chaque superpixel. Les résultats peuvent semblaient bons car ils dépassent souvent les 90% de superpixels possédant des pixels d'une même classe mais ces pourcentages sont calculés sur la totalité des superpixels. Si on ne prend que les superpixels se trouvant sur les bordures entre les deux classes (« feuillage de la rangée de devant » et « autre »), ce taux peut très vite chuter, voire même en dessous des 50%.

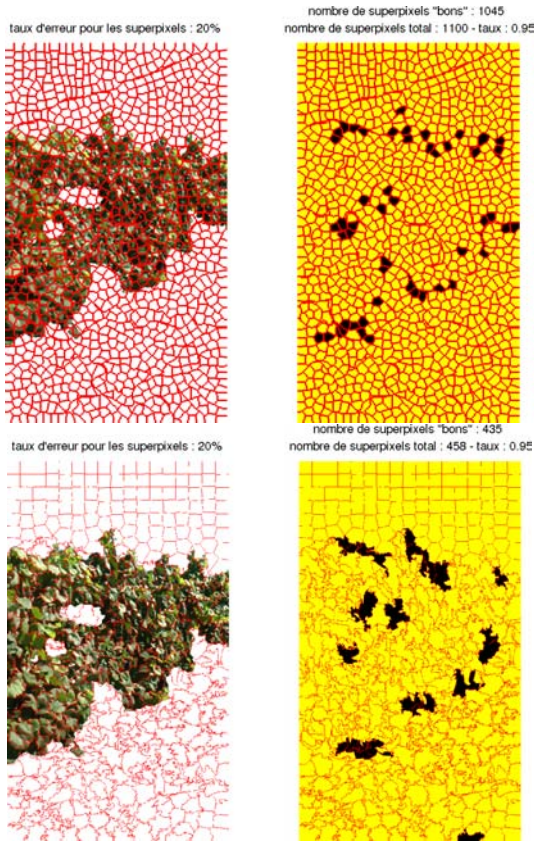

FIGURE A.4 – La première colonne montre des images segmentées manuellement (feuillage de la rangée de devant) sur lesquelles ont été surimprimées les segmentations en superpixels des images originales effectuées avec les algorithmes de Mori (en haut) et de Achanta (en bas). La deuxième colonne représente en jaune les superpixels ne contenant que des pixels d'une même classe (à 20% près) et en noir ceux ne respectant pas cette condition.

A.2 Conclusion

Les expérimentations détaillées ci-dessus nous ont permis de conclure que l'utilisation des superpixels n'étaient pas forcément justifié dans notre cas. En effet, le feuillage de la rangée de devant par rapport à celle des rangées de derrière n'est pas suffisamment discriminant pour être séparé par une méthode de segmentation par superpixels. C'est la raison pour laquelle nous avons fait le choix de ne pas l'utiliser dans notre méthode.

1ère image			
taux d'erreur	40 superpixels	206 superpixels	1100 superpixels
5%	62%	82%	90%
10%	70%	87%	93%
15%	80%	90%	94%
20%	82%	93%	95%
2ème image			
taux d'erreur	40 superpixels	212 superpixels	1108 superpixels
5%	75%	85%	91%
10%	90%	93%	94%
15%	97%	94%	95%
20%	97%	95%	96%
3ème image			
taux d'erreur	40 superpixels	219 superpixels	1119 superpixels
5%	65%	81%	91%
10%	78%	87%	93%
15%	82%	92%	95%
20%	82%	93%	95%

TABLE A.1 – Pourcentage de superpixels n'appartenant qu'à une seule classe en fonction de l'image traitée, du nombre de superpixels et du taux d'erreur pour l'algorithme de Mori.

