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

UNIVERSITE SAAD DAHLAB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

MEMOIRE DE FIN D'ETUDES

Pour l'obtention

d'un Diplôme de Master en Informatique

Option : Systèmes Informatiques et Réseaux

THÈME :

Conception et Développement d'un Logiciel de Micro-Gravure par Laser de Motifs 2D par la Stratégie des Contours Parallèles

Réalisé par :

Mr. KENNOUNE Amir

Mr. KHEMICI Ahmed

Soutenu devant :

Mr. BEY Mohamed	CDTA,	Encadreur
Mr. MESSAOUD Slimane	CDTA,	Encadreur
Mr. KAMECHE Abdallah Hicham	USDB,	Promoteur
Mr. BENYAHIA	USDB,	Président
Mr. OULD AISSA	USDB,	Examinateur

2018/2019

Résumé

Ce travail s'inscrit dans le cadre du développement d'un logiciel de micro-gravure par ablation laser au niveau de l'équipe Conception et Fabrication Assistées par Ordinateur «CFAO» de la Division Productique et Robotique «DPR» en collaboration avec l'équipe de Technologique des Système Lasers «TSL» de la Division Milieux Ionisés et Lasers «MIL».

L'objectif de ce travail est l'automatisation et l'optimisation du processus de micro-gravure par laser pour les motifs en 2D quelconque en utilisant la stratégie « Contours Parallèles» à partir de fichier en format « DXF » et générer en sortie un fichier avec les coordonnées (XY) associées à l'état du laser.

Mots clés : Ablation Laser, Fichier DXF, Motifs 2D, Automatisation, Optimisation, Contours Parallèles.

Abstract

This work is part of the development of laser ablation micro-engraving software at the Computer Aided Design and Manufacturing (CAD / CAM) team of the DPR Production and Robotics Division in collaboration with the Laser Technology Team « TSL » of the Division Io Ionized and Lasers Environments « MIL ».

The objective of this work is the automation and optimization of the laser micro-engraving process for any 2D pattern using the « Parallel Contours » strategy from « DXF » file format and generation of a file with the coordinates (XY) associated with the state of the laser.

Keywords: Laser Ablation, DXF File, 2D Patterns, Automation, Optimization, Parallel Contours.

ملخص

يعد هذا العمل جزءًا من تطوير برنامج الحفر الجزئي للتذرية بالليزر على مستوى فريق التصميم والتصنيع بمساعدة الكمبيوتر (CAD / CAM) في قسم الإنتاج والروبوتات في DPR. التعاون مع فريق تكنولوجيا الليزر «TSL» التابع لشعبة Milieu Ionisés و MIL» Lasers».

الهدف من هذا العمل هو أتمتة وتحسين عملية النقش بالليزر الدقيقة لأي نمط ثنائي الأبعاد باستخدام استراتيجية "Parallel Contours" من الملف بتنسيق «DXF». قم بإنشاء ملف بالإحداثيات (XY) المرتبطة بحالة الليزر.

الكلمات المفتاحية: التذرية بالليزر ، ملف DXFالأنماط ثنائية الأبعاد ، التشغيل الآلي ، التحسين ، الخطوط المتوازية

Dédicace :

A MES CHERS
PARENTS

Mon père, l'épaule solide, l'œil attentif compréhensif et la personne la plus digne de mon estime et de mon respect, qui peut être fier et trouve ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit. Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi…….

Ma mère, qui a œuvré pour ma réussite, par son amour, son soutien, tous les sacrifices consentis ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçoit à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude……..

A MES CHERS ET ADORABLES SŒURS

Lina, Hiba : en témoignage de mon affection fraternelle, de ma profonde tendresse et reconnaissance, je vous souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, vous protège et vous garde pour moi.

UN SPECIAL DEDICACE A MON BINOME '' Ahmed khemici''
En témoignage de l'amitié qui nous a uni et des souvenirs de tous les moments que nous avons passé ensemble, je te souhaite un avenir plein de joie, de bonheur et de santé, ainsi que son père pour sa patience et son accompagnement, veuillez trouver dans ce travail l'expression de mon grand respect.

À tous les étudiants de la promotion 2018/2019 Option SIR

À tous ceux qui, par un mot, m'ont donné la force de continuer …..

Amir

Dédicace :

Je dédie ce travail qui n'aura jamais pu voir le jour sans les soutiens indéfectibles et sans limites de mes chers parents qui ne cessent de me donner avec amour le nécessaire pour que je puisse arriver à ce que je suis aujourd'hui.

Je dédie également ce modeste travail :

~ A ma très chère sœur pour m'avoir soutenu et aidé tout au long de ce projet.

~ A Mon frère qui m'a donné la force de ne jamais abandonner, qu'il repose en paix.

~ A la personne qui m'a soutenu toute l'année, mon binôme et frère « Amir », merci.

~ A tous les membres de l'équipe CFAO.

~ A tous mes camarades et équipe du CDTA en particulier : Meriem, Radia, Asma, Fadwa, Hafsa, Kamilia, Latifa et Rym.

~ A tous mes amis Hicham,Rafik,Yacine,amine,sidahmed,sohaib,

~ Enfin, a tous les étudiants de la promotion 2018 / 2019 IL.

Ahmed

Table des matières

Table des figures

Chapitre 1 : Généralités sur le Laser

Chapitre 2 : Etude des Méthodes de Remplissage

Chapitre 3 : Etude Conceptuelle

Chapitre 4 : Implémentation Informatique et Validation

Introduction générale

0.1. Présentation du sujet

Depuis l'invention du Laser en 1960 et jusqu'à ce jour, il est devenu un outil indispensable dans les domaines scientifiques, industriels et militaires. Le traitement des matériaux est l'un des domaines d'utilisation du Laser ou plusieurs procédés sont utilisés (gravure, marquage, découpe, soudage, perçage et rechargement). Comparativement aux procédés plasma, chimique et mécanique, il apporte précision, économie, rapidité et évite le danger des produits chimiques. Dans ce projet, nous nous intéressons au procédé de micro-gravure. Un système de micro-gravure par Laser est composé de plusieurs éléments : source laser pulsée, système de guidage du faisceau Laser, table XY, PC, etc. Dans ce système, le faisceau Laser est statique tandis que l'échantillon à marquer est dynamique et se déplace par le moyen d'une table XY pour réaliser la micro-gravure du motif en 2D. Le motif est conçu dans un logiciel de conception « CAO » et exporté sous plusieurs formats. En raison de la diversité et de la complexité des motifs à marquer, la conception et le développement d'un logiciel est plus que nécessaire pour automatiser et optimiser la micro-gravure.

0.2. Problématique

Le marquage par Laser des motifs avec leurs contours et remplissages avec une précision est une tâche couteuse en termes de temps, de ressources et d'efforts. Elle nécessite du temps pour réaliser le marquage voulu, et les ressources (optiques, liquide de refroidissement, flashs, diodes de pompage, composants électronique, etc.) qui seront consommés pendant le marquage et l'effort qui doit être présent durant ce processus. En outre, le modèle qui va être utilisé dans ce processus est constitué d'un ensemble de formes géométriques pouvant être simples, complexes, linéaires, courbées et aussi dupliquées. Le trajet de marquage sera très long, lent et couteux. Ces inconvénients réduisent la production et augmentent les coûts.

0.3. Objectifs et besoins du travail

Ce travail rentre dans le cadre d'un projet de développement d'un système de micro-gravure par ablation Laser. Il s'insère aussi dans le cadre de la collaboration entre l'équipe Conception et Fabrication Assistées par Ordinateur « CFAO » de la Division Productique et

Robotique « DPR » et l'équipe Technologie des Systèmes Lasers « TSL » de la Division Milieux Ionisés et Lasers « MIL » pour automatiser et optimiser la micro-gravure par laser.

Dans ce projet, nous nous intéressons à la génération automatique du programme de pilotage du Système de micro-gravure par laser de motifs 2D sous le format « DXF » en utilisant la stratégie « Contours Parallèles ». Il s'agit de concevoir, de développer et d'intégrer à l'environnement de production numérique de pièces complexes développé par l'équipe « CFAO » du « CDTA », un module logiciel graphique et interactif pour l'automatisation et l'optimisation du processus de micro-gravure par Laser. Afin de remédier aux problèmes cités ci-dessus, nous avons assigné à notre étude les objectifs suivants :

- Récupérer l'ensemble des entités géométriques.
- Créer les contours et déterminer leurs hiérarchies.
- Générer les contours décalés.
- Simuler visuellement le processus de remplissage.
- Générer le programme de pilotage du système de micro-gravure.

0.4. Structure du mémoire

Le présent mémoire est composé des chapitres suivants :

- Le premier chapitre est réservé à l'étude bibliographique du Laser, de ses types, de ses applications, des différentes techniques de marquage par Laser.

- Le deuxième chapitre est réservé à l'étude des méthodes de remplissage, une étude comparative des techniques de génération des contours parallèles et une étude comparative des méthodes d'optimisation.

- Le troisième chapitre est réservé à la conception de l'application et à la présentation de l'approche de résolution du problème tout en mettant l'accent sur les différentes méthodes utilisées.

- Le quatrième chapitre est réservé à la présentation de l'interface graphique développée et de ses fonctionnalités avec une validation réelle sur un motif très complexe.

Le mémoire se termine par une conclusion générale et des perspectives à donner à ce travail.

PREMIÈRE PARTIE

Étude Bibliographique

Chapitre 1

Généralités sur le Laser

1. **Introduction.**
2. **Définition et historique.**
3. **Principe de fonctionnement du Laser.**
4. **Éléments de la source Laser.**
5. **Caractéristiques du faisceau Laser.**
6. **Principaux types de Lasers.**
7. **Domaines d'application du Laser.**
8. **Techniques de marquage.**
9. **Techniques de balayage par Laser.**
10. **Conclusion.**

1.1. Introduction

Depuis l'invention du Laser en 1960 et jusqu'à ce jour, il continue à couvrir une large gamme d'applications. Ses performances sont toujours plus extraordinaires. Il est omniprésent dans notre vie quotidienne pour les lecteurs de CD, de DVD et de code- barres, etc. Il est devenu un des outils essentiels dans les domaines scientifiques, industriels et militaires. Il peut être utilisé entre autres, pour le stockage de l'information, pour la télémétrie, dans les télécommunications, l'ophtalmologie, la chirurgie et la recherche. Un des principaux domaines d'utilisation du Laser concerne le traitement des matériaux où l'on trouve principalement des procédés tels que la gravure, le marquage, la découpe, le soudage, le perçage et le rechargement. Plusieurs types de matériaux peuvent être traités par Laser tels que le diamant, les tissus, le plastique, le bois, le verre et les métaux.

1.2. Définition et historique

Un Laser (**L**ight **A**mplification by **S**timulated **E**mission of **R**adiation) est un système photonique. Il s'agit d'un appareil qui produit un rayonnement lumineux spatialement et temporellement cohérent basé sur l'effet Laser. Descendant du maser [8], le Laser s'est d'abord appelé maser optique. Le principe de l'émission fut introduit par Albert Einstein [12] en 1917. Alfred Kastler (prix Nobel de physique en 1966) propose un nouveau procédé à savoir le pompage optique. C'est en 1953, que le premier maser (au gaz d'ammoniac) est conçu par J. P. Gordon et H. J. ZEIGER et CH. H. Townes. Plusieurs années plus tard, de nombreux scientifiques vont contribuer à adapter les théories aux longueurs d'ondes du visible. Théodore Maiman va marquer l'histoire du Laser, car c'est en 1960 qu'il obtient pour la première fois une émission Laser au moyen d'un cristal de rubis [23]. Ce n'est que plus tard qu'Ali Javan met au point un Laser au gaz (hélium et néon) en 1961, et en 1966 Peter Sorokin conçoit le premier Laser à liquide. Dans les années 70, les Lasers vont être utilisés dans le domaine industriel. La toute première application fut réalisée en 1965. Elle consiste à percer une pièce de diamant avec un Laser à rubis. C'est en 1974, qu'il y a l'introduction des lecteurs de codes-barres. Depuis son invention dans les années 1950, le Laser ne cesse de se voir attribuer de nouvelles fonctions [4].

1.3. Principe de fonctionnement du Laser

Le principe utilisé dans le Laser est l'émission stimulée, principe découvert par Einstein en 1917, qu'il appellera la théorie quantique du rayonnement. L'émission stimulée est similaire à deux autres phénomènes qui sont l'émission spontanée et l'absorption. Cette théorie quantique est basée sur la fréquence de bord, qui est un modèle de l'atome, dans lequel l'atome qui est composé d'un noyau contenant des nucléons (protons et neutrons) et des électrons qui gravitent autour de ce noyau, sur des orbites définis. Ces électrons se trouvent sur ces orbites, auxquelles sont associées des niveaux d'énergie (E1, E2, etc.) [5].

1.3.1. Absorption

Un atome stable absorbe un photon reçu à une longueur d'onde correspondant à une transition possible. Les électrons de cet atome vont se déplacer vers les orbites supérieures (énergie supérieure), l'atome passe d'un état fondamental E1 à un état instable « excité » d'énergie E2. Après, il retourne à son état stable en libérant un photon. Un photon a disparu de l'onde et celle-ci se trouve atténuée, et en rendant l'énergie qu'elle a acquise [7]. Le processus d'absorption est représenté par la Figure 1.1.

