REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLAB DE BLIDA

FACULTE DES SCIENCES
DEPARTEMENT INFORMATIQUE

Mémoire pour l'obtention

Du diplôme de Master en Informatique

Option : Ingénierie du logiciel

THEME

RECONSTRUCTION 3D PAR UN ALGORITHME D'EVOLUTION ARTIFICIELLE

Présenté par

M^elle^ Hamida BENNEGADI

M^elle^ Leila HAMADACHE

Promoteur

Dr A.Oualid DJEKOUNE

Co-promoteur

M^me^ Souhila KAHLOUCHE

à présidente du jury: Mme:
- Benblidia nadjia
- Examinateur: Mr Cherifzahar, Mr Hamoula.

Organisme d'accueil

Centre de Développement des Technologies Avancées (C.D.T.A)

2010/2011

Dédicaces

Parce que le savoir purifie notre âme et parce qu'il demeure le gage de notre prospérité, je dédie ce modeste travail

A la mémoire de mes chers parents qui auraient été très fiers de moi, je pense car ils adoraient le savoir.

A mes frères Khaled, Redouane, Tarek, Hichem, et Naoufel,

A ma sœur Naila,

A ma grande famille, oncles et tantes pour leur soutien moral,

Particulièrement A mon Oncle Khemici Omar qui m'a inspiré dans mon projet,

A mes belles sœurs, à mon beau frère,

A ma belle sœur Hafidha pour sa présence et son aide précieuse,

A mon cher binôme Hamadache Leila avec qui j'ai partagé les peines et les labeurs de la recherche,

A ma grande amie Saidani Mounira, pour notre complicité durant tant d'années.

A toute personne chère à mon cœur,

Et enfin à une prospérité soucieuse de la perfection.

Hamida

Dédicaces

Je tiens à dédier ce modeste travail, qui couronne mon cursus universitaire de Master en informatique à:

- *Mes chers parents pour leur encouragement, que Dieu leur réservent bonne santé et longue vie.*

- *Mon binôme Hamida qui a partagé avec moi les difficultés relatives à la réalisation de ce mémoire.*

- *mes frères Mohamed, Yassine et Walid.*
- *Mon amie intime Samia BOUZIDI pour son soutien moral.*

- *Toute ma grande famille et mes amies.*

Leila

Remerciement

Nous tenons à remercier du fond du cœur Allah le clément et le miséricordieux de nous avoir donné la force, la santé et la volonté pour arriver au bout de notre travail.

Nos sincères remerciements et notre profonde gratitude vont spécialement à notre promotrice Mme Souhila KAHLOUCHE, pour son aide précieuse, ses conseils, et ses encouragements tout au long de ce travail. Nous n'oublierons jamais son précieux apport scientifique, sa patience, sa bienveillance dont nous avant beaucoup bénéficié ainsi que le temps qu'elle nous a consacré et l'attention qu'elle nous a témoignés.

Nos vifs remerciements vont également :

* *Aux membres du jury qui nous font l'honneur d'examiner notre travail.*
* *Au Dr. O.Djekoune membre de l'équipe de « Vision Artificielle » de la Division Productique et Robotique du CDTA pour leur aide à la réalisation de notre travail.*
* *Nos enseignants qui nous ont transmis les connaissances nécessaires tout au long de notre cycle de formation.*
* *A tous ceux et à toutes celles qui nous ont encouragés et aidés, de prés ou de loin, pendant la réalisation de cette thèse.*

Résumé

Résumé

Ce travail a pour objectif, d'utiliser les algorithmes d'évolution artificielle pour la représentation 3D de la scène observée par les caméras et ceci pour guider le robot lors de son déplacement, pour cela, nous avons choisi l'algorithme de mouche dans lequel, le problème de la reconstruction tridimensionnelle à partir des images stéréoscopiques est considéré comme étant un problème d'optimisation, où l'on cherche le meilleur positionnement des points 3D dans l'espace. Pour ce faire, nous devons passer par des étapes incontournables que sont la calibration de la caméra par rapport à un repère lié au robot mobile ATRV2, le calcul du champ de vision commun des caméras gauche et droite, le calcul du gradient et l'application de l'algorithme des mouches.

Summary

This work aims to use artificial evolution algorithms for 3D representation of the scene observed by the cameras, and this to guide the robot as it moves. For this, we chose the fly algorithm in which, the problem of three-dimensional reconstruction from stereo images is considered as an optimization problem, in which we try to found the best position of 3D points in space. To do this we must pass by the essentials steps that are: the calibration of the camera according to the referential attached to ATRV2 mobile robot, the calculation of the common field of view camera left and right, the calculation of gradient and finally, the application of the fly algorithm.

ملخص

هذا العمل يهدف إلى استخدام خوارزميات التطور الصناعي لتمثيل المشهد ثلاثي الأبعاد الملاحظ عن طريق الكاميرات و هذا لتوجيه الروبوا وهو يتحرك لهذا لقد اخترنا خوارزمية الذباب التي تعتبر المشكلة ثلاثية الأبعاد لإعادة الأعمار باستعمال الصور ثنائية الأبعاد المأخوذة عن طريق الكاميرات مشكلة تحسين, حيث نبحث عن أحسن تموضع لنقاط ثلاثية الأبعاد في الفضاء لهذا يجب أن نمر بعدة مراحل أساسية : المعايرة الأساسية للكاميرا المرتبطة بالجهاز المتحرك ATRV2, حساب الحقل المشترك بين حقل الكاميرا اليمين و حقل الكاميرا اليسار, و حساب التدرج وصولا إلي تطبيق خوارزمية الذباب.

Mots clés

Vision artificielle, Reconstruction 3D, Calibrage de caméra, Algorithme évolutionnaire, Algorithme des mouches.

Sommaire

Chapitre II : la vision artificielle

Chapitre III : Conception

Chapitre IV : la réalisation

Liste des figures

Chapitre : I

Chapitre : II

Chapitre : III

Chapitre : IV

Liste des tableaux

Chapitre : II

Chapitre : III

Introduction générale

La vision par ordinateur est une science qui a pour objectif l'automatisation du processus de la perception visuelle humaine. Elle s'intéresse à la construction de systèmes artificiels qui permettent d'obtenir des informations à partir des images, en passant par une chaine complexe de traitement qui commence par l'acquisition des images, et qui procède par la suite à des traitements de bas niveau (filtrage, extraction des indices, etc.), et qui se termine par des traitements de haut niveau (la reconstruction tridimensionnelle, l'interprétation, la reconnaissance des scènes, etc.).

La représentation tridimensionnelle d'une scène observée par une ou plusieurs cameras embarquées sur un robot mobile, permet de fournir des informations sur la profondeur et la position des obstacles présents dans la scène. Ces dernières permettent au robot de se déplacer dans son environnement en évitant les obstacles. Beaucoup d'algorithmes pour la représentation tridimensionnelle ont été proposés. Les différences entre eux sont principalement liées au modèle utilisé pour représenter la scène.

Une nouvelle vision du problème de la représentation tridimensionnelle, considère cette dernière comme étant un problème d'optimisation, qui consiste à faire évoluer artificiellement un ensemble de solutions potentielles tout en favorisant des solutions « meilleures », d'une autre manière en recherchant le meilleur positionnement des points 3D dans l'espace, et ceci pour représenter le mieux possible les points de la scène observée à partir des images.

Les algorithmes évolutionnaires constituent une approche originale : il ne s'agit pas de trouver une solution analytique exacte ou une bonne approximation numérique, mais de trouver des solutions satisfaisant au mieux à différents critères, parfois contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes plus classiques, pour un même temps de calcul.

Le travail qui nous a été confié consiste à apporter une solution au problème de la reconstruction tridimensionnelle à partir d'images acquises par le banc stéréoscopique installé sur le robot ATRV2, disponible au niveau de l'équipe vision artificielle du CDTA, et ceci en utilisant un algorithme évolutionnaire.

Notre travail consiste à implémenter un algorithme qui permet la reconstruction des points 3D par une approche d'évolution artificielle (algorithme des mouches). Pour cela nous présentons ce mémoire qui est composé de quatre chapitres organisés comme suit :

Le premier chapitre présente un état de l'art sur les algorithmes évolutionnaires et leur fonctionnement.

Le deuxième chapitre consiste à présenter la vision artificielle ainsi que les différentes techniques utilisées pour la reconstruction 3D, à savoir le calibrage de caméras et la stéréovision.

Le troisième chapitre est consacré à la mise en œuvre de notre application en expliquant les différentes démarches suivies tout au long de la réalisation de cette dernière.

Le dernier chapitre illustre une description du logiciel réalisé, son mode d'utilisation ainsi qu'une expérimentation et les résultats obtenus.

Enfin, nous terminons par une conclusion générale et des perspectives.

Chapitre 1
Les algorithmes évolutionnaires

CHAPITRE I
LES ALGORITHMES EVOLUTIONNAIRES

I.1 Introduction

Il existe dans la littérature beaucoup de méthodes classiques pour la résolution de problèmes d'optimisation qui cherchent à atteindre l'optimum en résolvant un système d'équations souvent non linéaire, bien que ces méthodes basées sur le calcul ont été développées et améliorées. Une simple réflexion démontre leur manque de robustesse pour les raisons suivantes **[Goldberg, 1994]** :

- Elles s'appliquent localement et atteignent les optimums dans un voisinage de point de départ.
- Elles sont basées sur un calcul dépendant de l'existence de dérivées. Mais dans la pratique, un grand nombre de fonctions à optimiser ne sont pas dérivables et souvent même pas continues.

C'est ainsi que nous sommes amenés à cette intéressante conclusion : quand une performance robuste est souhaitée, la nature fait mieux les choses, c'est donc à travers l'étude de l'exemple biologique qu'un grand nombre de thèses de doctorat et d'articles de recherche établissent la validité des techniques d'optimisation par des algorithmes évolutionnaires. C'est une voie de recherche qui est très active de nos jours. Elle essaye de développer de nouvelles approches s'inspirant des principes de la théorie d'évolution pour traiter les différents problèmes notamment ceux portant sur l'optimisation **[Goldberg, 1994]**.

Les algorithmes évolutionnaires sont les algorithmes, basés sur la population, les plus étudiés. Ils ont résolu avec succès des problèmes d'optimisation difficiles dans divers domaines, et ceci a suscité un très grand intérêt pour le domaine de recherche connu sous le nom de calcul évolutionnaire (Evolutionary Computation ou EC) **[Goldberg, 1994]**.

Ce chapitre décrit les algorithmes évolutionnaires et explique en détails diverses méthodes de sélection et des opérateurs de variation [recombinaison (ou croisement) et mutation].

I.2 Les algorithmes évolutionnaires (AE)

I.2.1 Définition

Les algorithmes évolutionnaires sont des méthodes d'optimisation globale dites stochastiques. Par opposition aux méthodes déterministes, les méthodes stochastiques sont basées sur l'étude de phénomènes aléatoires dépendant du temps.

Les algorithmes évolutionnaires sont basés sur la théorie d'évolution des espèces biologiques, ils utilisent à la fois les principes de la survie des individus les mieux adaptés et ceux de la propagation du patrimoine génétique qui s'inspirent des mécanismes de sélection naturelle et des phénomènes génétiques tel que des mécanismes d'évolution de la nature : croisements, mutations, sélections, etc. Pour utiliser ces algorithmes, il faut disposer d'une population d'individus. Chaque individu dispose d'une chaîne chromosomique qui dirige son comportement. Cette chaîne s'apparente à l'ADN dans les organismes vivants. Comme dans les systèmes naturels, des croisements sont réalisés périodiquement et permettent à l'algorithme de créer la génération suivante d'individus, ainsi des mutations sont aussi effectuées. Ces mutations évitent à l'ensemble de la population de converger vers une solution qui ne serait pas optimale. Il existe plusieurs types de ces algorithmes, mais l'idée essentielle est la même : simuler l'évolution d'une population dans un espace de recherche à l'aide de trois opérateurs: sélection, croisement, mutation.

Malgré la simplicité du processus évolutionnaire, fabriquer un algorithme évolutionnaire efficace est une tâche difficile, car les processus évolutionnaires sont très sensibles aux choix algorithmiques et paramétriques. L'expérience a prouvé que les réussites les plus importantes sont fondées sur une très bonne connaissance du problème à traiter, et une compréhension délicate des mécanismes évolutionnaires **[Terki, 2003]**.

I.2.2 Bref historique des Algorithmes Evolutionnaires

Les Algorithmes Evolutionnaires (AEs) sont des méthodes d'optimisation inspirées par les idées de l'évolution naturelle. C'est donc en biologie qu'il faut chercher les racines de cette discipline.

L'évolution, telle qu'on la connait aujourd'hui, est un processus où les individus doivent s'adapter à un milieu naturel qui impose une concurrence induite par des ressources limitées. L'évolution naturelle peut donc être vue comme un processus d'optimisation ou les individus essayent de maximiser le profit qu'ils peuvent obtenir des conditions naturelles environnantes afin d'assurer leur descendance.

Les premières traces de l'histoire des AEs remontent aux années 70 avec les travaux d'Ingo Rechenberg et Hans-Paul Schwefel sur les stratégies d'évolution **[Rechenberg, 1973]**.

Les individus sont typiquement représentés par des vecteurs de réels, issus directement des variables du problème à résoudre. L'évolution des individus est produite en additionnant aux variables une valeur obtenue à partir d'une variable aléatoire Gaussienne.

La branche la plus populaire est probablement celle des algorithmes génétiques, proposées par John Holland dans les années 70 **[Holland, 1975]**. Ceux-ci sont caractérisés par un codage binaire pour représenter les variables du problème. Il existe des paradigmes similaires, qui en fonction du type de problème, travaillent avec un codage spécial ou utilisent des opérateurs différents. Parmi ceux-ci, on trouve la programmation génétique, développée par Ingo Rechenberg dans les années 60, qui a la particularité de travailler sur un codage d'arbres. Ce codage permet de représenter aisément des espaces de recherche complexes tels que les programmes informatiques. Un autre paradigme proche qui vit le jour à la même époque est la Programmation évolutionnaire proposée par Lawrence Fogel **[Fogel, 1996]**, également dédiée à l'évolution de programmes (voir Figure I.1).

Les algorithmes mémétiques proposées par Pablo Moscato en 1989 **[Moscato, 1989]**, résultent quant à eux de la fusion entre les algorithmes génétiques et les méthodes d'amélioration locale de la solution. Ces algorithmes, parfois appelés algorithmes évolutionnaires lamarckiennes, améliorent localement les individus afin d'accélérer la convergence vers de bonnes solutions **[Maturana, 2009]** (Figure I.1).

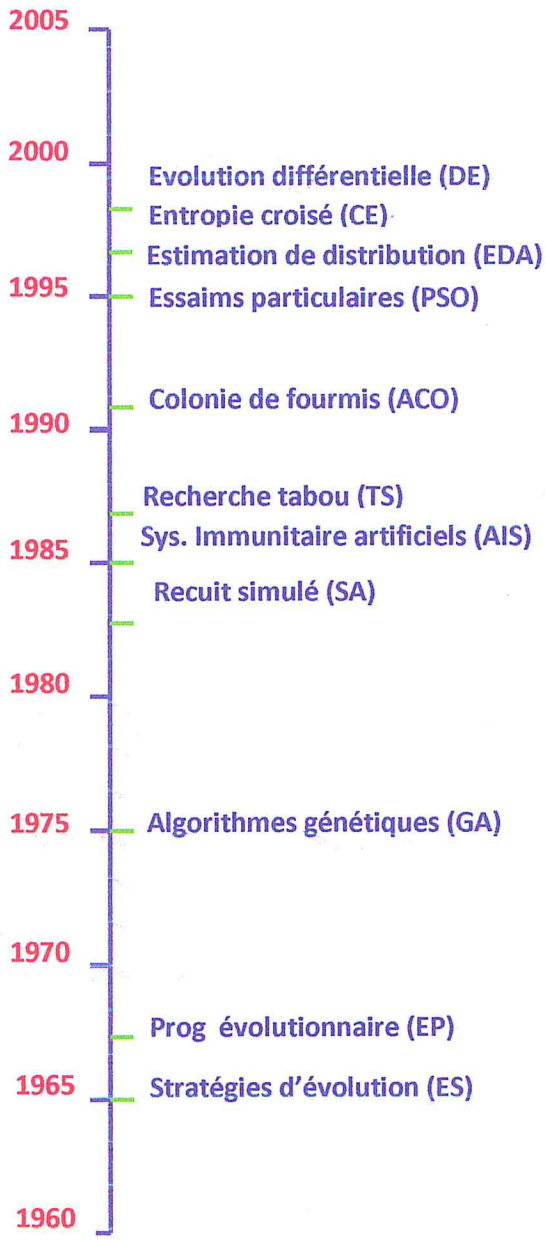


Figure I.1 Développement des différentes méthodes d'optimisation [1]

Le développement chronologique des différentes méthodes d'optimisation est présenté ci après (voir Figure I.1) :

- 1952 : premiers travaux sur l'utilisation de méthodes stochastiques pour l'optimisation.
- 1954 : Barricelli effectue les premières simulations du processus d'évolution et les utilise sur des problèmes d'optimisation généraux.
- 1965 : Rechenberg conçoit le premier algorithme utilisant des stratégies d'évolution.
- 1966 : Fogel, Owens et Walsh proposent la programmation évolutionnaire.
- 1970 : John Horton Conway conçoit le jeu de la vie, l'automate cellulaire le plus connu à ce jour.
- 1975 : travaillant sur les automates cellulaires, Holland propose les premiers algorithmes génétiques.
- 1980 : Smith utilise la programmation génétique.
- 1986 : Farmer, Packard et Perelson travaillent sur les systèmes immunitaires artificiels.
- 1988 : la première conférence sur les algorithmes génétiques est organisée à l'université de l'Illinois à Urbana-Champaign
- 1988 : Koza dépose son premier brevet sur la programmation génétique.
- 1989 : Goldberg publie un des livres les plus connus sur les algorithmes génétiques.
- 1989 : Evolver, le premier logiciel d'optimisation par algorithmes génétiques est publié par la société Axcelis.
- 1989 : le terme algorithme mémétique apparait.
- 1993 : le terme « Evolutionary Computation » (« calcul évolutionnaire » en français) se répand, avec la parution de la revue éponyme, publiée par le Massachusetts Institute of Technology.
- 1996 : Mühlenbein et Paaß proposent les algorithmes à estimation de distribution.
- 1997 : Storn et Price proposent un algorithme à évolution différentielle.
- 2000 : premiers algorithmes génétiques interactifs **[1]**.

I.2.3 Classification des algorithmes

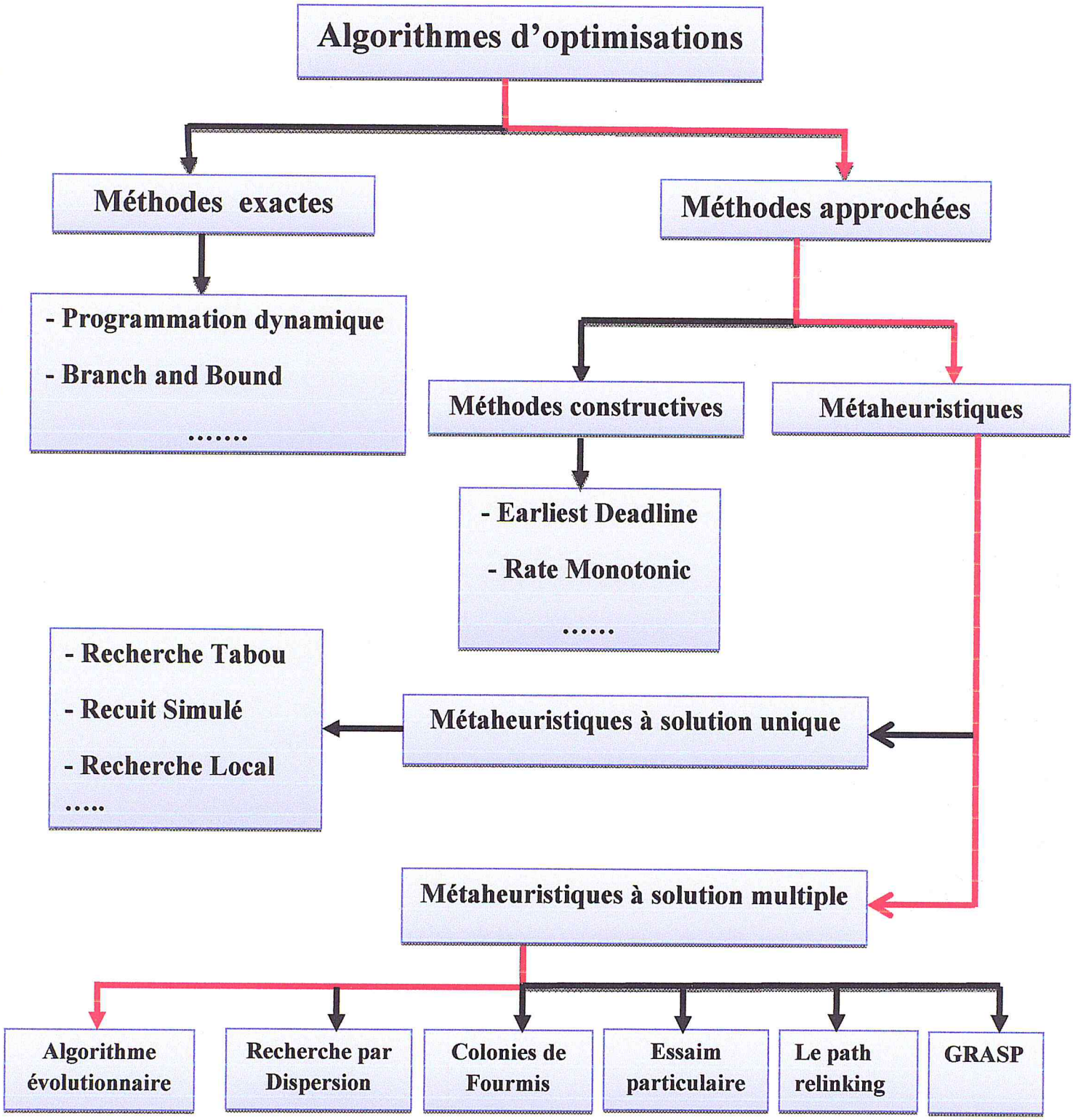


Figure I.2 Classification des algorithmes d'optimisations

Dans le monde de l'informatique, les AEs appartiennent à l'ensemble des métaheuristiques. La métaheuristiques est la traduction du grec, qui veut dire : trouver au delà.

Les méthodes métaheuristiques sont des méthodes stochastiques itératives destinées à résoudre des problèmes combinatoires qui sont difficilement traitables par des méthodes traditionnelles en raison de leur multi modalité, non différentialité ou la taille de leurs espaces de recherche. Ces méthodes sont généralement inspirées du système naturel, soit du point de vue physique comme le recuit simulé, soit du point de vue biologique comme les algorithmes évolutionnaires ou encore, du point de vue comportemental comme les colonies de fourmis. (Voir Figure I.2).

I.2.3.1 Algorithme d'optimisation

I.2.3.1.1 Méthodes exactes

Elles sont basées sur une recherche exhaustive et en évaluant chaque solution **S** de **X**, jusqu'à trouver la solution optimale globale. Plusieurs approches possèdent ce principe de recherche comme : Branch and bound (techniques de séparation et évaluation) et la programmation dynamique. Les méthodes de cette classe sont vite devenues inutilisables pour des instances de grandes tailles **[Boulmerka, 2009]**.

I.2.3.1.2 Méthodes approchées

Les méthodes approchées englobent deux classes :

- Méthodes constructive.
- Métaheuristiques.

1. Méthodes constructive

Leur but est de produire des solutions admissibles de la forme $S = (x_1, x_2, ..., x_n)$ en partant d'une solution vide, et en insérant à chaque étape une composante dans la solution partielle courante selon la décision suivante:

- Quelle est la composante x_i à insérer à l'étape i (son indice) dans la solution courante.
- La valeur que recevra cette composante.

2. Les Métaheuristiques

Les métaheuristiques sont des heuristiques modernes dédiées à la résolution des problèmes et plus particulièrement aux problèmes d'optimisation, qui visent à atteindre un optimum global généralement enfoui au milieu de nombreux optima locaux **[Boulmerka, 2009]**.

Les métaheuristiques englobent deux classes :

- Métaheuristiques à solution unique.
- Métaheuristiques à solution multiple.

2.1 Métaheuristiques à solution unique

- **Recherche Local**: la recherche locale est une métaheuristique utilisée pour résoudre des problèmes d'optimisation difficiles. La recherche locale peut être utilisée sur des problèmes de recherche d'une solution maximisant un critère parmi un ensemble de solutions candidates. Les algorithmes de recherche locale passent d'une solution à une autre dans l'espace des solutions candidates (l'espace de recherche) jusqu'à ce qu'une solution considérée comme optimale soit trouvée ou que le temps imparti soit dépassé **[2]**.

- **Recherche Tabou**: La recherche tabou est une métaheuristique d'optimisation présentée par Fred Glover en 1986. On trouve souvent l'appellation recherche avec tabous en français. Cette méthode est une métaheuristique itérative qualifiée de recherche locale au sens large. L'idée de la recherche tabou consiste, à partir d'une position donnée, à en explorer le voisinage et à choisir la position dans ce voisinage qui minimise la fonction objectif. Il est essentiel de noter que cette opération peut conduire à augmenter la valeur de la fonction (dans un problème de minimisation) : c'est le cas lorsque tous les points du voisinage ont une valeur plus élevée. C'est à partir de ce mécanisme que l'on sort d'un minimum local **[3]**.

- **Recuit Simulé**: Le recuit simulé est une métaheuristique inspirée d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser l'énergie du matériau. Elle est aujourd'hui utilisée en optimisation pour trouver les extrema d'une fonction. Elle

a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Cerny en 1985 **[4]**.

2.2 Méta heuristique à solution multiple

- **Algorithme évolutionnaire**: Les algorithmes évolutionnistes ou algorithmes évolutionnaires sont une famille d'algorithmes s'inspirant de la théorie de l'évolution pour résoudre des problèmes divers. Ils font ainsi évoluer un ensemble de solutions à un problème donné, dans l'optique de trouver les meilleurs résultats. Ce sont des algorithmes stochastiques, car ils utilisent itérativement des processus aléatoires **[5]**.

- **Recherche par Dispersion**: La recherche par dispersion, ou « Scatter Search » en anglais, a été proposée par Glover en 1977. Elle a été proposée dans le cadre de la résolution de programmes mathématiques en nombres entiers. Tout comme les algorithmes génétiques, elle est basée sur une population de solutions (vecteurs d'entiers) qui évolue dans le temps à l'aide à la fois d'un opérateur de sélection, de la combinaison linéaire de solutions de la population pour créer une nouvelle solution provisoire (non forcément entière ou admissible), d'un opérateur de projection permettant de rendre la solution provisoire admissible et d'opérateurs d'élimination. Ainsi, on peut voir la recherche par dispersion comme un algorithme génétique spécial présentant les particularités suivantes :
  - Les vecteurs binaires sont remplacés par des vecteurs d'entiers.
  - L'opérateur de sélection peut élire plus de deux solutions.
  - L'opérateur de croisement est remplacé par une combinaison linéaire convexe ou non convexe de vecteurs.
  - L'opérateur de mutation est remplacé par un opérateur de réparation ou de projection qui ramène la solution nouvellement créée dans l'espace des solutions admissibles **[Gardeux, 2008]**.

- **Colonies de Fourmis**: Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de métaheuristiques d'optimisation. Initialement proposé par Marco Dorigo et al. dans les années 1990, pour la recherche de chemins optimaux dans un graphe, le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes et plusieurs algorithmes ont vu le jour, s'inspirant de divers aspects du comportement des fourmis **[Boulmerka, 2009]**.

- **Essaim particulaire**: L'optimisation par Essaims Particulaires (OEP) est une métaheuristique inventée par Russel Eberhart et James Kennedy en 1995. L'algorithme s'inspire du modèle développé par Craig Reynolds à la fin des années 1980 afin de simuler les mouvements d'un groupe d'oiseau. Cette méthode possède des similitudes avec l'algorithme des colonies de fourmis **[Boulmerka A, 2009]**.

- **Le path relinking**: Path-relinking est une approche évolutive pour résoudre les problèmes d'optimisation. Il combine des éléments de paires de solutions en partant de l'une de ces solutions, une solution appelée initiation, et de générer une trajectoire dans l'espace du quartier qui se connecte à l'autre solution, appelée la solution de guidage. Pendant le chemin de liens, l'objectif principal est d'intégrer les attributs des paires de solutions et d'enregistrer une série de mesures allant de la solution initiée à la solution de guidage **[Renata, 2005]**.

- **GRASP** : La méthode GRASP (pour Greedy Randomized Adaptative Search Procedure) est une métaheuristique simple, développée à la fin des années 90 par Feo et Resende. Elle est adaptée aux problèmes dont les solutions se construisent pas à pas, comme le problème d'ordonnancement de tâches, dont la solution consistera en une certaine suite d'opérations à ordonner sur des machines. Son algorithme contient deux phases : une phase de construction d'une solution ; une phase d'amélioration de la solution qui sera souvent faite grâce à une autre métaheuristique, une recherche locale simple par exemple **[Baptiste, 2006]**.

I.2.4 Les branches des algorithmes évolutionnistes

Plusieurs variante d'algorithmes évolutionnaires ont été développés, donnant naissance à quatre grandes tendances : les Algorithmes Génétiques (ou Genetic Algorithms (GA)), les Stratégies d'Evolution (ou Evolution Strategies (ES)) et la Programmation Evolutive (ou Evolutionary Programing (EP)). Et la Programmation Génétique (ou Genetic Programing (GP)) (voir Figure I.3). De ces quatre méthodes classiques ont dérivé différentes techniques mélangeant les méthodes d'évolution les unes aux autres, leurs classement dans l'une des quatre familles citées ci-dessus n'est pas possible, elles sont néanmoins considérées comme des AEs **[Amrane, 2008]**.

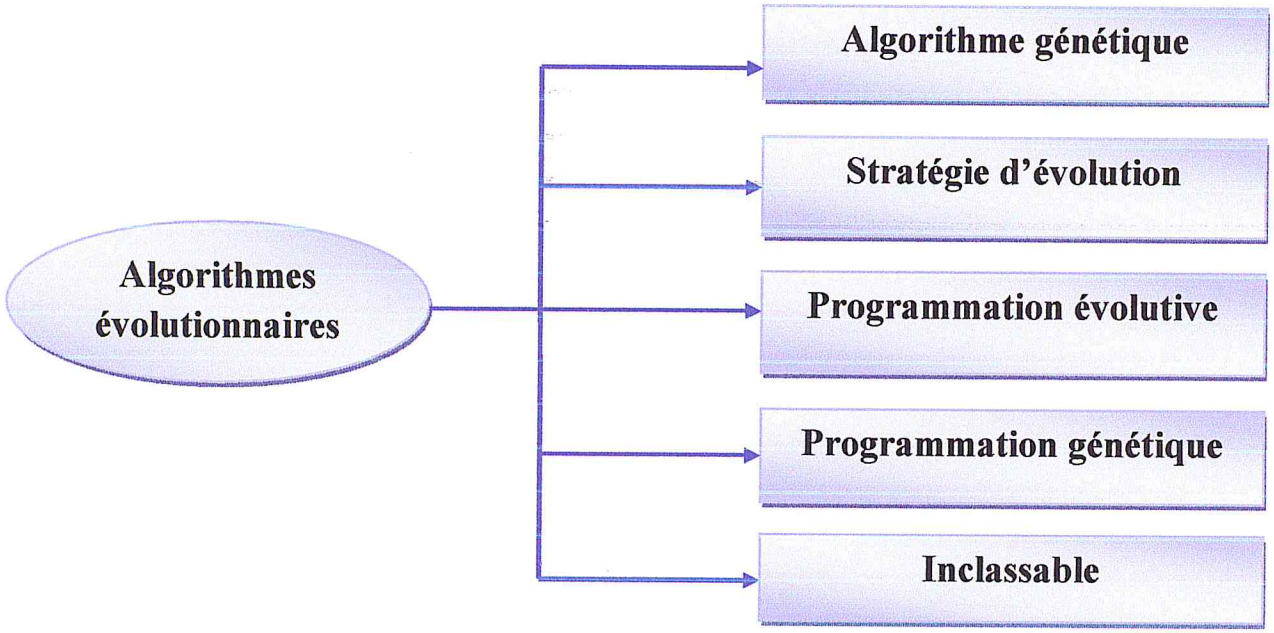


Figure I.3 Les différentes branches des algorithmes évolutionnistes

I.2.4.1 Stratégie d'évolution (SE)

Les stratégies d'évolution (SEs) constituent une technique d'AE développée par I. Rechenberg et H.-P. Schwefel à Berlin dans les années 1960 **[Rechenberg, 1973]**.

Les SEs représentent les individus comme un ensemble de caractéristiques de la solution potentielle. En général, cet ensemble prend la forme d'un vecteur de nombres réels de dimension fixe. Les SEs s'appliquent à une population de parents (de dimension μ) à partir de laquelle des individus sont sélectionnés aléatoirement afin de générer une population de descendants (de dimension $\lambda >= \mu$). Ceux-ci sont ensuite modifiés par des mutations qui

consistent à ajouter une valeur générée aléatoirement selon une fonction de densité de probabilité paramétrée. Les paramètres de la fonction de densité de probabilité, nommés les paramètres de la stratégie, évoluent eux aussi dans le temps selon les mêmes principes que les paramètres caractérisant les individus. Pour former la nouvelle population, μ individus sont choisis (approche (μ, λ)) parmi les μ meilleurs individus des λ descendants, ou (approche $(\mu + \lambda)$) parmi les μ meilleurs individus des μ parents et λ descendants. Les SEs modernes peuvent aussi intégrer une approche multi échelle où des stratégies d'évolution sont imbriquées **[Gagne, 2005]**.

I.2.4.2 Programmation évolutionnaire (PE)

La programmation évolutionnaire (PE) a été développée par L.J. Fogel dans les années 1960 et reprise par D.B. Fogel et d'autres chercheurs dans les années 1990 **[Fogel, 1990]**.

La PE a été initialement conçu dans le but de faire évoluer des machines à états finis et a été par la suite étendue aux problèmes d'optimisation de paramètres. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que de simuler des opérateurs génétiques d'inspiration naturelle. Contrairement aux trois autres AE classiques, la PE n'utilise pas une représentation spécifique mais plutôt un modèle évolutionnaire de haut niveau, jumelé à une représentation et à un opérateur de mutation directement appropriés au problème à résoudre.

Pour effectuer de la PE, une population de μ solutions potentielles au problème est d'abord générée aléatoirement. Chaque individu i de la population produit λ descendants résultant de mutations. Une opération de sélection naturelle est alors appliquée afin de former une nouvelle population de μ individus. Le processus de mutation λ sélection est répété itérativement jusqu' à ce qu'une solution acceptable soit trouvée **[Gagne, 2005]**.

