# Reconnaissance d'objets par la génération d'hypothèses de modèles de forme appliquée à l'extraction des feuilles de plantes dans des scènes naturelles complexes

B. de Mezzo

UNIVERSITÉ MONTPELLIER II – SCIENCES ET TECHNIQUES DU LANGUEDOC

Le Laboratoire d'Informatique, de Robotique et de Micro électronique de Montpellier
et
le Cemagref

# THÈSE

pour l'obtention du grade de Docteur
au titre de l'école doctorale *Information, Structures, Systèmes*
et de la formation doctorale *Informatique*
Section CNU : 27

présentée et soutenue publiquement par
Benoit DE MEZZO

# Reconnaissance d'objets par la génération d'hypothèses de modèles de forme appliquée à l'extraction des feuilles de plantes dans des scènes naturelles complexes

le 30 novembre 2004

JURY

| | |
|---|---|
| Bruno JOUVENCEL, | Président |
| Joël QUINQUETON, | Directeur de thèse |
| Michel CHAPRON, | Examinateur |
| Gilbert GRENIER, | Examinateur |
| Monique THONNAT, | Rapporteur |
| Jean SEQUEIRA, | Rapporteur |
| Christophe FIORIO, | Encadrant LIRMM |
| Gilles RABATEL, | Encadrant Cemagref |

Je dédie ce document à Gaètan et à ses futurs possibles.

*Tu me dis, j'oublie. Tu m'enseignes, je me souviens. Tu m'impliques, j'apprends.*

B. Franklin.

*L'homme sage apprend de ses erreurs,*
*L'homme plus sage apprend des erreurs des autres.*

Confucius.

# Avant propos

Les premiers pesticides de synthèse sont apparus dans les années 40, avec un résultat positif au niveau des rendements agricoles. Vingt ans plus tard, les premières accusations d'atteinte à la santé et à l'environnement se firent entendre [VDW97]. Depuis, les organismes officiels prennent en compte les effets environnementaux des traitements pesticides et imposent de plus en plus de restrictions et d'interdictions. C'est pourquoi nous observons depuis la fin des années 70 l'apparition d'une agriculture dite intégrée, où nous nous efforçons de réduire les intrants comme les engrais ou les produits phytosanitaires (incluant les insecticides, les fongicides et les herbicides).

L'ensemble de la communauté scientifique s'accorde sur le danger et la nocivité des produits phytosanitaires, dont font partie les herbicides, non seulement sur l'homme, mais également sur l'environnement.

## Toxicité chez l'homme et pollution de l'environnement

Outre les empoisonnements, la toxicité de ces produits se manifeste par des effets très divers : cancérigènes, immunodépresseurs, mutagènes, neurotoxiques, *etc.*. Il est montré que les pesticides sont capables d'endommager le système immunitaire ou de perturber les régulations hormonales [VDW97]. Ils sont également soupçonnés d'accroître le taux de certains cancers (sein, prostate) et de réduire la fécondité masculine.

Même si la plupart des traitements sont appliqués sur les parties aériennes des plantes, une bonne part des produits atteint toujours le sol, puis les eaux de surface et les eaux souterraines par ruissellement et lessivage [Sch98]. Les pesticides peuvent alors provoquer des dégâts importants dans la faune aquatique [DLSR01], notamment chez les poissons [DS93].

Depuis que la toxicité de ces produits est prouvée, le suivi des pesticides dans l'eau s'est intensifié. En 1989, une directive européenne fixait à 0.1 $\mu$g/l la concentration maximale de tout produit phytosanitaire dans l'eau destinée à la consommation humaine. Depuis, des données sur la recherche de produits phytosanitaires dans les eaux superficielles permettent aujourd'hui de dresser un bilan de la contamination des eaux [LHRB98]. Il apparaît ainsi que la plupart des matières actives retrouvées sont des herbicides [MS98]. Parmi ces herbicides, nous retrouvons la famille des triazines en particulier, dont fait partie l'atrazine, l'herbicide le plus utilisé et le plus étudié pour sa toxicité [GSA01].

## Pratiques culturales actuelles

Depuis plus d'une cinquantaine d'années, les herbicides sont utilisés pour contrôler les espèces adventices (plus communément appelées mauvaises herbes). Pourtant, l'application de ce type de produits ne les a pas fait disparaître. En effet, l'utilisation immodérée des mêmes molécules herbicides a conduit in-

éluctablement à la sélection et à la multiplication des génotypes résistants de mauvaises herbes. De plus, cette résistance des adventices aux herbicides s'accroît d'année en année [Gas98]. Ainsi, nous recensons à présent plus de 50 pays ayant des espèces adventices résistantes, correspondant à environ 250 espèces dans le monde [Hea01]. L'apparition de ces résistances rend encore plus difficile la gestion des herbicides pour réduire les doses prescrites [RG98]. D'autre part, les pratiques culturales peuvent être à l'origine du développement de la flore adventice. Ainsi, un système de culture, sans labour et employant des herbicides à champ d'activité restreint, est très favorable à l'apparition d'adventices [JG98].

Par conséquent, il apparaît comme un besoin primordial d'améliorer les stratégies de désherbage, non seulement pour limiter l'impact sur l'environnement et respecter les lois en vigueur, mais également pour savoir utiliser les herbicides de manière plus raisonnée et efficace.

## Les solutions existantes

Les moyens qui permettent de limiter l'impact environnemental et agronomique des herbicides relèvent aujourd'hui autour de principes simples :

- **Réduction des doses d'herbicides appliqués.** Il faut noter que cette réduction entraîne une diminution des coûts de traitements, ce qui rejoint l'intérêt des agriculteurs. Plusieurs solutions existent, entre autres, l'application d'herbicides en bandes (*i.e.* uniquement sur le rang, en sarclant entre les rangs), ou la culture sans herbicide, employant uniquement des contrôles de type mécanique (sarcleur, herse-peigne, *etc.*).
  L'agriculture de précision, sous sa définition de gestion intra-parcellaire des techniques culturales, offre une autre solution, le traitement localisé, qui consiste à traiter uniquement là où les mauvaises herbes se trouvent. En effet, il est montré que les mauvaises herbes se développent en taches, dans la parcelle [BSA94]. Cela permet de ne traiter que les zones où la densité de mauvaises herbes est suffisamment importante pour justifier son traitement.
- **Une gestion plus efficace par la connaissance des espèces en présence dans la culture.** En effet, il peut exister des mauvaises herbes qui ne constituent pas un véritable danger par rapport à la culture. Il convient donc de les identifier, afin de ne se focaliser que sur celles qui sont réellement nuisibles.
  La stratégie de désherbage est importante également, puisqu'il faut savoir à partir de quel stade de croissance de la culture les adventices engendrent des pertes de rendement. Il s'agit donc de les détecter et de les identifier le plus tôt possible, c'est-à-dire aux premiers stades de croissance, lorsqu'elles ne sont pas encore compétitives vis-à-vis de la culture [Ass98]. L'étude des interactions entre la culture et les adventices permet également de mieux comprendre les mécanismes de compétition, et de déterminer la période idéale de traitement.

La décision de désherbage se doit donc d'être optimisée dans le temps sachant qu'il s'agit de détecter les mauvaises herbes le plus tôt possible, et dans l'espace, puisque l'action doit être locale.

Afin de mieux gérer les intrants, il apparaît indispensable de récolter toutes les informations possibles à partir des observations sur le terrain, ce qui permettra par la suite de faire des cartographies d'adventices. Précisons les outils actuels de détection d'adventices sur le terrain. Actuellement, deux voies sont possibles [Zwa97] :

- **l'observation visuelle lors des passages spécifiques** en inter-culture ou en culture avec un engin de type quad équipé d'un GPS (Global Positioning System). Il s'agit alors de tourner autour des zones infestées pour réaliser la cartographie. Il a été montré qu'une réduction de plus de 50 % des produits phytosanitaires était possible dans le Midwest des États-Unis [Rob00]. Cependant, cette méthode est très contraignante au niveau du temps qu'elle nécessite (entre une et deux heures par hectare) et ne peut être envisagée qu'aux premiers stades de développement de la culture.

- **la cartographie pendant la récolte.** Certaines consoles de contrôle des capteurs de rendement permettent d'enregistrer des informations visibles par le chauffeur comme la présence de plusieurs espèces d'adventices, qui sont ainsi géoréférencées. Cela nécessite d'appuyer sur un bouton de la console. D'autres systèmes exploitent le développement de logiciels de reconnaissance vocale [Lut99].

Certaines recherches sont basées sur une grille de quadrats (le "Grid sampling") localisés par GPS (Global Positioning System), où la densité d'adventices est évaluée à chaque nœud de la grille [HN99], permettant une étude des dynamiques spatiales des espèces d'adventices dans une parcelle. Cette technique n'est cependant utilisée que dans un cadre de recherche, et n'est pas envisageable pour un traitement agricole réel. Comme nous pouvons le constater, la cartographie manuelle exige un travail colossal de récolte de données, qui n'est pas envisageable dans un contexte de réalité socio-économique. Compte-tenu du contexte ci-dessus, les travaux menés dans cette thèse visent une détection automatique des adventices, afin de favoriser la collecte d'informations au sein de la parcelle.

D

# Table des matières

# 1. Introduction

*Si je n'en affirme pas davantage, c'est que je crois l'insinuation plus efficace.*
André Gide.

## Sommaire

Notre objectif est d'identifier les plantes en présence, à partir de photographies de culture et de caractériser les espèces présentes et leur quantités respectives. Notre solution repose sur l'exploitation d'images de parcelles de champs prises à hauteur d'homme avec un appareil photographique. Nous nous situons, ainsi, à l'échelle de la plante. Dans notre cas d'images de culture, nous présumons que l'identification d'une plante peut se faire à l'aide de la caractérisation de ses feuilles et de l'agencement de ces dernières au sein de la plante. C'est pourquoi, l'objet de notre recherche est l'extraction des feuilles des plantes présentes dans ces images.

## 1.1 Problématique

Nous travaillons sur des images de scènes naturelles ce qui induit un certain nombre de problèmes que nous allons expliciter brièvement. Nous observons, en effet, des différences en terme de prises de vue d'une image à l'autre, provenant aussi bien des conditions atmosphériques que de l'heure ou encore de l'angle de prise de vue. Par ailleurs, nous travaillons sur des objets organiques d'une grande variabilité de formes et pouvant avoir interagi avec leur environnement : recouvrement par une autre feuille ou par un autre objet, détérioration par un animal, *etc.*. De plus, la feuille évolue dans un espace à trois dimensions et la prise de vue projette toute cette information sur un plan en deux dimensions.

Tous ces éléments contribuent à apporter de l'ambiguïté dans l'image, augmentant la difficulté pour extraire et interpréter l'information utile. Par exemple, dans la figure 1.1 qui symbolise des silhouettes végétales extraites par segmentation couleur, le problème se pose de savoir combien de feuilles distincts doivent être considérées ? Comment les délimiter ? Les réponses semblent évidentes pour la vision humaine, car les objets que nous cherchons à extraire de l'image ont une composante commune qu'il peut être intéressant de prendre en compte dans notre démarche pour faciliter le traitement. En effet, les objets recherchés ont une certaine régularité de forme qui permet de les définir globalement. Par conséquent, afin d'aider à lever les ambiguïtés d'interprétation, nous allons exploiter une connaissance *a priori* sur les objets recherchés, connaissance qui sera basée sur cette régularité de forme.

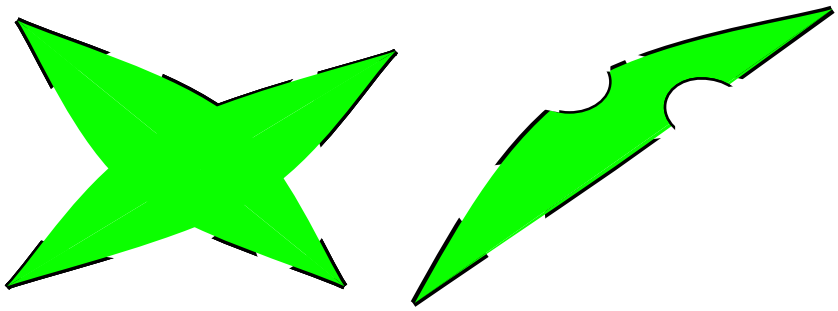


FIG. 1.1 – Caractérisation des feuilles dans une scène végétale : la partie gauche illustre les problèmes de chevauchement, la droite les problèmes liés aux déformations et détériorations.

L'introduction d'une connaissance *a priori* place notre solution dans la catégorie des méthodes descendantes (dites top-down), en opposition aux méthodes employant une analyse locale puis globale de la scène (méthodes ascendantes, dites bottom-up). Les méthodes ascendantes cherchent à extraire un schéma d'interactions entre les différentes caractéristiques d'objets trouvés localement puis, grâce à ces informations, déterminent la nature des objets présents.

Les systèmes de vison employant ce type de stratégie descendante sont nommés SPVS[1]. Dans ces

[1]Les Special Purpose Vision System sont expliqués dans [BCFM00] qui présente, aussi, diverses stratégies pour l'analyse de scènes extérieures.

systèmes tous les procédés mis en place sont pensés dans le but d'extraire un certain type d'objet (*i.e.* des feuilles dans notre cas) à partir d'une connaissance prédéfinie. Dans notre procédé, cette connaissance *a priori*, sera symbolisée par des modèles de forme à appliquer dans l'image.

En outre, comme Jain *et al.* [JZDJ98] l'ont précisé, il n'est pas envisageable de faire une recherche exhaustive de tous les positionnements possibles de nos modèles de forme dans l'image. En effet, cette recherche serait trop coûteuse à cause du nombre de paramètres à prendre en compte (position, rotation, taille, recouvrement, *etc.*) qui généreraient une infinité de possibilités. De plus, comme une forme figée de modèle ne peut pas s'adapter à tous les cas d'objets présents, nous allons utiliser des modèles déformables, nous permettant, ainsi, de prendre en compte des déformations limitées sur la forme des objets recherchés. D'autre part, les contours présents dans l'image peuvent être mis en relation de manière à définir une forme pouvant être comparée à un modèle (sur la figure 1.1 les traits noirs représentent des portions de contours extraits).

De fait, la solution mise en place pendant cette thèse repose sur une analyse globale des contours contenus dans l'image prenant en compte la connaissance *a priori*. Notre approche nécessite, donc, une information assez précise sur les contours. Cependant, les résultats des algorithmes de traitement d'image (par exemple, la segmentation) ne sont pas, dans tous les cas, ceux escomptés. Ainsi, le bruit contenu dans l'information extraite de l'image implique des contours qui peuvent être partiels et/ou incorrects.

Ainsi, considérant les remarques sur les modèles (impliquant des modèles déformables et simples) et sur les contours (pouvant être bruités), nous n'allons pas directement chercher dans l'image des applications exactes de nos modèles. Mais plutôt, développer des hypothèses sur la présence d'objets afin de prévenir des interprétations hâtives à partir d'informations pouvant être erronées. Ces hypothèses deviennent, alors, des pistes de recherche pour nos objets, et qui devront évoluer pour être confortées. Cette dernière remarque implique le besoin de renforcer chaque hypothèse par de l'information supplémentaire, afin de la rendre de plus en plus valide. De ce fait, l'hypothèse se construit de manière incrémentale, sous la contrainte d'informations issues de l'image (les contours).

Pour résumer, le but de notre analyse est de faire émerger des hypothèses sur la présence des objets recherchés en associant des morceaux de contours entre eux et des modèles déformables. De ce fait, notre approche peut aussi entrer dans la catégorie des méthodes ascendantes, nous allons alors grâce à la notion d'hypothèse établir un pont entre les approches ascendantes et descendantes. Aussi, notre approche ne relève pas des méthodes du type "template matching", bien qu'elle s'y apparente, car nous n'allons pas chercher à appliquer directement des instances exactes de nos modèles dans l'image.

La définition de nos modèles est l'un des fondements de notre méthode. Or, l'utilisation de lignes droites ou bien de tracés libres, ne permet pas de représenter correctement la forme des objets. En effet, les lignes droites ne permettent pas de décrire correctement les courbes sous-jacentes aux objets recherchés et les formes libres vont difficilement nous permettre de représenter la régularité et la variabilité de forme requise. Une autre approche possible pour décrire les objets de l'image est celle exploitant des modèles déformables paramétriques, comme ceux utilisés par A.-G. Manh [MRA02]. Cependant, dans [MRA02] les paramètres définissant les courbes sont ajustés par des fonctions de coût et des contraintes pour s'adapter à l'objet dans l'image. L'information de contour est alors utilisée seulement de manière indirecte sous la forme d'une mesure de gradient. Ce dernier participe, avec des forces de pression couleur, à annuler la progression du modèle. Mais ce procédé amène un certain nombre de limitations (initialisation du modèle, condition d'arrêt).

Dans notre cas pour une meilleure prise en compte de l'information de contours, nous avons besoin d'une représentation avec une géométrie adaptée, comprenant une définition analytique simple pour limiter la complexité des algorithmes mis en place et pour réduire les difficultés liées à la définition des modèles.

Cette contrainte de représentation implique la définition d'un modèle de forme simple pour le contour des objets recherchés et une symbolisation efficace de l'information extraite de l'image. En effet, comme nous ne cherchons pas des formes correspondant exactement à nos modèles, une définition plus "symbolique" du modèle est suffisante. Cette représentation doit aussi nous permettre de simplifier les calculs répondant aux besoins de la méthode, et surtout elle doit nous permettre de définir des algorithmes efficaces.

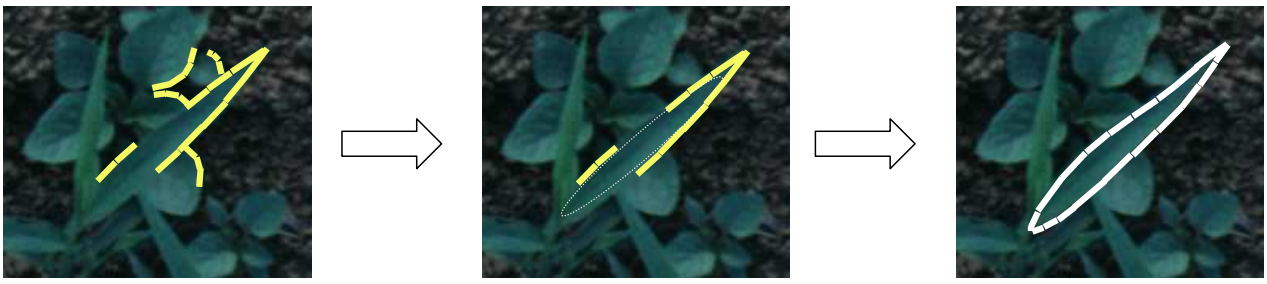
Notre choix s'est donc porté sur les courbes de Bézier de degré 2 pour les trois raisons suivantes :

- la définition d'une de ces courbes ne nécessite que trois points de contrôle et tous les points de la courbe se déterminent par un polynôme de degré 2 (cf. équation 3.1, p. 19),
- le caractère analytique de ces courbes simplifie un certain nombre de calculs,
- ces courbes sont particulièrement adaptées à la définition de formes simples et régulières.

Nous pouvons maintenant décrire la démarche dans laquelle s'inscrit la méthode développée pendant cette thèse.

## 1.2 Démarche

Dans le but de définir un procédé rapide et efficace, notre approche est définie comme une succession de méthodes pouvant être améliorées individuellement si nécessaire. Ainsi nous avons adopté la démarche illustrée par la figure 1.2.



Extraction de contours — Génération d'hypothèses — Renforcement d'hypothèses

FIG. 1.2 – Démarche adoptée.

Sur l'image numérisée, nous appliquons une méthode de segmentation qui a pour but de faire émerger des zones aux propriétés homogènes. Puis, nous extrayons les contours (frontières) des zones pour lesquelles ces propriétés homogènes correspondent aux objets recherchés et nous transformons ces contours en courbes de Bézier de degré 2, par un algorithme que nous avons défini. De la même manière, comme nous avons modélisé par des courbes de Bézier l'information extraite de l'image, nous représentons la connaissance *a priori* sur les objets recherchés, avec des modèles de forme définis selon le formalisme des courbes de Bézier. Enfin, à partir de ces courbes et des modèles de forme, nous générons un premier ensemble d'hypothèses basées sur la recherche et la corrélation de vecteurs caractéristiques. Avec cet ensemble d'hypothèses, nous appliquons notre algorithme de renforcement, basé sur la minimisation de l'erreur quadratique entre les courbes de l'hypothèse et celles du modèle. Cet algorithme exploite de l'information supplémentaire issue de l'image dans le but de conforter ces hypothèses. L'utilisation de cette méthode nous permet de minimiser l'écart entre le modèle et les courbes extraites de l'image en déterminant les meilleures valeurs pour les opérations de rotation, déplacement, échelle et déformation. Il résulte de ce procédé un certain nombre d'hypothèses qui sont triées selon un critère mis à jour tout au long du procédé. Les meilleures sont représentatives des objets présents dans l'image.

## 1.3 Description du document

Le présent document retrace les réflexions, les travaux mis en œuvre et les résultats obtenus durant cette thèse. Ces travaux ont été menés dans le but d'extraire d'une image représentant une parcelle de champs, les informations décrivant les plantes présentes. Dans un premier temps, nous proposons de nous limiter à l'extraction des feuilles de plantes et de leurs caractéristiques. Et, dans un futur proche, nous exploiterons ces informations pour reconnaître les plantes présentes dans la parcelle.

Nous allons présenter dans le chapitre 2 les différentes approches de modélisation existantes pour les modèles de formes déformables et les procédés de reconnaissance par hypothèses. Ensuite, nous présenterons les procédures que nous avons développées pour la modélisation par courbes de Bézier dans le chapitre 3, puis les méthodes impliquées dans la reconnaissance par renforcement d'hypothèses dans le chapitre 4. Nous détaillerons, dans le chapitre 5, le procédé mis en œuvre dans sa globalité et nous commenterons les résultats obtenus par ce procédé. Enfin, nous conclurons et présenterons les perspectives d'évolutions de notre procédé dans le chapitre 6.

# 2. Différentes approches de modélisation

## Sommaire

Notre but est l'extraction des objets feuilles à partir de photographies de parcelles prises à hauteur d'homme[1], afin de déterminer les espèces et les quantités des plantes présentes. Comme décrit dans notre démarche nous traitons d'abord l'image afin d'en extraire les contours pour ensuite les vectoriser. Enfin, nous appliquons notre méthode de génération et de renforcement d'hypothèses à partir des contours vectorisés et d'une connaissance *a priori* sur les formes recherchées. Notre processus est donc un enchaînement de plusieurs méthodes, mais le cœur de cette thèse réside dans la génération et le renforcement d'hypothèses sur la base de modèles de forme. Ce chapitre présente les possibilités de modélisations applicables aux objets recherchés ainsi que les méthodes de génération et renforcement d'hypothèses. Les autres fonctions qui interviennent dans la procédure font l'objet d'un simple aperçu de l'existant dans les parties dédiées (cf. section 3.2.1, annexes A et C).

L'introduction d'une connaissance *a priori* dans un système de reconnaissance de formes, implique souvent la définition de modèles représentant les objets à trouver. Plusieurs articles, comme par exemple [VH99, Lon98, AFG01, Pop94, PH91, CF01] ont déjà étudiés le domaine de l'analyse de forme. Dans [VH99] sont présentés les différents modèles, associés aux opérations pour les employer dans un ensemble d'informations à exploiter (*i.e.* mesures de similarité, méthodes d'extraction, *etc.*). [Lon98] expose, aussi, un certain nombre d'approches de modélisation et considère aussi bien les aspects locaux et globaux que les informations spatiales et de contours. Ces différents états de l'art sont complémentaires dans les méthodes et les modèles qu'ils présentent. Bien que des méthodes classiques, comme la généralisation de la transformée de Hough, soient souvent citées, chacun de ces articles présente des méthodes originales non exposées dans les autres articles.

À partir de la littérature concernant la reconnaissance de forme, il est fréquent de constater qu'un modèle de forme est associé à la méthode établissant la correspondance avec une forme quelconque. Plusieurs types de modèles peuvent être utilisés par un même type de procédures de correspondance (par exemple, procédures de régression linéaire et certaines mesures de similarité), d'un autre côté, plusieurs types de procédures de correspondance peuvent exploiter le même formalisme pour définir leur modèle (comme les modèles rigides qui sont utilisés par différentes méthodes).

Mais, comme indiqué dans la problématique, nous avons principalement besoin de définir un modèle adapté à nos besoins. Ainsi, nous allons d'abord présenter les principaux modèles, modélisant la connaissance *a priori*, en séparant modèle et méthode. Et, par type de modèles, nous ferons un aperçu des principales méthodes de correspondance.

Ensuite, du fait de la spécificité de notre méthode de correspondance incrémentale, nous présenterons un bref aperçu des méthodes basées sur la génération et le renforcement d'hypothèses.

## 2.1 Modélisation des objets recherchés

Vis à vis du problème étudié (reconnaissance de feuilles de plantes), nous ne considérerons pas les modèles représentant les caractéristiques internes de l'objet recherché, comme par exemple : l'ensemble des points particuliers définissant une empreinte digitale [VH99]. La raison est que nous ne disposons pas de suffisamment d'informations au sein de nos feuilles pour exploiter ce genre de modèles. En effet, dans les images que nous exploitons, la résolution n'est pas suffisante pour ces modèles. De plus, l'intérieur des feuilles est plutôt homogène en terme de texture et ne présente pas de détails, comme par exemple la nervure de la feuille ou bien des zones de couleurs différentes entre le bord et l'intérieur de la feuille, que nous pourrions exploiter aisément. Ce type de restrictions provient, aussi, du fait que nous exploitons des images couleurs prises dans la gamme visible du spectre. Dans l'infrarouge ou dans l'ultraviolet, nous aurions peut être eu des caractéristiques, comme l'absorbance de la chlorophylle, qui nous auraient fait choisir d'autres

[1] Images d'à peu près 70x70cm de résolution 2 pixels/mm.

catégories de modèles. Mais dans notre étude nous allons exploiter des images, d'une précédente campagne, prises dans le visible.

Nous allons, alors, nous concentrer sur les modèles caractérisant la silhouette d'un objet.

Deux points de vue sont possibles pour classifier les modèles. Nous pouvons proposer une division en deux sous-ensembles : les modèles rigides ne supportant pas de déformations, s'appliquant principalement aux objets manufacturés, et les modèles déformables pouvant plus facilement s'appliquer aux objets du vivant ayant une forte variabilité de forme. Nous pouvons, aussi, proposer quatre sous-ensembles de modèles : ceux basés sur les arêtes (points, segments, *etc.*), à forme libre et à courbes paramétriques, les invariants et les squelettes.

Les deux types de sous-ensembles cités ne sont pas disjoints. En effet, parmi les modèles basés arêtes, ceux gérant le multi-échelle (cf. [LLCS98]) peuvent supporter un certain nombre de "déformations" alors que les autres sont, en majeure partie, rigides. Il est intéressant de noter qu'il n'y a pas de modèle universel qui soit plus adapté qu'un autre dans tous les cas ou même à une catégorie de problèmes. Chaque problème a des contraintes spécifiques qui vont demander la définition d'un modèle spécifique. De ce fait, un modèle pourra être apparenté à plusieurs des catégories précitées.

Nous observons aussi que dans la majorité des cas, les modèles définissent des formes en deux dimensions. Néanmoins, il existe aussi des modèles à trois dimensions rigides (par exemple [BL99]) et, moins fréquemment, des modèles à trois dimensions déformables. Nous pouvons trouver des exemples utilisant ces modèles : Heap et Hogg [HH96] ont réussi à segmenter une main avec des mouvements de doigts sur des images issues d'une vidéo. Aussi, [BCA96] définit une méthode pour le suivi du ventricule gauche avec un modèle 3D déformable. Mais la complexité et les temps de calculs liés à l'application de modèles 3D déformables pour extraire des objets d'une image, explique la faible utilisation de ces modèles [HH96]. De plus, cette complexité implique souvent, de manière à simplifier le traitement, d'avoir des fonds d'images simples à segmenter (comme un fond uni).

Dans le cadre de notre approche et du support choisi (des photographies de parcelles), nous ne considérerons pas les modèles à trois dimensions. Cela peut se justifier à cause de la grande variabilité de formes des feuilles, de l'ambiguïté des scènes en extérieur et du nombre de possibilités, induit par ces faits, de correspondance entre un modèle 3D et un ensemble de contours 2D issus d'une photographie.

C'est pourquoi, nous allons étudier le choix d'un type de modèle en deux dimensions nous permettant de représenter la régularité de formes que nous pouvons observer d'un objet à l'autre d'une même feuille, nous permettant également un certain nombre de déformations limitées correspondant à la variabilité des formes de l'objet au sein de son espèce. Cette notion de régularité n'est pas sans rappeler celle d'un "invariant" que nous chercherions à représenter dans nos modèles.

### 2.1.1 Modèles basés arêtes

Les modèles les plus simples peuvent être définis comme un ensemble de points, ou de lignes, issues d'un seul exemplaire d'objet ou bien, à partir d'une base d'exemples autorisant un apprentissage, ou encore, définis par une image binaire. Ils représentent un objet réduit à l'expression de sa silhouette pouvant aussi comporter quelques caractéristiques internes au contour (cf. figure 2.1). Par exemple, [GLGZ03] utilise aussi bien la forme d'un simple rectangle pour représenter un sachet de thé que des images binaires pour représenter le gabarit d'une main ou d'une guitare.

Toutefois, un des avantages de ces modèles basés arêtes, est la simplicité de la définition du modèle : une suite de points ou de segments. Un autre avantage est la facilité de la mise en œuvre des opérations affines à appliquer au modèle et des relations, mesures et opérations géométriques applicables entre élé-

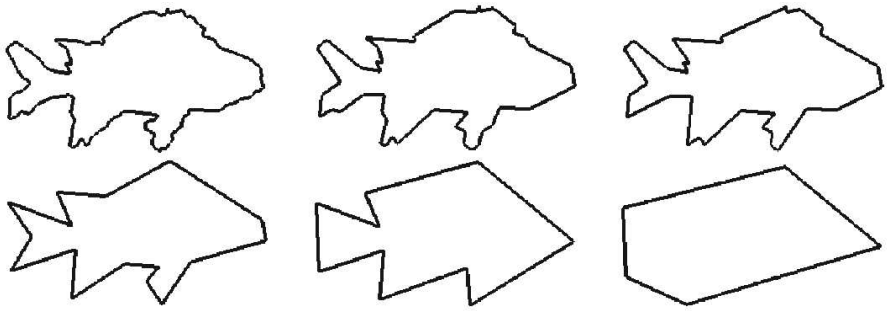


FIG. 2.1 – Exemple de plusieurs représentation d'un modèle de forme utilisant un ensemble de points.

ments. De plus, l'extraction d'informations (contours, gradient, *etc.*), utilisées pour placer le modèle au sein de l'image, peut se faire avec des procédures classiques et simples à réaliser, comme par exemple, des filtres de convolution, du seuillage, dérivée, laplacien, *etc.*. Néanmoins, ces modèles supportent peu de déformations, principalement les transformations affines nécessaires à leur positionnement dans l'image (déplacement, rotation, échelle, *etc.*). Outre ces déformations globales à l'objet, l'introduction de déformations locales comme, par exemple, l'articulation d'une partie représentant un bras ou une jambe, devrait impliquer l'ajout d'informations supplémentaires, comme un squelette avec ses articulations. Mais la littérature que nous avons étudiée n'en fait pas mention. De plus, il est assez difficile de définir une forme ayant des contours courbes mais réguliers par ce moyen. En effet, pour maintenir la régularité dans une suite de points il faut définir des procédures permettant de conserver l'aspect du contour, ou de la partie de contour, à travers les différentes transformations affines. L'application de ces contraintes nous rapproche des modèles déformables qui sont présentés au paragraphe 2.1.4. Aussi, les opérations de correspondance avec des objets dans l'image ne sont pas forcément des plus efficaces en temps de calcul et en espace mémoire. En effet, lorsque le modèle est utilisé tel quel, nous trouvons fréquemment des solutions basées sur la transformée de Hough qui nécessite beaucoup de ressources [GM96]. Néanmoins, ce type de modèle permet d'être utilisé avec des contours partiels d'objets lors de l'utilisation de cette transformée. D'autres solutions calculent un invariant sur le modèle (cf. paragraphe 2.1.2) et nécessitent l'utilisation d'une mesure de similarité pour établir une correspondance entre objet et modèle, mais ces méthodes vont difficilement prendre en charge les contours partiels.

### 2.1.2 Modèles à base d'invariants calculés

D'autres représentations vont chercher à modéliser la forme d'un objet par des invariants calculés à partir d'un exemple ou d'une base d'exemples. Plus précisément, ce type de procédé va chercher à extraire une "signature", spécifique à l'objet, à partir de l'information de contour. Avec ce type de modèle il est fréquent de devoir définir une métrique particulière permettant d'établir une mesure de similarité entre l'invariant du modèle et celui de l'objet trouvé (cf. [RW95]). Il est difficile de tous les citer car ce type de modèle est très courant [DTJT96, RW95, GS00, OST99, VS96] et chaque solution est souvent spécifique à un problème. Néanmoins, certains invariants sont utilisés fréquemment, comme les descripteurs basés sur une décomposition de Fourier ou bien une décomposition en ondelettes [LLCS98]. Ou encore, nous pouvons trouver des modèles basés sur une description fonction d'une composée affine (rotation, mesure de Hauffdorf, *etc.*) comme définie en [Bis01, GLGZ03]. Enfin, d'autres invariants se définissent par la recherche de la caractéristique commune à une base d'exemples et disjointe des caractéristiques des autres objets (cf. [KP02] et [BMP02]). La figure 2.2 illustre un de ces invariants.

Les invariants peuvent être séparés en deux catégories : locaux et globaux [RW95]. Dans le cas des invariants globaux, pour les déterminer et appliquer la mesure de similarité, il faut pouvoir extraire un contour complet de l'objet car la mesure de similarité compare la forme globale et donc nécessite le

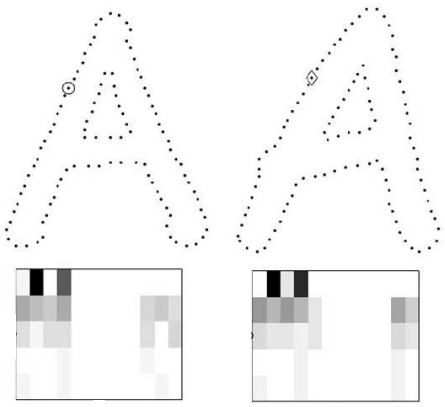


FIG. 2.2 – Exemple d'invariants calculés par la méthode de [BMP02]. La rangée du haut montre deux suites de points représentant la forme de deux 'A'. La rangée du bas représente leur invariant respectif (histogramme de niveaux de gris).

contour global de l'objet extrait. Les solutions permettant d'extraire les objets se chevauchant vont, donc, être difficiles à mettre en place. Néanmoins, avec les invariants locaux [KP02] des solutions peuvent être trouvées car la caractéristique recherchée ne correspond qu'à une partie du contour, mais encore faut-il que cette portion de contour caractéristique soit correctement extraite de l'image pour identifier l'objet. Il existe, aussi, la possibilité de définir des modèles par le biais de grammaires reprenant dans leur base lexicale les concepts de segment, courbe à gauche, courbe à droite, *etc.* (cf. [Maî] qui fait un aperçu des différents modèles et des possibilités offertes). Ces grammaires peuvent aussi permettre d'introduire des notions de relations topologiques entre les différents éléments. Mais il s'agit là de représentations n'exploitant que le contour complet de l'objet recherché.

Certains descripteurs, comme ceux basés sur des décompositions en ondelettes, sont invariants aux translations affines de l'objet sur lequel ils sont calculés. Néanmoins, d'autres méthodes peuvent présenter des limitations pour calculer l'invariant, par exemple [BMP02]), la définition du point de départ sur le contour extrait doit se situer au même endroit que celui du modèle, sinon l'invariant ne sera pas le même. De ce fait, il peut être nécessaire de définir des opérations particulières pour corréler deux invariants.

Il nous sera donc difficile d'utiliser ces modèles du fait que nous prévoyons de ne pouvoir extraire que des contours partiels à partir de notre image et que les invariants locaux nécessitent l'extraction d'une partie caractéristique que nous ne sommes pas sûr de pouvoir extraire dans tous les cas.

### 2.1.3 Modèles basés squelette

Une autre définition de modèle, qui peut être apparentée à un type particulier d'invariant, passe par la recherche des axes de symétrie dans une forme [RM93] ou par la recherche d'un squelette invariant par, par exemple, des procédés de morphologie mathématique [DTJT96], [ZY96], [Lon98]. La figure 2.3 illustre le procédé mis en place par [RM93].

Néanmoins, ce type d'approche n'est pas particulièrement robuste car pas toujours bijective : des formes différentes peuvent avoir le même squelette et *A contrario*, deux formes à peu près équivalentes, mais ayant subi quelques déformations mineures, peuvent avoir des squelettes différents [Lon98]. Ces erreurs peuvent survenir lors de la génération du squelette, étape qui simplifie la structure en supprimant certains axes considérés comme superflus. En effet, cette sélection peut entraîner la suppression d'axes importants mal représentés et donc induire un squelette incorrect. De plus, comme précédemment, ce type de modèle ne supporte que les contours complets extraits de l'image, et de ce fait, les objets se chevauchant vont être

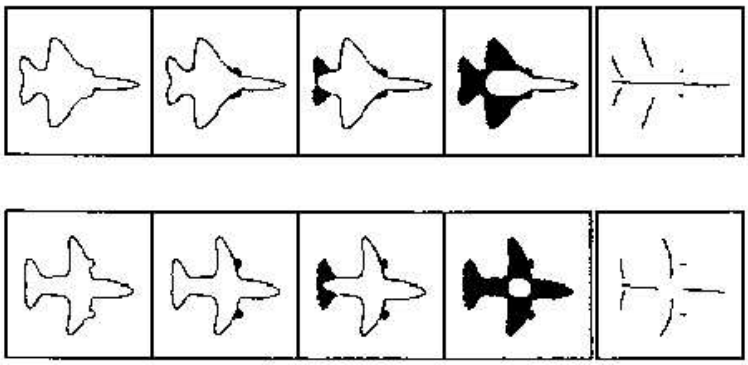


FIG. 2.3 – Exemple de décomposition en axes principaux selon la méthode de [RM93].

difficiles, voire impossibles, à détecter car ils ne vont pas générer des squelettes adéquats (cf. [RM93]). Il nous faudra aussi définir une mesure de correspondance adaptée qui puisse prendre en compte les différences de représentation pour un même modèle. Ce modèle va donc être difficilement utilisable pour répondre à nos besoins du fait du contour global qu'il nécessite et des difficultés de correspondance entre un modèle et un objet.

### 2.1.4 Modèles paramétriques et à forme libre

L'autre famille rencontrée est celle des modèles déformables qui peuvent être, par exemple, des snakes (contours actifs), des splines ou bien encore des courbes paramétriques, comme les courbes de Bézier. Ils peuvent être définis par un ensemble de points dont la cohérence est maintenue par des liens entre ces points – par exemple, une fonction gérant une tolérance sur la distance et l'angle entre deux points consécutifs –, ou par les B-splines ou des courbes représentées par des fonctions analytiques. Ces modèles de formes, très bien décrits dans [MRA02], ont déjà fait l'objet de plusieurs états de l'art, nous pouvons par exemple citer [JZDJ98]. La figure 2.4 illustre une application de ce type de modèles.

FIG. 2.4 – Exemple d'application d'un modèle déformable. Nous pouvons observer que la position d'initialisation (figure de gauche) est proche de la solution finale désirée (figure de droite (extrait de [XLT02]).

Ils ont l'avantage d'être malléables (au sens de la propagation des snakes [PD02]) et de supporter un grand nombre de déformations (au sens des transformations affines et articulations [BCGJ95, XLT02]). Ils sont généralement appliqués directement dans l'image et non pas dans un espace issu d'une transformée de l'image. Afin qu'ils s'appliquent convenablement à l'objet dans l'image et d'éviter qu'ils ne se déforment trop, ils sont généralement associés à une fonction de coût et à des contraintes. Ce type de fonction exploite soit des informations issues de l'image (couleurs, gradients, *etc.*) soit des contraintes liant les points de définition du modèle. Du fait de leur grande malléabilité, ils peuvent nécessiter, pour converger vers une

solution, une fonction d'arrêt qui peut être difficile à mettre en œuvre (cf. [LL93]) à cause de la présence de minima locaux, ne correspondant pas à la solution. Certains de ces modèles comme les snakes ou les B-splines peuvent, aussi, nécessiter une initialisation du modèle proche de la solution finale [LL93] à cause, ici encore, de la présence de minima locaux pouvant entraîner une divergence du modèle.

Ce type de solution est assez couramment utilisé dans le cadre de l'imagerie médicale car les formes des objets recherchés, par exemple un viscère en coupe à segmenter à partir d'une radiographie, s'y prêtent bien [MDSA00, DT02]. Nous pouvons aussi citer le domaine de la reconnaissance de l'écriture où ce type d'approche est utilisé car il peut prendre en compte les différences de formes qu'il est possible d'observer entre deux silhouettes du même caractère [CY98b, DTJT96]. L'utilisation des modèles à forme libre est plus délicate lorsque nous ne disposons que des portions de contours, car il faut pouvoir définir des contraintes sur le modèle afin d'éviter des déformations locales non désirées là où le contour est inexistant ou incorrectement défini. Les approches développées sont, du fait de la définition du modèle, souvent itératives (le modèle croît dans l'image à chaque itération) et peuvent, alors, demander un nombre conséquent de calculs pour converger.

Nous pouvons citer A.-G. Manh [MRA02] qui a exploité ces modèles pour représenter les feuilles de mauvaises herbes en utilisant des B-splines et des courbes paramétriques. Son approche s'avérait efficace pour les cas de feuilles isolées ou de chevauchement sans équivoques (par exemple, le croisement de deux feuilles) mais lorsque la feuille présentait des altérations (trous, déformations, *etc.*) le modèle ne se plaçait pas correctement. De plus, à cause de la difficulté de définir une fonction d'arrêt fonctionnant dans tous les cas, la procédure pouvait échouer lorsque, par exemple, deux feuilles se trouvaient dans le prolongement l'une de l'autre. Néanmoins, son approche présente un taux de réussite voisin de 80% dans les cas de configurations quasi idéales. En outre, l'approche reste trop lente et n'exploite pas réellement l'information de contours, ce qui pourrait, comme elle le suggère dans son document, améliorer ses performances.

Il est aussi intéressant de noter que nous n'avons pas trouvé dans la littérature de modèles exploitant explicitement des courbes de Bézier pour représenter une forme rigide ou déformable.

## 2.2 Reconnaissance par renforcement d'hypothèses

Notre approche a besoin d'un modèle pouvant représenter une forme déformable, dans une certaine limite, ayant des contours simples et réguliers. Nous devons, aussi, pouvoir établir avec ce modèle une mesure de similarité avec un contour d'objet incomplet. Les différents modèles étudiés présentent chacun des limitations ou des points qui ne conviennent pas forcément à nos contraintes. Les modèles rigides sont trop peu déformables et les méthodes de correspondance trop coûteuses, les modèles à forme libre peuvent, de leur côté, poser problèmes de part les contraintes à y imposer pour faire croître un modèle sous bon contrôle. Les modèles d'invariant ainsi que leur mesure de correspondance ne permettent pas facilement de définir la régularité requise, et aussi, ces modèles sont peu applicables à des images contenant des contours partiels. Enfin, les modèles définissant des squelettes manquent de robustesse en regard des déformations auxquelles nos objets peuvent être sujets et ils nécessitent un contour global que nous ne pouvons pas extraire de l'image. Ainsi, après avoir étudier les principes de ces approches, nous avons choisi un nouveau modèle adapté à nos besoins, entre les modèles de formes paramétrique et les modèles de formes définis par des segments de droites.

Nous avons choisi une représentation qui exploite les avantages du formalisme des courbes paramétriques et ceux des éléments 'simple' de géométrie. Cette représentation est celle des courbes de Bézier de degré 2 qui offrent des avantages analytiques et géométriques (cf. chapitre 3.1). Elle va nous permettre de définir des modèles de feuilles prenant en compte régularité et déformation. Nous l'utiliserons pour représenter la connaissance *a priori* (*i.e.* les modèles) et les contours extraits de l'image.

Les approches présentées dans la section précédente se caractérisent par l'aspect "direct" du résultat obtenu par ses méthodes. En effet, les procédés appliqués aux modèles et aux informations issues de l'image sont, généralement, calculés en une passe. Plus précisément, il n'y a pas de raffinement possible du résultat et de la décision prise concernant la similarité de l'objet avec le modèle. Ce type d'approche nécessite d'avoir une mesure de similarité quasi infaillible et des données peu bruitées, pour obtenir de bons résultats en une passe. De cette observation et du fait que nous allons principalement manipuler un ensemble de portions de contours extraits de l'image sujets à diverses erreurs, nous allons présenter un procédé progressif, *i.e.* incrémental, de reconnaissance de forme à partir d'un modèle.