1ère image			
taux d'erreur	175 superpixels	458 superpixels	1926 superpixels
5%	81%	87%	91%
10%	85%	91%	94%
15%	87%	93%	95%
20%	92%	95%	96%
2ème image			
taux d'erreur	181 superpixels	453 superpixels	1883 superpixels
5%	86%	90%	94%
10%	92%	92%	96%
15%	93%	93%	97%
20%	93%	95%	98%
3ème image			
taux d'erreur	183 superpixels	473 superpixels	1949 superpixels
5%	85%	91%	95%
10%	89%	95%	96%
15%	92%	97%	97%
20%	93%	97%	98%

TABLE A.2 – Pourcentage de superpixels n'appartenant qu'à une seule classe en fonction de l'image traitée, du nombre de superpixels et du taux d'erreur pour l'algorithme de Achanta.

Bibliographie

[Achanta 2010] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua et S. Susstrunk. *SLIC Superpixels*. Rapport technique 149300, Juin 2010. `Achanta2010.pdf`. (Cité en page 156.)

[Aurenhammer 1991] Franz Aurenhammer. *Voronoi diagrams – a survey of a fundamental geometric data structure*. ACM COMPUTING SURVEYS, vol. 23, no. 3, pages 345–405, 1991. (Cité en page 39.)

[Bai 2007] Xiang Bai, Longin Jan Latecki, Ieee Computer Society et Wen yu Liu. *Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution*. IEEE Trans. Pattern Anal. Mach. Intell, vol. 29, pages 449–462, 2007. (Cité en pages 27, 28 et 30.)

[Berger 1990] M. Berger. Géométrie 1 et 2. Éditions Nathan, 1990. (Cité en page 122.)

[Blum 1967] Harry Blum. *A transformation for extraction new descriptors of shape, models for the perception of speech and visual form*. Models for the Perception of Speech and Visual Form, pages 362–380, 1967. (Cité en pages 26 et 28.)

[Boudon 2006] F. Boudon, A. Meyer et C. Godin. *Survey on Computer Representations of Trees for Realistic and Efficient Rendering*. Research report, no. RR-LIRIS-2006-003, 2006. (Cité en page 4.)

[Brennecke 2004] Angela Brennecke et Tobias Isenberg. *3d shape matching using skeleton graphs*. In In Simulation and Visualization, pages 299–310, 2004. (Cité en page 69.)

[Catmull 1974] E. Catmull et R. Rom. *A class of local interpolating splines*. Computer aided geometric design, pages 317–326, 1974. (Cité en page 38.)

[Cavanagh 1999] Patrick Cavanagh. Top-down processing in vision. 1999. (Cité en page 58.)

[Chaubert-Pereira 2010] Florence Chaubert-Pereira, Yann Guédon, Christian Lavergne et Catherine Trottier. *Markov and semi-Markov switching linear mixed models used to identify forest tree growth components*. Biometrics, vol. 66, page 753, 2010. (Cité en page 6.)

[Cornea 2005] Nicu D. Cornea, Deborah Silver, Xiaosong Yuan et Raman Balasubramanian. *Computing hierarchical curve-skeletons of 3d objects.* The Visual Computer, vol. 21, 2005. (Cité en pages 28, 30 et 31.)

[Cornea 2007] Nicu D. Cornea, Deborah Silver et Patrick Min. *Curve-Skeleton Properties, Applications, and Algorithms.* IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 3, pages 530–548, Mai 2007. (Cité en pages 29 et 30.)

[Cross 1998] Geoffrey Cross et Andrew Zisserman. *Quadric Reconstruction from Dual-Space Geometry.* In ICCV, page 25, Washington, DC, USA, 1998. IEEE Computer Society. (Cité en pages 120, 122, 141 et 152.)

[de Almeida Barreto 2003] J. P. de Almeida Barreto. *General Central Projection Systems : Modeling, Calibration and Visual Servoing.* Ph.D. Thesis, University of Coimbra, 2003. (Cité en page 140.)

[de Reffye 1988] Phillippe de Reffye, Claude Edelin, Jean Françon, Marc Jaeger et Claude Puech. *Plant models faithful to botanical structure and development.* SIGGRAPH Comput. Graph., vol. 22, no. 4, pages 151–158, Juin 1988. (Cité en page 6.)

[Deussen 2005] Oliver Deussen et Bernd Lintermann. Digital Design of Nature : Computer Generated Plants and Organics. Springer-Verlag, 2005. (Cité en page 6.)