I.2.4.3 Programmation génétique (PG)

La programmation génétique (PG) est un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques basées sur les mêmes principes d'évolution que les AG : des opérations de variation génétique, par des croisements et des mutations, et des opérations de sélection naturelle. La différence entre la PG et les AG réside essentiellement dans la représentation des individus. En effet, la PG consiste à faire évoluer

des individus dont la structure est similaire à des programmes informatiques. La PG a été exprimée formellement par Koza au début des années 1990 **[Kosa, 1994]**.

La PG utilisée par Koza représente les individus sous forme d'arbres, c'est-`a-dire des graphes orientés et sans cycle, dans lesquels chaque nœud est associé à une opération élémentaire relative au domaine du problème. Plusieurs autres représentations comme des programmes linéaires et de graphes cycliques, ont été utilisées. La PG est particulièrement adaptée à l'évolution de structures complexes de dimensions variables **[Gagne, 2005]**.

I.2.4.4 Les algorithmes génétiques (AG)

Les algorithmes génétiques sont des procédures qui s'inspirent des mécanismes de sélection naturelle et des phénomènes génétiques. L'application des algorithmes génétiques pour résoudre un problème d'optimisation fait évoluer une population d'éléments (individus), ou chaque individu représente une configuration (solution) possible, et la population est un ensemble d'individus. Pour chaque individu, on peut mesurer son taux d'adaptation qui est la valeur de la fonction objectif (fitness). Pour l'évolution, on utilise un opérateur de « sélection» qui choisira les meilleurs individus, et des opérateurs génétiques : un opérateur de « croisement » et un opérateur de «mutation» sont appliqués sur les individus pour engendrer de nouveaux individus.

Un algorithme génétique repose donc sur plusieurs notions qui définissent la hiérarchie suivante (voir Figure I.4) :

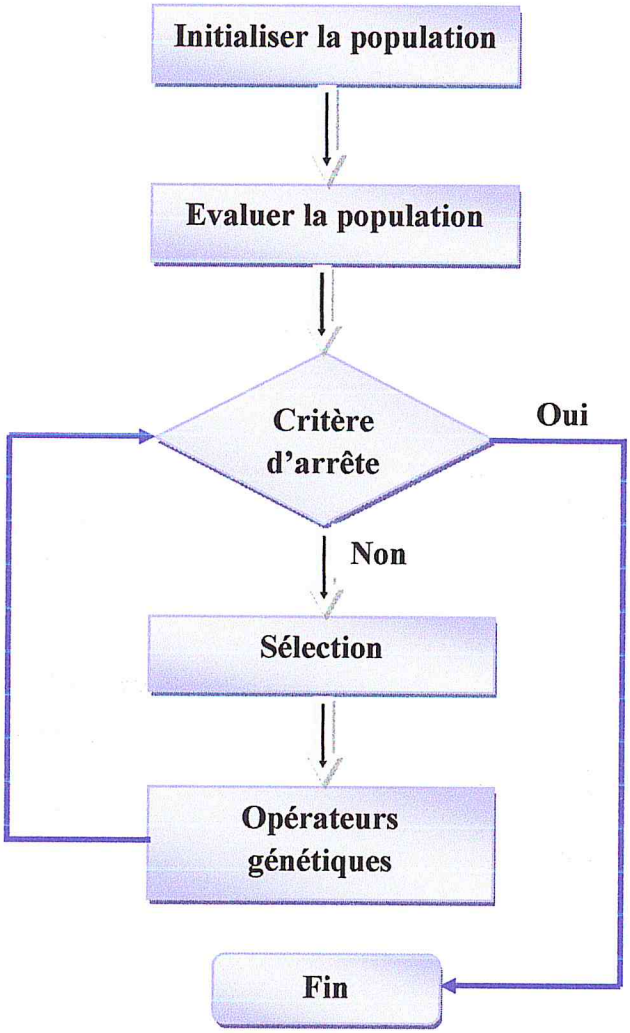


Figure I.4 Le déroulement d'un algorithme génétique

I.2.5 Différence entre les algorithmes évolutionnistes et les algorithmes classiques d'optimisation

Les algorithmes évolutionnistes diffèrent des algorithmes classiques d'optimisation et de recherche essentiellement en ces points :

- Les algorithmes évolutionnistes utilisent un codage des éléments de l'espace de recherche et non pas les éléments eux-mêmes.
- Les algorithmes évolutionnistes recherchent une solution à partir d'une population de points et non pas à partir d'un seul point.
- Les algorithmes évolutionnistes n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité..).

- Les algorithmes évolutionnistes ne sont pas déterministes, ils utilisent des règles de transition probabilistes **[Nguyen, 2007]**.
- Les algorithmes évolutionnistes ont l'avantage de pouvoir résoudre des problèmes difficiles à exprimer mathématiquement. En effet, les algorithmes classiques optimisent les paramètres de problèmes mis préalablement en équations mathématiques. Dans le cas des algorithmes évolutionnaires, ceci n'est pas nécessaire.
- Le grand avantage des algorithmes évolutionnistes est le fait que pour parvenir au résultat, on n'a pas besoin de connaître les caractéristiques de la solution du problème, mais seulement de déterminer parmi deux solutions quelle est la meilleure.

I.2.6 Principe général des algorithmes évolutionnaires

Les algorithmes évolutionnistes (AE) travaillent sur une population d'individus. Ces individus contiennent l'information décrivant la solution qu'ils représentent. Cette information est désignée par le terme de « génotype » par référence à la biologie, alors que la solution qu'ils représentent est nommée « phénotype », l'ensemble des phénotypes constituant le système dont on cherche à optimiser ou concevoir le comportement. Dans cet objectif, les individus subissent un processus de sélection dont les critères sont définis par le concepteur.

Génération après génération, seuls les individus les plus performants survivent et se reproduisent dans le but de générer des descendants plus performants que leurs parents. L'objectif de la version artificielle de l'évolution est de maximiser une fonction d'adaptation. Cette fonction est représentative de la performance d'un individu dans un problème d'optimisation **[Nguyen, 2007]**.

I.2.7 La représentation d'un Algorithme évolutionnaire

I.2.7.1 L'espaces de recherche

. Dans de nombreux cas, l'espace de recherche est totalement déterminé par le problème à résoudre, mais il est toujours possible de transporter son problème dans un espace habilement choisi (changement de "variable") où sa résolution sera plus aisée. Cet espace, où seront appliqués les opérateurs génétiques, est alors appelé **espace génotypique**, et l'espace de recherche initial, dans lequel est calculée la performance des individus, est appelé **espace phénotypique [6]**.

I.2.7.2 Les étapes d'un algorithme évolutionnaires

Le schéma de la Figure I.5 montre les principales étapes d'un AE :

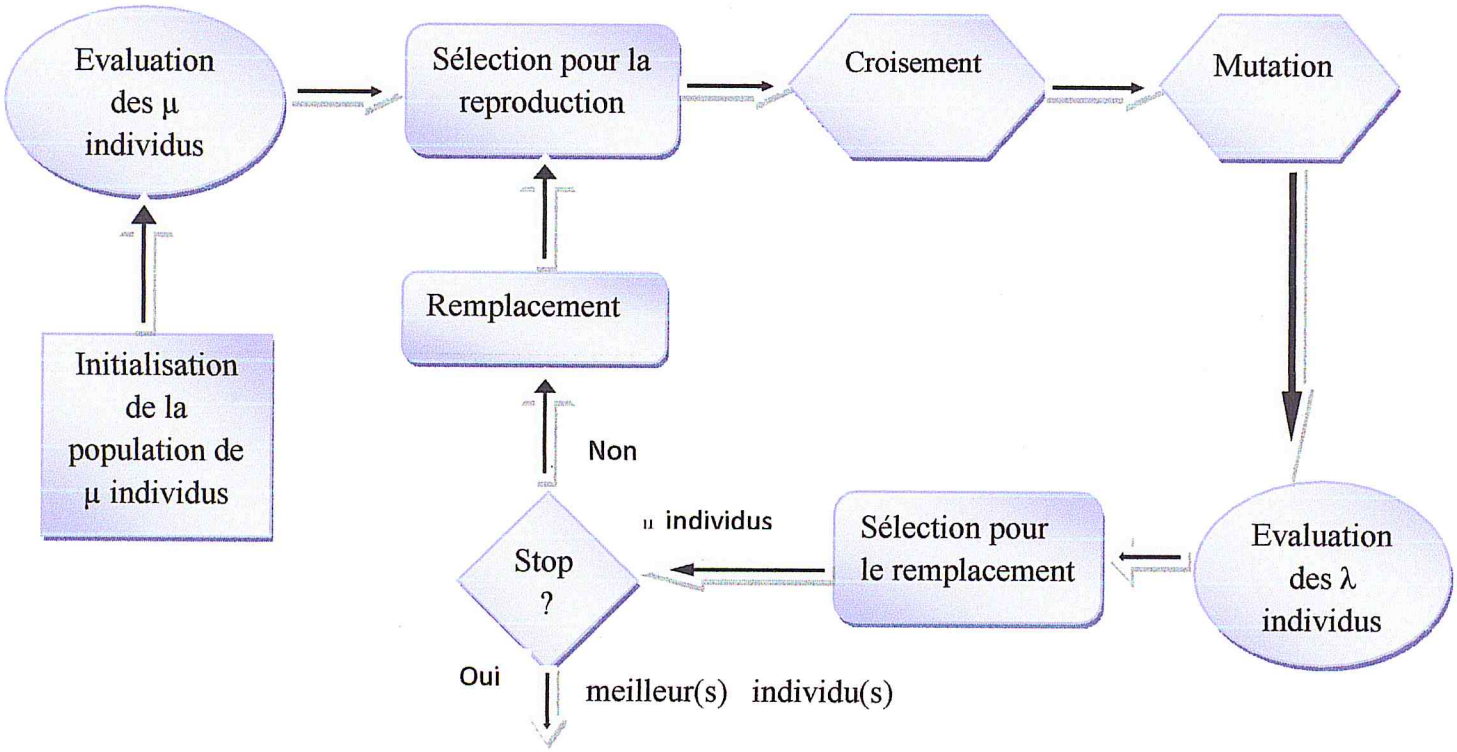


Figure I.5 Schéma général du fonctionnement d'un AE [Schoenauer, 2001]

I.2.7.2.1 Initialisation de la population initiale

Dans l'étape d'initialisation de la population initiale, on génère μ individus de façon aléatoire, dans un intervalle qui dépend bien entendu du problème donné.

I.2.7.2.2 Evaluation des μ individus

Dans cette étape, on évalue chaque individu μ de la population à l'aide de la fonction d'adaptation adéquate (fonction de fitness).

I.2.7.2.3 Sélection pour la reproduction

Le rôle de la sélection consiste à favoriser les individus avec la meilleure évaluation, soit pour devenir des parents, soit pour être conservés dans la population.

Le principe de la sélection pour la reproduction, est de choisir parmi les parents d'une génération, ceux qui vont se reproduire. La sélection pour le remplacement suppose que parmi une population d'individus μ, on applique un certain taux de reproduction. Par exemple, si le taux de reproduction est de 80%, cela voudra dire que 80% des individus μ vont être sélectionnés pour se reproduire. Nous allons tout de suite définir quelques méthodes de sélection **[Costanzo, 2007]**.

A. Sélection par tournois déterministes

Le principe de la sélection par tournois déterministes, consiste à effectuer n tournois de k individus. On suppose que n est le nombre d'individus choisis pour se reproduire. Le principe du tournoi est le suivant : pour chaque tournoi de k individus sélectionnés aléatoirement dans la population d'individus μ, on prend le meilleur. Et on refait la même opération n fois (n tournois). On peut choisir s'il s'agit d'un tournoi avec ou sans remise, à savoir si l'on autorise ou non de remettre un individu qui a remporté un tournoi précédent **[Costanzo, 2007]**.

B. Sélection aléatoire

L'idée de cette méthode est simple : on sélectionne n individus aléatoirement parmi la population d'individus μ pour se reproduire. Cette méthode simple privilégie le phénomène de diversification des solutions. On peut noter qu'en prenant la méthode de la sélection par tournois déterministes, il suffit de prendre comme paramètre $k = 1$ pour ainsi obtenir la sélection aléatoire **[Costanzo, 2007]**.

C. Sélection élitiste

Le principe de la sélection élitiste est aussi très simple. Il suffit de choisir les m meilleurs individus parmi les individus μ. Cette méthode privilégie ainsi le phénomène d'intensification **[Costanzo, 2007].**

I.2.7.2.4 Opérateurs de variation

A l'issu de l'étape de sélection pour la reproduction, on a besoin d'opérateurs de variation pour obtenir de nouvelles solutions qu'on appellera individus λ (enfants). Ces opérateurs se nomment croisement et mutation.

I.2.7.2.4.1 Croisement

C'est un opérateur génétique permettant la création de nouveaux individus enfants à partir d'individus parents. A l'aide de cet opérateur, les parents passent leurs caractéristiques à leurs enfants. Le croisement se fait (généralement) suivant une probabilité de croisement Pc qui doit être élevée pour garantir l'intervention de l'opérateur, relativement la création de nouveaux individus. La définition de cet opérateur est étroitement liée au type de codage utilisé.

On distingue trois types de croisement :

a) Croisement de chaîne de bits :

On va prendre les individus μ (parents) précédemment sélectionnés, pour se reproduire deux à deux et donner des individus λ (enfants). L'idée du croisement est proche de celle d'un brassage génétique, où on mélange les caractéristiques de chacun des individus. La Figure I.6 montre un exemple de croisement en un point. Pour chacun des deux individus, on choisit le même point de croisement. Les caractéristiques qui se trouvent à gauche de ce point pour le premier individu, et les caractéristiques qui se trouvent à droite de ce point pour le deuxième individu, vont être racolées pour donner naissance à un individu λ (figure de droite). On fait de même pour les caractéristiques qui se trouvent à droite du point pour le premier individu, et les caractéristiques qui se trouvent à gauche du deuxième individu. On obtient ainsi un deuxième enfant. Il existe d'autres types de croisements possibles, notamment les croisements en un point, deux points, n points, ou uniforme. Cet opérateur permet en outre de diversifier les solutions **[Costanzo, 2007]**.

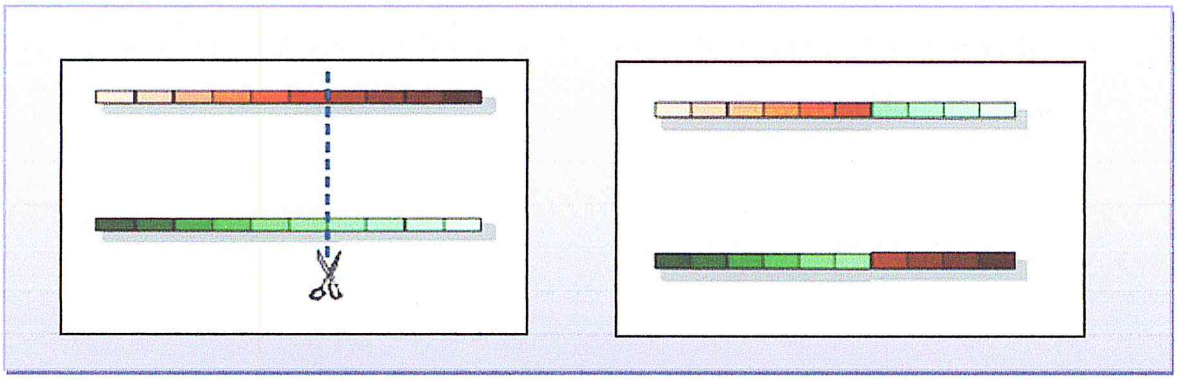


Figure I.6 Un exemple de croisement en un point [Costanzo, 2007]

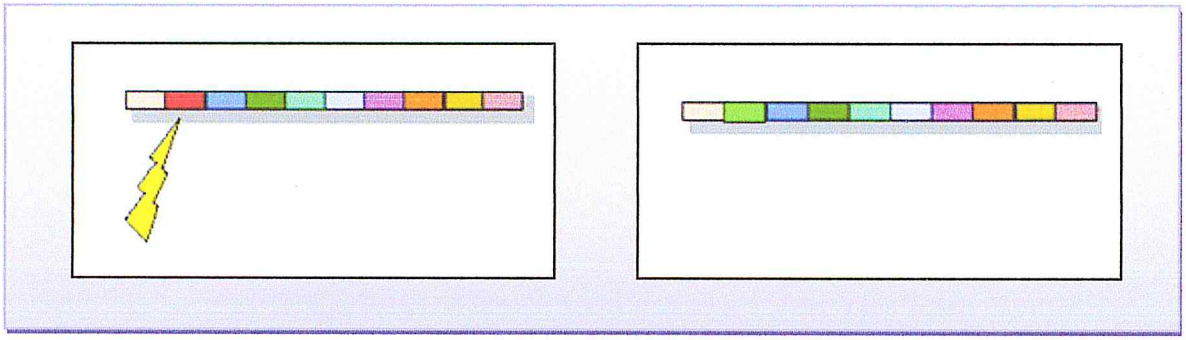


Figure I.7 Un exemple de mutation [Costanzo, 2007]

b) La mutation réelle :

La mutation la plus employée est la mutation Gaussienne. Le principe de cette mutation consiste à ajouter un bruit Gaussien à différentes composantes du vecteur. La difficulté de cette approche est l'ajustement de l'écart type σ du bruit généré.

I.2.7.2.5 Evaluation des individus enfants

On évalue par la suite chacun des enfants produits, en faisant appel à la fonction de calcul de la fitness.

I.2.7.2.6 Sélection pour le remplacement

L'étape de sélection pour le remplacement a pour but de sélectionner μ individus parmi toute la population, afin de constituer la nouvelle génération. Les méthodes de sélection précédemment citées sont aussi ici valables (tournois déterministes, sélection aléatoire, sélection élitiste) **[Costanzo, 2007]**.

I.2.7.2.7 Remplacement

Afin de réitérer le cycle évolutionnaire, on a besoin de définir un mode de remplacement pour la nouvelle population. Celui-ci va dépendre bien entendu de l'étape de sélection pour le remplacement.

Dans le cas de vecteurs réel, un exemple classique d'opérateur de croisement est le croisement qui recombine deux parents X et Y pour créer un enfant Z, par une combinaison linéaire qui peut être soit :

- commune à toutes les coordonnées du vecteur: $\mathbf{Z = (\alpha X + (1-\alpha) Y), \alpha = U([0,1])}$ (I.1)
- Par coordonnées: $\mathbf{\forall i \in \{0... n\}, Zi = (\alpha i .xi + (1-\alpha i) yi) / \alpha i = U([0, 1])}$ (I.2)

c) Croisement d'arbres :

Le croisement étant un échange entre deux ou plusieurs parents du patrimoine génétique, il se traduit dans le cas des représentations à base d'arbres, par un échange de sous-arbres. Le choix des sous-arbres est typiquement aléatoire, toutefois certaines précautions doivent être prises pour éviter la dégénérescence de la taille des arbres obtenus **[Koza, 1992]**.

I.2.7.2.4.2 Mutation

La mutation est un opérateur unaire, dont le rôle est de changer arbitrairement, avec une faible probabilité Pm, les composants xi des solutions contenues dans la population courante. Un tel opérateur simule les accidents occasionnels qui interviennent au cours d'un processus d'évolution naturelle. Son application est utile pour prévenir les blocages dans des minima locaux de la fonction objective et donner une impulsion suffisante à une population qui aurait convergé prématurément. La probabilité de mutation est généralement maintenue constante tout au long de l'algorithme.

a) La mutation binaire :

Le principe de l'opérateur de mutation est simple. Cela consiste à faire muter un certain nombre d'individus (généralement défini par une probabilité de mutation), en changeant de légères caractéristiques.

La Figure I.7 illustre un exemple de mutation. Sur la figure, si on considère qu'on traite par exemple un tableau de couleurs, l'opérateur de mutation consisterait ici à remplacer le deuxième élément du tableau par une autre couleur. Cet opérateur permet également de diversifier les solutions. Le plus souvent cet opérateur est couplé avec l'opérateur de croisement, c'est à dire qu'on applique la mutation sur les individus λ produits par le croisement. Mais rien n'oblige de cumuler avec l'opérateur de croisement **[Costanzo, 2007]**.

A. Le remplacement générationnel (μ, λ) :

Il s'agit de remplacer dans la génération suivante, tous les individus μ (parents) par tous les individus λ (enfants) précédemment sélectionnés **[Costanzo, 2007]**.

B. Le remplacement (μ + λ) :

La population de la génération suivante sera constituée d'individus choisis parmi toute une population entière (individus μ+ λ) **[Costanzo, 2007]**.

C. Le remplacement stationnaire (Steady-State) :

Pour éviter de faire varier trop vite la population dès la première génération, le principe du remplacement stationnaire est de remplacer un très peu nombre de parents dans la génération suivante, dans le but de maintenir une population homogène au fil des générations **[Costanzo, 2007]**.

I.2.7.2.8 Le critère d'arrêt

On dit qu'un processus d'optimisation basé sur l'algorithme évolutionnaire a convergé si la population actuelle est homogène. Une population est homogène si la totalité ou la majorité des individus portent les mêmes gènes ou des gènes dont les valeurs se rapprochent.

Il existe d'autres critères d'arrêt de l'algorithme comme :

- L'algorithme s'arrête quand le nombre de génération est atteint ;
- L'algorithme s'arrête quand un temps spécifique s'écoule ;
- L'algorithme s'arrête s'il n'ya pas un changement des valeurs fitness d'une population pour un nombre de génération spécifié;
- L'algorithme s'arrête quand la population n'évolue plus suffisamment, etc.

I.2.8 Domaines d'application

Les algorithmes évolutionnaires permettent de résoudre non seulement des problèmes purement théoriques en combinatoire, en économie, en apprentissage, dans la théorie des jeux, mais aussi des problèmes liés à des applications réelles complexes. Ainsi, ils sont par exemple utilisés pour analyser des sondages de sous-sol et détecter des champs pétrolifères, pour fabriquer des emplois du temps, pour prévoir les cours de la bourse (nombreuses applications

financières), pour contrôler les pipe-lines de gaz (aux Etats-Unis), dans la conception des automobiles, en logistique (meilleure solution actuelle au problème du voyageur de commerce, avec une approche couplant algorithmes évolutionnaires et recherche opérationnelle), pour optimiser les ailes d'avion, les empennages de missiles supersoniques, les aubes de turbines, les hélices, les tuyères de propulseurs, les manœuvres des avions de combat, les allocations de routes aériennes, les allocations dynamiques de fréquences en téléphonie mobile (meilleur résultat actuel), le positionnement d'antennes, le routage dans les réseaux, l'analyse d'images médicales, la trajectoire de robots, la recherche de gènes responsables de maladies génétiques, l'approximation de formes 2D par des fractales (pour compression fractale d'image)**[7]**.

I.3 Conclusion

Nous avons cité dans ce chapitre les principaux et concepts notions des algorithmes évolutionnaires. En outre, nous soulignons que les AE trouvent maintenant des applications dans des domaines aussi diversifiés que l'informatique, voire le traitement des signaux, les sciences sociales, le domaine militaire et le domaine médical, etc.

Les algorithmes évolutionnaires ont pour principal avantage d'explorer très largement l'ensemble des solutions possibles. Ainsi, ils se font moins facilement piéger par des optima locaux que les algorithmes d'optimisation classiques. Par contre, ce genre d'algorithme est très coûteux en temps de calcul, les paramètres comme la taille de la population et la fonction d'évaluation sont difficiles à établir.

Dans le chapitre suivant, nous allons présenter le domaine de la vision artificielle dans le but d'appliquer les AE, pour une application de reconstruction tridimensionnelle.

Chapitre II
La vision artificielle

CHAPITRE II
LA VISION ARTIFICIELLE

II.1 Introduction

La vision par ordinateur ou vision artificielle est la science de la vision des machines qui a pour but l'automatisation du processus de la perception visuelle humaine. C'est une discipline scientifique qui s'intéresse à la construction de systèmes artificiels qui permettent d'obtenir des informations à partir d'images (voir Figure II.1). Les données d'entrée peuvent prendre de nombreuses formes : photographies, séquences vidéo, images de caméras multiples ou données multidimensionnelles d'un scanner médical, etc. elle est constituée des sous domaines qui sont par exemple : la reconstruction de scènes, la détection d'évènements, la reconnaissance d'objets, l'apprentissage et la restauration d'images. Ses applications s'étendent à des activités diverses : robotique, industrie, médecin, science de l'espace, de la terre, de la mer, etc. **[Mostefaoui, 1996]**.

La vision par ordinateur est une longue chaîne de traitements complexes constituée de deux phases :

- **Acquisition de l'image**: représente le type de capteur d'acquisition utilisé.
- **L'amélioration, traitement, et Interprétation de la scène.**

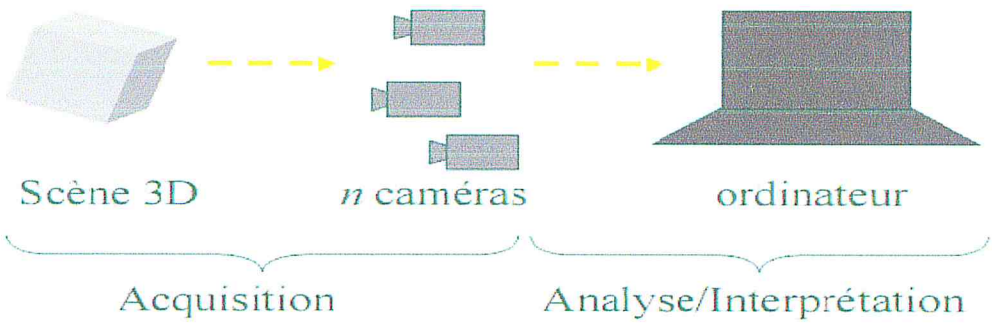


Figure II.1 Vision par ordinateur

II.2 Traitement des images

Le traitement d'images est un ensemble automatisé qui permet, à partir d'images numérisées de produire d'autres images numériques ou d'en extraire de l'information.

Le traitement d'images est souvent synonyme d'amélioration des images pour l'obtention d'une plus grande lisibilité, et de mise en évidence de l'information pertinente déjà présente dans l'image.

II.2.1 Système de traitement d'images

Un système de traitement numérique d'images est composé de :

II.2.1.1 Acquisition des données images

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système.

Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation. Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales : les caméras numériques et les scanners **[Tabari, 1999]**.

II.2.1.2 Dispositifs de numérisation d'images

La numérisation est la conversion d'un signal vidéo/image en une suite de nombre permettant de représenter un objet sur une machine. Suivant l'objet ou le document à numériser et le domaine d'application dans lequel l'image va être utilisée, il existe divers dispositifs de numérisation d'images allant du simple scanner à main au satellite de télédétection.

II.2.1.3 Prétraitements et post-traitements

Les prétraitements concernent les images acquises tandis que les post-traitements concernent les images traitées. Dans les deux cas, les images ne peuvent échapper aux effets de dégradations dus essentiellement aux phénomènes physiques tels que :

- La diffraction (déviation) du système optique.
- Le flou dû au mouvement de l'image durant son acquisition.

Pour pallier à ces dégradations on utilise en général le filtrage.

II.2.1.4 Filtrage numérique [Hadallah, 1997]

Pour améliorer la qualité visuelle de l'image, on doit éliminer les effets des bruits (parasites) en lui faisant subir un traitement appelé filtrage.

Le filtrage consiste à modifier la distribution fréquentielle des composantes d'un signal selon des spécifications données.

Parmi ces filtres, nous distinguons : les filtres passe-bas (lissage), filtres passe-haut (accentuation), filtres passe-bande (différenciation) et filtres directionnels.

II.2.1.5 Traitement numérique des images

Nous pouvons citer les traitements suivants :

- **La Convolution**

  La convolution est le remplacement de la valeur d'un pixel par une combinaison linéaire de ses voisins. Elle consiste à faire balayer une fenêtre (masque) sur l'ensemble des points de l'image.

- **La Segmentation**

  La segmentation d'image est une opération qui a pour but de rassembler des pixels entre eux suivant des critères prédéfinis. Les pixels sont ainsi regroupés en régions, qui constituent une partition de l'image, Il peut s'agir par exemple de séparer les objets du fond **[Hadallah, 1997]**.

Il existe de nombreuses méthodes de segmentation, regroupées en quatre principales classes :

- Segmentation fondée sur les régions. On y trouve par exemple : la croissance de région, décomposition/fusion.
- Segmentation fondée sur les contours.
- Segmentation fondée sur une classification ou le seuillage des pixels en fonction de leur intensité.
- Segmentation fondée sur la coopération entre les trois premières segmentations.

II.3 La caméra CCD

Depuis les années 70, les caméras CCD (Charge Coupled Device) ont été déterminantes dans l'évolution de la vision: la rapidité d'acquisition, la robustesse et la miniaturisation sont autant d'avantages qui ont facilité leur intégration. Elles sont très performantes en termes de portée, précision et quantité d'informations exploitables.

Elles sont de plus, les seules capables de restituer une image sensorielle de l'environnement la plus proche de celle perçue par l'être humain. C'est sans doute le capteur fournissant la plus grande variété d'informations à partir d'une seule acquisition. En effet, en appliquant divers algorithmes de traitement d'images, il est possible d'extraire des éléments caractéristiques de l'environnement comme des segments de droite, des plans, mais aussi des balises et des couleurs.

En revanche, l'inconvénient majeur de tels capteurs se situe d'abord au niveau de la gestion du flux important de données exploitables (traiter une image demeure une opération délicate et surtout coûteuse en temps de calcul), ensuite à leur sensibilité aux conditions d'éclairage.

Pour augmenter le champ de vision de ces caméras, on peut les associer soit avec d'autres caméras ou soit à des systèmes de réflexion de type miroir : il s'agit des techniques de vision omnidirectionnelle. Cette association permet d'obtenir une vue sur 360 degrés de l'environnement **[Drocourt, 2002]**.

Utilisée seule, une caméra CCD ne peut fournir qu'une information 2D. Les techniques qui permettront d'obtenir des informations 3D à partir d'un tel capteur sont généralement liées à l'adjonction d'un autre capteur, c'est le cas de la stéréovision.

II.4 La stéréovision

Le principe de la stéréovision consiste à observer une même scène avec deux caméras qui sont : la caméra gauche et la caméra droite (Figure II.2).

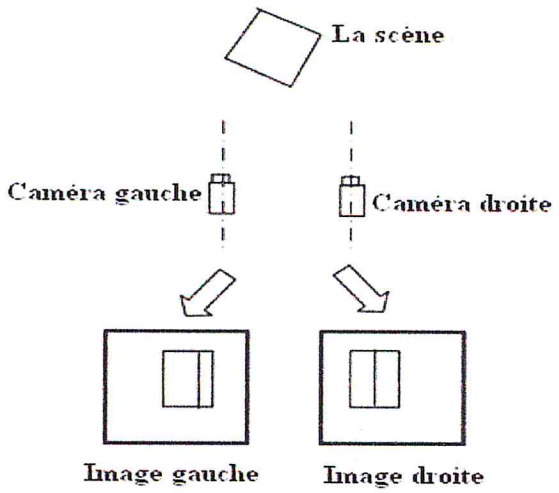


Figure II.2 La vision stéréoscopique

La stéréovision a pour but de reconstruire une scène 3D à partir de deux images bidimensionnelles représentant des prises de vue d'une même scène sous des angles légèrement différents **[El Zaart, 1996]** (voir Figure II.3). Connaissant la géométrie exacte du système stéréoscopique, la première étape de la reconstruction 3D consiste à mettre en correspondance les deux images. Cette phase réside dans la détermination de couples de points observés dans les deux images, ou dans l'appariement de points d'intérêt **[Luong, 1992].**

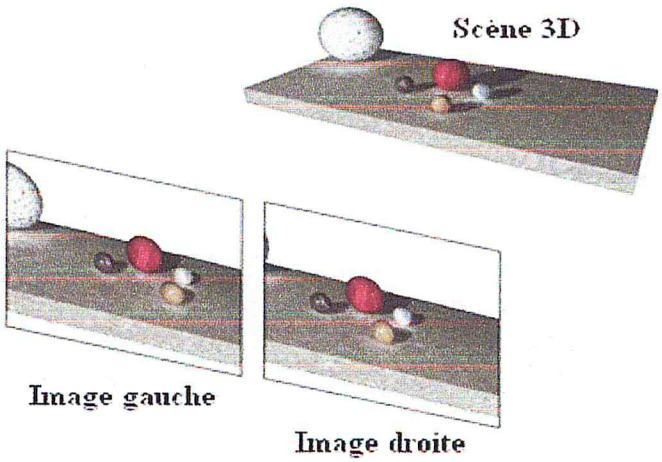


Figure II.3 Scène 3D reconstruite par la vision stéréoscopique

La stéréovision se compose de la séquence d'opérations suivante (Figure II.4) :

- Calibration du système de vision ;
- Rectification de la paire d'images ;
- Mise en correspondance (Appariement ou Corrélation) ;
- Reconstruction 3D.

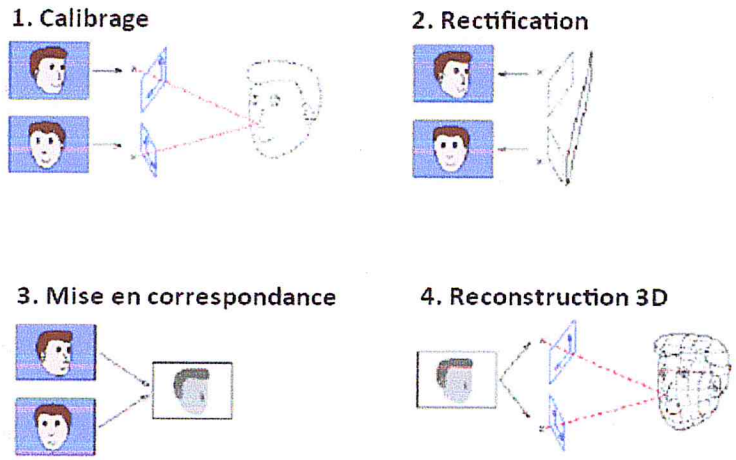


Figure II.4 Les opérations de la stéréovision

II.4.1 Calibration

La phase de calibration de la caméra est considérée comme une étape importante pour de nombreux algorithmes et applications de la vision par ordinateur, en particulier les systèmes qui s'intéressent à récupérer l'information 3D perdue lors de l'acquisition des images. La calibration revient à déterminer de manière analytique la fonction qui associe à un point de l'espace tridimensionnel sa projection dans l'image 2D donnée par la caméra.

II.4.1.1 Calibrage d'une seule caméra [Crouzil, 1997]

De nombreuses applications ne nécessitent pas un calibrage complet de caméras, mais, dès que nous souhaitons effectuer une reconstruction tridimensionnelle, il s'avère nécessaire d'effectuer ce calibrage. Le calibrage consiste à obtenir un modèle approché de la caméra.