L'utilisation d'hypothèses dans un procédé d'identification, ou de reconnaissance, permet d'éviter toute recherche exhaustive de similarité[2] de plusieurs sous-ensembles d'éléments, par exemple des contours, des arêtes, *etc.*, dans un ensemble d'informations issues d'une image. En effet, une recherche exhaustive peut induire une forte complexité (cf. [AF86] et [Low87]), nous pouvons exprimer cette complexité combinatoire par $\mathcal{O}(n,m) = k_n.\mathcal{C}_m^n$, avec $n$ la taille d'un sous-ensemble, $E$ l'ensemble complet de taille $m$, avec $n \ll m$ et $k_n$ la complexité de la mesure de similarité.

De ce fait, il peut être intéressant de définir un processus incrémental basé sur la recherche d'une caractéristique commune au modèle et à un sous-ensemble d'éléments (*i.e.* sous-ensemble $G$ de taille $l$, avec $l < n$). Ce processus permet, dans un premier temps, de positionner et d'ajuster une hypothèse basée sur le modèle qui a été déterminé en fonction de la caractéristique obtenue à partir du sous-ensemble $G$. Et ce processus permet, dans un deuxième temps, de rechercher incrémentalement dans l'ensemble $E$, des éléments dans la proximité d'une hypothèse (*i.e.* $h$) susceptible de lui être ajoutés. Cette deuxième étape est définie dans le but d'infirmer $h$ ou bien, au contraire, de raffiner ses propriétés (*i.e.* son orientation, échelle, *etc.*) en lui injectant de l'information supplémentaire.

Ce type de méthode, utilisée fréquemment dans divers domaines pour extraire des objets à partir d'une image et d'un modèle, permet la prédiction de présence d'un *objet*, en fonction d'un modèle, dans un ensemble d'informations sans ordre perceptif. Plus particulièrement dans le domaine de la vision

[2] Similarité définie selon des critères et des caractéristiques spécifiques à l'objet recherché.

numérique, nous pouvons citer [AF86] et [Low87] dont les méthodes ont été citées maintes fois, par exemple, par [Kel99, RB96, PL93, KDN93, Bas92, BL99, Low91, CHTH93]. Nous pouvons décrire la procédure généralement employée en trois étapes principales : l'extraction de contours ou d'informations exploitables dans la caractérisation des objets présents dans la scène, puis la recherche de motifs caractéristiques à partir des informations extraites, ensuite la génération et le placement d'hypothèses à partir des informations issues des motifs caractéristiques et enfin, le renforcement ou validation des hypothèses par l'établissement de la similarité entre le modèle sous-jacent à ces hypothèses et les informations extraites de l'image. Aussi, Lam *et al.* [LVW97] présentent une méthode similaire utilisant le raffinement d'hypothèses pour la prédiction de points de suivi d'un objet dont le mouvement est connu. [Kel99] met en place une théorie pour la reconnaissance d'objet basée sur la recherche de motifs caractéristiques et sur les hypothèses, et il l'applique dans le cas des images SAR pour la reconnaissance de cible (véhicules, *etc.*). [BL99] applique ce type de méthode pour la recherche et la définition d'index (*i.e.* motifs caractéristiques) pour émettre, par index, un certain nombre d'hypothèses sur le type des objets rencontrés et leur position, orientation, *etc.*. Sa méthode est appliquée à la reconnaissance des objets manufacturés, comme les objets qu'il est possible de trouver sur un bureau, et à la commande du bras d'un robot pour la reconnaissance et manipulation d'objets (en l'occurrence des parallélépipèdes) filmés par une caméra.

Ce procédé, que nous détaillons dans la section 4, est celui que nous avons utilisé dans notre méthode en collaboration avec des modèles déformables paramétriques basés sur Bézier pour la reconnaissance des feuilles oblongues présentes dans une image de terrain.

# 3. Modélisation par courbes de Bézier

## Sommaire

Les courbes de Bézier, et plus particulièrement celles de degré 2, représentent l'information de base autour de laquelle sont construits nos procédés d'extraction d'objets. Cette représentation est aussi bien utilisée pour représenter l'information de contours qui est extraite de l'image que dans la modélisation de la connaissance *a priori*.

Nous allons exposer les principales propriétés des courbes de Bézier de degré 2 qui nous ont intéressés dans notre étude (cf. section 3.1) puis nous allons présenter les méthodes développées nous permettant d'exploiter un certain nombre de facilités (cf. section 3.2) issues des caractéristique de ces courbes. Et enfin, nous détaillerons la structure de nos modèles de forme (cf. section 3.3).

## 3.1 Courbes de Bézier de degré 2

Une courbe de Bézier se définit selon trois points de contrôle ($A$, $B$, et $C$) et tout point $M_t$ de cette courbe vérifie un polynôme du second degré dont les coefficients sont ceux de Bernstein.

$$M_t = (1-t)^2 A + 2t(1-t)B + t^2 C \tag{3.1}$$

Avec $t \in [0, 1]$.

Ainsi, si nous pouvons vectoriser un contour discret par une suite de $k$ courbes de Bézier de degré 2, alors, nous sommes capables de compresser l'information utile en passant de $n$ coordonnées (où $n$ correspond au nombre de points utilisés pour définir la courbe discrète) à $3.k$ avec $k \ll n$.

L'algorithme de *de Casteljau* [dC93] permet de tracer une courbe de Bézier de manière géométrique. Ainsi, pour déterminer un point $M$ de la courbe à la position $t$, il considère les segments $[AB]$ et $[BC]$ pour calculer les points $M_t^1$ et $M_t^2$ tel que $[AM_t^1] = t.[AB]$ et $[BM_t^2] = t.[BC]$. Il obtient, ensuite, le point $M_t$ par $[M_t^1 M_t] = t.[M_t^1 M_t^2]$. Ce procédé est illustré par la figure 3.1.

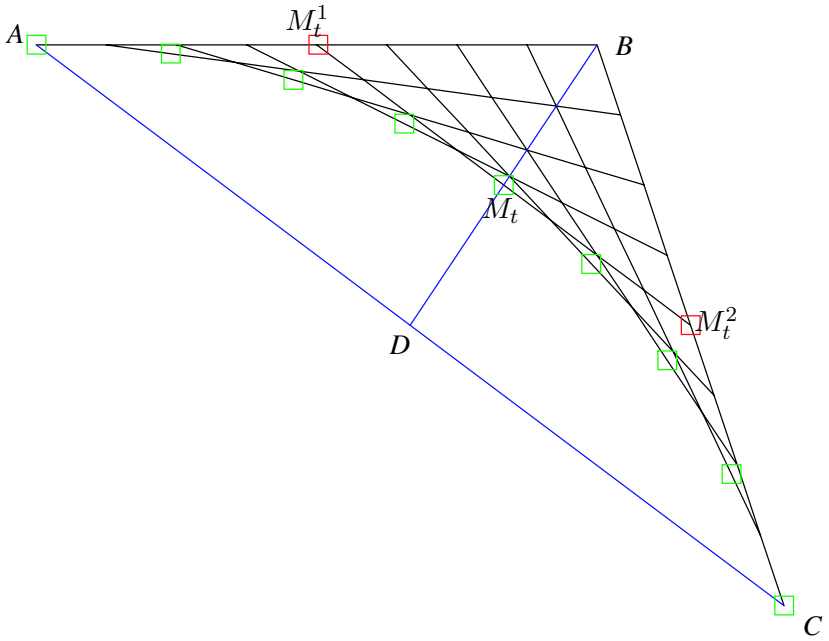



FIG. 3.1 – Création d'une courbe de Bézier par l'algorithme de *de Casteljau* pour $t$ défini par pas de $1/8$ème.

Les courbes de Bézier de degré 2 présentent, par ailleurs, des propriétés intéressantes au niveau des possibilités de calculs. Ainsi, le projeté d'un point quelconque à la perpendiculaire sur une courbe de degré 2 se calcule en résolvant un polynôme du troisième degré, lequel peut être résolu de manière analytique (cf. section 3.1.3). Ce calcul du projeté sera souvent utilisé dans les différentes étapes du processus. De même, la longueur d'une courbe de degré 2 se calcule de manière analytique comme expliqué à la section 3.1.2.

De plus, il est intéressant de remarquer que toute portion d'une courbe de Bézier est une courbe de Bézier. Le cas particulier où les portions de courbes sont choisies aux extrémités de la courbe est représentée sur la figure 3.8 page 29. Sur cette figure la courbe d'origine est définie par $A_1$, $B_3$ et $C_2$ ($bez_3$) et les portions de courbes sont définies par $A_1$, $B_1$ et $C_1$ ($bez_1$) et $A_2$, $B_2$ et $C_2$ ($bez_2$). La tangente en $A_1$ est commune à $bez_1$ et $bez_3$ et que celle en $C_2$ et commune à $bez_2$ et $bez_3$.

Cette propriété va nous permettre de regrouper des courbes de Bézier entre elles, par exemple, dans le cas où elles appartiendraient à une même courbe dans l'image qui a été coupée ou bien partiellement occultée (cf. section 3.2.2).

### 3.1.1 Mesure de colinéarité

Nous allons définir une notion dite de colinéarité entre deux courbes de Bézier de degré 2. Cette colinéarité nous la définissons comme une mesure issue de l'ensemble des mesures de colinéarité faites deux à deux à partir de tangentes, extraites de ces courbes, considérées comme similaires. Ce qui reviendrait à mesurer une sorte de colinéarité moyenne. Aussi, la mesure de colinéarité entre les droites définies par les extrémités des courbes (droite $\Delta_{AC}$ et $\Delta_{A'C'}$) n'est pas valable car elle ne prend pas en compte la courbure des courbes (cf. figure 3.2).

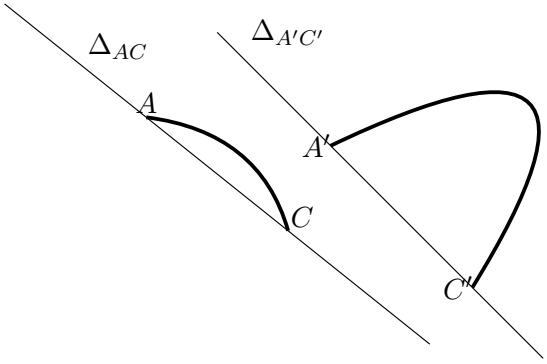



FIG. 3.2 – Colinéarité par les droites $\Delta_{AC}$ $\Delta_{A'C'}$

Néanmoins, pour pallier à ce problème, nous allons chercher à comparer les courbures. La comparaison exhaustive de toutes les tangentes est difficile à mettre en œuvre de manière efficace, et il est, aussi, inefficace de faire ce calcul pour un nombre prédéterminé de tangentes. Il est aussi difficile de choisir les tangentes à apparier afin de les comparer. Nous allons, donc, exploiter le fait qu'une courbe de Bézier de degré 2 se définit par trois points de contrôle (cf. section 3.1) pour simplifier le problème et, par conséquent, mesurer l'angle entre les tangentes aux extrémités $A$ $C$ et les droites $\Delta_{AC}$ (cf. angles $\alpha_1$, $\alpha_2$ et $\alpha_3$ sur la figure 3.3). L'angle $\alpha_3$ nous fournit l'information de similarité vis-à-vis de la transformation en rotation et la moyenne, compensée par $\alpha_3$, des angles $\alpha_1$ et $\alpha_2$, nous permet de déterminer une mesure de colinéarité globale. Ainsi, une valeur de rotation n'excédant pas les 20°(valeur empirique, permet de compenser les défauts de positionnement dû au cumul d'erreurs) et une moyenne se situant autour de zéro, définit notre mesure de colinéarité (cf. figure 3.3).

Nous pouvons définir la règle floue (cf. annexe D), avec les comparateurs flous $fRot = FuzzComp(20.0, 5.0)$

et $fMoyNulle = FuzzComp(3.0, 3.0)$, *i.e.* nous définissons un seuil de rotation de valeur 20 avec une tolérance de 5 et un seuil de valeur 3 pour représenter une moyenne nulle avec une tolérance de 3. Cette règle s'exprime par :

$$\mathcal{C}_1 = (fRot > \alpha_3) \underset{\text{prob}}{\wedge} \left( fMoyNulle > \left| \frac{(\alpha_1 - \alpha_3) + (\alpha_2 + \alpha_3)}{2} \right| \right)$$

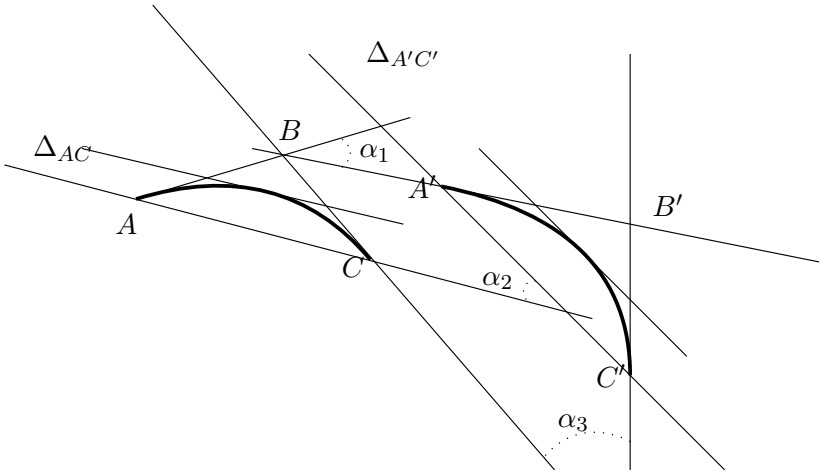



FIG. 3.3 – Colinéarité par les droites $\Delta_{AC}$ $\Delta_{A'C'}$ et les tangentes aux extrémités

### 3.1.2 Longueur d'une courbe de Bézier de degré 2

Par hypothèse, un point $M_t$ de la courbe de Bézier est donné par l'équation 3.1. Soit $V_t$, tel que :

$$\begin{aligned} V_t &= \frac{dM_t}{dt} \\ &= 2\,(tA + (1-2t)B - (1-t)C) \\ &= 2\,((A + C - 2B)t + B - A) \end{aligned} \tag{3.2}$$

Pour déterminer la longueur de la courbe $ds$, nous voulons intégrer

$$ds = \int |V_t|.dt = \int \sqrt{V_t^2}.dt$$

Nous obtenons :

$$\begin{aligned} V_t^2 &= 8\,\|A + C - 2\,B\|^2 t^2 + 8\,(A + C - 2\,B).(B - C)t + \|B - C\|^2 \\ &= at^2 + bt + c \end{aligned} \tag{3.3}$$

Pour intégrer $ds$ (d'après [Cho96]), nous posons :

$$\begin{aligned}\int \sqrt{V_t^2}.dt &= \int \sqrt{at^2+bt+c}.dt \\ &= a\int \frac{t^2.dt}{\sqrt{at^2+bt+c}} + b\int \frac{t.dt}{\sqrt{at^2+bt+c}} + c\int \frac{dt}{\sqrt{at^2+bt+c}} \\ &= aI_2 + bI_1 + cI_0 \end{aligned} \tag{3.4}$$

Avec $z_t = t\sqrt{a} + \dfrac{b}{2\sqrt{a}}$ et $k = c - \dfrac{b^2}{4\,a}$, nous calculons $I_0$ :

$$\begin{aligned} I_0 &= \int_0^1 \frac{dt}{\sqrt{at^2+bt+c}} \\ &= \int_{z_0}^{z_1} \frac{dz}{\sqrt{z^2+k}} \end{aligned} \tag{3.5}$$

Sachant que $L_t = \displaystyle\int \frac{dz_t}{\sqrt{z_t^2+k}} = \ln z_t + \sqrt{z_t^2+k}$, nous obtenons $I_0 = L_t - L_0$.

Avec les équations de récurrence suivantes nous obtenons $I_1$ puis $I_2$ et calculons la longueur de la courbe avec l'équation 3.4.

$$2\,aI_1 + bI_0 = 2\sqrt{at^2+bt+c} \tag{3.6}$$

$$4\,aI_2 + 3\,bI_1 + 2\,cI_0 = 2\,t\sqrt{at^2+bt+c} \tag{3.7}$$

### 3.1.3 Projection d'un point sur une courbe de Bézier

La méthode mise en place a pour but de déterminer la position, sur une courbe de Bézier de degré 2, de la projection perpendiculaire d'un point quelconque. Nous déterminons, aussi, par cette méthode, la distance la plus courte d'un point à une courbe.

L'équation 3.1 permet de calculer la position d'un point d'une courbe de Bézier en fonction d'une valeur $t \in [0,1]$ (en 0 et 1, nous obtenons les points d'extrémité de la courbe). Il nous est possible de calculer la position de points ne se situant pas entre $A$ et $C$, en choisissant des valeurs de $t \notin [0,1]$. De ce fait, un point dans l'espace 2D aura toujours, au moins, un projeté sur une courbe de Bézier, même si ce point est hors bornes.

Soit $X$ un point quelconque et $M_t$ le point à la position $t$ sur la courbe. Nous cherchons à minimiser la distance séparant $X$ de $M_t$ en fonction de $t$. D'après l'équation 3.1 :

$$\overrightarrow{XM} = t^2\overrightarrow{XA} + 2\,t(1-t)\overrightarrow{XB} + (1-t)^2\overrightarrow{XC}$$

En développant nous obtenons :

$$\begin{aligned}\|\overrightarrow{XM}\|^2 &= t^4(\overrightarrow{XA} + \overrightarrow{XC} - 2\,\overrightarrow{XB})^2 + 4\,t^3\big((\overrightarrow{XB} - \overrightarrow{XC}).(\overrightarrow{XA} + \overrightarrow{XC} - 2\,\overrightarrow{XB})\big) \\ &\quad + t^2\big(4\,(\overrightarrow{XB} - \overrightarrow{XC}).(\overrightarrow{XB} - 2\,\overrightarrow{XC}) + 2\,\overrightarrow{XC}.(\overrightarrow{XA} - \overrightarrow{XC})\big) \\ &\quad + 4\,t\big(\overrightarrow{XC}.(\overrightarrow{XB} - \overrightarrow{XC})\big) + \overrightarrow{XC}^2 \end{aligned} \tag{3.8}$$

En posant $\overrightarrow{K} = \dfrac{\overrightarrow{XA} + \overrightarrow{XC}}{2} - \overrightarrow{XB}$, nous obtenons :

$$\begin{aligned} \|\overrightarrow{XM}\|^2 &= 4\,t^4\overrightarrow{K}^2 + 8\,t^3(\overrightarrow{K}.\overrightarrow{CB}) + 4\,t^2(\overrightarrow{CB}^2 + \overrightarrow{K}.\overrightarrow{XC}) \\ &\quad +4\,t(\overrightarrow{XC}.\overrightarrow{CB}) + \overrightarrow{XC}^2 \end{aligned} \tag{3.9}$$

Comme nous cherchons à minimiser la distance séparant $X$ de $M_t$, nous dérivons l'équation 3.9 en fonction de $t$ :

$$\frac{d\|\overrightarrow{XM}\|^2}{dt} = 16\,t^3\overrightarrow{K}^2 + 24\,t^2(\overrightarrow{K}.\overrightarrow{CB}) + 8\,t(\overrightarrow{CB}^2 + \overrightarrow{K}.\overrightarrow{XC}) + 4\,(\overrightarrow{XC}.\overrightarrow{CB}) \tag{3.10}$$

Nous obtenons alors une équation du troisième degré que nous résolvons en utilisant la méthode décrite par [LFA71] pour obtenir les extrema de distance. Les solutions obtenues sont comprises entre $-\infty$ et $+\infty$ mais doivent être entre $0$ et $1$ pour avoir une solution sur la courbe. Sur l'ensemble des solutions obtenues (jusqu'à trois en théorie), nous retenons alors celle donnant le $\|XM\|^2$ le plus petit. Il reste, alors, à déterminer si cet extremum est un minimum par comparaison à $\|XA\|^2$ et $\|XC\|^2$. Si la solution n'est pas sur la courbe, deux approches sont proposées : soit nous retournons une erreur ou soit nous retournons l'extrémité la plus proche.

Autre remarque :

$$\begin{aligned} \frac{d\|\overrightarrow{XM}\|^2}{dt} &= 2.\overrightarrow{XM}.\frac{d\overrightarrow{XM}}{dt} \\ &= 2.\overrightarrow{XM}.\overrightarrow{V_t} \\ &= 0 \end{aligned}$$

$\overrightarrow{XM}$ est bien orthogonal à la tangente à la courbe en $M_t$.

## 3.2 Principes des algorithmes développés

Notre procédé exploite des courbes de Bézier pour représenter l'information qu'il extrait à partir des contours discrets des objets de l'image. Un contour discret est au départ décrit par un ensemble de points discrets contiguës. Nous allons, donc, chercher à exprimer ces suites de points discrets en des suites de courbes de Bézier de degré 2. Puis, pour pallier les erreurs de segmentation et les altérations que les objets de l'image ont pu subir, nous allons chercher à regrouper les courbes de Bézier entre elles lorsqu'elles sont compatibles et dans le prolongement l'une de l'autre.

Dans cette section du document nous appellerons un contour discret, une partie de frontière séparant deux régions et contenant une suite de points contigus ordonnés selon un des sens de la courbe. La méthode d'identification va, ainsi, s'appliquer à tous les contours discrets extraits de l'image.

### 3.2.1 Identification de courbes de Bézier

La littérature abonde d'articles retraçant des méthodes transformant des suites de points en des suites de lignes (approximation par polygones), ou de segments et d'arcs de cercle, ou de splines. Nous pouvons, par exemple, citer [KG99], [SSS03], [RR92], [TC89], [CC99], [CY98a], [ZC95] et [Ram87]. Chan et Yan [CY98a] exploitent des courbes de Bézier de degré 3, ajustées sur des portions de contours permettant leur placement, les paramètres des courbes sont déterminés *via* une méthode de minimisation. Certaines approches, parmi celles citées, cherchent une résolution sur la base d'une résolution analytique du problème [KG99], tandis que d'autres travaillent par divisions récursives pour définir la meilleur approximation polygonale ou encore recherchent des sommets (*i.e.* "corners") pour définir des zones sur lesquelles ajuster des splines ou d'autres courbes. [MS04] introduit une procédure intéressante d'identification polygonale au vu des bons résultats obtenus et du comparatif qu'il fournit. Sa méthode non paramétrée permet, de manière simple, de déterminer les points dominants d'un contour en se basant sur le calcul d'un score par point et sur l'élection des meilleurs points vis à vis de ce score.

Notre approche va être principalement algorithmique, *i.e.* nous allons, comme [CY98a], diviser un contour selon des motifs caractéristiques et ensuite, ajuster des courbes de Bézier de degré 2.

Dans le cas général, plusieurs courbes contiguës devront être identifiées pour parvenir à représenter un contour discret dans son ensemble (cf. figure 3.4). Par exemple, un point d'inflexion sur un contour n'est pas représentable par une courbe de Bézier de degré 2 ce qui signifie qu'une courbe devra être définie des deux côtés de ce point.

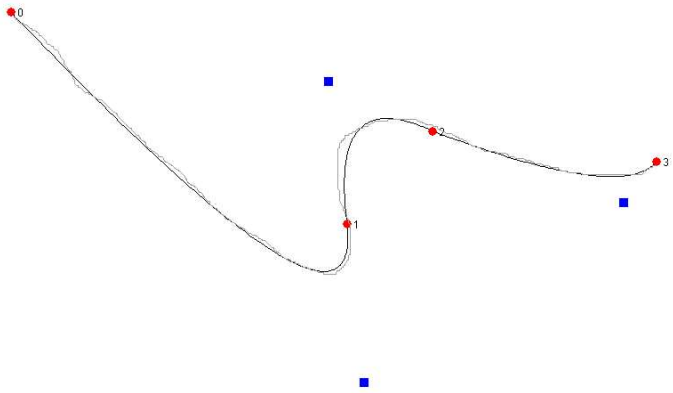



FIG. 3.4 – Un contour discret peut nécessiter plusieurs courbes de Bézier pour être identifié.

La première étape de l'identification d'une courbe de Bézier sur un contour discret, est la recherche

des motifs caractéristiques définissant les extrémités de notre courbe (points $A = M_0$ et $C = M_1$, cf. équation 3.1). Principalement, les points d'inflexion, les points de forte courbure et les extrémités de zones de courbure constante vont définir nos motifs caractéristiques et permettre la définition de ces balises d'extrémités. Cette recherche de balises est guidée par un automate associé au parcours du contour (cf. section 3.2.1.1).

La deuxième étape consiste à identifier le troisième point (point $B$) par une première estimation issue de l'automate, puis par une méthode géométrique complémentaire (cf. section 3.2.1.2).

#### 3.2.1.1 Automates d'identification

La recherche des points d'extrémité et du troisième point de chacune des courbes, est contrôlée par un automate (illustré en figure 3.5 et figure 3.6). Cet automate exploite l'information d'angle absolu en chaque point ($\alpha_i$, où $i$ représente la position du point $P_i$ considéré dans le contour discret) du contour discret précédemment lissé. Pour déterminer l'angle absolu en un point discret nous calculons l'angle entre l'horizontale et le vecteur $\overrightarrow{P_iP_{i+r}}$, où $r$ est la taille de la région de support[1] (paramètre utilisateur). Le lissage est effectué selon la méthode de Shen-Castan [SC92]. L'automate exploite aussi les dérivées première et seconde de cette information d'angle, pour déterminer les segments de courbure constante et les passages à zéro.

Il est à noter que tous les passages à zéro n'ont pas la même importance (l'état n°2 est un état transitoire) et il en va de même pour les segments de courbure constante (certains permettront la génération de plusieurs courbes alors que d'autres seront ignorés). Nous utiliserons l'appellation d'*arc* pour les segments avec une courbure constante.

Pour permettre la localisation des balises nous avons défini les conditions de détection suivantes pour labelliser les points du contour discret par les états de l'automate :

**Une pointe** est un point pour lequel la variation de courbure est proche de zéro et la courbure est supérieure à un seuil.

$$\mathcal{C}_2 = (|\ddot{\alpha}_i| < lim) \wedge (|\dot{\alpha}_i| > courbureMin)$$

où $lim \approx 0$ et $\wedge$ correspond au ET logique.

**Un arc** est un segment de courbure faible, de variation de courbure proche de zéro et de longueur significative.

$$\mathcal{C}_3 = (|\ddot{\alpha}_i| < lim) \wedge (|\dot{\alpha}_i| < courbureMin) \wedge (longueurArc > arcMin)$$

**Un point d'inflexion** est un point où le sens de courbure est différent du point précédent. De plus, il ne doit pas s'agir d'un arc fluctuant autour de zéro.

$$\mathcal{C}_4 = \left(\dot{\alpha}_i.\dot{\alpha}_{i-1} < 0\right) \wedge \left(\left|\frac{\sum_{j=i-n}^{i} \ddot{\alpha}_j}{n}\right| > nonZero\right)$$

où, par exemple, $nonZero = 1.0$.

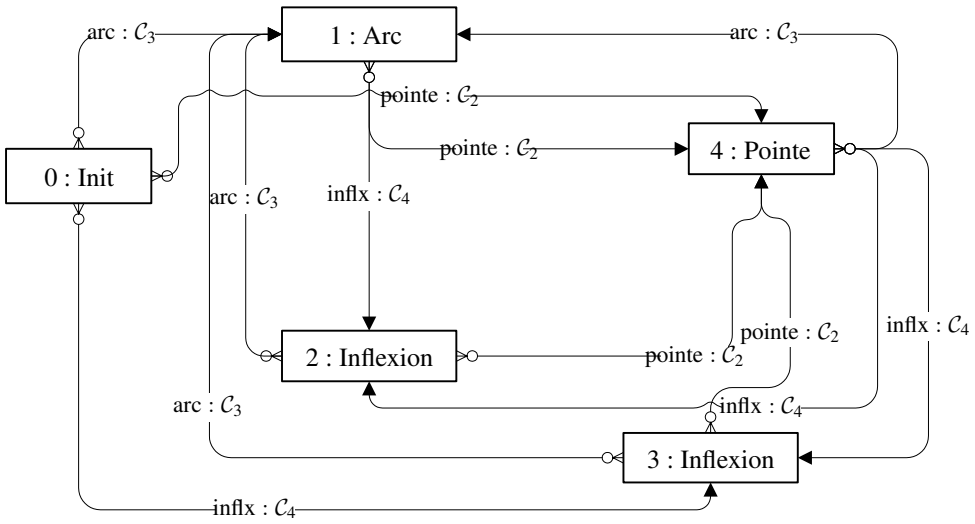



FIG. 3.5 – Automate de recherche des points de contrôle des courbes de Bézier avec les conditions de transition.

Les trois types d'*objets* recherchés par l'automate (*i.e.* Arc, Pointe et Inflexion) ne sont pas associés à un point précis du contour discret mais peuvent être définis sur un ensemble de points contigus. De ce fait, chaque *objet* a une position de démarrage (par exemple, $startAngle$) et une position de fin (par exemple, $endLev0$).

Sur la figure 3.6, nous notons '`Mil` : i' (milieu en $i$) pour l'action qui va générer un point $M_{0.5}$ (cf. équation 3.1, p. 19) à la position $i$ sur le contour discret et nous notons '`Ext` : t' (extrémité en $i$) l'action qui va générer un point $A$ ou $C$ en $i$. La position $i$ peut être le milieu d'un *objet* (par exemple, '`Mil` : (endAngle-startAngle)/2') ou bien une extrémité (par exemple, '`Ext` : StartLev0').

En sortie de l'automate, nous nous trouvons dans l'un des quatre états de l'automate (*i.e.* 0 : Init, 1 : Arc, 4 : Pointe, 2 : Inflexion, 3 : Inflexion). Si nous nous situons encore à l'état d'initialisation, nous relançons l'algorithme en abaissant nos valeurs de seuil jusqu'à réussir. Ou bien jusqu'à obtenir des valeurs de seuil qui ne soient plus utilisables et dans ce cas l'algorithme retourne une erreur signifiant que le contour discret n'est pas assimilable à une courbe de Bézier. Dans les cas où nous nous situons dans un autre état que l'initialisation, nous générons les courbes de Bézier permettant de combler l'espace entre l'extrémité de dernière courbe générée et l'extrémité du contour discret.

#### 3.2.1.2 Estimation du point $B$

Pour créer une courbe, une recherche du troisième point (point $B$) est effectuée en se basant sur l'estimation faite, par l'automate précédent, de la position du point $M_{0.5}$ (cf. équation 3.1, p. 19). Nous considérons deux cas : le cas où la courbure est suffisamment faible pour définir une courbe plate[2] et le cas où ce point $B$ est à calculer.

Dans le cas plat, le troisième point $B^d$ (l'exposant $d$ correspond à la notation pour la courbe discrète) est simplement placé au milieu des deux points d'extrémité.

[1] La région de support est la zone définie par la longueur des segments pouvant se trouver à gauche et à droite d'un point considéré et permettant de calculer l'angle en ce point [MS04].

[2] $\mathcal{C}_5 = \widehat{ABC} < angArc$ , où $angArc$ est fixée à une valeur prédéfinie.

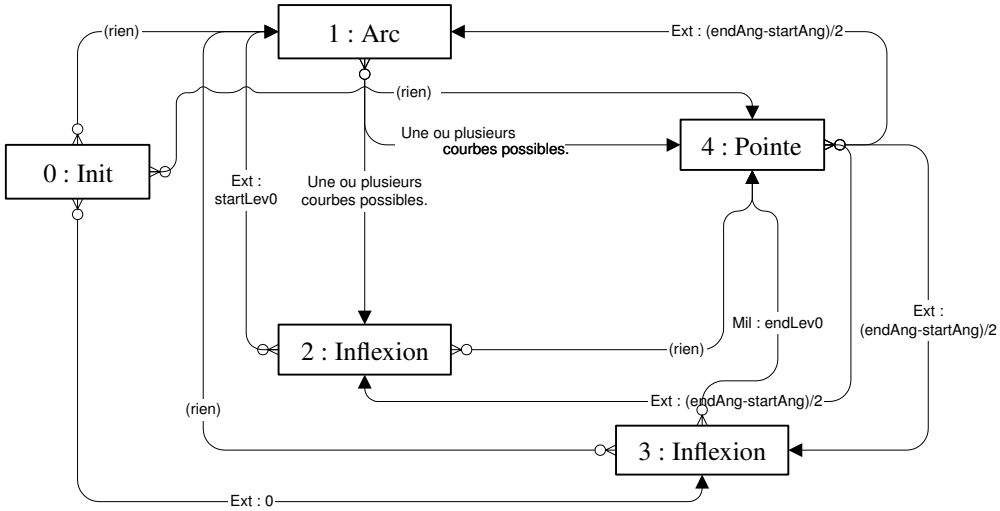



FIG. 3.6 – Actions générées. L'information portée par les flèches représente non pas ce qui enclenche la transition mais l'action qui est engendrée par cette transition.

Dans l'autre cas (cf. figure 3.7) nous allons avoir besoin de calculer trois points $M^d_{0.25}$, $M^d_{0.5}$ et $M^d_{0.75}$, supposés être les points $M_x$ sur la courbe discrète. Nous supposons par construction que $M_{0.5}$ est le point le plus éloigné de la droite $(AC)$ sur une courbe et qu'il se trouve à mi-distance de $B$ et de $D$. Nous utilisons cette méthode pour calculer $M^d_{0.5}$. Néanmoins, ce calcul est coûteux en terme de temps car il s'agit de calculer le projeté orthogonal de chaque point de la courbe discrète sur $(AC)$ puis de calculer la distance entre chacun de ces points discrets et leur projeté, afin d'élire le point discret le plus éloigné de $(AC)$. En toute rigueur, ce calcul devrait être répété, une fois déterminé $M^d_{0.5}$, pour trouver $M^d_{0.25}$ (resp. $M^d_{0.75}$) avec la droite $(AM^d_{0.5})$ (resp. $(M^d_{0.5}C)$). Mais, dans un souci de gain de temps, nous déterminons la position approchée de $M^d_{0.25}$ en le situant au milieu de $A$ et de $M^d_{0.5}$ et plaçons $M^d_{0.75}$ au milieu de $M^d_{0.5}$ et $C$. Cette approximation peut se justifier par le fait que plus nous extrayons une petite portion d'une courbe de Bézier plus cette portion se rapproche d'une droite. Les points de contrôle de cette portion se définissent toujours par les extrémités et par l'intersection des tangentes aux points d'extrémité, le point $B$ devient de plus en plus équidistant à $A$ et $C$.

Ainsi, nous déterminons le troisième point $B^d$ avec $[DB^d] = 2\,[DM^d_{0.5}]$.

Puis, nous appliquons un vecteur de correction à ce point $B^d$ de la courbe de Bézier intermédiaire définie par les points $A$, $B^d$, $C$. Nous utiliserons l'exposant $i$ pour la notation pour la courbe intermédiaire. Ce vecteur de correction est issu de la somme des vecteurs entre les points à $0.25$ de la courbe de Bézier générée ($M^i_{0.25}$) vers celui de la courbe discrète ($M^d_{0.25}$), et ceux à $0.75$ ($M^i_{0.75}$ et $M^d_{0.75}$). Cette somme doit être divisée par deux pour appliquer la moyenne des deux corrections au point $M_{0.5}$. Mais nous appliquons cette correction (calculé pour le point $M_{0.5}$) au point $B$, ainsi, nous multiplions cette moyenne par deux, car $[DB] = 2.[DM_{0.5}]$.

Nous pouvons résumer cette approche, avec $\overrightarrow{V_X}$ la notation pour le vecteur en $X$, par :

$$\overrightarrow{V}_B = 2 \times \overrightarrow{V}_{0.5} = 2 \times \frac{\overrightarrow{V}_{0.25} + \overrightarrow{V}_{0.75}}{2} = \overrightarrow{V}_{0.25} + \overrightarrow{V}_{0.75}$$

Les informations concernant l'erreur relative (écart) entre la courbe générée et la courbe discrète sont associées à la nouvelle courbe. Le calcul se fait en sommant la distance séparant chaque point du contour discret avec sont projeté orthogonal sur la courbe calculée (cf. section 3.1.3). Soit :

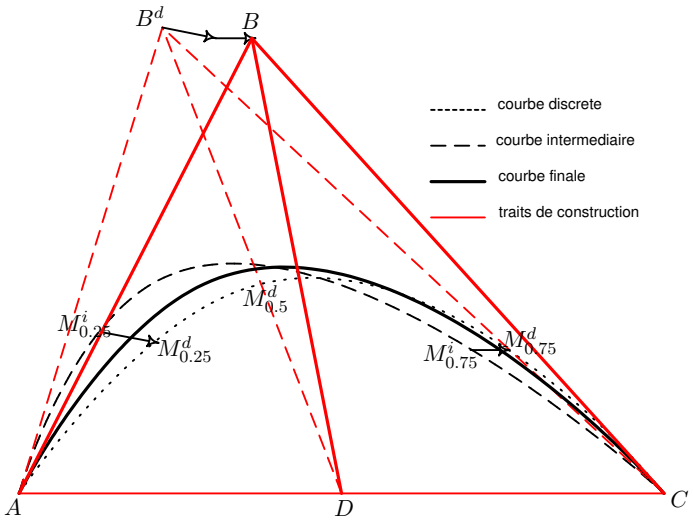



FIG. 3.7 – Procédé de correction par vecteurs.

$$E = \sum_{p \in Cd} ||p - \perp^p_{bez}||^2$$

où $Cd$ est le contour discret et $bez$ la courbe générée.

Il est aussi à noter que les courbes de Bézier générées sont orientées. C'est-à-dire qu'elles partent du point $A$ pour aller au point $C$. Grâce à cette information nous pouvons définir une gauche et une droite à une courbe de Bézier en 'regardant' une courbe de $A$ vers $C$. Cela nous permettra de stocker une information fort utile dans les sections suivantes : celle de la position de l'objet recherché par rapport à l'orientation de la courbe ($bez.objectSide = OBJ\_AT\_LEFT$ ou $bez.objectSide = OBJ\_AT\_RIGHT$).

### 3.2.2 Appariement de courbes de 2-Bézier

L'appariement de deux courbes de Bézier en une seule englobant les deux premières, a pour but la réduction du nombre de données de base à manipuler ainsi que la suppression des trous et/ou artefacts. Cet appariement ne se conçoit qu'entre des courbes placées dans le prolongement l'une de l'autre. Il est à noter que cette notion de prolongement (deux courbes qui peuvent faire partie d'une même troisième) est donc fortement liée à la courbure des courbes à apparier. Par exemple, il est possible d'apparier deux courbes si elles sont quasiment plates (cf. la note de bas de page $\mathcal{C}_5$, *p. 26*) et dans la continuité l'une de l'autre, mais il est aussi possible d'apparier des courbes à forte courbure qui ne sont pas forcement face à face.

La méthode d'appariement prend en compte des aspects géométriques pour le cas général (théorique) et aussi des aspects de déformations, d'approximation et de vecteurs de correction dans notre cas où les courbes (comportant des informations d'écarts quadratiques entre le contour discret et la courbe) sont issues de l'étape d'identification.

#### 3.2.2.1 Observations géométriques

Ainsi, deux courbes ($bez_1$ et $bez_2$) de Bézier de degré 2 peuvent être assemblées en une seule ($bez_3$) si tout point de $bez_1$ et $bez_2$ appartient à $bez_3$ ; alors le troisième point ($B_3$) se situe à l'intersection des

tangentes en $A_1$ et $C_2$. La courbe $bez_3$ est donc définie par les points $A_1$, $B_3$ et $C_2$. Nous noterons : $bez_3 = app(bez_1, bez2)$.

Note : $M^3_{0.5}$ est le point $M_{0.5}$ (cf. équation 3.1, p. 19) pour $bez_3$ ; pour des raisons de lisibilité, nous allons le noter $M$.

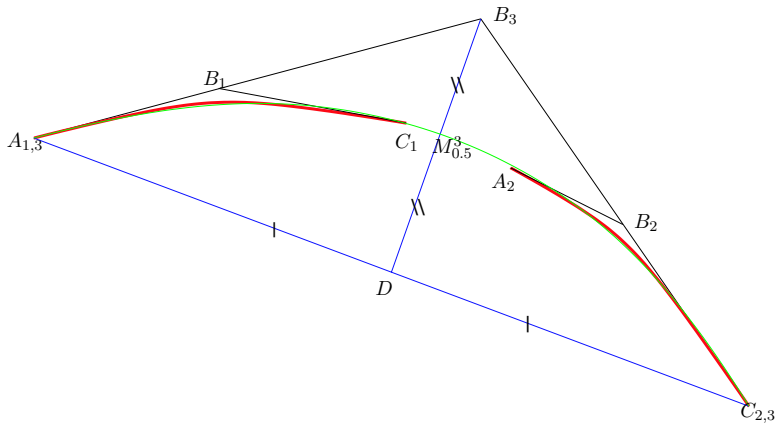



FIG. 3.8 – Appariement de deux courbes de Bézier.

En plus de l'intersection des tangentes en $B_3$, nous avons deux contraintes à respecter. La première est relative à l'intersection des tangentes en $C_1$ et $A_2$ : la projection de ce point sur la droite $(MB_3)$ doit se situer sur le segment $[MB_3]$.

$$\mathcal{C}_6 = \perp^{(B_1C_1)\cap(A_2B_2)}_{(MB_3)} \subset [MB_3]$$

La seconde contrainte est géométrique : il s'agit de la courbure. En effet, nous n'allons pas apparier deux courbes qui ont un sens de courbure différent ou qui ne sont pas dans la continuité l'une de l'autre.

Ces observations et contraintes définissent l'approche de la méthode géométrique mise en place.

#### 3.2.2.2 Méthode géométrique

De ces observations géométriques et de ces contraintes nous extrayons deux grandes lignes de conduite : la première permet de réduire le calcul dans le cas où les courbes $bez_1$ et $bez_2$ sont quasiment plates et la deuxième permet de réaliser l'appariement dans le cas général.

Pour cela nous commençons par mesurer 6 angles (cf. figure 3.9) dans cet ensemble de deux courbes, pour, ensuite, prendre les décisions qui s'imposent.

Où $\alpha_1$ et $\alpha_2$ sont les angles intra-courbes, *i.e.* ils mesurent la courbure ; $\alpha_3$ est l'angle inter-courbes selon les tangentes intérieures ; $\alpha_4$ est l'angle inter-courbes selon les droites $(A_1C_1)$ et $(A_2C_2)$ ; $\alpha_5$ et $\alpha_6$ sont les angle représentant l'écart d'alignement entre les courbes, *i.e.* $\alpha_5 = \widehat{B_1C_1A2}$.

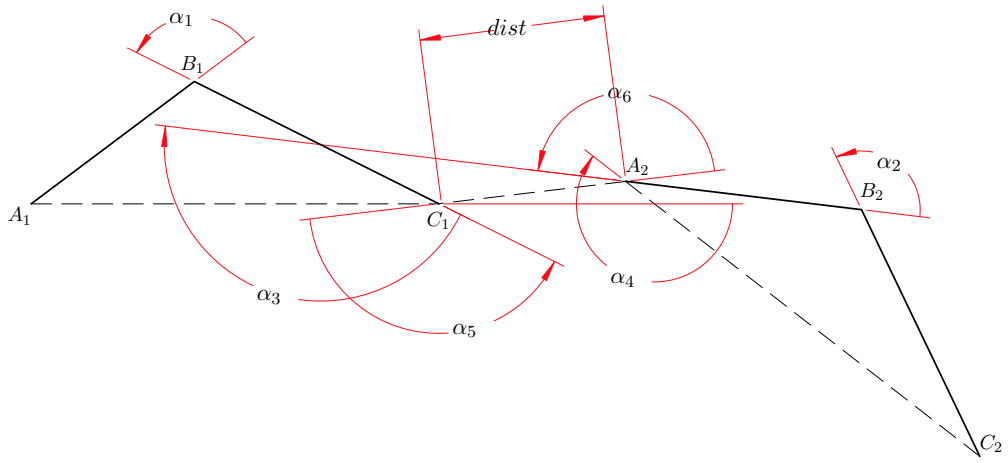



FIG. 3.9 – Angles pris en compte lors de l'appariement de deux courbes de Bézier.

***Courbes plates***

Pour le cas des courbes plates nous utilisons principalement les angles $\alpha_1$, $\alpha_2$ (angles intra-courbes), $\alpha_4$ (pour vérifier l'alignement) et $\alpha_5$ (pour vérifier l'écart d'alignement).

$$\mathcal{C}_7 = (|\pi - \alpha_1| < intraAng) \wedge (|\pi - \alpha_2| < intraAng) \wedge (|\pi - \alpha_4| < interAng) \wedge \mathfrak{C}_7^1$$

où

$$\mathfrak{C}_7^1 = (|\pi - \alpha_5| < interAng) \vee (dist < maxDist)$$

avec $intraAng$, $interAng$ et $maxDist$ proche de 0 et $\vee$ correspond au OU logique.