[Diener 2009] Julien Diener, Mathieu Rodriguez, Lionel Baboud et Lionel Reveret. *Wind projection basis for real-time animation of trees.* 2009. (Cité en page 10.)

[Faugeras 2001] O. Faugeras, Q.T. Luong et T. Papadopoulou. The geometry of multiple images. MIT Press, 2001. (Cité en page 8.)

[Ferri 1993] M. Ferri, F. Mangili et G. Viano. *Projective pose estimation of linear and quadratic primitives in monocular computer vision.* CVGIP, vol. 58, no. 1, pages 66–84, 1993. (Cité en pages 121, 122 et 142.)

[Goldman 1983] Ronald Goldman. *Quadrics of Revolution.* IEEE Comput. Graph. Appl., vol. 3, no. 2, pages 68–76, 1983. (Cité en page 124.)

[Golub 1983] G. H. Golub et C. F. Van Loan. Matrix computations. Johns Hopkins University Press, third edition 1996 édition, 1983. (Cité en page 107.)

[Golub 1996] Gene H. Golub et Charles F. Van Loan. Matrix computations. The Johns Hopkins University Press, 1996. (Cité en page 132.)

[Gong 2004] Rubin Gong et Gang Xu. *Quadratic Surface Reconstruction from Multiple Views using SQP*. IEICE, vol. E87-D, pages 215–223, 2004. (Cité en pages 121 et 122.)

[Gregory 1971] Richard Langton Gregory. The intelligent eye. Oxford University Press, 1971. (Cité en page 57.)

[Gurdjos 2005] Pierre Gurdjos et In so Kweon. *Geometric and Algebraic Constraints of Projected Concentric Circles and Their Applications to Camera Calibration*. PAMI, 2005. (Cité en pages 80, 117 et 146.)

[Gurdjos 2009] Pierre Gurdjos, Vincent Charvillat, Géraldine Morin et Jérôme Guénard. *Multiple View Reconstruction of a Quadric of Revolution from Its Occluding Contours*. ACCV, pages 1–12, 2009. (Cité en pages 99, 126 et 129.)

[Hartley 2000] Richard Hartley et Andrew Zisserman. Multiple view geometry in computer vision. Cambridge University Press, 2000. (Cité en page 8.)

[Hartley 2003] R. Hartley et A. Zisserman. Multiple view geometry in computer vision. Cambridge Univ. Press, 2003. (Cité en pages 1, 82, 84, 102, 108, 118, 124 et 136.)

[Hoiem 2005] Derek Hoiem, Alexei A. Efros et Martial Hebert. *Automatic photo pop-up*. ACM Trans. Graph., vol. 24, no. 3, pages 577–584, Juillet 2005. (Cité en page 81.)

[Huang 2013] Chun-Yen Huang, Wan-Ting Jheng, Wen-Kai Tai, Chin-Chen Chang et Der-Lor Way. *Procedural grape bunch modeling*. Computers and Graphics, 2013. (Cité en page 15.)

[Iqbal 2011] S. Md. Iqbal, A. Gopal et A. S. V. Sarma. *Volume estimation of apple fruits using image processing*. Image Information Processing (ICIIP), pages 1–6, 2011. (Cité en page 14.)

[Izadi 2011] Shahram Izadi, Richard A. Newcombe, David Kim, Otmar Hilliges, David Molyneaux, Steve Hodges, Pushmeet Kohli, Jamie Shotton, Andrew J. Davison et Andrew Fitzgibbon. *KinectFusion : real-time dynamic 3D surface reconstruction and interaction*. pages 23 :1–23 :1, 2011. (Cité en page 153.)

[Kumar 2009] Sanjeev Kumar, Balasubramanian Raman et Nagarajan Sukavanam. *Reconstruction of cubic curves from two or more images using geometric intersection*. IJISS, pages 98–111, 2009. (Cité en pages 121 et 122.)