Une caméra est un capteur sensible au rayonnement lumineux émis par l'environnement. Elle est définie comme étant un moyen pour obtenir une image 2D à partir d'un objet 3D.

Un modèle de caméra est souvent défini par des paramètres intrinsèques et extrinsèques. Les paramètres intrinsèques d'une caméra modélisent ses caractéristiques internes et ne dépendent donc pas de sa position ni de son orientation dans l'espace. Les paramètres extrinsèques, quant à eux, relient le système de coordonnées de la caméra au système de coordonnées du monde par une série de transformations 3D (rotation et translation). Ils représentent la position et l'orientation de la caméra par rapport à un repère fixe (repère du monde).

Le but du calibrage consiste à établir une relation mathématique entre les coordonnées 3D du monde réel et leurs coordonnées 2D correspondantes sur une image issue de la caméra. Cette relation est généralement obtenue sous forme d'une matrice, appelée matrice de transformation 3D-2D qui contient les paramètres intrinsèques et extrinsèques. Une fois cette matrice calculée, l'information 3D peut être inférée à partir des coordonnées 2D et vice-versa. La calibration est donc une opération prérequise à toute application pour laquelle cette relation entre images 2D et monde 3D est nécessaire.

II.4.1.2 Calibrage d'un capteur stéréoscopique

Dans la vision par ordinateur, le calibrage d'un capteur stéréoscopique est un cas particulier des méthodes de calibrage géométrique d'une seule caméra. En effet, chaque caméra est calibrée distinctement en utilisant une méthode de calibrage d'une seul caméra, le résultat de cette opération est l'obtention de paramètres intrinsèques propres à chacune des caméras qui globalement resteront stables durant le reste du processus, et des paramètres extrinsèques propres à chaque caméra par rapport au même repère qui est le repère du monde.

Les deux calibrages vont donner deux matrices de projection perspective M_L et M_R à partir desquelles nous pouvons construire les différentes transformations (repère absolu/repère caméra, repère caméra/repère rétinien et repère rétinien/repère image) pour les deux caméras. Nous calculons ensuite la matrice $A_{L->R}$ qui décrit la transformation rigide permettant de passer du repère caméra gauche au repère caméra droite en utilisant la relation suivante :

$$A_{L\to R} = A_R {A_L}^{-1} \qquad \text{(II.1)}$$

Où A_L et A_R sont respectivement la transformation repère absolu/repère caméra gauche et droite. La figure suivante (Figure II.5) décrit la géométrie du capteur stéréoscopique.

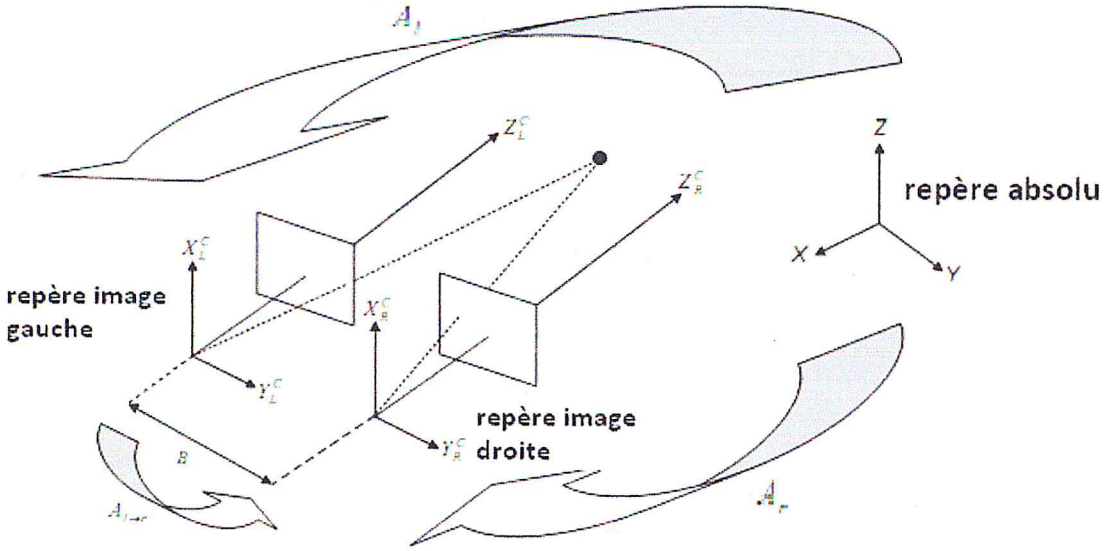


Figure II.5 Géométrie du capteur stéréoscopique

II.4.1.2.1 Modèle géométrique de la caméra [Crouzil, 1997]

Une caméra réalise une transformation ponctuelle qui fait passer d'un point physique de l'espace réel 3D à un point 2D sur le plan image de celui-ci. Ce qui revient à une transformation mathématique de R^3 vers R^2.

On suppose généralement que la transformation réalisée est une projection centrale par rapport au centre de la caméra : c'est le modèle Sténopé (ou Pinhole en anglais).

Le modèle sténopé se caractérise par un plan de projection (le plan image) et un centre de projection (le centre optique F). La Figure II.6 montre qu'un point B de la scène se projette sur le plan image en un point b qui est l'intersection de la droite (FB) avec le plan image.

Avec ce modèle, le processus d'obtention d'une image peut être décrit de manière synthétique par la matrice de projection perspective (ou encore matrice de transformation 3D-2D). Il s'agit en fait de la matrice qui décrit la transformation qui permet de passer des coordonnées de B exprimées par exemple en millimètres dans un repère absolu tridimensionnel, à sa projection b sur le plan image décrite par ses coordonnées exprimées en pixels dans un repère bidimensionnel lié au plan image.

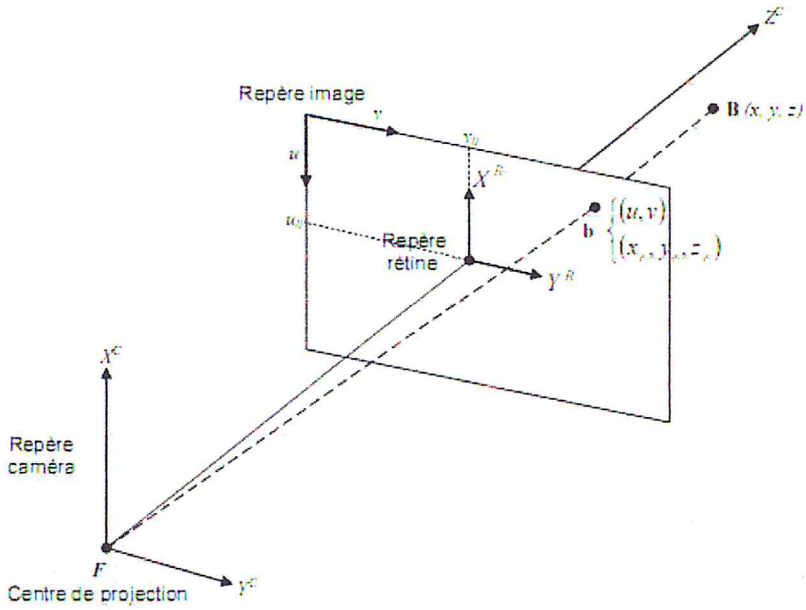


Figure II.6 Modèle sténopé d'une caméra

Si les coordonnées de B dans le repère absolu sont (x, y, z) alors les coordonnées (u, v) de sa projection b dans le repère image sont obtenues par une transformation linéaire en coordonnées homogènes qui s'écrit sous la forme matricielle suivante :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \text{(II.2)}$$

$$\text{avec : } M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \qquad \text{(II.3)}$$

M est la matrice de projection perspective. Elle permet de passer des coordonnées d'un point dans le repère absolu aux coordonnées de sa projection dans le repère image. Elle peut être décomposée en d'autres transformations à savoir (voir figure II.7) :

- Une transformation qui permet de passer des coordonnées d'un point dans le repère absolu (repère scène) aux coordonnées de ce même point dans un repère lié à la caméra, les paramètres de cette transformation sont appelés les paramètres extrinsèques de la caméra.

- Une transformation qui permet de passer des coordonnées d'un point de la scène dans le repère caméra aux coordonnées de sa projection dans le repère image, les paramètres de cette transformation sont appelés les paramètres intrinsèques de la caméra.

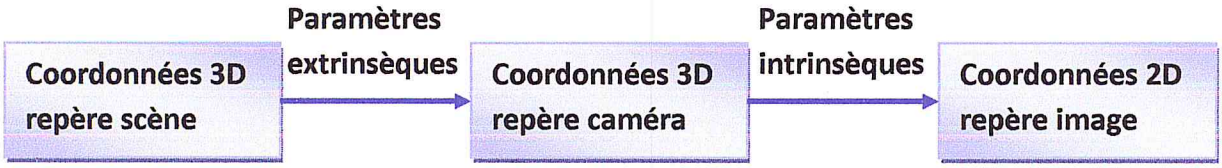


Figure II.7 Schéma illustrant les principales phases du calibrage d'une camera

1. Transformation repère absolu/ repère caméra

Soit (x, y, z) les coordonnées d'un point dans le repère absolu et (x_c, y_c, z_c) ses coordonnées dans le repère caméra. Le déplacement entre ces deux repères est constitué d'une rotation R et d'une translation T.

La transformation entre ces deux repères peut s'écrire sous la forme suivante :

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad \text{(II.4)}$$

où en coordonnées homogènes :

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \text{(II.5)}$$

2. Transformation repère caméra/ repère image

Dans le repère caméra, la projection centrale d'un point de l'espace sur le plan rétinien est:

$$\begin{cases} x_r = f\, x_c / z_c \\ yr = f\, y_c / z_c \end{cases} \quad \text{(II.6)}$$

Ce système d'équations peut être réécrit sous forme matricielle en utilisant les coordonnées homogènes :

$$\begin{pmatrix} sx_r \\ sy_r \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \quad \text{(II.7)}$$

Les points images sont mesurés en pixels dans un repère bidimensionnel associé à l'image. Afin de pouvoir écrire la matrice de transformation du repère caméra au repère image, nous devons introduire les paramètres suivants : u_0 et v_0, coordonnées de l'intersection de l'axe optique avec le plan image (mesurées en pixels), ainsi que k_v et k_u qui sont respectivement le facteur d'échelle vertical et horizontal (pixels/mm) car les pixels d'une caméra sont rarement carrés.

La relation entre les coordonnées d'un point de l'image dans le repère rétinien et celles du même point dans le repère lié à l'image est :

$$\begin{cases} u = -k_u x_r + u_0 \\ v = k_v y_r + v_0 \end{cases} \quad \text{(II.8)}$$

où encore en coordonnées homogènes :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} -k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} sx_r \\ sy_r \\ s \end{pmatrix} \quad \text{(II.9)}$$

II.4.1.2.2 Transformation repère absolu/ repère image

La matrice de projection perspective permet de passer du repère absolu au repère image et vice-versa. En d'autres termes, elle permet le passage des coordonnées 3D (x, y, z) aux coordonnées 2D (u, v).

Ce passage repère absolu/image se décompose en trois étapes :

- Une transformation repère absolu/repère caméra.
- Une transformation repère caméra/repère rétinien.
- Et enfin, une transformation repère rétinien/repère image.

Ainsi à partir des équations II.5, II.7 et II.9, la matrice de projection M s'écrit comme suit :

$$M = \begin{pmatrix} -k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -fk_u r_{11} + u_0 r_{31} & -fk_u r_{12} + u_0 r_{32} & -fk_u r_{13} + u_0 r_{33} & -fk_u t_x + u_0 t_z \\ fk_v r_{21} + v_0 r_{31} & fk_v r_{22} + v_0 r_{32} & fk_v r_{23} + v_0 r_{33} & fk_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_x + u_0 t_z \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v r_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \quad \text{(II.10)}$$

En identifiant l'équation (II.10) avec (II.3), et en tenant compte des propriétés d'orthogonalité de la matrice de rotation R, nous obtenons un ensemble d'équations qui permettent de calculer les paramètres intrinsèques et extrinsèques en fonction des coefficients de la matrice M. Nous aurons donc :

$$\begin{cases} r_3 = m_3 \\ u_0 = m_1 . (m_3)^t \\ v_{0=} m_2 . (m_3)^t \\ \alpha_u = -\|m_1 \wedge m_3\| \\ \alpha_v = -\|m_2 \wedge m_3\| \\ r_1 = (m_1 - u_0 m_3)/\alpha_u \\ r_2 = (m_2 - v_0 m_3)/\alpha_v \\ t_x = (m_{14} - u_0 m_{34})/\alpha_u \\ t_y = (m_{24} - v_0 m_{34})/\alpha_v \\ t_z = m_{34} \end{cases} \quad \text{(II.11)}$$

Pour trouver les paramètres intrinsèques et extrinsèques, nous devons donc :

1. Calculer la matrice de projection M.
2. Extraire les paramètres de la caméra à partir de la matrice M grâce aux formules fournies par le système d'équations (II.11).

II.4.1.2.3 Calcul des paramètres du modèle géométrique d'une caméra

Calibrer une caméra va consister à estimer la matrice de projection M (équation II.2 et II.3). Il suffit de disposer d'un ensemble de points de référence $P_i = (x_i, y_i, z_i)$ appartenant au repère absolu et de leur projection (u_i, v_i) dans l'image numérique **[Crouzil, 1997]**.

D'après l'équation II.2, chaque correspondance donne deux équations :

$$u_i = \frac{m_{11}x_i+m_{12}y_i+m_{13}z_i+m_{14}}{m_{31}x_i+m_{32}y_i+m_{33}z_i+m_{34}} \quad \text{(II.12)}$$

$$v_i = \frac{m_{21}x_i+m_{22}y_i+m_{23}z_i+m_{24}}{m_{31}x_i+m_{32}y_i+m_{33}z_i+m_{34}} \quad \text{(II.13)}$$

Ces équations sont linéaires par rapport aux coefficients m_{ij} de la matrice de projection *M*, donc au moins six points non coplanaires suffisent pour les déterminer.

Les équations (II.12) et (II.13) peuvent se réécrire comme une combinaison linéaire des m_{ij} :

$$m_{11}x_i + m_{12}y_i + m_{13}z_i + m_{14} - u_i m_{31}x_i - u_i m_{32}y_i - u_i m_{33}z_i = u_i m_{34} \quad \text{(II.14)}$$

$$m_{21}x_i + m_{22}y_i + m_{23}z_i + m_{24} - v_i m_{31}x_i - v_i m_{32}y_i - v_i m_{33}z_i = v_i m_{34} \quad \text{(II.15)}$$

On obtient donc 2n équations pour n points et on peut écrire ces équations sous forme matricielle :

$$\begin{pmatrix} & & & & & & & & \vdots & & \\ x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i \\ & & & & & & & & \vdots & & \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} = \begin{pmatrix} \vdots \\ u_i m_{34} \\ \vdots \end{pmatrix} \quad \text{(II.16)}$$

Le système précédent étant homogène, il faut utiliser une contrainte supplémentaire. Nous utilisons pour cela la méthode de calibrage décrite par Faugeras et Toscani **[Faugeras, 1986][Toscani, 1987]:**

Si l'on écrit M sous la forme suivante :

$$M = \begin{pmatrix} m_1 & m_{14} \\ m_2 & m_{24} \\ m_3 & m_{24} \end{pmatrix} \quad \text{(II.17)}$$

où $m_i = (m_{i1}\ m_{i2}\ m_{i3})$ (II.18)

et si l'on pose la contrainte suivant : $\|m_3\| = 1$ (II.19)

alors le système d'équations (III.16) peut s'écrire sous la forme suivante :

$$Bx_9 + Cx_3 = 0 \quad \text{(II.20)}$$

où :

$$B = \begin{pmatrix} . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i \\ 0 & 0 & 0 & 0 & 1 & x_i & y_i & z_i & -v_i \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{pmatrix} \quad \text{(II.21)}$$

$$C = \begin{pmatrix} . & . & . \\ . & . & . \\ -u_i\, x_i & -u_i\, y_i & -u_i\, z_i \\ -v_i\, x_i & -v_i\, y_i & -v_i\, z_i \\ . & . & . \\ . & . & . \end{pmatrix} \quad \text{(II.22)}$$

$$x_9 = (m_1\ m_{14}\ m_2\ m_{24}\ m_{34})^t \quad \text{(II.23)}$$

$$\text{et } x_3 = (m_3)^t \quad \text{(II.24)}$$

Le critère à minimiser étant $\|Bx_9 + Cx_3\|^2$ sous la contrainte$\|m_3\|^2 = 1$, après calculs, l'estimation des coefficients de la matrice de projection M est donnée par les étapes suivantes **[Horaud, 1993]**:

1. Construction d'une matrice D telle que : $D = C^tC - C^tB(B^tB)^{-1}B^tC$ (II.25)
2. Calcul des valeurs et vecteurs propres de D.
3. x_3 est le vecteur propre correspondant à la plus petite valeur propre.
4. Normalisation de x_3 : $x_3 \leftarrow \frac{x_3}{\|x_3\|}$ (II.26)
5. Calcul de x_9 : $x_9 = (B^tB)^{-1}B^tCx_3$ (II.27)
6. Si $m_{34} \leq 0$ alors $M \leftarrow (-M)$ (II.28)

D'autres techniques de calibrage ont été proposées dans la littérature. Dans ce qui suit, nous présentons une classification des techniques de calibrage selon certains critères.

II.4.1.3 Classification des techniques de calibrage

La nature de l'application et la précision désirée dicte la méthode la plus appropriée pour un cas donné. Une première classification basée sur le modèle de la caméra peut être proposée :

- **Méthodes utilisant un modèle de caméra basé sur la projection perspective**: Ces méthodes nécessitent généralement une distorsion optique maitrisée (pouvant être estimée). Généralement ce modèle exige plus de 5 points **[Wiley, 1995]** d'apprentissage (points connue d'une mire par exemple) en utilisant plusieurs images. Due à sa nature non linéaire, ce modèle nécessite une méthode d'approximation de type moindre carré afin de trouver les paramètres géométriques des caméras.

- **Méthodes utilisant un modèle de caméra projective**: Un tel modèle est caractérisé par les matrices essentielle (matrice modélisant le positionnement de deux caméras dans l'espace) et fondamentale (matrice qui fait correspondre tous les pixels de deux images stéréo). Il peut gérer des distances focales variables ou inconnues, mais a besoin d'au minimum 6 à 8 points d'apprentissage pour faciliter l'obtention des paramètres géométriques des caméras (problème linéaire). Les paramètres de distorsion qui sont des paramètres non linéaires sont difficilement traités avec ce modèle.

L'autre critère peut être utilisé afin de classifier les techniques de calibrage de caméras qui sont classées en deux grandes classes selon qu'elles utilisent une mire de calibrage ou pas.

Une mire de calibrage est un objet du monde réel dont on connaît parfaitement la géométrie et à partir duquel seront extraits les points qui seront utilisés dans la phase de calibrage. Elle est généralement composée de motifs répétitifs (cercles ou rectangles) afin de choisir des points d'intérêt dont les coordonnées image (2D) peuvent être mesurées avec une plus grande précision.

- Les techniques de calibrage qui utilisent une mire sont basées sur l'observation de cette dernière. Les coordonnées 3D de quelques points d'intérêt sont alors déterminées et leurs projections sont mesurées dans le repère image et on obtient

ainsi les coordonnées 2D de ces points (en pixels).

Une fois les coordonnées 2D et 3D de ces points connues, il est possible d'estimer la matrice de transformation 3D-2D. La précision de la calibration dépend bien sûr de la précision de la grille (mire) et de celle des points d'intérêts choisis.

- Les techniques de calibrage qui n'utilisent pas de mire (ou connues aussi sous le nom de calibrage automatique ou semi-automatique) supposent, elles, que seuls les paramètres intrinsèques de la caméra sont connus. Ce calibrage ne nécessite donc aucun appareillage de calibration et aucune connaissance a priori sur le système d'imagerie ou la scène, si ce n'est une liste de primitives 2D (points d'intérêt) mises en correspondance entre différentes vues.

La littérature fait souvent aussi état de calibrage faible ou fort ; la distinction se situe au niveau d'une estimation globale de la matrice de projection: cas faible, ou d'une estimation de chaque paramètre qui compose cette matrice: cas fort.

Dans ce qui suit, nous présentons un panorama non exhaustif des principaux algorithmes de calibrage en vision par ordinateur.

II.4.1.4 Principaux algorithmes de calibrage de caméra existants

Les algorithmes sont généralement basés sur le modèle caméra perspective ou projective (modèle Pinhole). Les méthodes de calibrage pour la vision par ordinateur ont traditionnellement utilisés des grilles de référence. La matrice de calibrage « K » est déterminée en utilisant plusieurs images d'une grille d'où des points de coordonnées connues peuvent être extraits (une mire d'échiquier par exemple). Les méthodes les plus connues et utilisées sont celles de Tsai, Heikkilä & Silven et Zhang. Elles se basent toutes sur le modèle caméra « Pinhole » à projection perspective et incluent des fonctions de modélisation de la distorsion optique.

Le modèle de calibrage de Tsai **[Tsai, 1987]** suppose que certains paramètres de la caméra soient fournis par le constructeur, cela réduit l'estimation initiale des paramètres intrinsèques. Elle requiert « n » points par image ($n > 8$), et résout le problème de calibrage en utilisant « n » équations linéaires. Un deuxième modèle de distorsion radiale est utilisé quand aucun décentrement du point focal sur le CCD lié à la distorsion n'est considéré. Les deux étapes supportent aussi bien une entrée d'une ou plusieurs images d'une mire aussi

bien planaire que tridimensionnelle, à condition que les coordonnées des points soit connues.

La technique développée par Heikkilä & Silven **[Heikkilä, 1997]** extrait en premier lieu des estimations initiales des paramètres de caméras utilisant l'algorithme DLT (Direct Linear Transformation) (Abdel-Aziz et Karara 1971) ensuite une approximation non linéaire à base de moindres carrés et utilisant l'algorithme de Levenberg-Marquardt est appliquée afin de raffiner les paramètres intrinsèques incluant la distorsion. Ce modèle utilise deux coefficients pour la distorsion radiale et de décentrement, cette méthode supporte assez bien une entrée d'une ou plusieurs images d'une mire aussi bien planaire que tridimensionnelle.

La méthode Zhang **[Zhang, 2000]** requière des images d'une mire de calibrage planaire placée à différentes orientations (plus que deux) face à la caméra. L'algorithme utilise la détection de coins afin d'extraire les points de la mire afin de calculer une transformation projective les points des « n » différentes images. Ensuite, les paramètres intrinsèques et extrinsèques sont calculés en utilisant des fonctions linéaires, tandis que les paramètres de distorsion sont optimisés avec une méthode non linéaire de type moindre carré. Une minimisation finale non linéaire de l'erreur de projection est appliquée utilisant l'algorithme de Levenberg-Marquardt afin de raffiner tous les paramètres de calibrage. L'approche de Zhang est assez similaire à celle de Triggs **[Triggs, 1998]** qui a besoin d'au minimum 5 vues d'une scène plane.

Le terme d'auto calibration en vision par ordinateur est utilisé quand aucun objet de calibrage de type mire est employé, et les propriétés métriques de la caméra sont calculés à partir d'images non calibrées, utilisant les contraintes de la caméra ou de la scène à modéliser. L'auto calibrage est généralement adoptée en modélisation 3D en améliorant la reconstruction projective. En général, trois types de contraintes sont appliquées (séparément ou en ensemble) afin de commencer l'auto calibrage : Les contraintes de la scène, les contraintes de déplacement de la caméra ou les contraintes liées aux paramètres intrinsèques de la caméra. Mais dans le cas où nous avons une caméra inconnue, un déplacement de caméra inconnu et une scène inconnue aussi, seules les contraintes liées à l'orientation interne (CCD) de la caméra peuvent être utilisés.

II.4.2 Rectification d'une paire d'image

L'étape de rectification des images permet de simplifier le processus de la vision stéréoscopique dans certaines applications. En effet, elle permet de se ramener à une géométrie épipolaire simple, vu que nous obtenons des lignes épipolaires parallèles aux lignes des images (voir Figures II.8 et II.9). Donc, les points homologues des deux images ont la même ordonnée (ils se trouvent sur la même ligne épipolaire), ce qui réduit la recherche des correspondants à une recherche monodimensionnelle. Ceci rend cette configuration beaucoup plus intéressante du point de vue algorithmique.

La rectification est donc une transformation des images qui dépend uniquement de la géométrie du système d'acquisition (paramètres des caméras et leur position relative).

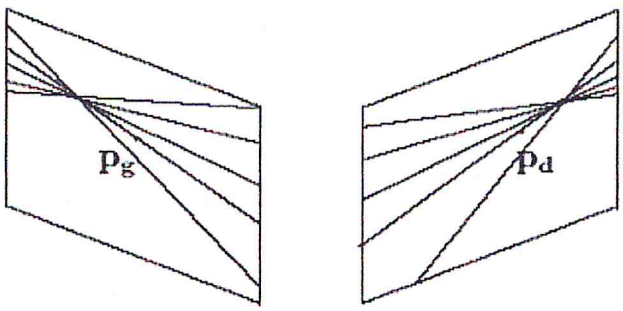


Figure II.8 Faisceaux de lignes épipolaires: cas général non rectifié

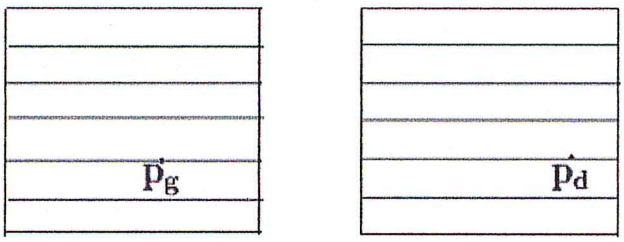


Figure II.9 Faisceaux de lignes épipolaires: cas rectifié

II.4.3 La mise en correspondance

La mise en correspondance est communément considérée comme une étape clé dans le processus de stéréovision. C'est aussi l'une des étapes les plus difficiles du fait de son aspect combinatoire.

De nombreux travaux ont été effectués dans le domaine de la stéréovision pendant plus de quatre décennies. Cependant, sa théorie a été définie par le travail de David Marr en 1977(David Marr est un neuroscientifique anglais intéressé d'abord par une théorie générale de la cognition puis par résoudre le problème de la vision en particulier). Marr a identifié l'ambiguïté du problème d'appariement et il a défini deux contraintes principales permettant de formuler le problème avec précision **[Kostkovà, 2002]** :

- **L'unicité**: un point d'une image ne peut pas avoir plus d'un correspondant parce que le résultat de la projection d'un point est au maximum un point.
- **La continuité** : si la matière est séparée en objet dont les surfaces suffisamment lisses, alors la disparité est une fonction continue par morceaux **[Rabaud, 2005]**.

Dans l'optique de récupérer l'information la plus importante qui est la profondeur perdue lors de l'acquisition des images, nous utilisons les techniques de mise en correspondance entre les images prises selon différents points de vue. C'est grâce au changement du point de vue que nous pourrons retrouver l'information 3D.

Mais malheureusement, cette étape possède des difficultés causées par plusieurs facteurs dont les plus importants sont:

1. Occultation :

Ce sont des zones visibles de la scène dans une image et invisibles dans les autres, ceci est dû principalement au changement du point de vue. Ces zones n'auront donc pas de correspondant, ce qui représente un problème crucial. C'est pour ces raisons que leur détection est nécessaire afin qu'elle n'influe pas sur l'appariement.

2. Bruit :

L'acquisition des images se produit avec un certain bruit qui est dû en particulier aux imperfections des capteurs (caméra ou détecteur électronique). Un bon capteur est celui qui produit un faible bruit de lecture. Les bruits ne pouvant pas être éliminés une fois introduits dans l'image brute, il convient de les minimiser car ce sont eux qui limitent la précision de mesure dans les traitements des images en particulier la mise en correspondance **[Gonzales, 1977]**.

3. Les distorsions photométriques :

Les distorsions photométriques déforment les valeurs d'intensités des pixels.

4. Les distorsions projectives :

Elles déforment la forme des objets, conduisant ainsi à une perte d'informations.

Afin de déterminer les correspondants, il est important de définir les éléments à mettre en correspondance. Ces derniers peuvent être des primitives ou des intensités de pixels. On peut classer grossièrement les approches à la mise en correspondance en deux catégories selon qu'elles font usage de primitives ou de pixels.

- **Méthodes à base de primitives**

Dans le contexte de la stéréovision, le but des approches à base de primitives est d'obtenir des correspondances fiables, même en présence d'une certaine quantité de bruit dans les images. Puisqu'on s'intéresse à la reconstruction de la scène 3D, les primitives sont celles comportant une information sur la structure 3D de la scène, en général la structure tridimensionnelle peut être décrite par : les contours et les coins (les intersections des contours), c'est pourquoi la plupart des approches à base de primitives se sont intéressées aux contours et aux coins, ce qui nécessite une phase pendant laquelle sont extraits de l'image en utilisant un détecteur automatique **[Rabaud, 2005]**.

L'avantage de ces méthodes est qu'elles sont plus rapides grâce à la réduction de l'information contenue dans les images. Cependant, elles sont fortement limitées par le nombre des primitives, et leurs résultats sont des données éparses.

- **Méthodes à base de pixels**

Les méthodes à base de pixels exploitent directement l'intensité des pixels. Pour identifier les correspondances (appariements) de pixels entre les images (pour chaque pixel d'une image on cherche son correspondant dans l'autre image) une mesure de ressemblance est utilisée. Ces méthodes sont très sensibles aux distorsions dues au changement de vue, et très sensibles aux différences d'illumination et de contraste **[Rabaud, 2005]**.

II.4.4 Reconstruction 3D

La reconstruction tridimensionnelle consiste à calculer la géométrie de la scène observée par le système d'acquisition. Elle est obtenue à partir des résultats de la phase de mise en correspondance en utilisant les paramètres de calibrage du système de caméras utilisées. Selon le type de calibrage utilisé, nous pouvons procéder à trois types de reconstruction **[Faugeras, 1992]**.

II.4.4.1 Reconstruction projective [Hartley, 1994]

La reconstruction projective de la surface est utilisée dans le cas où le système est faiblement calibré. Ce type de reconstruction est le moins riche vu qu'il ne requiert aucune connaissance a priori sur la scène ou sur les paramètres des caméras (voir tableau II.1).

II.4.4.2 Reconstruction affine [Zeller, 1996]

La reconstruction affine requiert une connaissance du plan à l'infini qui permet de déterminer les caractéristiques affines de la scène. Pour cela, il suffit d'identifier trois paires de droites parallèles ou bien connaître les paramètres de calibrage (voir tableau II.1 pour plus d'informations sur les invariantes de ce type de reconstruction).

II.4.4.3 Reconstruction euclidienne [Faugeras, 1995]

C'est le type de reconstruction le plus utilisé et le plus riche en terme d'information. A partir des paramètres obtenus par auto-calibrage ou par calibrage fort, nous pouvons procéder à une reconstruction euclidienne (triangulation) caractérisée par des paramètres invariants cités dans le tableau II.1. La reconstruction affine peut être aussi utilisée en imposant d'autres contraintes euclidiennes.

La représentation	**Les invariantes**
Projective	Le birapport de quatre points alignés
Affine	Le parallélisme, le barycentre
Euclidienne	Les angles, les distances

Tableau II.1 Les invariants des trois type de reconstruction

II.5 Conclusion

Tout au long de ce chapitre, nous avons présenté les techniques de la stéréovision qui permettent de réaliser la reconstruction tridimensionnelle dans le domaine de la vision artificielle, en passant par une chaine de traitement. En premier lieu, on commence par une acquisition des images ainsi que leur traitement afin de les améliorer. Ensuite, une étape de calibrage des caméras est entreprise pour calculer les paramètres nécessaires qui permettent d'entamer l'étape finale, à savoir la mise en correspondance des points acquis des images afin de reconstruire la scène 3D.

Le chapitre suivant sera consacré à l'implémentation d'un algorithme évolutionnaire en utilisant les techniques de la stéréovision pour la reconstruction tridimensionnelle d'une scène observée.

Chapitre III
Conception

CHAPITRE III
CONCEPTION

III.1 Introduction

Après avoir détaillé dans les chapitres précédents les notions des algorithmes évolutionnaires, la vision artificielle et la reconstruction 3D, nous consacrons ce chapitre au développement de notre approche.

Deux phases successives couronnent notre approche ; la première c'est la phase de prétraitement ; elle repose sur trois étapes à savoir, le calibrage de la caméra, le calcul du champ de vision et enfin le calcul du gradient. La deuxième quant à elle est la phase de traitement qui consiste à implémenter un algorithme évolutionnaire pour traiter le problème de reconstruction 3D.

Parmi les algorithmes se basant sur l'approche évolutionnaire, nous avons appliqué l'algorithme des mouches, qui est considéré comme un volet récent et vaste dans le domaine de la stéréovision.

III.2 Description de l'approche proposée

Dans ce qui suit, nous allons décrire notre approche qui est basée sur les deux phases suivantes : le prétraitement et le traitement.

III.2.1 Prétraitement

Il se déroule en trois phases :

- Calibrage de la caméra ;
- Le calcul du champ de vision ;
- Le calcul du gradient.

III.2.1.1 Calibrage de la Caméra

Nous avons vu au chapitre II (§ II.4.1) que le calibrage d'une caméra consiste à déterminer la fonction qui associe à un point de l'espace tridimensionnel, sa projection dans l'image, ce qui nous permet de retrouver les positions des points dans l'espace, connaissant uniquement leurs projections sur les images. Pour ce faire, nous commençons par décrire le matériel utilisé.

III.2.1.1.1 Le système mobile ATRV2

Le système mobile ATRV2 (Figure III.1) est équipé d'une ceinture de capteurs à ultrasons à travers laquelle le robot peut acquérir rapidement une perception panoramique complète à 360°.

Figure III.1 Le système mobile ATRV2

III.2.1.1.2 Système de vision

Le système de vision du système mobile ATRV2 est composé de deux caméras CCD couleurs montées sur une tourelle Pan Tilt (Figure III.2). Leur utilisation a pour avantage de fournir des informations très riches sur l'environnement. A partir de ces informations tels que des points et segments de droite extraits de l'image de la scène observée, on peut contrôler le déplacement relatif et la position du système mobile vis à vis d'éléments caractéristiques de l'environnement.

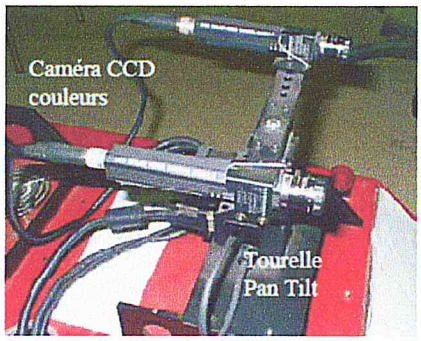


Figure III.2 Système de vision de l'ATRV2

III.2.1.1.3 Méthode de calibrage

Afin de calibrer les deux caméras (caméra gauche et caméra droite) nous avons appliqué la méthode de calibrage fort qui consiste à utiliser un objet de calibrage appelé : mire de calibrage afin de déterminer les paramètres du modèle de la caméra à l'aide des points d'interêt (points de la mire) dont les coordonnées images sont mesurées avec une plus grande précision.