Ainsi, si $\mathcal{C}_7$ est vérifiée, la courbe $bez_3$ est générée avec pour points de contrôle $A_1$, $B_3 = [A_1C_2]/2$ et $C_2$. Sinon nous essayons dans le cas général.

***Cas général***

Pour le cas général, nous considérons que les angles $\alpha_1$, $\alpha_3$, $\alpha_2$ et $\alpha_4$ (cf. 3.9) doivent avoir le même signe afin que les courbes $bez_1$ et $bez_2$ puissent être réunies en une seule.

$$\mathcal{C}_8 = \mathfrak{C}_8^1 \vee \mathfrak{C}_8^2$$

$$\mathfrak{C}_8^1 = (\alpha_1 < 0) \wedge (\alpha_2 < 0) \wedge (\alpha_4 < 0) \wedge (\alpha_5 < 0) \wedge (\alpha_6 < 0)$$

$$\mathfrak{C}_8^2 = (\alpha_1 > 0) \wedge (\alpha_2 > 0) \wedge (\alpha_4 > 0) \wedge (\alpha_5 > 0) \wedge (\alpha_6 > 0)$$

Si cette nouvelle condition est vérifiée, nous générons la courbe $bez_3$ avec pour points de contrôle $A_1$, $B_3 = (A_1B_1) \cap (C_2B_2)$ et $C_2$. Enfin, nous vérifions ce qui est défini par la condition $\mathcal{C}_6$, *p. 29*.

***Bilan***

Pour finir, et cela, dans les deux cas, nous générons l'écart quadratique moyen entre la nouvelle courbe et les courbes initiales et aussi le recouvrement de $bez_3$ par $bez_1$ et par $bez_2$. Nous refusons alors les courbes dont l'erreur maximum relevée est supérieure à la limite fixée.

#### 3.2.2.3 Prise en compte de l'erreur générée à l'identification

Les courbes sur lesquelles nous travaillons sont issues de contours discrets, elles ne constituent qu'une des représentations possibles des points de frontière. L'erreur entre les points discrets et la courbe va nous permettre de calculer des déformations sur les courbes $bez_1$ et $bez_2$, afin d'envisager un appariement qui aurait pu être refusé dans le simple cas théorique. En fait, nous allons itérer un certain nombre de fois ($nbCycle$) la recherche de meilleure déformation à appliquer sur $bez_1$ et $bez_2$ afin d'élire le meilleur appariement.

La déformation correspond à une homothétie[3] d'origine $D$ appliquée sur le point $B$ d'une courbe selon le segment $[DB]$ et qui est fonction de l'erreur relevée sur cette courbe et d'un coefficient ($coef$) qui évolue à chaque itération. Cette fonction de déformation homothétique génère une nouvelle courbe de Bézier : $newBez = f(bez, coef)$.

Comme nous déformons les courbes afin d'obtenir des résultats là où nous n'en trouvions pas, il est possible que plusieurs solutions acceptables soient rencontrées. De ce fait, la meilleure solution n'est pas forcément la première que nous allons créer.

C'est pourquoi, quatre possibilités d'appariement sont étudiées :
$appBez_1 = app\big(f(bez_1, coef), f(bez_2, coef)\big)$,
$appBez_2 = app\big(f(bez_1, -coef), f(bez_2, coef)\big)$,
$appBez_3 = app\big(f(bez_1, coef), f(bez_2, -coef)\big)$,
$appBez_4 = app\big(f(bez_1, -coef), f(bez_2, -coef)\big)$.

La condition suivante est appliquée sur chaque $appBez_k$ afin de choisir les appariements viables :

$$\mathcal{C}_9 = \frac{currentError}{currentCoverage} < \frac{maxError}{minCoverage}$$

Où $currentError$ correspond à l'erreur relevée, $currentCoverage$ au recouvrement des deux courbes initiales sur celle générée, $maxError$ au paramètre utilisateur représentant l'erreur maximale tolérée et enfin $minCoverage$ au paramètre utilisateur représentant le taux de recouvrement minimale toléré. Dans ce critère nous cherchons à maximiser le recouvrement tout en minimisant l'erreur.

Puis, nous choisissons celui dont le rapport $\frac{currentError}{currentCoverage}$ est le plus petit. Si nous avons trouvé un appariement qui est meilleur que ceux obtenus aux étapes précédentes, alors nous continuons en appliquant à cette configuration un paramètre de progression fixé par l'utilisateur ($pp$). Ainsi, la valeur de $coef$ évolue de manière géométrique et est initialement fixée à $0.5$ de manière à progresser doucement sans pour autant prendre trop d'itération pour affecter concrètement une courbe.

$$coef_{i+1} = coef_i \times pp$$

[3]Le facteur d'homothétie est exploité par la fonction $f(bez, coef)$ et se calcule selon : $fact = 1 + \frac{bezError}{\|[DB]\|} \times coef$, où $bezError$ est l'erreur intrinsèque relevée sur la courbe de Bézier $bez$.

Sinon, nous rentrons dans un processus de régression pour $coef$, de manière à réduire la déformation appliquée aux courbes afin de trouver un meilleur appariement. Le coefficient est modifié de manière à prendre en compte le nombre d'itérations qu'il reste à effectuée par :

$$\begin{aligned} coef_{i+1} &= coef_i \times pp_{new} \\ pp_{new} &= \left( \frac{max(coef_{i-1}, coef_i)}{min(coef_{i-1}, coef_i)} \right)^{\frac{1}{nbCycle-i}} \end{aligned} \tag{3.11}$$

avec $i$ correspondant à l'itération courante et $i \in [0, nbCycle[$.

Ensuite, si de cette étape itérative, une courbe a pu être trouvée, c'est-à-dire un appariement a généré un taux d'erreur minime et un taux de recouvrement suffisant (cf. $\mathcal{C}_9$, *p. 31*), nous affinons la courbe par l'application d'un vecteur de correction au point $B_3$ (cf. section 3.2.1.2).

Afin de limiter la génération de courbes ayant une forte courbure, un filtre est appliqué pour ne sélectionner que les courbes générées dont l'angle $\alpha_1$ est supérieur à 90 degrés.

#### 3.2.2.4 Mise en application

**Remarque** : Nous considérons qu'une courbe ne peut faire partie que d'un seul appariement. En effet, les contours que nous avons extraits précédemment correspondent, si la méthode d'extraction est correcte, aux contours des objets recherchés et ne peuvent donc appartenir à un seul contour d'un seul objet. Ainsi, l'appariement correct et précis avec un autre contour ne peut s'effectuer qu'avec un contour du même objet.

Pour la mise en application, nous avons développé une méthode qui traite toutes les courbes de notre espace de courbes extraites. Pour chaque courbe nous vérifions s'il existe une possibilité d'appariement avec une de ses voisines proches (nous utilisons pour cela le graphe de distances, cf. annexe E). Grâce à cette étape nous générons une liste des appariements possibles pour *toutes* ces courbes. Nous trions cette liste de manière à examiner les meilleurs appariements en premier. Ainsi, comme nous considérons qu'une courbe ne peut faire partie que d'un seul appariement, nous conservons le meilleur appariement de la liste ($newBez_1 = app\big(bez_1, bez_2\big)$) et supprimons de l'espace des courbes celles ayant permis cet appariement ($bez_1$ et $bez_2$). Enfin, nous parcourons la liste des appariements de manière à détruire les autres appariements dépendant des courbes choisies dans l'appariement en cours ($bez_1$ ou $bez_2$) et forcément de moins bonne qualité puisque la liste est ordonnée. Ainsi, nous permettons à d'autres appariements d'être acceptés par la suite.

Par exemple : soient trois appariements $newBez_1 = app\big(bez_1, bez_2\big)$, $newBez_2 = app\big(bez_1, bez_4\big)$ et $newBez_3 = app\big(bez_5, bez_4\big)$, avec cet ordonnancement $newBez_1 > newBez_2 > newBez_3$, le choix de $newBez_1$ induit une dépendance sur $newBez_2$ mais pas sur $newBez_3$ donc $newBez_2$ est détruit, subsiste $newBez_3$.

Ce procédé est itéré jusqu'à ce qu'aucune nouvelle courbe ne soit générée. Il est intéressant de noter que dans le cas d'un artefact ayant entraîné une erreur de segmentation et de vectorisation du contour d'une feuille, il peut résulter de cette étape des courbes se chevauchant. Supposons, deux grandes courbes séparées par une petite courbe représentant l'artefact (non alignée avec le reste du contour), l'appariement va regrouper les deux grandes courbes en une seule mais la petite, de part son alignement, ne sera pas prise en compte. Il en résultera que la petite courbe et la grande courbe seront superposées, ce qui va à l'encontre de la remarque faite en début de section.

Une alternative est possible afin de prendre en considération toutes les possibilités d'appariement. En

effet, la méthode présentée ci-dessus ne conserve que les appariements 'viables', mais ce choix est basé sur un critère qui n'est pas infaillible (cf. *$\mathcal{C}_9$, p. 31*) car il est basé sur l'erreur relevée sur la courbe. Cette erreur peut provenir de différentes sources comme la qualité de l'image, la résolution de l'image, la tolérance de segmentation en région, *etc.*. De ce fait, il peut paraître judicieux de reconsidérer la remarque du début de section et de supposer, à la vue des différentes sources d'erreur, qu'un contour peut faire parti de plusieurs appariements à cause de sa relative imprécision. De ce fait, il est possible d'obtenir des appariements corrects mais ayant un score plus faible qu'un appariement incorrect. Ainsi, pour éviter de supprimer un appariement correct mais ayant un score plus faible (suppression du fait des dépendances), une autre approche peut tenter de conserver tous les appariements de manière 'intelligente'. Il faut, en effet, un procédé permettant de réduire l'information redondante (*i.e.* les doublons et les chevauchement) générée par plusieurs itérations de cette approche. Par exemple, avec trois courbes contiguës : $bez_1$ contiguë à $bez_2$, $bez_2$ contiguë à $bez_3$ ; la méthode va générer trois appariements possibles ($bez_1 + bez_2$, $bez_2 + bez_3$ et $bez_1 + bez_3$) qui, à l'itération suivante, généreront chacun des courbes de Bézier regroupant les trois courbes d'origines (par exemple, $(bez_1 + bez_3) + bez_2$). La résolution de ces conflits (afin de réduire le nombre de courbes au strict nécessaire) est d'ordre $O(n^2)$, avec $n$ le nombre de courbes à l'origine des conflits, ce qui fait que cette alternative est trop coûteuse.

Cet excès de complexité ne semble pas nécessaire si nous supposons que les étapes précédentes sont suffisamment fiables pour que le précédent critère de choix (cf. *$\mathcal{C}_9$, p. 31*), basé sur la liste triée, ne se trompe que rarement. Ainsi, nous n'utiliserons que la première méthode décrite dans cette section pour définir l'appariement.

## 3.3 Description du modèle de forme

Dans notre approche, un modèle définit le contour de la silhouette de l'objet que nous cherchons à représenter. De ce fait, il représente une vue de l'objet à segmenter qui peut varier selon l'échelle, mais, aussi, selon l'angle de vision ou encore, comme il s'agit dans notre cas de végétaux (êtres vivants), de caractéristiques physiologiques (par exemple, la vigueur d'une feuille dépend de sa teneur en eau). Le modèle doit donc être capable de supporter un certain nombre de déformations pour s'adapter à la variabilité de l'objet, mais ces déformations doivent être limitées de manière à respecter la connaissance qu'il représente.

Plusieurs type de modèles s'offrent à nous : des modèles rigides, des modèles déformables, *etc.*. Les différentes approches de modélisation sont décrites à la section 2.1. Pour notre part, nous avons défini nos modèles sur la base d'un ensemble de courbes de Bézier de degré 2 représentant le contour de l'objet recherché. Le choix du degré se justifie par l'augmentation de la quantité de calculs nécessaires pour la résolution de problèmes analytiques (tel que la longueur d'une courbe, le calcul du projeté orthogonal d'un point, *etc.*) lorsque le degré de la courbe augmente. De plus, une courbe de degré 2 est un élément suffisamment simple pour permettre la représentation de n'importe quelle forme avec plusieurs de ces courbes. Cette silhouette de l'objet à représenter est définie de manière schématique, à savoir que nous ne considérons pas les détails de la forme en dessous d'un certain niveau de précision. De ce fait, une feuille d'ortie, dont le pourtour est crénelé, pourra avoir la même représentation qu'une feuille de ficus au pourtour continu. Pour passer outre cette limitation, une approche multi-échelles (comme, par exemple, [LLCS98]) a fait l'objet de discussions afin de représenter la forme de l'objet à différents niveaux de détails et ainsi, représenter un maximum d'information utiles. Mais, nous avons préféré nous orienter vers des modèles aux détails plus limités qui correspondent plus aux objets recherchés dans notre étude (cf. figure 3.10).

De plus, nos modèles comportent une information de nervure, pouvant aussi se définir comme un squelette si nous en considérons un enchaînement. Nous utilisons cette information pour associer un certain nombre de courbes du modèle à une nervure. Ces associations nous permettent de définir et limiter les déformations à appliquer à aux courbes du modèle en limitant l'application des déformations au seul squelette (voir section 4.2.1). Plusieurs nervures peuvent être définies, elle seront alors liées entre elles par leurs extrémités afin de représenter comme des articulations.

Dans nos modèles, ces courbes de silhouette sont ordonnées de manière spécifique facilitant la définition des motifs caractéristiques (cf. section 4.1.1). De plus, les courbes sont chaînées, c'est-à-dire définies de sorte que le point d'extrémité finale de l'une corresponde au point d'extrémité initiale de l'autre. De ce fait, la dernière courbe d'un modèle est liée à la première. Le sens de la boucle défini par cet enchaînement de courbes est le même pour tous modèles et nous permet, ainsi, de ne pas introduire d'ambiguïté dans la définition de nos motifs caractéristiques (intérieur et extérieur de l'objet). De plus, aux points de contrôles de chaque courbe appartenant au modèle, est associée une information d'erreur relative qui nous permet de définir des variations sur la forme de chaque courbe. Cette information est utilisée dans la génération des motifs caractéristiques pour en définir l'incertitude. Les courbes du modèle peuvent, ainsi, être considérés comme la moyenne des silhouettes observées pour un type d'objet donné et les informations d'erreurs peuvent permettre de définir un écart type sur cette silhouette moyenne afin de prendre en compte un maximum de cas. Nos modèles contiennent les informations de taille minimale et maximale de l'objet recherché et, aussi, l'information de classe colorimétrique (moyennes et covariances). Cette classe colorimétrique est déterminée de manière supervisée en sélectionnant les régions représentatives de la couleur de l'objet recherché.

Dans notre étude nous allons utiliser des modèles avec deux courbes pour définir le contour de l'objet et, aussi, des modèles à quatre courbes.

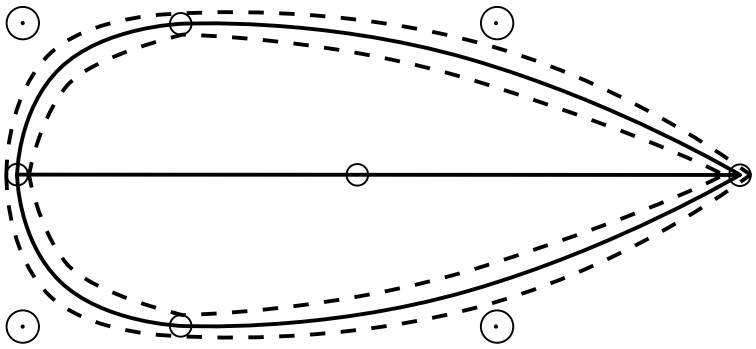


FIG. 3.10 – Exemple de modèle de forme à quatre courbes. Les cercles représentent l'espace des positions possibles pour un point de contrôle, les courbes en pointillées sont les maximum et minimum qui peuvent être définies en fonction des cercles précédents.

# 4. Reconnaissance par renforcement d'hypothèses

## Sommaire

Nous allons présenter dans ce chapitre les travaux de reconnaissance de formes que nous avons menés pour extraire des feuilles de plantes à partir d'une photographie. Le procédé exploite un ensemble de courbes de Bézier représentant les contours partiels des objets présents dans l'image, et il exploite une connaissance *a priori* sur les objets à trouver qui est représentée par un ensemble de modèles, eux-mêmes basés sur le formalisme de courbes de Bézier. Nous allons, donc, rechercher des possibilités d'appariement entre les contours partiels issus des objets de l'image avec des modèles de formes définis par l'utilisateur.

Nous remarquons, dans un premier temps, qu'il est difficile de déterminer l'existence d'un appariement à partir d'une seule courbe extraite et d'un ensemble de modèles, du fait du nombre important de solutions possibles. Aussi, dans un deuxième temps, nous pouvons noter qu'il est aussi difficilement concevable de vérifier s'il existe un appariement entre un modèle et un n-uplet quelconque de courbes extraites du fait du nombre de possibilités qu'il nous faudrait tester et de l'explosion combinatoire que cela engendrerait.

En partant de ces constatations, nous allons, d'abord, rechercher des correspondances entre un motif caractéristique issu d'un ensemble réduit de courbes extraites, et les motifs caractéristiques qui peuvent exister dans nos modèles. Cette étape va nous générer un ensemble d'hypothèses. Ensuite, nous allons vérifier s'il est possible de renforcer ces 'hypothèses' d'objet en leur ajoutant des courbes extraites se situant dans leur voisinage proche.

Dans le cadre de notre étude, nous définissons une hypothèse comme un modèle de forme composé de $i$ courbes et associé à un ensemble de $j$ courbes extraites de l'image, qui sont supposées appartenir au même objet (avec des valeur de $i$ et $j$ indépendantes).

Cette méthode sera illustrée par un exemple : celui des feuilles oblongues, modélisées par des modèles à deux et quatre courbes de Bézier.

## 4.1 Génération d'hypothèses

La génération d'hypothèses est basée sur la recherche de motifs caractéristiques dans l'espace des courbes, qui vont servir à initialiser notre hypothèse de modèle. Dans le cas des objets que nous recherchons (feuilles oblongues), nous avons défini comme motif caractéristique le sommet formé par l'intersection de deux courbes de Bézier car il est le plus pertinent dans notre cas d'étude. De plus, pour des raisons algorithmiques il peut être plus intéressant de définir des motifs caractéristiques simples et non composés d'un ensemble plus complexe de relation entre différents éléments. Pour d'autres types de forme, d'autres motifs caractéristiques pourront être définis afin d'être mieux adaptés à la forme choisie.

### 4.1.1 Motif caractéristique

Cette information de sommet peut, aussi bien, être calculée entre des courbes $bez_1$ et $bez_2$ issues de l'image (sommet $\alpha_{img}$), que celles issus des modèles $\alpha_{mod}$. L'information concernant les motifs caractéristiques des modèles est calculée une seule fois au début de notre procédé contrairement aux motifs caractéristiques issus de l'image qui sont, eux, calculés au moment opportun.

Nous manipulons un objet *sommet* comportant entre autres, les informations d'angle réel, d'intervalle d'angle (calculé selon l'erreur relevée sur les courbes) et des coordonnées 2D du point d'intersection. Ainsi, un sommet issu de l'image et un autre issu des modèles sont équivalents, si l'intersection de leur intervalle d'angles respectif n'est pas vide.

La précision du calcul de ce point d'intersection est importante car les résultats des étapes suivantes en dépendront. Il s'est avéré que dans certains cas le point d'intersection des tangentes (noté $\cap_{\tan}$ sur la figure

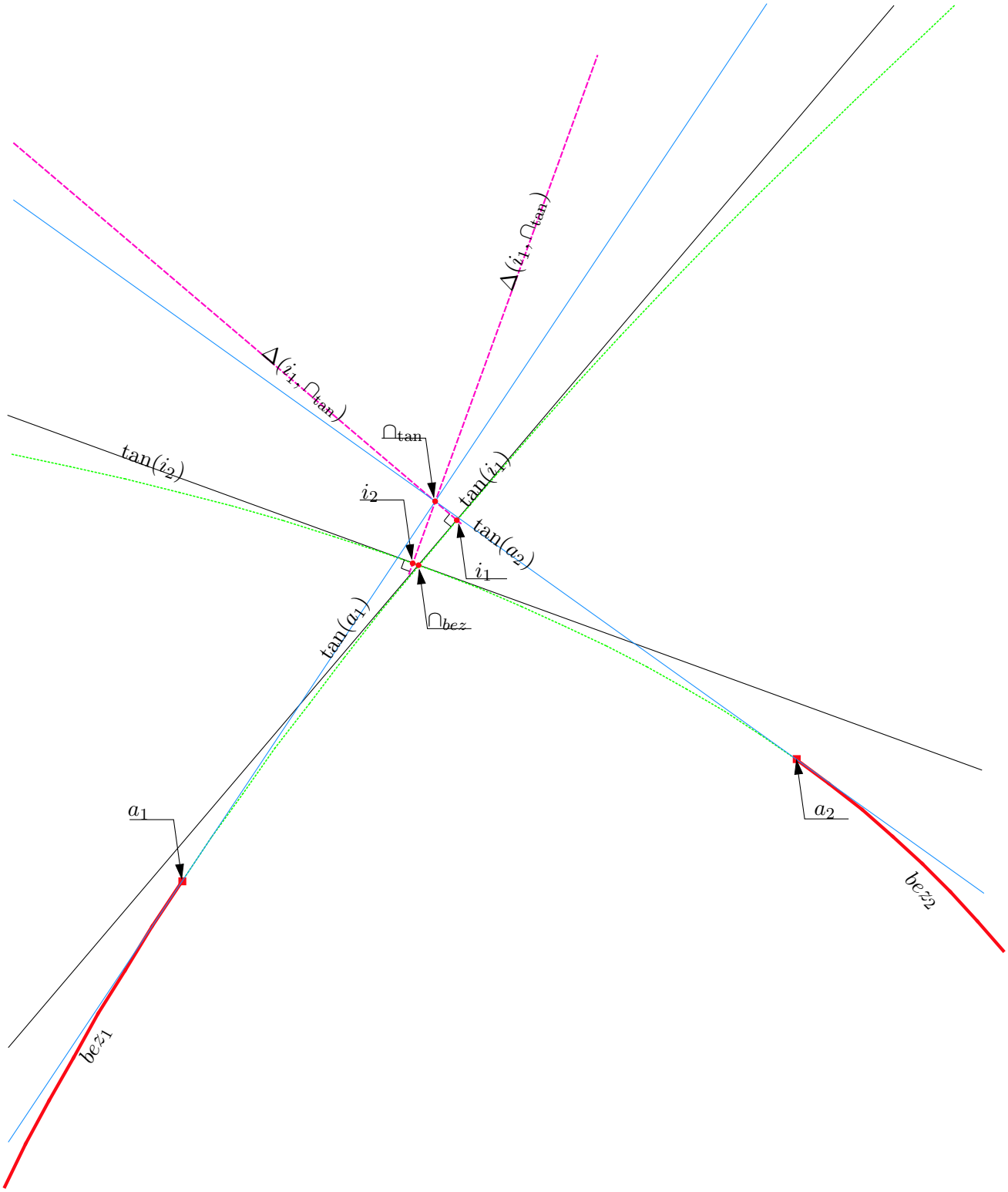



FIG. 4.1 – Approximation du point d'intersection de deux courbes de Bézier. $bez_x$ représente une courbe partielle, $a_x$ est un point d'extrémité d'une courbe, $\tan(a_x)$ est la tangente de $bez_x$ en $a_x$, $\cap_{\tan}$ est le point d'intersection des $\tan(a_x)$, $i_x = \perp_{bez_x}^{\cap_{\tan}}$ est le projeté perpendiculaire de $\tan(a_x)$ sur $bez_x$, $\Delta(i_x, \cap_{\tan})$ est la droite passant par $i_x$ et $\cap_{\tan}$, $\tan(i_x)$ est la tangente de $bez_x$ en $i_x$, enfin $\cap_{bez}$ est l'approximation du point d'intersection des courbes.

4.1) était trop éloigné de la position approximée par $\cap_{bez}$ (cf. figure 4.1) et que la méthode de recherche d'hypothèses ne trouvait pas de solutions avec ces courbes.

Le calcul de l'intersection des tangentes ne répond pas complètement au problème car le résultat est trop approximatif. Nous avons amélioré ce calcul par un procédé qui peut être répété autant de fois que nécessaire, améliorant la précision à chaque itération.

La figure 4.1 illustre le procédé d'approximation. Ce processus commence par déterminer l'intersection des tangentes des deux courbes de Bézier en leur point d'extrémité. Puis, il calcule le projeté (cf. section 3.1.3) de ce point d'intersection sur chacune des courbes : nous obtenons les points $i_1$ et $i_2$. Ensuite, il calcule, à nouveau, les tangentes aux courbes en ces points. Et enfin, l'intersection de ces tangentes ; ce point d'intersection est $\cap_{bez}$ retenu comme point d'intersection approximé des courbes.

En pratique, une seule étape amène à suffisamment de précision pour que nous n'itérions pas plus dans notre méthode. Nous pouvons observer sur la figure 4.1 que le point d'intersection des courbes se confond avec $\cap_{bez}$.

Avec les informations issues des étapes précédentes (courbes de Bézier, motifs caractéristiques viables, *etc.*), nous allons vérifier s'il existe des hypothèses de modèle compatibles avec les courbes image.

### 4.1.2 Méthode

Nous allons, donc, chercher dans l'espace des courbes image, les couples de courbes qui peuvent former un sommet 'viable'. Cette recherche est effectuée dans l'espace des couples possibles *via* le graphe de distances (cf. annexe E). La condition $\mathcal{C}_8$, *p. 30* nous permet de sélectionner, pour chaque couple de courbe, ceux qui peuvent être assimilés à des motifs caractéristiques. De plus, la condition pré-citée est complétée grâce à une information inhérente à chaque courbe. Cette information supplémentaire nous renseigne sur la position de l'objet recherché par rapport à une courbe. Ainsi, nous ne sélectionnons que les couples dont l'orientation de l'objet est cohérente avec notre problème, c'est-à-dire l'objet est situé à l'intérieur du couple.

Lorsque nous avons trouvé une équivalence entre deux sommets, du fait que nous recherchons une intersection d'intervalles, les valeurs d'angle pour ces deux sommets ne sont pas forcement égales. De cette observation, nous devons re-dimensionner le modèle choisi, de manière à passer de l'angle de $\alpha_{mod}$ à l'angle de $\alpha_{img}$, *i.e.* son échelle est ajustée selon l'axe $X$ (dilatation). Puis, nous déplaçons le modèle au point d'intersection déterminé par les courbes image ($\alpha_{img}.intPt$) et faisons subir une rotation de manière à ce que notre modèle soit bien placé. Ensuite, nous allons rechercher l'intervalle d'échelle le plus adapté de notre modèle pour correspondre avec les courbes image $bez_1$ et $bez_2$.

Ainsi, pour un somment $\alpha_{img}$ donné, nous pouvons vérifier s'il existe un modèle parmi l'ensemble des modèles, dont un sommet $\alpha_{mod}$ peut correspondre à $\alpha_{img}$. Nous allons, alors, essayer de trouver une position, une rotation et une échelle applicable au modèle choisi. Nous appliquons cette recherche pour tous les sommets qui peuvent être extraits de l'espace des courbes de Bézier.

### 4.1.3 Rotation de l'hypothèse

Une fois l'hypothèse placée au point d'intersection déterminé précédemment, nous devons lui faire subir une rotation pour l'aligner avec les courbes issues de l'image.

Plusieurs possibilités de calculs d'angles se sont offertes à nous pour choisir la bonne valeur de rotation (illustrées sur la figure 4.2).

- De la tangente de la première (resp. deuxième) courbe du modèle à la tangente de la première (resp.

**Algorithme 1** : *Génération d'hypothèses*

*Entrée : graphe de distance / sortie : ensemble d'hypothèses*
**pour** $\forall$ *couple de courbes du graphe* **faire**
  **si** *ces courbes forment un motif caractéristique valide* **alors**
    **pour** $\forall$ *angle* $\alpha_{mod}$ *existant dans les modèles* **faire**
      **si** $\alpha_{mod} \cap \alpha_{img} \neq \phi$ **alors**
        hypo = créer hypothèse sur le modèle de $\alpha_{mod}.modele$ ;
        déplacer hypo à la position de $\alpha_{img}$ ;
        ajuster l'échelle en $X$ de hypo pour passer de $\alpha_{mod}.angle$ à $\alpha_{img}.angle$ ;
        $\alpha_{Rot}$ = calculer l'angle de rotation en fonction des courbes image / modèle ;
        faire tourner hypo de $\alpha_{Rot}$ ;
        **si** *(hypo = calcul de l'échelle par la méthode directe)* == $\phi$ **alors**
          hypo = calcul de l'échelle par la méthode itérative ;
        **si** *hypo* $\neq \phi$ *ET* $\mathcal{C}_{I0} > 0$ **alors**
          ajouter hypo à la liste des hypothèses ;

deuxième) courbe image (flèches noires trait continu)

- De la médiane entre les courbes du modèle à la médiane des courbes image (flèche rouge – pointillés).
- De la tangente de la veine du modèle à la médiane des courbes image (flèche bleue – tirets).

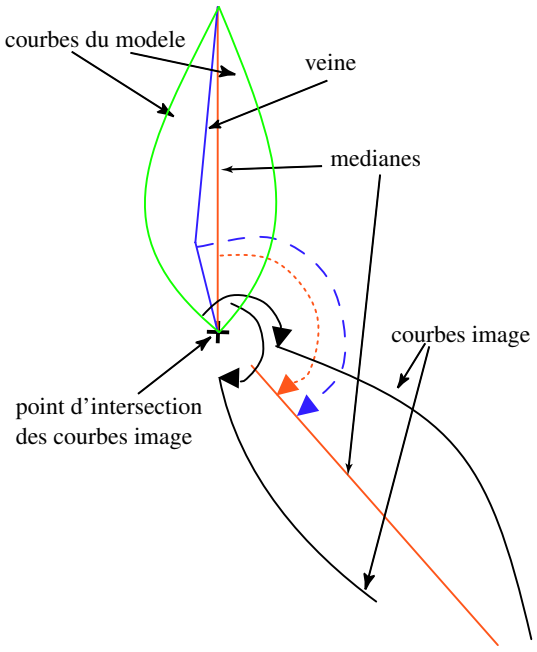



FIG. 4.2 – Rotation du modèle

Il n'y a pas de choix parmi ces différentes mesures qui soit plus pertinent qu'un autre dans tous les cas. Car la forme de l'objet que nous cherchons à identifier ne va pas être exactement celle du modèle que nous testons, même si le modèle choisi est le bon. Il est ainsi assez difficile de déterminer quelle est la meilleure approche. La première donne les meilleurs résultats empiriques, la troisième prend en compte l'information d'orientation de la feuille en fonction de la nervure ce qui devrait être plus précis mais ce n'est pas le cas. En effet, si les courbes image ont des rayons de courbure trop différents, le calcul de la médiane sera faussée

et l'alignement avec la tangente à la veine du modèle sera incorrect. La deuxième devrait être plus générale, car elle considère la veine du modèle et la médiane des courbes image, mais elle manque aussi de précision pour les mêmes raisons.

La première méthode a été retenue pour déterminer l'angle de rotation. Aussi, elle permet d'obtenir deux angles mais leur moyenne ne minimise pas l'erreur dans tous les cas, car, dans certaines configurations, la mesure de l'un des deux angles est très imprécise et son erreur associée est plus importante. Cette imprécision provient en général de la longueur de la courbe : plus la courbe est courte plus l'erreur lors de la mesure d'angle est importante. De ce fait, pour limiter l'erreur nous appliquons la première méthode, non pas en fonction de la moyenne des angles, mais en fonction de l'angle déterminé à partir de la courbe du modèle la plus longue.

### 4.1.4 Mise à l'échelle de l'hypothèse

Pour déterminer l'échelle la mieux adaptée pour le modèle en fonction des courbes image, nous utilisons deux approches : l'une permettant de calculer directement l'échelle à appliquer, appelée 'méthode directe', et une autre de progression géométrique 'méthode itérative', utilisée si la première échoue. Les deux sont guidées par le calcul d'un score basé sur des valeurs d'erreur et de recouvrement relatif calculées à chaque changement d'échelle.

#### 4.1.4.1 Calcul du score

Le score $\mathcal{C}_{10}$ est calculé à partir des taux d'erreur (qui doit être inférieure à une valeur de tolérance donnée), de recouvrement (qui doit être supérieure à une valeur de tolérance donnée) et de la taille de l'hypothèse. Nous utilisons pour cela les comparateurs flous (cf. annexe D) suivants (le choix des valeurs est le meilleur compromis empirique relevé à partir d'une série de tests) :

- $fMinError = FuzzComp(exp, 10.0)$ où $exp$ correspond à une fonction regroupant la valeur d'erreur et celle de recouvrement pondérée par le nombre de courbes de l'hypothèse actuelle : plus le nombre de courbes d'une hypothèse est important plus le critère est strict, ce qui nous permet d'être plus tolérant lorsque les hypothèses contiennent trop peu d'information pour être valables :

$$exp = \frac{maxError}{minCoverage \times g(nbCurvesInHypo)}$$

  où $g$ est une fonction de pondération définie par

$$g = max(1, (5 - nbCurvesInHypo))$$

  elle nous permet de corriger l'erreur en la minimisant plus le nombre de courbes dans l'hypothèse est inférieure à une borne (fixée empiriquement à 5).
- $fMinErrorMax = FuzzComp(maxError, 10.0)$. Ce comparateur nous permet de vérifier que la moyenne de l'erreur n'a pas absorbée une valeur d'erreur discriminante. Par exemple, dans le cas où nous relevons l'écart entre deux courbes en cinquante points, les quarante-huit premiers écarts sont proches de zéro et les deux derniers proches de vingt. Alors, la moyenne de ces cinquante valeurs va diminuer l'importance discriminante des deux dernières valeurs.
- $fMaxSize = FuzzComp(tailleMaxModele, 5.0)$ et $fMinSize = FuzzComp(tailleMinModele, 5.0)$. Ces comparateurs s'assurent que le modèle a une taille raisonnable.

Ce qui nous permet de définir la règle floue suivante (le nom spécifié sous le $\wedge$ caractérise le type de T-norme et de T-conorme utilisé, cf. annexe D et la section des notations en début de document) :

$$\mathcal{C}_{10} = \mathfrak{C}^1_{10} \overset{\wedge}{\underset{\text{prob}}{}} \mathfrak{C}^2_{10}$$

où

$$\mathfrak{C}^1_{10} = \left( fMinError > \left( \frac{currentError}{currentCoverage} \right) \right) \overset{\wedge}{\underset{\text{zadeh}}{}} (fMinErrorMax > maxCurrentError)$$

$$\mathfrak{C}^2_{10} = (fMaxSize > tailleHypo) \overset{\wedge}{\underset{\text{zadeh}}{}} (fMinSize < tailleHypo)$$

#### 4.1.4.2 Méthode directe

Cette méthode est définie sur l'hypothèse qu'en fonction des courbes images sur lesquelles nous souhaitons caler le modèle, il est possible de déterminer, par un calcul non-itératif, l'échelle du modèle. Ce calcul se fait sur toutes les courbes image que nous souhaitons intégrer à l'hypothèse.

Chacune des courbes image est associée à une courbe du modèle, en fonction de la distance les séparant et du degré de colinéarité (cf. section 3.1.1). Des informations relatives à cette association sont stockées, comme celles de la zone de recouvrement maximal entre les deux courbes (partie hachurée sur la figure 4.3). Cette zone est délimitée par $p^1_{img}$ et $p^2_{img}$ sur la courbe image et $p^1_{mod}$ et $p^2_{mod}$ sur la courbe du modèle associée.

La figure 4.3 illustre l'idée de la méthode : à partir des points précédemment cités et dans le but d'annuler l'écart reproduit par la zone hachurée, nous calculons un ensemble de rapports de distances entre des points d'une courbe image et des points d'une courbe modèle et nous sélectionnons le plus grand rapport comme facteur d'échelle (identique en $X$ et $Y$). Ainsi, pour une courbe image donnée et pour une position de recouvrement (pour l'exemple, nous choisissons $p^1$), nous déterminons les projetés des points $p^1_{img}$ et $p^1_{mod}$ selon les axes $X$ et $Y$ du repère de la veine[1], nous obtenons les points $\perp^{p^1_{mod}}_X$, $\perp^{p^1_{img}}_X$, $\perp^{p^1_{mod}}_Y$ et $\perp^{p^1_{img}}_Y$. Puis, nous calculons les distances entre ces projetés et l'axe du repère correspondant.

Nous calculons les rapports $R^1_x = \frac{x^1_{img}}{x^1_{mod}}$ et $R^1_y = \frac{y^1_{img}}{y^1_{mod}}$. Et, comme nous cherchons à combler l'espace entre le modèle et la courbe, alors nous prenons le maximum des deux rapports : $R^1 = max(R^1_x, R^1_y)$. $R^1$ sera une des valeurs que nous allons utiliser comme coefficient d'échelle.

Ce procédé va donc aussi s'appliquer sur $p^2_{img}$ afin de calculer $R^2$. Le procédé sera répété sur les $p^k_{img}$ de chacune des courbes images sur lesquelles nous souhaitons caler le modèle ce qui nous permettra de sélectionner le facteur d'échelle le plus adapté.

Nous pouvons noter que pour le point $p^2_{img}$, le projeté sur la veine du point du modèle se confond avec ce même point, il est donc assez difficile de calculer un rapport $R^2_y = \frac{y^2_{img}}{y^2_{mod}}$ avec un $y^2_{mod}$ quasi nul. De ce fait, cette méthode présente un certain nombre de cas limites qui peuvent donner des coefficients trop grands ou trop petits, donc, inexploitables.

Lorsque cette méthode d'approximation échoue nous utilisons une méthode itérative décrite dans la section suivante. Néanmoins, la méthode directe reste plus rapide que la méthode itérative et donne de bons résultats dans nombre de cas.

[1] Ce repère est actuellement défini comme un repère orthonormé situé à la base de la veine.

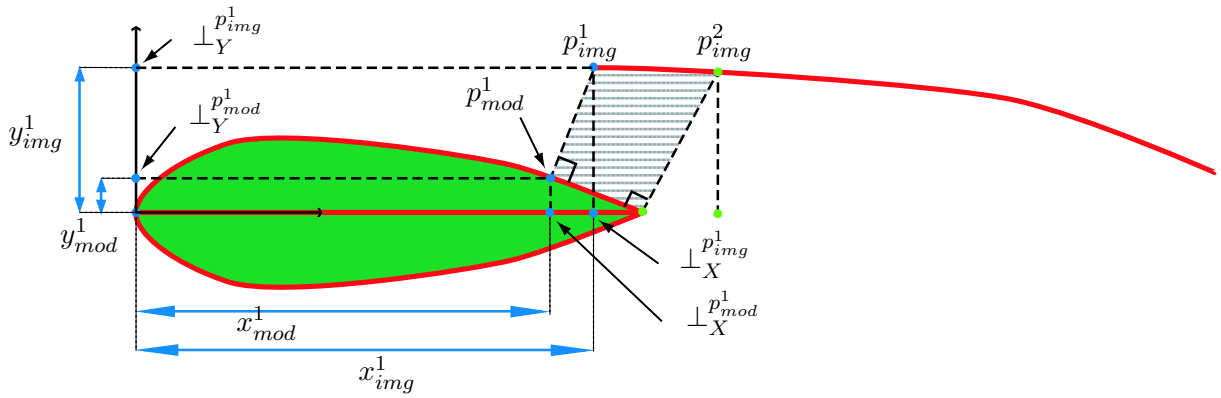



FIG. 4.3 – Calcul direct du facteur d'échelle. La feuille représente l'hypothèse de modèle déjà existante. La courbe en haut à droite est une courbe image. La partie hachurée représente la zone de recouvrement, au sens des projetés perpendiculaires, entre les courbes modèle et image.

#### 4.1.4.3 Méthode itérative

La méthode, qui nous permet de déterminer la meilleure échelle dans tous les cas, est une dérivée de l'approche dichotomique bornée sur un nombre d'itérations donné ($nbCycle$). Comme il n'y a pas particulièrement de borne au facteur d'échelle ($Ech \in ]0, +\infty[$), nous commençons par déterminer le sens de progression de cette recherche : nous essayons les échelles $Ech_1 = Ech_0 \times coef^{-1}$ et $Ech_1 = Ech_0 \times coef$ et nous déterminons laquelle des deux minimise l'erreur en calculant les scores obtenus par $\mathcal{C}_{10}$. $coef$ est un paramètre utilisateur, la valeur empirique (issue d'une série de tests) que nous utilisons est 1.2.

Le score résultant nous permet d'entamer un processus itératif qui va modifier l'échelle du modèle à chaque itération selon un coefficient donné ($coef$). Nous pouvons remarquer que l'évolution de l'échelle du modèle est géométrique car nous multiplions l'échelle du modèle par ce coefficient. Lorsque ce processus a engendré un modèle trop petit ou trop grand, c'est-à-dire le score actuel (cf. $\mathcal{C}_{10}$, *p. 44*) est plus petit qu'à l'étape précédente, nous rentrons alors dans une nouvelle étape itérative avec une valeur de coefficient de progression recalculée. Nous allons modifier la valeur de ce coefficient de manière à faire évoluer l'échelle du modèle dans l'autre sens afin de raffiner cette valeur et de déterminer une meilleure échelle. Le coefficient est modifié de manière à prendre en compte le nombre d'itérations qu'il reste à effectuer par l'équation 3.11, p. 32. Le processus itère selon un nombre d'itération fixé ($nbCycle$).

Il est intéressant de noter que la valeur de l'échelle du modèle sera la même pour les axes $X$ et $Y$ (mis à part la transformation – dilatation – d'échelle appliquée à l'étape de rotation précédente).

### Bilan

À la fin de cette étape de génération d'hypothèses, nous obtenons un ensemble d'hypothèses ajustées au mieux dans l'image en ne considérant que seulement deux courbes image pour répondre à ce problème. Chacune de ces hypothèses a été jugée viable en fonction du critère d'erreur défini par $\mathcal{C}_{10}$, *p. 44*. Nous allons maintenant renforcer chaque hypothèse en y ajoutant de l'information supplémentaire pour les conforter ou bien les infirmer.

## 4.2 Renforcement d'hypothèses

Pour résoudre le problème qui nous est donné de renforcer les hypothèses générées précédemment nous utilisons le procédé illustré en figure 4.4. Nous allons rechercher dans l'espace des courbes image celles qui sont dans la proximité de l'hypothèse à traiter puis, parmi celles-ci, sélectionner en fonction de critères, les courbes candidates à l'ajout. Enfin, la dernière étape va nous permettre de trouver une minimisation de l'erreur entre une hypothèse et les courbes image qu'elle contient, c'est-à-dire de déterminer le meilleur placement / orientation / échelle / déformation possible entre une hypothèse existante et des courbes images additionnelles.

Pour valider les nouvelles courbes images à injecter dans une hypothèse existante, une dérivé multidimensionnelle de la méthode de *Newton* (cf. [Sch74]) va être exploitée. En effet, la procédure utilisée à l'étape précédente (cf. section 4.1) ne convient plus du fait des procédés qui sont impliqués pour obtenir de bonnes hypothèses avec peu d'information. Ces procédés pourraient, maintenant, ne pas permettre de trouver de meilleurs placements et rotations (car ces valeurs ne sont pas remises en cause dans ce procédé) et la valeur d'échelle est la même pour les deux dimensions. Cette méthode ne pourrait, donc, pas permettre de renforcer convenablement l'hypothèse. De plus, la méthode de minimisation que nous allons employer ne peut être utilisée correctement dans l'étape précédente car le peu d'information exploitée la ferait diverger et ne nous permettrait pas d'obtenir les premières hypothèses.

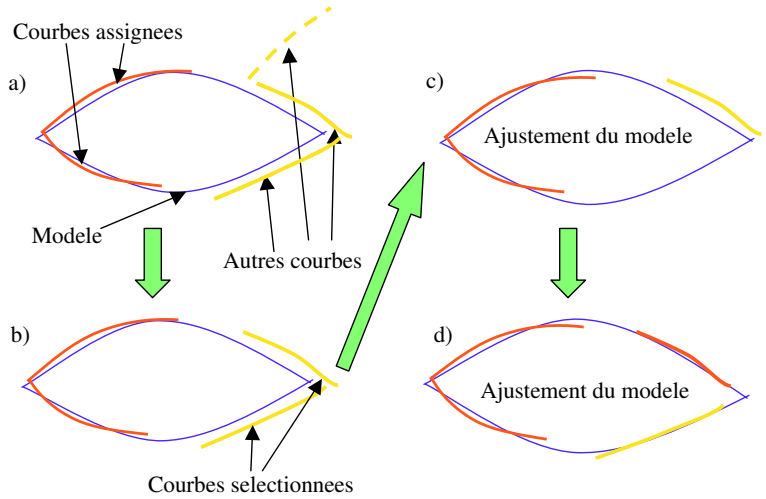



FIG. 4.4 – a) recherche dans le voisinage, b) sélection colinéaire, c) d) ajustement du modèle courbe par courbe.