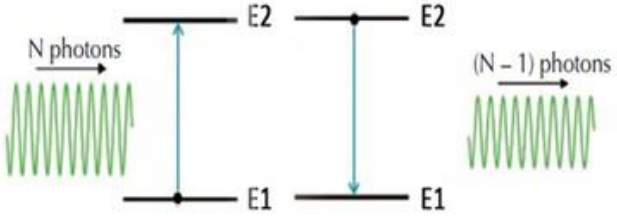


Figure 1.1. Processus d'absorption.

1.3.2. Émission stimulée

C'est le responsable de l'amplification de la lumière dans le Laser. C'est lorsqu'un atome est excité et il doit se désexciter sous l'effet d'un photon, qui le force à émettre un deuxième photon qui a la même quantité d'énergie, et se propagent dans la même direction avec le premier [7]. Le processus d'émission est représenté par la Figure 1.2.

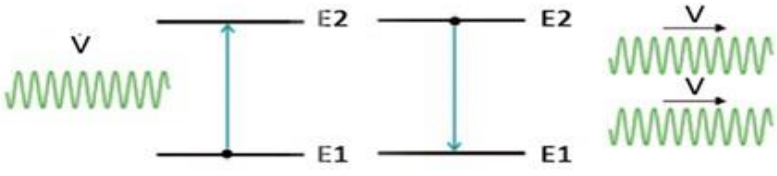


Figure 1.2. Processus d'émission stimulée.

1.3.3. Émission spontanée

Un atome dans un état excité peut se désexciter même en absence d'un photon. Le rayonnement est émis dans une direction aléatoire [7]. Ce processus se produit de façon spontanée. Le processus d'émission est représenté par la Figure 1.3. Dans le cas du Laser, l'émission stimulée est utilisée exclusivement car c'est le fondement même du raisonnement qui a mené au Laser. Pour qu'une émission stimulée puisse avoir lieu, un inversement de population doit exister entre les deux niveaux énergétiques E1 et E2. C'est-à-dire qu'au niveau E2, on a N2 atomes et au niveau E1 on a N1 atomes tel que N2 > N1. Cependant, pour que les atomes passent de l'état E1 à l'état E2, il faut fournir une énergie primaire qui est de nature thermique. Celle-ci peut provenir d'une excitation électrique ou d'un flash optique (dispositif de pompage) (Figure 1.4). La Figure 1.5 représente un système à trois (03) niveaux énergétiques (Laser rubis). Le pompage produit une inversion de population entre le niveau fondamental 1 et le niveau 3, puis, le niveau 3 se dépeuple vers le niveau 2 par une transition non radiative, puis, le passage du niveau 2 vers le niveau 3 donnera un rayonnement par émission stimulée.

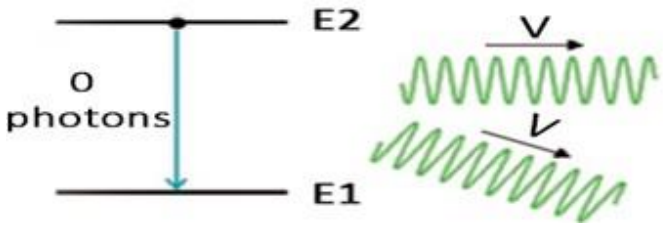


Figure 1.3. Processus d'émission spontanée.

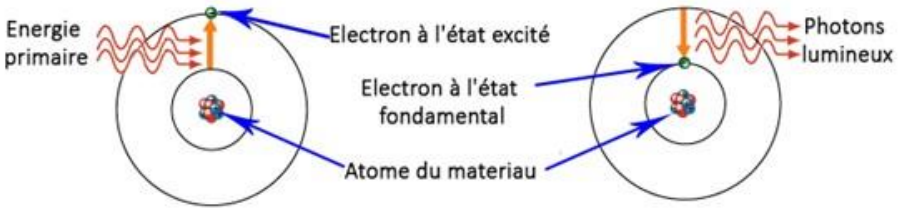


Figure 1.4. Processus d'émission d'un photon lumineux.

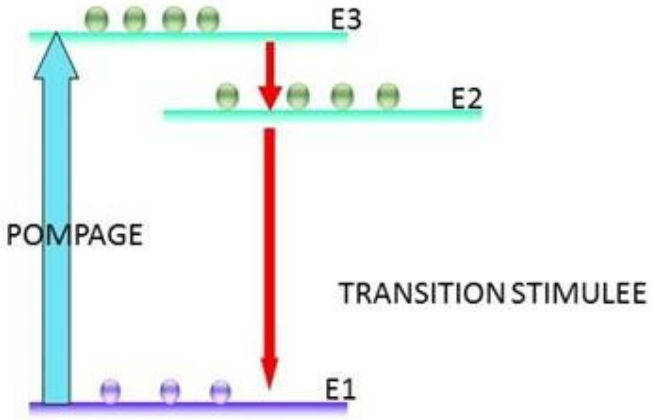


Figure 1.5. Trois niveaux énergétiques du Laser-rubis.

1.4. Éléments de la source Laser

Un oscillateur Laser doit comporter au moins trois (03) composants (Figure 1.6) :

- Cavité résonnante : constituée par le milieu actif placé entre deux miroirs plans perpendiculaires à l'axe optique. Un des deux miroirs est semi-transparent tandis que l'autre miroir est totalement réfléchissant.
- Matériau actif : solide, liquide ou gaz.
- Dispositif de pompage : un flash, une décharge électrique ou une diode Laser.

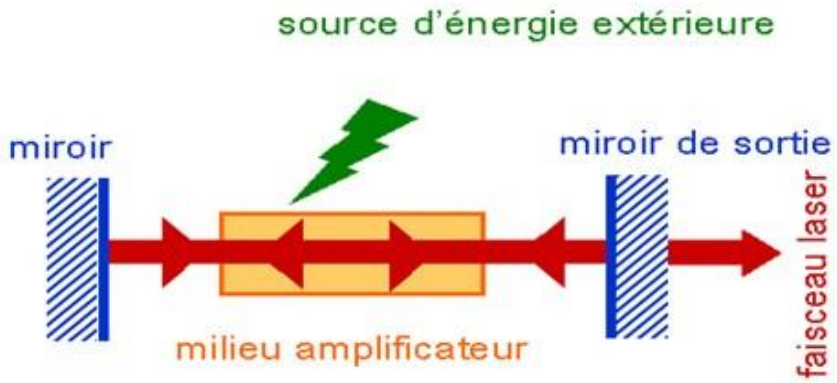
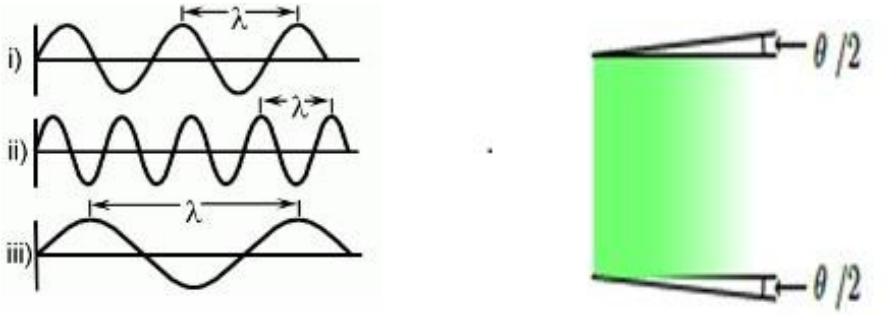


Figure 1.6. Composants d'un Laser.

1.5. Caractéristiques du faisceau Laser

Le faisceau Laser est caractérisé par sa cohérence spatiale et temporelle comparativement à d'autres sources connues dans le domaine optique. Les caractéristiques spécifiques du faisceau Laser sont :

- Sa longueur d'onde d'émission [2] (Figure 1.7.a).
- Sa puissance continue ou pulsée.
- Ses caractéristiques spatiales (diamètre et divergence) (Figure1.7.b).
- La répartition de son énergie dans une section transverse (structure de modes spatiaux).
- La zone focalisée et la stabilité spatiale.
- La puissance moyenne et la stabilité temporelle (pour un faisceau continu).
- La polarisation (linéaire ou circulaire).
- La durée d'impulsion, la fréquence de répétition, l'énergie de l'impulsion et la densité d'énergie (pour un faisceau pulsé).



a. Schéma d'une longueur d'onde. b. Divergence d'un faisceau Laser.

Figure 1.7. Caractéristiques spécifiques du faisceau Laser.

1.6. Principaux types de Lasers

Les principaux types de Lasers utilisés sont :

1.6.1. Laser au dioxyde de carbone (CO_2)

Est un des plus anciens Lasers à gaz. Il émet dans l'infrarouge (9,4-10.6 μm). Il a été développé par Kumar Patel dans les laboratoires BELL [3] en 1964. Les Lasers à dioxyde de carbone sont très efficaces. Leur rapport entre la puissance de pompage et la puissance de sortie atteint 20%. Le schéma explicatif du Laser à CO_2 est représenté par la Figure 1.8.

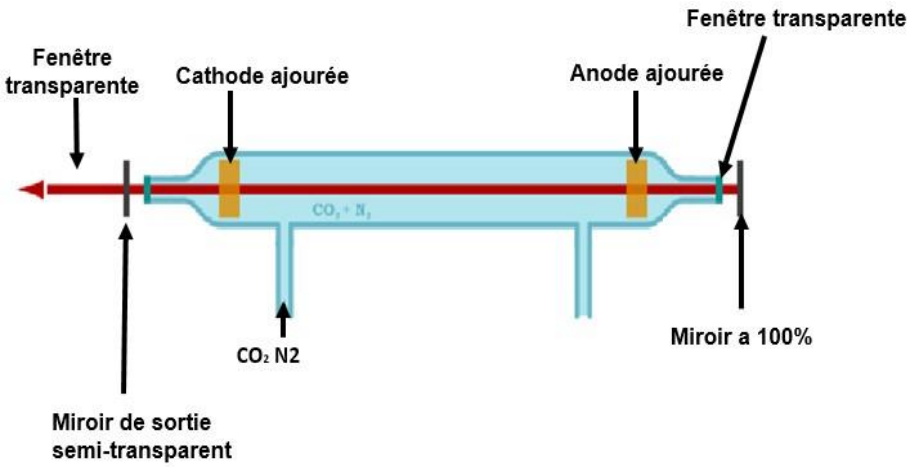


Figure 1.8. Architecture d'un Laser à gaz.

1.6.2. Laser à semi-conducteurs (Diode Laser)

La diode Laser [25] émet un rayonnement monochromatique cohérent. Ce dispositif permet le transport des informations sur de longues distances (système de télécommunication) ou de servir pour le pompage de certains Lasers (Laser à fibre). La diode Laser est aussi un composant essentiel des lecteurs et graveurs de disques. Comme tout Laser, la diode Laser fonctionne à l'aide d'un milieu amplificateur et à l'aide d'un pompage qui est électrique. La Figure 1.9 donne l'architecture de ce type de Laser.

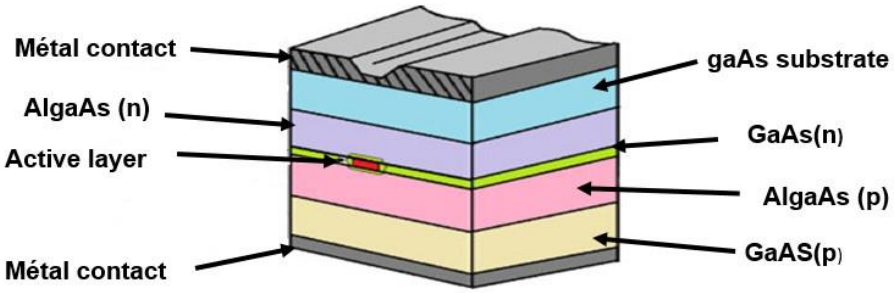


Figure 1.9. Architecture de Laser à diode.

1.6.3. Laser à Néodymium-YAG (Nd-YAG)

Il utilise des verres ou des cristaux comme milieux actifs [18]. Ce Laser est utilisé avec un pompage optique (énergie lumineuse). Leur émission peut être située dans tous les domaines (visible, ultra-violet et infrarouge). Il est le Laser qui fournit la plus grande puissance et il est souvent utilisé à impulsions. La Figure 1.10 donne l'architecture de ce type de Laser.

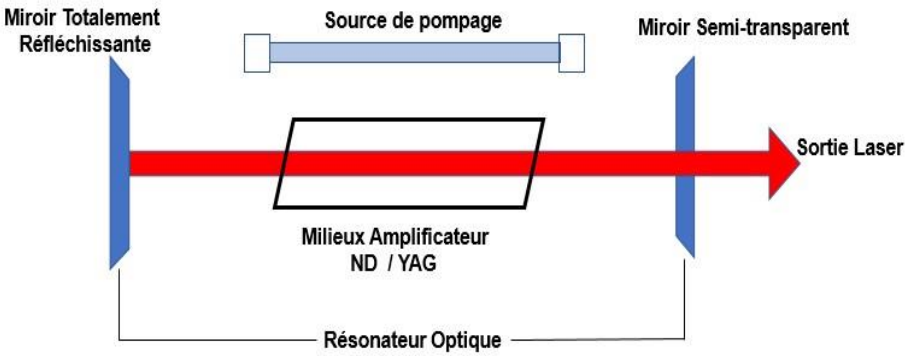


Figure 1.10. Architecture de Laser à Nd-YAG.