La mire est placée devant la caméra dont les points sont parfaitement connus dans un repère de la mire (x, y, z) différent du repère caméra. Chaque point de la mire se projette dans l'image et on mesure ses coordonnées dans le repère image (u, v). Le calibrage est décomposé en deux transformations (Figure III.3):

- Transformation qui permet de passer du repère mire (repère absolu) au repère caméra (équation II.5).
- Transformation qui permet de passer du repère caméra au repère image (équation II.7 et II.9).

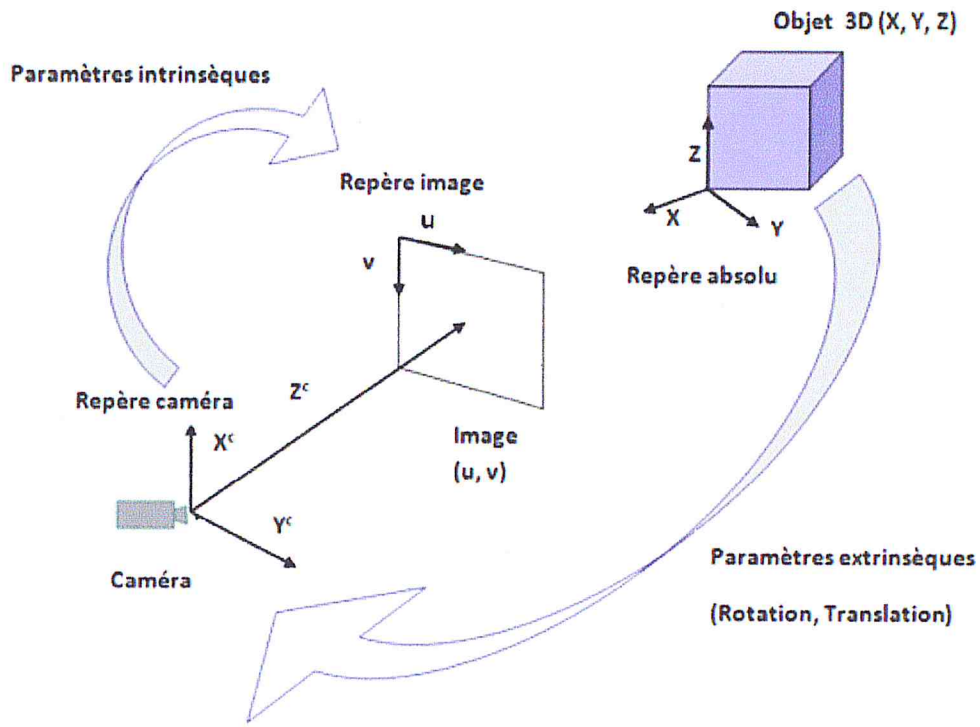


Figure III.3 Calibrage de la caméra

A partir de ses deux transformations nous pouvons définir la matrice de projection perspective M (équation II.10). Afin de calculer la matrice M, il suffit de résoudre le système d'équations (II.11) moyennant l'application de la méthode de Faugeras et Toscani qui ont proposé la solution citée auparavant dans §II.4.1.2.3, la résolution de ce système permettant de déduire les paramètres extrinsèques et intrinsèques propre aux caméras.

III.2.1.1.4 Résultat du calibrage de caméras de l'ATRV2

Le calibrage est réalisé avec une mire constituée d'un ensemble de petits cercles noirs de 10 mm de diamètre et un espacés entre eux de 50 mm. Le motif de cercle est choisi pour que sa projection dans l'image numérisée puisse être mesurée avec une grande précision, donc les centres des cercle représentent les points d'intérêt, telles que leurs coordonnées 2D et 3D sont connues dans le repère de la mire (voir figure III.4).

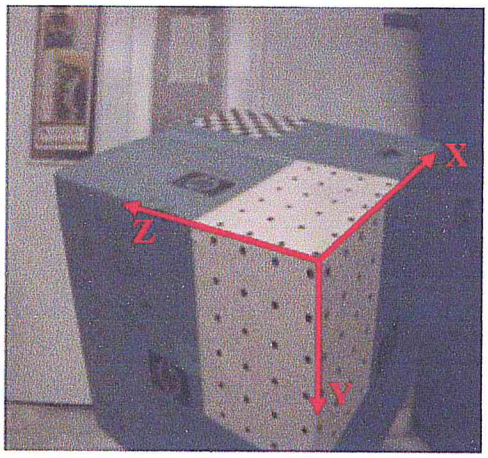


Figure III.4 La mire de calibrage utilisée

Afin de valider les résultats du calibrage du banc stéréoscopique du système de robotique mobile ATRV2, nous avons reconstruit les points 3D de la mire utilisée et comparé les valeurs des points reconstruits avec leur vraie valeur acquise lors du calibrage. Si les points reconstruits et les points d'intérêt se superposent, on peut dire que le calibrage est validé.

Ci-dessus nous présentons les résultats de calibrage obtenus pour plusieurs paires d'images stéréoscopique :

Les cercles rouges (Figures III.5, III.6, III.7, III.8, III.9) désignent les points 2D extraits manuellement et les croix bleues (Figures III.5, III.6, III.7, III.8, III.9) désignent les points 2D calculés à partir des matrices de projection perspective M1 (paire1), M2 (paire2), M3 (paire3).

- **Paire d'image stéréoscopique 1 :**

Figure III.5 Validation 2D des résultats du calibrage 1

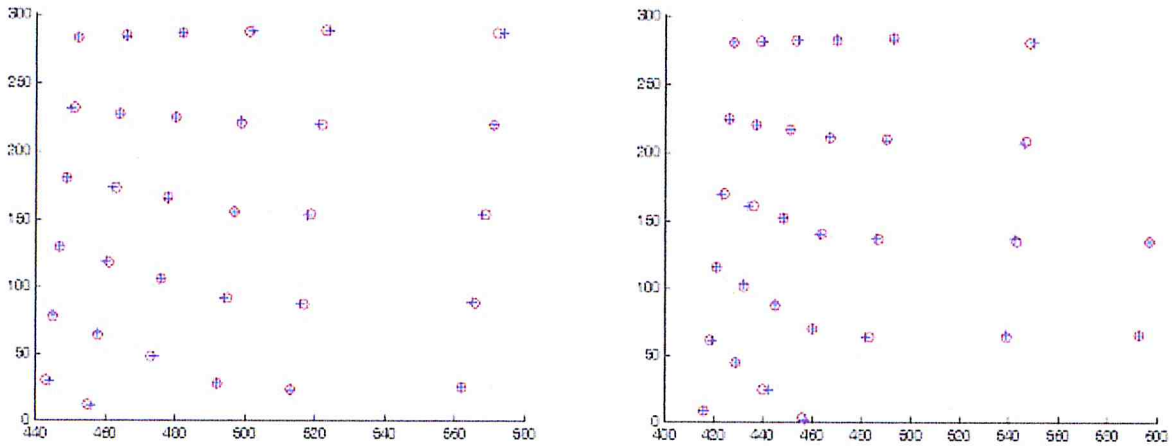


Figure III.6 Validation 2D graphique des résultats du calibrage 1

Le tableau suivant représente 5 points parmi les 31 points extraits de la mire et utilisés lors du calibrage :

Caméra Gauche					Caméra Droite				
X	**Y**	**Z**	**U**	**V**	**X**	**Y**	**Z**	**U**	**V**
730	-120	830	272	523	695	-270	830	276	493
730	-170	830	274	572	695	-320	830	280	548
730	-120	780	340	522	695	-270	780	350	490
730	-170	780	340	571	695	-320	780	325	547
730	-120	730	406	519	695	-270	730	424	487

Tableau III.1 coordonnées des points extraits de la mire de calibrage 1

- **Paire d'image stéréoscopique 2 :**

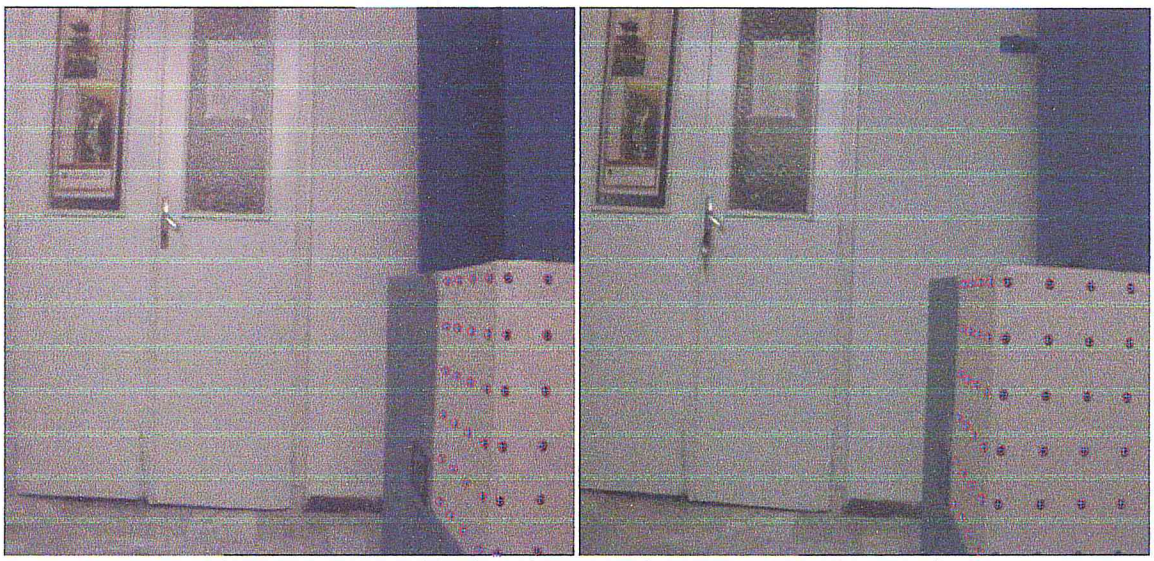


Figure III.7 Validation 2D des résultats du calibrage 2

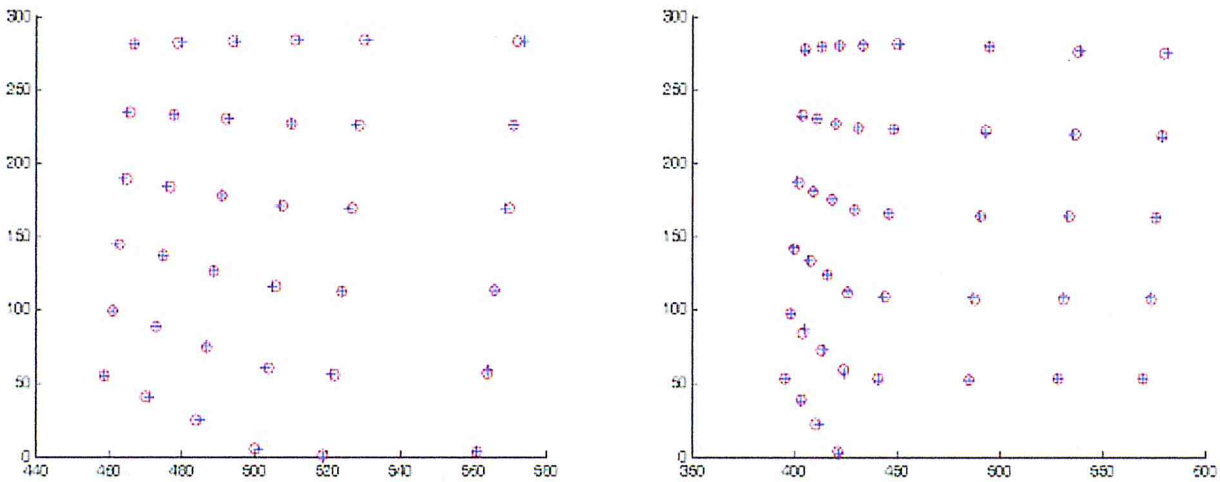


Figure III.8 Validation 2D graphique des résultats du calibrage 2

Les coordonnées 2D et 3D de 5 points parmi 36 points extraits de la mire de calibrage présentée dans la paire d'image stéréoscopique 2, sont données dans le tableau suivant :

Caméra Gauche					**Caméra Droite**				
X	**Y**	**Z**	**U**	**V**	**X**	**Y**	**Z**	**U**	**V**
810	-165	830	276	530	830	-270	830	279	450
258	-150	830	276	511	830	-270	680	452	444
875	-150	730	382	491	895	-255	830	280	422
810	-215	680	447	566	830	-420	780	341	579
925	-150	730	376	447	945	-255	830	281	413

Tableau III.2 coordonnées des points extraits de la mire de calibrage 2

- **Paire d'image stéréoscopique 3 :**

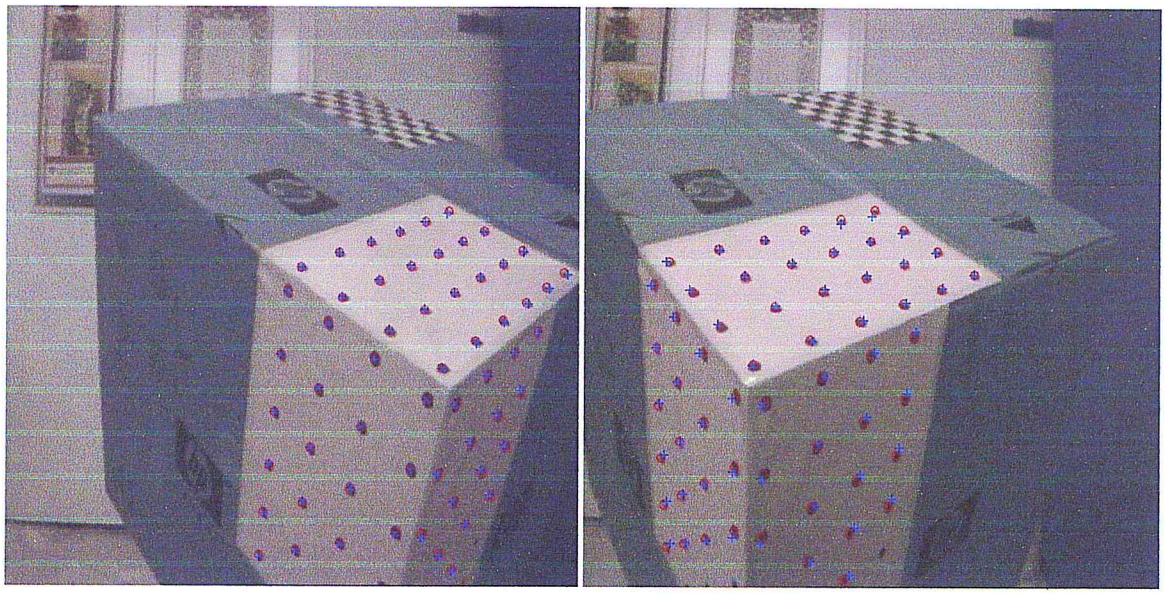


Figure III.9 Validation 2D des résultats du calibrage 3

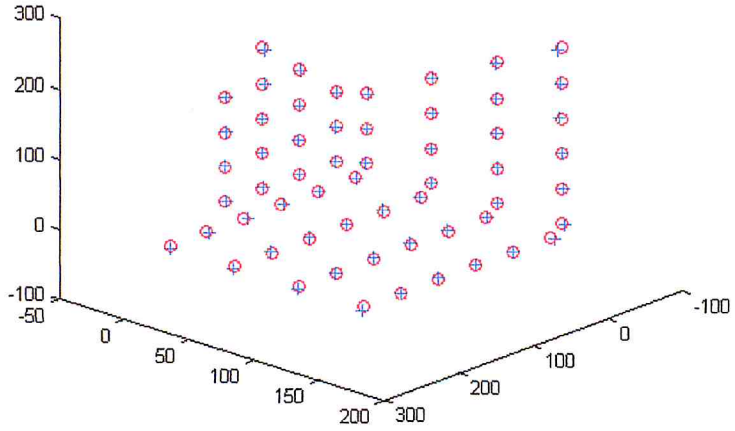


Figure III.10 Validation 3D graphique des résultats du calibrage 3

Les cercles rouges affichés dans la Figure III.10 représentent les points 3D calculés à partir des points 2D extraits manuellement, et les croix bleues dans la même figure représentent les points 3D reconstruits en utilisant la matrice de projection perspective M3.

Le tableau suivant montre les coordonnées 2D et 3D de 5 points parmi 58 points d'intérêts mesurés à partir de la mire de la paire d'image stéréoscopique 3, et utilisés lors du calibrage :

Caméra Gauche					**Caméra Droite**				
X	**Y**	**Z**	**U**	**V**	**X**	**Y**	**Z**	**U**	**V**
15	0	15	380	442	15	0	15	375	159
115	0	65	370	328	115	0	65	363	99
165	0	65	336	289	165	0	65	329	77
65	0	115	467	361	65	0	115	460	127
6	115	15	334	533	6	115	15	335	298

Tableau III.3 coordonnées des points extraits de la mire de calibrage 3

A partir des résultats de validation (Figure III.5 à Figure III.10), on remarque que les points mesurés manuellement qui sont affichés avec « o » rouge correspondent bien avec les points 2D de la mire designés par un « + » bleu, ce qui valide notre calibrage.

Les résultats des trois calibrages qui sont les matrices de projection perspective : M1, M2 et M3, permettent de définir la matrice globale M gauche et droite, en calculant la moyenne de ces trois matrices. La matrice M, ainsi que les paramètres intrinsèque et extrinsèque sont montré comme suit :

➢ **Résultat du calibrage de la caméra gauche**

- **Matrice de projection perspective M_g :**

$$\begin{pmatrix} -289.4185355 & 73.1741906 & 754.9244796 & -5559864.6341 \\ -278.1906589 & 679.1509504 & 4.224512436 & -3723.5804002 \\ -1.0000000 & 0.192004788 & 0.063559176 & 153.14168173 \end{pmatrix}$$

- **Paramètres Intrinsèques :**

$$\begin{pmatrix} -749.9792493 & 0 & 351.4507056 \\ 0 & 627.31322 & 408.8593731 \\ 0 & 0 & 1.0000000 \end{pmatrix}$$

Avec :

$u_0 = 351.4507056$, $v_0 = 408.8593731$

$\alpha_u = -749.9792493$, $\alpha_v = 627.31322$

- **Paramètres extrinsèques :**

$$R = \begin{pmatrix} -0.0827118 & -0.0075921 & -0.9768091 \\ 0.2082989 & 0.9574929 & -0.0346912 \\ -1.0000000 & 0.19200474 & 0.0635591 \end{pmatrix} \text{ et } T = \begin{pmatrix} 813.10011 \\ -105.74779 \\ 153.14168 \end{pmatrix}$$

Où : R et T sont respectivement la matrice de rotation et le vecteur de translation.

➢ **Résultat du calibrage de la caméra droite**

- **Matrice de projection perspective M_d :**

$$\begin{pmatrix} -289.3345357 & 67.38023438 & 740.27454 & -529918.420 \\ -298.6667067 & 643.4704952 & -3.6203302 & 145921.787 \\ -1.000000000 & 0.151022566 & 0.0625703 & 198.023869 \end{pmatrix}$$

- **Paramètres Intrinsèques :**

$$\begin{pmatrix} -730.5241163 & 0 & 345.8297221 \\ 0 & 600.1698002 & 395.6187469 \\ 0 & 0 & 1.000000000 \end{pmatrix}$$

Avec :

$u_0 = 345.8297221$, $v_0 = 395.6187469$

$\alpha_u = -730.5241163$, $\alpha_v = 600.1698002$

- **Paramètres extrinsèques :**

$$R = \begin{pmatrix} -0.0773351 & -0.0207414 & -0.9837263 \\ 0.1615410 & 0.9725966 & -0.0472771 \\ -1.0000000 & 0.1510225 & 0.0625703 \end{pmatrix} \text{ et } T = \begin{pmatrix} 819.13922 \\ 1.1260118 \\ 198.02386 \end{pmatrix}$$

Où : R la matrice de rotation et T le vecteur de translation.

III.2.1.1.5 Algorithme de calibrage

Variable utilisé :

nbr: type entier, c'est le nombre de points dans la mire.

a : type réel, matrice de taille [nbr][5].

Début

- Lire le fichier contenant les points 3D (x, y, z) et les points 2D (u, v) de la mire ;
- Copier les valeurs de fichier dans la matrice a qui contient 5 colonne, tel que les 3 premières colonnes représentent (x, y, z) et les deux derniers (u, v) ;
- Initialisation de la matrice A [2*nbr][9] ;
- Initialisation de la matrice C [2*nbr][3] ;

Pour i=1 à nbr

Faire

- X = a[i] [1]; -Y = a[i] [2]; - Z = a[i] [3];
- V = a[i] [4]; -U= a[i] [5];

Fait;

- X_9= transposé (m_1, m_{14}, m_2, m_{24}, m_{34});
- X_9= transposé (m_{11}, m_{12}, m_{13}, m_{14}, m_{21}, m_{22}, m_{23}, m_{24}, m_{34});
- X_3 = transposé (m_{31}, m_{32}, m_{33}) ;

// Le remplissage de la matrice A

K=1 ;

Pour i=1 à nbr

Faire

A [K][i]=[X(i) Y (i) Z(i) 1 0 0 0 0 –C(i)];

A [K+1][i]=[0 0 0 0 X(i) Y (i) Z(i) 1-V[i]; K=K+2;

Fait;

K=1;

Pour i=1 à nbr // remplissage de la matrice C

Faire

C[K][i]=[-C(i)*X(i) -C(i)*- C (i) * Y (i) –C(i)*Z(i)];

C[K+1][i]=[-V(i)* X(i) –V(i)* Y (i) –V(i)* Z(i)];

K=K+2;

Fait;

- D = transposé(C)* C - transposé(C)* A* inverse (transposé(A) *A)- transposé(A)*C) ;
- lam c'est la valeur propre de D ;
- Calculé v le vecteur propre qui correspond à la valeur propre ;
- X_3= transposé (v) ;
- X_9= inverse (transposé(A)*(A)* transposé(A)*C *X_3 ;
- X_9= inverse (transposé(A)*(A)* transposé(A)*C *v ;

// Construction de la matrice M

- M=[X_9 (transposé (m_{11}, m_{12}, m_{13})) X_9(m_{14})

  X_9(transposé (m_{21}, m_{22}, m_{23})) X_9(24)

  X_3 X_9(m_{34})];

Si (X_9(9) <=0) **alors**

M= -M;

Fin Si;

Fin;

III.2.1.2 Calcul du champ de vision

Théoriquement, le champ de vision est déterminé en fonction de l'angle de prise de vue de la caméra. Pour un banc stéréoscopique, ce champ est déterminé par l'intersection de champ de vision des deux cameras.

Nous avons utilisé pour le calcul du champ de vision commun du banc sterescopique, la même démarche que celle utilisée dans la thése de Dr. O.Djekoune **[Djekoune, 2010]** au niveau de l'équipe de « Vision Artificielle » de la Division Productique et Robotique du CDTA.

Dans notre cas le champ de vision commun entre la caméra gauche et la caméra droite est représenté par une forme géométrique composé de six sommets de coordonnées 3D (voir figure III.11) nommé : P1, P2, P3, P4, P5, P6. Le problème de calcul de champ de vision est alors de trouver les coordonnées de ces points 3D.

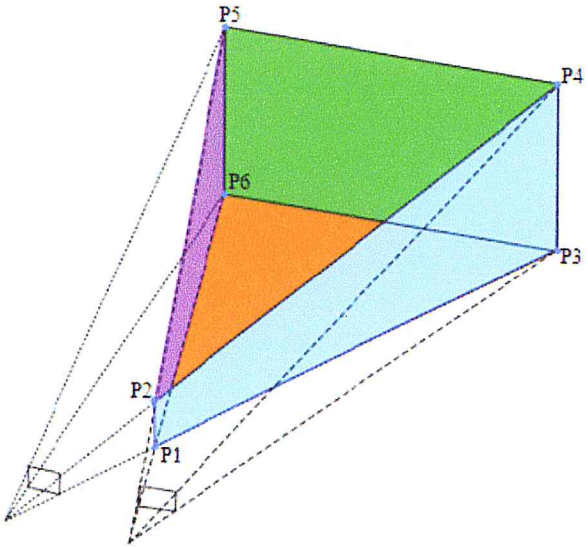


Figure III.11 Le champ de vision commun théorique

Afin de calculer les points formant le champ de vision, nous avons appliqué la méthode suivante qui est composée de deux étapes :

Etape 1 : consiste à calculer les 6 points du champ du vision en 2D.

Notons I_1 l'image droite avec une taille $Max_{ud} \times Max_{vd}$ et I_2 l'image gauche avec une taille $Max_{ug} \times Max_{vg}$, chaque image est caractérisée par 4 points qui sont : les points d'extrémité (voir Figure III.12). Le tableau III.4 montre les coordonnées des huit points extrait des deux images, (on a utilisé des images de taille 600x560 pixels donc $Max_{ud} = Max_{ug} = 600$ et $Max_{vd} = Max_{vg} = 560$) .

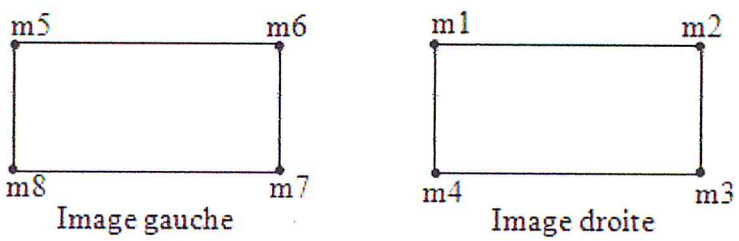


Figure III.12 Les points extrémité de l'image gauche et droite

Point	U	V
m1	0	0
m2	600	0
m3	600	560
m4	0	560
m5	0	0
m6	600	0
m7	600	560
m8	0	560

Tableau III.4 Coordonnées des points des extrémités des deux images

Chaque point 2D du champ de vision est défini par l'intersection d'un point des extrémités de l'image droite et l'autre point de l'image gauche (voir Figure III.11), alors chaque point est formé par une paire de coordonnées (U_d,V_d) et (U_g,V_g). Dans le tableau suivant sont illustrées les coordonnées des points 2D du champ de vision :

Point 2D	Coordonnées
P1	m4, m7
P2	m1, m6
P3	m3, m7
P4	m2, m6
P5	m1,m5
P6	m4, m8

Tableau III.5 Coordonnées 2D des points du champ de vision

Etape2 : Transformation 2D_3D des points du champ de vision.

Cette transformation se fait à l'aide des matrices Mg et Md calculés dans l'étape de calibrage des deux caméras gauche et droite, définie comme suite:

$$Mg = \begin{pmatrix} Mg_{11} & Mg_{12} & Mg_{13} & Mg_{14} \\ Mg_{21} & Mg_{22} & Mg_{23} & Mg_{24} \\ Mg_{31} & Mg_{32} & Mg_{33} & Mg_{34} \end{pmatrix} \quad \text{(III.1)}$$

$$Md = \begin{pmatrix} Md_{11} & Md_{12} & Md_{13} & Md_{14} \\ Md_{21} & Md_{22} & Md_{23} & Md_{24} \\ Md_{31} & Md_{32} & Md_{33} & Md_{34} \end{pmatrix} \quad \text{(III.2)}$$

Pour arrive à calculer les points 3D il faut utiliser la formule mathématique suivante :

$$P = I \times E^t \times C \quad \text{(III.3)}$$

d'où :

$$I = (E^t \times E)^{-1} \quad \text{(III.4)}$$

et E=A-B, suivie d'un calcul de la matrice inverse du produit de E et E^t

avec :

$$A = \begin{pmatrix} Mg_{11} & Mg_{12} & Mg_{13} \\ Mg_{21} & Mg_{22} & Mg_{23} \\ Md_{11} & Md_{12} & Md_{13} \\ Md_{21} & Md_{22} & Md_{23} \end{pmatrix} \quad \text{(III.5)}$$

$$B = \begin{pmatrix} ug_i Mg_{31} & ug_i Mg_{32} & ug_i Mg_{33} \\ vg_i Mg_{31} & vg_i Mg_{32} & vg_i Mg_{33} \\ ud_i Mg_{31} & ud_i Mg_{32} & ud_i Mg_{33} \\ vd_i Mg_{31} & vd_i Mg_{32} & vd_i Mg_{33} \end{pmatrix} \quad \text{(III.6)}$$

$$C = (ug_i Mg_{34} - Mg_{14}\,,\ vg_i Mg_{34} - Mg_{24}\,,\ ud_i Md_{34} - Md_{14}\,,\ vd_i Md_{34} - Md_{24}) \quad \text{(III.7)}$$

avec : ug_i, vg_i, ud_i, vd_i sont les coordonnées gauche respectivement droite des points 2D du champ de vision (i ϵ [1,6]).

Donc le champ de vision commun des deux caméras est défini par les coordonnées des points 3D représentés dans le tableau suivant :

Point	X	Y	Z
P1	422	-27	739
P2	430	-22	895
P3	8711	-4972	-2589
P4	6479	-3315	3546
P5	3751	1528	2034
P6	3986	1631	-555

Tableau III.6 Des points 3D formant le champ de vision

➢ **Algorithme de calcul du champ de vision**

Début

- Calculer la matrice A ;

Pour i=1 à 6
Faire
- Calculer la matrice B ;
- Calculer la matrice C ;
- E=A-B ;
- I= inverse (transposé (E) × E) ;
- P=I × transposé(E) × C ; /*P le vecteur finale représente les coordonnées 3D */

Fait;

Fin;

III.2.1.3 Le calcul du gradient

Parmi Les opérateurs utilisés dans le calcul du gradient, nous pouvons citer ceux de Sobel, Prewitt et celui de Laplace, etc. Dans notre cas, nous nous sommes intéressés aux gradients de Sobel. Le but de cet opérateur est de faire apparaitre les pixels qui changent brusquement de valeur par rapport à leur voisinage, donc ceux qui correspondent aux frontières qui séparent les différents objets de l'image.

III.2.1.3.1 L'opérateur de Sobel

Sobel est un opérateur utilisé en traitement d'image pour la détection de contours. Il s'agit d'un des opérateurs les plus simples qui donne toutefois des résultats corrects.

L'operateur de Sobel permet d'estimer localement la norme du gradient spatial bidimensionnel d'une image en niveau de gris. Il détecte les régions de fortes variations locales d'intensité correspondantes aux contours.

III.2.1.3.2 Principe de fonctionnement

Cet opérateur consiste en une paire de masques de convolution 3x3 (voir Figure III.13).
Une rotation de 90° permet de passer d'un masque à l'autre.

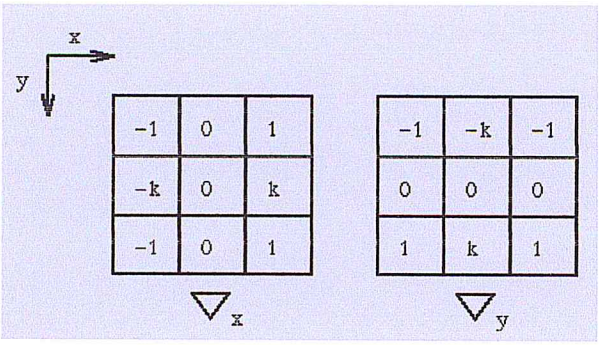


Figure III.13 Les masques de convolution de l'operateur de Sobel

L'operateur de Sobel est défini pour une valeur de k=2. Ce masque est conçu pour répondre aux contours horizontaux et verticaux.

L'application séparée de chacun des masques donne une estimation des composantes horizontales et verticales du gradient, notées respectivement ∇_xet ∇_y par un simple filtrage linéaire avec un masque 3x3.

Il est ensuite possible de calculer la norme du gradient en chaque point à partir des composantes ∇_xet ∇_y . La norme du gradient de chaque pixel est donnée par la relation:

$$||\nabla|| = \sqrt{\nabla_x{}^2 + \nabla_y{}^2} \qquad \text{(III.8)}$$

Bien qu'elle soit souvent approchée par la formule ci-dessous, plus rapide à calculer.

$$||\nabla|| \approx |\nabla_x| + |\nabla_y| \qquad \text{(III.9)}$$

La norme du gradient ainsi estimée correspond à l'intensité attribuée au pixel courant, c'est donc l'image de la norme du gradient que l'on visualise généralement dans l'image de contours.

Le schéma suivant résume la détection du gradient avec la procédure pour le calcul du gradient par l'opérateur de Sobel :

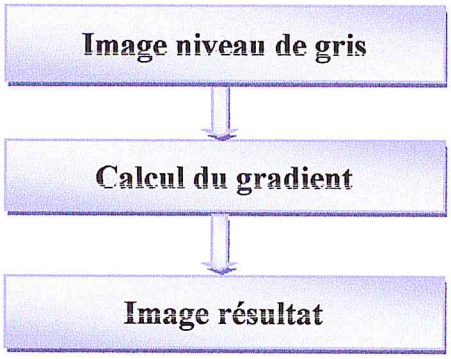


Figure III.14 Schéma de détection du gradient

III.2.1.3.3 Procédure de Sobel

Variable utilisé

I : Image en niveau de gris.

haut : la hauteur de l'image I.

larg : la largeur de l'image I.

pour chaque pixel, le niveau de gris est définie comme suite :

$$\forall x \in [0, haut], \forall y \in [0, larg] \quad I(x, y) = N \text{ tq } N \in [0, 255] \qquad (III.10)$$

Cx, Cy: les masques de convolution respectivement horizontale et verticale.

Pour chaque pixel de l'image, les gradients horizontal G_x et vertical G_y sont définis par la somme du produit de chaque valeur de niveau de gris de voisinage et du coefficient de la matrice de convolution

L'algorithme suivant exprime la procédure de calcul d'une image gradient par l'opérateur de Sobel :

Début

Pour i=1 à haut-1

Faire

Pour j=1 à larg-1

Faire

G_x= I(i-1, j-1)*Cx(0,0) + I(i-1, j)*Cx(0,1) + I(i-1, j+1)*Cx(0,2) +
I(i, j-1)*Cx(1,0) + I(i, j)*Cx(1,1) + I(i, j+1)*Cx(1,2)+
I(i+1, j-1)*Cx(2,0) + I(i+1, j)*Cx(2,1) + I(i+1, j+1)*Cx(2,2) ;

G_y= I(i-1, j-1)*Cy(0,0) + I(i-1, j)*Cy(0,1) + I(i-1, j+1)*Cy(0,2) +
I(i, j-1)*Cy(1,0) + I(i, j)*Cy(1,1) + I(i, j+1)*Cy(1,2)+
I(i+1, j-1)*Cy(2,0) + I(i+1, j)*Cy(2,1) + I(i+1, j+1)*Cy(2,2) ;

Gradient (i, j) = $\sqrt{G_x^2 + G_y^2}$;

Fait ;

Fait ;

Fin ;

III.2.1. 3.4 Résultat du gradient de Sobel

La figure III. 15 représente l'image gradient résultat en appliquant l'opérateur de Sobel sur l'image initiale en niveau de gris représentée par la figure III.16, en passant par les gradients intermédiaires horizontal G_x et vertical G_y.