### 4.2.1 Déformation des courbes de l'hypothèse

Un processus de déformation peut être appliqué aux courbes d'une hypothèse en fonction d'un vecteur de déformation. Ce vecteur peut être assimilé à une quantité de mouvement à affecter à un point $Pt$ de la courbe, en fonction d'un point de base $p^1_{ref}$, d'un point d'application $p^2_{ref}$ et d'un vecteur de déformation $\overrightarrow{F}$ (cf. figure 4.5). Nous pouvons l'exprimer par :

$$\overrightarrow{f_{Pt}} = \sqrt{\frac{||p^1_{ref} - \perp^{Pt}_{\Delta_{ref}}||}{d_{ref}}} \times \overrightarrow{F}$$

Où $\Delta_{ref}$ correspond à la droite passant par $p^1_{ref}$ et $p^2_{ref}$. Ce calcul permet de réduire progressivement,

pour un point $Pt$, l'influence du vecteur de déformation, plus $\perp_{\Delta_{ref}}^{Pt}$ est proche de $p_{ref}^1$ (*i.e.* est éloigné de $p_{ref}^2$). L'application de ce calcul dans le cas des courbes de Bézier présente un avantage car il nous suffit de l'appliquer que sur les points de contrôle (voir figure 4.5 et figure 4.6).

Remarque : nous n'utilisons pas l'information de distance entre un point de la courbe et son projeté sur la droite de référence $\Delta_{ref}$, car sinon la déformation ne serait plus réversible. Il peut sembler étrange de ne pas utiliser cette information de distance, car plus un point est éloigné de la droite de référence plus l'influence du vecteur de déformation devrait être faible. C'est pourquoi, pour pallier à cette simplification, la définition du modèle associe à une veine, les courbes du modèle qui lui sont proches. Ainsi, le procédé de déformation est appliqué selon les veines du modèle, et donc, aux courbes associées du modèle (car ces courbes sont relativement proches) ce qui permet de justifier cette approximation.

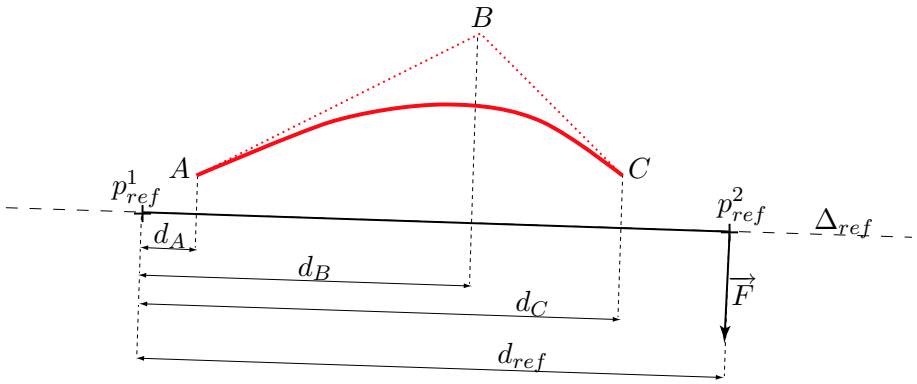



FIG. 4.5 – Données prises en compte lors du calcul de déformation.

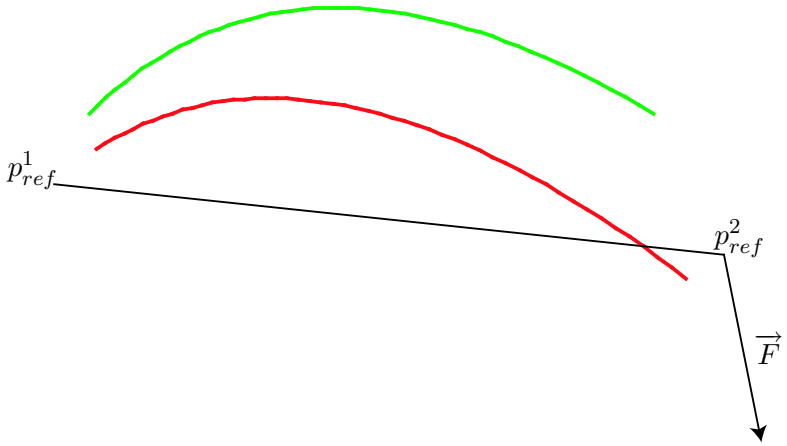



FIG. 4.6 – Exemple de déformation. Nous déformons la courbe verte – gris clair – en la courbe rouge – gris foncé – en appliquant le vecteur $\overrightarrow{F}$.

### 4.2.2 Choix des nouvelles courbes

Le choix des nouvelles courbes à ajouter à l'hypothèse en cours de traitement, se fait selon plusieurs critères. Premièrement, selon un critère de voisinage, ainsi, sont candidates les courbes images incluses dans la boîte maximale englobante du modèle. C'est-à-dire, une hypothèse fait référence au modèle sur lequel elle se base, et ce modèle contient une information de taille minimale et maximale de l'objet recherché. De ce fait, nous pouvons calculer, en fonction de l'orientation de l'hypothèse, de son placement dans l'image et de la taille maximale, un rectangle circonscrit au modèle de taille maximale. Il ne nous reste plus qu'à

déterminer grâce au graphe de distances les courbes proches de notre hypothèse de départ et inclues dans la boîte englobante.

Deuxièmement, nous utilisons un critère de compatibilité avec les courbes déjà existantes dans l'hypothèse : le procédé vérifie pour chaque nouvelle courbe, s'il existe un appariement possible avec une des courbes déjà existantes de l'hypothèse (cf. section 3.2.2). Cette étape n'est pas disqualifiante, mais elle permet de réduire l'information contenue dans l'hypothèse et de combler les espaces entre les courbes qui n'auraient pas été appariées précédemment. Comparativement à l'appariement qui peut être effectué sur l'ensemble des courbes image comme décrit dans l'aperçu de la méthode (cf. section 5.1), les paramètres d'appariement sont moins stricts car, même s'il est moins fiable, l'appariement sera confirmé si l'hypothèse est valide.

Troisièmement, nous utilisons un critère de compatibilité avec le modèle sous-jacent à l'hypothèse : le procédé vérifie si les courbes image choisies sont colinéaires (cf. section 3.1.1) avec une des courbes du modèle. Cette étape nous permet d'établir une association possible entre une courbe image et une courbe du modèle qui sera utile dans l'étape suivante.

### 4.2.3 Méthode de renforcement par ajustement

Ensuite pour chaque courbe image considérée comme acceptable pour être ajoutée à une hypothèse donnée, la méthode de *Newton* va être appliquée pour placer et ajuster le modèle.

Ainsi, nous utilisons une dérivée multidimensionnelle de la méthode de *Newton* (cf. [Sch74]) qui est optimisée grâce aux facilités offertes par les courbes de Bézier de degré 2. Cette procédure, itérative, cherche à annuler le résultat $Y$ d'une fonction $f(X)$ en appliquant des corrections sur le paramètre $x$.

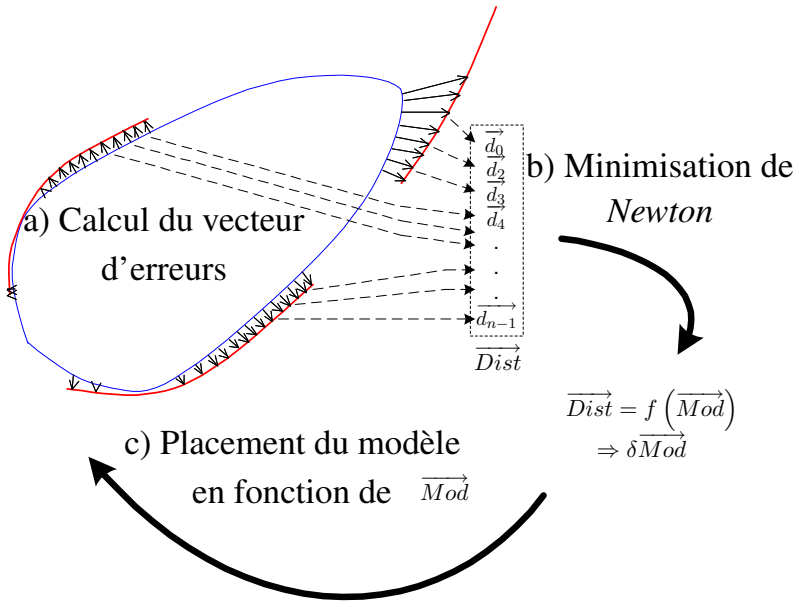



FIG. 4.7 – Ajustement du modèle par la méthode de minimisation de *Newton*.

Dans notre cas, la fonction à annuler est une équation à variables multiples où le paramètre $x$ est identifié à un vecteur reprenant toutes les caractéristiques du modèle ($\overrightarrow{Mod}$ de taille $k$). Le vecteur $\overrightarrow{Mod}$ contient les informations de positionnement du modèle en $x$ et $y$, ainsi que l'angle de rotation, les facteurs d'échelle pour $x$ et $y$ et le vecteur de déformation en $x$ et $y$. Le résultat $Y$ correspond, alors, à l'ensemble ($\overrightarrow{Dist}$) des vecteurs distance calculés qui représentent l'erreur résiduelle entre les courbes du modèle et

celles de l'image. Nous avons donc : $\overrightarrow{Dist} = f\left(\overrightarrow{Mod}\right)$.

La correction itérative à appliquer correspond à $\delta X = -J^I.Y$, où $J^I$ est la pseudo-inverse du Jacobien de $f(X)$ de taille $n \times k$. Ainsi, nous obtenons : $\delta\overrightarrow{Mod} = -J^I.\overrightarrow{Dist}$.

À chaque étape du processus de minimisation, nous calculons un vecteur d'écarts ($\overrightarrow{Dist}$) à annuler, correspondant à l'erreur résiduelle entre les courbes du modèle et celle de l'image (cf. figure 4.7.a). Ensuite, grâce au processus de *Newton* nous obtenons un vecteur de correction pour le modèle ($\delta\overrightarrow{Mod}$) avec lequel nous pouvons calculer le nouveau modèle (cf. figure 4.7.c).

Dans notre méthode optimisée, le principal gain de temps significatif provient du calcul de l'erreur résiduelle entre les courbes du modèle et celles de l'image ($\overrightarrow{Dist}$). Avec un type quelconque de courbe, nous aurions dû calculer l'intégrale de l'erreur résiduelle par un processus numérique. Grâce aux courbes de Bézier de degré 2, nous la calculons en utilisant la procédure analytique qui permet d'obtenir, pour un point donné, sa position la plus proche sur une courbe (cf. 3.1.3). Nous itérons cette méthode analytique pour un nombre prédéfini de points sur la courbe et, ainsi, déterminons l'intégrale de l'erreur résiduelle.

Cette étape de renforcement nous permet de réajuster les hypothèses car le vecteur $\overrightarrow{Mod}$ contient les informations de position, échelle, rotation et déformation applicable au modèle sous-jacent à l'hypothèse. Nous répétons cette étape de minimisation jusqu'à obtenir un score acceptable (calculé avec la règle floue $\mathcal{C}_{10}$, *p. 44*).

Cette étape est répétée jusqu'à ce qu'il n'y ait plus de nouvelle hypothèse générée par le processus.

### 4.2.4 Génération des résultats

Dans l'étape de renforcement d'hypothèses, de nouvelles hypothèses peuvent être créées depuis une même hypothèse générée à l'étape précédente, en fonction des courbes image ajoutées. Nous posons la propriété suivante : si à une hypothèse $h$ est ajoutée une courbe $a$ puis une courbe $b$ l'hypothèse obtenue est identique à celle obtenue à partir de $h$ à qui nous ajoutons $b$ puis $a$. De ce fait, nous pouvons observer une redondance dans les résultats obtenus selon le moyen utilisé pour ajouter des courbes à une hypothèse.

Actuellement, deux approches ont été prises en compte lors de l'ajout, à une hypothèse $h$, d'une nouvelle courbe issu d'un ensemble de courbes $E$.

La première, prend les nouvelles courbes une par une. Elle ajoute une de ces courbes à l'hypothèse $h$ générant du même coup une nouvelle hypothèse $h'$ (si $h'$ est viable). Ainsi, si nous disposons d'un ensemble $E$ de taille $n$ de courbes candidates, nous pouvons avoir $n$ nouvelles hypothèses $h'$, toutes se basant sur $h$. Ensuite, chaque nouvelle hypothèse va, à nouveau, considérer son voisinage, en extraire le nouvel ensemble $E'$ de courbes à ajouter (avec $E'$ pouvant être égal à $E$), et réitérer l'étape de renforcement d'hypothèses. Nous générons, alors, comme un arbre d'hypothèses à partir de $h$ de toutes les combinaisons obtenues à partir de $E$. Car $h$ génère $h'_1$, $h'_2$, *etc.* avec chaque courbe de $E$ puis pour chaque $h'$ nous réitérons avec $E'$ ce qui nous donne des $h''$, nous obtenons, ainsi un arbre d'hypothèses. Mais nous générons aussi beaucoup de doublons car, par exemple, nous allons générer une hypothèse pour l'ensemble de courbes $\{1, 2, 3\}$ mais aussi pour l'ensemble $\{3, 2, 1\}$. Le résultat de la méthode avec les courbes illustrées par la figure 4.8.a est présenté sur la figure 4.9.

La deuxième méthode, ajoute la première courbe $bez$ de l'ensemble $E'$ à $h$ (avec $E'$ initialisé à $E$) et si l'hypothèse est viable, recommence ce cycle d'ajout en sélectionnant la première courbe de $E'$ avec $E' = E' - \{bez\}$ jusqu'à ce que $E'$ soit vide (*i.e.* nous avons obtenu une feuille de l'arbre dont l'hypothèse est valide) ou bien que l'hypothèse générée ne soit pas viable. Lorsque l'hypothèse générée n'est pas viable ou que $E'$ est vide, nous remontons dans l'arbre des hypothèses jusqu'à la dernière hypothèse générée viable

et continuons le procédé avec le $E'$ utilisé par cette dernière hypothèse viable. Et nous supprimons de $E'$ l'élément $bez$ à l'origine de notre sous-arbre ($E' = E' - \{bez\}$). Les résultats de la méthode avec les courbes illustrées par les figures 4.8.a,b,c sont présentés sur les figure 4.10, 4.11 et 4.12.

Il est intéressant de remarquer que les deux méthodes partent de deux propositions différentes : le premier procédé suppose que le voisinage d'une hypothèse est susceptible de changer au fur à mesure que l'hypothèse est renforcée du fait que son échelle est ajustée et son orientation et position sont réajustées. Le deuxième procédé suppose lui que l'exploration du voisinage contenu dans la boîte englobante maximale est suffisant. Le premier est théoriquement plus valable car il semble possible que l'ajustement de la première hypothèse générée ne soit par le meilleur et que le voisinage extrait à ce moment ne soit pas le plus représentatif. Néanmoins, le surplus d'informations qui pourrait être extrait par la re-exploration du voisinage n'est pas pertinent car notre but est de rechercher une approximation d'un modèle et non pas une application exacte. De plus, cette première méthode est plus coûteuse que la seconde.

Les résultats des deux méthodes sont présentés sur les figures 4.8 à 4.12. La figure 4.8 illustre les configurations de courbes utilisées. Sur cette figure, les courbes du haut sont celles qui ont permis la génération de la première hypothèse (cf. section 4.1), de plus, deux modèles de feuille différents peuvent être adaptés selon les courbes image choisies (en pointillés sur la figure 4.8.a). La notation employée dans les nœuds des arbres est la suivante :

*{courbes employées dans l'hypothèse}{courbes restantes}*

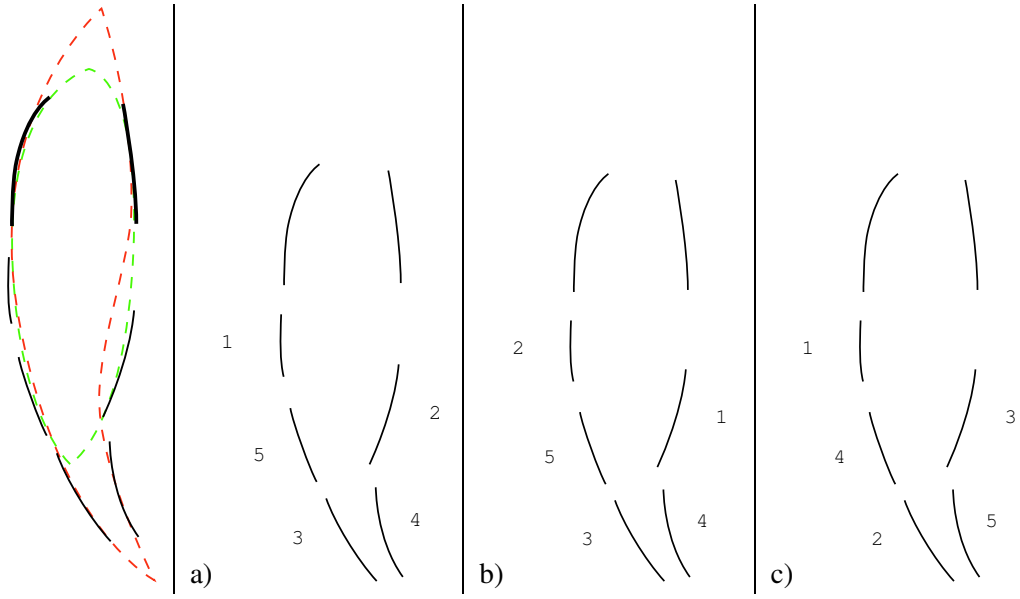



FIG. 4.8 – Courbes testées avec les deux méthodes. À gauche, les objets feuille que nous pouvons extraire avec ces courbes. a), b) et c) Figures servant de support aux graphes, les courbes sont les mêmes, seul change l'ordre d'ajout dans l'hypothèse. Les courbes du haut (traits continus) initialisent l'hypothèse.

#### 4.2.4.1 Méthode gloutonne

Le premier arbre (cf. figure 4.9) n'est représenté que partiellement car il est trop grand et qu'une partie permet de représenter les désavantages de la première méthode. En effet, cette méthode génère beaucoup de doublons car il n'y a pas une mémoire de ce qui a déjà été traité. De plus, le parcours de l'arbre se faisant en largeur, nous allons aussi bien tester l'hypothèse $\{1,2,3\}\{4,5\}$ que $\{3,1,2\}\{4,5\}$, alors qu'elles représentent le même objet. Néanmoins, cette procédure est celle qui a été implantée pour des raisons de temps. Ces désavantages sont compensés dans la deuxième méthode en ordonnant les courbes et ainsi, en

conservant une mémoire du travail déjà effectué.

#### 4.2.4.2 Méthode ordonnée

Les trois autres arbres représentent les différents résultats obtenus par la deuxième méthode appliquée sur les trois configurations de courbes (figures 4.8.a, b et c). Sur l'arbre illustré en figure 4.10, les cadres, utilisant le même type de pointillé, représentent les schémas dans l'arbre qui se répètent. Il découle de l'observation de ces cadres semblables que :

- l'ensemble de courbes employées pour générer les hypothèses d'un cadre à droite est un sous-ensemble ayant servi pour son semblable à gauche (*i.e.* $E_d \subset E_g$, $E_d$ et $E_g$ désignent les ensembles utilisés pour générer le cadre de *d*roite et de *g*auche).
- un cadre de droite est un 'oncle' du cadre de gauche.

De ces observations nous pouvons éviter la génération de doublons en limitant l'exploration de certaines branches de l'arbre si nous détectons une branche similaire déjà calculée. En effet, si $E_d \subset E_g$, *i.e.* $\{2,3\}\{F\} \subset \{1,2,3\}\{F\}$, alors nous pouvons dire que l'hypothèse $h_d$ issues de $E_d$ est, au moins, aussi bonne que l'hypothèse $h_g$ issue de $E_g$. Car, cette méthode, de par le fait qu'elle retire progressivement les courbes du même ensemble, génère un type d'ordre sur ces courbes et évite, ainsi, d'explorer les mêmes hypothèses. Ainsi, si $E_d \subset E_g$, $h_d$ et $h_g$ sont valides, alors $h_d$ est moins contrainte que $h_g$ et, donc, a au moins un meilleur score. De ce fait, il ne sera pas utile de générer $h_d$.

De plus, si $h_g$ n'est pas valide mais que $h_d$ est valide (ou *vice-versa*), alors les sous-arbres sont différents et il nous faudra continuer l'exploration du sous-arbre de droite. Et, nous pouvons remarquer une dernière propriété intéressante : si la branche la plus à gauche issue de $E_d$ est similaire à la branche la plus à gauche issue de $E_g$, alors le sous-arbre issus de $E_d$ est similaire à celui issu de $E_g$ et les frères de $E_d$ sont similaires à ceux de $E_g$. L'exploration de $E_d$ et de ses frères n'est plus nécessaire. Par exemple, sur la figure 4.11, seules les hypothèses $\{1,2,5\}\{\phi\}$ et $\{2,3,4,5\}\{\phi\}$ ont besoin d'être générées, toutes les autres sont des copies de l'une de ces deux hypothèses.

Les trois arbres illustrés sur les figure 4.10, 4.11 et 4.12 sont obtenus à partir du même jeu de courbes, seul l'ordre est différent. Il est intéressant d'observer que les résultats ne sont pas les mêmes d'un ordre à l'autre et que l'arbre de la figure 4.11 permet d'éviter l'exploration d'une bonne partie de l'arbre (*i.e.* le sous-arbre issu de la courbe 1 est suffisant) alors que, sur la figure 4.10, l'exploration du sous-arbre issu de la courbe 1 n'empêche pas l'exploration du reste. Ainsi, il serait intéressant de pouvoir déterminer un critère permettant de calculer l'ordre limitant, au maximum, les explorations dans l'arbre. Néanmoins, ce critère peut être difficilement réalisable car il nécessite de connaître, à l'avance, le résultat de l'exploration des sous-arbres pour déterminer le meilleur ordre. L'ordre utilisé dans ces exemples est celui issu de l'ordre de construction des courbes et du graphe de distances.

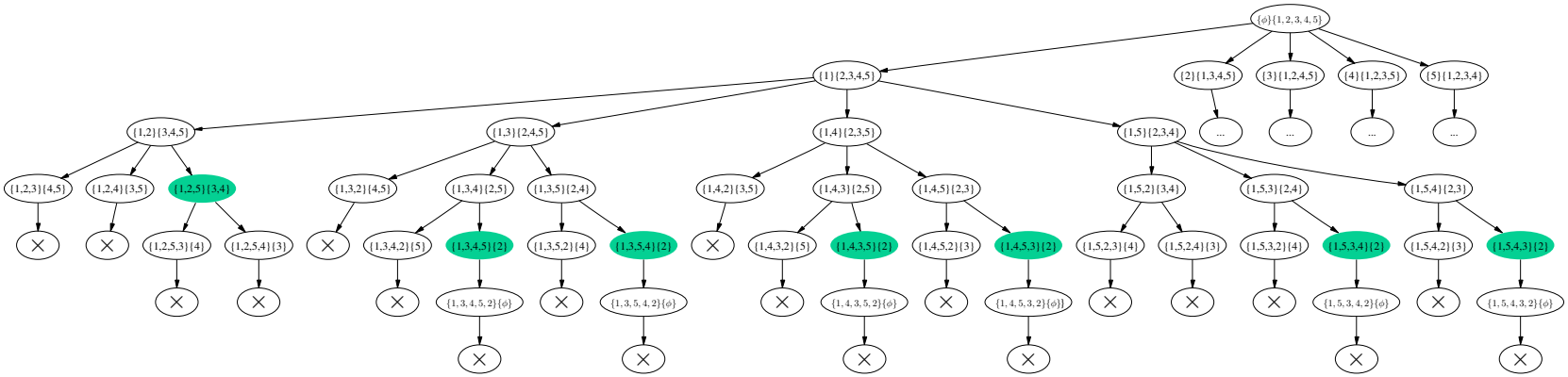



FIG. 4.9 – Arbre des hypothèses générées : avec les courbes de la figure 4.8.a selon la première méthode. Les nœuds colorés ont généré un bon résultat.

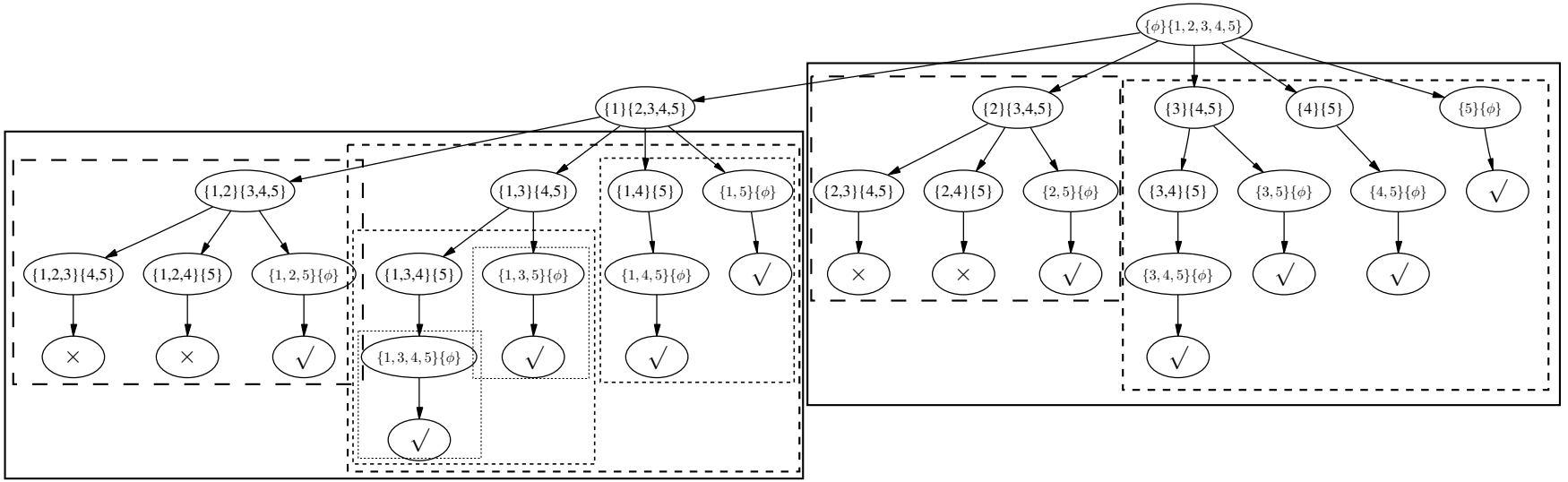



FIG. 4.10 – Arbre des hypothèses générées : avec les courbes de la figure 4.8.a selon la deuxième méthode.

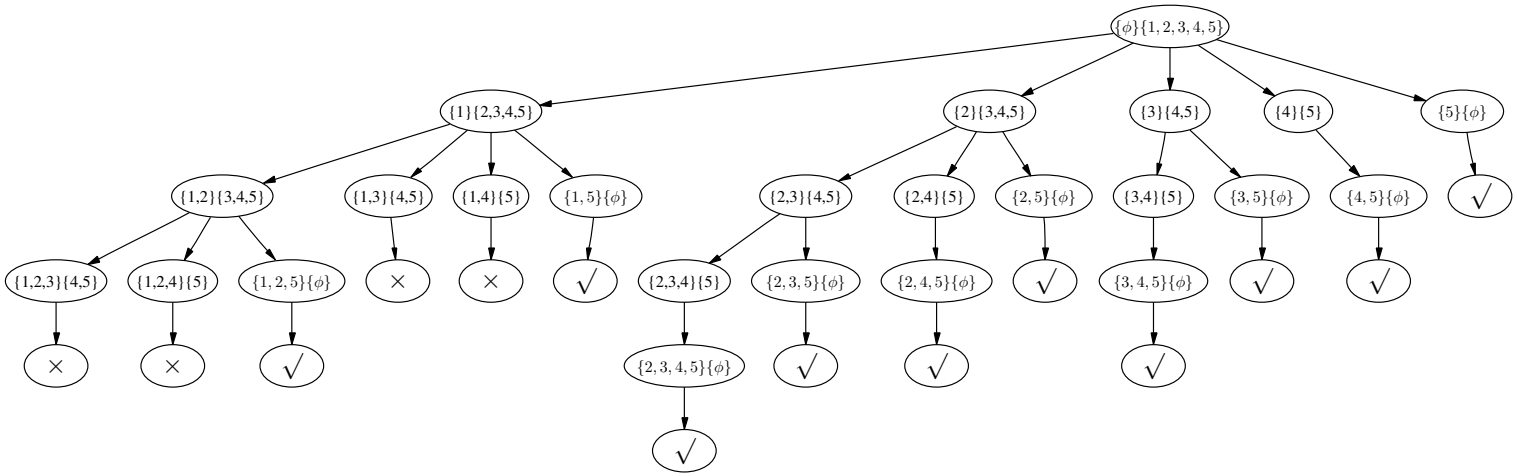



FIG. 4.11 – Arbre des hypothèses générées : avec les courbes de la figure 4.8.b selon la deuxième méthode.

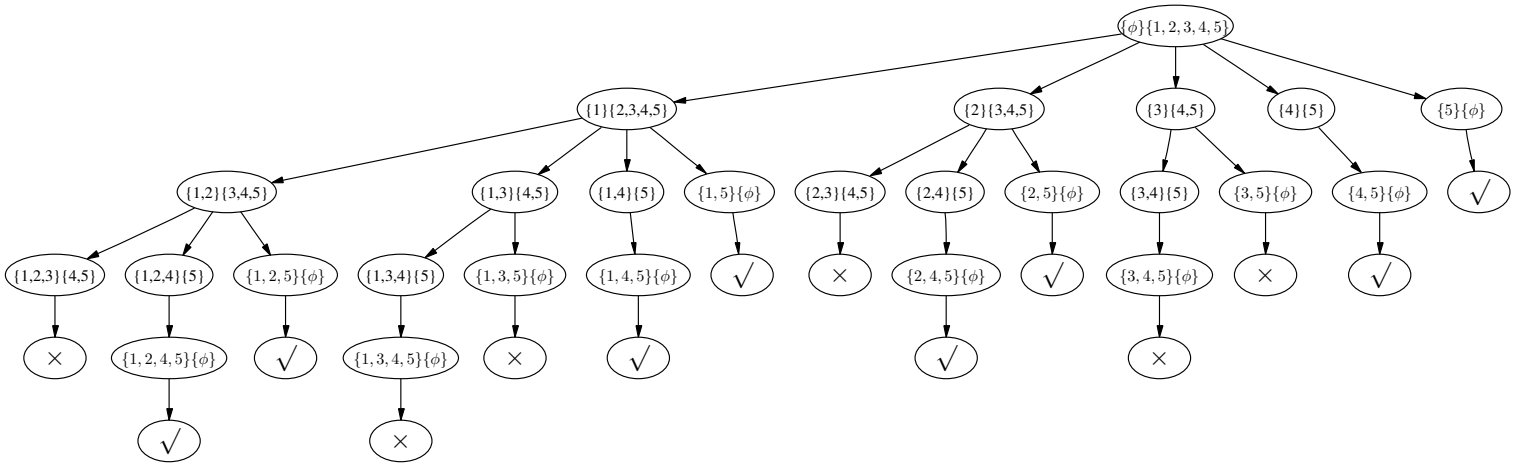



FIG. 4.12 – Arbre des hypothèses générées : avec les courbes de la figure 4.8.c selon la deuxième méthode.

## 4.3 Bilan

La méthodologie présentée dans ce chapitre, permet, de manière incrémentale, la génération d'hypothèses de présence d'objets à partir d'une connaissance *a priori* modélisant ces objets et d'un ensemble de courbes extraites de l'image. Ces hypothèses essayent de représenter au mieux un objet présent dans la scène en adaptant leur forme à ses contours. Notre méthode permet l'extraction d'un certain nombre d'objets indépendemment de leur configuration dans l'image, à partir du moment où il est possible d'extraire un minimum d'information sur les contours de cet objet et qu'il est possible d'en extraire un motif caractéristique correctement défini et exploitable par notre algorithme.

Nous allons présenter dans le chapitre suivant, le processus que nous avons mis en œuvre dans sa globalité et, aussi, présenter les résultats que nous avons pu obtenir avec un tel procédé.

# 5.
# Mise en œuvre & résultats

*When you think of how well basic appliances work,*
*it's hard to believe anyone ever gets on a plane.*

Calvin & Hobbes.

## Sommaire

Nous allons présenter dans ce chapitre un aperçu général de notre méthode et les résultats que nous avons obtenus avec notre implantation.

Le procédé de reconnaissance que nous avons mis en place est un enchaînement de méthodes : segmentation d'image, extraction et vectorisation de contours, génération et renforcement d'hypothèses. Il repose principalement sur la modélisation par courbes de Bézier (cf. chapitre 3), nous permettant de représenter toute l'information manipulée et sur le renforcement par hypothèses (cf. chapitre 4), nous permettant l'extraction des objets présents dans l'image.

## 5.1 Aperçu général du processus

La première étape consiste, à partir d'une image couleur et de données colorimétriques sur la classe des objets recherchés (ici, des feuilles), à extraire les contours des objets considérés, puis transformer ces contours en des courbes de Bézier.

La seconde étape est celle de la reconnaissance des feuilles. À partir d'un ensemble de courbes de Bézier extraites à la première étape, et de modèles de forme représentant les objets à reconnaître, nous allons chercher à générer puis renforcer des hypothèses de présence de ces formes parmi les courbes issues de l'image.

### 5.1.1 Première étape : extraction de courbes de Bézier

L'extraction des courbes de Bézier se fait sur la base d'une suite de points contigus (les contours), eux-mêmes issus des frontières entre les différentes régions résultantes d'une étape de segmentation d'images.

La segmentation d'images (cf. annexe A) a pour but de regrouper des ensembles de pixels contigus de manière à obtenir des zones – ou régions – homogènes, assimilables à des objets. Notre algorithme (cf. ALGO. 2, p. 114) génère, en plus de l'ensemble des régions de pixels similaires, un graphe d'adjacence entre ces régions (cf. figure A.2). Ce graphe nous permet de définir les frontières entre les différentes régions et nous permettra, par la suite, de déterminer les contours des objets recherchés.

Les contours issus de la segmentation sont des suites de points discrets. Chacune de ces suites est ordonnée selon le sens de lecture de l'image, c'est-à-dire de haut en bas et de gauche à droite. Il faut, donc, les ordonner selon une des orientations de la courbe représentant le contour. Nous proposons à l'annexe C un algorithme réalisant cet ordonnancement à la volée.

Enfin, les contours de ces objets sont vectorisés en courbes de Bézier du second degré. Les courbes de Bézier de degré 2, de part leur nature analytique (cf. équation 3.1), vont nous faciliter un certain nombre de calculs, comme la fusion de courbes ou la détermination analytique de l'écart quadratique entre deux courbes de Bézier. Notre algorithme de vectorisation (cf. 3.2.1) assure, pendant le parcours du contour, à la fois le découpage et la détermination des points de contrôles de ces courbes.

Un graphe de distances est généré afin d'optimiser les opérations de recherche de courbes proches et compatibles (cf. annexe E). Ensuite, de manière à réduire et à optimiser l'information manipulée par les étapes suivantes, le procédé d'appariement – *i.e.* regroupement de deux courbes dans le prolongement l'une de l'autre – est appliqué sur les courbes image selon la méthode décrite à la section 3.2.2. L'algorithme d'appariement parcourt l'ensemble des couples de courbes possibles issus du graphe de distances. De plus, grâce à l'information de position de l'objet recherché par rapport à une courbe, nous ne sélectionnons que les couples dont l'orientation par rapport à l'objet est cohérente, c'est-à-dire l'objet recherché est à l'intérieur du couple (sur l'hypothèse que nos objets sont convexes).

### 5.1.2 Deuxième étape : reconnaissance par hypothèses

En considérant les courbes de Bézier issues de l'étape d'extraction précédente nous allons rechercher les possibilités d'appariement avec des modèles des formes. Un modèle de forme est défini par un ensemble de courbes de Bézier représentant la silhouette de l'objet recherché. Nous allons essayer de définir un premier ensemble d'hypothèses en recherchant des correspondances entre un motif caractéristique issu d'un ensemble réduit de courbes extraites, et les motifs caractéristiques qui peuvent exister dans nos modèles. Puis, nous essayons de définir une hypothèse à partir d'un couple de motifs caractéristiques en plaçant et ajustant le modèle (cf. chapitre 4).

Nous allons utiliser le premier ensemble d'hypothèses comme base de travail pour compléter ces hypothèses en leur ajoutant d'autres courbes extraites. À la fin de ce processus itératif, nous obtenons un ensemble d'hypothèses de modèle – représentant les objets trouvés – dont la fiabilité est associée à un score mis à jour durant le traitement. Ce score va nous permettre d'élire les meilleures hypothèses.

## 5.2 Mise en œuvre et résultats

Nous allons présenter les résultats que nous avons obtenus par notre méthode. Dans un premier temps nous illustrerons les choix possibles en mettant en avant les avantages et désavantages. Puis, dans un deuxième temps nous commenterons les résultats de notre procédé obtenus à partir de trois images réelles.

Il est intéressant de noter que notre procédé est un enchaînement de sous-méthodes : les données exploitées à chacun de ces étages sont issues du traitement effectué par l'étage précédent. De ce fait, les procédures que nous avons développées ont à gérer les erreurs introduites et cumulées par les étapes précédentes en utilisant, par exemple, la logique floue ou bien, pour la reconnaissance de forme, un procédé d'hypothèses renforcées incrémentalement. Nous allons présenter les résultats obtenus par chaque étape en détaillant les paramètres et les sources d'erreurs possibles.

### 5.2.1 Segmentation d'images

La méthode de segmentation d'images que nous avons développée, présentée dans l'annexe A, génère un ensemble de zones de couleur homogène. Un exemple de segmentation est présenté en figure 5.1. Les résultats sont intéressants car nous obtenons une zone conséquente représentant le fond de l'image (*i.e.* la terre), les autres zones représentent des objets isolés par la méthode (feuilles, cailloux, *etc.*) . Néanmoins, certains de ces objets correspondent à des erreurs de segmentation et ne représentent pas des objets en eux-mêmes.

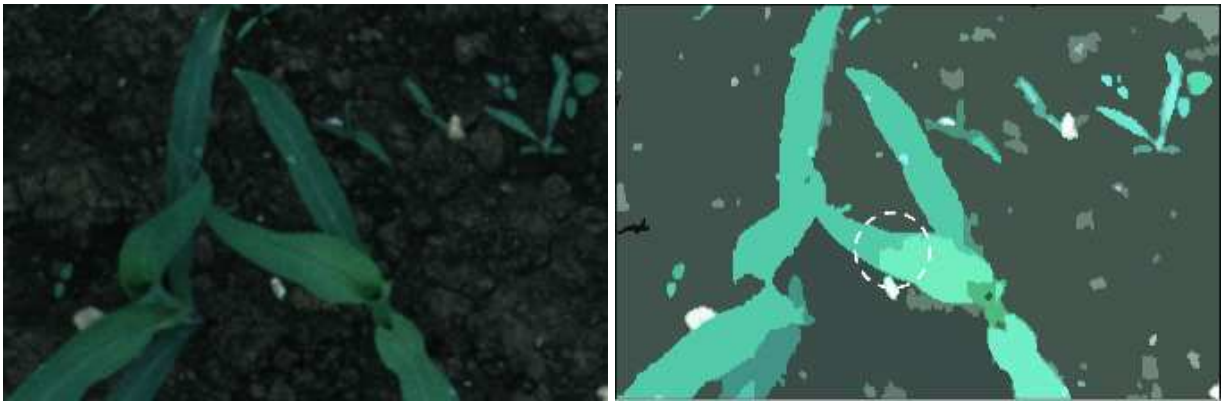


FIG. 5.1 – Résultat obtenu en RGB avec notre méthode de segmentation.

Les erreurs de segmentation que nous pouvons observer se produisent essentiellement en bordure d'objets et peuvent être aussi dues à des zones comportant un dégradé de couleurs. En effet, les pixels situés sur le bord d'un objet peuvent être ambigus et leur appartenance à telle ou telle région difficile à déterminer. En particulier, lorsque ce bord est situé à côté d'une zone d'ombre portée ou recouvert par de la poussière ou de la terre, c'est-à-dire adjacent à un dégradé de couleurs. De la même manière, lorsque deux feuilles se chevauchent, à leur intersection se forme une zone d'ombre portée sur la terre et sur la feuille, pouvant générer des erreurs lors de la segmentation. Ces erreurs de segmentation sont à la base des contours d'objets erronés et bruités. De par ces faits, il est intéressant de prendre en compte ces erreurs possibles en intégrant, dans notre procédé, la gestion de l'incertitude *via*, par exemple, la logique floue. Nous observons sur la figure 5.1 que les feuilles ont été correctement segmentées, sauf, comme mentionné plus haut, dans les régions très sombres où l'information couleur peut manquer. De même, sur cette figure nous pouvons voir l'implication des dégradés de couleurs dans les régions obtenues : la feuille est coupée en deux régions de luminosité différente (représentée par le cercle blanc dans la figure 5.1).

Notre méthode de segmentation se base sur un critère de similarité défini dans l'espace RVB (décrit dans l'annexe B). Un critère HSI a aussi été développé afin d'exploiter les informations de teinte pour améliorer la séparation du vert des feuilles, des autres couleurs ne faisant pas partie de la classe feuille. Mais, cet espace fournit des valeurs de teinte incohérentes lorsque, par exemple, la luminosité est faible. De ce fait, et comme nos images sont souvent définies dans des teintes sombres, les résultats de segmentation obtenus avec ce critère ne sont pas satisfaisants, même en corrigeant les valeurs HSI dans les zones instables et en compensant avec les informations RVB. Nous avons aussi essayé un critère générant un histogramme dans le plan teinte-saturation, afin de séparer les pics spécifiques des feuilles du reste (cf. [Ber02], annexe 2). Ce travail s'apparente à celui de Clément et Vigouroux [CV03] sur la segmentation par arbres d'histogrammes permettant l'obtention de classes représentatives des feuilles et du sol. Mais la séparation des pics n'est pas évidente et les résultats sont moins bons que ceux obtenus avec le critère RVB.

L'étape de segmentation repose sur des images dont la qualité (déformation et luminosité) de prise de vues peut être grandement améliorée. Par exemple, par l'utilisation d'un support permettant des prises de vues dans le même plan que le terrain afin de limiter les déformations de perspective, ou bien par l'utilisation d'un dispositif diffusant d'une manière uniforme la lumière sur les plantes, ou encore en plaçant l'appareil plus haut et en utilisant un capteur plus précis et un objectif à focale plus longue afin, là aussi, de limiter les déformations de perspective. Aussi, il est évident que l'espace couleur utilisé lors de la segmentation (*i.e.* RGB) ne correspond pas exactement à nos besoins et qu'un espace plus adapté aurait dû être défini. De plus, notre méthode de segmentation ne permet pas d'extraire les informations de contour au sein d'un objet composé de plusieurs autres (comme une plante composée de feuilles) à partir du moment où la couleur de ces objets adjacents est similaire. De cette observation, il parait correct de penser qu'une collaboration contours-régions peut grandement améliorer cette étape. En effet, un certain nombre de contours séparant des régions adjacentes de couleur similaire peuvent être extraits par des algorithmes connus (*i.e.* Canny, Deriche, *etc.*) alors qu'ils ne sont pas détectés par la segmentation couleur. Ce type de collaboration peut nous permettre de diviser une région plante en un ensemble de régions feuille. Par ailleurs, ce type de collaboration régions / contour a été étudiée par N. Gorretta [Gor03] avec l'exploitation de gradients couleur. Il pourrait être, aussi, intéressant d'exploiter l'information contenue dans les modèles afin de guider la segmentation, néanmoins ce type d'approche nous ramène aux problèmes liés à la reconnaissance de forme, car il faut être capable de déterminer l'alignement du modèle en fonction d'information disponible en début de segmentation pour guider la segmentation elle-même, ce qui peut poser un certains nombre de difficultés.

Le temps de calcul est relativement intéressant car il faut dix secondes pour traiter une image de 10 mégaoctets soit un gain de soixante fois par rapport à la version d'origine non optimisée.