1.6.4. Laser à fibre

Dans ce type de Laser, la fibre optique est le milieu amplificateur, dopée avec des ions de terres rares. La longueur d'onde obtenue dépend de l'ion choisi (Samarium 0,6µm, Ytterbium 1,05µm, Erbium 1,55µm, Thulium 1,94µm, Holmium 2,1µm). Le dispositif de pompage est à base des diodes Lasers. Sa longueur d'onde est le dixième de celle d'un Laser à CO_2 (10,6µm) [31]. Son intensité est jusqu'à 100 fois plus élevée qu'un Laser à CO_2 de même puissance moyenne délivrée. Les autres avantages du Laser à fibre : transport du faisceau par fibre optique, compacité et longue durée de vie. La Figure 1.11 donne l'architecture de ce Laser.

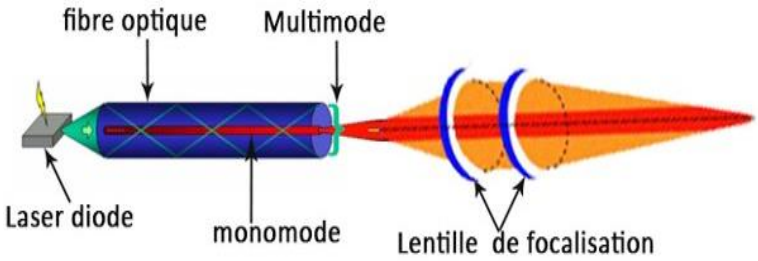


Figure 1.11. Architecture de Laser à fibre.

1.7. Domaines d'application du Laser

1.7.1. En médecine

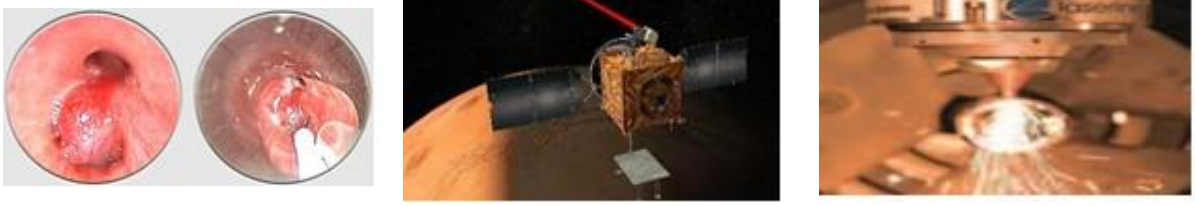
Il est beaucoup solliciter dans le domaine médical car il utilise les propriétés de directivité, de monochromatique et de cohérence du faisceau Laser pour arriver à ses fins (Figure 1.12.a). Il est utilisé pour la destruction des cellules (tumeurs bénignes, effacer des angiomes), en chirurgie, en dermatologie, en ophtalmologie, etc.

1.7.2. En télécommunication

Dans ce domaine, le Laser est transformé en signal et sert les télécommunications à cause de sa forte caractéristique de garder une fiabilité de l'information envoyée en parcourant de longues distances (Figure 1.12.b).

1.7.3. Dans l'industrie

Dans l'industrie, le Laser est utilisé dans les domaines d'usinage tels que le marquage, la gravure, le soudage, la découpe, le perçage et le rechargement (Figure 1.12.c). Ce dernier est utilisé pour remplacer les technologies classiques à cause de sa forte précision, sa flexibilité, sa rapidité et son faible coût.



a. En médecine. b. En télécommunication. c. Dans l'industrie.

Figure 1.12. Domaines d'application du Laser.

1.8. Techniques de marquage

Le marquage Laser est une technique éprouvée d'impression. L'ajout de décors, de logos, de dessins, de data matrix, de codes-barres, de numéros de série et de zones de texte rend cette technique polyvalente incontournable pour le marquage industriel. Le marquage Laser regroupe les procédés suivants : la gravure, l'ablation sélective (ou l'enlèvement), le recuit, la décoloration et le moussage. Ces techniques sont préférentiellement choisies en fonction des matériaux et de l'application de marquage recherchée.

- **Technique du Recuit :** cette technique de marquage Laser permet de travailler sans ablation de matière. Par le passage du faisceau Laser, la matière est localement chauffée à une température proche de son point de fusion. Par effet thermique, cet échauffement entraîne un changement de couleur, produisant le marquage du matériau. Plusieurs colorations sont réalisables en fonction du temps d'exposition, de la puissance et de la cadence du Laser, et de la nature des oxydes composant la matière. Le marquage Laser par recuit concerne uniquement les métaux, en particuliers le titane et les métaux ferreux (acier, inox, etc.) [6].

- **Technique de la décoloration :** le marquage est créé par effet thermique. Le plastique est rapidement chauffé, et s'ensuit un changement de couleur suivant le trajet de déplacement du faisceau Laser. Le marquage peut être foncé ou clair, en fonction de la composition du plastique. Durant la fabrication du plastique, l'ajout de pigments spécifiques peut permettre de déterminer la coloration permise par le marquage Laser. Le marquage Laser par décoloration s'opère uniquement sur les matières plastiques, et la couleur du marquage dépend principalement de leur composition [6].

- **Technique du moussage :** le matériau est fondu par l'apport énergétique fourni par le faisceau Laser. Il en résulte la formation de petites bulles de gaz à la surface du matériau, donnant un aspect de mousse et reflétant la lumière de manière diffuse. Cette technique de marquage Laser permet d'obtenir un bon contraste visuel, principalement sur les plastiques foncées. Comme pour la décoloration, la technique de marquage Laser par moussage s'applique sur les matières plastiques, comme les polycarbonates [6].

- **Technique de la gravure :** elle permet la réalisation d'impressions profondes. Le faisceau Laser, focalisé et de haute puissance, entraine la fusion puis l'évaporation du matériau à traiter, faisant apparaitre un « creux », la gravure, en surface. Selon la nature du matériau traité, il est également possible de générer une coloration ou des reflets améliorant la visibilité du marquage. La gravure Laser s'applique sur tous les matériaux : plastiques, céramiques, bois, verres, métaux, etc. [6].

- **Technique de l'ablation sélective :** le marquage Laser par ablation de matière consiste à éliminer, par vaporisation, une surcouche (peinture, anodisation, vernis, etc.), recouvrant la pièce à traiter. La mise à jour de la portion de pièces nues permet alors de créer un contraste, correspondant au marquage. Comme la technique de la gravure Laser, l'ablation sélective s'applique sur tous les matériaux [6].

1.8.1. Avantages du marquage par Laser

Les avantages du marquage par Laser sont :

- Pas de consommables (encres, étiquettes, solvants, acides, etc.).
- Marquage permanent et de haute définition.
- Technique polyvalente (marquage sur biens industriels comme sur les produits de luxe ou de grande consommation).
- Marquage Laser possible sur zone inaccessible aux techniques conventionnelles.

- Technique de marquage facilement intégrable sur ligne de production [6].

1.9. Techniques de balayage par Laser

Plusieurs techniques de balayage par Laser peuvent être utilisées :

1.9.1. Balayage du faisceau Laser (tête de marquage)

C'est une méthode qui permet d'envoyer un faisceau Laser sur la surface à marquer en utilisant des miroirs galvanométriques [26]. Ces miroirs servent à orienter le faisceau lumineux du Laser sur des endroits précis concernés par le marquage ou la gravure. Dans cette méthode, l'échantillon à marquer est fixe tandis que le faisceau lumineux est mobile et il est orienté par des miroirs (Figure 1.13).

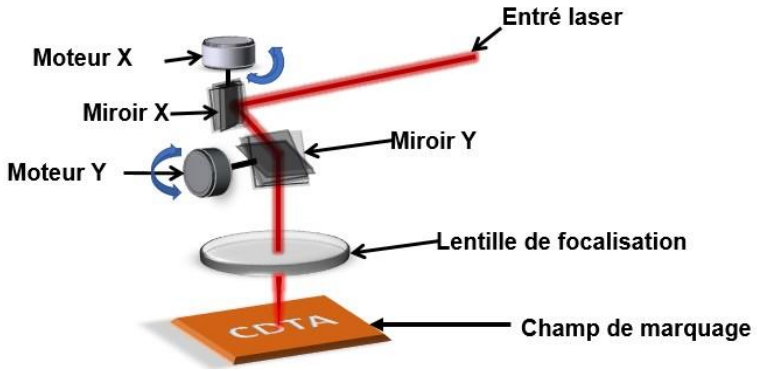


Figure 1.13. Marquage par tête de marquage.

1.9.2. Balayage du faisceau Laser (Plotter XY)

Le système de guidage comprend trois (03) miroirs de renvoi dont deux miroirs mobiles et une lentille de focalisation. La lentille se déplace selon les axes X et Y du Plotter grâce aux moteurs pas à pas contrôlés par un PC à travers un logiciel. Dans ce cas, le faisceau Laser est mobile par contre l'échantillon est statique. Le Plotter XY est utilisé pour la découpe ou la gravure des échantillons de grande surface (Figure 1.14).

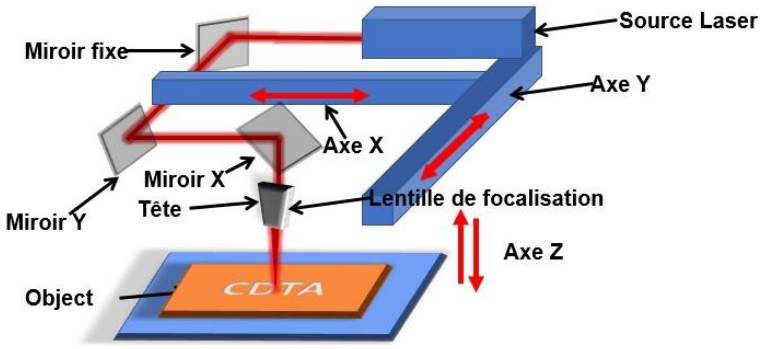


Figure 1.14. Marquage par Ploter XY.

1.9.3. Projection de masque

Cette technique consiste à utiliser un masque métallique qui représente le négatif du motif à imprimer (marquer). Ce masque est éclairé par le faisceau Laser [11]. En général, un faisceau Laser de haute énergie est utilisé où chaque impulsion permettra de marquer le motif. Dans cette méthode, le faisceau Laser et l'objet à marquer sont statiques (Figure 1.15).

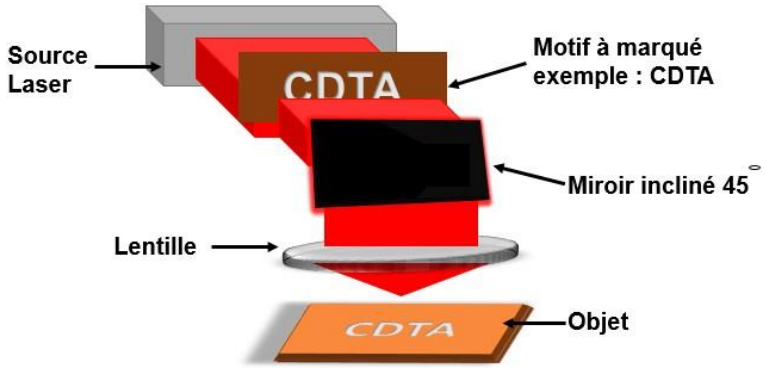


Figure 1.15. Marquage par projection de masque.

1.9.4. Déplacement de l'échantillon (table XY)

Cette technique consiste à envoyer un faisceau Laser depuis une source Laser sur la surface de l'objet à marquer à travers un système de guidage optique. Cette dernière est équipée de deux moteurs pas à pas contrôlés par le logiciel d'exécution de micro gravure pour déplacer l'objet à marquer (fixé sur la table XY) selon les coordonnées X et Y. Le faisceau Laser est activé sur les zones à marquer. Cette technique de marquage est utilisée pour marquer des objets de petite surface (Figure 1.16).

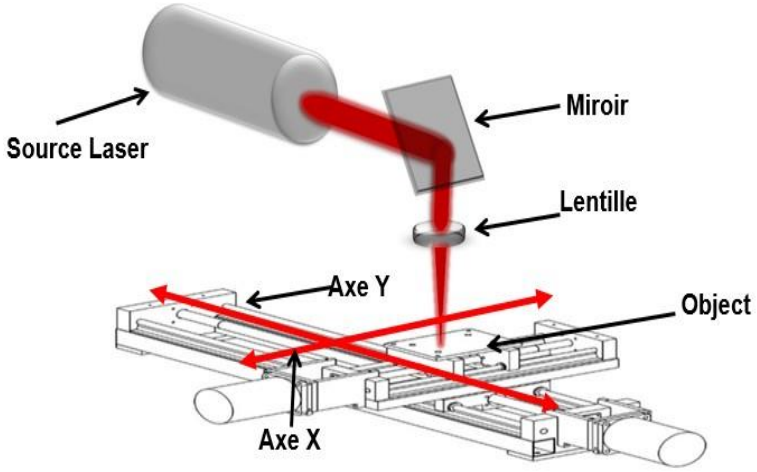


Figure 1.16. Marquage par table XY.

1.9.5. Comparaison des techniques de balayage par Laser

Table 1.1. Comparaison des différentes techniques de balayage

Type de Balayage	Dimension	Avantage
Tête de marquage	Dépend de la lentille utilisée	Vitesse de gravure rapide
Ploter XY	Dépend la longueur des deux axes X et Y	Précision de traçage
Masque de projection	Espace très Limité	Facile à implémenter
Table XY	Dépend de la longueur des deux axes X et Y	Précision de traçage

1.10. Conclusion

Comparativement à d'autres techniques conventionnelles de marquage telles que jet d'encre, sérigraphie et tampographe, le marquage par Laser est une nouvelle technique permettant de marquer de nombreux supports de manière inaltérable et indélébile. Dans ce travaille nous avons choisi la technique de balayage par déplacement de l'échantillon au moyen d'une table XY pour la réalisation de la micro-gravure du motif et garder le faisceau Laser statique. Le motif en 2D à transférer sur l'échantillon est réalisé à l'aide d'un logiciel de conception. Il sera gravé sur l'échantillon par faisceau Laser.