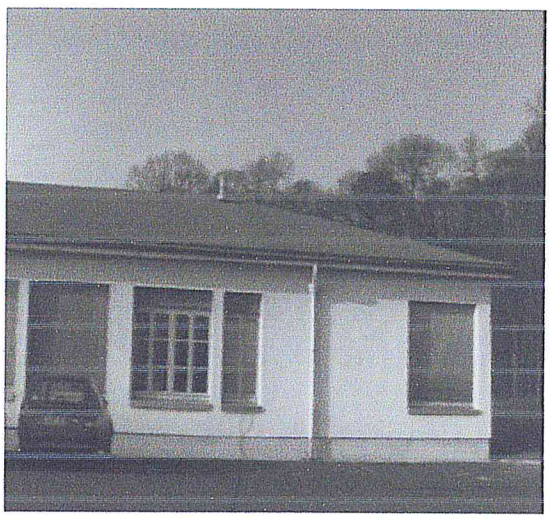


Figure III.15 Image niveau de gris

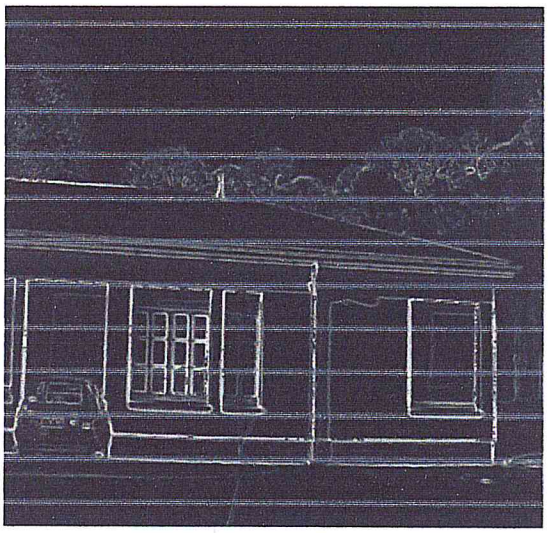


Figure III.16 Image gradient par l'opérateur de Sobel

III.2.2 Le traitement

La phase du traitement consiste à implémenter un algorithme évolutionnaire, et nous avons jeté notre dévolu sur l'algorithme des mouches car ce dernier est plus récent que les algorithmes classiques.

III.2.2.1 L'algorithme des mouches

III.2.2.1.1 Présentation

L'algorithme des mouches fut introduit par Jean Louchet en l'an 2000 il est inspiré des Stratégies d'Evolution (μ, λ)-ES ou de la Programmation Evolutionnaire (μ, λ)-EP **[Louchet, 2000].**

L'algorithme des mouches est un algorithme évolutionnaire utilisé pour la reconstruction 3D stéréoscopique. Inversement aux approches classiques de la reconstruction 3D, où, des points sont extraits de chaque paire stéréos, ensuite, appariés pour déduire la représentation 3D de la scène. L'algorithme des mouches quand à lui, construit potentiellement des modèles 3D de la scène et teste leur cohérence avec les deux images stéréos.

III.2.2.1.2 Application de l'algorithme de mouche pour la reconstruction 3D

Dans le chapitre I, nous avons parlé des algorithmes évolutionnaires ainsi que de leurs différentes étapes, la figure III.17, nous rappelle la démarche d'un algorithme évolutionnaire :

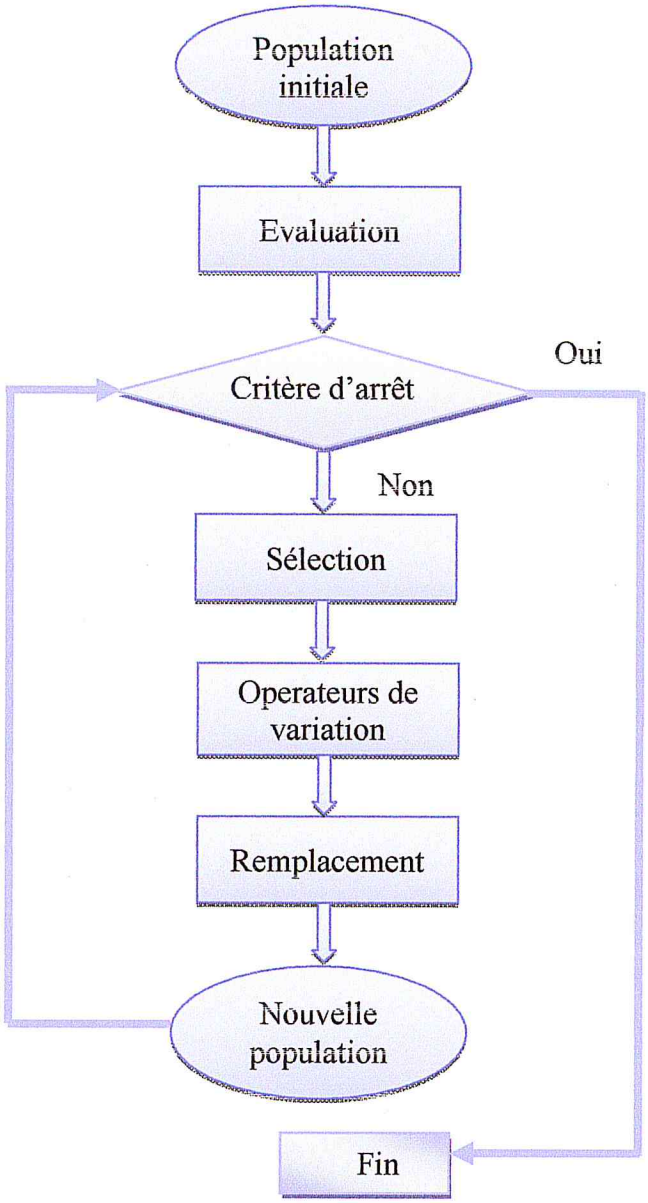


Figure III.17 Schéma global d'un algorithme évolutionnaire

Etape01 : Population Initiale

La population initiale est vue sous forme d'un nuage de points 3D, où chaque point est considéré comme une mouche (individu).

Chaque individu est définie par un chromosome de triplet (x, y, z).

La population des points est initialisée dans le champ de vision commun des deux caméras c.-à-d. l'intersection entre le champ de vision de la caméra gauche et de la caméra droite (voir Figure III.18).

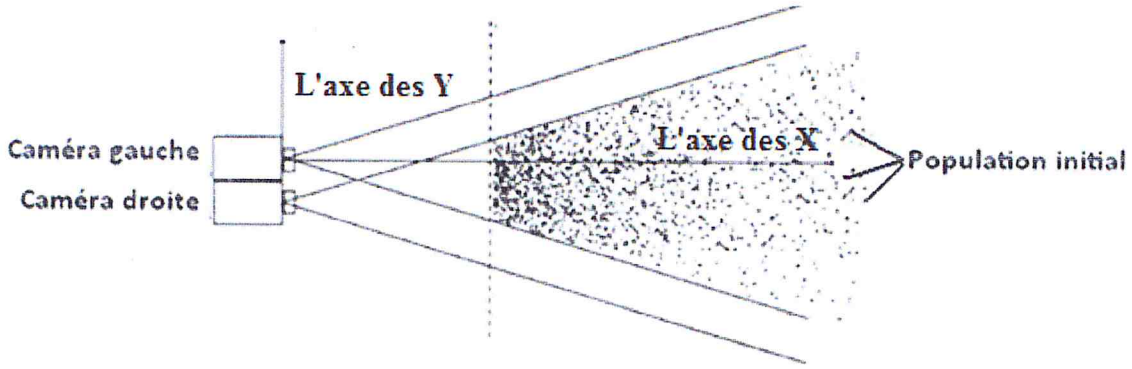


Figure III.18 Initialisation de la population dans le champ d'intersection des deux caméras

Etape 02 : Evaluation des mouches

Chaque individu (mouche) doit être évalué par rapport à une fonction d'évaluation qui s'appelle fonction fitness.

La projection de ces individus sur l'image gauche est donnée par les coordonnées (X_L, Y_L) et sur l'image droite par les coordonnées (X_R, Y_R). Connaissant les paramètres de calibration des caméras, il est aisé de calculer X_L, Y_L, X_R, Y_R à partir de X, Y, Z par géométrie projective (III.5 au niveau du §III.2.1.1.3).

Si la mouche se trouve sur un objet réel dans la scène, alors les pixels correspondants dans les deux images auront normalement le même niveau de gris et un voisinage ressemblant (figure III.19). Inversement, si la mouche n'est pas sur l'objet réel dans la scène, la ressemblance entre le voisinage de ses projections sera faible.

Pour favoriser l'évolution de la population des mouches vers les surfaces apparentes des objets, nous exprimons cela sous la forme d'une fonction objectif qui caractérise la ressemblance du voisinage des projections de la mouche dans les deux images.

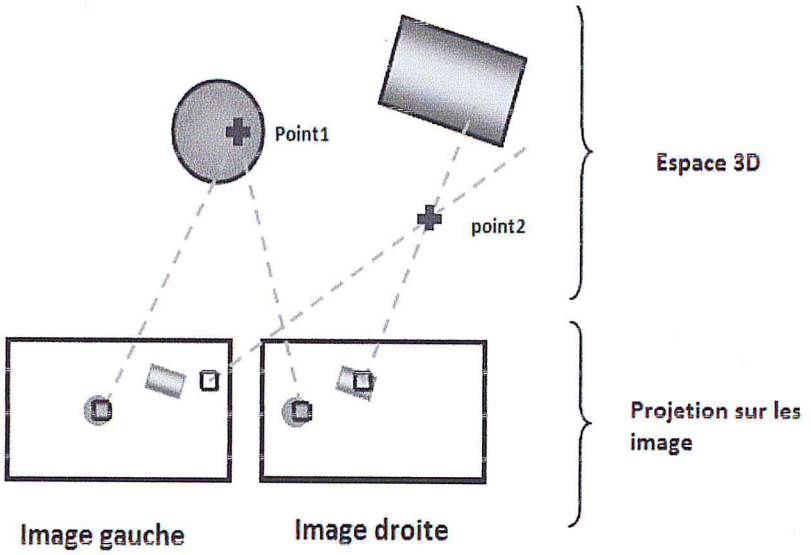


Figure III.19 Projection de deux points 3D sur les deux images

1. La fonction fitness

Nous pouvons maintenant définir la fonction fitness associée à chaque individu représenté par la formule mathématique suivante :

$$fitness(indiv) = \frac{G}{\sum_{(i,j)\in N}(L(X_L+i,Y_L+j)-R(X_R+i,Y_R+j))^2} \qquad \text{(III.11)}$$

- (X_L, Y_L) et (X_R, Y_R) sont les coordonnées des projections droite et gauche de l'individu courant.
- $L(X_L+i, Y_L+j)$ est le niveau de gris de l'image gauche au pixel (X_L+i, Y_L+j), et de même pour R.
- **N** est un voisinage introduit pour que la comparaison des projections soit plus discriminante.
- **G :** est la racine carrée du gradient de Sobel.

2. Exemple

Soit à optimiser la fonction fitness (III.11) :

a. Champs de vision

Le champ de vision commun est défini sur l'intervalle d'entiers :

$X \in [430, 3751]$; $Y \in [-3305, 1626]$; $Z \in [-555, 2034]$.

b. Population initiale

En prenant une population initiale de taille 10, choisie aléatoirement, l'ensemble de la population est donné comme suit

Population = {{520,-2187,335} ;{2120,-1120,1500} ;{3523,-85,509} ;{2790,1389, 426} ; {3777, 391,805} ;{2520,-1244,1188} ;{2223,-846,-194} ;{976, 816,-421} ; {1363,-2645,116} ;{1921,-3217,409}}.

c. La projection

Après la projection des points 3D sur les deux images gauche et droite on a :

- **Les valeurs de projection de l'mage gauche**

Projection2dG = {{-624.410421206294 ; 2074.725118767933} ; {629.24816303692 ;-530.064930408675} ;{264.4118954432565 ; 260.400922672208} ; {71.3836851469568 ; 163.72486283548} ;{132.706599651001 ; 55.23201113554721} ; {449.022341907132 ;-636.01770972159} ;{-254.118010616585 ;-758.921958335444} ; {-172.5295657579664 ; 342.986727076151} ;{46,1497572303219 ; 2719.47177322813} ; {-182.602981488598 ; 3817.72287419839}}.

- **Les valeurs de projection de l'mage droite**

Projection2dD = {{-668.364516053801; 2496.43172110805} ; {540.267881800377 ;-407.987454442572} ;{-138 .427696575345 ; -185.518649570345} ; {107.964648511663 148.800728534074} ;{161.223439123668 ; -27.5409190297374} ; {397.292145040016 ;-470.091590071186} ;{-113.1831950756;-561.940308859008} ; {-115.774084099493 ; 259.98631173005791} ;{23.9759516609486 ; 4121.45540074036} ; {-581.590452120672 ; 8739.37174055481}}.

d. Gradient : calculer le gradient avec l'opérateur de Sobel

Dans notre travail, nous utilisons des images de taille 600×560 pixels, donc on a une grande matrice de niveau de gris N défini comme suit :

$$N = \begin{pmatrix} 19 & 29 & 20 & \cdots & 55 \\ 21 & 135 & 124 & \cdots & 129 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 23 & 53 & 140 & \cdots & 117 \\ 19 & 159 & 152 & \cdots & 116 \end{pmatrix}$$

Afin de calculer la valeur gradient de chaque individu, nous avons appliqué la méthode citée dans le paragraphe III.2.1.3.3 en utilisant les coordonnées de projections ainsi que le niveau de gris de chaque individu, donc nous obtiendrons les valeurs de gradient suivantes :

G= {539, 465, 378, 120, 25, 925, 10920, 1750, 4784, 310}.

e. La fitness

{37.122557003 ; 9.839716911 ; 0.589275181 ; 36.37124252 ; 37.71891403 ; 0.02479558; 33.37654495 ; 4.921626567 ; 0.353565126 ; 36.67530822}.

Etape 03 : La sélection

Nous avons déjà parlé dans le chapitre I des différentes méthodes de sélection qui sont :

- Basées sur la fitness (déterministe)
- Sélection aléatoire
- Sélection par tournoi

Dans notre algorithme nous avons choisi la sélection basée sur la fitness (déterministe), qui consiste à sélectionner les meilleurs individus (environ 50%) qui possèdent les valeurs les plus élevées de la fonction fitness.

Etape04 : Variation

Les opérateurs de variations sont appliqués sur la population sélectionnée. Nous avons utilisé trois opérateurs: le croissement, la mutation et l'immigration.

1- Le croisement

Nous avons utilisé la méthode de croisement par combinaison linéaire (barycentrique) et plus précisément un croisement 1-D, qui est une combinaison linéaire de deux individus. Un croisement 1-D prend deux individus X_1 et X_2 et crée X_3, qui est définie par la formule suivante :

$$X_3 = \lambda X_1 + \mu X_2 \qquad \text{(III.12)}$$

Où λ est tiré uniformément dans l'intervalle [0,1] et $\mu = 1-\lambda$.

Exemple :

Soit P une population de taille 10, et X_1, X_2 deux individus qui appartient à P

Avec $\lambda=0.21$; et $\mu = 1-\lambda = 0.79$

X_1= {520,-2187,335} ; X_2 = {976, 816,-421}

On applique le croisement barycentrique (équation III.12) :

X_3= {109.2,-459.27, 70.35} + {7710.4, 644.64,-332.59}

X_3= {7819.6 ,185.37,-262.24}

2- La mutation

La mutation qu'on a utilisée est la mutation gaussienne, qui consiste à rajouter un bruit Gaussien à tous les gènes d'un individu X.

La formule de la mutation est donnée par :

$$\vec{X} = \vec{X} + \sigma . \vec{N}(\vec{0}, \vec{H}) \qquad \text{(III.13)}$$

Où σ est un paramètre positif appelé pas de la mutation et N (0, H) représente un tirage de la loi normale centrée de matrice de covariance associée à chaque image.

Tout l'art est alors bien sûr dans le choix des paramètres σ et H. Le choix de σ influe directement sur la vitesse de convergence de l'algorithme. Un pas trop petit permet d'obtenir des résultats assez proches de la réalité en termes de fitness, mais après une longue exploration de l'espace. Inversement un σ grand permet une exploration rapide mais donne des résultats grossiers en termes de précision.

Afin de respecter la positivité de σ et d'avoir des variations symétriques par rapport à 1, il faut muter les variables par une loi centrée réduite, donc notre variable devient :

$$X = X + \sigma'.\gamma \qquad \text{(III.14)}$$

Où γ est généré à partir d'une loi normale centrée réduite et d'écart type σ'.

Exemple :

Soit P une population de taille 10, et X un individu qui appartient à P

Avec σ =0.40, N =1.

Donc X = X+ RandG(1,0.40), tel que RandG est un générateur de valeur qui suit la loi normal centré.

Soit X= {2223,-846,-194} + 0,975328922271729

X= {2223.975328922271729, -845.0246717772827, -193,0246710777283}

3- l'immigration

Cette opération est utilisée pour étendre la zone d'exploration dans l'espace de recherche, et créer de nouveaux individus au hasard dans le champ de vision commun des deux caméras. Elle assure une exploration constante de tout l'espace de recherche.

Etape 05 : Le remplacement

Après la production de nouveaux individus, on exécute l'étape de remplacement qui consiste à remplacer toute ou une partie de la population pour produire une nouvelle

Dans notre algorithme, nous avons choisi le remplacement (μ+ λ) (cité dans chapitre I § I.2.7.2.7), donc la population de la génération suivante sera constituée d'individus choisis parmi toute une population entière (individus parent μ+ enfants λ), ensuite l'algorithme recommence de l'étape 1 jusqu'à ce qu'un critère d'arrêt particulier soit atteint.

- **Le critère d'arrêt**

Dans le chapitre I, nous avons vu les différents critères d'arrêts utilisés dans les approches évolutionnaires, nous avons choisis le critère d'arrêt fixé par le nombre de génération, car il permet de voir la convergence de l'algorithme visuellement.

III.2.2.2 Organigramme de l'algorithme utilisé

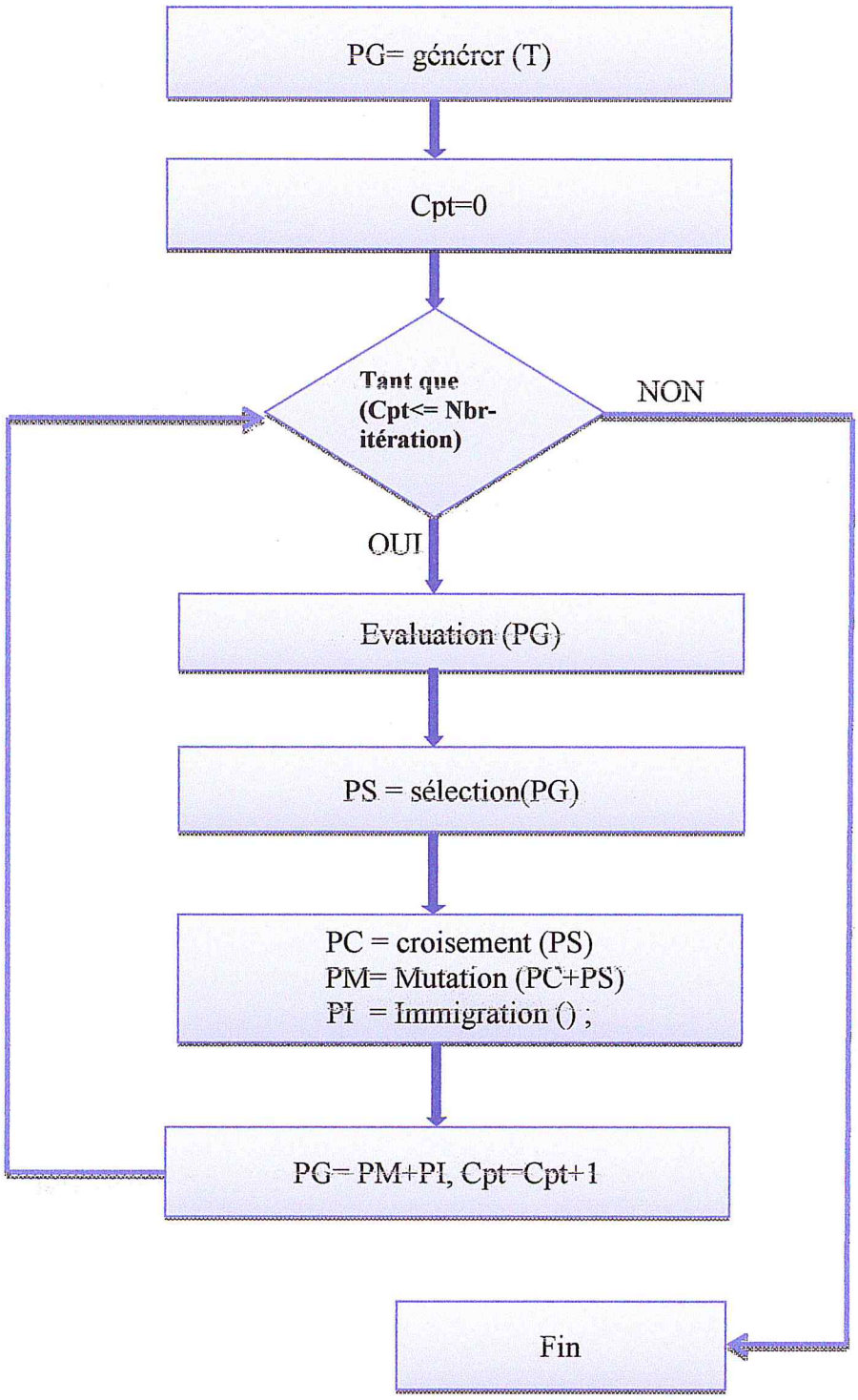


Figure III.20 L'organigramme de l'algorithme des mouches

III.2.2.3 Les pseudos algorithmes développés

Les variables globales utilisées dans les algorithmes développés sont les suivantes :

I_G et **I_D**: Image gauche et image droite.
Nbr_individu: entier; c'est le nombre d'individu qu'on souhaite générer.
Min_X et **Max_X** : entier; sont les extrémités de l'intervalle de la coordonnée X, calculés à partir du champ de vision.
Min_Y et **Max_Y**: entier; sont les extrémités de l'intervalle de la coordonnée Y, calculés à partir du champ de vision.
Min_Z et **Max_Z**: réel; sont les extrémités de l'intervalle de la coordonnée Z, calculés à partir du champ de vision.
Voisinage: réel; calcule la somme des similitudes des niveaux de gris du voisinage de chaque pixel.
$MatG_X$: entier; la matrice du gradient de Sobel pour l'image gauche.
$MatD_X$: entier; la matrice du gradient de Sobel pour l'image droite.
P_crois: réel; la probabilité de croisement.
P_mut: réel; la probabilité de mutation.
PIM: réel ; la probabilité de immigration.
nbr-itération: entier ; le nombre maximum de génération.
i, j : entier ; compteurs d'incrémentation

1. **Algorithme de génération de la population initiale :**

Début
  Pour (i=0 à Nbr_individu)
  Faire
    - Choisir un nombre aléatoire X dans l'intervalle [Min_X, Max_X] ;
    - Choisir un nombre aléatoire Y dans l'intervalle [Min_Y, Max_Y];
    - Choisir un nombre aléatoire Z dans l'intervalle [Min_Z, Max_Z] ;

  Fait ;
Fin ;

2. Algorithme pour le calcul de la fonction fitness

Début

Pour (i = 0 à Nbr_individu)

Faire

- Calculer les valeurs de projection (X_L, Y_L) de l'image gauche et (X_R, Y_R) pour l'image droite pour chaque individu ;
- Voisinage [i] = $(I_G[X_L-1][Y_L-1]-I_D[X_R-1][Y_R-1])^2 +$
  $(I_G[X_L-1)][Y_L]-I_D[X_R-1][Y_R])^2 +$
  $(I_G[X_L-1][Y_L+1]-I_D[X_R-1][Y_R+1])^2 +$
  $(I_G[X_L][Y_L-1]-I_D[X_R][Y_R-1])^2 +$
  $(I_G[X_L][Y_L+1]-I_D[X_R][Y_R+1])^2 +$
  $(I_G[X_L+1][Y_L-1]-I_D[X_R+1][Y_R-1])^2 +$
  $(I_G[X_L+1][Y_L]-I_D[X_R+1][Y_R])^2 +$
  $(I_G[X_L+1][Y_L+1]-I_D[X_R+1][Y_R+1])^2$;
- Fitness= $MatG_V[X_L][Y_L] * MatD_V[X_R][Y_R]$ / voisinage [i];

Fait ;

Fin ;

3. Algorithme de séiection

Début

- Tri décroissant de la population selon les valeurs de fitness ;
- Sélection de la moitié supérieure de la population (les 50% meilleurs individus ;

Fin ;

4. Algorithme de l'opérateur de croisement

Début

Pour (i=0 à Nbr_individu)

Faire

Pour (j=i+1 à Nbr_individu)

Faire

- Générer un nombre aléatoire t dans l'intervalle [0,1] ;

Si (t <P_crois) **alors**

Début

- Gérer un nombre aléatoire λ dans l'intervalle [0,1] ;
- Appliquer le croisement barycentrique entre l'individu courant i et l'individu j en utilisent λ (équation III.12);
- Ajouter le nouveau individu (enfant) dans la population /*population intermédiaire */;

Fin ;

Fin Si ;

Fait ;

Fait ;

Fin ;

5. Algorithme de l'opérateur de mutation

Début

- Entrer les valeurs de σ et H ; /* paramètres du générateur gaussien */

Pour (i=0 à Nbr_individu)
Faire

- Générer un nombre aléatoire n dans l'intervalle [0, 1] ;

Si (n< P_mut) **alors**
Début
- Générer trois valeurs du bruit gaussien en fonction de sigma et H ;
- Appliquer la mutation gaussien sur l'individu courant (équation III.13);
- Ajouter le nouveau individu (enfant) dans la population ;

Fin ;
Fin Si ;

Fait ;

Fin ;

6. Algorithme pour l'opérateur d'immigration

Début

Pour (i = 0 à Nbr_individu)
Faire
- Générer aléatoirement pr-imig ;

Si (pr_imig<PIM) **alors**
- Générer des individus aléatoires
- Choisir un nombre aléatoire X dans l'intervalle $[Min_X, Max_X]$;
- Choisir un nombre aléatoire Y dans l'intervalle $[Min_Y, Max_Y]$;
- Choisir un nombre aléatoire Z dans l'intervalle $[Min_Z, Max_Z]$;

Fin Si ;
Fait ;

Fin ;

7. Algorithme général des mouches

Début

I. PG = création de la population (T) ;

II. Cpt←0;

III. **Tant que** (Cpt < nbr-itération)

Faire

1. Evaluation (PG) ;
2. PS= sélection (PG) ;
3. Individu = croisement (select(PS), select(PS)) ;
   PC=individu croisé ;
4. Pop-tem = PC+ PS ;
5. Individu = Mutation (select (Pop-tem)) ;
   PM= individu Muté ;
6. Pop-tem = Pop-tem + PM ;
7. Individu= Random () ;
   PI= individu immigrants ;
8. PG= Pop-tem + PI ;

Fait ;

Cpt← Cpt +1 ;

Fin ;

- I : la population PG de taille T est initialisée aléatoirement, uniformément dans le champ de vision commun de la caméra de gauche et la caméra de droite.
- II : conteur utilisé pour le test d'arrêt
- IV. A ce stade l'algorithme entre dans la boucle des générations:

  1 : Evaluation des individus

  2 : Sélection des meilleurs individus

  3 : Application de l'operateur de croisement

  4 : Création de la population temporaire

  5 : Application de l'opérateur de mutation

  6 : Remplacement de population temporaire

  7 : Application de l'opérateur de l'immigration

  8 : Remplacement de la population temporaire
- où les enfants sont créés à partir des PG parents, parmi lesquels :
  - PM sont créés par mutation ;
  - PC sont créés par croisement ;
  - PI sont créés par l'opérateur d'immigration.

III.2.2.4 Organigramme de l'application

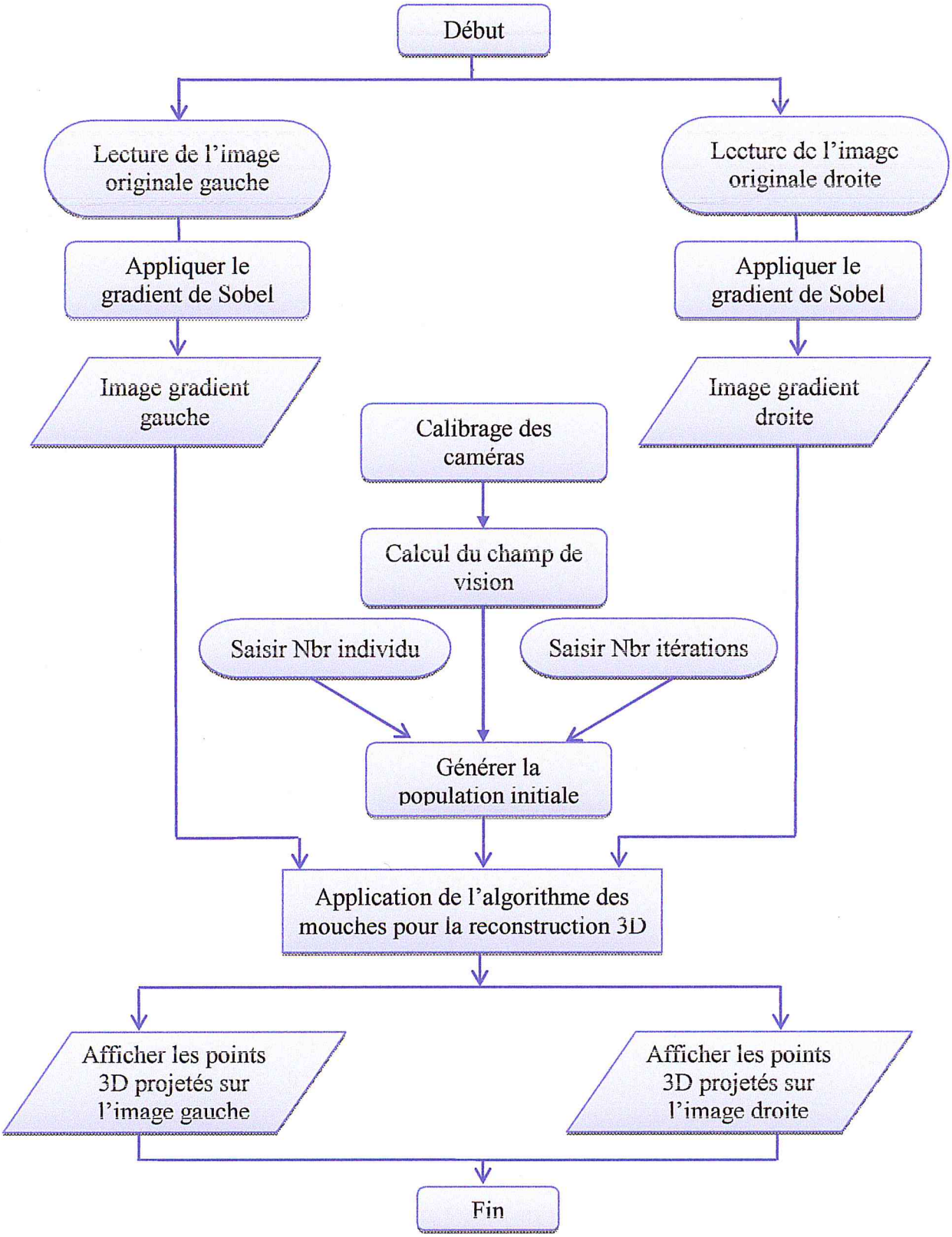


Figure III.21 Organigramme de l'application de la reconstruction 3D par l'algorithme des mouches

III.3 Conclusion

Dans ce chapitre, nous avons évoqué les différents points se rapprochant à la mise en œuvre de notre application de la reconstruction 3D en utilisant l'algorithme des mouches.

En premier lieu, nous avons présenté le calibrage de la caméra, le calcul du champ de vision et une technique classique pour le calcul du gradient avec l'opérateur de Sobel.

Enfin, l'implémentation de l'algorithme des mouches a été faite pour arriver à la reconstruction 3D.

Le chapitre suivant est consacré à l'implémentation de l'algorithme des mouches dans le but de réaliser la reconstruction 3D par l'approche définie dans ce chapitre III.

Chapitre IV
Réalisation

CHAPITRE IV
RÉALISATION

IV.1 Introduction

Dans ce chapitre, des résultats expérimentaux seront illustrés ; nous allons ainsi pouvoir juger notre travail à travers la qualité de ses résultats en appliquant l'algorithme des mouches sur des images stéréos réelles.

Procédons dans un premier lieu à la présentation de l'environnement de développement de notre logiciel, puis à une description de ce dernier, pour une meilleure compréhension et manipulation.

IV.2 Mise en œuvre

Après avoir exposé les différentes techniques retenues pour la réalisation de notre logiciel et expliqué les différentes approches adoptées, nous arrivons à présent à la mise en œuvre de ce dernier.

IV.2.1 Spécification de l'application

Nous rappelons que le but de notre travail est d'implémenter l'algorithme des mouches pour une reconstruction des points 3D à partir des images stéréoscopiques.

IV.2.2 Outils de développement

L'expansion de l'environnement Windows a offert à l'utilisateur un ensemble d'outils performants et faciles à employer. En effet, cela a permis la réalisation de logiciels à interface conviviale.

Concernant le langage de programmation, notre choix s'est porté sur le C++ Builder 6, qui est orienté objet ; ce qui nous a facilité la tâche de développement en adoptant une organisation en classe de notre application.

IV.2.4.1 Description de notre logiciel de la reconstruction 3D

A chaque lancement de l'application sous Windows, correspond l'affichage d'une fenêtre d'accueil de notre application (voir Figure IV, IV.2).