La méthode de segmentation que nous avons développée, permet, en plus de l'extraction des régions de couleur similaire, l'extraction des contours des objets. Les suites de points représentant ces contours sont, dans l'étape suivante, ordonnés et vectorisés en courbes de Bézier.

### 5.2.2 Génération et vectorisation de contours

Afin d'exploiter l'information de contours d'objets extraits à l'étape précédente, pour identifier les objets recherchés, nous devons vectoriser ces contours. Mais, la vectorisation suppose un ordonnancement de ces points de contours. Ces procédés de génération et d'ordonnancement sont non paramétrés et correspondent à des transformations qui ne nécessitent pas de calculs susceptibles d'être entachés d'erreur. En effet, la génération de contour se déroule pendant la création du graphe d'adjacence (cf. étape de segmentation d'image, algorithme GenereGraphe, p. 115) et consiste à relever les points séparants deux régions. Aussi, l'ordonnancement de points de contours peut être assimilé à un tri sur le critère de la connexité et ne nécessite pas de calculs paramétrés (cf. annexe C).

#### 5.2.2.1 Vectorisation

En revanche, le procédé de vectorisation, décrit dans la section 3.2.1, peut être source d'erreurs car il lisse le contour à traiter puis approxime des motifs caractéristiques sur ce contour en fonction de divers paramètres (région de support, seuil pour la détection d'un sommet, *etc.*) et enfin calcule les courbes de Bézier. Ce procédé permet d'obtenir les courbes les plus longues possibles tout en minimisant l'erreur. Les résultats s'apparentent correctement aux contours d'origines si les paramètres sont choisis avec soin : ne pas trop lisser pour éviter de perdre de l'information utile mais suffisamment afin de limiter le bruit, choisir (pour la détermination des angles) une taille de région de support qui ne soit pas trop importante sinon elle produit un effet de lissage mais suffisante afin d'avoir une information d'angle correcte, *etc.*. La méthode est assez sensible aux variations des paramètres et ces derniers sont déterminés de manière empirique mais sont valables pour plusieurs images du même type. De ce fait, une méthode non paramétrée pour trouver les points dominants (comme celle de [MS04]) serait préférable. Les résultats sur une image de synthèse sont présentés figure 5.2 et sur une image réelle en figure 5.3 et 5.4. Les petits ensembles de quatre points rouges représentent les points de contrôles utilisés pour définir les courbes de Bézier.

Les cercles en surimpression sur la figure 5.2 indiquent les erreurs qui peuvent survenir du fait de l'approximation générée par ce procédé. Le même cercle sur la figure 5.4 illustre un cas d'oscillation qui n'a pas été considéré comme pertinent et qui n'a pas été retenu. Les écarts observés entre les courbes générées et les contours discrets sont dûs aux seuils fixés par l'utilisateur qui peuvent être variables d'un type d'images à l'autre. En effet, sur une image représentant des objets de synthèse ou bien des objets manufacturés, les coins d'objets ainsi que les variations d'angles représentent des traits caractéristiques de l'objet, ce qui serait intéressant de mettre en valeur. Tandis que dans le cas des objets naturels ces variations peuvent être dues à des altérations de formes et ne pas représenter des caractéristiques en soi.

La vectorisation des contours discrets en courbes de Bézier, comme nous l'avons précisé dans la partie bibliographique concernée, peut être améliorée par l'utilisation de méthodes plus efficaces pour déterminer les points dominants d'un contour discret. Et, cela sans l'utilisation de l'ensemble des paramètres de la méthode que nous avons définie, laquelle fonctionne correctement mais nécessite un ajustement précis des paramètres. Néanmoins, l'extraction des points dominants par une autre méthode ne définit pas pour autant les points de contrôle des courbes de Bézier, ce qui nous ramène à l'utilisation de notre automate. Aussi, cet automate, fini, est défini pour approximer un contour discret issu d'un objet organique, comme une feuille de plante, mais se révèle moins efficace sur des images de synthèses. Il pourrait être nécessaire de l'améliorer afin d'obtenir des résultats plus pertinents dans le cas général et aussi, pour limiter l'utilisation des paramètres. Par contre, le choix du degré des courbes de Bézier – aussi bien pour la vectorisation que pour les courbes du modèle – reste primordial car il limite les possibilités d'appariement (*i.e.* plus le degré est important, plus le nombre de possibilité d'appariement l'est aussi), simplifie les calculs et évite ainsi de retomber dans une complexité combinatoire.

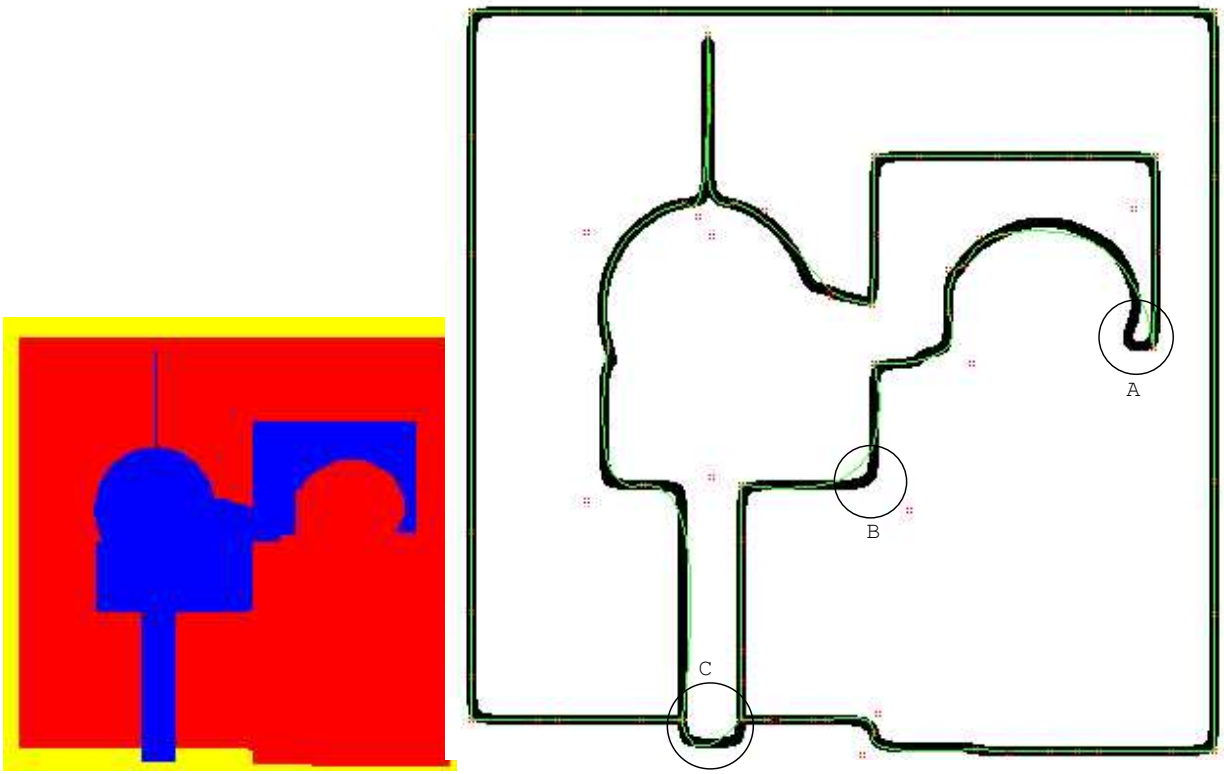



FIG. 5.2 – À droite l'image de synthèse segmentée et à gauche les contours discrets et courbes de Bézier obtenus. Le trait noir (qui a été épaissi pour l'illustration) représente le contour extrait de l'étape de segmentation et le trait clair fin représente les courbes de Bézier obtenues.

Il est intéressant de remarquer que la vectorisation obtenue n'est correcte qu'à l'échelle du pixel car nous n'exploitons pas à cette étape de notion de taille de pixel (*i.e.* un pixel représente $x$ centimètre). De ce fait, il nous est impossible de déterminer si la vectorisation obtenue est cohérente à l'échelle des objets que nous recherchons et cela peut entraîner des erreurs lors de l'exploitation de ces courbes par les étapes suivantes.

Avec notre procédé, les résultats sont généralement corrects dans le cas d'objets organiques car les erreurs possibles qui peuvent intervenir lors de la vectorisation sont compensées par les variations de formes des contours de l'objet. Afin de recouvrir les erreurs de segmentation et de vectorisation, nous essayons d'apparier deux à deux les courbes dans le prolongement l'une de l'autre.

#### 5.2.2.2 Appariement

Lorsque l'image se prête à une bonne segmentation mais que certains éléments, comme des cailloux, interrompent le contour d'une feuille, la présente méthode nous permet d'estimer le contour occulté. Mais, l'appariement de deux courbes de Bézier peut être, lui aussi, source d'erreurs car il essaye de compenser les erreurs de définition des points de contrôles des courbes impliquées, en jouant sur leurs déformations possibles pour générer un appariement qui minimisera l'erreur en respect d'un seuil fixé. Ainsi, une vectorisation correcte obtenue à l'étape précédente peut induire un refus d'appariement avec une courbe dans son prolongement (premier cas). En effet, cette courbe va être cohérente localement avec les points de contours discrets sur lesquels elle a été calculée, mais pas forcement cohérente globalement avec l'objet. Inversement, le processus d'appariement peut combiner deux courbes qui n'appartiennent pas au même objet mais à deux objets situés dans le prolongement l'un de l'autre (deuxième cas).

FIG. 5.3 – Image réelle segmentée utilisée pour l'extraction et la vectorisation de contours de la figure 5.4.

Les résultats obtenus sont très sensibles aux variations des paramètres car il est fréquent, lorsque nous augmentons la tolérance des paramètres pour l'appariement, que le premier cas (vrai négatif[1]) soit accepté mais le deuxième cas aussi (faux positif). Des résultats obtenus à partir d'images réelles sont reproduits en figure 5.5. Un cas de vrai négatif est illustré sur la figure 5.5 par la courbe marquée de la flèche A, la courbe résultante passe d'une feuille à l'autre. Les cas de faux positif ne sont pas représentés car non retenus par notre algorithme du fait qu'ils ont été considérés comme faux. Par exemple, sur cette même figure, la flèche B pourrait illustrer un cas de faux positif.

Nous pouvons aussi citer le problème de la "rosace", illustré en figure 5.6 où les courbes sont mariables d'un objet à l'autre. En effet, nous pourrions penser que l'appariement de deux courbes appartenant à des régions différentes est à proscrire mais il est fréquent d'observer (comme sur la figure 5.1) qu'un même objet soit représenté par plusieurs régions et, donc, l'appariement doit pouvoir s'effectuer entre les courbes de différents objets. De ce fait, il est possible d'avoir des appariements, même sous fortes contraintes, entre des courbes d'objets différents comme sur la figure 5.5.

La complexité du procédé de vectorisation est en $O(n)$ avec $n$ le nombre de points à traiter et la complexité de l'appariement est en $O(n^2)$ avec $n$ le nombre de courbes prises en compte, sachant que nous limitons grandement cette complexité en utilisant le graphe de distance.

Nous avons, à la fin de cette étape, généré l'information de base exploitable par notre méthode de génération et de renforcement d'hypothèses d'objets. Nous allons, maintenant, détailler les résultats obtenus avec cette dernière méthode.

[1]Cette notation sous-entend que le résultat a été considéré comme vrai (*i.e.* valide) alors qu'il s'agit d'une configuration invalide.

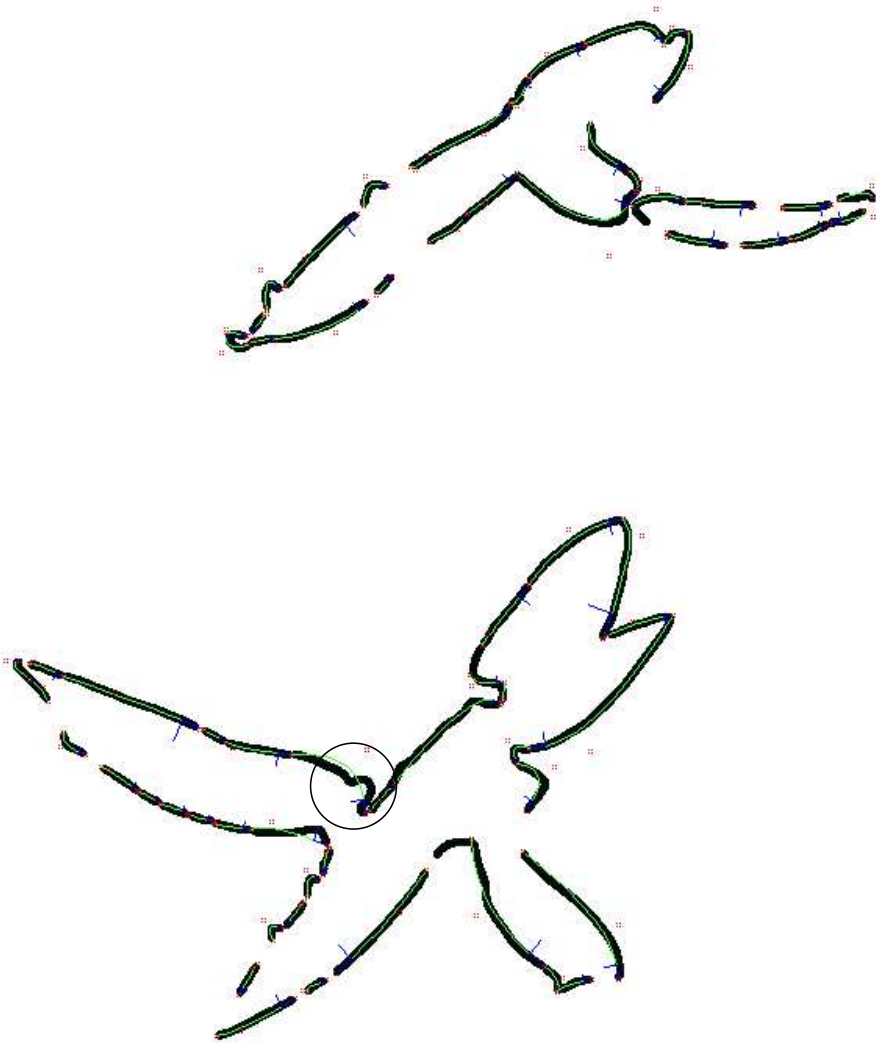


FIG. 5.4 – Contours discrets et courbes de Bézier obtenus à partir de 5.3. Le trait noir (qui a été épaissi pour l'illustration) représente le contour extrait de l'étape de segmentation et le trait clair fin représente les courbes de Bézier obtenues.

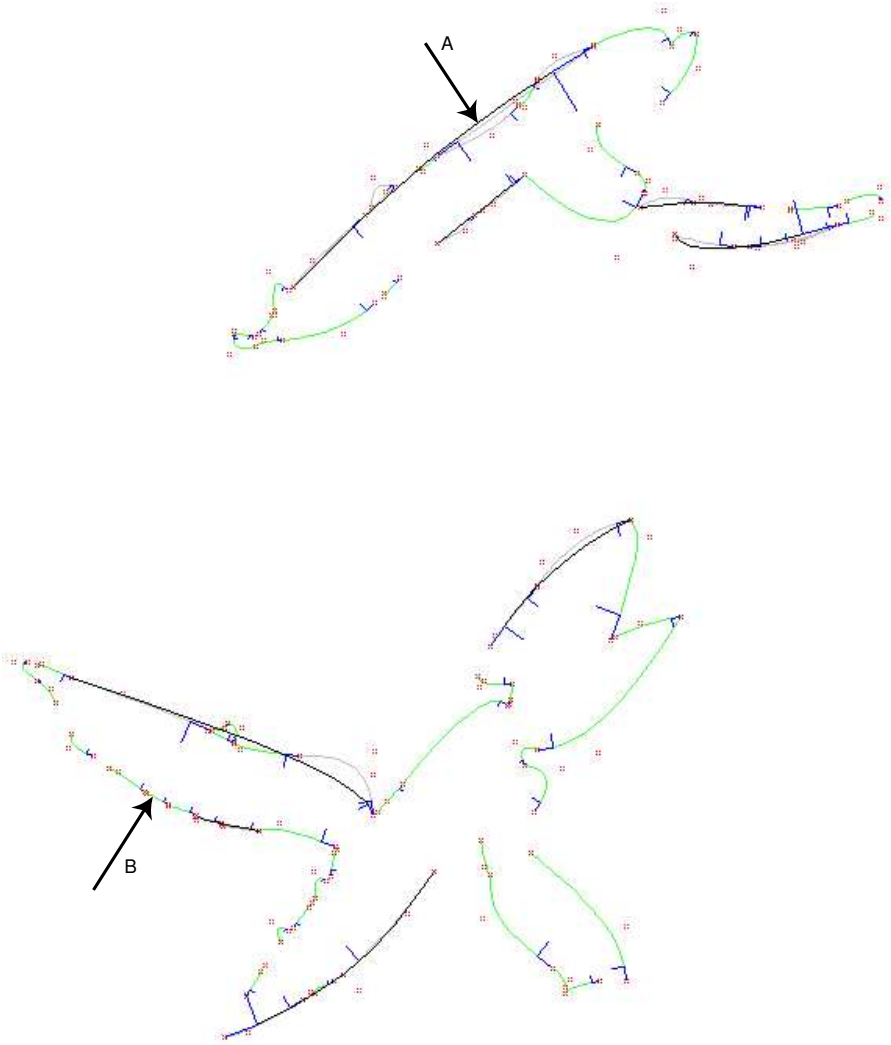



FIG. 5.5 – Appariement obtenu à partir des courbes de la figure 5.4. Les courbes noires correspondent aux courbes issues de l'appariement, les grises claires sont les courbes à l'origine d'un appariement et les autres sont des courbes non appariées.

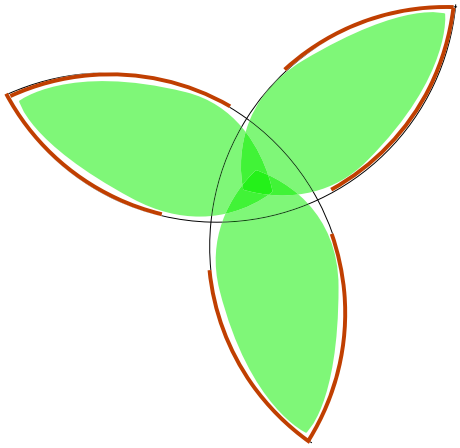


FIG. 5.6 – Problème d'ambiguïté lors de l'appariement. Les courbes avec le trait épais sont les courbes susceptibles d'être appariées et les autres représentent les appariements possibles. Les objets feuille sont représentés en vert.

### 5.2.3 Génération et renforcement d'hypothèses

La génération et le renforcement d'hypothèses, décrites au chapitre 4, se basent sur les courbes de Bézier obtenues lors des étapes précédentes. L'exactitude des informations contenues dans ces courbes va déterminer la qualité des hypothèses générées. Nous sommes conscients de manipuler des informations contenant une erreur potentiellement importante. Aussi, comme les hypothèses résultantes de notre procédé sont issues de l'étape de génération d'hypothèses et comme cette même étape va exploiter peu d'informations valides (erreur cumulée), nous allons examiner un nombre élevé de combinaisons pour favoriser la génération d'hypothèses valides.

#### 5.2.3.1 Génération d'hypothèses

Cette étape est particulièrement sensible à la qualité de l'identification des contours discrets en courbes de Bézier, du fait de la dépendance de ses différentes sous-méthodes (*i.e.* point d'intersection, rotation, mise à l'échelle) à la définition des tangentes aux extrémités des courbes.

Sur la figure 5.7 nous avons mis en valeur les motifs caractéristiques qui ont généré des hypothèses. Nous pouvons voir sur cette figure que le procédé n'a pas pu générer de motif caractéristique pour la feuille se situant le plus à droite (cercle en pointillé), pourtant clairement définie. Cela peut s'expliquer par le fait que la rotation subie par l'hypothèse (calculée à partir des tangentes), a placé cette dernière trop en dehors de la feuille pour générer un score acceptable, elle a donc été rejetée. Cette remarque souligne le fait que notre procédé de génération d'hypothèses ne remet pas en cause la valeur de rotation calculée à partir du point caractéristique. De ce fait, certaines hypothèses, dont le motif caractéristique est bien placé, ne pourront pas être validées car mal orientées (cas de faux positifs).

FIG. 5.7 – Génération d'hypothèses – motifs caractéristiques. Les courbes blanches représentent les couples de courbes sélectionnées pour générer des hypothèses, les courbes grises représentent les courbes de Bézier issues de l'identification.

Sur la figure 5.8 sont illustrés les résultats obtenus à partir des motifs caractéristiques de la figure 5.7. Plusieurs hypothèses peuvent être générées à partir d'un même motif caractéristique car plusieurs modèles peuvent avoir des motifs caractéristiques similaires et, par exemple dans notre cas, les deux extrémités

d'un modèle représentent à peu près le même motif caractéristique. Par exemple, sur cette figure, le cercle en pointillé illustre précisément ce cas. La différence d'orientation est dû à une légère asymétrie dans le modèle. Aussi, nous pouvons observer que l'échelle d'une hypothèse n'est pas toujours la plus adaptée à l'objet sous-jacent. Mais, il faut rappeler que l'échelle choisie pour une hypothèse, comme elle est en construction, est celle qui minimise l'erreur en fonction des deux courbes de Bézier à la base du motif caractéristique et non pas celle qui minimise l'erreur vis à vis de l'objet feuille en entier.

FIG. 5.8 – Génération d'hypothèses – hypothèses générées. Les courbes blanches représentent les hypothèses générées, les courbes grises représentent les courbes de Bézier issues de l'identification.

Le problème majeur de la génération d'hypothèse est la définition du motif caractéristique. En effet, ce point est la base de la recherche d'un modèle pouvant correspondre et, aussi, est la base de la position, orientation et échelle de ce premier jet d'hypothèse. De plus, il est intéressant de noter que dans l'étape de génération d'hypothèses, nous ne remettons pas en cause l'orientation du modèle, ce qui induit des faux positifs et, donc, des cas que nous n'explorerons pas. L'orientation pourrait être recalculée en déterminant les vecteurs d'écart entre les courbes images et celles du modèle mais pour cela il faut avoir des courbes image suffisamment longues pour être représentatives de l'objet. Nous pouvons, aussi, remarquer que le motif caractéristique que nous avons défini est valable dans le cas que nous avons considéré pendant cette thèse, *i.e.* les feuilles oblongues. De ce fait, la reconnaissance de feuilles rondes, par exemple, n'est pas aisément faisable avec celui-ci, ce qui induit le besoin d'un autre type de motif caractéristique dont la définition n'est pas forcement évidente. Les feuilles rondes sont un cas particulier, car dans la littérature, le choix du motif caractéristique, dans le cas du traitement d'images contenant des objets manufacturés, est souvent un coin. Une autre possibilité pour la définition d'un type de motif caractéristique revient à interpréter les rapports d'un ensemble d'éléments extraits de l'image dans l'espace. Cette approche permet de passer au delà de l'aspect local à l'objet, que peut avoir tel ou tel élément, pour passer à un aspect local à l'image ou global à l'objet. Ce type d'approche est utilisé, par exemple, par [Low87], et est nommée "perceptual grouping". Il permet, dans le cas de [Low87], de grouper certains éléments entre eux selon leur proximité et connexité, et il utilise cette information comme motif caractéristique qu'il va rechercher dans ses modèles. Cette recherche de rapports entre éléments, s'apparente à la reconnaissance de forme elle-même dans le cas où nous manipulons des objets organiques (sous-entendant des algorithmes gérant cette variabilité de forme) et suppose que les rapports recherchés sont définis à l'avance (sous-entendant une sorte de modèle). Cette approche, ainsi posée, pose un problème de fond car nous sommes revenus à notre problème initial. De cette observation, il peut être intéressant de considérer le problème initial en sous-

problèmes eux-mêmes décomposables en sous-problèmes, comme en intelligence artificielle. Nous pouvons, ainsi, considérer des modèles de rapport entre éléments définis pour traiter les éléments les plus basiques (ceux extraits de l'image) puis une autre série de modèles établissant des rapports possibles entre les rapports obtenus précédemment. Et, ainsi de suite jusqu'à ce que l'agencement des différents sous$^n$-modèles arrivent à définir un modèle de forme assimilable à, par exemple, un objet feuille. Cette représentation, rappelle, là encore, la représentation multi-échelle qui peut permettre de définir des modèles pour chaque niveau de détail et permettre la reconnaissance d'objets composés (*i.e.* pointe de feuille / limbe, feuille, plante, rangée de plantes, champs). Cette suggestion de traitement de l'information peut nécessiter un type de modélisation spécifique pour chaque niveau hiérarchique de modèle. Ce choix n'a pas été pris pendant cette thèse du fait de la difficulté de mise en œuvre.

La génération d'hypothèses est, pour sa part, quasi instantanée bien que la complexité est d'ordre $O(n^2)$. Cela du fait de l'utilisation du graphe du distance et d'une fonction qui nous permet de vérifier si deux courbes, de ce graphe, peuvent former un motif caractéristique valide.

#### 5.2.3.2 Renforcement d'hypothèses

Les hypothèses générées par l'étape précédente sont renforcées à cette étape de manière à infirmer ou confirmer que l'objet est bien du type de celui que nous recherchons. Pour ce faire, une certaine précision sur le premier jet d'hypothèses est nécessaire pour que le procédé ne diverge pas. Aussi, pour éviter toute divergence de notre modèle, notre procédé progresse de manière incrémentale et les variations des paramètres définissant les transformations appliquées au modèle sont limitées.

Il est aussi intéressant de noter que la méthode que nous avons mise en place, pour renforcer une hypothèse, n'exploite que les informations de contour mais pas celles des autres motifs caractéristiques disponibles. En effet, notre méthode cherche à aligner les courbes du modèle avec celles existant dans l'image en se basant, lors de la génération de l'hypothèse, sur un seul motif caractéristique. Mais l'étape du renforcement ne prend pas en compte les autres motifs caractéristiques pour améliorer cet alignement. Cette prise en compte pourra faire partie des améliorations à apporter à l'algorithme actuel. Mais, cette seule information de motif caractéristique ne permet pas de caler correctement un modèle, car dans le cas où plusieurs motifs caractéristiques potentiels sont proches, le choix du bon point caractéristique dépend aussi de l'alignement des courbes sous-jacentes (modèle / image).

La méthode actuellement développée n'exploite pas le parcours ordonnée de l'arbre que nous présentons à la section 4.2.4, car nous n'avons pas encore pu implanter ces optimisation de recherches dans l'arbre. De ce fait, des hypothèses quasi similaires sont générées à partir de différents sous-ensembles et nous générons trop d'hypothèses dont beaucoup de doublons. Mais cette première approche nous permet d'illustrer sur la figure 5.9 que le procédé est capable de regrouper des portions de contour appartenant au même objet sous la même hypothèse et de donner une idée de l'objet sous-jacent.

La méthode que nous utilisons pour le moment est très gourmande en ressources. À chaque étape de récursion la complexité est d'ordre de $O(m.n^2)$, où $m$ est le nombre d'hypothèses et $n$ est le nombre moyen de courbes que nous allons essayer d'intégrer par hypothèses, sachant qu'à la fin de chaque étape de récursion le nombre $m$ d'hypothèses augmente.

La figure 5.9 illustre les résultats obtenus pour une feuille donnée à partir d'une portion d'image et la figure 5.10 montre toutes les hypothèses trouvées dans cette portion d'image.

Sur la figure 5.10 nous pouvons voir que les hypothèses générées permettent d'extraire les objets partiellement occultés.

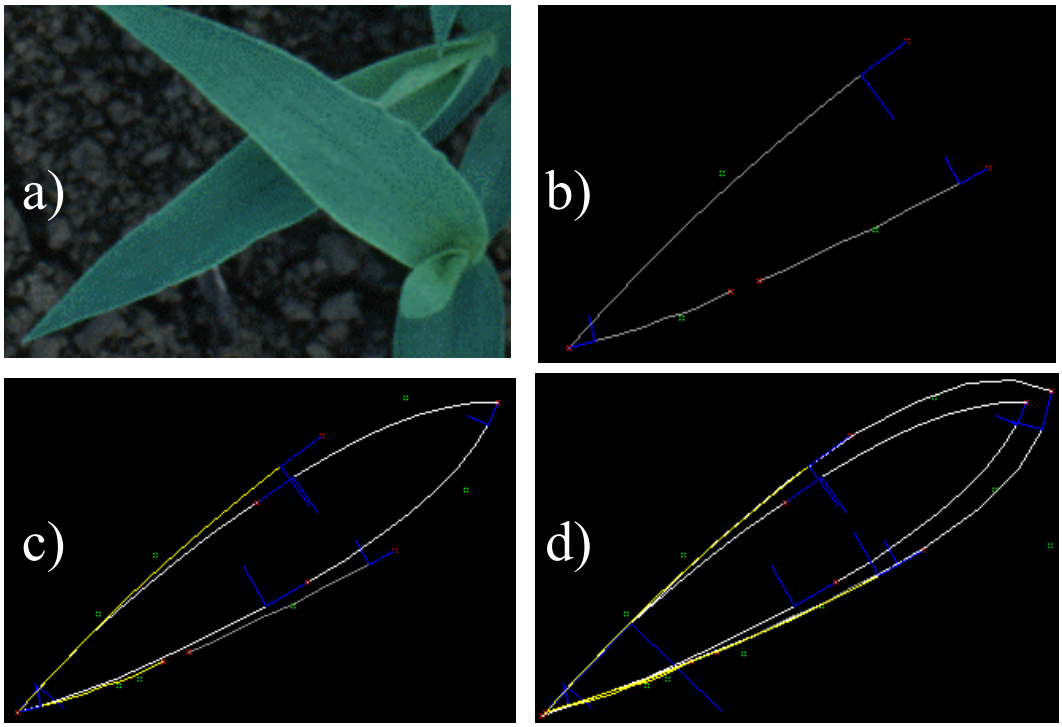



FIG. 5.9 – Renforcement d'hypothèses – exemple pour une feuille. La figure *a* montre l'image d'origine, la *b* les courbes extraites, la *c* l'hypothèse générée et la *d* les hypothèses renforcées.

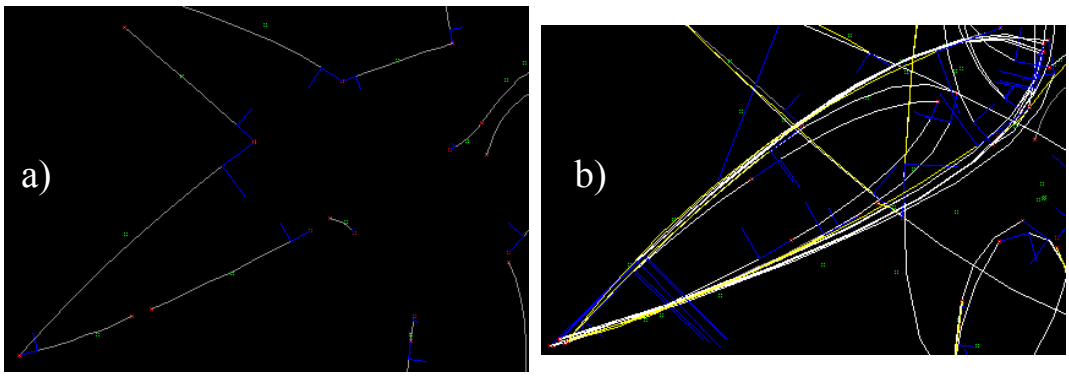



FIG. 5.10 – Renforcement d'hypothèses – exemple complet pour la même portion. La figure *a* montre toutes les courbes extraites et la *b* présente toutes les hypothèses générées.

### 5.2.4 Illustration par l'exemple

Dans cette section nous allons illustrer notre méthode par les résultats intermédiaires obtenus avec trois images réelles. Nous donnerons, par étape, les pourcentages de réussite déterminés manuellement et les commenterons.

Dans le but d'isoler les problèmes, la troisième image est la même que la deuxième sauf que le fond (terre, cailloux, *etc.*) a été isolé et qu'il ne reste que les feuilles.

Les résultats de l'étape de segmentation sont donnés dans le tableau 5.1. Les pourcentages de réussite illustrent le fait que les feuilles superposées ou trop juxtaposées à d'autres ne sont pas visibles par notre méthode. Aussi, les feuilles trop sombres peuvent être "absorbées" par des régions représentant de la terre.

| Nom de l'image | # feuilles observées | # mal ou non segmentées | % de réussite |
|---|---|---|---|
| st16 | 23 | 4 | 86% (19 sur 23) |
| st44 | 34 | 13 | 61% (21 sur 34) |
| st44 masquée | 34 | 12 | 64% (22 sur 34) |

TAB. 5.1 – Résultats obtenus à la fin de l'étape de segmentation.

Les résultats de la vectorisation, de l'appariement et de la recherche de motifs caractéristiques sont donnés par le tableau 5.2. Ces résultats illustrent bien le besoin d'améliorer la méthode de vectorisation car certains cas d'hypothèses sont refusés à cause d'erreurs générées par cette méthode. Paradoxalement, les résultats obtenus avec la troisième image sont moins bons que ceux de la deuxième image. Cela provient du fait que la vectorisation peut générer des erreurs aux extrémités de feuilles. Plus précisément le contour est respecté mais la position des points de contrôles n'est pas optimale, en partie, car la vectorisation n'exploite pas de notion d'échelle du pixel. Cela entraîne alors la définition erronée des tangentes aux extrémités. Aussi, comme le contour est "net" et continu, il n'y a pas d'autre courbe sur ce contour pouvant servir de support à la génération d'autres motifs caractéristiques. En effet, dans le cas de la deuxième image les extrémités des feuilles ne sont pratiquement pas représentées par des courbes et donc, les motifs caractéristiques sont déduits à partir des portions de contours et ainsi, mieux représentés. L'appariement nous aide beaucoup dans cette étape car il nous permet de recréer des contours de feuilles plus représentatifs. Néanmoins, il est aussi fréquent qu'il regroupe des portions qui ne le devraient pas et cela nous fait perdre les courbes sous jacentes qui ont été fusionnées et qui auraient pu être riches en informations sur les contours des feuilles.

| Nom de l'image | # représentées par une hypo | % de réussite |
|---|---|---|
| st16 | 12 | 63% (12 sur 19) |
| st44 | 12 | 57% (12 sur 21) |
| st44 masquée | 10 | 45% (10 sur 22) |

TAB. 5.2 – Résultats obtenus à la fin de l'étape de génération d'hypothèses.

Le tableau 5.3 illustre les résultats obtenus par l'étape de renforcement. Le choix d'une hypothèse considérée comme bien placée pour notre comptage est défini par les courbes image intégrées dans cette hypothèse et par son échelle, orientation et placement. Le faible taux d'échec montre que la pertinence de l'étape précédente est déterminante pour la réussite du renforcement, en effet, toutes les hypothèses convenablement générées ont été correctement renforcées.

| Nom de l'image | # hypo bien placées | % de réussite de l'étape | versus segmentation | versus feuilles comptées |
|---|---|---|---|---|
| st16 | 12 | 100% (12 sur 12) | 63% (12 sur 19) | 52% (12 sur 23) |
| st44 | 10 | 83% (10 sur 12) | 47% (10 sur 21) | 30% (10 sur 34) |
| st44 masquée | 8 | 80% (8 sur 10) | 36% (8 sur 22) | 24% (8 sur 34) |

TAB. 5.3 – Résultats obtenus à la fin de l'étape de renforcement d'hypothèses.

FIG. 5.11 – Image st16.

FIG. 5.12 – Image st16, résultat de segmentation et visualisation des régions prises en compte.

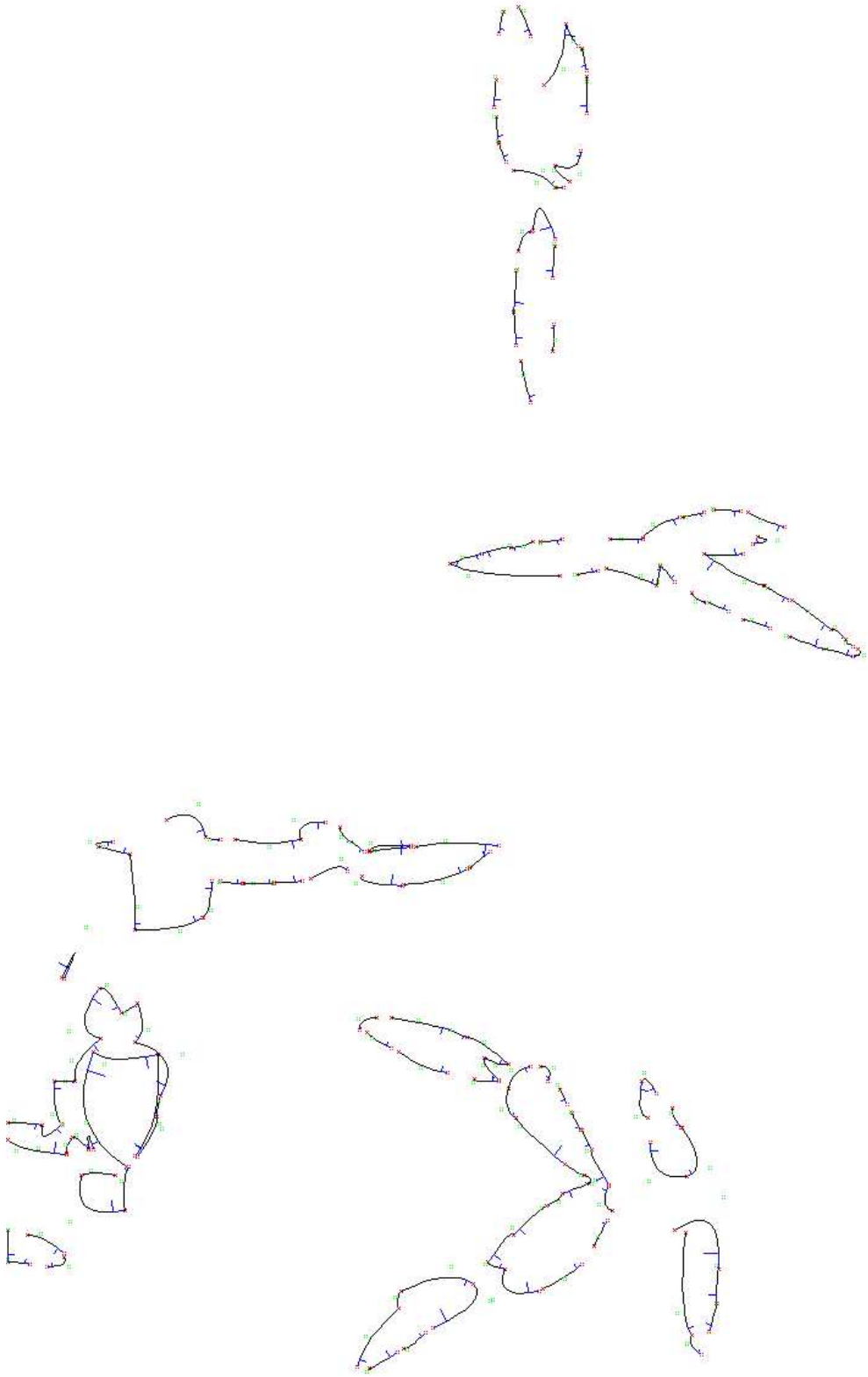


FIG. 5.13 – Image st16, résultat de vectorisation.

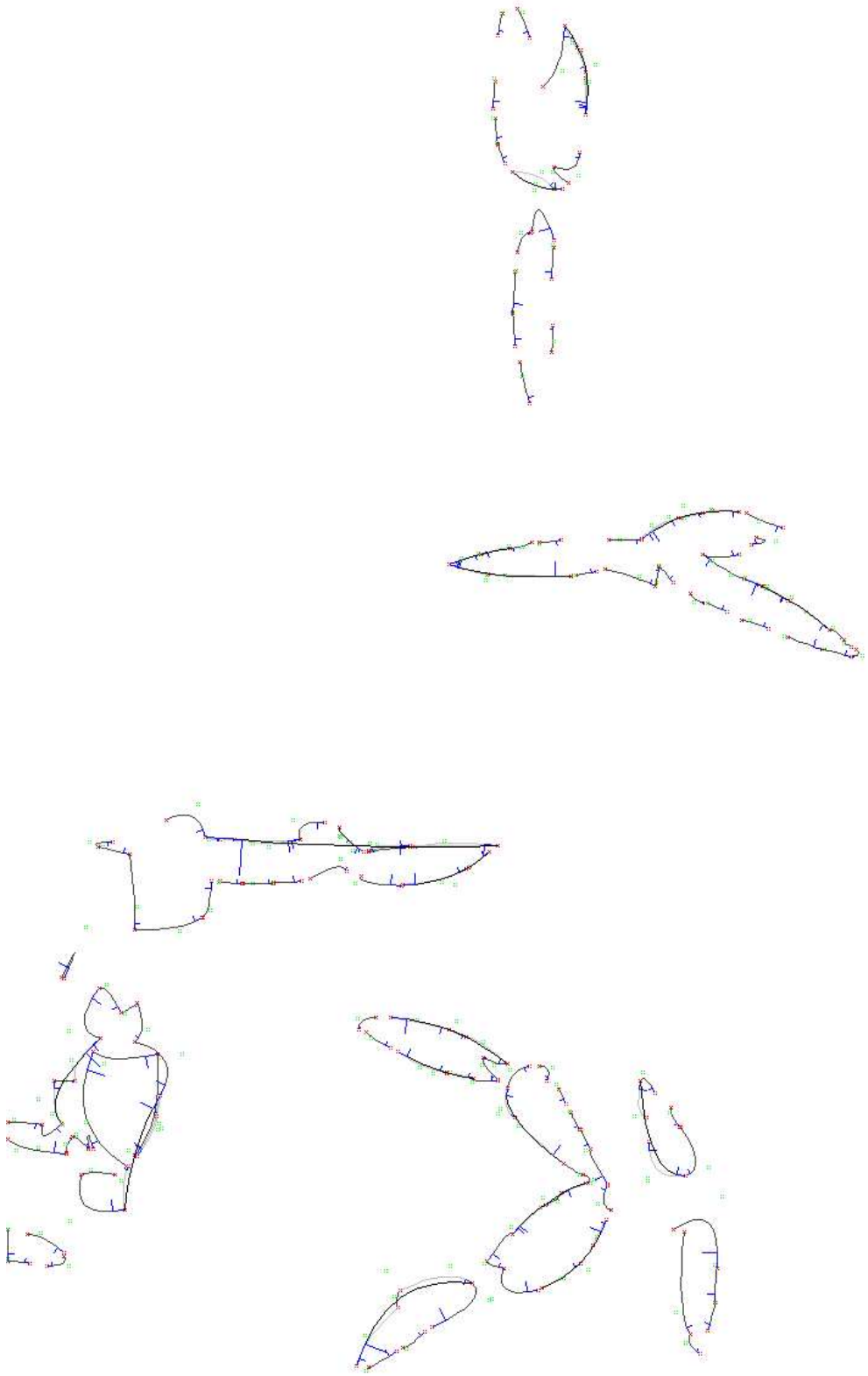


FIG. 5.14 – Image st16, résultat d'appariement.