Chapitre 2

Étude des Méthodes de Remplissage

1. **Introduction.**

2. **Motifs en 2 dimensions (2D).**

3. **Méthodes de remplissage des motifs en 2D.**

4. **Techniques de génération des contours parallèles.**

5. **Méthodes d'optimisation.**

6. **Conclusion.**

2.1. Introduction

Les motifs sont utilisés dans plusieurs domaines d'activités tels que l'architecture, l'infographie, l'informatique, etc. à cause de leurs importances de décrire, de modéliser, d'exprimer et de représenter une expression et une information bien précise. Ces motifs se diffèrent selon leurs contenus et leurs usages. Pour réaliser le remplissage de ces motifs différentes méthodes et techniques peuvent être utilisées.

2.2. Motifs en 2 dimensions (2D)

2.2.1. Définition d'un motif en 2D

Un motif 2D est une structure visuelle expressive ou décorative réalisée en 2 dimensions et qui peut se répéter plusieurs fois. Il peut être un dessin ou un ornement sur un support quelconque comme par exemple le tissu à motifs de fleurs. Le motif est composé de formes géométriques telles que polygones, cercles, ellipses, droites, demi-droites, Spline, etc. Il est à signaler que le motif peut être en contours seulement, en remplissage seulement ou bien combinaison des deux modes contours et remplissage [7] (Figure 2.1).

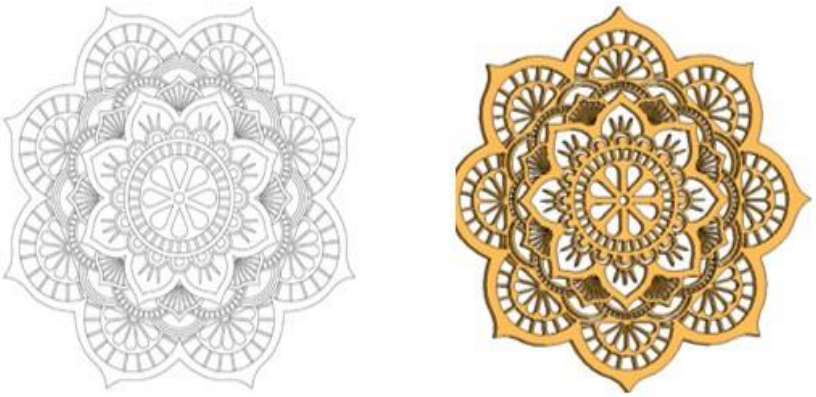


Figure 2.1. Motifs sans remplissage et avec remplissage.

2.2.2. Objectifs de création de motifs

Un motif 2D est utilisé pour représenter et/ou visualiser sur l'outil informatique les structures à créer et à réaliser pour :

- Créer et élaborer de nouveaux motifs.
- Imprimer sur les objets.
- Utiliser la micro-gravure par Laser.

- Décorer (remplir des champs vides et les espaces).

Parmi les usages des motifs, la micro-gravure par Laser qui utilise ces derniers dans les outils informatiques. Ils sont utilisés à travers des programmes qui offrent des fonctionnalités et des outils permettant de créer et de manipuler les motifs selon des formats et des documents spéciaux pour les exporter. Les logiciels de création de motifs sont nombreux : GravoStyle, LaserStyle, AutoCAD, ARCHICAD, Adobe Illustrator, SolidWorks, etc. Ces derniers utilisent, importent, exportent et lisent des fichiers de différents formats tels que DXF, IGS, DWG, etc.

Dans la pratique industrielle, les motifs à graver sont composés de multitudes de formes complexes. Ces motifs peuvent être gravés suivant quatre manières :

- ***1er mode :*** c'est le contournage qui consiste à graver uniquement les courbes limites définissant le motif.
- ***2ème mode :*** c'est le remplissage qui consiste à graver l'intérieur de quelques courbes fermées du motif.
- ***3ème mode :*** c'est la combinaison du contournage et du remplissage.
- ***4ème mode :*** c'est le découpage qui consiste à répéter le contournage sur quelques courbes jusqu'au détachement de la matière comprise entre ces courbes.

D'un point de vue géométrique, les motifs sont de formes quelconques et sont composés d'entités géométriques dont le nombre est variable.

2.2.3. Identification des paramètres des entités géométriques

Les différentes entités géométriques utilisées dans la conception des motifs qui doivent être identifiées à partir du fichier « DXF » (annexe B) sont les suivantes : ligne, polygone, polyline, cercle, arc de cercle, ellipse, arc d'ellipse, courbe quelconque (Spline) (Figure 2.2). Chaque entité géométrique est caractérisée par ses propres paramètres de définition dans le fichier « DXF ».

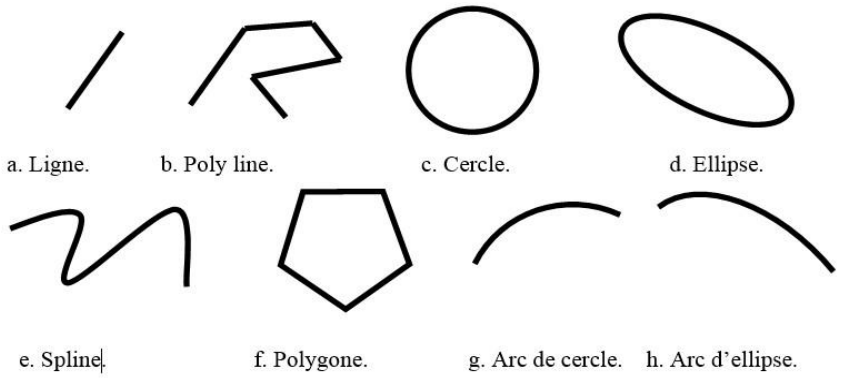


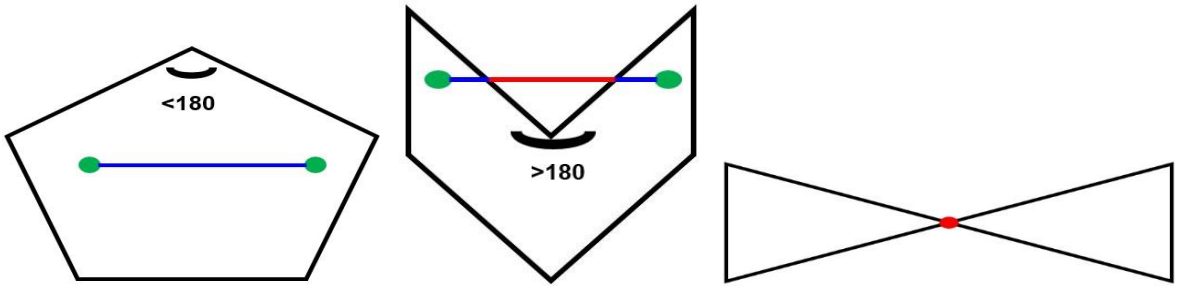
Figure 2.2. Entités géométriques à identifier.

2.3. Méthodes de remplissage des motifs en 2D

2.3.1. Types de polygones

Les différents types de polygones sont les suivants (Figure 2.3) :

- **Polygone convexe :** un polygone est dit convexe si (Figure 2.3.a) :
  - Tous les points du segment de droite reliant deux points du polygone sont également à l'intérieur du polygone [28].
  - Tous les angles intérieurs sont inférieurs à 180 degrés.
  - Peu importe où on dessine une ligne qui traverse la forme, elle passera toujours par seulement deux des lignes ou polygones constituant la forme...
- **Polygone concave :** un polygone est dit concave si (Figure 2.3.b) :
  - Quelques points du segment de ligne reliant deux points du polygone ne sont pas à l'intérieur du polygone [27].
  - Au moins un angle intérieur est supérieur à 180 degrés [27].
  - Si vous essayez la même chose avec une forme concave, il peut passer à travers plus de deux des lignes.
- **Polygone avec auto-intersection :** un polygone est dit auto-intersection si au moins deux de ses côtés sont sécants, c'est-à-dire si au moins deux de ses côtés non consécutifs se coupent (Figure 2.3.c).



a. Polygone convexe. b. Polygone concave. c. Auto intersection.

Figure 2.3. Types de polygones.

2.3.2. Test Intérieur – Extérieur

Méthode du nombre de points d'intersection :

La méthode la plus simple pour remplir un polygone est d'implémenter un test permettant de déterminer, pour chaque pixel (x, y) de l'écran, s'il est intérieur au polygone, et dans ce cas, à l'allumer [24]. Pour déterminer si un point est intérieur ou extérieur au polygone, on considère la demi-droite horizontale **Δ** partant du point (x, y) vers les abscisses positives. On compte le nombre de points d'intersection de **Δ** avec chaque arête non horizontale privée de son sommet d'ordonnée minimale [30] (pour éviter de compter deux intersections lorsque cette demi-droite passe par un sommet du polygone).

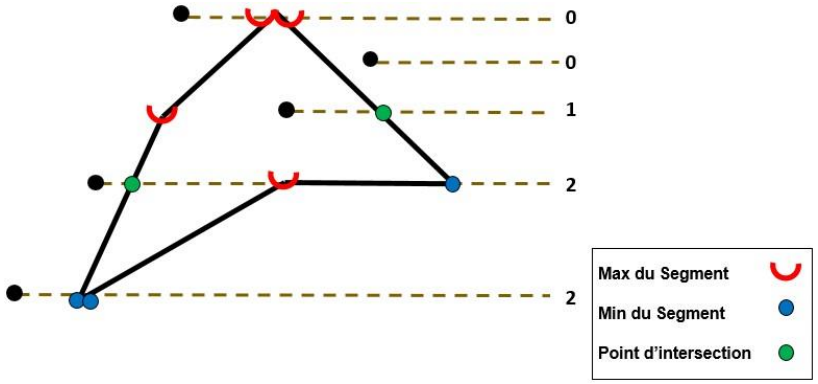


Figure 2.4. Méthode du nombre de points d'intersection.

On peut vérifier qu'au bout de ce traitement :

- Si le nombre total d'intersection compté est impaire, alors (x,y) est intérieur au polygone, ou bien sur le bord du polygone.
- Si le nombre total d'intersection compté est pair, alors (x,y) est extérieur au polygone ou bien sur le polygone.

Méthode du nombre d'enroulement :

Au lieu de simplement compter le nombre d'intersections, chaque arête traversée est étant donné un numéro de direction [30]. La valeur du numéro de direction est :

- La direction est dans le sens des aiguilles d'une montre.
- La direction est dans le sens antihoraire.

Le point est à l'intérieur du polygone si la somme des nombres de direction est non nulle, sinon il est en dehors (Figure 2.5).

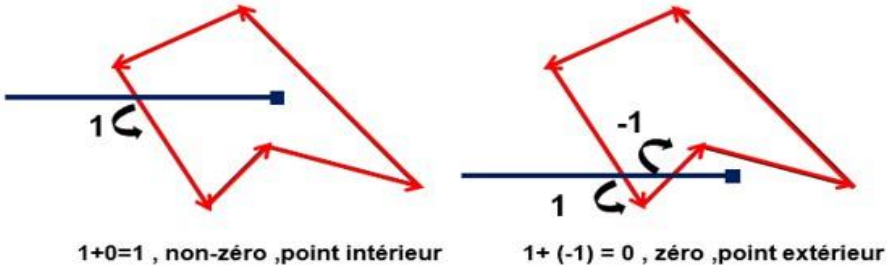


Figure 2.5. Méthode du nombre d'enroulement.

2.3.3. Méthodes de remplissage

Remplir un polygone est le processus de coloration de chaque pixel entrant dans la région du polygone. Pour cela, plusieurs techniques peuvent être utilisées.

- **Méthode de remplissage des limites :** aussi connue sous le nom de « méthode de remplissage des semences » [19]. Ses étapes sont les suivantes :
  - Tracer les limites du polygone.
  - Remplir le point de départ.
  - Un point de départ, c'est-à-dire qu'un point intérieur arbitraire est pris comme point de départ (utilisé une des méthodes de test Intérieur – Extérieur).
  - Testez les pixels voisins pour déterminer s'ils correspondent au pixel limite.
  - Si non, peignez-les avec la couleur de remplissage et testez les pixels voisins (stocker les voisins dans la pile).
  - Continuez jusqu'à ce que tous les pixels aient été testés.

Une pile considérable est utilisée pour stocker des informations sur les pixels. Deux modes sont possibles (Figure 2.6) :

1. Remplissage de semences 4-connecté.

2. Remplissage de semences 8-connecté.

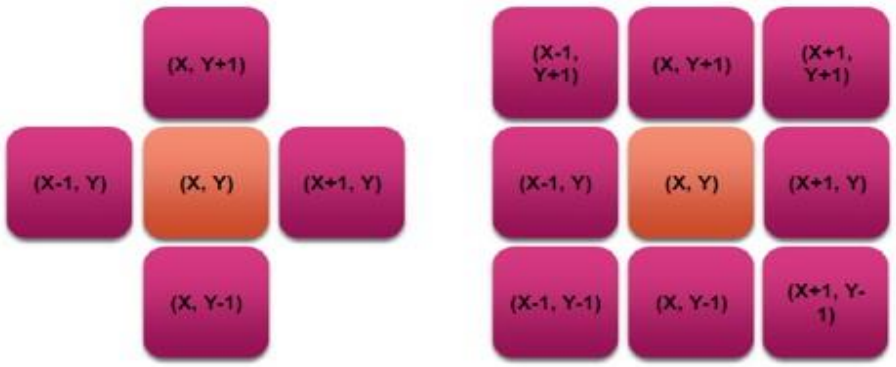


Figure 2.6. Remplissage de semences 4-connecté, semences 8-connecté.