[Ladegaillerie 2003] Y. Ladegaillerie. Géométrie affine, projective, euclidienne et anallagmatique. Licence, CAPES, agrégation. Ellipses, 2003. (Cité en pages 108 et 111.)

[Latecki 1999] Longin Jan Latecki et Rolf Lakamper. *Polygon Evolution by Vertex Deletion*. In Scale-Space Theories in Computer Vision. Proc. of Int. Conf. on Scale-Spaceâ99, volume LNCS 1682, Corfu, pages 398–409. Springer, 1999. (Cité en pages 27 et 33.)

[Lebon 2004] Eric Lebon, Anne Pellegrino, FranÃ§ois Tardieu et Jérémie Lecoeur. *Shoot Development in Grapevine (Vitis vinifera) is Affected by the Modular Branching Pattern of the Stem and Intraâ and Interâshoot Trophic Competition*. vol. 93, pages 263–274, 2004. (Cité en page 17.)

[Lhuillier 2005] Maxime Lhuillier et Long Quan. *A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 3, pages 418–433, Mars 2005. (Cité en page 8.)

[Li 2011] Chuan Li, Oliver Deussen, Yi-Zhe Song, Phil Willis et Peter Hall. *Modeling and generating moving trees from video*. In Proceedings of the 2011 SIGGRAPH Asia Conference, SA '11, pages 127 :1–127 :12, New York, NY, USA, 2011. ACM. (Cité en pages 10 et 11.)

[Lindenmayer 1968] Aristid Lindenmayer. *Mathematical models for cellular interaction in development : Parts I and II.* Journal of Theoretical Biology, vol. 18, 1968. (Cité en page 6.)

[Liu 2010] Jia Liu, Xiaopeng Zhang, Hongjun Li et Mingrui Dai. *Creation of tree models from freehand sketches by building 3D skeleton point cloud*. In Proceedings of the Entertainment for education, and 5th international conference on E-learning and games, Edutainment'10, pages 621–632, Berlin, Heidelberg, 2010. Springer-Verlag. (Cité en page 12.)

[Louarn 2005] GaÃ«tan Louarn. *Analyse et modélisation de l'organogenèse et de l'architecture du rameau de vigne (Vitis vinifera L.)*. Ph.D. Thesis, Ecole nationale supérieure agronomique (Montpellier), 2005. (Cité en page 17.)

[Ma 1996] S. D. Ma et L. Li. *Ellipsoid Reconstruction from Three Perspective Views*. In ICPR, page 344, Washington, DC, USA, 1996. IEEE Computer Society. (Cité en pages 120 et 122.)

[Mebatsion 2011] H.K. Mebatsion, F. Boudon, C. Godin, C. Pradal, M. Génard, C. Goz-Bac et N Bertin. *A novel profile based model for virtual repre-*

sentation of quasi-symmetric plant organs. Computers and Electronics in Agriculture, vol. 75, no. 1, pages 113–124, 2011. (Cité en pages 14 et 15.)

[Mori 2005] G. Mori. *Guiding model search using segmentation.* volume 2, pages 1417–1423, Octobre 2005. `Code,Mori2005.pdf`. (Cité en page 156.)

[Nair 2008] Vinod Nair, Josh Susskind et Geoffrey E. Hinton. *Analysis-by-Synthesis by Learning to Invert Generative Black Boxes.* ICANN, pages 971–981, 2008. (Cité en pages 59 et 61.)

[Neubert 2007] Boris Neubert, Thomas Franken et Oliver Deussen. *Approximate Image-Based Tree-Modeling using Particle Flows.* ACM Transactions on Graphics (Proc. of SIGGRAPH 2007), 2007. (Cité en pages 9 et 10.)

[Okabe 2006] Makoto Okabe, Shigeru Owada et Takeo Igarashi. *Interactive design of botanical trees using freehand sketches and example-based editing.* 2006. (Cité en pages 11, 12, 40 et 41.)

[Omid 2010] M. Omid, M. Khojastehnazhand et A. Tabatabaeefar. *Estimating volume and mass of citrus fruits by image processing technique.* Journal of Food Engineering, vol. 100, no. 2, pages 315–321, 2010. (Cité en page 14.)