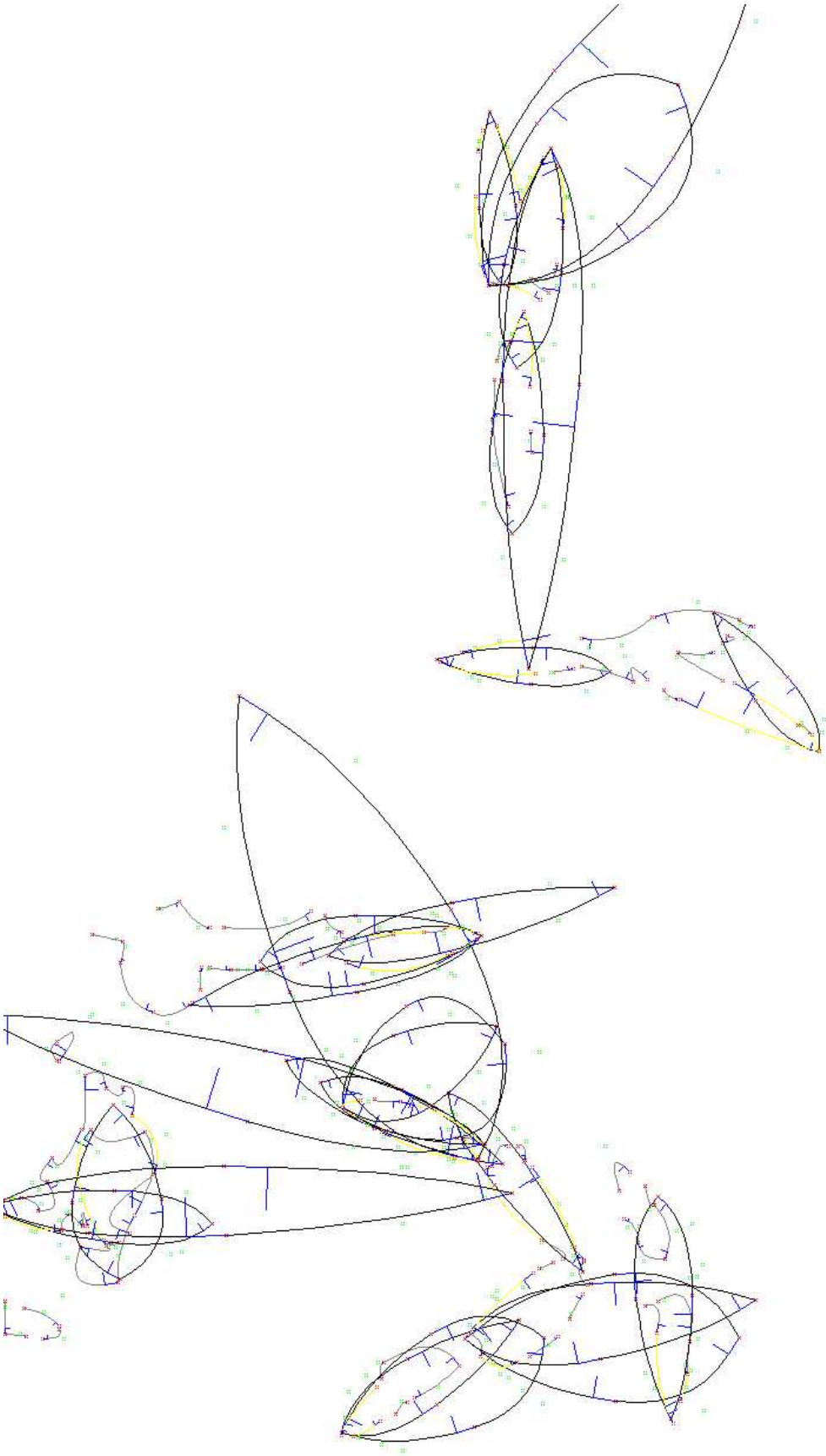


FIG. 5.15 – Image st16, résultat de génération d'hypothèses.

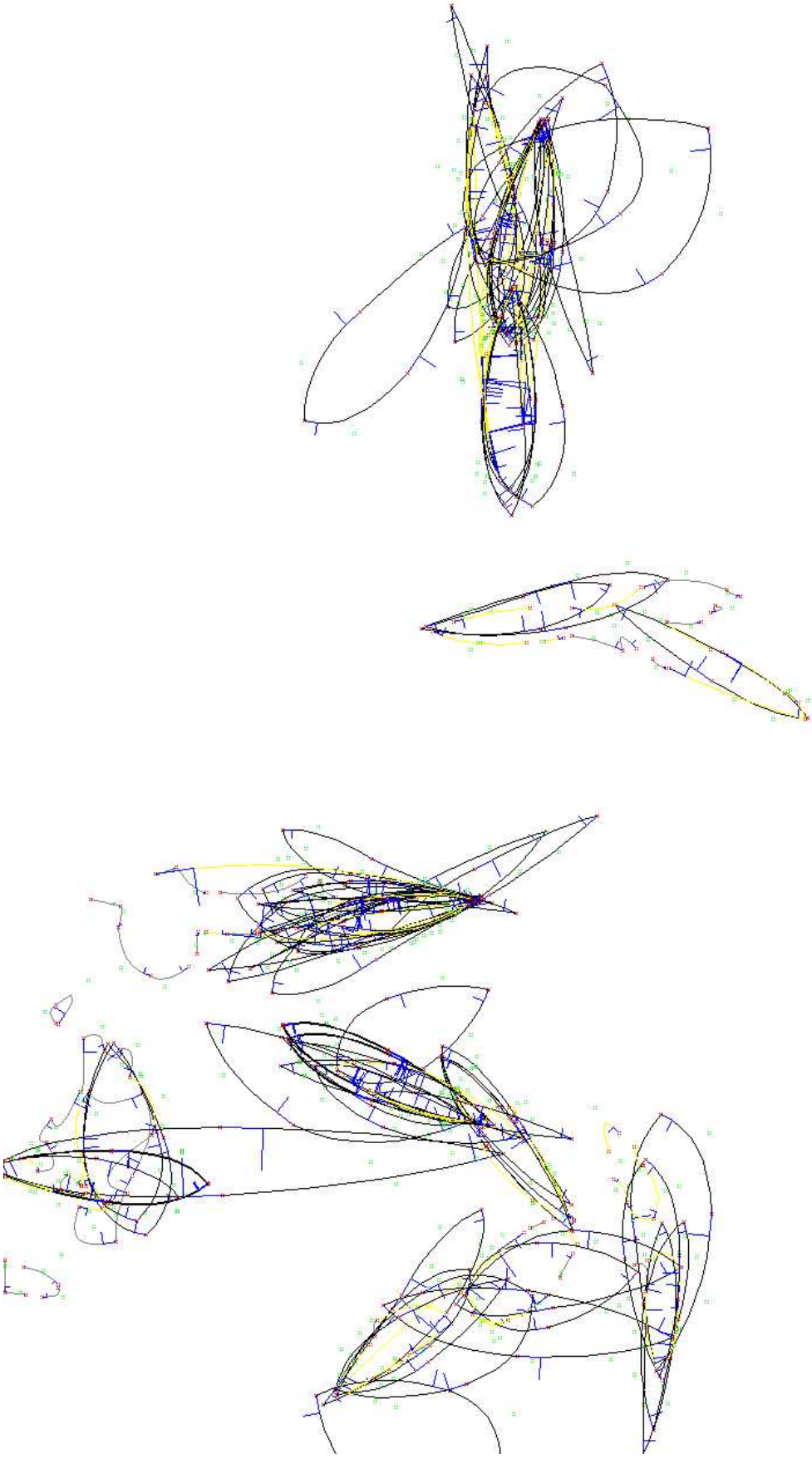


FIG. 5.16 – Image st16, résultat de renforcement d'hypothèses.

FIG. 5.17 – Image st44.

FIG. 5.18 – Image st44, résultat de segmentation et visualisation des régions prises en compte.

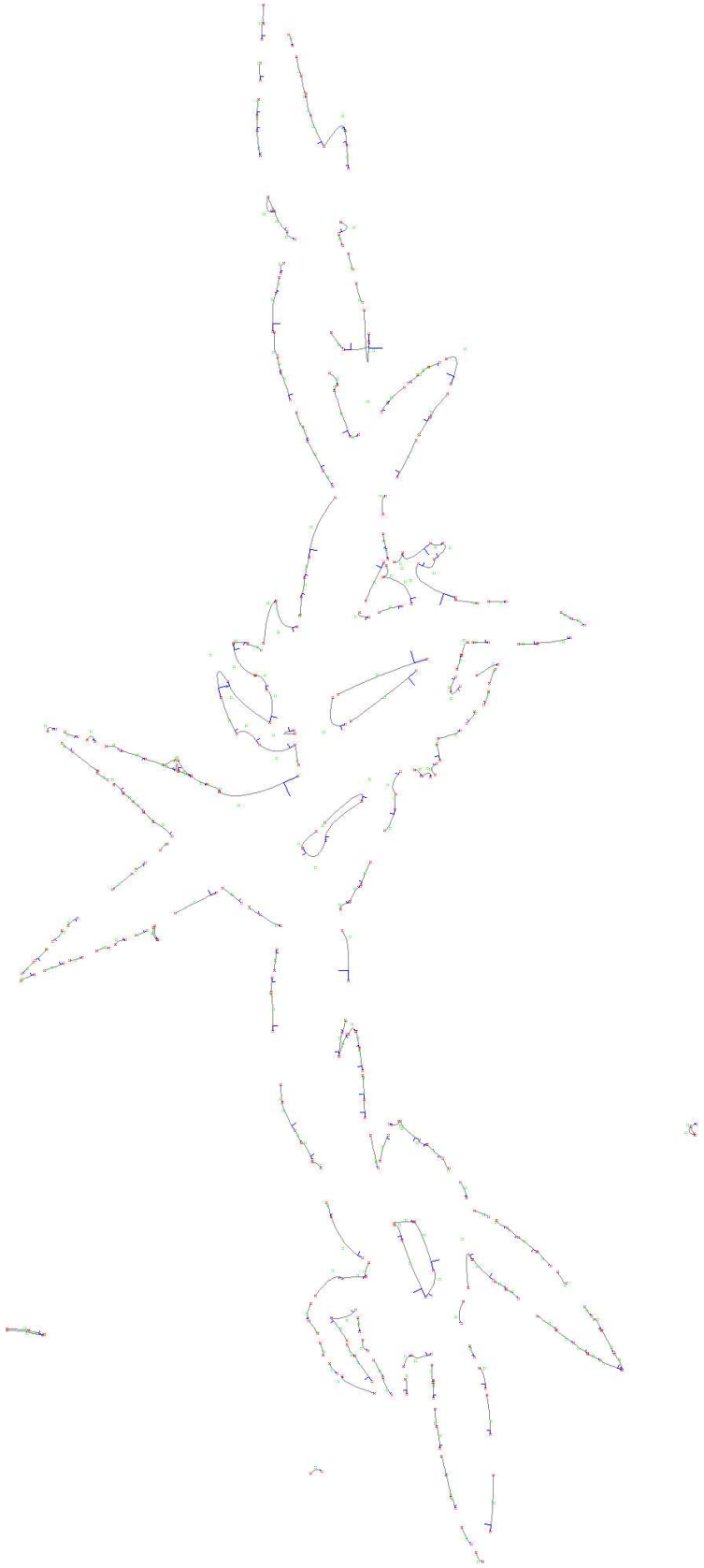


FIG. 5.19 – Image st44, résultat de vectorisation.

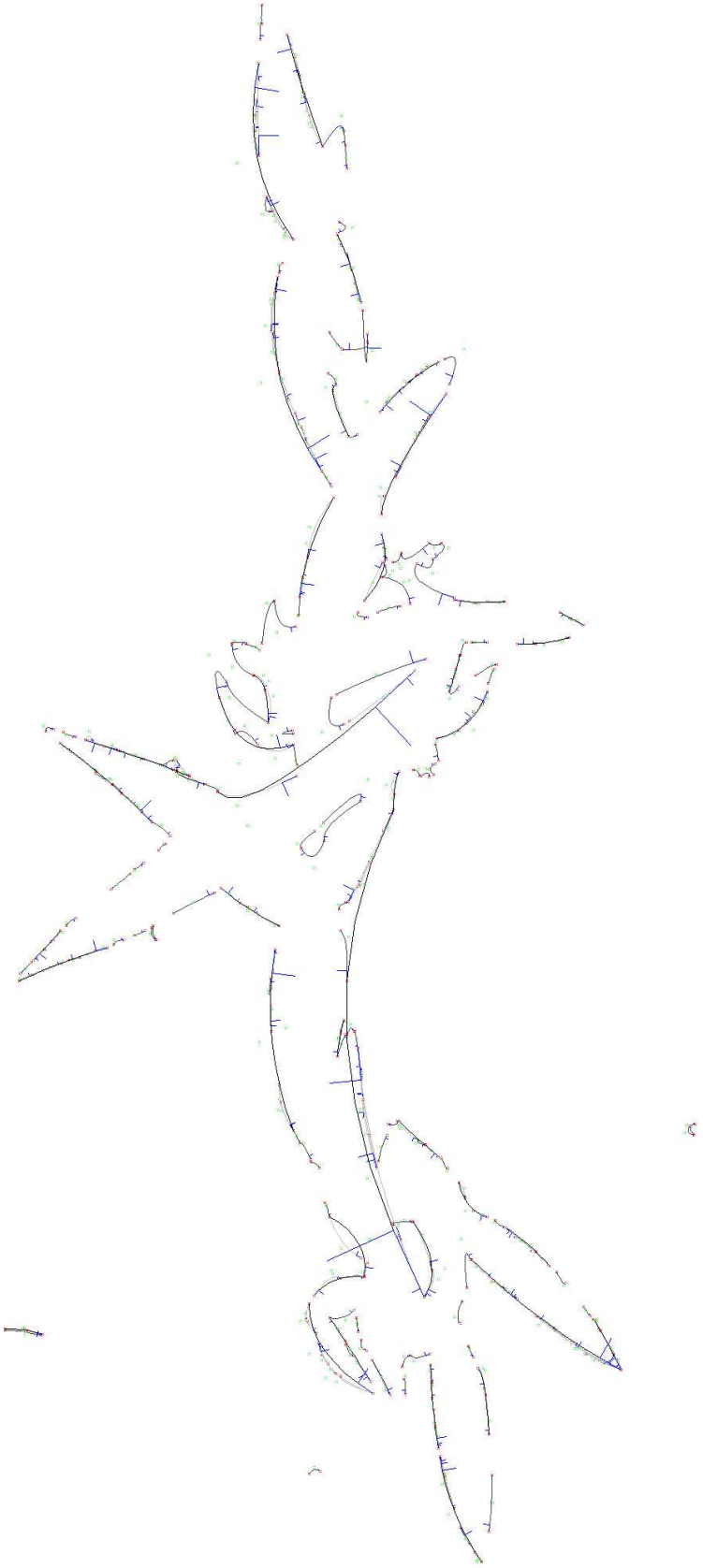


FIG. 5.20 – Image st44, résultat d'appariement.

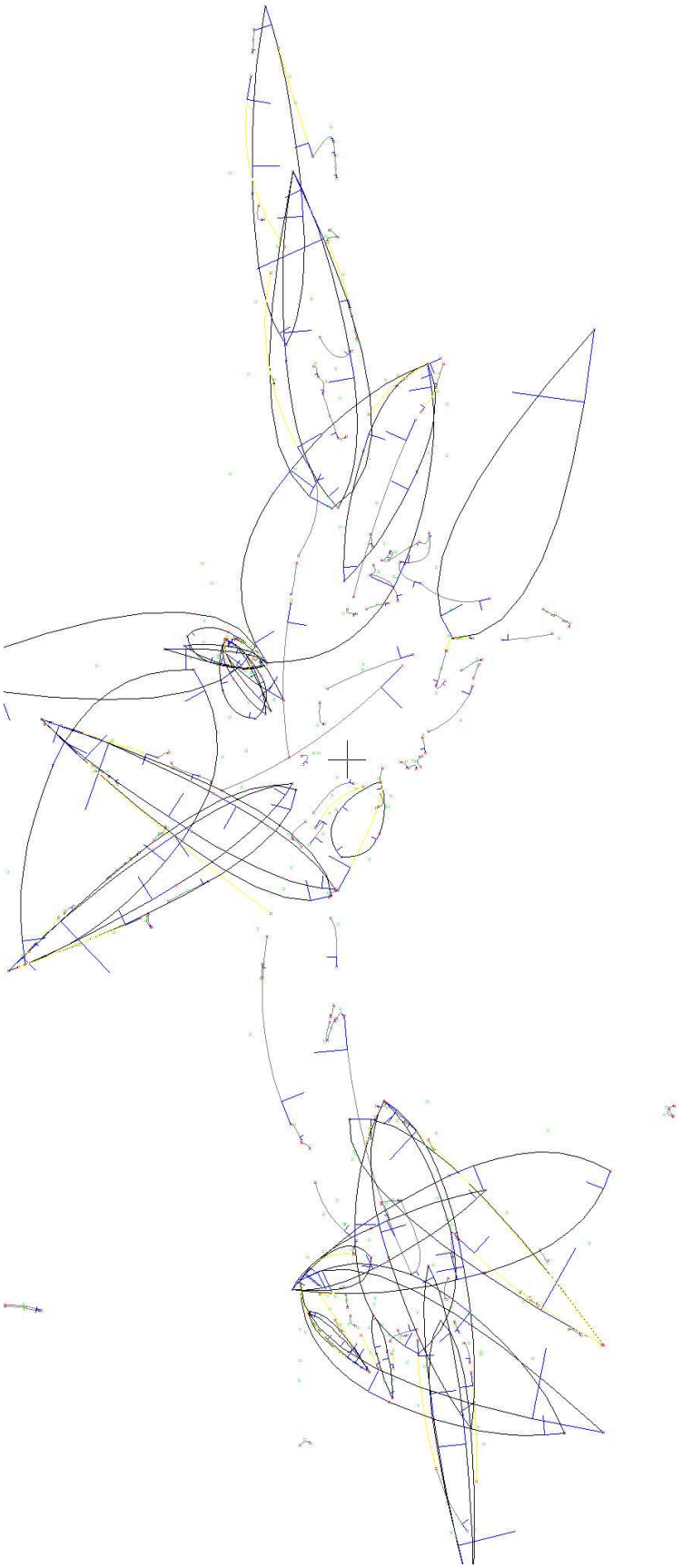


FIG. 5.21 – Image st44, résultat de génération d'hypothèses.

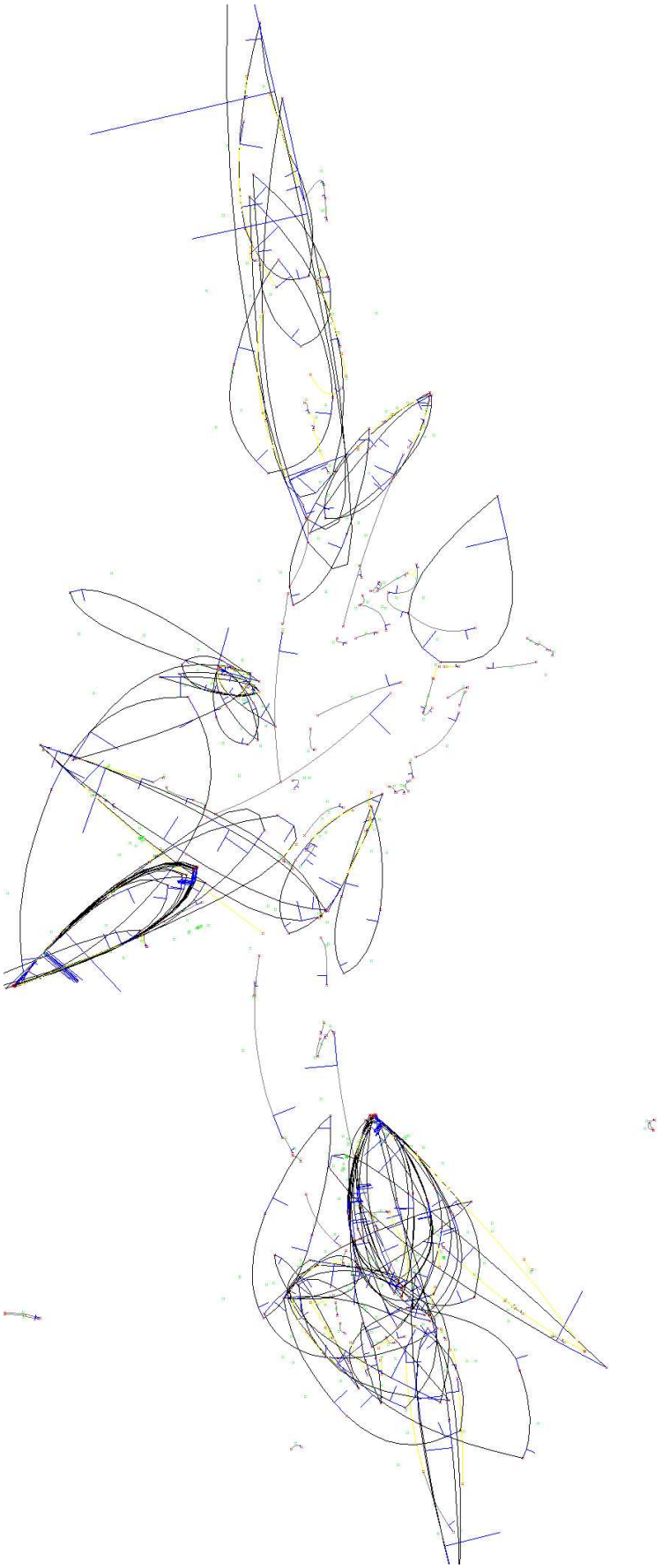


FIG. 5.22 – Image st44, résultat de renforcement d'hypothèses.

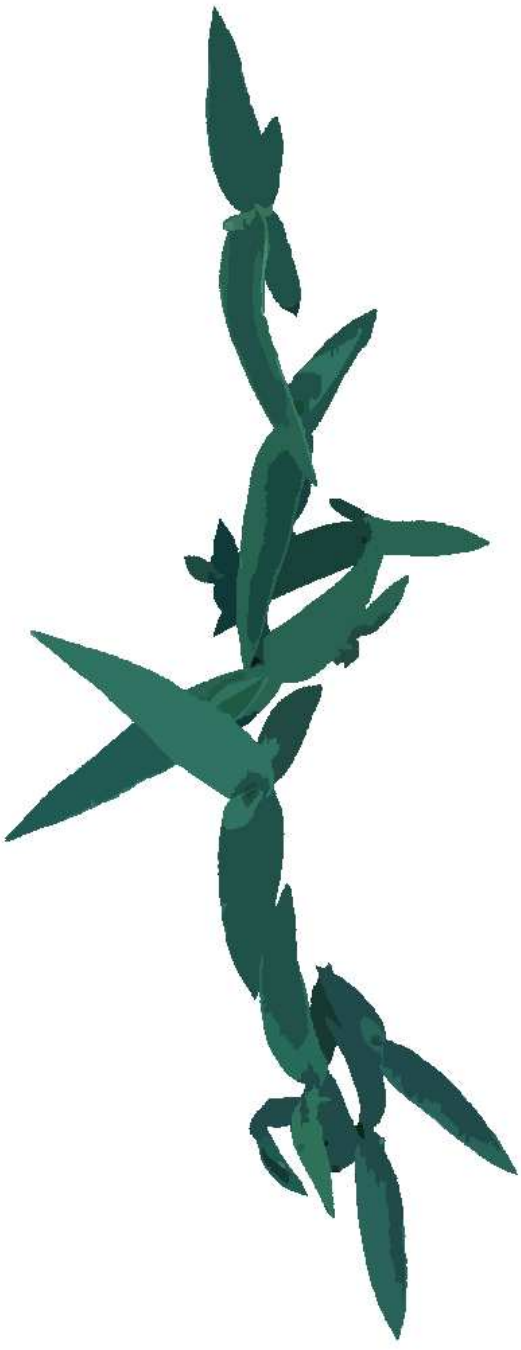


FIG. 5.23 – Image st44 masquée, résultat de segmentation.

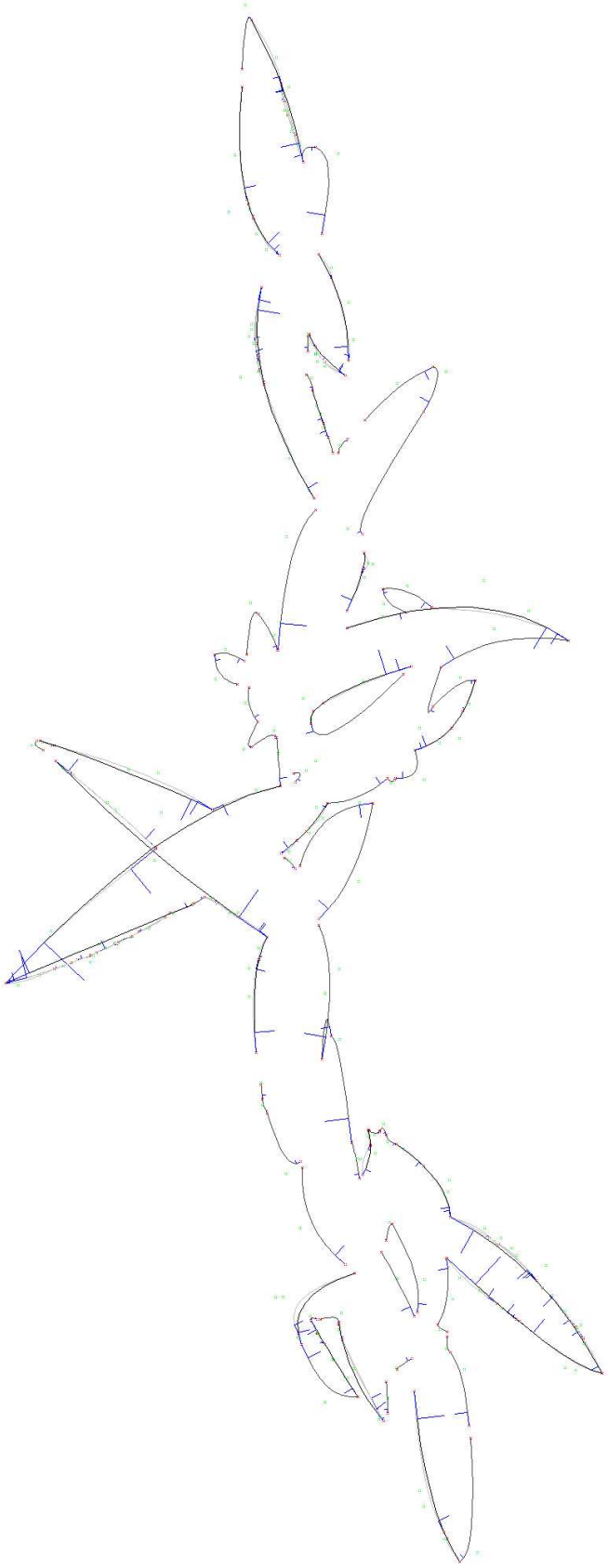


FIG. 5.24 – Image st44 masquée, résultat d'appariement.

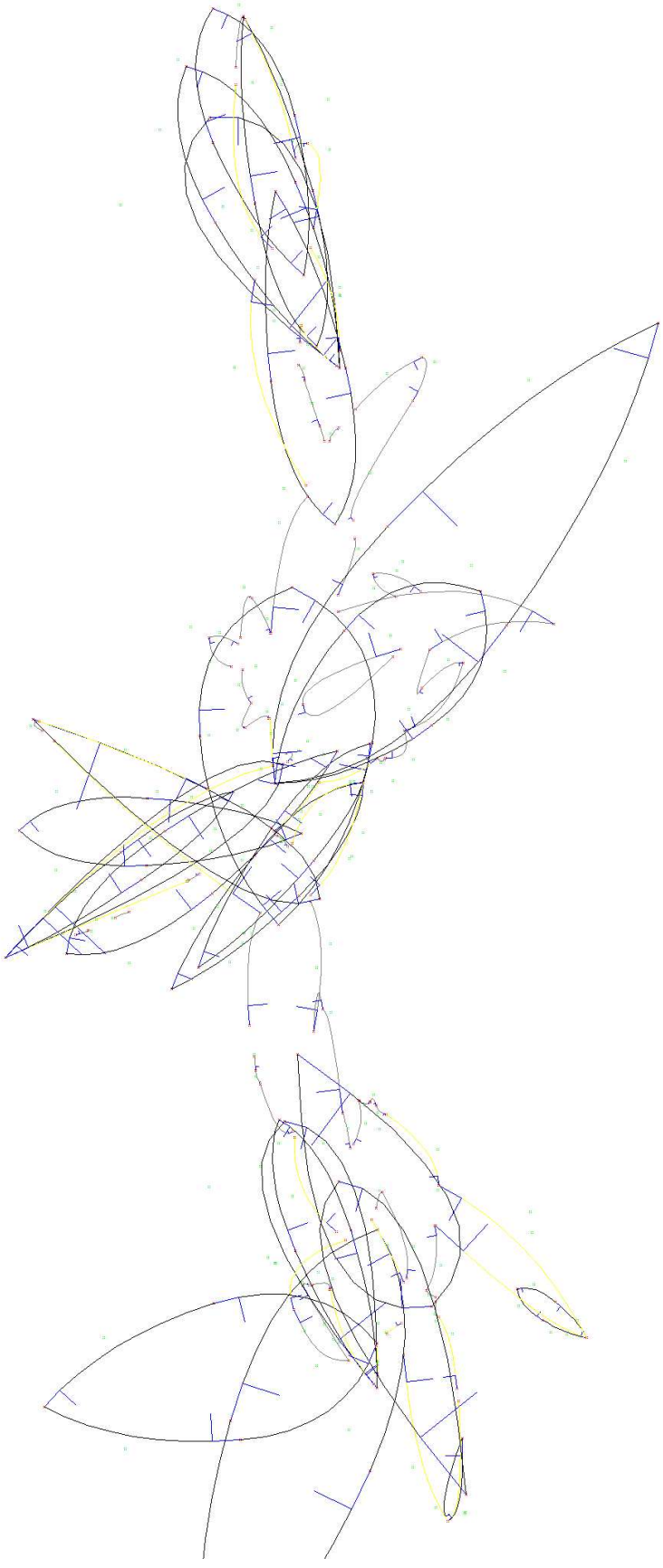


FIG. 5.25 – Image st44 masquée, résultat de génération d'hypothèses.

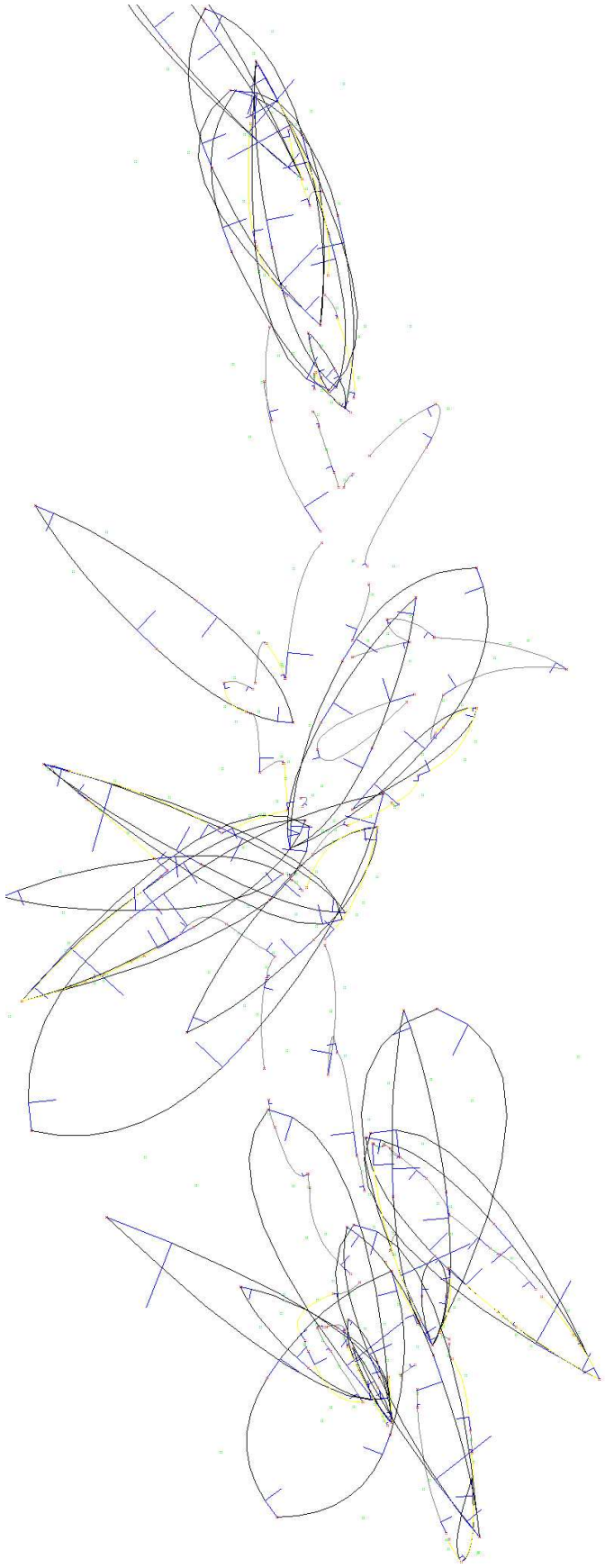


FIG. 5.26 – Image st44 masquée, résultat de renforcement d'hypothèses.

## 5.3 Bilan

Notre procédé d'extraction de feuilles commence par l'exploitation des données brutes issues de l'image. Et, étape par étape, il construit, au mieux et en fonction des résultats des étapes précédentes, des ensembles d'informations les plus valides possibles, tout en conservant une certaine incertitude, nécessaires à la reconnaissance des objets feuille.

La pertinence des résultats obtenus par l'étape de segmentation est la plus cohérente par rapport aux limitations de la méthode elle-même. Plus précisément, comme nous n'exploitons pas une information résultant d'un algorithme de détection de contours, il nous est impossible d'extraire les bords d'objets se chevauchant ou se superposant entièrement. Par contre, les défauts observés dans les résultats obtenus avec la méthode de vectorisation en courbes de Bézier sont surtout liés à des problèmes d'implantation (algorithmique et précision numérique) et à la recherche, ajout et la gestion des cas particuliers pour améliorer les résultats. Néanmoins, la vectorisation donne de bons résultats dans la plupart des cas et nous permet de définir un nombre conséquent de motifs caractéristiques exploitables. La génération d'hypothèses pêche principalement par le manque de précision et d'information des motifs caractéristiques, induisant par là même des erreurs de rotations et de calculs d'échelles, mais elle nous permet de localiser bon nombre de feuilles extractibles. La dernière étape peut sembler inefficace par le nombre de solutions qu'elle génère et la pertinence des résultats, mais cela est principalement dû à des hypothèses, trop mal placées ou incorrectes, qui ont été détectées par l'étape précédente. Aussi, du fait que notre modèle va rarement correspondre à une feuille réelle de l'image, nous sommes contraints d'être plus tolérants en regard des erreurs rencontrées et cela nous génère des cas impossibles.

Un certain nombre d'améliorations ont été proposées afin de parvenir à supprimer une partie des défauts relevés. L'intégralité du procédé a été développée en *C++* sous un environnement de gestion de fenêtres : *Traitim* (cf. annexe F). Ce langage nous a permis de construire une application modulaire décomposant chaque notion abordée en une structure facile à manipuler et à faire évoluer.

# 6.
# Conclusion et perspectives

*Prendre une décision, c'est transformer le doute en incertitude.*

Dans le cadre de la réduction de l'épandage des herbicides dans des parcelles cultivées, nous avons développé une approche basée sur la vision numérique. Cette approche exploite des photographies prises dans ces parcelles pour déterminer la distribution spatiale et les espèces des plantes en présence. Pour atteindre cet objectif, nous cherchons à extraire les feuilles des plantes présentes afin d'évaluer leur espèce. Les principaux problèmes que nous avons dû prendre en compte pour traiter ce problème sont la grande variabilité de forme des objets recherchés et les conditions extérieures qui peuvent altérer ces objets. Nous avons alors introduit une connaissance *a priori* sur les feuilles (*i.e.* nos modèles de formes) afin de faciliter cette recherche dans notre traitement. Le procédé mis en œuvre exploite principalement les contours extraits des objets présents dans l'image, parmi lesquels nous recherchons des correspondances possibles avec notre connaissance *a priori*.

Notre procédé se déroule en plusieurs étapes dont les plus importantes sont la segmentation couleur, l'extraction et la vectorisation des contours en courbes de Bézier, la génération et le renforcement d'hypothèses. Ces différents étapes appliquées successivement à l'information issue de l'image ou à l'information issue de transformations induisent une erreur qui se cumule au fur et à mesure. Cette erreur entraîne, de ce fait, une incertitude qu'il est important de prendre en compte afin d'éviter des prises de décisions hâtives. Dans le cadre de notre étude le choix de la logique floue a été pris du fait de la mise en œuvre rapide, la définition possible de règles et des possibilités d'extension, bien que nous n'utilisons pas d'inférence floue dans notre procédé. Ainsi, la fiabilité de cette procédure dépend de la robustesse des algorithmes employés et de la pertinence des résultats intermédiaires obtenus.

Les résultats que nous pouvons obtenus, nous permettent d'extraire une partie des feuilles contenues dans l'image, ainsi que les feuilles se chevauchant ou celles partiellement dégradées ou encore les feuilles déformées. Néanmoins, certaines configurations de feuilles, de part leur position ou chevauchement, peuvent être mal extraites ou induire des erreurs lors du traitement d'autres feuilles voisines. Pour l'heure, nous avons surtout cherché à jeter les bases d'un procédé se déroulant suivant un enchaînement d'étapes cohérentes dont chacune pourrait être améliorée.

La méthode de segmentation développée permet d'obtenir des régions représentatives des objets feuilles dans la plupart des cas, même si l'information extraite en bord d'objet ou au niveau d'un dégradé manque parfois de précision. Comme cette étape est la base du procédé, elle se doit d'être améliorée pour permettre d'obtenir, globalement, de bons résultats. Le choix d'un espace couleur plus approprié et la collaboration contours-régions sont les points les plus importants à travailler.

La vectorisation bien qu'elle permette d'obtenir des résultats honorables peut, aussi, faillir dans certains cas et nécessite d'être améliorée pour optimiser les résultats de la génération d'hypothèses. Nous pouvons, par exemple, exploiter des procédés non paramétrés afin de déterminer les meilleurs points de contrôle sur des courbes issues d'objets de types différents.

Un des désavantages de la modélisation employée est qu'elle n'exploite qu'une échelle de représentation pour un modèle de forme, ce qui peut être limitant dans les objets que nous souhaiterions représenter. Actuellement, il nous est, par exemple, impossible de différencier une feuille d'ortie d'une feuille de ficus. Une utilisation multi-échelle du modèle pourrait résoudre ce problème, car cela permettrait de représenter différents niveaux de détail ("coarse to fine") pouvant être exploités selon l'échelle à laquelle est observée l'hypothèse. Mais, la modélisation par courbes de Bézier ne serait peut être plus adaptée. La représentation multi-échelles semble être intuitive lorsque nous regardons comment un mammifère supérieur reconnaît un objet lointain se rapprochant. En effet, à partir de cette forme vague, il nous est possible d'émettre un certain nombre d'hypothèses quant à la nature de l'objet en question. Hypothèses se précisant au fur et à mesure que l'objet se rapproche et qu'il est possible d'identifier des détails privilégiant telle ou telle hypothèse. Néanmoins, cette approche n'a pas été intégrée dans notre travail et peut nécessiter des images avec une résolution plus fine mais elle offre un bon nombres de perspectives d'améliorations.

La dernière étape, celle de la génération et du renforcement d'hypothèses, peut aussi être améliorée,

aussi bien dans la définition et l'intégration des motifs caractéristiques, par exemple *via* une représentation multi-échelle, que dans les paramètres pris en compte lors de l'étape de minimisation. Aussi, la déformation n'est actuellement appliquée que dans le cas d'une feuille oblongue au squelette très simple, alors qu'il faudrait pouvoir gérer la déformation sur les différentes parties articulées d'un squelette.

Actuellement, du fait de l'utilisation d'une méthode de renforcement non optimisée, le temps de calcul, dont 90% peut être impliqué à cette dernière étape, est très important : de l'ordre de la quinzaine de minutes. Alors que ce même temps pourrait être réduit à quelques minutes en utilisant le procédé optimisé.

L'analyse du travail accompli par rapport aux méthodologies existantes, montre une certaine similitude avec des procédés déjà éprouvés, comme, par exemple, le travail effectué par [Low87] ou bien [Pop94]. L'élaboration d'une série de tests, sur la base de l'étude bibliographique, pour déterminer les meilleures méthodes à employer pour chacune de ces étapes aurait sûrement pu améliorer la qualité des résultats. Néanmoins, l'originalité du travail accompli durant cette thèse repose, principalement, sur l'exploitation d'images de scènes naturelles et complexes, sur l'utilisation des courbes de Bézier de degré 2 associées à la génération et au renforcement d'hypothèses et enfin, sur l'agencement de ces étapes. Ce type de méthode, basée sur cet agencement, est un moyen efficace pour résoudre ce problème, car il permet de réduire la complexité de part l'aspect incrémental et d'éviter des décisions trop affirmées de part la gestion des hypothèses et du flou.