- **Méthode de remplissage par inondation :** le concept de base est semblable au remplissage de limites. Remplissez un polygone en commençant par un point de départ connu pour être à l'intérieur du polygone et définir les pixels voisins jusqu'à ce que nous rencontrions les pixels de la limite. Le polygone est rempli comme si vous versiez de l'eau dans un seau vide [19].

Remplissage limite : agrandissez et remplissez la région jusqu'à ce que vous atteigniez la couleur limite (Figure 2.7).

Figure 2.7. Remplissage limite.

Remplissage inondé: agrandir et remplir une région pendant que vous trouvez la couleur intérieure (Figure 2.8).

Figure 2.8. Remplissage inondé.

- **Méthode de remplissage par ligne de balayage :** cette méthode fonctionne en croisant la ligne de balayage avec les arêtes du polygone [7] et en remplissant le polygone entre des paires d'intersections. Les étapes suivantes décrivent l'exécution de cette Méthode [19] :

  - **Étape 1 :** trouvez Ymin et Ymax à partir du polygone donné.

- **Étape 2 :** la ligne de balayage intersecte les segments du polygone de Ymin à Ymax. Nommez chaque point d'intersection du polygone. Selon la Figure 2.9, ils sont nommés p0, p1, p2, p3.
- **Étape 3 :** triez les points d'intersection dans l'ordre croissant de la coordonnée X, à savoir (p0, p1), (p1, p2) et (p2, p3).
- **Étape 4 :** remplissez toutes les paires de coordonnées qui se trouvent à l'intérieur des polygones et ignorez les paires alternatives.

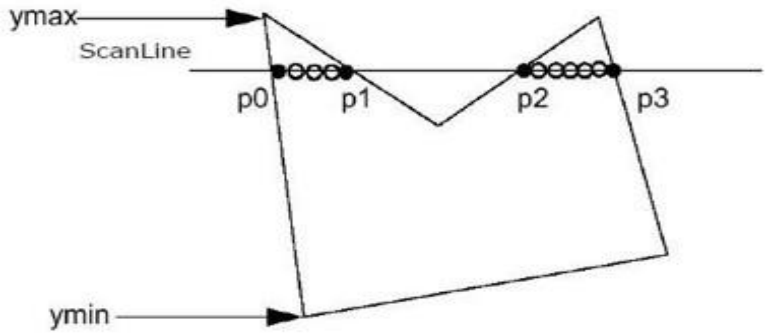


Figure 2.9. Remplissage par ligne de balayage.

2.4. Techniques de génération des contours parallèles

La génération d'offset (contour décalé) est une tâche importante et l'une des opérations géométriques les plus difficiles et les plus complexes. Le décalage des contours de certains polygones peut générer une auto-intersection. Ce dernier va produire deux types de boucles à savoir valides et invalides. Mais éliminer toutes les boucles invalides est très difficile, car la détection d'intersection est le temps consommé, et l'enlèvement de boucles est sujet à des erreurs numériques.

2.4.1. Travaux connexes

Diagrammes de Voronoï

En mathématiques, un diagramme de Voronoï [16] est un pavage (découpage) du plan en cellules (régions adjacentes) à partir d'un ensemble discret de points appelés « germes ». Chaque cellule enferme un seul germe et forme l'ensemble des points du plan plus proches de ce germe que d'aucun autre. La cellule représente en quelque sorte la « zone d'influence » du germe. Le découpage est aussi appelé décomposition de Voronoï, partition de Voronoï ou tessellation de Dirichlet. De manière plus générale, il représente une décomposition d'un espace métrique en cellules (régions adjacentes), déterminée par les distances à un ensemble

discret d'objets de l'espace. Persson a été le premier à appliquer les diagrammes de Voronoï à la génération de chemins de coupe pour l'usinage de poches de forme arbitraire. Chou et Cohen utilisent des diagrammes de Voronoï pour créer des chemins parallèles au contour [15]. Kim donne un algorithme simple mais efficace basé sur un diagramme de Voronoï pour calculer un décalage ajusté d'un seul polygone composé des segments de droites et d'arcs [20].

Algorithme de décalage par paire

Cet algorithme pour les courbes de séquences ponctuelles 2D fermées (courbe PS). Un élément clé de l'algorithme est que toutes les boucles non valides locales sont supprimées de la courbe PS d'entrée avant de construire une courbe de décalage brute, en invoquant un test de détection d'interférence par paire (PWID). Dans le test PWID, chaque paire de segments de décalage élémentaires est testée pour les interférences, puis les segments en interférence sont supprimés successivement [14]. L'algorithme proposé a été mis en œuvre et testé avec diverses courbes PS (Figure 2.10).

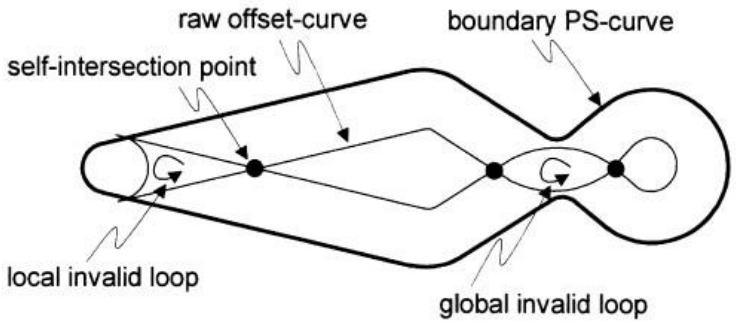


Figure 2.10. Entités géométriques de base dans l'approche par paire.

Décalage du polygone par calcul du nombre d'enroulements

Cet algorithme construit un intermédiaire « Courbe de décalage brute » comme entrée dans les routines de Tessellator dans la bibliothèque d'utilitaires OpenGL (GLU), qui calcule l'enroulement (numéro pour chaque région connectée). Par construction, la boucle invalide des courbes liées à la courbe de décalage brute avec un nombre d'enroulement non positif peut donc être enlevée. Cet algorithme prend O((n + k) log n) temps et O (n + k) espace, où n est le nombre de sommets dans le polygone en entrée et k est le nombre des auto-intersections dans la courbe de décalage brute [13] (Figure 2.11).

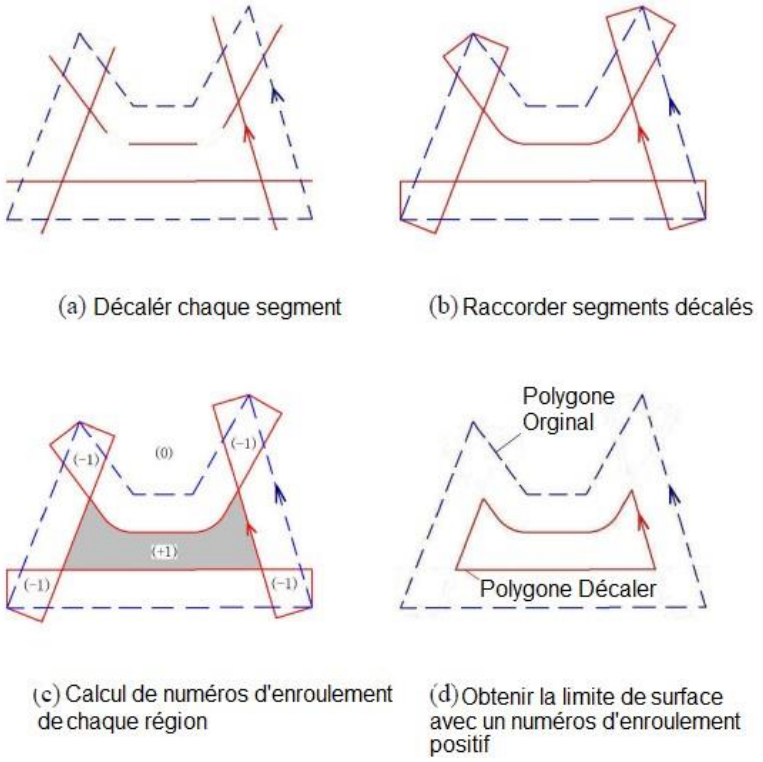


Figure 2.11. Construction du polygone de décalage intérieur.

Algorithme de décalage pour les courbes polylignes

L'algorithme de décalage comprend trois étapes. Premièrement, les décalages de tous les segments des courbes polylignes sont calculés. Ensuite, tous les décalages sont rognés ou joints pour créer des courbes polylignes appelées courbes de décalage non rognées. Enfin, un algorithme de découpage est appliqué aux courbes de décalage non limitées pour obtenir les résultats finaux [22] (Figure 2.12).

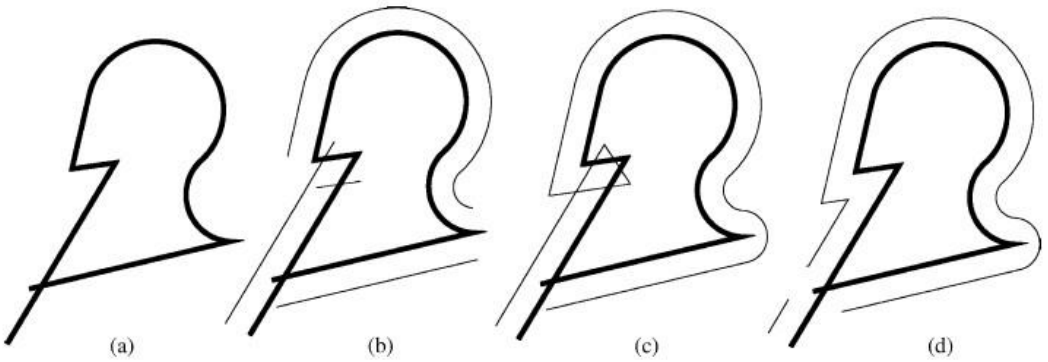


Figure 2.12. Courbes offset.

- (a) La polyligne originale qui comprend quatre segments de ligne et deux arcs.

- (b) Les courbes de décalage de tous les segments de la courbe d'origine.
- (c) La courbe de décalage non limité.
- (d) Les résultats finaux de décalage après application de l'algorithme de découpage.

Algorithme de génération de chemins de contours pour le prototypage robotique

La compensation de polygone est une opération fondamentale en infographie. Les opérations joue un rôle important dans la CAO / FAO et la planification des chemins des robots. Le décalage des contours de certains polygones peut générer une auto-intersection, venir à être la boucle de soi. La Figure 2.13.a montre un exemple d'auto-intersection du contour décalé. Cet algorithme utilise la modification de l'algorithme de balayage en ligne de Bentley-Ottmann [10] en utilisent les chaines monotones. Cet algorithme prend O ((n + k) log m) temps, où k est le nombre de points d'intersection et m est le nombre de chaînes monotones. Le résultat exact est le contour après élimination des auto-intersections non valides [17] (Figure 2.13.b).

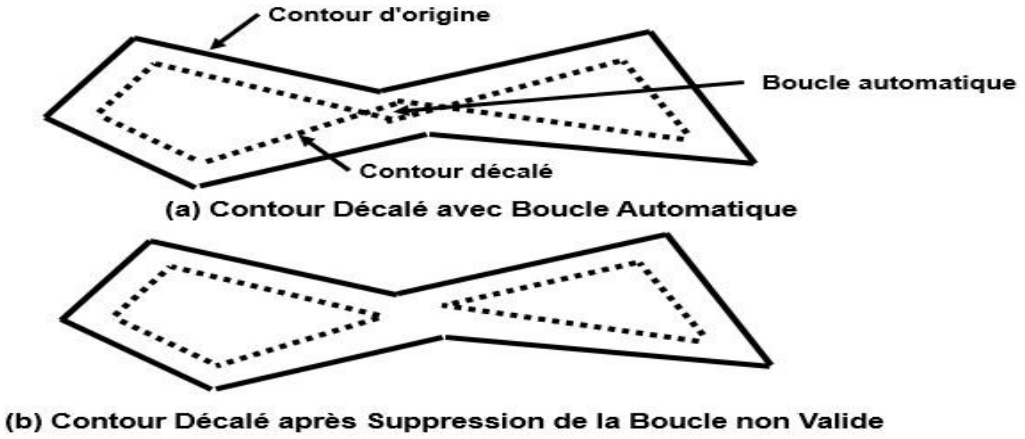


Figure 2.13. Boucle invalide et sa suppression.

2.4.2. Comparaison des différentes techniques

Le tableau suivant donne une comparaison entre les différentes techniques de génération des contours parallèles.

Table 2.1. Comparaison entre les techniques de génération des contours parallèles.