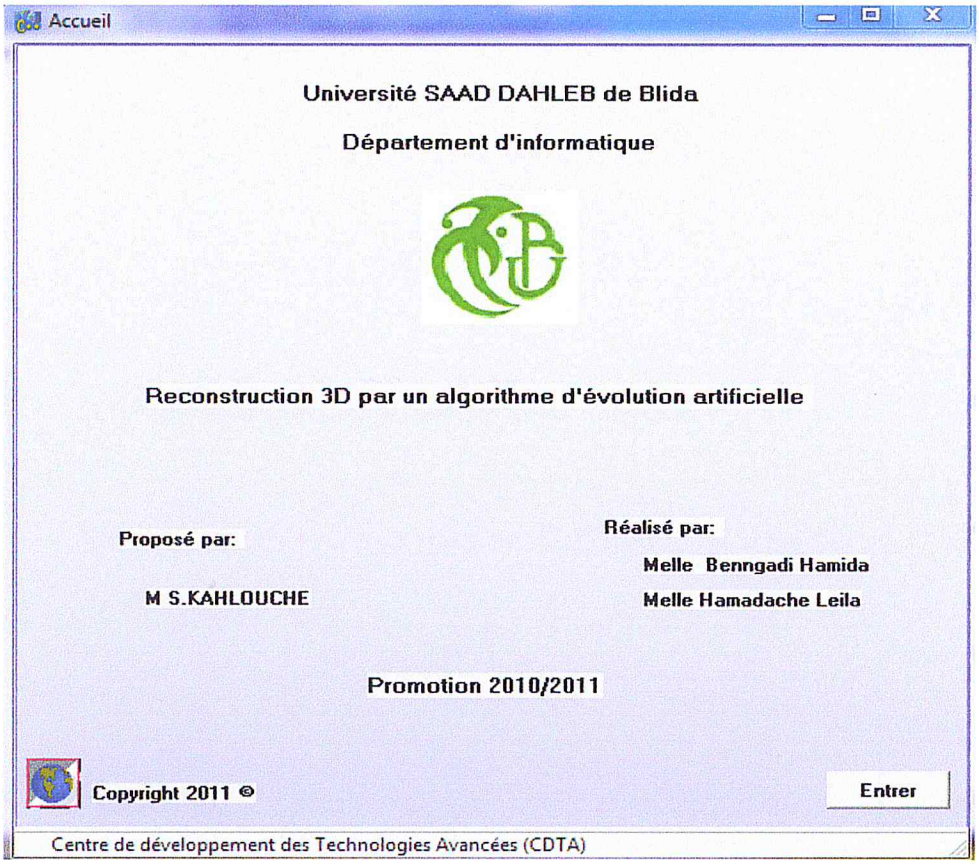


Figure IV.1 Page d'accueil

Le logiciel réalisé englobe notamment :

- Un système de menu déroulant donnant accès à toutes les fonctions du logiciel,
- Une interface conviviale en multi-fenêtrage,
- L'utilisation de la souris pour les différentes manipulations,
- Un ensemble de boites de dialogue permettant une interaction facile avec les commandes,
- Des boutons qui répondent sous forme visuelle et directe aux articles les plus utilisés du menu.

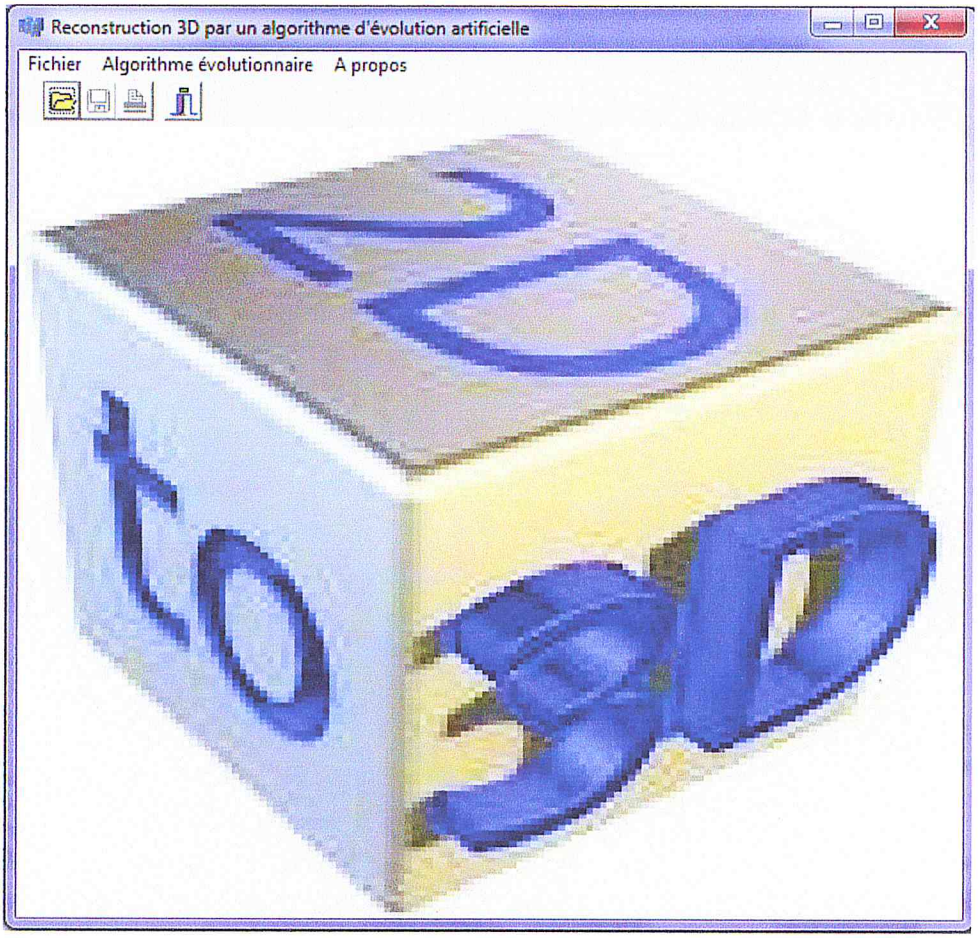


Figure IV.2 Interface de l'application

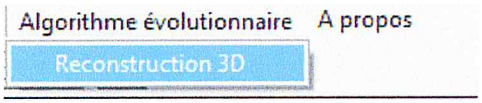
IV.2.4.1.1 Les menus

- **Menu Fichier**

Il permet de configurer une imprimante, de lancer l'impression d'une fenêtre, d'enregistrer les images ou de quitter l'application.

- **Menu Algorithme évolutionnaire**

Le menu algorithme évolutionnaire donne accès à la fenêtre principale de notre application présentée ci-dessous :



- **Menu Apropos**

Il permet de représenter la page d'accueil.

IV.2.4.1.2 La fenêtre de la reconstruction

La fenêtre reconstruction contient plusieurs boutons dont chacun a un rôle très important dans notre application (voir Figure IV.3).

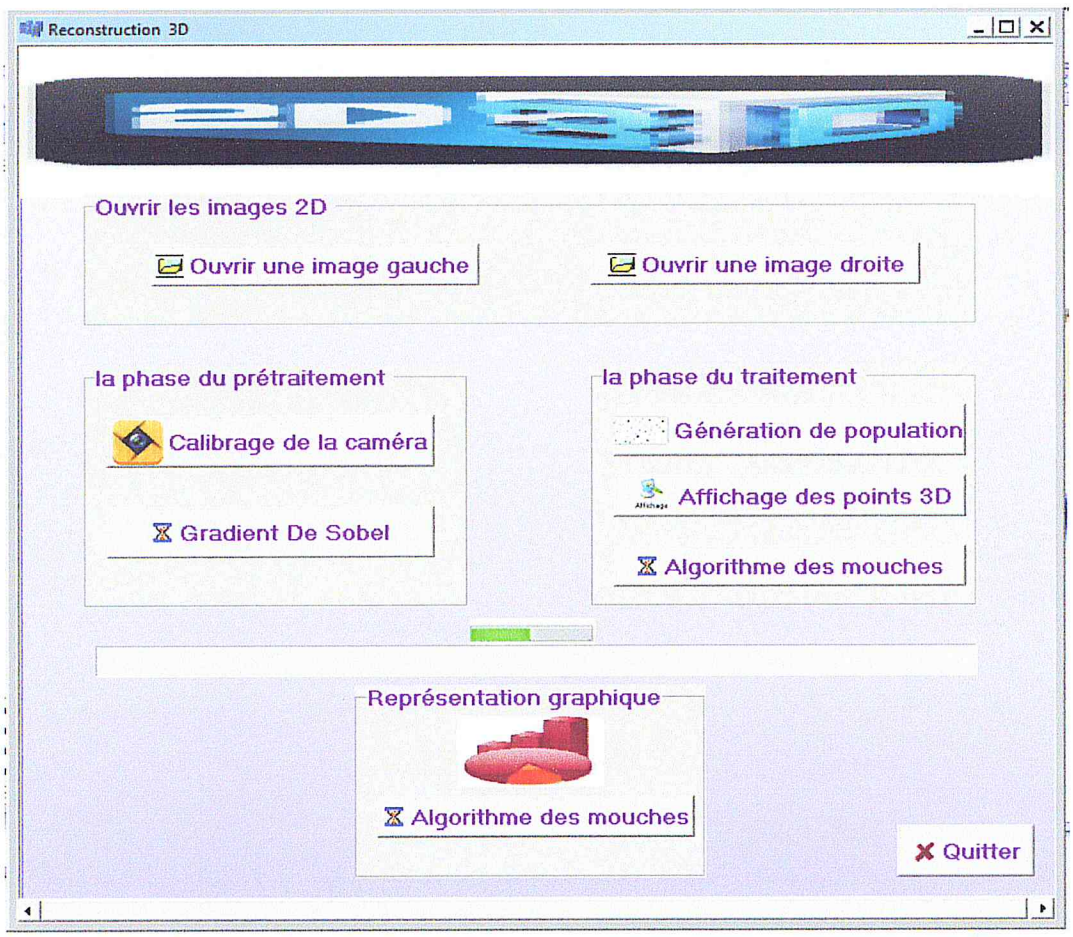


Figure IV.3 La fenêtre de la reconstruction 3D

Ci-contre est représenté un schéma qui nous montre l'organisation de la fenêtre reconstruction (voir Figure IV.4).

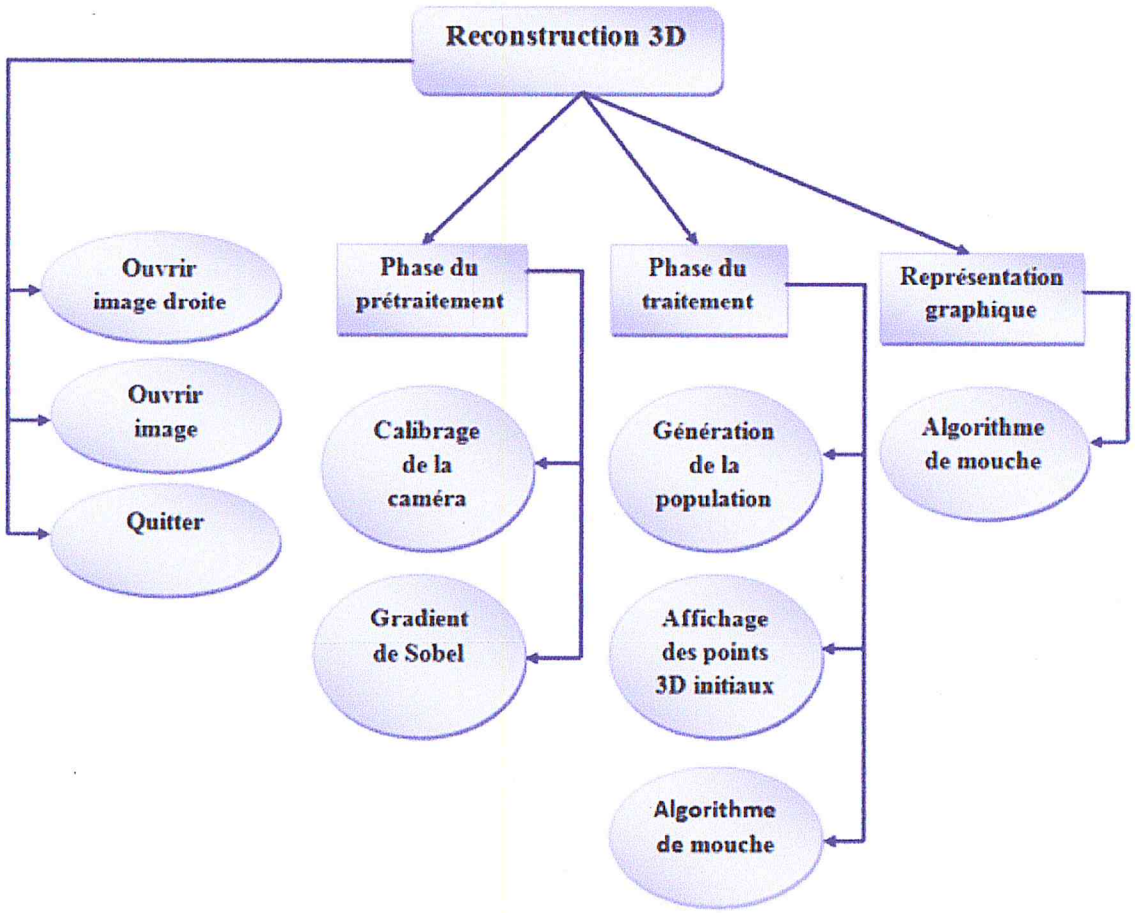


Figure IV.4 Les fonctions de la fenêtre de la reconstruction 3D

- **Le bouton ouvrir une image gauche**

Il permet de choisir une image gauche parmi l'ensemble des images capturées (voir Figure IV.5).

- **Le bouton ouvrir une image droite**

Il permet de choisir une image droite parmi l'ensemble des images capturées (voir Figure IV.5).

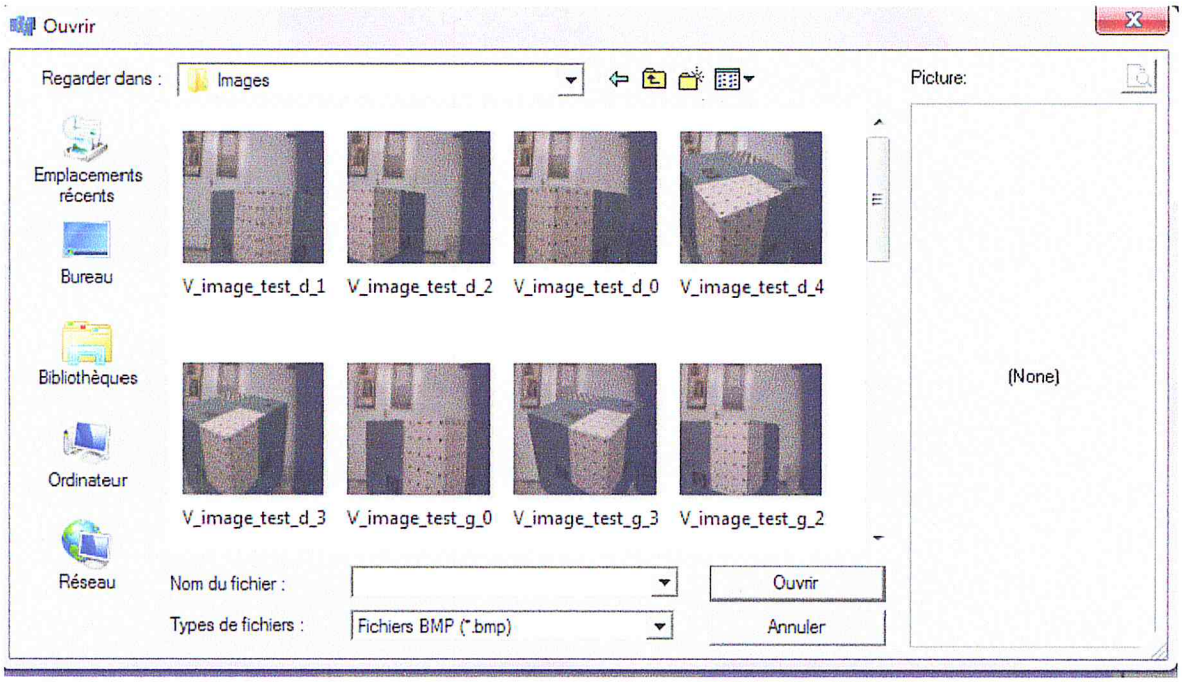


Figure IV.5 Boite de dialogue « ouvrir une image »

- **Phase de prétraitement**

Rappelons que cette phase nous permet de faire le calibrage de la caméra, le calcul du champ de vision et le calcul du gradient de Sobel.

Dans notre application, certaines fonctionnalités nous obligent de présenter des points en 3D pour voir le résultat s'il est bon ou pas, et malheureusement le Builder ne nous permet pas de voir des représentations des points en 3D. A cause de cela, on a fait une interface qui permet de présenter des résultats de calibrage et le calcul de champs de vision avec le langage de programmation Matlab qui est classé parmi les bons éditeurs d'affichage 3D.

- **Représentation du langage Matlab**

Matlab (matrix laboratory) est un système interactif de calcul numérique utilisable comme une calculette et qui dispose d'un grand nombre de fonctions, d'un langage de programmation et un puissant outil de visualisation graphique **[Bernard, 2008]**.

L'intérêt de Matlab tient, d'une part, à sa simplicité d'utilisation : pas de compilation, déclaration implicite des variables utilisées et, d'autre part, à sa richesse fonctionnelle : arithmétique matriciel et nombreuses fonctions de haut niveau dans de nombreux domaines

(analyse numérique, graphique, ...). La programmation sous Matlab consiste à écrire des scripts de commandes Matlab, exécutables dans la fenêtre d'exécution **[9]**.

Matlab contient également une interface graphique puissante, ainsi qu'une grande variété d'algorithmes scientifiques. On peut enrichir Matlab en ajoutant des "boîtes à outils" (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.) **[Alfred ,2004],** Matlab s'enrichit au fur et à mesure.

- **Le bouton calibrage**

Il permet de présenter l'interface du calibrage et le calcul du champ de vision montré dans la figure suivante (Figure IV.5) qui contient sept boutons : résultat de calibrage, paramètre de calibrage, le calcul du champ de vision, la vérification de la projection et deux autres boutons pour la suppression du résultat et enfin un bouton pour quitter l'application. De plus, une barre menu déroulante a été créée dans cette interface pour effectuer des taches telles que l'enregistrement des images, l'impression, le zoom entre autres.

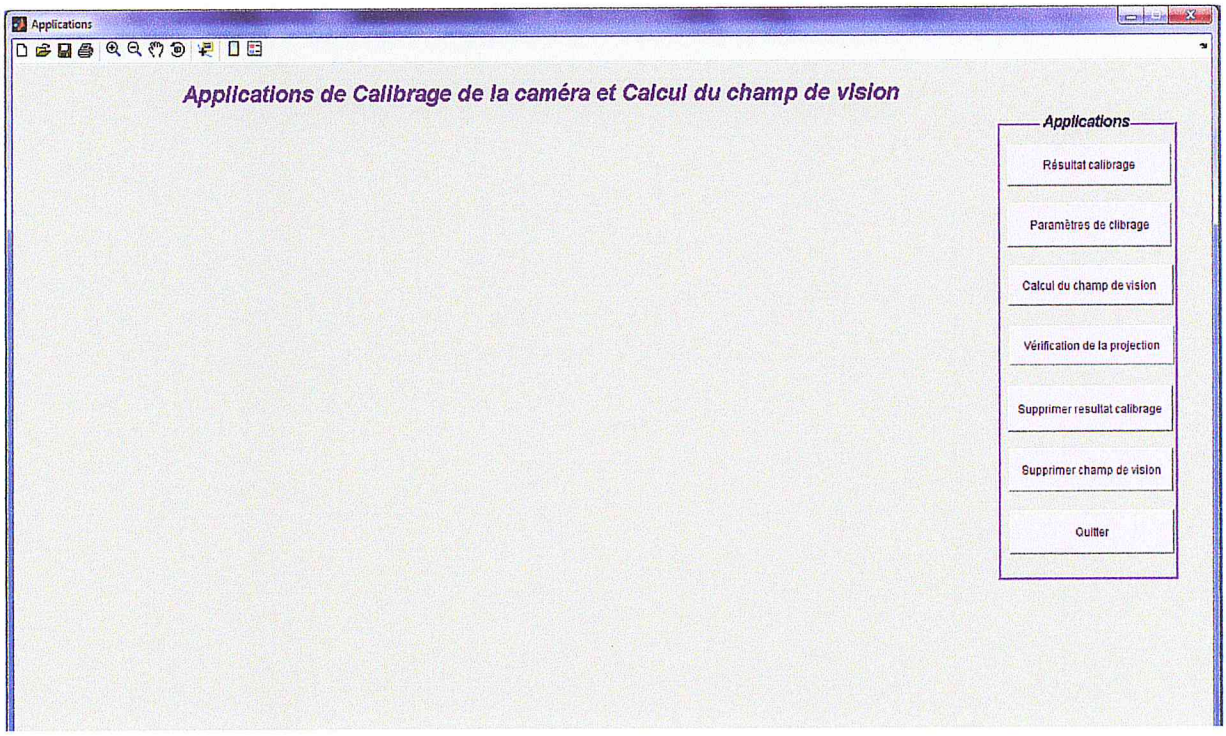


Figure IV.6 Application du calibrage et le calcul du champ de vision

- **Le bouton gradient** :

Il permet de calculer le gradient de Sobel et afficher les images contours (voir Figure IV.3)

- **La phase traitement**

Elle est présentée en 4 boutons :

- **Le bouton génération de la population initiale**

C'est une boite de dialogue qui apparait lorsque l'utilisateur clique sur ce bouton et l'invite à saisir le nombre d'individus et le nombre d'itérations (voir Figure IV.7).

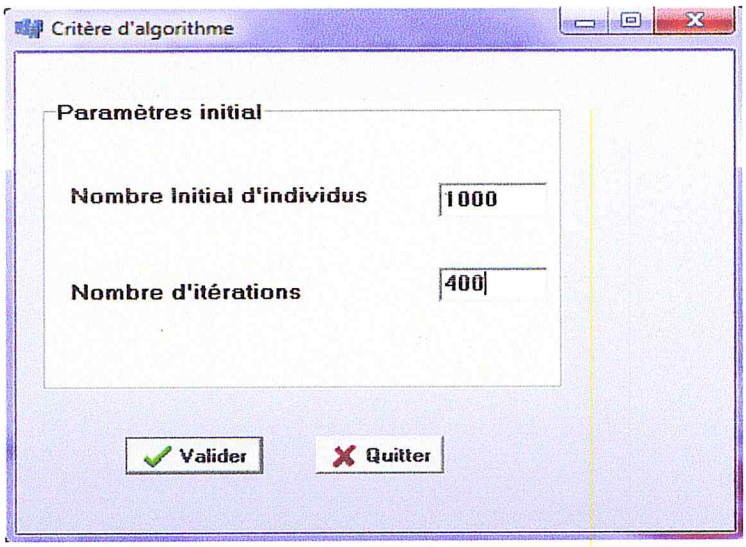


Figure IV.7 Boite de dialogue « paramètre initial »

- **Le bouton affichage de la population**

Il permet d'afficher la population générée aléatoirement.

- **Le bouton algorithme des mouches**

Quand on clique sur ce bouton, une boite de dialogue personnalisée apparait. Elle permet à l'utilisateur de saisir et de changer les paramètres de l'algorithme des mouches, ou de laisser les paramètres par défaut (voir Figure IV.8). Ce bouton permet aussi d'exécuter l'algorithme des mouches.

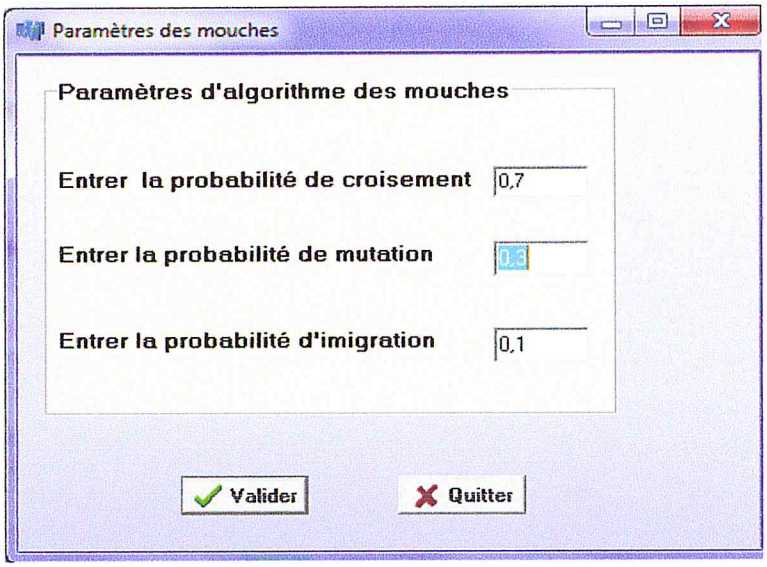


Figure IV.8 Boite de dialogue « paramètre de l'algorithme des mouches »

- **La représentation graphique**

Elle consiste à représenter les graphes des résultats intermédiaires, illustrant la distribution des valeurs fitness dans la population en cours (voir Figure IV.9).

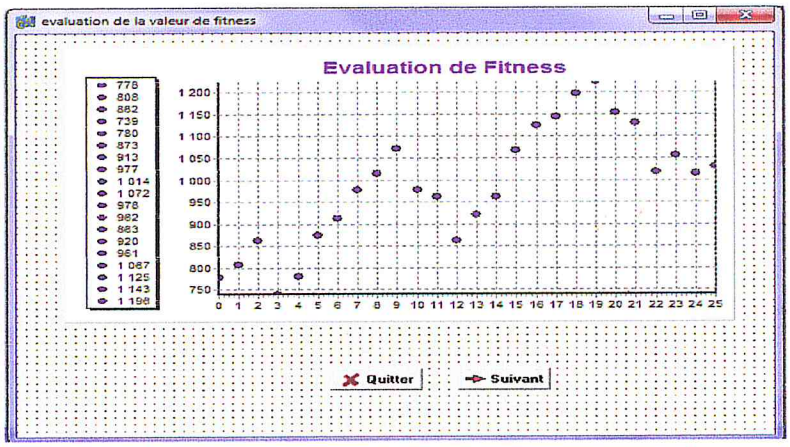


Figure IV.9 Représentation graphique de l'évolution de l'application

- **Le bouton quitter**

Il permet de quitter la fenêtre de reconstruction.

IV.2.5 Matériel et architecture

- **Machine utilisée**

Pour le développement de notre application, nous avons disposé d'un micro-ordinateur dont les caractéristiques sont :

- Un micro-processeur Intel ® core (TM) 2 duo CPU T5870 @2.00 GHz 2.00GHz
- Mémoire installée (RAM) 2.00 GHz
- Système d'exploitation: Windows 7 Edition Intégral
- Type du système : system d'exploitation 32 bits
- Disque dur 250 GO
- Carte graphique ATI 1 GO.

IV.2.6 Expérimentation

Lors de la phase d'application de l'algorithme des mouches à notre cas, nous avons remarqué qu'il n'y'a pas une convergence totale. Cette dernière n'est que partielle. De plus, il a été remarqué un grand temps de traitement. Notons que le temps de traitement est parmi les paramètres critiques qui valident la performance d'une solution implémentée pour résoudre un problème évolutionnaire.

La réalisation de plusieurs essais de l'algorithme des mouches en invariant les différents paramètres, nous a conduit à conclure que cette convergence partielle est due entre autres aux paramètres suivants :

- **La taille des images** : L'algorithme des mouches doit parcourir l'image point par point selon le nombre de points généré aléatoirement pour la population initiale. Dans notre cas, nous avons utilisé des images de taille (600 × 560 pixels), qui nécessite la génération d'un grand nombre d'individu initiale pour couvrir le maximum de la taille de l'image, ce qui veut dire, que plus l'image à une grande dimension, plus le temps de traitement est important.

- **La capacité mémoire** : dans ce genre d'algorithme, pour atteindre le résultat, il faut passer par un certain nombre d'itérations, telles que chacune d'elle applique des opérateurs de variations (croisement, mutation et immigration) permettant de générer plusieurs individus et augmenter ainsi la taille de la population qui nécessitent une grande capacité mémoire. En d'autres termes, plus la population est grande plus la taille de la mémoire doit être grande. Aussi, le temps de traitement devient important.

- **Le champ de vision** : pour l'algorithme des mouches, le champ de vision est considéré comme un paramètre clé, il permet de définir l'intervalle de génération aléatoire de la population initiale. Le champ de vision utilisé dans notre application est calculé par rapport au système de vision ATRV2 (cité dans le chapitre III §III.2.1.1.2). nous avons remarqué que ce champ couvre une partie de l'image ce qui ne permet de reconstruire que cette partie. Pour atteindre un champ de vision plus large, il est nécessaire de modifier la position des deux caméras, chose qui n'est pas possible dans notre cas parce que le système de vision est figé.

IV.2.7 Tests et résultats

Le jugement d'aboutissement des résultats dans notre cas est essentiellement visuel et ceci en regardant la convergence des points 3D projeter sur les deux images vers les contours de chaque image.

Néanmoins notre application prend en charge les différentes tailles d'image possible. Les tests de la reconstruction 3D par l'algorithme des mouches sont effectués sur des images de taille (600×560 pixels),

Nous pouvons diviser nos résultats en deux phases qui sont les tests du prétraitement et les tests du traitement en commençant par les tests de prétraitement.

- **Tests de prétraitements**

A l'aide de la figure IV.6 qui représente l'application pour le calcul du calibrage, le champ de vision ainsi la vérification des projections des points générés on a :

- **Le bouton calibrage**
- Si on clique sur le bouton résultat du calibrage on a (Figure IV.10) :

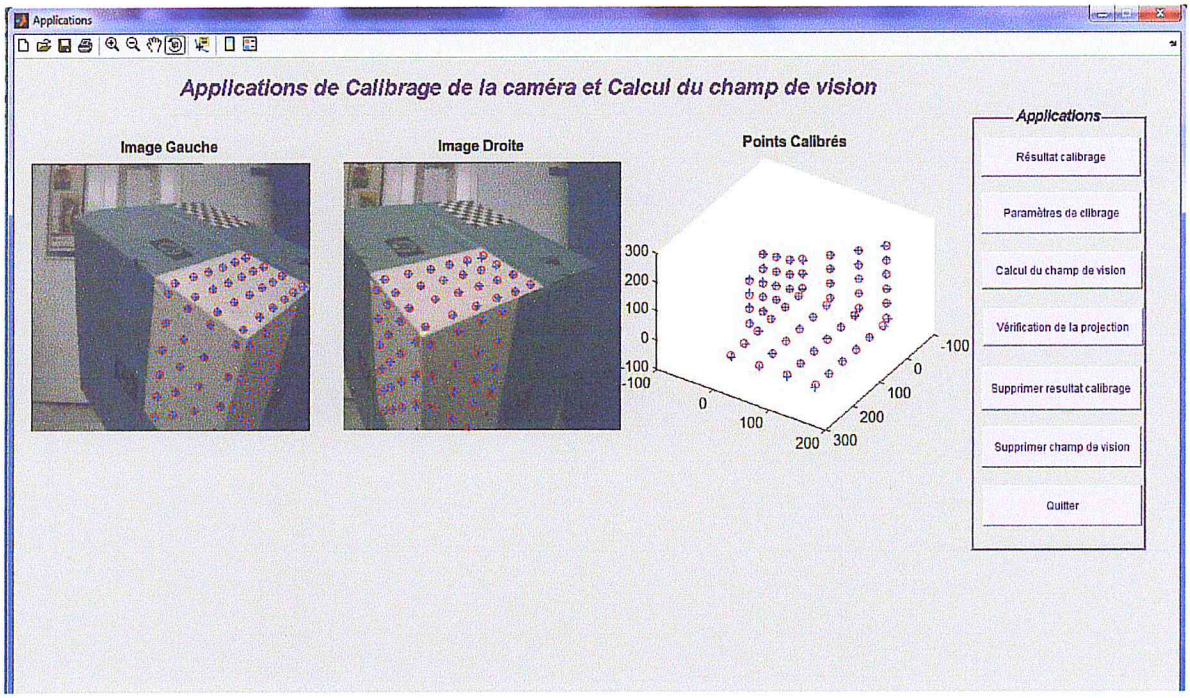
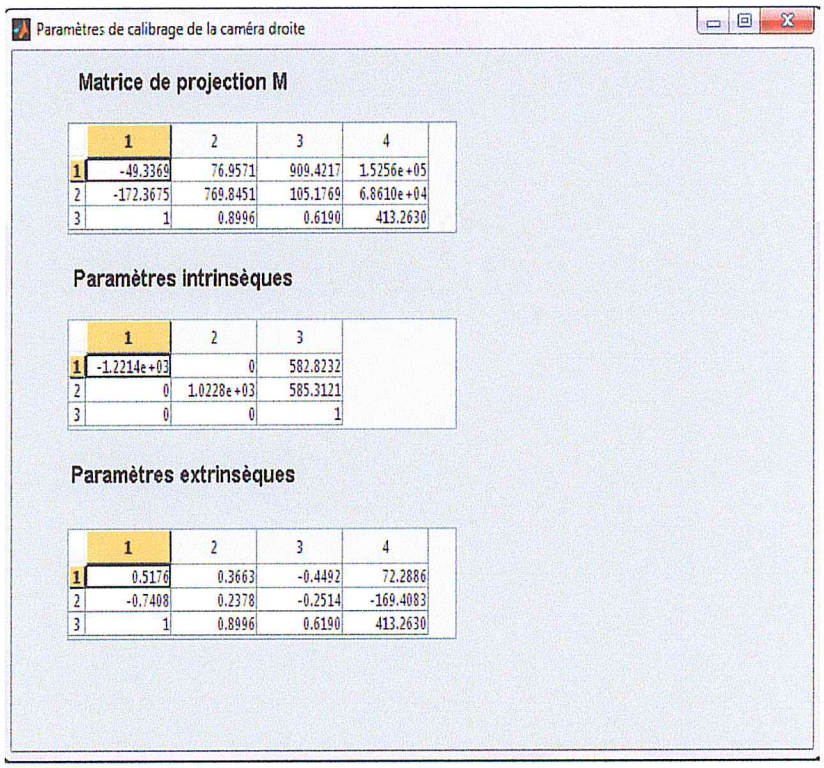


Figure IV.10 Validation 3D graphique des résultats du calibrage

- Si on clique sur le bouton paramètre de calibrage on a (Figure IV.11, IV.12):

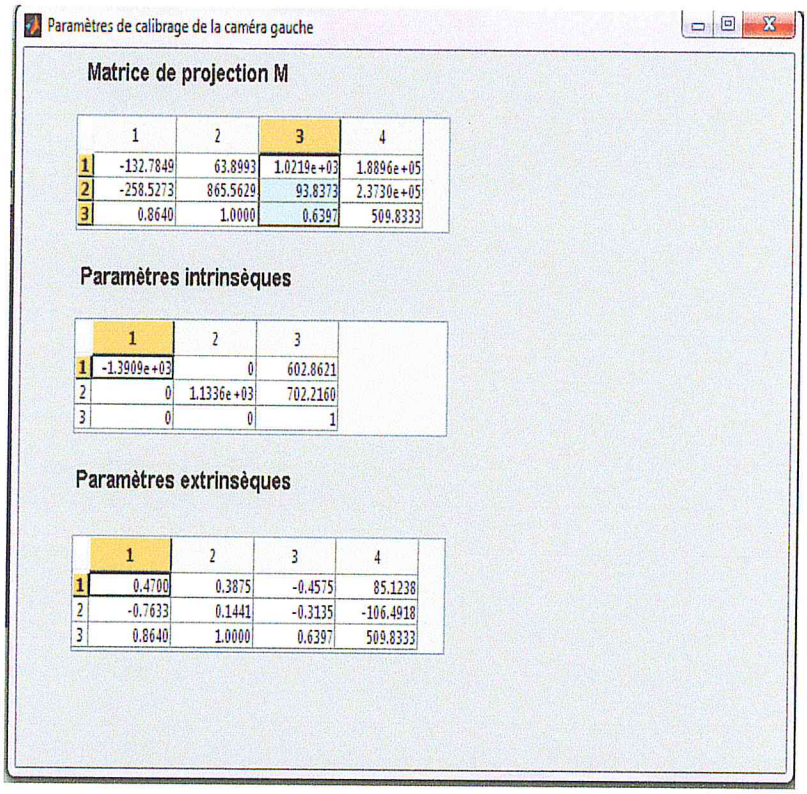


	1	2	3	4
1	-49.3369	76.9571	909.4217	1.5256e+05
2	-172.3675	769.8451	105.1769	6.8610e+04
3	1	0.8996	0.6190	413.2630

	1	2	3
1	-1.2214e+03	0	582.8232
2	0	1.0228e+03	585.3121
3	0	0	1

	1	2	3	4
1	0.5176	0.3663	-0.4492	72.2886
2	-0.7408	0.2378	-0.2514	-169.4083
3	1	0.8996	0.6190	413.2630

Figure IV.11 Paramètres de calibrage pour l'image droite



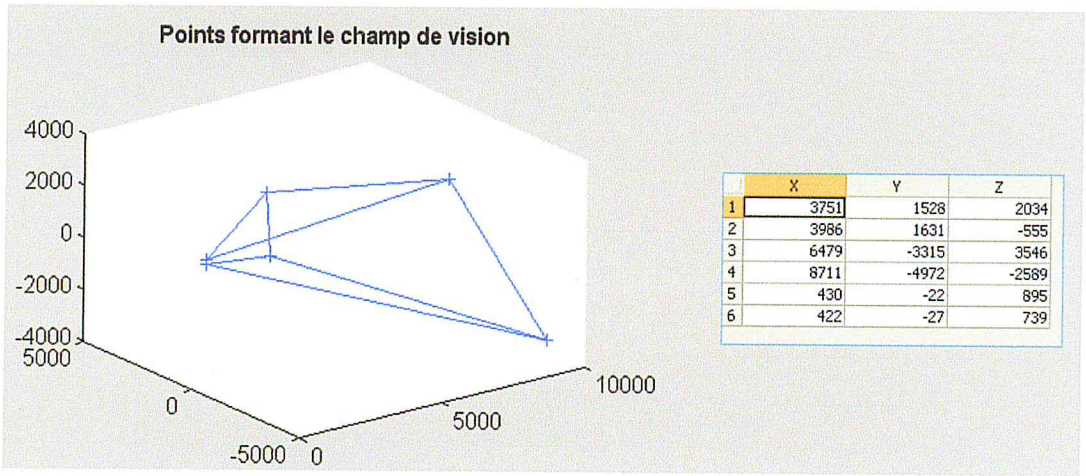
	1	2	3	4
1	-132.7849	63.8993	1.0219e+03	1.8896e+05
2	-258.5273	865.5629	93.8373	2.3730e+05
3	0.8640	1.0000	0.6397	509.8333

	1	2	3
1	-1.3909e+03	0	602.8621
2	0	1.1336e+03	702.2160
3	0	0	1

	1	2	3	4
1	0.4700	0.3875	-0.4575	85.1238
2	-0.7633	0.1441	-0.3135	-106.4918
3	0.8640	1.0000	0.6397	509.8333

Figure IV.12 Paramètres de calibrage pour l'image gauche

- Si on clique sur le bouton calcul du champ de vision on obtient le champ de vision commun entre les deux caméras comme on le voit dans la Figure IV.13 : c'est l'intersection des 6 points.