[Palubicki 2009] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch et Przemyslaw Prusinkiewicz. *Self-organizing tree models for image synthesis.* SIGGRAPH, pages 1–10, 2009. (Cité en pages 6 et 7.)

[Piegl 1997] Les Piegl et Wayne Tiller. The nurbs book (2nd ed.). Springer-Verlag New York, Inc., 1997. (Cité en page 53.)

[Prusinkiewicz 1990] Przemyslaw Prusinkiewicz et Aristid Lindenmayer. The algorithmic beauty of plants. Springer Verlag, 1990. (Cité en pages 6 et 48.)

[Quan 1996] Long Quan. *Conic Reconstruction and Correspondence From Two Views.* PAMI, pages 151–160, 1996. (Cité en pages 121 et 122.)

[Quan 2006] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang et Sing Bing Kang. *Image-based plant modeling.* ACM Trans. Graph., pages 599–604, 2006. (Cité en pages 8 et 9.)

[Quan 2007] Long Quan, Jingdong Wang, Ping Tan et Lu Yuan. *Image-Based Modeling by Joint Segmentation.* IJCV, pages 135–150, 2007. (Cité en page 8.)

[Rahmann 2003] Stefan Rahmann. *Reconstruction of quadrics from two polarization views.* In IBPRIA, pages 810–820, 2003. (Cité en pages 121 et 122.)

[Reche-Martinez 2004] Alex Reche-Martinez, Ignacio Martin et George Drettakis. *Volumetric reconstruction and interactive rendering of trees from photographs.* ACM Trans. Graph., pages 720–727, 2004. (Cité en page 9.)

[Runions 2007] Adam Runions, Brendan Lane et Przemyslaw Prusinkiewicz. *Modeling Trees with a Space Colonization Algorithm.* 2007. (Cité en page 12.)

[Salmon 1882] G. Salmon. A treatise on the analytic geometry of three dimensions, volume 1 of *4th edition.* Dublin, Hodges, Figgis, & Co, 1882. (Cité en page 112.)

[Semple 1952] J.G. Semple et G.T. Kneebone. Algebraic projective geometry. Oxford University Press, 1998 édition, 1952. (Cité en pages 102 et 111.)

[Shashua 1997] Amnon Shashua et Sebastian Toelg. *The Quadric Reference Surface : Theory and Applications.* Int. J. Comput. Vision, vol. 23, no. 2, pages 185–198, 1997. (Cité en pages 121 et 122.)

[Shlyakhter 2001] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey et Seth Teller. *Reconstructing 3D Tree Models from Instrumented Photographs.* IEEE Comput. Graph. Appl., pages 53–61, 2001. (Cité en pages 7 et 8.)

[Sturm 1999] Peter F. Sturm et Stephen J. Maybank. *On plane-based camera calibration : A general algorithm, singularities, applications.* pages 432–437, 1999. (Cité en page 117.)

[Sundar 2003] H. Sundar, D. Silver, N. Gagvani et S. Dickinson. *Skeleton Based Shape Matching and Retrieval.* In Proceedings of the Shape Modeling International 2003, SMI '03, pages 130–, Washington, DC, USA, 2003. IEEE Computer Society. (Cité en page 69.)

[Talton 2011] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch et Vladlen Koltun. *Metropolis procedural modeling.* ACM Trans. Graph., vol. 30, no. 2, pages 11 :1–11 :14, Avril 2011. (Cité en pages 10 et 11.)

[Tan 2007] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang et Long Quan. *Image-based tree modeling.* ACM Trans. Graph., page 87, 2007. (Cité en pages 8 et 9.)

[Tan 2008] Ping Tan, Tian Fang, Jianxiong Xiao, Peng Zhao et Long Quan. *Single image tree modeling.* SIGGRAPH, pages 1–7, 2008. (Cité en pages 13 et 25.)