Techniques	Avantages	Inconvénients
Méthode du diagramme de Voronoï	Efficace et robuste	Lente, implémentation robuste est extrêmement complexe à mettre en œuvre
Algorithme de décalage par paire	Rapide avec une complexité linéaire O(n)	Applicable uniquement aux polygones sans trous
Algorithme de décalage de polygone par calcul du nombre d'enroulements	Rapide, précis et extrêmement facile à mettre en œuvre et sa complexité O ((n+k) log n) où n est le nombre de sommets en entrée et k est le nombre d'auto-intersections	Complexe est difficile à implémenter
Algorithme de décalage pour les courbes polylignes	Efficace et traite toutes les formes géométriques	Difficile à implémenter si la courbe polyligne a plus d'une boucle. Beaucoup de boucles d'auto-intersection rendent plus difficile à déterminer la direction de décalage de la courbe
Algorithme de génération de chemins de contours pour le prototypage robotique	Efficace, réduit la complexité du temps O ((n + k) log m), facile à mettre en œuvre	Ne traite pas les cas de dégénérescences, tels que chevauchements, des contacts et des intersections multiples

2.5. Méthodes d'optimisation

L'optimisation est un domaine très grand et très large qui consiste à utiliser des méthodes et des algorithmes pour réduire et pour diminuer la réalisation d'une ou de plusieurs tâches. Les méthodes utilisées sont de deux sortes.

2.5.1. Méthodes déterministes (analytiques)

Elles sont basées sur le calcul et la détermination d'un plus court chemin exact en appliquant un ensemble de traitements. Les techniques qui peuvent être utilisées sont Djikstra, Bellman et Dantzig.

Méthode de Djikstra

C'est un algorithme proposé en 1959 par Edsger Wybe Dijkstra qui permet de déterminer le plus court chemin entre un sommet source et les autres sommets d'un graphe connexe pondéré (orienté ou non). Il se base sur :

- Si le plus court chemin reliant le premier sommet au dernier sommet passe par les sommets S1, S2, . . ., Sk, alors les différentes étapes sont aussi les plus courts chemins reliant le premier sommet aux différents sommets S1, S2, . . ., Sk.
- Le chemin cherché est construit de proche en proche en choisissant à chaque itération de l'exécution de cette méthode un sommet Si du graphe parmi ceux qui n'ont pas encore été traités telle que la longueur connue provisoirement du plus court chemin allant du premier sommet à Si soit la plus courte possible.

L'algorithme de Dijkstra trouve son utilité dans le calcul des itinéraires routiers. Le poids des arcs pouvant être la distance, le temps estimé, réseau téléphonique, routage IP, . . . etc.

2.5.2. Méthodes heuristiques

Elles sont des méthodes inspirées de la nature ou d'une créature existante. Elles donnent des résultats approximatifs selon des taux de précision spécifiés. Parmi ces derniers, la méthode du recuit simulé.

Recuit simulé

Cette méthode a été réalisée en 1953 dans le but de simuler l'évolution du processus du recuit. Cette méthode est beaucoup utilisée dans le domaine de la métallurgie. Le recuit simulé est une adaptation de l'algorithme Metropolis-Hastings Monte Carlo.

- **Inspiration :** le recuit simulé est inspiré du processus de recuit en métallurgie. Dans ce processus naturel, un matériau est chauffé et lentement refroidi sous contrôle pour augmenter la taille des cristaux dans le matériau et réduire leurs défauts. Il a pour effet d'améliorer la résistance et la durabilité du matériau. La chaleur augmente l'énergie des atomes leur permettant de se déplacer librement, et le programme de refroidissement lent permet une nouvelle configuration à basse énergie pour qu'elle puisse être découverte et exploitée.

- **Stratégie :** l'objectif du traitement de l'information de la technique est de localiser la configuration du coût minimum dans l'espace de recherche. Le plan d'action des algorithmes

consiste à ré-échantillonner de manière probabiliste l'espace problématique où l'acceptation de nouveaux échantillons dans l'échantillon actuellement retenu est gérée par une fonction probabiliste qui devient plus perspicace du coût des échantillons qu'elle accepte pendant le temps d'exécution de l'algorithme. Cette décision probabiliste est basée sur l'algorithme de Metropolis-Hastings pour simuler des échantillons d'un système thermodynamique. Cette méthode est appliquée dans de nombreux domaines dans lesquels il est question de résoudre des problèmes d'optimisation difficiles (conception des circuits électroniques, etc.).

2.5.3. Comparaison entre Djikstra et recuit simulé

Le tableau suivant (Table 2.2) donne une comparaison entre la méthode déterministe Djikstra et la méthode heuristique recuit simulé. Il ressort du tableau que le recuit simulé est moins complexe que Djikstra en particulier lors du traitement d'un grand nombre de données.

Table 2.2. Comparaison des deux méthodes d'optimisation.

	Djikstra	**Recuit Simulé**
Complexité	$O(X^2)$	O(Ln(X))
Plus court chemin entre	Sommet source et autres sommets	Plus court chemin global
Type de graphe	Non orienté	Non orienté

2.6. Conclusion

Dans ce chapitre, nous avons mené une étude bibliographique sure les différentes techniques de remplissage, Par la suite, nous avons étudié les différentes techniques de génération des contours parallèles .Dans la dernière partie de ce chapitre, nous avons présenté deux méthodes d'optimisation. Le chapitre suivant sera consacré à la conception informatique.

DEUXIÈME PARTIE

Étude Conceptuelle

Chapitre 3

Etude Conceptuelle

1. **Introduction.**

2. **Architecture générale du système.**

3. **Conception du système.**

4. **Conclusion.**

3.1. Introduction

Dans le chapitre précédent, nous avons mené une étude bibliographique sur le Laser et les techniques de remplissage. Ce chapitre sera consacré à la présentation de la problématique, des solutions envisagées ainsi qu'à la partie conception informatique.

3.1.1. Problématique

L'utilisation de la micro-gravure par Laser a connu un grand essor dans divers domaines tels que la décoration, la fabrication de médicaments, le découpage et bien-sûr dans le domaine industriel. Dans la pratique industrielle, les motifs à graver sont composés de multitudes de formes complexes. Ces motifs peuvent être gravés suivant quatre manières :

- **Contournage :** il consiste à graver uniquement les courbes limites définissant le motif.
- **Remplissage :** il consiste à graver l'intérieur de quelques courbes fermées du motif.
- **Remplissage et contournage :** c'est la combinaison du contournage et du remplissage.
- **Découpage :** il consiste à répéter le contournage sur des courbes jusqu'au détachement de la matière comprise entre ces courbes.

D'un point de vue temps de micro-gravure, le temps nécessaire pour réaliser une micro-gravure, pour les quatre modes, augmente avec la complexité géométrique et le nombre des entités utilisées dans la conception du motif, la précision exigée, la vitesse de déplacement des parties mobiles et du diamètre du Laser.

3.1.2. Objectifs du travail

Au vu des problèmes posés lors de la micro-gravure par Laser, l'objectif principal de ce travail est le développement d'un logiciel graphique et interactif pour le pilotage automatique d'un système de micro-gravure par Laser en utilisant la stratégie des « Contours Parallèles » pour optimiser le trajet de micro-gravure à partir du fichier « DXF » d'un motif 2D quelconque.

3.2. Architecture générale du système

L'architecture générale du système est donnée par la Figure 3.1. Cet environnement est composé de plusieurs tâches qui doivent être traitées jusqu'au trajet de micro-gravure. Ces dernières vont être développées dans les paragraphes suivants.

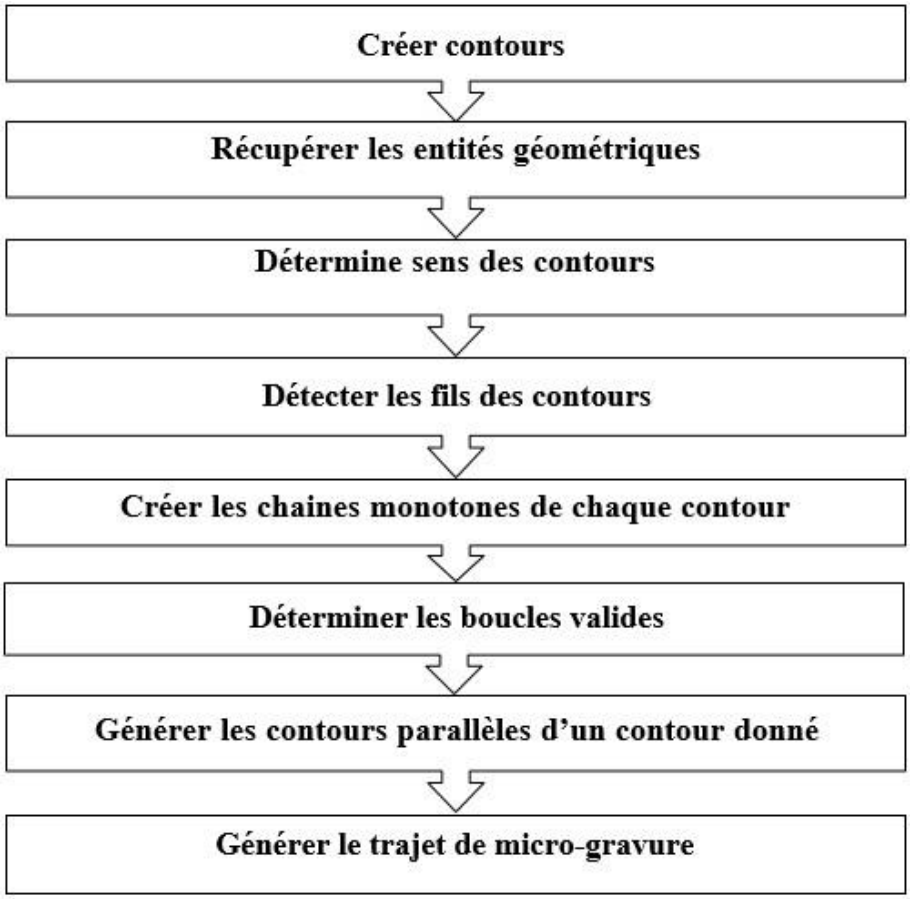


Figure 3.1. Architecture générale du système.

3.2.1 Récupération des entités géométriques.

Après la lecture de fichier DXF voire (l'annexe A) du motif et la création des contours, notre travail consiste à récupérer ces entités géométriques. Cette tâche passe par deux étapes :

- **Traitement des entités géométriques :** pendent la récupération des entités géométriques, un ensemble de traitements doivent être réalisés :
  1. Récupérer uniquement les contours fermés. Un contour est défini par un ensemble de segments. Il est fermé si les coordonnée X, Y du sommet de début du premier segment sont identiques aux coordonnées du sommet de fin du dernier segment du contour (Figure 3.2.b). Dans l'autre cas, il est ouvert (Figure 3.2.a).

a. Contour ouvert. b. Contour fermé.

Figure 3.2. Types de contours.

2. Éliminer les segments qui ont les sommets de début et de fin superposés.
3. Remplacer tous les segments colinéaires par un seul segment. Deux segments sont colinéaires si le produit vectoriel de leurs vecteurs directeurs est égal à 0. Ce traitement a pour but de minimiser le calcul par rapport au nombre de segments. Le pseudo Algorithme 1 suivant montre ce traitement.

Algorithm 1: Enlever les points colinéaires

Data: Contour

Result: Contour optimisé

```
1  int i=0;
2  while i < nombre_segment − 1 do
3      if segments_contour[i].verifier_colinier (segments_contour [i + 1]) then
4          segments_contour[i].setsommet fin (segments_contour [i +1].getsommet_fin ());
5          Supprimer le segment [i + 1];
6          nombre_segment = nombre_segment − 1;
7      else
8          i + +;
9      end
10 end
11 /*Vérifier le premier segment avec le dernier s'ils sont colinéaires */
12 if segments_contour [0].verifier_colinier (segments_contour[nombre_segment − 1]) then
13     Raccorder le début de premier segment avec le début de dernier segment ;
14     Supprimer le dernier segment de contour ;
15     nombre_segment = nombre_segment − 1;
16 end
```

➢ **Calcul des paramètres des entités géométriques**

Après le traitement des entités géométriques, un ensemble de calculs sont menés :

1. Boucler sur tous les contours du motif et calculer pour chaque segment sa longueur :

$$|AB| = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2} \quad (3.1)$$

2. Calculer pour chaque segment ses limites (Figure 3.3).

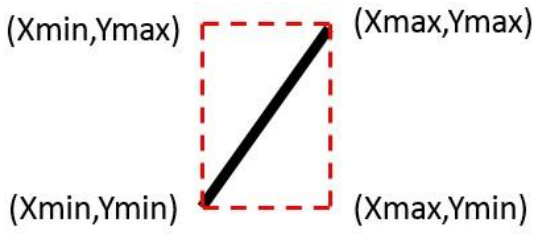


Figure 3.3. Limites d'un segment.

3. Calculer pour chaque contour ses limites (Xmin, Xmax, Ymin, Ymax) (Figure 3.4).

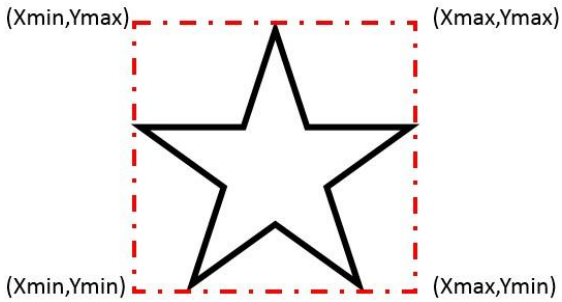


Figure 3.4. Limites d'un contour.

4. Calculer pour chaque segment son vecteur normal unitaire (Figure 3.5).

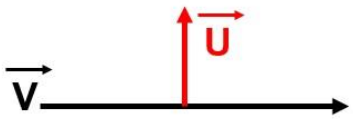


Figure 3.5. Vecteur normal unitaire d'un segment.