# Notations

| | |
|---|---|
| $bez$ | Courbe de Bézier de degré 2 |
| $A$, $B$, $C$ | Points de contrôle pour une courbe de Bézier de degré 2 |
| $D$ | Raccourci de notation pour $D = {}^{[AC]}/_{2}$ |
| $M_t$ | Point sur une courbe de Bézier de degré 2 en la position $t$, cf. équation 3.1, p. 19 |
| $bez_3 = app(bez_1, bez2)$ | opération d'appariement de deux courbes ($bez_1$ et $bez2$) en une seule $bez_3$ (cf. section 3.2.2) |
| $length(bez)$ | Retourne la longueur d'une courbe de Bézier (cf. section 3.1.2) |
| $\cap_{obj}$ | Intersection de deux objets $obj$, où $obj$ peut être courbes, droites, segments |
| $\tan_{bez}^{pt}$ | Tangente au point $pt$ à la courbe $bez$ |
| $\perp_{obj}^{p}$ | Projection perpendiculaire du point $p$ sur l'objet $obj$ |
| $nom = FuzzComp(V, d)$ | Comparateur flou nommé $nom$ définit selon une valeur $V$ et un décalage $d$ autour de cette valeur (cf. annexe D) |
| $\overset{\wedge}{}_{\text{genre}}$ ou $\overset{\vee}{}_{\text{genre}}$ | opérateur *et* et *ou* pour des valeurs floues, $genre$ représente un type d'opérateur qui peut être $zadeh$, $prob$ (probabiliste), *etc.*. Cf. annexe D |

# Table des figures

# Bibliographie


[AF86] N.J. Ayache and O.D. Faugeras, *Hyper : A new approach for the recognition and positioning of two-dimensional objects*, PAMI **8** (1986), no. 1, 44–54. 2.2

[AFG01] M.G. Albanesi, M. Ferretti, and F. Guerrini, *A taxonomy for image authentication techniques and its application to the current state of the art*, CIAP01, 2001, pp. 535–540. 2

[Ass98] L. Assémat, *Compétitivité des mauvaises herbes : définitions, limites et perspectives*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 9–16. (document)

[Bas92] Ronen Basri, *Recognition by prototypes*, Tech. Report AIM-1391, The Weizmann Institute of Science, 1992. 2.2

[BB99] A. Baraldi and P. Blonda, *A survey of fuzzy clustering algorithms for pattern recognition - part i*, IEEE Transactions on systems, man and cybernetics. Part B **29** (1999), no. 6, 778–785. D.1

[BCA96] Eric Bardinet, Laurent D. Cohen, and Nicholas Ayache, *Analyzing the deformation of the left ventricle of the heart with a parametric deformable model*, Tech. Report RR-2797, INRIA, 1996. 2.1

[BCFM00] J. Batlle, A. Casals, J. Freixenet, and J. Marti, *A review on strategies for recognizing natural objects in colour images of outdoor scenes*, Image and Vision Computing **18** (2000), no. 6-7, 515–530. 1

[BCGJ95] R. Basri, L. Costa, D. Geiger, and D.W. Jacobs, *Determining the similarity of deformable shapes*, PBMCV95, 1995, p. SESSION 5. 2.1.4

[Ber02] Nicolas Bercher, *Traitement d'images pour la caractérisation d'adventices dans les cultures de maîs*, Tech. report, Cemagref, 2002. 5.2.1

[Big] *Keith price bibliography annotated computer vision bibliography*, http ://iris.usc.edu/Vision-Notes/bibliography/contents.html. A.1.2

[Bis01] S. Biswas, *Contour coding through stretching of discrete circular arcs by affine transformation*, PR **34** (2001), no. 1, 63–77. 2.1.2

[BJ89] George Boolos and Richard Jeffrey, *Computability and logic*, third ed., Oxford University Press, 1989. D.1

[BL99] Jeffrey S. Beis and David G. Lowe, *Indexing without invariants in 3d object recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999), no. 10, 1000–1015. 2.1, 2.2

[Blo99a] I. Bloch, *Fuzzy relative position between objects in image processing : a morphological approach*, IEEE Transaction on Pattern Analysis and Machine Intelligence **21** (1999), no. 7, 657–664. D.1

[Blo99b] _____, *On fuzzy distances and their use in image processing under imprecision*, Pattern Recognition **32** (1999), 1873–1895. D.1

[BM94] I. Bloch and H. Maître, *Fusion de données en traitement d'images : modèles d'information et décisions*, Traitement du signal **11** (1994), no. 6, 435–446. D.1

[BMP02] S. Belongie, J. Malik, and J. Puzicha, *Shape matching and object recognition using shape contexts*, IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002), no. 4, 509–522. 2.1.2, 2.2, 2.1.2, 6

[Bou95] B. Bouchon, *La logique floue et ses applications*, Addison Wesley, New York, 1995. D.1, D.1

[BSA94] R.B. Brown, J.-P. G.A. Steckler, and G.W. Anderson, *Remote sensing for identification of weeds in no-till corn*, Transactions of the ASAE **37** (1994), no. 1, 297–302. (document)

[CC99] T.J. Cham and R. Cipolla, *Automated* B-S*pline curve representation incorporating* MDL *and error-minimizing control point insertion strategies*, IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999), no. 1, 49–53. 3.2.1

[CF01] R.J. Campbell and P.J. Flynn, *A survey of free-form object representation and recognition techniques*, CVIU **81** (2001), no. 2, 166–210. 2

[CGHK99a] B. Carvalho, C.J. Gau, G.T. Herman, and T.Y. Kong, *Algorithms for fuzzy segmentation*, PAA **2** (1999), no. 1, 73–81. A.1.1.8

[CGHK99b] B.M. Carvalho, C.J. Gau, G.T. Herman, and T.Y. Kong, *Algorithms for fuzzy segmentation*, Pattern Analysis and Applications **2** (1999), 73–81. D.1

[Cho96] Maurice Chossat, *Aide-mémoire de mathématiques de l'ingénieur*, Dunod, 1996. 3.1.2

[CHTH93] Timothy F. Cootes, A. Hill, Christopher J. Taylor, and J. Haslam, *The use of active shape models for locating structures in medical images*, IPMI, 1993, pp. 33–47. 2.2

[CJSW00] H. D. Cheng, X. H. Jiang, Y. Sun, and Jingli Wang, *Color image segmentation : advances and prospects*, The Journal Of Pattern Recognition Society **34** (2000), 2259–2281. A.1.2

[CMVm86] F. Cheevasuvit, H. Maitre, and D. Vidal-madjar, *A robust method for picture segmentation based on a split-and-merge procedure*, CVGIP **34** (1986), 268–281. A.1.1.5

[CS95] B. B. Chaudhuri and Nirupam Sarkar, *Texture segmentation using fractal dimension*, IEEE Transactions on Pattern Analysis and Machine **18** (1995), no. 1, 72–77. A.1.1.7

[CV03] A. Clement and B. Vigouroux, *Unsupervised segmentation of scenes containing vegetation (forsythia) and soil by hierarchical analysis of bi-dimensional histograms*, PRL **24** (2003), no. 12, 1951–1957. 5.2.1

[CY98a] H.H. Chang and H. Yan, *Vectorization of hand-drawn image using piecewise cubic bezier curves fitting*, PR **31** (1998), no. 11, 1747–1755. 3.2.1

[CY98b] K.W. Cheung and D.Y. Yeung, *A bayesian framework for deformable pattern recognition with application to handwritten character recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 12, 1382–1388. 2.1.4

[dC93] Paul de Casteljau, *Polar forms for curve and surface modeling as used at citroen*, Fundamental Developments of Computer-Aided Geometric Modeling (Les Piegl, ed.), Academic Press, 1993, pp. 1–12. 3.1

[Dij82] Edsger W. Dijkstra, *Finding the maximum strong components in a directed graph*, Selected Writings on Computing : A Personal Perspective, Springer-Verlag, 1982, pp. 22–30. A.1.1.8

[DLSR01] M.E. De Lorenzo, G.I. Scott, and P.E. Ross, *Toxicity of pesticides to aquatic microorganisms : a review*, Environmental Toxicology and Chemistry **20** (2001), no. 1, 84–98. (document)

[DS93] M. Dunier and A.K. Siwicki, *Effects of pesticides and other organic pollutants in the aquatic environment on immunity of fish : a review*, Fish and Shellfish Immunology **3** (1993), no. 6, 423–438. (document)

[DT02] Jiangwen Deng and H. T. Tsui, *A fast level set method for segmentation of low contrast noisy biomedical images*, Pattern Recognition Letters **23** (2002), 161–169. 2.1.4

[DTJT96] Oivind Due Trier, Anil K. Jain, and Torfinn Taxt, *Feature extraction methods for character recognition-a survey*, Pattern Recognition **29** (1996), no. 4, 641–662. 2.1.2, 2.1.3, 2.1.4

[FBC91] R Falah, P Bolon, and J Cocquerez, *A region-region and region-cooperative approach of image segmentation*, International Conference on Image Processing, vol. 3, 1991, pp. 127–140. A.1.2

[FG96] C. Fiorio and J. Gustedt, *Two linear time union-find strategie for image processing*, In Theoretical Computer Sciences **54** (1996), 165–181. A.2

[FGL00] Christophe Fiorio, Jens Gustedt, and T. Lange, *Union-find volume segmentation*, IWCIA 00 : 7th International Workshop on Combinatorial Image Analysis, Caen, France, 7 2000, pp. 181–197. A.1.1.4

[Fio95] C. Fiorio, *Approche interpixel en analyses d'images, une topologie et des algorithmes de segmentation*, Thèse de doctorat, Université Montpellier II, Montpellier, France, NOV 1995. C

[FWF02] X. Feng, C.K.I. Williams, and S.N. Felderhof, *Combining belief networks and neural networks for scene segmentation*, PAMI **24** (2002), no. 4, 467–483. A.1.1.8

[Gas98] J. Gasquez, *Les graminées adventices résistantes aux herbicides antigraminées en france*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 133–140. (document)

[GLGZ03] J. M. Gonzalez-Linares, N. Guil, and E. L. Zapata, *An efficient 2d deformable objects detection and location algorithm*, Pattern Recognition **36** (2003), no. 11 SU, 2543–2556. 2.1.1, 2.1.2

[GM96] G. Guy and G.G. Medioni, *Inferring global perceptual contours from local features*, IJCV **20** (1996), no. 1/2, 113–133. 2.1.1

[Gor03] N. Gorretta, *Segmentation d'images naturelles par collaboration de méthodes de segmentation régions/contours*, Rapport de dea, SRI, Cemagref, 24 juin 2003. 5.2.1, A.1

[GS00] T. Gevers and A.W.M. Smeulders, *Pictoseek : Combining color and shape invariant features for image retrieval*, IEEE Transactions on Image Processing **9** (2000), no. 1, 102–119. 2.1.2

[GSA01] M. Graymore, F. Stagnitti, and G. Allinson, *Impacts of atrazine in aquatic ecosystems*, Environment International **26** (2001), no. 7-8, 483–495. (document)

[Hea01] I. Heap, *International survey of herbicide-resistant weeds*, 2001. (document)

[HH97] M.W. Hansen and W.E. Higgins, *Relaxation methods for supervised image segmentation*, PAMI **19** (1997), no. 9, 949–962. A.1.1.2

[HH96] AJ Heap and DC Hogg, *Towards 3d hand tracking using a deformable model*, FG96, Second International Conference on Automatic Face and Gesture Recognition, 96. 2.1

[HN99] A. Hausler and H. Nordmeyer, *Characterizing spatial and temporal dynamics of weed seedling populations*, Agricultural Engineering (AgEng99), 1999, pp. 463–472. (document)

[HS85] R. M. Haralick and L. G. Shapiro, *Image segmentation techniques, and image processing*, Computer Graphics qnd Image Processings **29(1)** (1985), 100–132. A.1.2

[HZ80] R.A. Hummel and S.W. Zucker, *On the foundations of relaxation labeling processes*, McGill Univ. TR, 1980. A.1.1.2

[IH02] Hitoshi Iyatomi and Masafumi Hagiwara, *Scenery image recognition and interpretation using fuzzy inference neural networks*, Pattern Recognition **35** (2002), no. 8, 1793–1806. D.1

[JG98] L. Jouy and F. Guilbert, *Influence des pratiques culturales sur l'évolution de la flore adventice en grandes cultures*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 79–90. (document)

[JZDJ98] A.K. Jain, Y. Zhong, and M.P. Dubuisson-Jolly, *Deformable template models : A review*, SP **71** (1998), no. 2, 109–129. 1.1, 2.1.4

[KDN93] D. Koller, K. Daniilidis, and H.H. Nagel, *Model-based object tracking in monocular image sequences of road traffic scenes*, IJCV **10** (1993), no. 3, 257–281. 2.2

[Kel99] Martin Garcia Keller, *Matching algorithms and feature match quality measures for model-based object recognition with applications to automatic target recognition*, Ph.D. thesis, www.cs.nyu.edu, May 1999. 2.2

[KG99] D. Keren and C. Gotsman, *Fitting curves and surfaces with constrained implicit polynomials*, IEEE Transaction on Pattern Analysis and Machine Intelligence **21** (1999), no. 1, 31–41. 3.2.1

[KH93] Charles Kervrann and Fabrice Heitz, *A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics*, Tech. Report RR-2062, INRIA, 10 1993. A.1.1.7

[KM99] J. Keller and P. Matsakis, *Aspects of high level computer vision using fuzzy sets*, 8th IEEE Int. Conf. on Fuzzy Systems, 1999, pp. 847–852. D.1

[KP02] K.D. Kuhnert and D. Pechtel, *Towards creating abstract features of complex objects : The fusion of contourpoints in significant contour sections for object recognition*, PRL **23** (2002), no. 6, 713–718. 2.1.2, 2.1.2

[KSH02] F. Kurugollu, B. Sankur, and A.E. Harmanc, *Image segmentation by relaxation using constraint satisfaction neural network*, IVC **20** (2002), no. 7, 483–497. A.1.1.8

[KY96] Il Y. Kim and Hyun S. Yang, *A integration scheme for image segmentation an labelling based on markov random field model*, IEEE Transactions on Pattern Analysis and Machine **18** (1996), no. 1, 69–73. A.1.1.7

[LFA71] Jacqueline Lelong-Ferrand and Jean-Marie Arnaudiès, *Cours de mathématiques. tome 1 : Algèbre*, Dunod, 1971, 536 p. 3.1.3

[LHRB98] G. Le Henaff, J.C. Richard, and S. Baud, *La contamination des eaux par les phytosanitaires en bourgogne et franche-comté*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 361–368. (document)

[LL93] F. Leymarie and M.D. Levine, *Tracking deformable objects in the plane using an active contour model*, IEEE Transactions on Pattern Analysis and Machine Intelligence **14** (1993), no. 1, 56–75. 2.1.4

[LLCS98] Wen-Huei Lin, Jiann-Shu Lee, Chin-Hsing Chen, and Yung-Nien Sun, *A new multiscale-based shape recognition method*, Signal Processing **65** (1998), no. 1, 103–113. 2.1, 2.1.2, 3.3

[Lon98] S. Loncaric, *A survey of shape analysis techniques*, Pattern Recognition **31** (1998), no. 8, 983–1001. 2, 2.1.3, 2.1.3

[Low87] David G. Lowe, *Three-dimensional object recognition from single two-dimensional images*, Artificial Intelligence **31** (1987), no. 3, 355–395. 2.2, 5.2.3.1, 6

[Low91] ———, *Fitting parameterized three-dimensional models to images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-13** (1991), no. 5, 441–450. 2.2

[Lut99] P.J.W. Lutman, *Methods of weed patch detection in cereal crops*, Brighton Conference - Weeds (Brighton, UK), 1999, pp. 627–634. (document)

[LVW97] C.P. Lam, S. Venkatesh, and G.A.W. West, *Hypothesis verification using parametric models and active vision strategies*, CVIU **68** (1997), no. 2, 209–236. 2.2

[LWR] Xiuwen Liu, Deliang Wang, and J. Raul Ramirez, *A two-layer neural network for robust image segmentation and its application in revising hydrographic features*. A.1.1.2, A.1.1.8

[Maî] Henri Maître, *Vision par ordinateur et raisonnement dans les images*, TELECOM PARIS, http ://www.infres.enst.fr/dfi/briques/consult-eleves.phtml ?sigle=VOIR. 2.1.2

[MDSA00] J. Montagnat, H. Delingette, N. Scapel, and N. Ayache, *Representation, shape, topology and evolution of deformable surfaces. application to* 3D *medical image segmentation*, Rapport de Recherche RR 3954, INRIA, janvier 2000. 2.1.4

[MRA02] A-G Manh*, G. Rabatel*, L. Assemat**, and M. J. Aldon **, *Weed leaf segmentation by deformable templates*, Ph.D. thesis, Ensam, 2002. 1.1, 2.1.4, 2.1.4

[MS98] R. Mestre and L. Souliac, *Contamination des eaux par les produits phytosanitaires en france*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 377–385. (document)

[MS04] Majed Marji and Pepe Siy, *Polygonal representation of digital planar curves through dominant point detection–a nonparametric algorithm*, Pattern Recognition **37** (2004), no. 11, 2113–2130. 3.2.1, 1, 5.2.2.1

[OST99] P. Olver, G. Sapiro, and A. Tannenbaum, *Affine invariant detection : edge maps, anisotropic diffusion, and active contours*, Acta Applicandae Mathematicae **59** (1999), 45–77. 2.1.2

[PD02] Nikos Paragios and Rachid Deriche, *Geodesic active regions : A new framework to deal with frame partition problems in computer vision*, Journal of Visual Communication and Image Representation **13** (2002), no. 1-2, 249–268. 2.1.4

[PH91] C.G. Perrott and L.G.C. Hamey, *Object recognition : A survey of the literature*, MacQuarie Univ., 1991. 2

[PL93] A.R. Pope and D.G. Lowe, *Learning object recognition models from images*, ICCV93, 1993, pp. 296–301. 2.2

[Pop94] A.R. Pope, *Model-based recognition. a survey of recent research*, Tech. Report 94-04, Univ. of British Columbia, janvier 1994. 2, 6

[PP93] N.R. Pal and S.K. Pal, *A review on image segmentation techniques*, Pattern Recognition **26** (1993), no. 9, 1277–1294. A.1.2

[Pre70] J.M.S. Prewitt, *Object enhancement and extraction*, PPP70, 1970, pp. 75–149. A.1.1.1

[Ram87] Lyle Ramshaw, *Blossoming : A connect-the-dots approach to splines*, HP Labs Technical Reports **SRC-RR-19** (1987), 172. 3.2.1

[RB96] Rajesh P. N. Rao and Dana H. Ballard, *Dynamic model of visual recognition predicts neural response properties in the visual cortex*, Tech. Report NRL96.2, Department of Computer Science, University of Rochester, 1996. 2.2

[RG98] X. Reboud and J. Gasquez, *Atouts et faiblesses des mélanges d'herbicides : une analyse globale à partir de 92 formulations homologuées sur céréales*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 157–164. (document)

[RM91] A.A. Rodriguez and O.R. Mitchell, *Image segmentation by succesive background extraction*, PR **24** (1991), no. 5, 409–420. A.1.1.1

[RM93] H. Rom and G.G. Medioni, *Hierarchical decomposition and axial shape description*, PAMI **15** (1993), no. 10, 973–981. 2.1.3, 2.3, 2.1.3, 6

[Rob00] P.C. Robert, *L'agriculture de précision : les verrous liés à la technologie et à la gestion agronomique*, Agriculture de précision : Avancées de la recherche technologique et industrielle. Colloque UMR Cemagref-ENESAD (Dijon), 2000, pp. 11–29. (document)

[Ros01] Paul L. Rosin, *Unimodal thresholding*, The Journal Of Pattern Recognition Society, vol. 34, 2001, pp. 2083–2096. A.1.1.1

[RR92] Bimal Kumar Ray and Kumar S. Ray, *An algorithm for detection of dominant points and polygonal approximation of digitized curves*, Pattern Recognition Letters **13** (1992), no. 12, 849–856. 3.2.1

[RW95] E. Rivlin and I. Weiss, *Local invariants for recognition*, IEEE Transactions on pattern analysis and machine intelligence **17** (1995), no. 3, 226–238. 2.1.2, 2.1.2

[SC92] J. Shen and S. Castan, *An optimal linear operator for step edge detection*, GMIP **54** (1992), no. 1, 112–133. 3.2.1.1

[Sch74] H. Schwetlick, *Zur Minimierung von Funktionen mehrerer Veränderlicher mittels ableitungsfreier Verfahren vom Newton-Typ*, Zh. Vychisl. Mat. i Mat. Fiz. **14** (1974), 278–291, English translation : *Minimization of functions of several variables by derivative-free methods of Newton type*, U.S.S.R. Comput. Math. and Math. Phys. 14 (1974), No. 2, 3–16 (1975). 4.2, 4.2.3

[Sch98] M. Schiavon, *La pollution des eaux par les produits phytosanitaires*, 17ème conférence du COLUMA. Journées Internationales sur la lutte contre les mauvaises herbes (Dijon) (ANPP, ed.), vol. 1, 1998, pp. 419–428. (document)

[SHB99] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image processing analysis, and machine vision*, 2 ed., PWS Publishing, 1999. A.1.1.3, A.1.2

[Sin99] S. Singh, *A single nearest neighbor fuzzy approach for pattern recognition*, International Journal of Pattern Recognition and Artificial Intelligence **13** (1999), no. 1, 49–64. D.1

[SN02] Marc Sebban and Richard Nock, *A hybrid filter/wrapper approach of feature selection using information theory*, Pattern Recognition **35** (2002), no. 4, 835–846. A.1.2

[SSS03] B. Sarkar, L.K. Singh, and D. Sarkar, *Approximation of digital curves with line segments and circular arcs using genetic algorithms*, PRL **24** (2003), no. 15, 2585–2595. 3.2.1

[Tar75] R. E. Tarjan, *Efficiency of a good but not linear set union algorithm*, Journal of the ACM **22** (1975), 215–225. A.2

[TB97] Alain Trémeau and Nathalie Borel, *A region growing and merging algorithm to color segmentation*, The Journal Of Pattern Recognition Society **30** (1997), no. 7, 1191–1203. A.1.1.4

[TC89] C.H. Teh and R.T. Chin, *On the detection of dominant points on digital curves*, IEEE Transaction on Pattern Analysis and Machine Intelligence **11** (1989), no. 8, 859–872. 3.2.1

[The97] J. Theiler, *A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation*, 1997. A.1.1.6

[Tré98] A. Trémeau, *Analyse d'images couleurs : du pixel à la scène*, Habilitation à diriger des recherches, Université Jean Monnet, St Etienne, France, 1998. A.1.2

[VDW97] H.M.G. Van Der Werf, *Evaluer l'impact des pesticides sur l'environnement*, Le Courrier de l'Environnement **31** (1997), 0–10. (document)

[VH99] R. Veltkamp and M. Hagedoorn, *State-of-the-art in shape matching*, Tech. Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999. 2, 2.1

[VMB97] A. Verikas, K. Malmqvist, and L. Bergman, *Color image segmentation by modular neural-network*, PRL **18** (1997), no. 2, 173–185. A.1.1.8

[VS96] K. Voss and H. Suesse, *Invariant fitting of planar objects by primitives*, Proceedings of the 13th International Conference on Pattern Recognition (Vienne, Autriche), vol. 1, IEEE Computer Society Press, 1996, pp. 508–512. 2.1.2

[XLT02] Zhong Xue, Stan Z. Li, and Eam Khwang Teoh, *Ai-eigensnake : an affine-invariant deformable contour model for object matching*, Image and Vision Computing **20** (2002), no. 2, 77–84. 2.4, 2.1.4, 6

[Zad65] L.A. Zadeh, *Fuzzy sets*, Information Control **8** (1965), 338–353. D.1

[ZC95] Pengfei Zhu and Paul M. Chirlian, *On critical point detection of digital shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995), no. 8, 737–748. 3.2.1

[Zuc76] S.W. Zucker, *Childhood and adolescence*, Computer Graphics qnd Image Processings **5** (1976), 382–399. A.1

[Zwa97] P. Zwaenepoel, *Les matériels d'application modulée*, Perspectives Agricoles **222** (1997), 36–41. (document)

[ZY96] S. Zhu and A. Yuille, *Forms : A flexible object recognition and modeling system*, 1996. 2.1.3

# Annexes

# A. Segmentation d'images

## A.1 Aperçu de l'existant

La segmentation d'images en régions est définie par *Zucker* [Zuc76] de la façon suivante : "une segmentation en régions est une partition de l'image au sens topologique pour laquelle chaque région vérifie des propriétés (ou prédicats) d'homogénéité". La segmentation d'image a pour but d'isoler les objets (ou zones) de couleur ou de texture homogène. Ces méthodes sont utilisées, par exemple, dans l'extraction d'objets ou bien dans l'analyse de scène. Beaucoup de méthodes couleur[1] reprennent des méthodes définies initialement pour des images en niveau de gris[2] et les adaptent pour travailler dans un espace couleur. Le problème essentiel en segmentation couleur est d'exprimer au mieux l'information contenue dans la région ou dans le pixel par choix de l'espace couleur le plus adapté au problème posé. Il peut être difficile de trouver un espace couleur répondant correctement aux attentes. Il existe, en effet, 7 principaux espaces de couleurs (cf. tableau B.1) permettant de s'affranchir de telle ou telle composante, comme la luminosité ou la saturation. L'espace RGB, par exemple, ne permet pas d'avoir directement les informations de luminosité et de teinte bien séparées contrairement à l'espace HSI. Mais l'espace HSI a aussi ses défauts et n'est pas stable, par exemple, pour des valeurs de luminosité faibles. De ce fait, aucune représentation n'est vraiment idéale et certaines procédures vont jusqu'à exploiter des vecteurs de dimension 30 (issus des composantes des différents espaces) pour obtenir toute l'information nécessaire à la segmentation couleur.

D'autre part, il n'existe pas de procédé universel capable de segmenter n'importe quelle image correctement. De nombreuses méthodes que nous pouvons trouver dans la littérature sont des méthodes hybrides utilisant les avantages de différentes techniques plus classiques pour obtenir de bons résultats. Parmi celles-ci, nous notons aussi l'existence de méthodes de collaboration entre segmentation régions et extraction de contours[3]. L'intérêt est d'exploiter les deux informations (régions et contours), qui sont duales, pour améliorer les résultats de segmentation (par exemple, [Gor03]). Les procédures existantes dans la littérature se recoupent souvent dans les procédés de bases qu'elles utilisent. Nous allons, donc, exposer ces principales techniques de bases.

[1] La couleur est représentée, en général, par un espace à plusieurs dimensions.

[2] Les niveaux de gris sont représentés, en général, par un espace mono dimensionnel.

[3] Extraction de contours obtenus, par exemple, en utilisant des filtres comme ceux de Canny, Deriche, *etc.*.

### A.1.1 Méthodes de segmentation

#### A.1.1.1 Techniques de segmentation basées sur le seuillage

C'est un principe basique qui suppose que les objets contenus dans l'image sont homogènes en terme de couleur (pas de texture). Ainsi, si un seuillage est appliqué sur l'image en choisissant par pixel une des valeurs le composant (valeur issue, par exemple, en sélectionnant une des dimensions de l'espace couleur), cela permet de séparer les objets les uns des autres ou d'en ignorer certains [RM91]. Il est aussi possible d'utiliser une fonction de distance métrique pour associée, afin d'être appliqué avec le seuil. Cette notion de seuillage est très souvent utilisée, par exemple, pour déterminer simplement l'appartenance, ou non, d'un point à telle ou telle classe en fonction d'une série de seuils (représentant les classes) et d'une distance métrique pour l'espace couleur considéré. Fréquemment les méthodes de seuillage sont appliquées sur des histogrammes générés à partir de l'image afin d'isoler des pics significatifs des zones d'intérêts. Puis, ces pics sont exploités pour déterminer les régions sous-jacentes de l'image. Par exemple, nous pouvons citer [Pre70] pour illustrer la méthode. [Ros01] a, aussi, développé une procédure de résolution sur des histogrammes uni-modaux ou bi-modaux.

#### A.1.1.2 Segmentation par Relaxation

Des règles ou des contraintes sont définies permettant de déterminer la nature de l'objet inspecté (cf. [HZ80], [HH97]). Au début, tous les objets de la scène ont tous les labels possibles puis l'application des règles (ou des contraintes) permet de réduire les labels par objet afin d'en obtenir un (au mieux ou sinon plusieurs) par objet. Donc pour déterminer le nombre de labels et le type des contraintes, une connaissance *a priori* est supposée exister sur le contenu de la scène. Si l'analyse s'effectue au niveau du pixel elle avance de proche en proche en acceptant ou refusant des labels selon des critères basés, par exemple, sur une notion de différence de niveau de gris ou selon une distance métrique. Néanmoins, cette approche n'a pas de convergence théorique et est particulièrement lente (cf. [LWR]).

#### A.1.1.3 Algorithmes de partage des eaux

L'image est supposée être assimilable à une carte topographique (montagnes et ravins). Par exemple plus la couleur du pixel est foncée plus la profondeur est importante. À partir de ce principe l'image est considérée comme étant composée de plaines pas forcement régulières susceptibles de représenter des régions (aux endroits où l'image forme des tâches homogènes). Alors, une inondation est simulée dans ce relief. Si un bassin est trop rempli et qu'il se déverse dans un autre, alors une région est trouvée. Cette méthode se rapproche un peu de la logique des principes basés sur les histogrammes dans la méthode de sélection des pics (creux). Un des problèmes de cette procédure est le temps de calcul pour déterminer et remplir tous les bassins. Voir [SHB99], p. 592 pour une illustration de cette procédure.

#### A.1.1.4 Techniques de segmentation par agrandissement de région

La méthode dispose un certain nombre de graines dans l'image afin de les faire grandir en fonction d'un critère d'homogénéité. Souvent la graine est l'élément de base de l'image (pixel) qui est considéré comme une région à lui tout seul avec un ensemble de caractéristiques. Après, de proche en proche il lui est ajouté des points appartenant à son voisinage selon des critères d'homogénéité locaux ou prenant en compte toutes les caractéristiques des régions concernées. Le but de la méthode est d'obtenir une nouvelle région qui reste homogène selon l'ensemble des informations et des seuils. Là aussi, nous pouvons trouver nombre

de méthodes reprenant ce principe, nous pourrons en citer deux [TB97] et [FGL00]. Le principal défaut de ces procédures est généralement le temps de calcul pris par les agrandissements successifs.

#### A.1.1.5 Techniques de segmentation par *Split and Merge*

L'image entière est considérée et divisée en parts, en général de taille égale, il est ainsi possible de labelliser l'image d'origine comme le père des parts créées et les parts comme ses fils. Si les divisions conservent la même homogénéité par rapport à l'étape précédente, alors cette zone n'est plus divisée, sinon le procédé recommence division par division (les fils ont eux-mêmes des enfants, *etc.*). Une sorte d'arbre généalogique est donc généré. Ceci correspond à l'étape du split, pour le merge, le procédé remonte l'arbre généalogique en commençant par considérer les feuilles de l'arbre (les plus petits enfants). Au fur et à mesure il va fusionner les régions adjacentes ayant la même homogénéité. La méthode est décrite dans [CMVm86], nous pouvons prendre l'exemple du quad-tree qui est obtenu par la division d'une région en quatre sous régions. Un des problèmes de ce type de méthode est la recherche et la réunion des zones contiguës de même homogénéité mais n'ayant pas le même père dans l'arbre.

#### A.1.1.6 Techniques de segmentation par regroupement (*K-means clustering*)

Il s'agit ici d'algorithmes que nous retrouvons dans d'autres domaines que la segmentation. En effet, ils peuvent être utilisés pour, par exemple, faire de la classification afin de regrouper des éléments proches selon une métrique (grouper les points proches dans un nuage de points). Ici, la méthode s'applique de la même manière en définissant le centre des classes dans l'image puis en attachant de nouveaux points proche à la classe, puis le centre de la classe est recalculée. La méthode itère jusqu'à ce qu'il n'y ait plus de point qui ne soit pas assigné à une classe. Voir [The97] pour un exemple d'implantation.

#### A.1.1.7 Techniques de segmentation basées sur les textures

Les textures sont définies par la répétition, plus ou moins régulière, de différents motifs pouvant être, eux-mêmes, composés d'autres motifs ou par la présence de fréquences spatiales privilégiées dans l'image. Les matrices de co-occurrences sont fréquemment utilisées pour représenter des textures mais nous pouvons, aussi, trouver d'autres méthodes comme celles ci-dessous :

- Segmentation basée sur les MRF (Markov Random Field)
  Cette méthode de segmentation exploite un champ de Markov associé à une distribution de Gibbs. Le champ de Markov définit une relation probabiliste de voisinage (clique) sur des éléments de l'image (sachant qu'un élément peut être un pixel ou un ensemble de pixels). Si cette relation probabiliste est une distribution de Gibbs, elle permet alors de définir, selon une mesure de distance, un voisinage potentiel pour une clique en fonction d'un ensemble de cliques (de préférence appartenant au voisinage de la clique à tester). En disposant d'un ensemble de fonctions permettant de mesurer la distance entre deux cliques (ou deux zones de textures), il est possible de les combiner de manière à définir ce potentiel. La distribution de Gibbs permet d'exprimer ce potentiel sous forme d'une probabilité qui peut être, par exemple, exploitée dans des algorithmes de regroupement (cf. paragraphe précédent). Voir [KH93] et [KY96].
- Segmentation exploitant la notion de fractals
  Cette méthode est adaptée aux textures qui contiennent plusieurs fois la même information à différentes échelles, comme dans un objet fractal. La méthode cherche à caractériser l'image en calculant la dimension fractal en chaque point, pour ensuite segmenter l'image en fonction de cette

caractéristique. Pour cela, l'image est divisée en éléments de taille $s \times s$ sur lesquels est cherchée la valeur $min$ et $max$ (niveau de gris). En faisant varier la taille de $s$ et en comparant les différents $max - min + 1$ (plus l'image est fractal plus cette valeur est stable à toutes les échelles $s$) une valeur de base est déterminée permettant de calculer la dimension fractal de Haussdorff. Cette dimension combinée à d'autres ($smoothness$, $min$, $max$, *etc.*) donne un vecteur de caractéristiques qui peut être utilisé dans un algorithme de type K-means clustering ou similaire permettant de segmenter l'image. Voir [CS95] pour plus de détails.

#### A.1.1.8 Intelligence artificielle

D'autres procédés vont plutôt employer des méthodes utilisées pour simuler des comportements d'entités biologiques. Nous allons en présenter deux :

- Logique Floue
  Cette méthode exploite le principe d'affectation incertaine, elle est assimilable aux méthodes de relaxation où une région peut avoir plusieurs labels, sauf que l'appartenance à une région est pondérée. Ici, le choix de l'affectation est défini selon un ensemble de conditions de type (IF THEN ELSE) et de caractéristiques floues. Par exemple, [CGHK99a] effectue d'une segmentation par recherche d'un chemin entre deux pixels. Le coût de cette recherche détermine une mesure de similarité. Cette recherche s'effectue de proche en proche selon l'affinité des pixels (ici définie par une gaussienne) et un seuil. Cette méthode présente de bons résultats de segmentation si elle est guidée en lui spécifiant des points d'intérêt dans l'image. Par contre, ce type de parcours est trop lent : la recherche de chemin est plus lente que celle définie par la procédure de Dijkstra [Dij82].
- Réseau de Neurones
  Les méthodes exploitant ce principe se basent sur des méthodes de regroupement ou relaxation. Outre leur usage devenu courant en classification, les réseaux de neurones sont parfois utilisés avec une configuration de réseau directement associée à l'espace de l'image. Les procédés de relaxation sont appliqués en fonction de statistiques locales associées à un graphe d'adjacence représentant le réseau de neurones. Ces principes agissent localement et peuvent être parallélisé. Cela permet d'obtenir de petites régions et à chaque itération, le réseau permet d'affiner la segmentation. De plus, plusieurs couches de réseaux de neurones peuvent intervenir dans le but de contrôler les couches précédentes et d'améliorer la segmentation. Voir [LWR, VMB97, FWF02, KSH02].

### A.1.2 Bilan

Beaucoup de méthodes (partage des eaux, relaxation, agrandissement de régions, *etc.*) utilisent des prédicats de similarité pour établir une mesure entre des éléments de couleur. De nombreux prédicats d'homogénéité ou de similarité ont été proposés dans la littérature (par exemple, [SN02]). Nous pourrons nous référer, pour une présentation de ces critères, aux travaux suivants : *Trémeau* [Tré98], *Haralick et Shapiro* [HS85], *Falah et al* [FBC91].

Plus d'informations sur les méthodes présentées peuvent être trouvées dans le livre [SHB99] retraçant les différentes procédures d'analyse et de traitement d'images (comme la segmentation). Il décrit les méthodes dans leurs principes généraux et permet d'avoir un bon aperçu de l'ensemble de cibler les méthodes qui nous intéressent le plus. Nous pouvons aussi parler de l'article [CJSW00] prenant en compte un certain nombre de procédures mais aussi d'espaces de couleur et expliquant les meilleurs implantations (quel espace de couleur, quel vecteur de caractéristiques, *etc.*). Voir aussi [PP93], qui est un article plus ancien retraçant les diverses techniques de segmentation et qui a été cité de nombreuses fois. Une autre source d'information est la bibliographie sur l'imagerie [Big] maintenue à jour en permanence.

Les principaux défauts des méthodes présentées sont le temps de calcul et la complexité de mise en œuvre. Alors que certaines méthodes sont relativement simples à implanter (comme le seuillage), mais ne donnent pas de résultats probant pour des images complexes, d'autres, comme le partage des eaux, sont plus difficiles à mettre en place mais peuvent donner de bons résultats sur des images réelles.

Notre choix s'est porté sur la segmentation par croissance de régions. Pour parvenir à de bonnes performances nous avons optimisé un algorithme développé par Christophe Fiorio exploitant l'*Union-Find*. Nous allons maintenant décrire cette méthode et les améliorations que nous y avons apportées.

## A.2 Méthode mise en place

La segmentation d'images a pour but de regrouper des pixels contigus de manière à obtenir des zones ou régions homogènes assimilables à des objets. La méthode mise en place, de type *region growing* (accroissement de régions), exploite un algorithme ensembliste (l'*Union-Find*, [Tar75]) qui a été adapté par C. Fiorio et J. Gustedt [FG96] pour la mise en œuvre de la segmentation. Nous avons optimisé cette adaptation pour la place mémoire et le temps d'exécution. Nous avons aussi ajouté la possibilité pour cette méthode de supporter différents espaces couleur (cf. tableau B.1) à partir du moment où il est possible d'y définir une métrique pour comparer deux points/régions de cet espace. Le résultat de cette procédure, en utilisant une métrique adaptée[4], est un ensemble de régions au sein desquelles la couleur est homogène. Une région contient un certain nombre d'informations utiles à notre algorithme, comme son nombre de pixels et sa couleur moyenne.

Le principe de la méthode est le suivant : nous parcourons l'image et comparons chaque pixel (assimilé à la région de plus petite taille) avec ses voisins. Si un de ses voisins peut être considéré comme similaire[5], alors une nouvelle région est créée regroupant les deux régions initiales. Nous pouvons alors assimiler cet ensemble de régions à un arbre généalogique : la nouvelle région est le père et les deux régions initiales sont les fils. Cette nouvelle région cumule l'ensemble des caractéristiques (couleurs et nombre de pixels) des régions initiales. Ainsi, à force de regroupement de régions et, donc, de création de grand-père, arrière grand-père, *etc.*, l'algorithme va générer une sorte d'image composée d'arbres de régions. Dans cette image, le père d'un de ces arbres est le représentant d'une région finale et ses fils sont les sous-régions qui ont été fusionnées.

L'ALGO. 2, p. 114 décrit la méthode qui se décompose en trois phases : initialisation, segmentation et mise à jour de l'image.

Une des améliorations apportées à la méthode est le remplacement de la phase d'initialisation, où toutes les régions initiales sont créées, par l'ajout d'une phase où les régions sont créées uniquement lors de la première fusion d'un élément dans un pixel. Ainsi, nous économisons énormément de place mémoire. Aussi, contrairement à la version précédente où les comparaisons ne se faisaient qu'entre régions, les comparaisons peuvent, aussi, se faire entre un pixel et une région.

Le type de parcours dans l'image utilisé pour cette procédure est le sens de lecture en $x$ puis en $y$. Ainsi, du fait du parcours utilisé ($x$ puis en $y$) et de l'utilisation de la 4-connexité, nous n'allons explorer que deux voisins par point à traiter, comme illustré sur la figure A.1.

Cette méthode génère, en plus de l'ensemble des régions de pixels similaires, un graphe d'adjacence entre ces régions (cf. figure A.2).

Ce graphe est généré à la volée lors de la mise à jour de l'image. La fonction analyse le voisinage du

[4]Les espaces couleurs HSI et RVB ont été testé, les métriques utilisées sont détaillées en annexe : annexe B

[5]La similarité est déterminée grâce à la mesure de distance dans l'espace couleur utilisé

**Algorithme 2** : Union-Find

*Entrée : image de pixels / sortie : région de pixels et graphe d'adjacence*
*// Initialisation* ;
**pour** ∀ point Pt de l'image **faire**
  définir une nouvelle région Reg ;
  couleur de Reg = couleur de Pt ;
  nombre de pixels de Reg = 1 ;
  région de Pt = Reg ;
*// Segmentation* ;
**pour** ∀ point Pt de l'image **faire**
  **pour** ∀ point PtV voisin de Pt **faire**
    *// Find du représentant de la région courante* ;
    RegPere = Find(région associée à Pt) ;
    *// Find du représentant de la région voisine* ;
    RegVPere = Find(région associée à PtV) ;
    **si** RegPere $\neq$ RegVPere et Dist($col$(RegVPere), $col$(RegPere)) $<$ Seuil **alors**
      *// Union de RegVPere dans RegPere* ;
      Union(RegVPere, RegPere) ;
*// Mise à jour de l'image* ;
**pour** ∀ point Pt de l'image **faire**
  père de la region Pt = plus grand ascendant de la région de Pt ;
  couleur de Pt = couleur de la région de Pt ;
  GenereGraphe(Pt) ;

**Procédure** Union(*père du voisin, père courant*)

père de RegVPere = RegPere ;
couleur de RegPere = couleur moyenne entre RegPere et RegVPere ;
nombre de pixel de RegPere += nombre de pixel de RegVPere ;

**Fonction** Find(*région*)

**si** père de Reg $\neq \phi$ **alors**
  **retourner** Find(père de Reg) ;
**sinon**
  **retourner** Reg ;

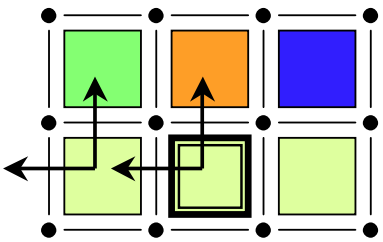


FIG. A.1 – Voisinage visité lors du parcours de l'image.

point considéré et détermine les actions à mener. Cette fonction de génération est personnalisable et permet de stocker toutes sortes d'informations utiles sur les arêtes reliant les nœuds du graphe et ainsi, permet

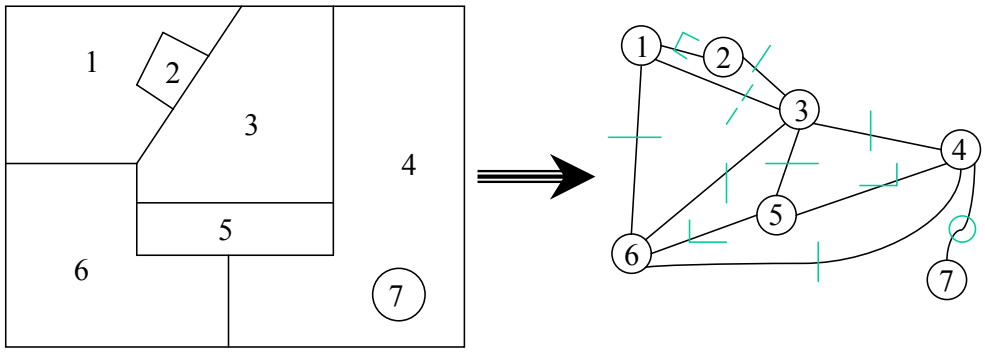



FIG. A.2 – Graphe d'adjacence. À gauche les régions issues de la segmentation et à droite le graphe associé (les frontières entre régions sont représentées sur les arêtes).

de donner une sémantique au graphe. En l'occurrence, comme nous cherchons à extraire l'information de contour de nos régions nous allons stocker, sur les arêtes de ce graphe, les points de frontière entre deux régions (représentées par leur nœud respectif dans le graphe). La fonction de génération est décrite par l'algorithme GenereGraphe, p. 115.

**Procédure** `GenereGraphe(`*point*`)`

```
Reg = région sous-jacente au point Pt ;
si Reg n'a pas de nœud dans le graphe alors
    créer nouveau nœud de Reg ;
Noeud = nœud de Reg ;
pour ∀ voisin PtV déjà exploré faire
    NoeudV = nœud de région sous-jacente au point PtV ;
    si NoeudV ≠ Noeud alors
        si !∃ une arête entre NoeudV et Noeud alors
            créer une arête entre NoeudV et Noeud ;
        ajouter à l'arête entre NoeudV et Noeud l'information utile ;
```

Ce graphe nous permet, lors d'une seconde passe, d'améliorer la segmentation en fusionnant les régions similaires non regroupées, les petites régions et les régions non fusionnées avec une région adjacente. En effet, pendant la segmentation la couleur moyenne des régions est mise à jour au fur et à mesure des unions, et de ce fait deux régions adjacentes, n'ayant pas la même couleur moyenne lors du traitement des pixels proches de la frontière commune, peuvent converger vers la même couleur moyenne mais ne pas être fusionnées du fait du type de parcours dans l'image. Aussi, après l'étape de segmentation, l'image contient des points qui n'ont pas pu être associés à une région car leur information de couleur est trop éloignée de celle des voisins pour qu'une association soit acceptable. De plus, après la segmentation, des petites régions (par exemple, moins de 10 points) peuvent être fréquentes en bord d'objet, lorsque nous générons une segmentation sur une image avec du bruit ou encore lorsque le seuil de fusion de région est trop strict. Ces petites régions et ces points non fusionnés peuvent être considérés comme négligeables dans les cas où la variation de la covariance couleur au sein d'une même région n'a pas d'importance dans les traitements futurs. Néanmoins, dans des cas où il est souhaitable de conserver l'intégrité de l'information couleur, ce procédé d'amélioration de la segmentation peut ne pas être utilisé. En effet, la méthode utilisée pour fusionner ces petites régions avec une région adjacente, se base sur l'élection de la région adjacente avec laquelle la distance couleur calculés est la plus faible. Ce qui veut dire que la fusion peut se faire entre deux régions de couleur proche mais différente. Ainsi, l'information de moyenne et de covariance couleur (celle de la région qui reçoit la petite région) va être faussée par des informations couleur hétérogènes.

Il est intéressant de noter que les frontières entre régions, issues de ce graphe, sont définies dans l'espace inter-pixel, c'est-à-dire entre les points définissant l'image. Les points de cet espace sont représentés sur la figure A.1 par des petits ronds et arêtes noires. Cette représentation nous permet ainsi de bien situer les frontières entre les points des régions et non pas selon les pixels d'une région ou d'une autre.

À la fin de cette étape de segmentation, afin de réduire les futures extractions de contours, nous fusionnons ensemble les régions adjacentes qui n'appartiennent pas à la classe colorimétrique des objets recherchés (par exemple, tout ce qui est de la terre, des cailloux, *etc.*). Puis, nous sélectionnons les frontières séparant une région appartenant à cette classe et une région n'appartenant pas à cette classe, afin d'extraire des contours partiels des objets recherchés (le détail de cette sélection illustre l'utilisation des règles floues, cf. annexe D.2).

# B. Espaces couleur

## B.1 Caractéristique de différents espaces de couleurs

| Espace couleur | Avantages | Désavantages |
|---|---|---|
| RGB | Pratique pour faire de l'affichage | Pas très bon pour traiter des images due à la forte corrélation |
| YIQ | Peut être utilisé pour encoder l'information couleur pour la TV américaine ; Corrige en partie la corrélation RGB ; Induit moins de temps de calcul ; la couche $Y$ (luminosité) est bonne pour la détection de contour | La corrélation existe encore due à la transformation linéaire, mais moindre que le RGB |
| YUV | Peut être utilisé pour encoder l'information couleur pour la TV Européenne ; Corrige en partie la corrélation RGB ; Induit moins de temps de calcul ; la couche $Y$ est bonne pour la détection de contour | La corrélation existe encore due à la transformation linéaire, mais moindre que le RGB |
| I1I2I3 | Corrige en partie la corrélation RGB ; Induit moins de temps de calcul ; peut être utile pour traiter des images | La corrélation existe encore due à la transformation linéaire, mais moindre que le RGB |
| HSI | Basé sur la perception humaine de la couleur ; utile dans certains cas où l'illumination varie, car il est invariant à certains types de lumières et d'ombres ; la teinte peut être utile pour séparer des objets avec des couleurs différentes | Singularités incontournables et numériquement instable vers les basses luminosités due à une transformation non linéaire |
| Nrgb (rgb normalisé) | Les composantes couleurs sont indépendant à la lumière ; utile pour représenter le plan couleur ; robuste au changement d'illumination | Très bruité vers les zones de faible luminosité due à une transformation non linéaire. |
| espaces CIE ($L_{uv}$ ou $L_{ab}$) | Peut contrôler la couleur et la luminosité séparément ; la comparaison entre les couleurs peut être directe car basée sur une séparation géométrique dans l'espace CIE, efficace pour mesurer des petites différences | A les mêmes singularités que les autres dues à une transformation non linéaire |

## B.2 Mesure dans les espaces couleur

Pour chaque représentation couleur que nous avons utilisée dans notre méthode, une distance a été défini de manière à comparer deux régions de pixels.