	X	Y	Z
1	3751	1528	2034
2	3986	1631	-555
3	6479	-3315	3546
4	8711	-4972	-2589
5	430	-22	895
6	422	-27	739

Figure IV.13 Calcul du champ de vision commun

- Si on clique sur le bouton vérification de la projection on obtient une fenêtre (Figure IV.14) qui contient un cube en 3D et respectivement ses projections gauche et droite à l'aide des formules de projection en utilisant les résultats de calibrage.

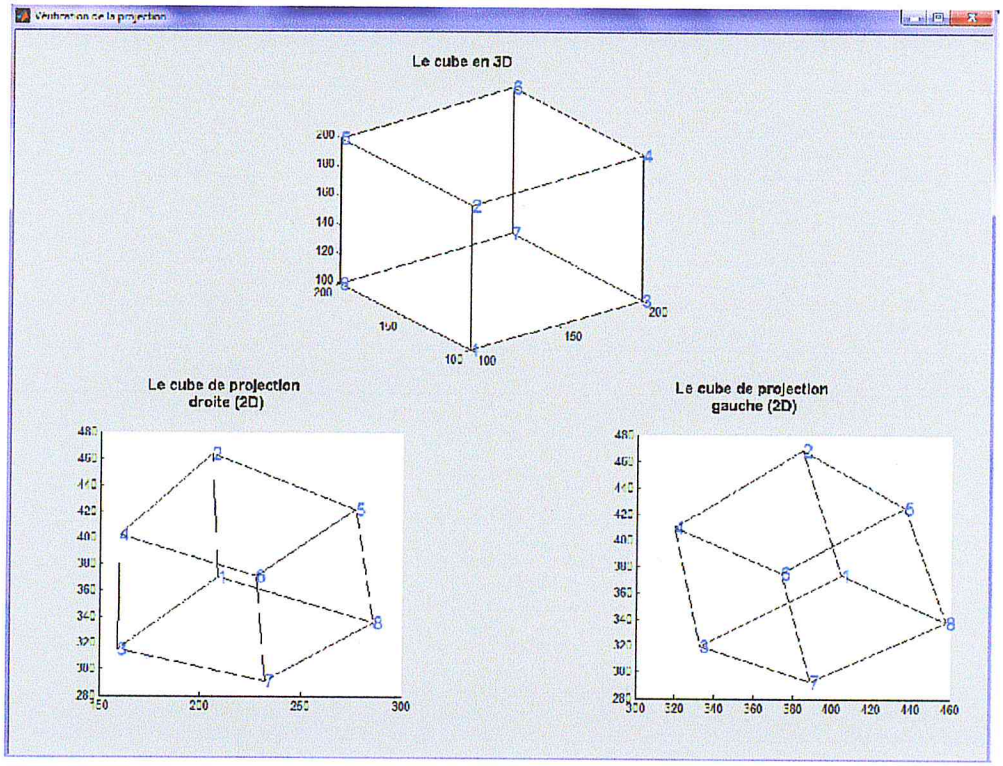


Figure IV.14 Vérification de la projection

- **Le bouton gradient**

L'application du gradient sur une image (parmi les images utilisées dans l'application) en niveau de gris est donnée comme suit (Figure IV.15, IV.16, IV.17, IV.18) :

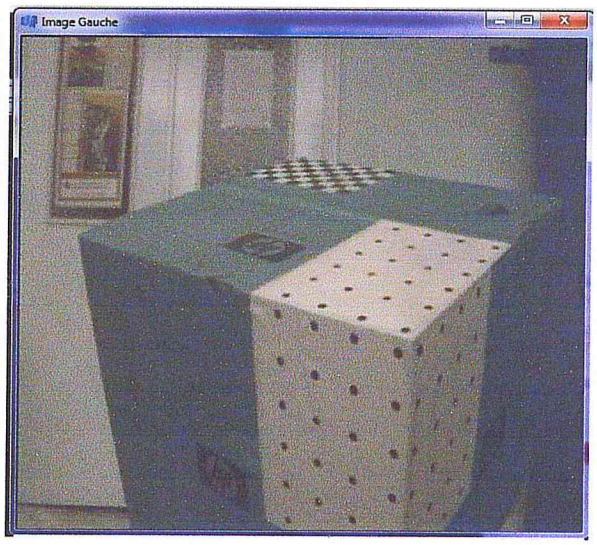


Figure IV.15 Image initiale gauche

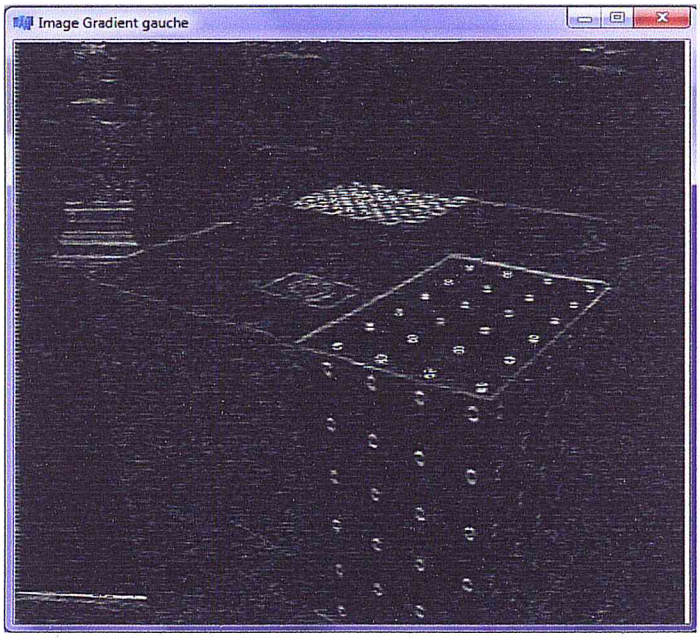


Figure IV.16 Image gradient horizontale

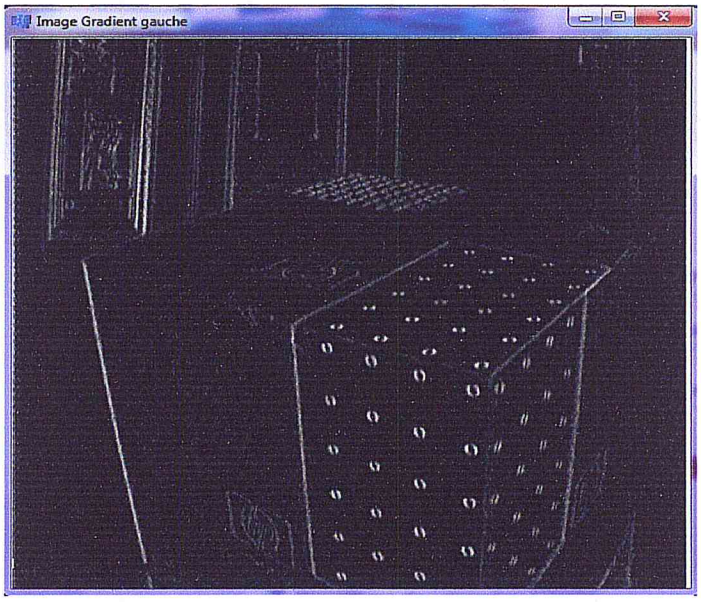


Figure IV.17 Image gradient verticale

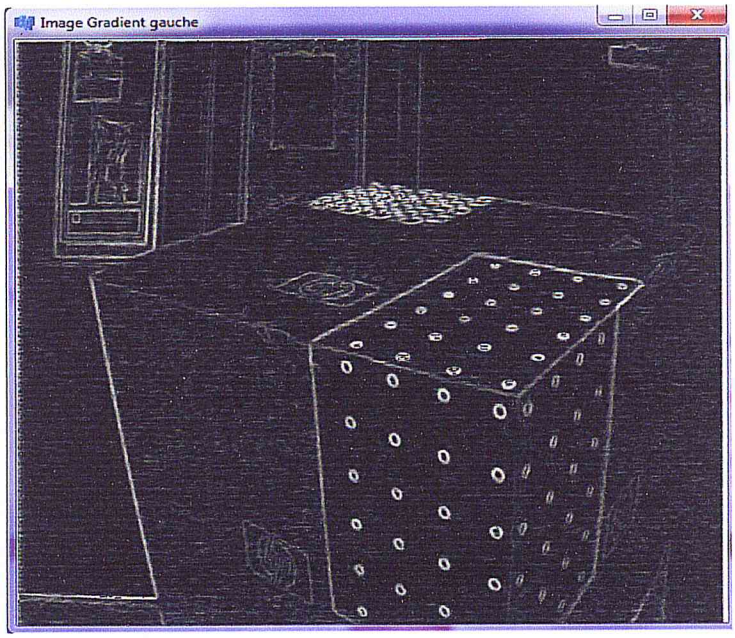


Figure IV.18 Image gradient horizontale et verticale

- **Phase traitement**

Après le clic sur le bouton génération de la population initiale, on a généré une population aléatoire et avec le bouton affichage on peut l'afficher (Figure IV.19).
NB=2000 individu

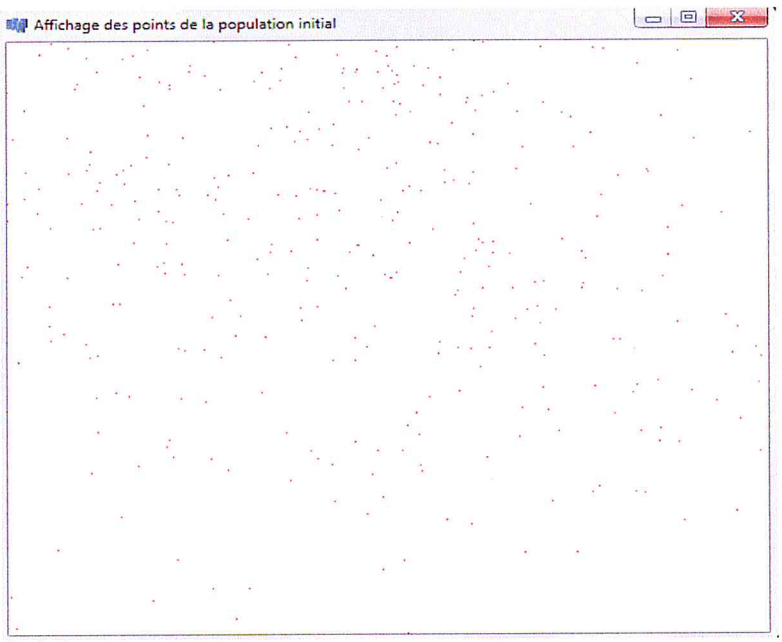


Figure IV.19 Affichage de la population générée

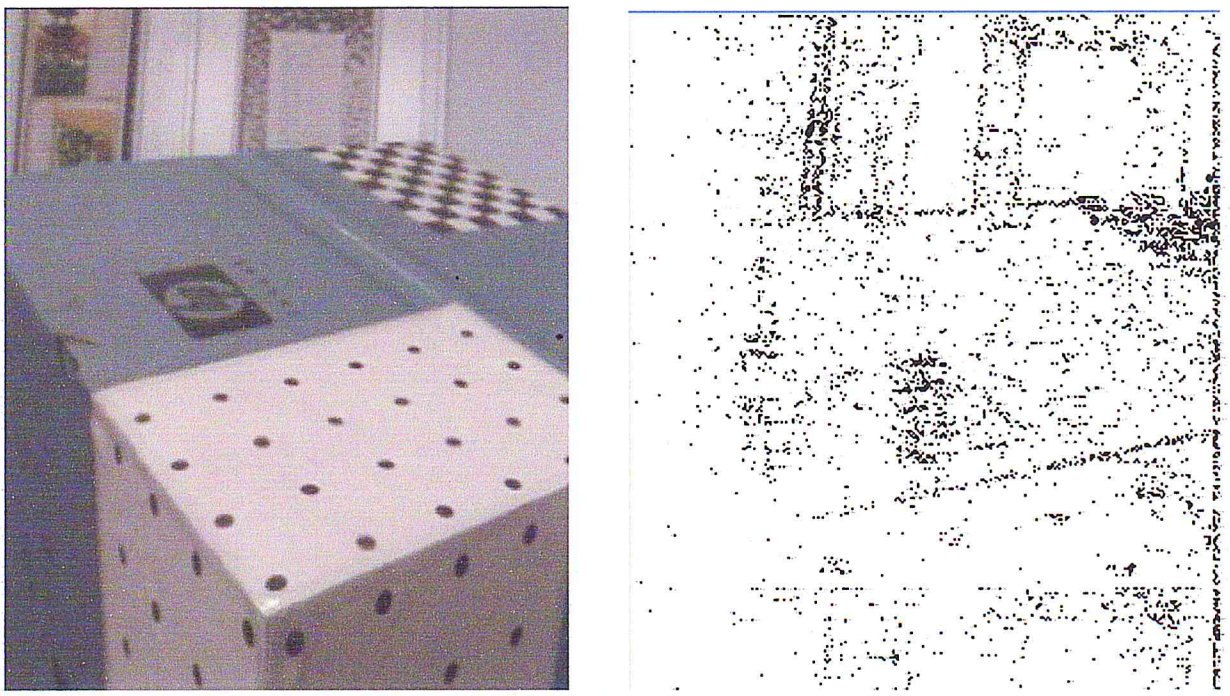
- **Les résultats de l'algorithme des mouches**

Pour les résultats ci-dessus on fixe la probabilité de croisement et la probabilité de mutation et on varie la taille de la population et le nombre d'itérations.

- **Résultat 1**: Taille de la population : 1300 individus, Nombre d'itération : 80, Temps d'exécution : 20 m.

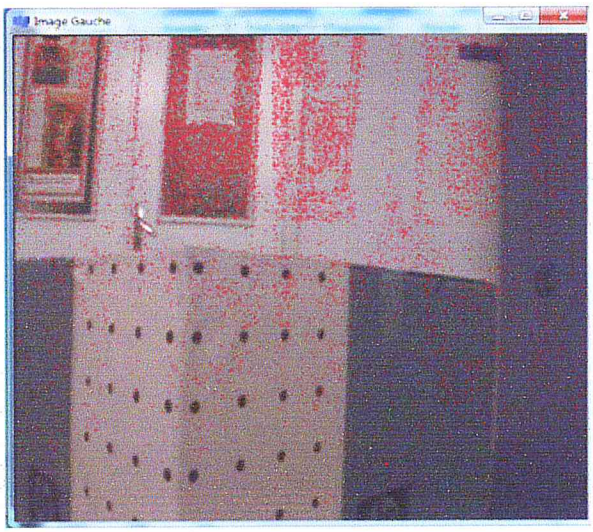
(a)

- **Résultat 2**: Taille de la population : 1800 individus, Nombre d'itération : 800, Temps d'exécution : 45 m.



(b)

- **Résultat 3**: Taille de la population : 1200 individus, Nombre d'itération : 350, Temps d'exécutions : 1h.

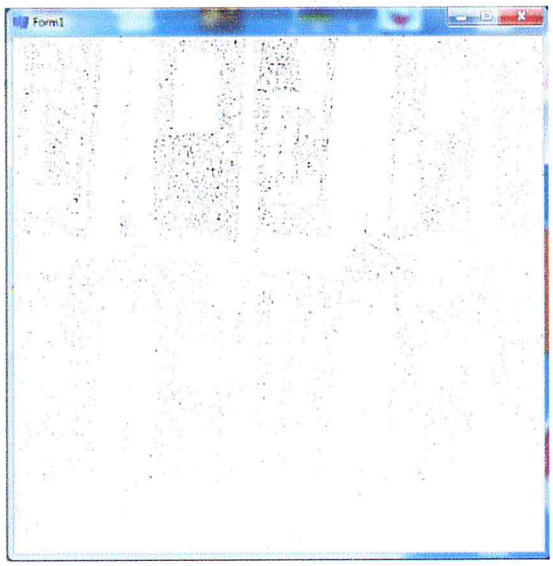


(c)

- **Résultat 4**: Taille de la population : 1500 individus, Nombre d'itération : 200, Temps d'exécution : 30m.

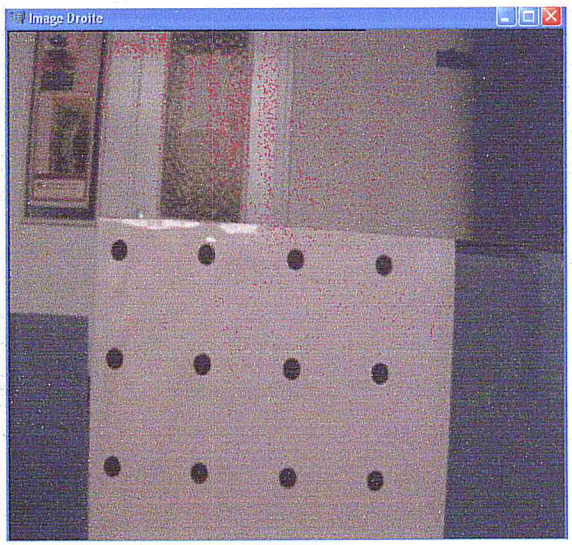
(d)

- **Résultat 5**: Taille de la population : 2000 individus, Nombre d'itération : 200, Temps d'exécution : 1h 05m.



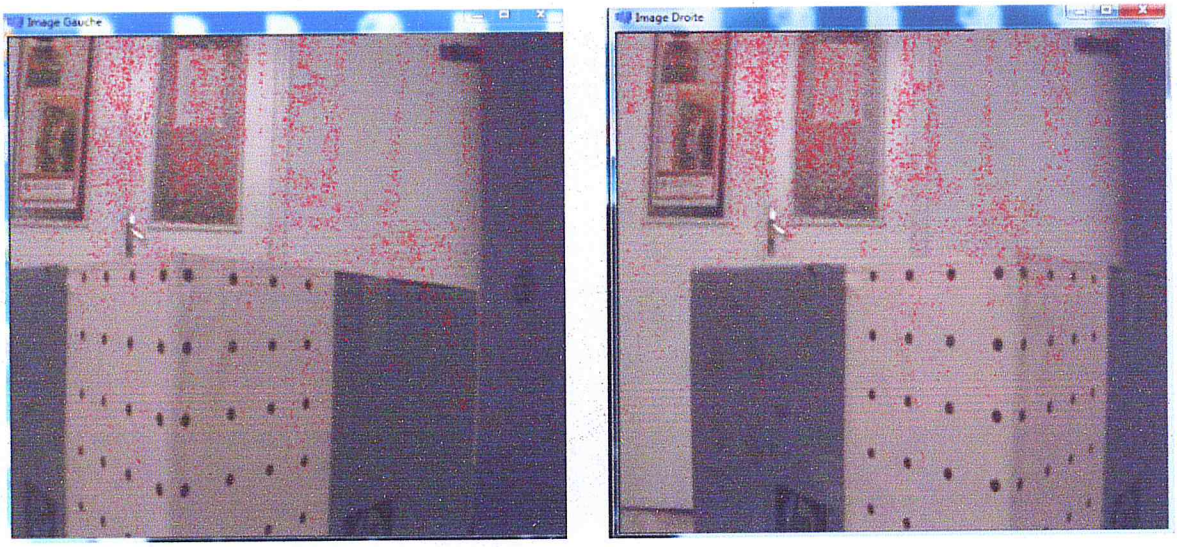
(e)

- **Résultat 6**: Taille de la population : 2000 individus, Nombre d'itération : 500, Temps d'exécution : 1h 10m.



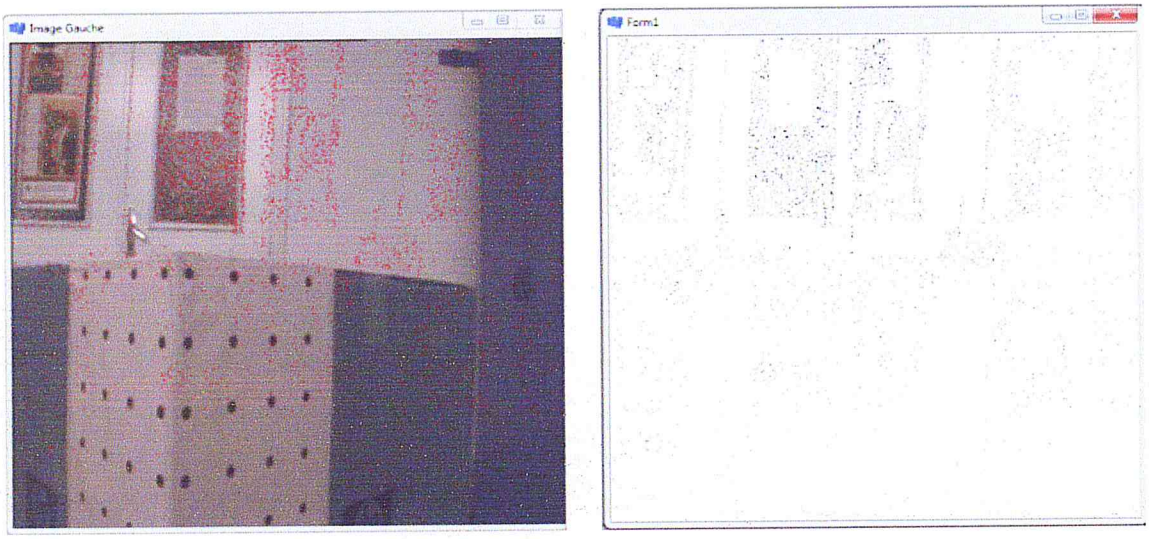
(f)

- **Résultat 7**: Taille de la population : 4000 individus, Nombre d'itération : 100, Temps d'exécution : 1h 20m.



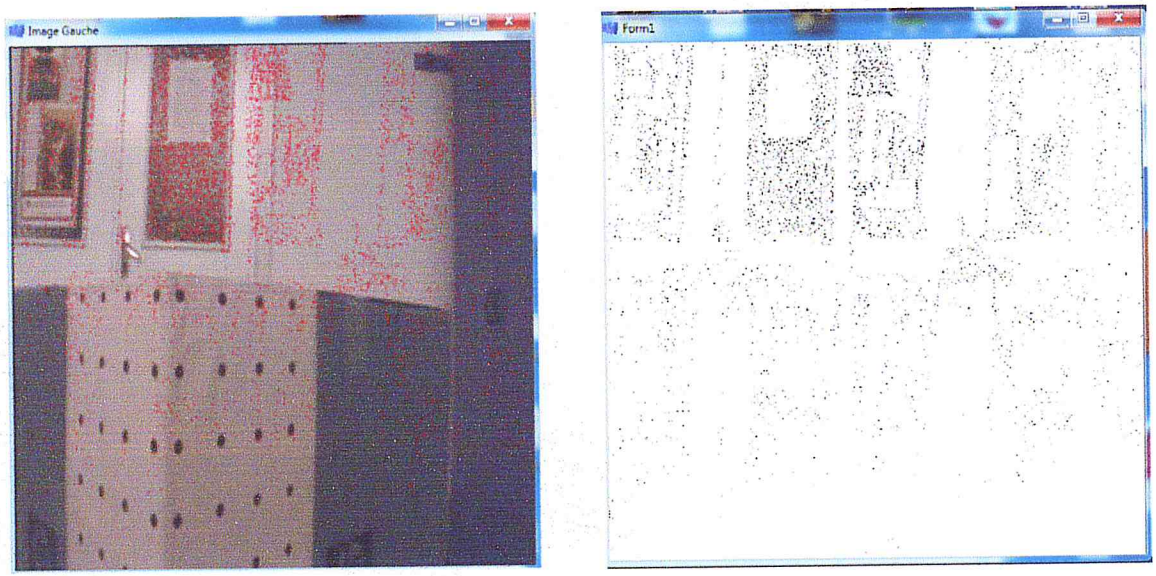
(g)

- **Résultat 8**: Taille de la population : 1500 individus, Nombre d'itération : 200, Temps d'exécution : 45m.



(h)

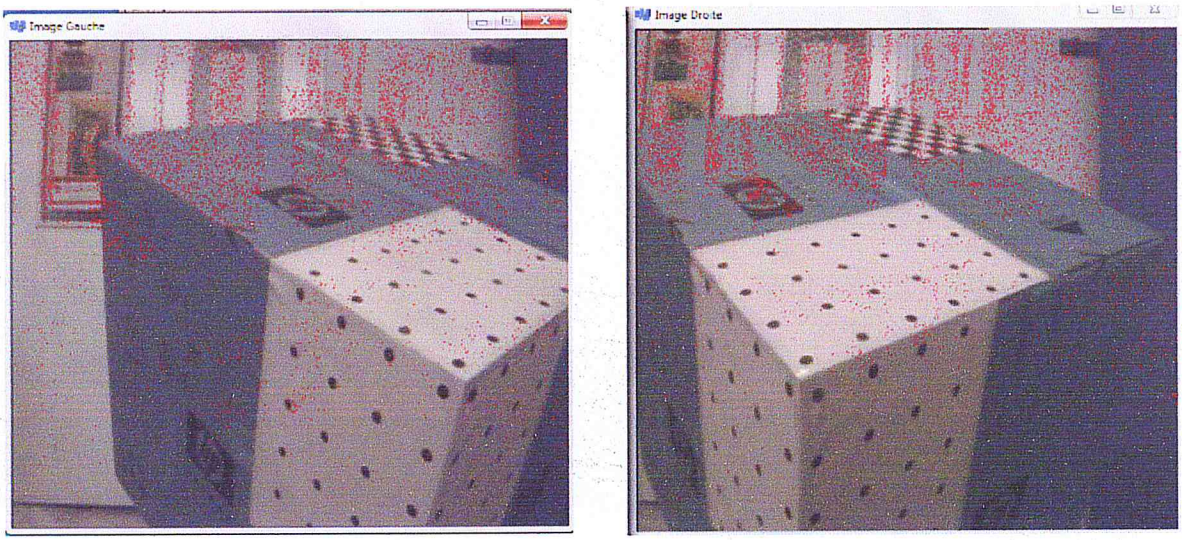
- **Résultat 9**: Taille de la population 1800 individus, Nombre d'itération : 300, Temps d'exécution : 50m.



(i)

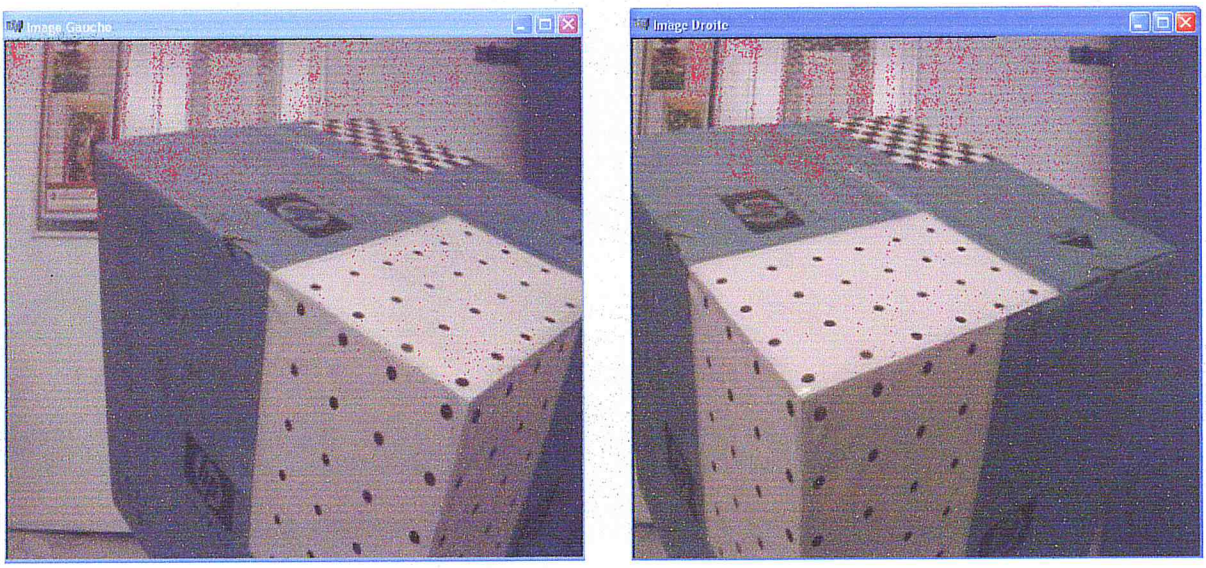
Les résultats présentés ci-dessous ((a)…(i)) sont des résultats obtenus d'une population qui varie entre 1000 et 4000 individu. Nous remarquons que les individus sont concentrés vers les contours de chaque image mais la convergence des points sur les contours se trouve dans la partie supérieure gauche de chaque image à cause du champ de vision commun des deux caméras qui ne couvre pas toute l'image mais juste la partie supérieure gauche. Mais pour trouver une bonne convergence, il faut aller plus loin dans le nombre d'individus et le nombre d'itérations (voir résultat 13 (n)) et ça va prendre un temps d'exécution trop long et une grande capacité mémoire. Par ailleurs, nous reprenons ci-dessous, d'autres résultats :

- **Résultat 10**: Taille de la population 8000 individus, Nombre d'itération : 10, Temps d'exécution : 30m.



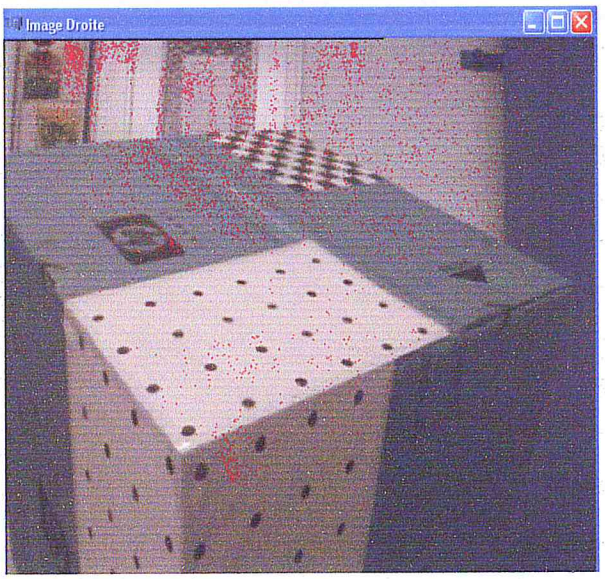
(j)

- **Résultat 11**: Taille de la population : 10000 individus, Nombre d'itération : 500, Temps d'exécution : 50m.