La normale d'un segment est calculée en trois étapes :

- **Calculer la norme du segment :** la norme d'un segment est la distance entre ses points extrêmes (longueur).

$$Norme = \sqrt{(V_X)^2 + (V_Y)^2} \quad (3.2)$$

- **Normaliser le vecteur directeur du segment :** normaliser un vecteur, c'est mettre sa norme à 1 en utilisant l'équation suivante :

$$Normalisé = \left(\frac{V_X}{norme}, \frac{V_Y}{norme}\right) \quad (3.3)$$

- (c) **Calculer sa normale :** la normale d'un vecteur est un vecteur qui lui est perpendiculaire. Le vecteur perpendiculaire au vecteur unitaire est donné par :

$$Normale = \left(-Normalisé_y, Normalisé_x\right) = \left(-\frac{V_Y}{norme}, \frac{V_X}{norme}\right) \quad (3.4)$$

3.2.2. Orientation des contours

Dans cette tâche, nous avons déterminé et changé l'orientation des contours suivant la démarche montré par La Figure 3.6.

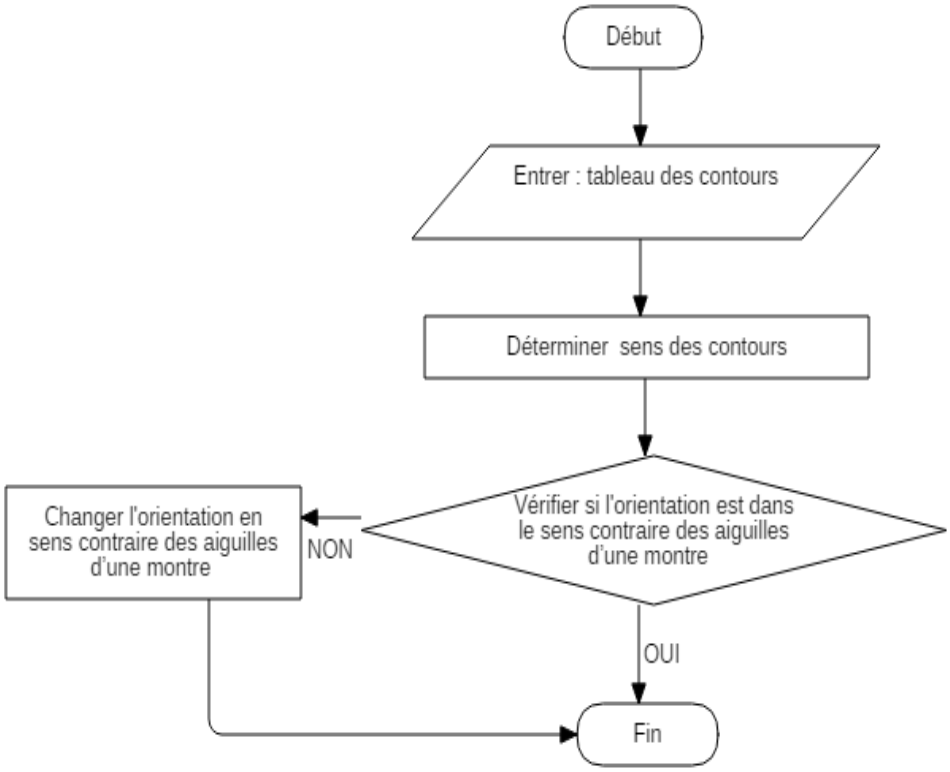
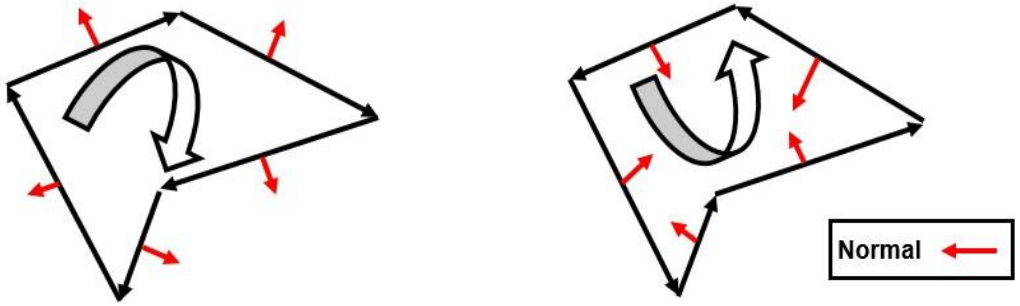


Figure 3.6. Organigramme sens d'un contour.

Un contour peut être dans le sens des aiguilles d'une montre (Clockwise - CW -) ou dans le sens contraire des aiguilles d'une montre (Counter-Clockwise - CCW -) (Figure 3.7).



a. Sens des aiguilles d'une montre. b. Sens contraire des aiguilles d'une montre.

Figure 3.7. Sens d'un contour.

Détermination du sens d'un contour : le pseudo Algorithme 2 suivant montre comment nous avons déterminé le sens du contour.

Algorithm 2: Déterminer sens du contour

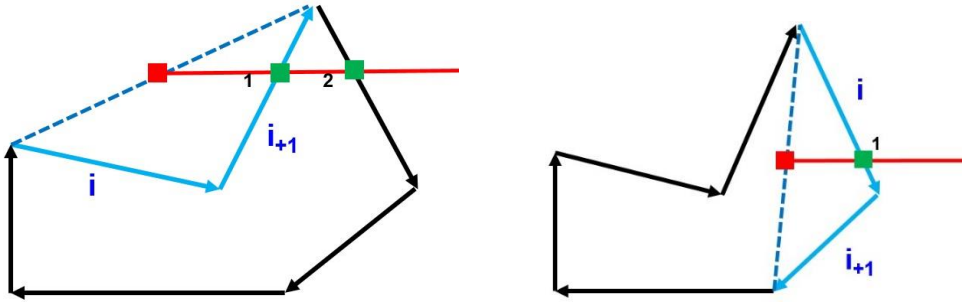
Data: Contour

Result: Sens de contour

```
Sommet_DXF milieu ;
double produit, epsilon = 1e-6;
Ansistring sens;
for i ← 0 to nombre_segment do
    calculé vecteur directeur de segments contour[i] ;
    calculé vecteur directeur de segments contour [i+1] ;
    if segments_contour[i] et segments contour [i+1] sont colinéaire then
        Continue ;
    end
    milieu = calculé le milieu entre le sommet début du segment contour[i] et le sommet fin du
      segment contour [i+1] ;
    if Appel au Teste d'intériorité (point milieu) ==false then
        Continue ;
    end
    produit = calculé le produit vectoriel entre segments contour[i] et segments contour [i+1] ;
    if produit < epsilon then
        return "Sens  des aiguilles d'une montre " ;
    else
        return "Sens contraire des aiguilles d'une montre " ;
    end
end
```

Test d'intériorité

Pour tester si le point milieu est à l'intérieur du contour, nous avons utilisé la méthode du nombre de points d'intersection présentée dans le chapitre précédent. La Figure 3.8 montre les deux cas pouvant être rencontrés.



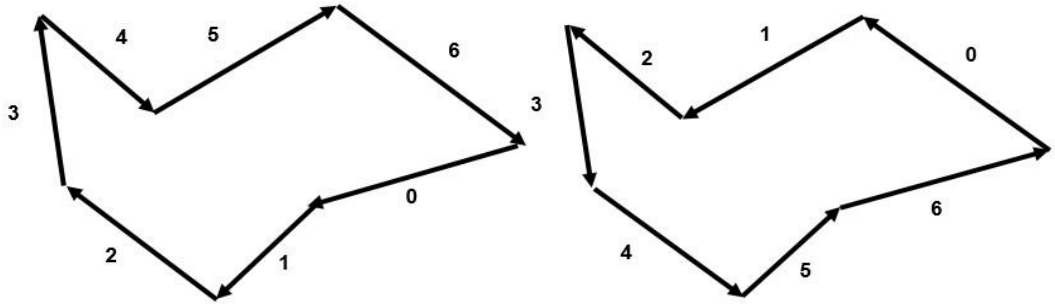
a. Cas convexe. b. Cas concave.

Figure 3.8. Test d'intériorité du point milieu.

Changer l'orientation des contours

Après la détermination du sens des contours, nous avons changé le sens des aiguilles d'une montre des contours au sens contraire des aiguilles d'une montre. Pour faire ce changement, nous avons procédé de la manière suivante :

- Permuté pour chaque segment d'un contour le sommet de début avec le sommet de fin.
- Permuté pour chaque contour ses segments (Figure 3.9).



a. Contour d'origine. b. Contour permuté.

Figure 3.9. Changement du sens d'un contour.

3.2.3. Hiérarchie des contours

Dans cette tâche, nous avons déterminé les contours contenus (contours fils) dans un contour donné (contour père). Cette tâche se déroule en deux étapes :

- **Détecter tous les fils d'un contour donné :** la Figure3.10 montre un contour et ses fils.

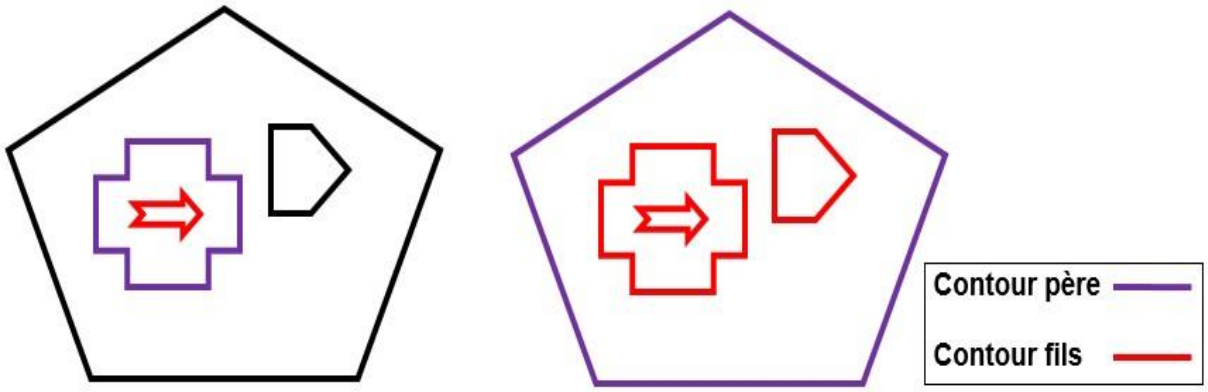


Figure 3.10. Hiérarchie des contours.

Pour vérifier si un contour est à l'intérieur d'un contour, nous avons utilisé la méthode du nombre de points d'intersection pour tester l'intériorité du sommet de début de chaque segment d'un contour donnée. L'Algorithme 3 montre les étapes suivies.

Algorithm 3: Algorithmes récupérer les fils de contour sélectionner

```
Data: le nombre de contour et indice contour
Result: retourner un tableau des fils de contour
int tableaux_fils ;
for i ← 0 to nombre_contour do
    if indice contour==indice de contour[i] then
        continue ;
    end
    if le contour[i] n'est pas le père du contour [indice contour] then
        if contour[i] est à l'intérieur du contour [indice contour] then
            Remplir le tableau fils par l'indice i ;
        end
    end
end
return tableaux_ fils ;
```

- **Détecter les fils du premier degré :** après avoir trouvé tous les fils du contour donné, il suffit de supprimer les enfants de chaque fils (Figure 3.11).

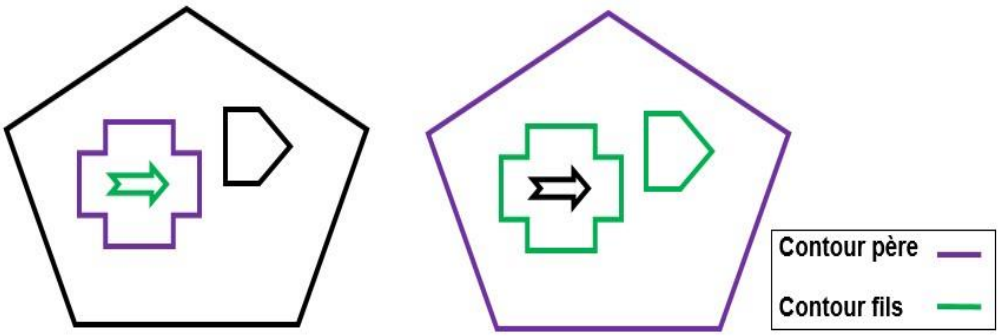


Figure 3.11. Contours fils du premier degré.

3.2.4. Détection des points d'intersection

Dans cette tâche, nous avons détecté les points d'intersection pour chaque contour en utilisant les chaines monotones (Figure 3.12).