[Telea 2003] Alexandru Telea et Anna Vilanova. *A robust level-set algorithm for centerline extraction.* In Proceedings of the symposium on Data visualisation

2003, VISSYM '03, pages 185–194, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. (Cité en page 27.)

[Tomasi 1991] Carlo Tomasi et Takeo Kanade. *Detection and Tracking of Point Features*. IJCV, 1991. (Cité en page 81.)

[Triggs 1998] Bill Triggs. *Autocalibration from Planar Scenes*. In Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I, ECCV '98, pages 89–105, London, UK, UK, 1998. Springer-Verlag. (Cité en page 83.)

[Tu 2002] Zhuowen Tu et Song Chun Zhu. *Parsing Images into Region and Curve Processes*. In Proceedings of the 7th European Conference on Computer Vision-Part III, ECCV '02, pages 393–407, London, UK, UK, 2002. Springer-Verlag. (Cité en page 59.)

[Tu 2003] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille et Song-Chun Zhu. *Image Parsing : Unifying Segmentation, Detection, and Recognition*. PAMI, 2003. (Cité en pages 59 et 60.)

[Wang 2006] Rui Wang, Wei Hua, Zilong Dong, Qunsheng Peng et Hujun Bao. *Synthesizing trees by plantons*. Vis. Comput., vol. 22, no. 4, pages 238–248, Avril 2006. (Cité en page 10.)

[Weber 1995] Jason Weber et Joseph Penn. *Creation and rendering of realistic trees*. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, pages 119–128, New York, NY, USA, 1995. ACM. (Cité en page 6.)

[Wijewickrema 2006] Sudanthi N. R. Wijewickrema, Andrew P. Paplinski et Charles E. Esson. *Reconstruction of Spheres using Occluding Contours from Stereo Images*. In ICPR, pages 151–154, Washington, DC, USA, 2006. IEEE Computer Society. (Cité en pages 121 et 122.)

[Wither 2009] J. Wither, F. Boudon, M.-P. Cani et C. Godin. *Structure from silhouettes : a new paradigm for fast sketch-based design of trees*. In Computer Graphic Forum. Special issue : Eurographics 2009, volume 28 (2), page 541â550, 2009. (Cité en page 12.)

[Wong 2003] Kwan-Yee K. Wong, Paulo R. S. Mendonça et Roberto Cipolla. *Camera Calibration from Surfaces of Revolution*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 2, pages 147–161, 2003. (Cité en page 117.)

[Wu 2006] Yihong Wu, Guanghui Wang, Fuchao Wu et Zhanyi Hu. *Euclidean Reconstruction of a Circular Truncated Cone only from its Uncalibrated Contours*. IVC, pages 810–818, 2006. (Cité en pages 121 et 122.)

[Yang 2008] Xingwei Yang, Xiang Bai, Xiaojun Yang et Wenyu Liu. *An Efficient Quick Thinning Algorithm*. In Proceedings of the 2008 Congress on Image and Signal Processing, Vol. 3 - Volume 03, CISP, pages 475–478, Washington, DC, USA, 2008. IEEE Computer Society. (Cité en pages 27, 28 et 30.)

[Yuille 2006] Alan L Yuille et Daniel Kersten. *Vision as Bayesian Inference : Analysis by Synthesis ?* Trends in Cognitive Sciences In Probabilistic models of cognition, 2006. (Cité en pages 57 et 58.)

[Zeng 2006] Jiguo Zeng, Yan Zhang et Shouyi Zhan. *3D Tree Models Reconstruction from a Single Image*. ISDA, pages 445–450, 2006. (Cité en pages 13, 25 et 40.)

[Zhang 2007] Hui Zhang, Kwan-Yee K. Member-Wong et Guoqiang Zhang. *Camera Calibration from Images of Spheres*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 3, pages 499–502, 2007. (Cité en page 117.)

[Zhao 2011] L.G. Zhao et C.K. Wu. *A Planes Detection Algorithm Based on Feature Distribution*. Intelligent Networking and Collaborative Systems, International Conference on, vol. 0, pages 172–177, 2011. (Cité en page 82.)