#### B.2.0.1 Critère gris

La différence entre les niveaux de gris des deux régions à comparer peut s'exprimer par :

$$\mathcal{C}_{11} = |col(reg_i) - col(reg_k)| < seuilGris$$

Il s'agit là d'une distance tout à fait standard.

#### B.2.0.2 Critère RVB

Pour des régions définies dans l'espace couleur RVB, la distance se mesure par la norme du vecteur de la différence des deux couleurs :

$$\mathcal{C}_{12} = \sqrt{\left(R_{reg_i} - R_{reg_k}\right)^2 + \left(V_{reg_i} - V_{reg_k}\right)^2 + \left(B_{reg_i} - B_{reg_k}\right)^2} < seuilRVB$$

#### B.2.0.3 Critère HSI

Pour déterminer si deux régions $reg_i$ et $reg_k$, définies selon l'espace couleur HSI, sont similaires, nous avons défini le critère suivant.

$$\mathcal{C}_{13} = \frac{valRVB + valHSI}{2} < seuilHSI$$

où

$$valRVB = (R_{reg_i} - R_{reg_k})^2 + (V_{reg_i} - V_{reg_k})^2 + (B_{reg_i} - B_{reg_k})^2$$

et

$$valHSI = a \times valSat + b \times valHue$$

avec $valSat$ et $valHue$ définies plus loin.

Si $\mathcal{C}_{13}$ est vérifiée, alors nous fusionnons la région $i$ et la région $k$.

L'information de couleur dans l'espace HSI nous permet de ne prendre en considération que des couleurs cohérentes en terme de luminosité et saturation. Dans notre cas, l'information de saturation est

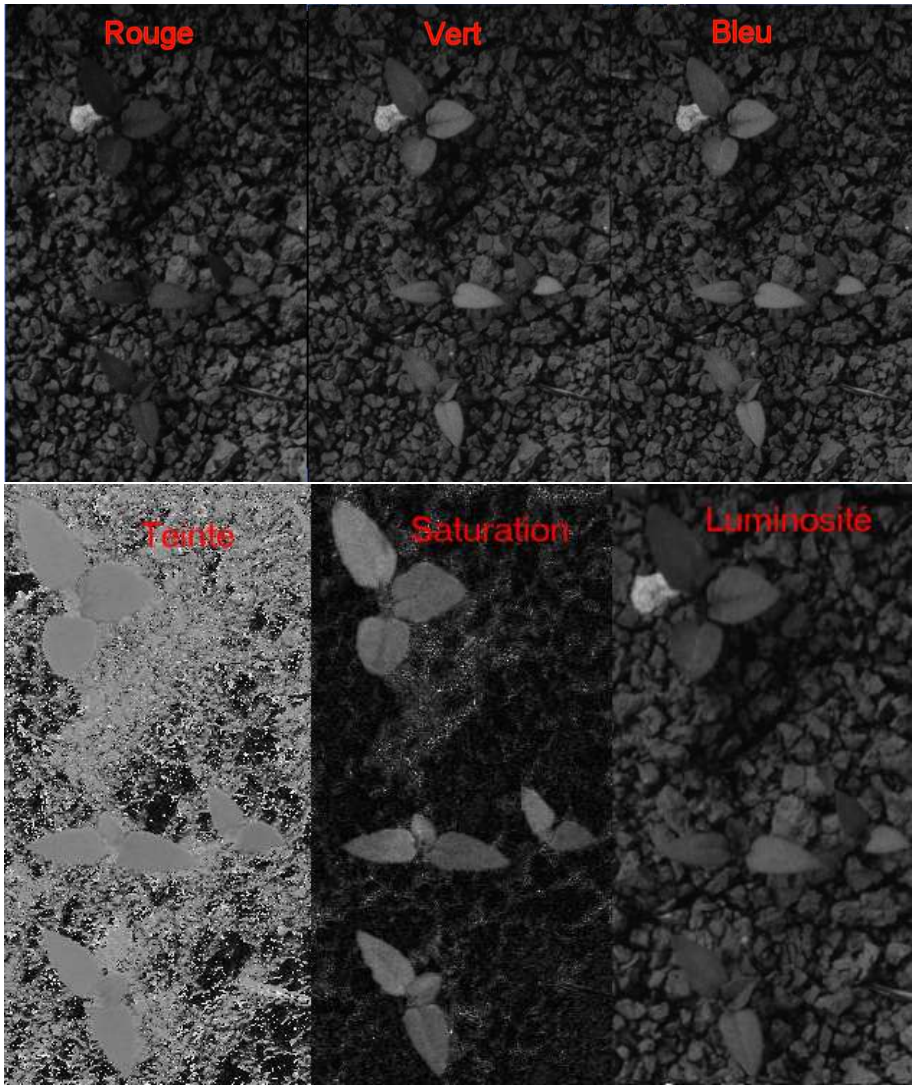



FIG. B.1 – Séparation en plans RVB & HSI.

plus riche que celle de teinte (cf figure B.1), ainsi le choix de la valeur de $\mathcal{C}_{13}$ est pondéré de manière empirique avec les coefficients $a$ et $b$.

$valHSI$ correspond à la définition d'une mesure dans l'espace HSI. Cet espace comporte des zones qui peuvent être imprécises, pour cela nous faisons deux tests afin de traiter ces cas :

$valHue$ **:** lorsque la luminosité est faible, la teinte n'est pas fiable. Mais nous considérons tout de même que si la différence de luminosité entre les deux régions est faible, alors la teinte peut avoir une valeur exploitable.

---

**si** $min(int_{reg_i},\ int_{reg_k}) < minInt$ **alors**
  **si** $|int_{reg_i} - int_{reg_k}| < (minInt/2)$ **alors**
    $valHue = (hue_{reg_i} - hue_{reg_k})^2$
  **sinon**
    $valHue = valErreur$
**sinon**
  $valHue = (hue_{reg_i} - hue_{reg_k})^2$

---

$valSat$ **:** idem lorsque la saturation est trop faible, sa valeur n'est pas fiable. Mais nous considérons tout de même que si la différence de saturation entre les deux régions est faible, alors la saturation peut avoir une valeur exploitable.

---

**si** $min(sat_{reg_i},\ sat_{reg_k}) < minSat$ **alors**
  **si** $|sat_{reg_i} - sat_{reg_k}| < (minSat/2)$ **alors**
    $valSat = (sat_{reg_i} - sat_{reg_k})^2$
  **sinon**
    $valSat = valErreur$
**sinon**
  $valSat = (sat_{reg_i} - sat_{reg_k})^2$

---

Ce critère semble bien prendre en compte les différentes zones de l'espace couleur et pouvoir donner des résultats plus probant que ceux obtenus avec le RVB. Néanmoins, les résultats se sont avérés moins bon sur les frontières des objets que le RVB. Et le plus important dans notre approche reste les contours des objets, même si certains contours manquent ou sont de trop lorsque nous utilisons le critère RVB, ils restent plus 'propres' et donc plus exploitables. Nous préférerons donc utiliser le critère RVB.

# C.
# Ordonnancement de points de contour

De l'étape de segmentation et d'extraction de contour, nous avons extrait un ensemble de frontières séparant deux régions. Ces frontières sont représentées par des ensembles de points ordonnés selon les coordonnées $y$ puis selon les coordonnées $x$. Notre problème va être de générer, par frontière, des sous-ensembles de points connexes ordonnés selon l'un des sens de la courbe discrète sous-jacente. Il est à noter que tous les contours seront considérés comme non fermés. Les points de contours sont stockés selon la topologie de l'espace inter-pixel, ce qui nous simplifie le traitement.

L'algorithme est inspiré de celui proposé par C. Fiorio [Fio95] qui ne traitait que des contours fermés. Il a été adapté pour travailler sur des contours ouverts. Cet algorithme fonctionne à la volée, c'est-à-dire les points sont ajoutés à la frontière existante, supposée ordonnée, puis placés en son sein de manière à respecter l'ordre. Ci-après l'illustration pour un contour représentant un 'm', avec un ordre de lecture des points dans l'image et un ordre désiré (les points sont représentés par leur numéro d'ordre, par exemple 01).

```
    010203          040506      |      091011          252627
    07  08          09  10      |      08  12          24  28
111213  141516  171819  202122  |  050607  131415  212223  293031
23          24  25          26  |  04          16  20          32
27          282930          31  |  03          171819          33
32                          33  |  02                          34
34                          35  |  01                          35
```

FIG. C.1 – Ordre de lecture des points et un ordre désiré

Cet algorithme, de part son mode de fonctionnement incrémental, a pu être directement intégré dans la fonction de création du graphe de la méthode *Union-Find*. Ce qui permet, à partir de ce même graphe, d'extraire pour une région donnée, les contours constituant les frontières avec les régions adjacentes.

En fait, l'algorithme exploite, par frontière, une liste de segments de points où chaque segment est constitué de points contigus (ordonnés). Les segments d'une liste représentent des bouts de contours non contigus entre eux.

Pour ajouter un point à une frontière, nous devons rechercher quel est le segment qui convient et à quelle extrémité de ce segment il doit être ajouté. Un point ne peut être ajouté qu'à une extrémité d'un segment, sauf si ce point ferme le contour. Il est à noter que le parcours dans l'image se fait selon l'espace image, alors que nous ajoutons des points de l'espace inter-pixel. L'ajout d'un point à la liste de segments est alors guidé par une connaissance *a priori* des actions à faire selon les possibilités de positions des points, de l'espace image, entourant le point courant, dans l'espace inter-pixel.

**Algorithme 6** : Chaînage de contours.

*Entrée : un ensemble de points lus / sortie : contours/frontières sous forme de suites de points ordonnées. La fonction* `ajouter` *ajoute un point à l'extrémité la plus proche.*

*segCourant* = $\phi$ ;
**pour** $\forall$ *point* $\in$ *ensemble des points lus* **faire**
  **maj** **si** *point.y* > *yCourant* **alors**
    *yCourant* = *point*.y ;
    *segCourant* = *listeSegment*.debut ;
  **si** *point.*`estConnexe` *(segCourant)* **alors**
    *segCourant*.`ajouter` (*point*) ;
  **sinon**
    *cond* = (*point*.x < *segCourant.extremite1*.x) $\vee$ (*point*.x < *segCourant.extremite2*.x) ;
    **si** *cond est VRAI* **alors**
      **avant** *segCourant* = *listeSegment*.`nouveauSegmentAvant` (*segCourant*);
      *segCourant*.`ajouter` (*point*) ;
    **sinon**
      **après** *segCourant* = *listeSegment*.`rechercherSegment` (*segCourant*, *point*) ;
      **si** *segCourant n'existe pas* **alors**
        *segCourant* = *listeSegment*.`nouveauSegmentApres` (*listeSegment*.fin);
      *segCourant*.`ajouter` (*point*) ;
  **fin** **si** *point.flag* == *LINK_POINT* **alors**
    *segCourant2* = *listeSegment*.`rechercherSegment` (*segCourant*, *point*) ;
    *listeSegment*.`fusionner` (*segCourant*, *segCourant2*) ;

Nous dénombrons 5 configurations possibles de positions de points – pré-codes – illustrées en figure C.2. Sur cette figure, pour chaque cas illustré, le point courant est le point en bas à droite. Les cas sont ordonnés de droite à gauche et de haut en bas. Le premier cas illustre les points image explorés à partir du point courant, il illustre aussi le cas où il n'y a rien à faire (par exemple, lorsque nous nous situons au milieu d'une région). Dans un cas donné, les points hachurés sont des points qui ne sont pas considérés. Les ronds et traits noirs représentent des inter-pixels.

Nous constatons à partir de cette figure que les inter-pixels sont ajoutés, en règle générale, deux par deux : un point et un trait. Une opération spéciale est générée lorsque nous ajoutons un point réalisant la jonction entre deux segments. Pour les fusionner nous réorientons l'un des deux segments, lorsque les extrémités de fusion sont mal placées, et ainsi, nous pouvons représenter ce contour par un seul segment. Aussi, au moment de l'identification du pré-code de fin, un marqueur est positionné sur le point de jonction (`LINK_POINT`).

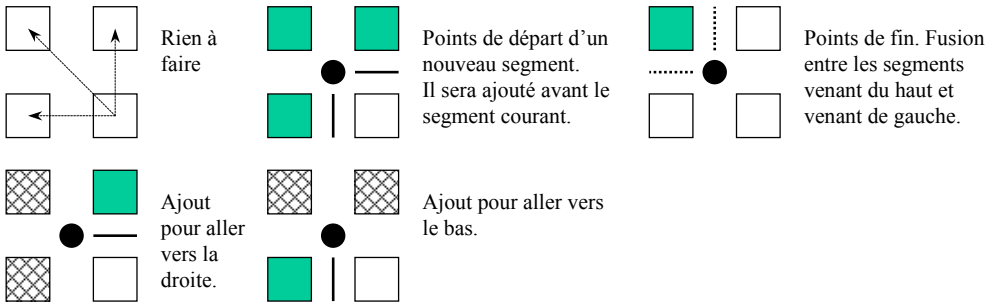



FIG. C.2 – Pré-codes possibles.

L'algorithme ALGO. 6, p. 122 commence la recherche du segment adéquat en utilisant le dernier segment auquel il a ajouté un point. Ainsi, en considérant le mode de progression dans l'image, nous pouvons définir les comportements suivants :

- Si le point se situe avant (en terme d'abscisse) une extrémité du segment courant, un nouveau segment est alors inséré avant le segment courant (cf. ALGO. 6.**avant**).
- Sinon, si le point à ajouter se situe plus loin que les extrémités du segment courant, la recherche du segment adéquat va se faire parmi les segments suivants de la liste (cf. ALGO. 6.**après**). Si la recherche a échoué, un nouveau segment est alors créé à la fin de la liste.
- Si un point de 'fin' (de jonction) doit être ajouté, les deux segments qui se retrouvent soudés doivent être fusionnés (cf. ALGO. 6.**fin**).

De plus, lorsque le parcours dans l'image reprend au début d'une nouvelle ligne, nous déplaçons, alors, le pointeur de segment courant au début de la liste (cf. ALGO. 6.**maj**).

Nous obtenons comme résultat, par frontière, une liste de plusieurs segments si la frontière est interrompue (comme sur la figure A.2 avec l'arête entre la région 1 et 3) où chaque segment représente une partie continue de la frontière. Ou bien, nous obtenons une liste contenant un seul segment représentant l'intégralité de la frontière lorsque celle-ci n'est pas interrompue.

# D.
# Règles floues

## D.1 Mise en place

La logique floue, dont les principes ont été définis par L. Zadeh [Zad65], est présentée dans [Bou95] de manière suivante :

> *La logique floue intervient dans la manipulation de connaissances imparfaites. Elle aide à formaliser la représentation et le traitement des connaissances imprécises ou approximatives, afin de pouvoir traiter des systèmes d'une grande complexité dans lesquels sont, par exemple, présents des facteurs humains.*

La logique floue est aussi exploitée dans le cadre de l'analyse d'images réelles pour prendre en considération les variations de formes des objets ou bien la faible qualité de la prise de vue. La littérature abonde d'articles, nous pouvons citer, par exemple, [Blo99a] [Blo99b] [BM94] [IH02] [KM99] [BB99] [CGHK99b] [Sin99]. La logique floue peut être utilisée à plusieurs niveaux : pixels, objets, relation entre des éléments, *etc.*. Dans le cadre de notre étude nous l'employons pour prendre en compte l'incertitude due à la variabilité des objets que nous manipulons, ainsi que pour inférer des caractéristiques qui vont influencer nos décisions.

Dans notre procédé, nos décisions sont généralement prises en fonction d'un seuil paramétré et d'une valeur calculée. Dans tous les cas il s'agit de savoir si la valeur à tester se situe en dessous ou au-dessus de ce seuil. C'est-à-dire, nous cherchons à comparer une valeur fixée avec une valeur pouvant varier. Cette comparaison engendre en général une décision brutale si l'on considère, pour une variable donnée, la somme des erreurs qui peuvent être cumulées tout au long de notre procédé. À la vue de ce cumul d'erreurs, le résultat d'une comparaison peut ne pas s'avérer judicieux lorsque la valeur est proche du seuil auquel elle est comparée. Ainsi, en de nombreux endroits de la méthode certains principes de la logique floue ont été utilisés de manière à moduler les décisions prises.

Nous avons alors défini un comparateur flou répondant à nos besoins de comparaison, stockage de valeur et d'inférence floue. L'univers du discours que nous avons fixé pour ce comparateur (cf. figure D.1) se décompose en deux zones : A et B. Ce comparateur se définit en fonction d'une valeur $V$ (*i.e.* la valeur utilisée comme seuil fixe) et d'un décalage $d$ autour de cette valeur. Les zones $A$ et $B$ peuvent être considérées comme des termes linguistiques à redéfinir selon ce que nous souhaitons représenter avec ce comparateur, par exemple : "proche" et "éloigné". $B$ est la zone dans laquelle sont considérées les valeurs supérieures à $V$ et $A$ celles qui lui sont inférieures.

En fait, si nous nous intéressons à ce qui est relatif au domaine de variation de $B$, nous pouvons écrire les degrés d'appartenance des valeurs $X_1$, $X_2$ et $X_3$ de la figure D.1 par :

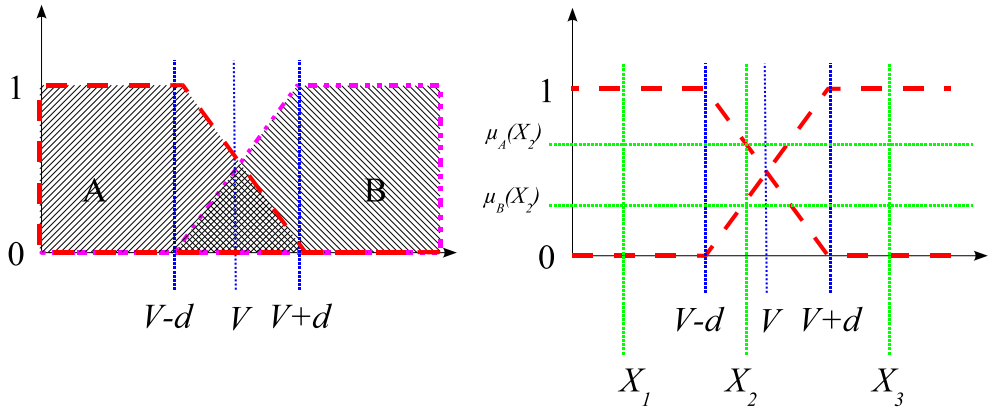



FIG. D.1 – Comparateur flou : $Comp_1 = FuzzComp(V, d)$ centré sur $V$ avec un décalage $d$ autour de $V$.

$$\mu_B(X_3) = 1.0, \mu_B(X_1) = 0.0 \text{ et } \mu_B(X_2) = 0.33$$

Ceci est équivalent à utiliser un opérateur '<' sur l'opérateur flou $Comp_1$ de la figure D.1. Ainsi $Comp_1 < n = m$ pourrait se lire comme : "*la possibilité que* $n$ *soit supérieur à* $Comp_1$ *est de l'ordre de* $m$". Par conséquent nous pouvons réécrire les expressions précédentes par :

$$Comp_1 < X_3 = 1.0, Comp_1 < X_1 = 0.0 \text{ et } Comp_1 < X_2 = 0.33$$

De la même manière si nous nous intéressons à ce qui relatif au domaine de variation de $A$ nous allons utiliser l'opérateur '>' :

$$Comp_1 > X_3 = 0.0, Comp_1 > X_1 = 1.0 \text{ et } Comp_1 > X_2 = 0.66.$$

Les règles que nous allons pouvoir définir avec ces opérateurs flous peuvent être combinées par les opérateurs logiques 'et', 'ou' et 'non', comme en logique propositionnelle [BJ89]. Néanmoins, comme les valeurs floues appartiennent à l'intervalle $[0, 1]$, différents type d'opérations existent permettant de représenter un comportement du 'et', 'ou' et du 'non' flou (respectivement appelés t-norme, t-conorme et négation) qui ne soit pas limité aux valeurs 0 et 1. Les types d'opérateurs les plus fréquents sont définis dans [Bou95], nous y trouvons, par exemple, le type défini par Zadeh (t-norme : $min(u, v)$, t-conorme : $max(u, v)$, négation : $1 - u$) ou le type probabiliste (t-norme : $u.v$, t-conorme : $u + v - u.v$, négation : $1 - u$). Pour les différencier nous notons, par exemple pour le 'et' défini par Zadeh, $\overset{\wedge}{}_{\text{zadeh}}$ et pour le 'ou' : $\overset{\vee}{}_{\text{zadeh}}$. Néanmoins, comme nous utilisons par défaut le type défini par Zadeh, nous n'utiliserons cette notation que lorsqu'il est utile de lever une ambiguïté.

## D.2 Exemple d'application

Dans un premier temps, après l'étage de segmentation d'image et d'ordonnancement des points de contours, les contours sont extraits de l'image à la condition qu'ils soient d'une longueur suffisante et qu'ils séparent une région "feuille" d'une région "non feuille". En utilisant le comparateur flou défini ci-dessus nous pouvons affecter des termes linguistiques à $A$ et à $B$. Sachant que selon la distance de Mahalanobis, plus un point de l'espace couleur est dans une classe couleur donnée, plus sa distance à cette classe est faible.

Ainsi, avec un seuil de tolérance donné, nous pouvons affecter à $A$ le terme linguistique "est une feuille" et à $B$ "n'est pas une feuille".

Nous allons alors chercher à ce que la couleur de la région $r_1$ soit associée au terme linguistique "est une feuille" et que $r_2$ soit associée au terme linguistique "n'est pas une feuille", ou le contraire, *i.e.* $r_2$ soit associée au terme linguistique "est une feuille" et que $r_1$ soit associée au terme linguistique "n'est pas une feuille".

Pour savoir si une frontière séparant la région $r_1$ et la région $r_2$ peut être considérée comme valide nous allons vérifier si cette expression floue est vraie :

$$\mathcal{C}_{14} = \big(\mu_A(col_1) \wedge \mu_B(col_2)\big) \vee \big(\mu_B(col_1) \wedge \mu_A(col_2)\big)$$

avec $A$ et $B$ définies ci-dessus et $col_x$ une fonction permettant de connaître la distance couleur de la région $x$ en fonction de la classe couleur utilisée (*i.e.* celle des feuilles dans notre cas). Nous définissons le comparateur flou $fSD = FuzzComp(seuilDist, 0.75)$ centré sur la valeur $seuilDist$ avec un décalage de $0.75$ (valeur empirique). Avec ce comparateur l'expression précédente est équivalente à :

$$\mathcal{C}_{15} = \big(fSD > col_1 \wedge fSD < col_2\big) \vee \big(fSD < col_1 \wedge fSD > col_2\big)$$

signifiant que $col_1$ doit être inférieure à $seuilDist$ et $col_2$ doit être supérieure à $seuilDist$, *i.e.* $r_1$ peut être considérée comme une région feuille et $r_2$ comme une région non-feuille. Ou, représenté par le symbole $\vee$, l'inverse : $r_1$ n'est pas une région feuille et $r_2$ l'est.

Le test de décision pour un contour discret $Cd$ prend aussi en considération sa longueur qui doit être supérieure à $LongueurMin$. Nous définissons alors un autre comparateur flou : $fMinLength = FuzzComp(LongueurMin, 5.0)$. Le test devient :

$$\mathcal{C}_{16} = \mathcal{C}_{15} \wedge fMinLength < nbPoint(Cd)$$

# E.
# Graphe de distances

De manière à optimiser le processus de recherche de courbes dans l'espace des courbes de Bézier, un graphe de distances et de compatibilité est généré. Ce graphe se base sur la distance entre deux courbes et sur leurs caractéristiques. Pour que deux nœuds (*i.e.* courbes) soient liées dans ce graphe il faut que la distance les séparant ne soit pas supérieure à la longueur de la plus petite des deux courbes, et cela à un rapport $k$ près. Cette remarque nous permet de définir un comparateur flou[1] $fGraphDist = FuzzComp(k, 0.5)$ avec $k = 1.25$, centré sur $k$ avec un décalage de $0.5$ (valeurs empiriques relevées pour obtenir des résultats au mieux) :

$$\mathcal{C}_{17} = fGraphDist > \frac{dist(bez_1, bez_2)}{min(longueur(bez_1), longueur(bez_2))}$$

La distance $dist$ entre deux courbes est la plus courte des distances entre $\|AA'\|$, $\|AC'\|$, $\|CA'\|$ et $\|CC'\|$. La longueur d'une courbe de Bézier de degré 2 est celle définie à la section 3.1.2, p. 21.

De plus, pour que les courbes soient liées il faut qu'elles soient orientées de la même manière vis à vis de l'objet (plante du même côté). L'arête qui les lie est pondérée par la valeur floue obtenue par $\mathcal{C}_{17}$.

[1] cf. annexe D

# F.
# L'environnement graphique *TraitIm*

Cet environnement a été conçu et développé au Cemagref (Gilles Rabatel) pour répondre à des besoins de représentation, stockage et traitement d'images dans différents contextes (prototypes embarqués, UC à faible espace mémoire, mode texte, *etc.*). L'ensemble est basé sur une structure d'image simple permettant d'effectuer des actions basiques comme l'écriture et la lecture de pixels. Cette base a ensuite été dérivée et intégrée pour faire partie d'un environnement graphique (sous Windows) liant la visualisation de plusieurs images et l'incorporation de librairies de traitement d'image (développées pour cet environnement). Ces librairies reprennent des outils essentiels comme l'extraction de contours (Deriche, Laplace), la morphologie mathématique, des fonctions sur les objets binaires, *etc.*.

Cet outil a été créé sous l'IDE Borland 5 et une bonne partie de son code dépendait directement des fonctionnalités spécifiques de la librairie graphique OWL de Borland. Du fait que Borland ne maintient plus ce produit et les librairies qui en découlent, il a fallu opérer des transformations de fond pour minimiser ce qui devait être dépendant de l'environnement OWL. Ainsi, une organisation a émergée comportant une DLL 'graphique' spécifique à au gestionnaire de fenêtres de Windows, une interface 'client' pour utiliser cette DLL de manière transparente et des structures de bases comme la représentation d'une image ou comme le modèle de DLL utilisateur.

La DLL 'graphique' comporte l'ensemble des fonctions nécessaire à l'environnement *TraitIm* comme la création d'une fenêtre image, l'ouverture d'un fichier image, la gestion de paramètres, la gestion du menu dynamique (permettant l'intégration de nouvelles DLL outils), *etc.*. Cette forme permet de ne pas se limiter à une librairie graphique comme OWL et de s'adapter au type de la plate-forme (Windows, Linux, *etc.*) en redéfinissant chaque fonction avec une librairie compatible avec le système ou bien en utilisant une librairie multi-plateforme comme Fox, GTK, ou autre. L'utilisation avec Java a été envisagée mais la compatibilité entre le C++ et Java, à travers les JNI, reste à être bien étudié pour converser un maximum de performances.

La migration vers *Visual C++* et création de cette nouvelle organisation a demandé un certain nombre de semaines mais était nécessaire pour pouvoir travailler rapidement et tester facilement les méthodes conçues. L'utilisation d'autres environnements[1] n'a pas été retenue car la migration de l'existant était plus importante à long terme (plusieurs développements sous *TraitIm* ont déjà vu le jour depuis une dizaine d'année). De plus le besoin d'efficacité et de confort nous a maintenu dans le choix du langage de programmation C++.

Tous les développements ont été conçues pour cet environnement. Ainsi, la segmentation *Union-Find* fait l'objet d'une DLL particulière qui peut être aisément corrigée pour supporter d'autres environnements graphique. D'autres DLL on été créées pour être réutilisées au sein du laboratoire comme pour la librairie de gestion des éléments de géométrie 2D (point, ligne, courbe de Bézier, vectorisation, *etc.*), pour les structures

[1] MathLab, Optimas, package Java JAI, *etc.*

de données génériques (cf. annexe G), *etc.*.

# G.
# Les outils généraux développés

## G.1 La structure de graphe

Comme exposé précédemment une structure de manipulation des graphes était nécessaire. Mais les outils disponibles sur Internet[1] correspondaient à des librairies prévues pour être très génériques et non limitées aux graphes (ce qui complexifie encore plus leurs structures). Ainsi, le code à mettre en place allait prendre des proportions importantes et, en règle générale, la lenteur d'un outil est fonction de sa capacité à être générique. Le choix s'est donc porté sur le développement d'une petite librairie de gestion de graphes.

La structure prend en compte trois classes : $TEdge$, $TNode$ et $TGraph$.

$TEdge$ représente une arête et stocke un pointeur sur un objet (de type paramétré) correspondant à la pondération de l'arête. La classe pondérant l'arête doit définir un certain nombre d'opérations pour permettre des actions classiques, comme l'opérateur de recopie, l'opérateur d'affectation et d'addition (fonctions prenant en paramètre des références).

$TNode$ correspond à un nœud dans le graphe. Cette classe stocke un pointeur sur un objet (de type paramétré) représentant l'information à stocker dans le graphe. De plus, une table de hachage mémorise quels nœuds et quelles arêtes sont adjacentes au nœud courant. La table de hachage offrant un gain de rapidité important par rapport à un arbre binaire de recherche ou à une liste chaînée d'éléments.

$TGraph$ est la classe proposant les fonctions de bases pour ajouter un nœud au graphe, parcourir le graphe par les nœuds ou par les arêtes, *etc.*. Elle stocke aussi la liste des arêtes et celle des nœuds.

Il est à noter que cette structure a aussi été choisie à cause des problèmes occasionnés par le traitement de grandes images. En effet, la segmentation de grandes images entraîne la création d'un nombre de régions très important ($\simeq$ 100000), augmentant, par conséquent, le temps d'ajout et de parcours de l'arbre. De même, cette structure a été choisie du fait que les librairies de graphes[1] ne mentionnaient aucunement la capacité à gérer des gros graphes, et, que la création d'un graphe assez simple (5 nœuds) induisait une bonne quinzaine de lignes de code pas forcement explicites.

Dans un premier temps la table de hachage et les listes provenaient de la librairie *STL* qui est assez rapide en comparaison de la complexité de structure qu'elle comporte. Mais comme la table de hachage ne semblait pas encore assez rapide, une nouvelle structure devant gérer les tables de hachage a été développée.

[1] comme par exemple : la librairie *Boost all* ou bien la librairie *LEDA*.

## G.2 La structure de table de hachage

### G.2.1 Rappel

Petit rappel sur les avantages et inconvénients des différentes structures existantes pour stocker plusieurs données de même type sous le même nom :

**Le tableau** est une structure très employée qui permet l'accès direct à une information si son numéro d'index ($\in [0, n-1]$) est connu ce qui suppose de connaître les données à l'avance. Mais elle pose problème lorsque la taille tableau est inférieure au nombre de données à y mettre. Dans ce cas il faut créer un nouveau tableau plus grand, copier tous les éléments de l'ancien tableau vers le nouveau puis détruire l'ancien. De plus, il est parfois intéressant de stocker une valeur référencée par un index n'appartenant pas à l'intervalle $[0, n-1]$ ce qui n'est pas possible avec un tableau.

**La liste** est une structure d'éléments dont le nombre n'est pas défini à l'avance. Chaque élément stockant un objet (paramétré ou non), un lien vers l'élément suivant s'il existe (voir un lien sur l'élément précédent dans le cas des listes doublement chaînées). Elles peuvent aussi être circulaires, c'est-à-dire : le début est lié à la fin ou/et vice versa. Par contre la recherche d'un élément ne peut se faire qu'en parcourant la liste (de manière linéaire ou dichotomique) jusqu'à l'élément voulu. Pour répondre à l'un des problèmes précédents il est possible d'ajouter dans la structure de l'élément, un champ représentant l'index de référence. Le seul inconvénient des listes en est le parcours (l'ajout et la suppression ne posant pas trop de perte de temps).

**L'arbre** binaire de recherche est une structure comportant trois liens, un sur le père de la branche courante, un sur la branche fille de gauche et un pour celle de droite. L'ajout d'un élément se fait selon la valeur de la valeur ou de l'index de référence, par exemple les valeurs à gauche du nœud courant sont inférieures à la valeur de ce nœud et celles de droites supérieures. Ainsi, pour ajouter ou bien trouver un élément il faut parcourir l'arbre selon les règles indiquées précédemment. Plus l'arbre est grand plus les temps d'accès sont longs.

**La table de hachage** est LA structure de donnée, car elle permet un accès pseudo direct et autorise autant d'éléments que nécessaire. Le stockage des éléments se fait par une valeur d'index (ou clé) représentant en quelque sorte la position de l'élément dans la table. Par exemple pour stocker 100 éléments dont les clés vont de 0 à 9999 il faut allouer un tableau de 10000 cases, la perte de place est alors de 99%. Avec une liste il n'y aurait pas de perte de place mais toujours une perte de temps. La table de hachage se présente donc comme un bon compromis.

Une table de hachage reprend les mêmes idées que les tableaux et les listes, d'ailleurs il s'agit généralement d'une liste de tableaux. Par exemple, les tableaux peuvent stocker une structure comprenant une valeur et la clé associée. Enfin, la position dans le tableau principal est déterminée par l'application d'une fonction de hachage paramétrée, entre autres, par la clé de l'élément à stocker.

L'algorithme ALGO. 7, p. 135 détaille la version développée, au niveau de l'ajout d'un élément à la table. Deux fonctions `Hach1` et `Hach2` ont été définies de manière à être complémentaire et ainsi permettre un meilleur placement. `ExistColTable` renseigne si une table de collision a été définie pour l'élément concerné[2]. `ReHach` est la seule fonction qui est coûteuse : en effet, elle crée une nouvelle table de hachage suffisamment grande pour pouvoir contenir tous les éléments de la table actuelle, puis elle ajoute tous les éléments de l'ancienne table dans la nouvelle, enfin supprime l'ancienne et ajoute le nouveau couple (clef, val).

La structure de donnée se décompose en trois classes : $\_hash\_cell$ qui stocke le couple (clef, val) et un lien vers une éventuelle table de collision puis $\_hash\_table$ qui représente la table de hachage (et la table

[2] comme la plupart des fonctions de hachage peuvent donner pour deux clés différentes la même position dans le tableau, il faut prévoir les cas de collision.

**Algorithme 7** : Ajout à une table de hachage

```
Entrées : une clé (clef) pour indexer l'élément et une valeur (val) à stocker
pos1 = Hach1(clef) ;
si Estvide(table [pos1 ]) alors
    table [pos1 ] = NouvelÉlément(clef, val) ;
sinon
    si ≠ ExistColTable(table [pos1 ]) alors
        table [pos1 ].colTable = NouvelleTable() ;
    pos2 = Hach2(clef) ;
    si Estvide(table [pos1 ].colTable [pos2 ]) alors
        table [pos1 ].colTable [pos2 ] = NouvelÉlément(clef, val) ;
    sinon
        Trouver une case vide i dans table [pos1 ].colTable ;
        si pas de case vide trouvée alors
            ReHach(clef, val) ;
        sinon
            table [pos1 ].colTable [i ] = NouvelÉlément(clef, val) ;
```

de collision) stockant un tableau de $\_hash\_cell$, la taille du tableau, le masque (voir le paragraphe suivant), un lien vers une fonction de hachage ($\_hash\_fun$), *etc.*. Et enfin, $\_hash\_fun$ représentant la fonction de hachage et qui peut être redéfinie en suivant le schéma des deux fonctions qu'elle contient.

### G.2.2 Fonction de hachage

Comme expliqué précédemment, dans cette organisation, les fonctions de hachage sont définies dans les dérivées de la classe $\_hash\_fun$.

**`maxHash`** qui prend comme argument une clé (entier sur 4 octets) et une $\_hash\_table$ (pour le masque et le reste). Cette fonction retourne une valeur de hachage comprise entre 0 et $\infty$.

**`hash`** qui prend comme argument une clé (entier sur 4 octets) et une $\_hash\_table$ (pour le masque et le reste). Cette fonction retourne une valeur de hachage comprise entre 0 et la taille du tableau-1.

Actuellement, l'algorithme de la fonction `maxHash` est celui de ALGO. 8, p. 135.

**Algorithme 8** : Détail de la fonction de hachage

```
Entrées : une clé (clef) pour indexer l'élément et un pointeur (hTable) sur une _hash_table
Sorties : out ∈ [0, ∞]
a = clef & hTable.masque ;
b = (clef ≫ 8) & hTable.masque ;
c = (clef ≫ 16) & hTable.masque ;
d = (clef ≫ 24) & hTable.masque ;
out = a ;
inc = hTable.pas ;
out += b ≪ (inc ≫ 2) ;
inc += hTable.pas ;
out += c ≪ (inc ≫ 2) ;
inc += hTable.pas ;
out += d ≪ (inc ≫ 2) ;
```

Notes : pas et masque sont deux valeurs de type octet. L'opérateur & correspond au $et$ binaire et les opérateurs $\gg$ et $\ll$ correspondent, respectivement, aux opérations de décalage binaire vers la droite et vers la gauche.

En fait, l'algorithme extrait octet par octet de la clef, avec le masque, des bits qui seront placés dans des valeurs temporaires. Elles-mêmes seront multipliées en fonction du pas puis additionnées pour donner un chiffre de placement dans le tableau. Il est à noter que le pas est calculé tel que

$$\texttt{maxHash}(\text{0xFFFF}, \text{hTable}) > \text{hTable.tailleTableau} + \epsilon$$

Au final, la fonction `hash` ne fait que le modulo entre `maxHash` et hTable.tailleTableau.

Pour choisir les masques un petit programme a été fait pour surcharger une table de hachage de nombres aléatoires et afficher le nombre de collisions. Ainsi, les masques choisis sont : 01110111 pour la première fonction et 10011001 pour la deuxième. Il y en a deux car une limitation a été introduite dans l'algorithme. En effet, l'algorithme peut créer une infinité de niveaux de tables de collisions, ici dans l'algorithme ALGO. 7, p. 135 un seul niveau de profondeur est défini (plus ne s'étant pas avéré nécessaire). Donc deux fonctions de hachage ont été définies (une pour le niveau 0 et une autre pour le niveau 1), en fait une seule a été définie et deux masques ont été choisis.

## G.3 La structure de gestion de la mémoire

### G.3.1 Cas pour les objets "classiques"

Par défaut, dans un langage comme le C++, la création d'un nouvel objet puis d'un autre, et encore d'un autre, enfin d'une manière générale la création d'une série d'objets met bien plus de temps que la création en, une fois, d'un ensemble d'objets. De cette observation, un outil de gestion mémoire a semblé approprié pour palier à ce problème.

L'outil va, donc, allouer des blocs mémoire fonction de la taille $t$ des objets à y mettre et du nombre de composantes souhaitées du bloc. Un 'bloc' mémoire est défini pour un seul type d'objet ou tout du moins, pour tous objets ayants la même taille $t$. La taille du bloc étant définie au début de l'allocation (tableau à taille fixe), la création d'un nouvel objet lorsque le 'bloc' est plein induit la création d'un nouveau 'bloc' lié au précédent.

L'allocation d'un objet dans le 'bloc' ne pose pas de problème : un index peut être utilisé pour indiquer la prochaine position libre dans le 'bloc'. Mais la dé-allocation peut être la source de saturation mémoire si la zone mémoire libérée par un objet n'est pas rendue au système ou recyclée dans l'algorithme. Ainsi, il faut pouvoir se souvenir de tout ce qui a été dé-alloué pour l'utiliser plus tard, ce qui sous-entend la gestion d'un petit tableau, de la taille du nombre d'élément dans le bloc, représentant les places vides. Actuellement, ce tableau est géré comme une pile FIFO. De plus, pour permettre de rendre la mémoire d'un objet, il faut savoir à quel 'bloc' et à quelle position se situe l'objet. Ainsi, chaque objet contient un champ désignant le 'bloc' et un autre désignant la position dans le 'bloc'. D'autres choix auraient pu être pris quant à la position de l'objet, comme par exemple en exploitant directement l'adresse de l'objet (décomposition adresse + offset) et en ne désignant que l'adresse du 'bloc'.

La structure développée contient deux classes, l'une décrivant un 'bloc' et l'autre définissant les algorithmes d'allocation et de dé-allocation mémoire. De plus pour pouvoir utiliser cette structure il faut redéfinir les opérateurs d'allocation et de dé-allocation, intégrer la déclaration des deux champs de position de bloc et déclarer un allocateur pour le type de l'objet. Pour cela, deux macros ont été définies,

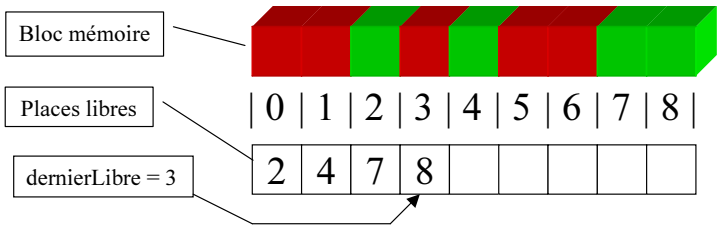



FIG. G.1 – Organisation d'un 'bloc' mémoire.

permettant à l'utilisateur une utilisation assez simple : la première `DECL_ALLOC_STD(TYPE_OBJ)` déclare un allocateur statique (*i.e.* pour être accessible par tous les objets du même type) et la deuxième `DECL_ALLOC_ATTRIB(TYPE_OBJ)` définit les deux champs et les opérateurs d'allocation et de déallocation. Le tout est ensuite complètement transparent pour l'utilisateur.

### G.3.2 Cas pour les objets paramétrés

Pour le cas des objets pouvant prendre des paramètres dont la taille n'est pas connue au départ (template), une deuxième structure, très proche de la précédente, a été mise en place. La seule différence notable tient en la gestion, au sein du même allocateur, de la taille différente des objets. En effet, si pour un même type d'objet, deux objets de taille différente (dû au paramètre) sont à allouer, il ne faut pas les allouer dans le même 'bloc' (car les conséquences, en terme d'écrasement d'un autre objet, pourraient être désastreuses). Ainsi, une table de hachage (hachant des 'bloc's selon la taille de l'objet) a été intégrée pour permettre l'utilisation du bon bloc en fonction de la taille de l'objet.

## G.4 Intégration de ces structures

Dans un premier temps, à des fins de tests, les graphes dans leur version d'origine ont été intégrés dans l'*Union-Find*. Puis, dans un deuxième temps, les tables de hachage de la *STL* ont été remplacées par celles développées et enfin la gestion mémoire a été importée dans les graphes, dans les sous-structures de l'*Union-Find* et dans la structure de l'arbre.

Titre : Reconnaissance d'objets par la génération d'hypothèses de modèles de forme appliquée à l'extraction des feuilles de plantes dans des scènes naturelles complexes.


## Résumé

Afin de réduire l'application de pesticides, de nouvelles stratégies de désherbage se basent sur la caractérisation de la distribution spatiale et de la population d'adventices. Dans ce but, l'identification des adventices dans le champ peut être réalisée par vision numérique. À cause de la complexité de la scène, une connaissance *a priori* de la forme recherchée peut fortement améliorer la segmentation de l'image. Nous proposons une approche basée sur une analyse globale des portions de contours issues de l'image. Comme la forte variabilité d'une scène naturelle induit beaucoup de difficultés dans son interprétation, notre analyse se base sur la modélisation de la forme recherchée. Cette méthode permet la génération et le renforcement d'hypothèses concernant les feuilles présentes dans la scène. Les résultats obtenus nous autorisent l'extraction de feuilles complètes, se chevauchant et partiellement occultées. De plus, le processus que nous avons développé peut être étendu à la reconnaissance d'autres types d'objets.


Title : Object Recognition using Model Matching and Hypotheses Generation Applied to Weed Characterization in Complex Natural Scenes.


## Abstract

New weeding strategies designed to reduce pesticide application rely on the spatial distribution and characterization of weed populations. For this purpose, identification of weeds in the field can be achieved using machine vision. Due to scene complexity, *a priori* knowledge on the shape sought would greatly enhance the image segmentation process. Here, we propose an approach based on a primary analysis of the object boundary portion within the image. As the high variability of natural scene induce many difficulties, this analysis relies on shape modeling, and leads to the generation and the reinforcement of hypotheses about actual leaves in the scene. Valuable results are brought about, allowing us the extraction of full leaves, overlapped and partially occluded ones. In addition, the process we developed can be extended to the recognition of other objects.



Discipline : Informatique.

Mots-clés : reconnaissance de formes organique, hypothèse de présence d'objets, modèle paramétrique, courbe de Bézier, segmentation couleur, vectorisation.



LIRMM : UMR 5506 – 161 rue Ada – 34392 Montpellier Cedex 5 – France

Cemagref : 361, rue J.F. Breton – BP 5095 – 34196 Montpellier Cedex 5 – France