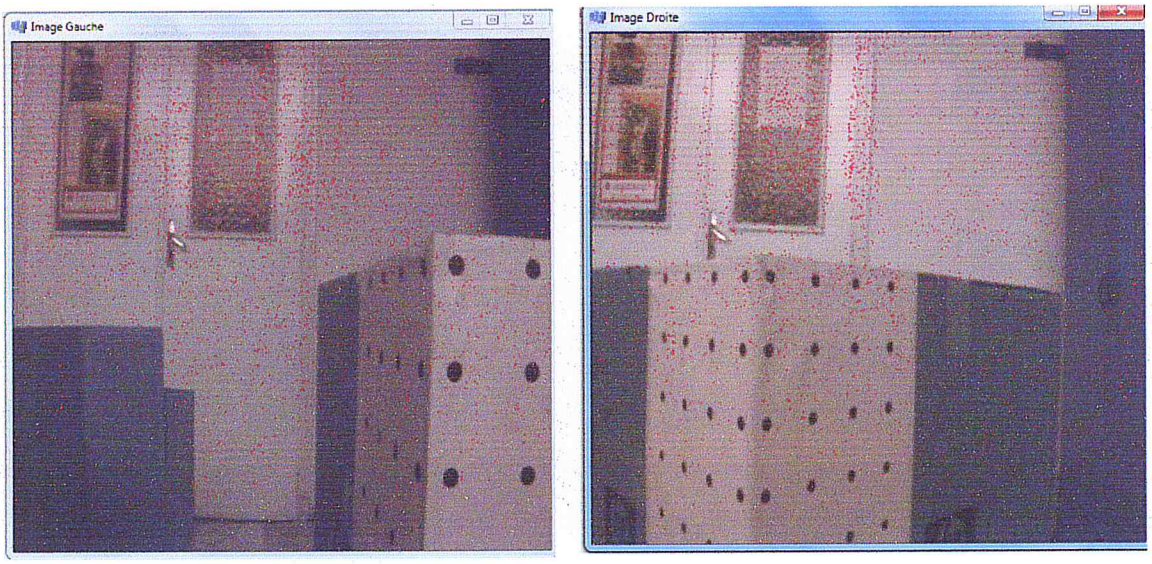
(k)

- **Résultat 13**: Taille de la population : 20000 individus, Nombre d'itération : 50, Temps d'exécution : 1h30.



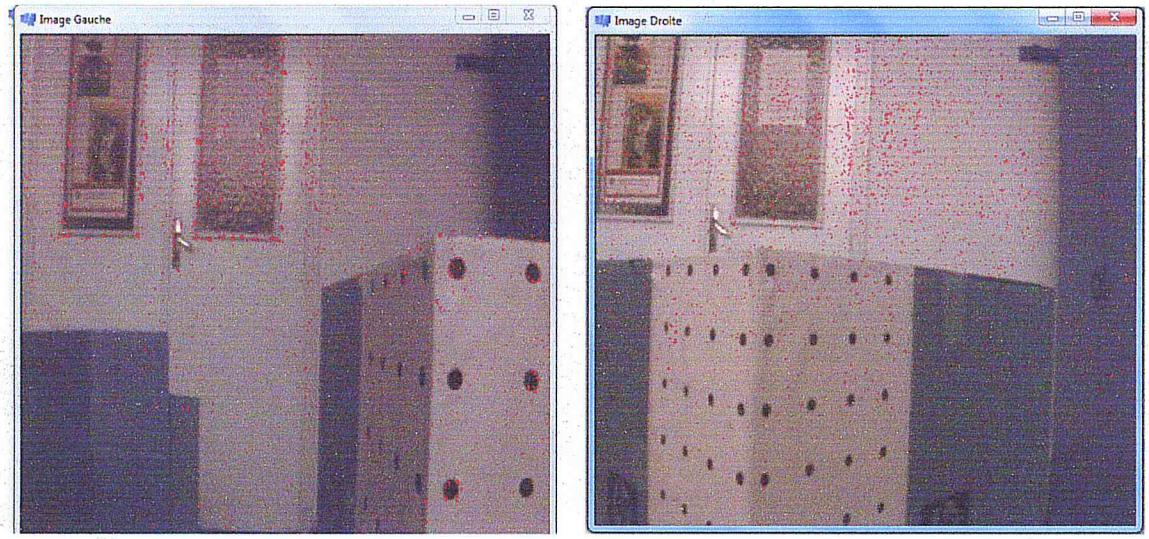
(l)

- **Résultat 13**: Taille de la population : 40000 individus, Nombre d'itération : 50, Temps d'exécution : 2h.



(m)

- **Résultat 14**: Taille de la population : 80000 individus, Nombre d'itération : 10, Temps d'exécution : 3h.

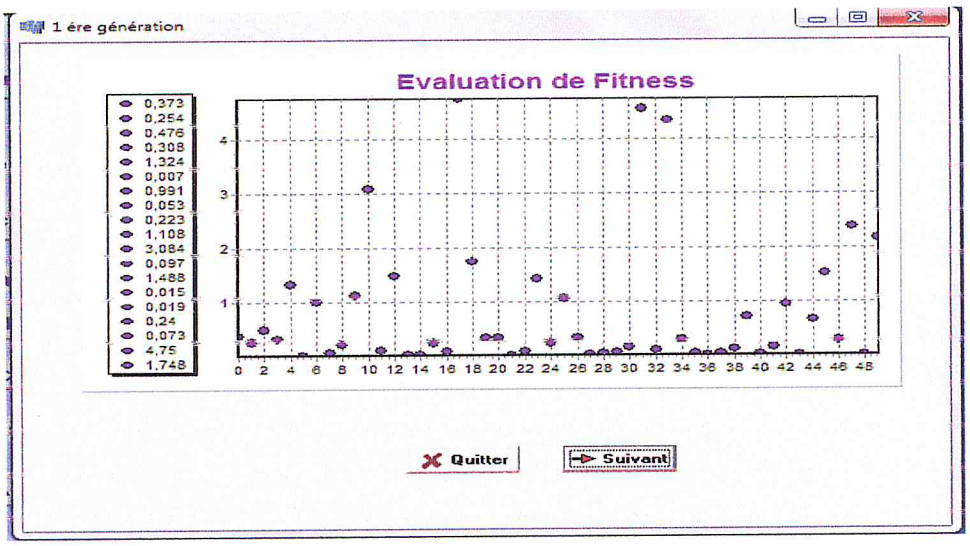


(n)

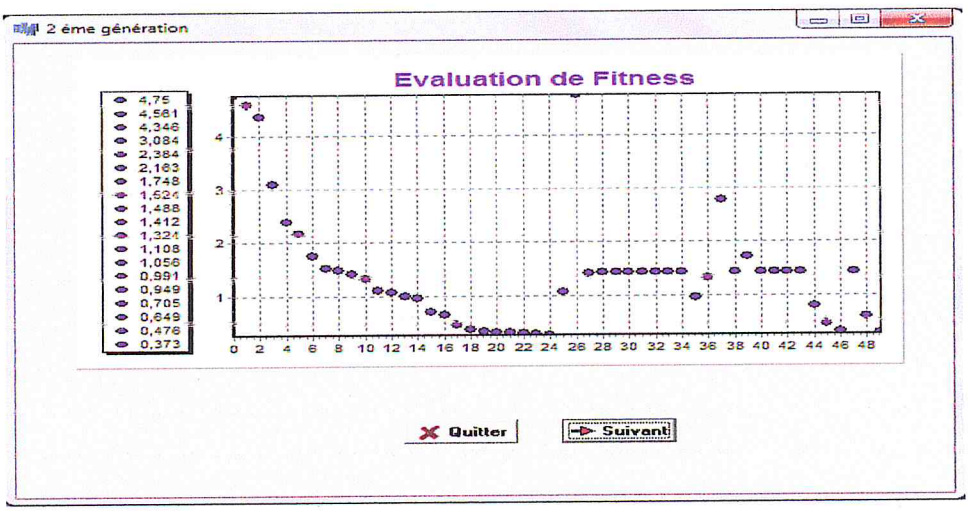
(n)

- **La représentation graphique**

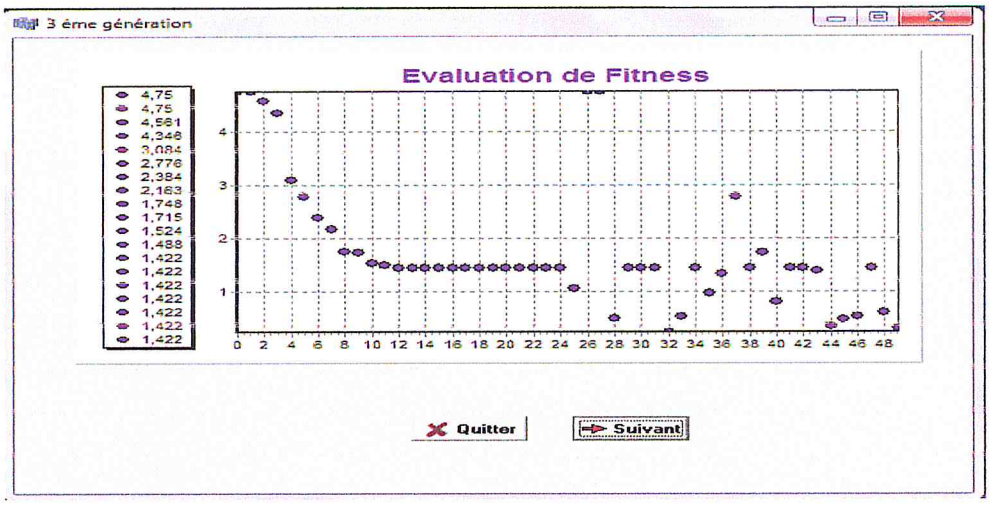
- **Résultat 15**: Une population de taille 50 individu, Nombre d'itération : 5.



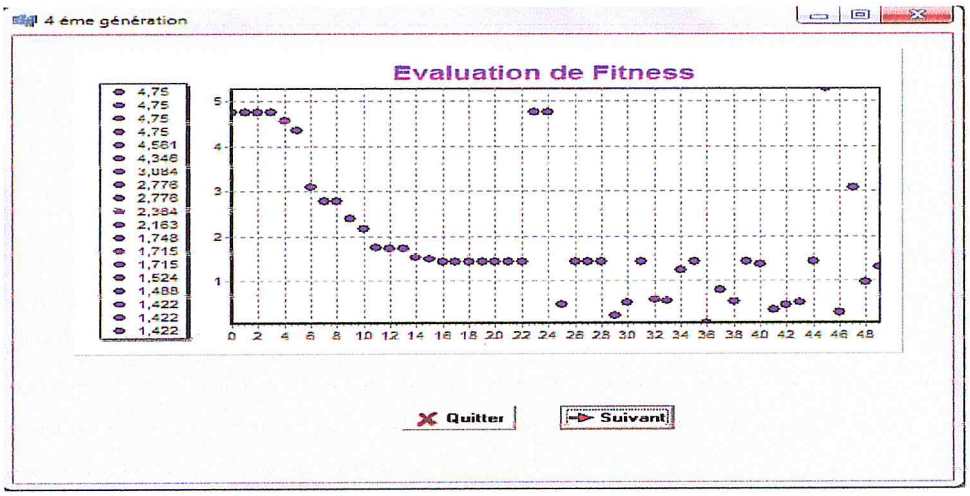
(p) : 1ére génération



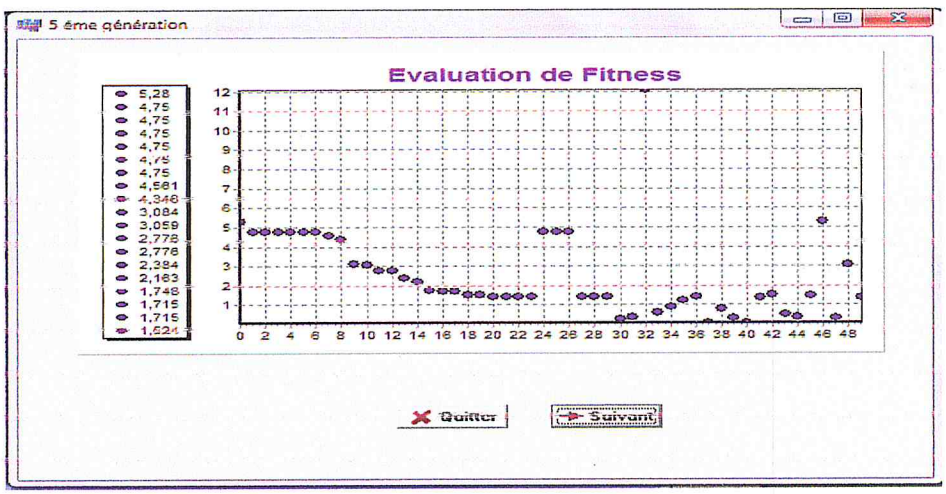
(p) : 2eme génération



(p) : 3éme génération



(p) : 4éme génération



(p) : 5éme génération

IV.3 Conclusion

Nous venons de donner dans ce chapitre une description de notre réalisation pour familiariser l'utilisateur avec les différentes boites de dialogues et boutons ainsi que leurs fonctions. En deuxième lieu, nous avons présenté un ensemble d'essais de la reconstruction 3D par un algorithme d'évolution artificielle.

Notons que nous avons obtenu une convergence de l'algorithme vers des optimums locaux qui ne couvre pas toute l'image pour des raisons liée aux contraintes matérielles sur lesquelles nous avons travaillé que sont : les images de grande taille, un champ de vision qui ne couvre qu'une partie de nos images capturées, ainsi qu'un grand temps de traitement.

Conclusion générale

L'objectif de notre mémoire est l'application d'un algorithme évolutionnaire pour la reconstruction 3D à partir de deux images stéréoscopiques. Pour ce faire, nous avons commencé par donner une description générale des approches évolutionnaires, ainsi que leurs domaines d'application, notamment dans le domaine pétrolier pour la détection des champs de pétrole et pour contrôler les pipe-lines, dans le domaine financier pour la prévention des cours de la bourse, dans le domaine industriel pour la conception des automobiles et l'optimisation des ailes d'avions, dans le domaine militaire pour les empennages de missiles et les manœuvres des avions de combat, dans le domaine médical pour l'analyse d'images médicales, et la recherche de gènes responsables de maladies génétiques et dans la robotique pour le calcul de trajectoire de robots, etc.

Nous avons présenté par la suite le domaine de la vision par ordinateur ainsi que la reconstruction tridimensionnelle qui nécessite une longue chaine de traitement très complexe, en commençant par le calibrage des cameras, l'extraction des points des images, leurs mise en correspondances et en terminant par le calcul des coordonnées 3D de la scène observée.

Récemment, les algorithmes évolutionnaires sont utilisés dans le domaine de la vision par ordinateur pour l'approximation de formes 2D ou pour la reconstruction 3D des points de la scène, les travaux sur ce sujet sont encore peu nombreux.

A cet effet, nous avons implémenté un algorithme évolutionnaire appelé algorithme des mouches dans lequel le problème de la reconstruction 3D est considéré comme étant un problème d'optimisation, où on cherche les meilleurs positionnements des points 3D à partir des images gauche et droite.

Les résultats obtenus confirment le principal inconvénient des algorithmes évolutionnaires qui est leur temps d'exécution. En effet, la convergence de l'algorithme des mouches implémenté nécessite un compromis entre le nombre d'itération et la taille de population. Il est clair que l'augmentation de ces deux paramètres conduit à l'augmentation du temps de calcul.

L'ajustement d'autres paramètres tels que les probabilités de croisement, de mutation et d'immigration, etc, reste un problème majeur pour la convergence de l'algorithme, du fait que ces derniers peuvent changer d'une image à une autre, et d'une application à une autre.

Nous pouvons dire que nous avons obtenu des résultats jugés acceptables, dans lesquels notre objectif a été atteint.

Enfin, nous signalons que la réalisation d'un système de reconstruction 3D par l'algorithme des mouches est un problème très complexe. Cette réalisation nous a permis d'effectuer les taches suivantes :

- La calibration du banc stéréoscopique embarqué sur le robot mobile ATRV2.
- Le calcul du champ de vision commun des caméras gauche et droite.
- L'application de l'opérateur de Sobel pour le calcul du gradient
- L'implémentation de l'algorithme des mouches.

Comme tout sujet de recherche, ce modeste travail reste ouvert à une exploration plus étendue qui viendrait l'enrichir et le rendre plus efficient, et de ce fait, pourrait être le point de départ à d'autres projets de recherche. Comme perspective à ce travail, nous proposons :

- De définir une fonction fitness plus appropriée qui prend en compte la mise en correspondance entre points.
- D'utiliser plusieurs images au lieu de deux.
- D'introduire un mécanisme d'accélération de l'exécution.
- D'appliquer l'algorithme sur des images de petites tailles.
- D'augmenter le champ de vision commun.

Bibliographie

[Alfred ,2004] : Alfred A, «Element de Matlab », Atelier d'outils informatiques pour la physique (InfoPhys), université de Genève, 15 Octobre 2004.

[Amrane, 2007]: leila Amrane, « Méthodes d'appariement d'images basées sur des approches évolutives », mémoire de Magister, INI, Algérie, 2007.

[Baptiste, 2006]: Autin Baptiste, « Les métaheuristiques en optimisation combinatoire », mémoire pour l'obtention d'un examen probatoire en informatique, Conservatoire national des arts et métiers Paris, 9 mai 2006.

[Boulmerka, 2009]: Aissa Boulmerka, « Adaptation des métaheuristiques à l'ordonnancement hors-ligne des tâches temps réel à contraintes strictes en environnement monoprocesseur », mémoire de Magister en Informatique(I.S.I), USTHB ,20 Avril 2009.

[Costanzo, 2007]: Costanzo Andrea, Luong thé Van, Marill Guillaume, Brillault Philippe « Algorithmes évolutionnaires pour la construction automatique de ponts », l'association Evolution Artificielle (Ecole d'été d'Yravals, Espagne), 20 mars 2007.

[Crouzil, 1997]: Alain Crouzil, « Perception du relief et du mouvement par analyse d'une séquence stéréoscopique d'images », thèse de Doctorat, université Paul Sabatier de Toulouse, Septembre 1997.

[Djekoun, 2010]: A.Oualid Djekoun, « Localisation et guidage du robot mobile ATRV2 dans un environnement naturel », thèse de Doctorat, USTHB, 15 décembre 2010.

[Drocourt, 2002]: Cyril Drocourt, « Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels », thèse de Doctorat, université de technologie de Compiègne, février 2002.

[El Zaart, 1996]: A. El Zaart, « Modèle de régularisation pour l'estimation de la disparité », université de Sherbrooke, Québec, Canada, Octobre 1996.

[Eshelman , 1993]: L.J. Eshelman and J.D. Shafer, « Real coded genetic algorithms and interval schemata », In L.D.Whitley, editor, Foundations of Genetic Algorithms II, pages187-202. Morgan Kaufmann, 1993.

[Faugeras, 1986]: O. Faugeras, G. Toscani, « The calibration problem for stereo », Proceedings of Computer Vision and Pattern Recognition, Miami Beach, Florida, USA, juin 1986, pp.15-20.

[Faugeras, 1992]: O. Faugeras, « What can be seen in three dimensions with an uncalibrated stereo », In Proceedings of the 2nd European Conference on Computer Vision (ECCV), mai 1992, pp. 563–578.

[Faugeras, 1995]: O. Faugeras, S. Laveau, L. Robert, G. Csurka, C. Zeller, « 3-D reconstruction of urban scenes from sequences of images », Rapport technique numéro 2572, INRIA, juin 1995.

[Fogel, 1966]: Lawrenc J. Fogel, A.J. Owens, and M.J.Walsh, «Artificial Intelligence through Simulated Evolution », Publisher: John Wiley, New York, 1966.

[Fogel, 1990]: David B. Fogel, Thomas Back, et Zbigniew Michalewicz, éditeurs. «Evolutionary Computation 1: Basic Algorithms and Operators». Institute of Physics Publishing, Bristol, UK, 1990.

[Gagne, 2005] : Christian Gagné, « Algorithme évolutionnaire appliqués à la reconnaissance des formes et à la conception optique », thèse de Doctorat PHD, université Laval, Québec, 2005.

[Gardeux, 2008]: Vincent Gardeux, « Optimisation par Essaim Particulaire, mémoire Adaptative, Recherche Tabou », université de Paris 12, 4 décembre 2008.

[Goldberg, 1994]: D. Goldberg, « Algorithmes génétiques, exploration, optimisation et apprentissage automatique », David E. Goldberg, Editions Addison-Wesley France, SA. Juin 1994.

[Gonzales, 1977]: R.C.Gonzales, « Digital Image Processing », publisher: Addison Wessley, United States, 1977.

[Hadallah, 1997]: M.Hadallah, « Codage des images fixes par une méthode hybride basée sur la QV et les approximations fractales », Projet de fin d'étude, USTHB, 1997.

[Hartley, 1994]: R. Hartley, « Projective reconstruction and invariants from multiple images », Pattern Analysis and Machine Intelligence (PAMI), 16 octobre 1994, pp. 1036–1040.

[Heikkilä, 1997]: J.Heikkilä, O. Silven, «A four-step camera calibration procedure with implicit image correction», IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR97), ETATS-UNIS, 1997.

[Holland, 1975]: J. H. Holland, « Adaptation in Natural and Artificial Systems », University of Michigan Press, 1975.

[Horaud, 1993]: R. Horaud, O. Monga, « Vision par ordinateur, outils fondamentaux », Traité des nouvelles technologies, série informatique, Hermes, 993.

[Kostkovà, 2002]: J.Kostkovà, « Stereoscopic Matching: Problems and Solutions », PHD thesis, Czech Technical University in Prague, Czech (Tchèque), 4 September 2002.

[Koza, 1992] : JohnR.Koza, « Genetic Programming: On the Programming of Computers by Meansof Natural Selection », MIT Press, Cambridge, MA, USA, 1992.

[Koza, 1994]: John R. Koza « Genetic Programming II: Automatic Discovery of Reusable Programs », MIT Press, Cambridge, MA, USA, 1994.

[Louchet, 2000]: Jean Louchet, « Stereo analysis using individual evolution strategy », In the Proceedings of the International Conference on Pattern Recognition, ICPR00, Barcelona, Spain, September 2000.

[Luong, 1992]: Q.T. Luong. « Matrice Fondamentale et Autocalibration en Vision par Ordinateur », thèse de Doctorat, université de Paris-Sud, Orsay, France, Décembre 1992.

[Maturana, 2009]: Jorg Maturana «Contrôle Générique de Paramètres pour les Algorithmes évolutionnaires », thèse de doctorat, Ecole Doctorale STIM, France, juin 2009.

[Moscato, 1989]: P. Moscato. «On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms » Technical Report 826, Caltech Concurrent Computation Program, 1989.

[Mostefaoui, 1996]: G.K.Mostefaoui, « Etude et implémentation des méthodes de segmentation d'image par extraction du contour, basé sur le modèle de Markov », Projet de fin d'étude, Blida 1996.

[Nguyen, 2007]: Nguyen Thanh Tuan, «Coordination globale de comportements coopératifs», Rapport Finale : Travail Personnel Encadré, institut de la francophonie pour l'informatique (IFI), Hanoï, Viêtnam, 2007.

[Rabaud, 2005]: Christophe Rabaud, « une nouvelle approche de mise en correspondance stéréoscopique dense par méthodes possibilistes », thèse de Doctorat, université Montpellier I, juillet 2005.

[Rechenberg, 1973]: I.Rechenberg « Evolutions strategie – Optimierung technisher Systeme nach Prinzipien der biologischen Evolution», Frommann-Holzboog, Stuttgart, Germany, 1973.

[Renata,2005] : Renata M. Aiex , Mauricio G. C. Resende , Panos M. Pardalos ,Gerardo Toraldo , « GRASP with Path Relinking for Three-Index Assignment » , INFORMS Journal on Computing Vol. 17, No. 2, Spring 2005, pp. 224–247, issn 1091-9856, _ eissn 1526-5528 _ 05 _ 1702 _ 0224.

[Roudenko, 2004]: Olga Roudenko, «Application des Algorithmes Evolutionnaires aux Problèmes d'Optimisation Multi-Objectif avec Contraintes », thèse de Doctorat, école Polytechnique, 2004.

[Schoenauer, 2001] : Marc Schoenauer, « Les algorithmes évolutionnaires : état de l'art et enjeux », Algorithms Seminar, INRIA, France, pp.113118, 15 octobre 2001.

[Tabari, 1999]: K.Tabari, S.Tagma, « Compression d'images animées à très faible débit par la géométrie des fractales », Projet de fin d'étude USTHB, 1999.

[**Terki, 2003]**: Terki Amel, « Analyse des performances des algorithmes génétiques utilisant différentes techniques d'évolution de la population», mémoire de Magister, université de Constantine, 2003.

[Toscani, 1987]: G. Toscani, « Système de Calibration et perception du mouvement en vision artificielle », thèse de Doctorat, université Paris Sud, 15 décembre 1987.

[Triggs, 1998]: Bill Triggs, «Autocalibration from planar scenes», Proceedings of the 5th European Conference on Computer Vision(ECCV), pp. 89-105, Freiburg, Germany, 1998.

[Tsai, 1987]: R.Y.Tsai, «A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses», IEEE Int. Journal Robotics and Automation, Vol.3(4), pp.323-344, ETATS-UNIS, 1987.

[Wiley, 1995]: A.G.Wiley, K.W. Wong, «Geometric calibration of zoom lenses for computer vision metrology», Photogrammetric Engineering and Remote Sensing(PE&RS),Vol. 61(1), pp.69-74, ETATS-UNIS, 1995.

[Zeller, 1996]: C. Zeller, « Calibration Projective Affine et Euclidienne en Vision par Ordinateur », thèse de Doctorat, école Polytechnique, 1996.

[Zhang, 2000]: Zhengyou Zhang «A flexible new technique for camera calibration» IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI), Vol.22(11), pp. 1330-1334, ETATS-UNIS, 2000.

Webographie

[1]: http://fr.wikipedia.org/wiki/Algorithme_%C3%A9volutionniste , visité le 15/01/2011

[2]: http://fr.wikipedia.org/wiki/Recherche_locale, visité le 15/01/2011

[3]: http://fr.wikipedia.org/wiki/Recherche_tabou, visité le 20/01/2011

[4]: http://fr.wikipedia.org/wiki/Recuit_simul%C3%A9, visité le 02/02/2011

[5]: http://fr.wikipedia.org/wiki/Algorithme_%C3%A9volutionniste, visité le 03/02/2011

[6]:http://www.enseignement.polytechnique.fr/profs/informatique/Eric.Goubault/poly/cours009.html, visité le 01/03/2011

[7]:http://www.studya.com/formations_metiers/scientifique_interview/specialiste_algorithmes_evolutionnaires.htm, visité le 02/03/2011

[8] : http://eric.univ-lyon2.fr/~ricco/cours/cours/intro_cpp_builder.pdf visité le 10/06/2011

[9] : http://www.ensta-paristech.fr/~ciarlet/Doc-Matlab/Doc-Matlab-Couleur.pdf visité le 10/06/2011

Annexe

La projection à pour but de trouver la position d'un point 3D dans les deux images : image gauche et image droite. Cette opération consiste à calculer les coordonnées 2D (u, v) gauche et (u, v) droite correspondant a un point de coordonnées 3D (x, y, z) en fonction de la matrice de projection perspective M.

Les coordonnées de projection (u, v) sont calculées à l'aide de l'équation ci-dessus :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Afin d'obtenir les coordonnées (u, v), la premier étape consiste à calculer le produit matriciel entre la matrice de projection M et les coordonnées du point 3D, la deuxième étape quant à elle consiste à diviser les valeurs su et sv sur s.

1. Exemple d'application :

On va appliquer la projection sur 7 points 3D parmi les points de la mire de calibrage montrer dans §III.1.1.4 au niveau du chapitre III pour calculer leurs coordonnées 2D sur l'image gauche et l'image droite.

On a les matrices de projection perspectives gauche Mg et droite Md, telle que :

$$Mg = \begin{vmatrix} -289.4185355 & 73.1741906 & 754.9244796 & -5559864.6341 \\ -278.1906589 & 679.1509504 & 4.224512436 & -3723.5804002 \\ -1.0000000 & 0.192004788 & 0.063559176 & 153.14168173 \end{vmatrix} \text{ et}$$

$$Md = \begin{vmatrix} -289.3345357 & 67.38023438 & 740.27454 & -529918.420 \\ -298.6667067 & 643.4704952 & -3.6203302 & 145921.787 \\ -1.000000000 & 0.151022566 & 0.0625703 & 198.023869 \end{vmatrix}$$

Le tableau suivant repesent les coordonnées 3D et leurs coordonnées de projection 2D :

X	Y	Z	Ug	Vg	Ud	Vd
15	0	15	380	442	375	159
65	0	15	339	386	334	125
115	0	15	305	338	299	97
165	0	265	531	267	433	325
115	0	215	526	305	483	238
165	215	0	208	440	205	267
0	115	215	545	468	526	310

2. Vérification de la projection

La projection est considérée comme une étape clé pour d'autre étape comme la reconstruction 3D, donc après chaque étape de projection il est nécessaire de vérifier la sureté des coordonnées 2D calculées si elle donne les même formes de l'image 3D ou non.

Afin de vérifier nos coordonnées de projection gauche et droite nécessaire pour la reconstruction 3D on s'est basé sur une forme géométrique représentée en 3D qui est : le cube.

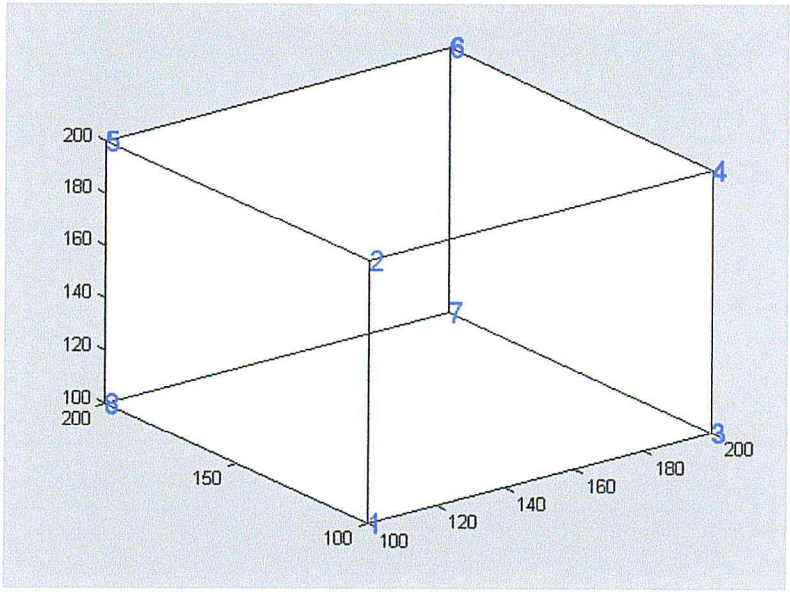
Connaissant les coordonnées 3D (x, y, z) des huit sommets du cube nous pouvons calculer leurs coordonnées de projection (X_L, Y_L) et (X_R, Y_R) correspondantes. Si la représentation des points résultat forme un cube, alors la matrice de projection perspective utilisée est correcte.

Les coordonnées 3D des sommets du cube sont définies comme suit :

N° Sommet	X	Y	Z
1	100	100	100
2	100	100	200
3	200	100	100
4	200	100	200
5	100	200	200
6	200	200	200
7	200	200	100
8	100	200	100

On a appliqué la vérification de la projection sous l'environnement de développement « MATLAB ». Le « MATLAB » est caractérisé par la représentation 3D de différentes formes qui n'est pas accessible sous « C++ Builder ».

Le cube utilisé pour la vérification de la projection est représenté sur la figure ci-dessous :

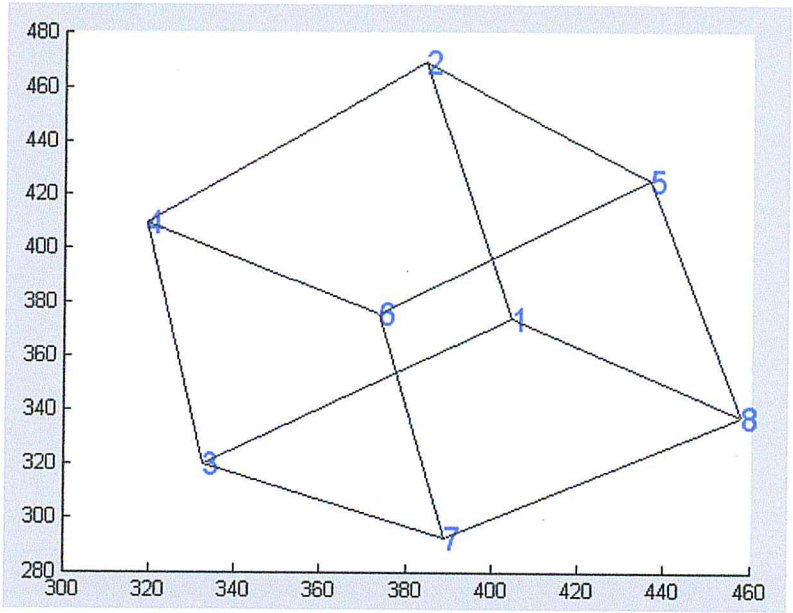


Cube 3D utilisé pour la vérification

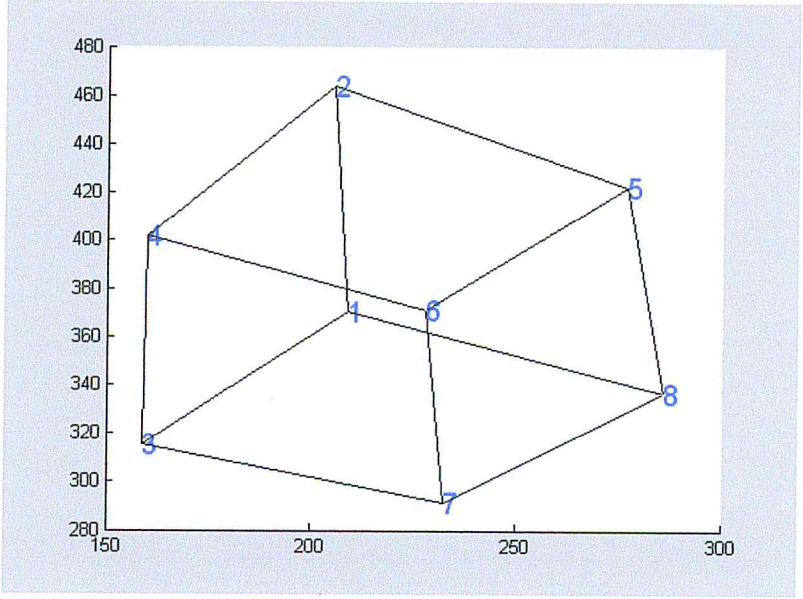
Après la projection, nous obtiendrons les coordonnées 2D gauche et droite définies dans le tableau suivant :

N° Sommet	Ug	Vg	Ud	Vg
1	404	374	209	370
2	384	469	205	464
3	333	320	159	315
4	319	410	160	402
5	437	425	277	422
6	374	375	228	371
7	389	293	232	291
8	458	338	286	336

Les deux figures suivantes représentent les résultats graphiques de la projection respectivement sur l'image gauche et droite :



Cube résultant de la projection gauche



Cube résultant de la projection droite