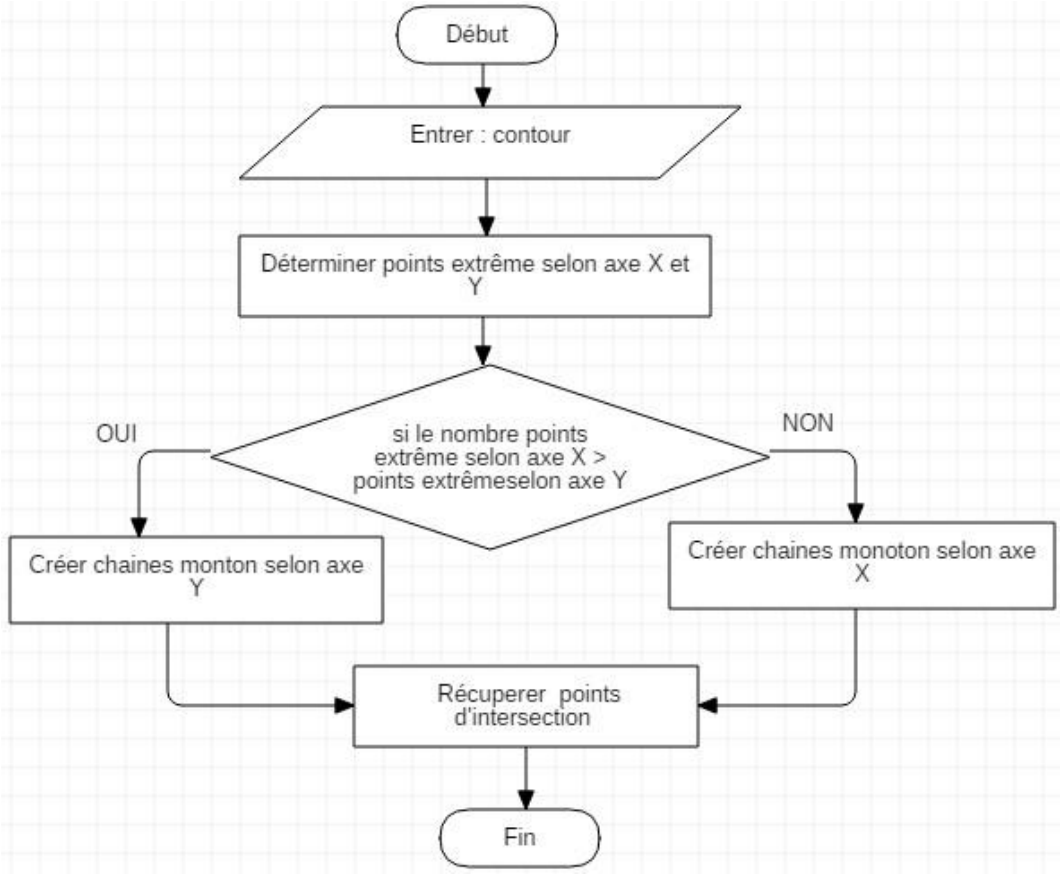
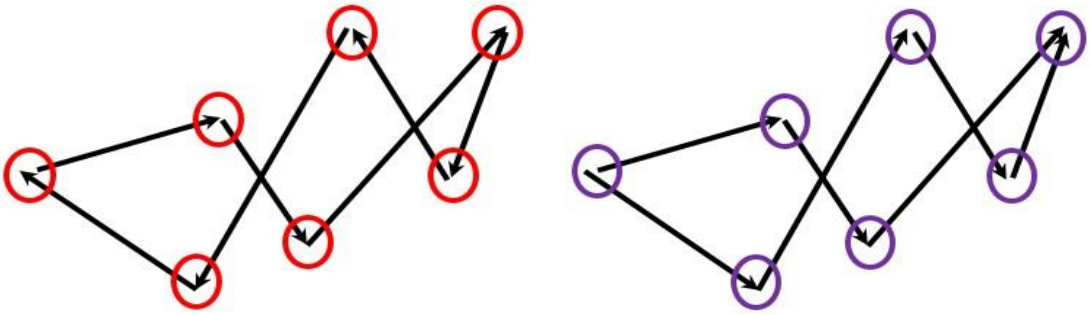


Figure 3.12. Détermination des points d'intersection.

➢ **Détermination des points extrêmes selon l'axe X et selon l'axe Y :** avant de déterminer les points extrêmes, nous avons changé la direction de tous les segments du contour (Figure 3.13). Nous avons choisi la direction de gauche à droite pour l'axe X et de bas en haut pour l'axe Y.



a. Direction avant le changement. b. Direction après le changement.

Figure 3.13. Changement de direction des segments selon l'axe X.

A cet effet, pour chaque segment comparer la coordonnée X du sommet de début avec la coordonnée X du sommet de fin.

- Si la cordonnée X du sommet de début est plus grande que la cordonnée X du sommet de fin, alors les deux sommets sont permutés.
- Sinon, passer au segment suivant.

La même procédure est suivie pour l'axe Y mais en remplaçant la coordonnée X par la coordonnée Y.

Après ce traitement, nous avons traité tous les cas possibles pour déterminer les points extrêmes :

✓ ***Cas simple :***

❖ Deux segments ou plus partagent le même sommet de début, c'est un point extrême négatif (Figure 3.14).

❖ Deux segments ou plus partagent le même sommet de fin, c'est un point extrême positif (Figure 3.14).

✓ ***Cas dégénéré :*** pour ce cas, un sommet est un point extrême positif et un point extrême négatif en même temps (Figure 3.15).

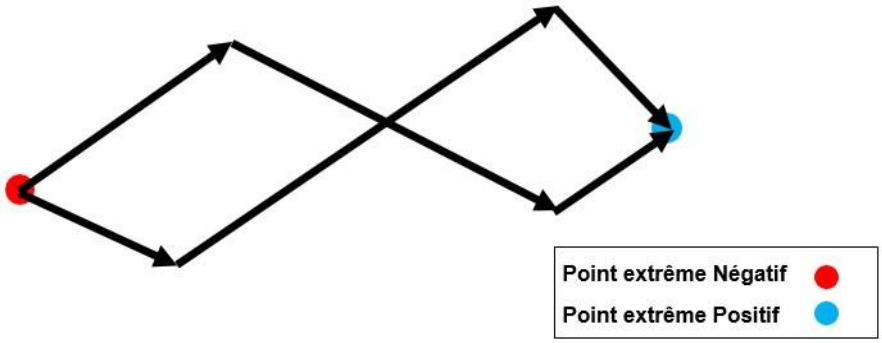


Figure 3.14. Points extrêmes pour le cas simple.

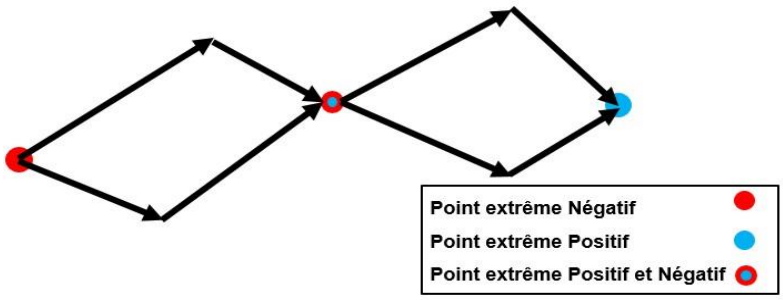


Figure 3.15. Points extrêmes pour le cas dégénéré.

✓ ***Cas particulier :*** ce cas se produit si le segment est vertical pour la direction suivant l'axe X ou bien s'il est horizontal pour la direction suivant l'axe Y (Figure 3.16). La détermination des points extrêmes des segments horizontaux (verticaux) passe par :

- ❖ Si pour un segment son sommet de début a les mêmes coordonnées qu'un sommet d'un segment vertical ou horizontal, alors c'est un point extrême négatif.
- ❖ Si pour un segment son sommet de fin a les mêmes coordonnées qu'un sommet d'un segment verticale ou horizontale, alors c'est un point extrême positif.

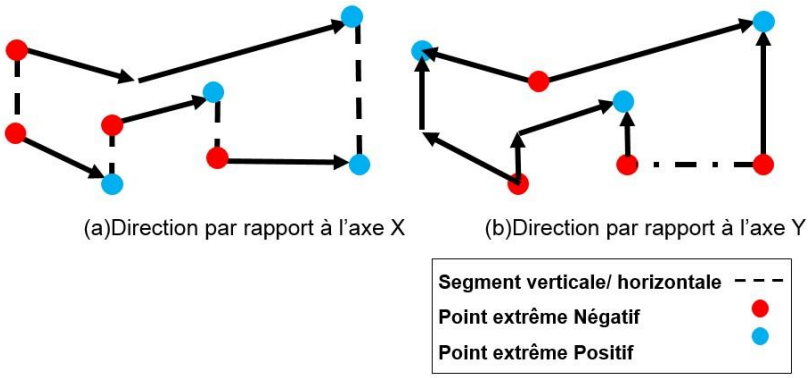


Figure 3.16. Points extrêmes pour le cas particulier.

➢ **Création des chaines monotones :** une chaine est une séquence connectée de segments de lignes selon une direction (axe X ou axes-Y) [17]. Le premier segment de la chaine monotone son sommet de début c'est un point extrême négatif et le dernier segment de la chaine son sommet de fin c'est un point extrême positif. Ils existent deux types de chaines monotones :

- ➢ Chaines monotones non parallèles ne contenant pas de segments parallèles à la direction de balayage (Figure 3.17 et Figure 3.18).
- ➢ Chaines monotones parallèles contenant au moins un seul segment parallèle à la direction de balayage (Figure 3.19).

Le pseudo Algorithme 4 montre le traitement effectué.

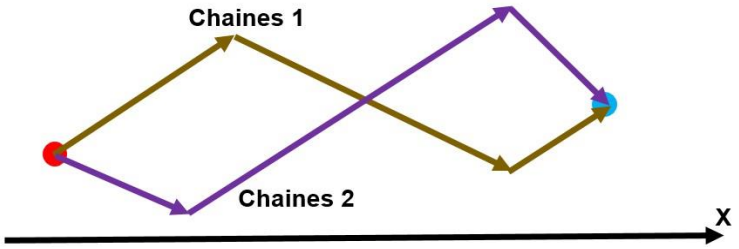


Figure 3.17. Chaines monotones non parallèles selon l'axe X.

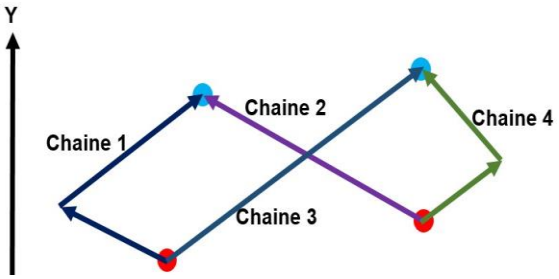


Figure 3.18. Chaines monotones non parallèles selon l'axe Y.

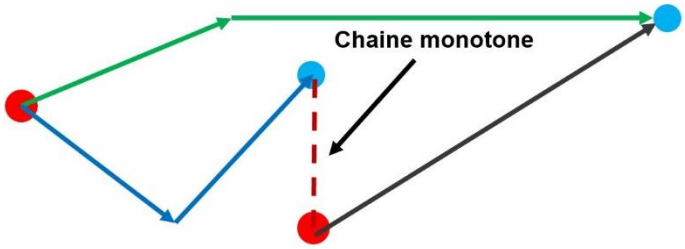


Figure 3.19. Chaine monotone parallèle à la direction de balayage.

Algorithm 4: Déterminer chaines monotone

```
Data: tableaux point_extrême_négatif
    Result: matrice chaines monotone
49 Segment_DXF segment_temporaire ;
50 Matrice [nombre de point extrême] [] chaine monotone ;
51 for i ← 0 to nombre nombre de point extrême do
52     for j ← 0 to nombre points extrêmes négatifs do
53         for k ← 0 to nombre segments relier extrême négatifs do
54             segment_temporaire = segments_contour [k] ;
55             while le sommet_fin de segment_temporaire n'est pas un point extrême positif do
56                 chaine_monoton[i][ajouter le segment temporaire à la matrice de chaine
                   monotone] ;
57                 segment temporaire = récupérer le segment suivant ;
58             end
59         end
60     end
61 end
62 Ajouter toutes les segments verticaux ou horizontaux a la matrice chaine monotone ;
63 return chaine monotone ;
```

➢ **Récupérer les points d'intersection :** une intersection peut être détectée en vérifiant s'il y a une chaîne monotone actuelle intersectée par une autre chaine monotone car il ne sort pas de l'intersection dans une chaîne monotone (Figure 3.20). Le point d'intersection I* entre les segments P_i et P_k ,P_{k+1} peut être calculé comme suit :

$$I^* = P_i + t\overrightarrow{P_iP_{i+1}} = P_k + s\overrightarrow{P_kP_{k+1}} \text{ [17]} \quad (3.5)$$

Les paramètres t et s sont obtenus après la résolution de l'équation (3.5). Ils sont donnés par :

$$t = \frac{\overrightarrow{P_iP_k} \times \overrightarrow{P_kP_{k+1}}}{\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_kP_{k+1}}} \text{ [17]} \quad (3.6)$$

$$s = \frac{\overrightarrow{P_iP_k} \times \overrightarrow{P_iP_{i+1}}}{\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_kP_{k+1}}} \text{ [17]} \quad (3.7)$$

Le point d'intersection I* est valide si les deux conditions suivantes sont vérifiées :

$$0 \leq t \leq 1.0$$
$$0 \leq s \leq 1.0$$

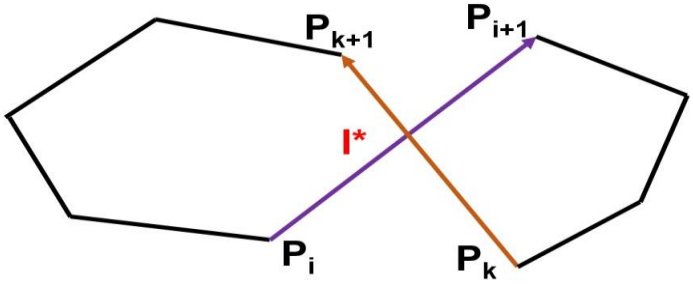


Figure 3.20. Méthode de calcul de l'auto-intersection.

L'algorithme suivant montre le processus suivi pour détecter efficacement les intersections.

Algorithme Détection des points d'intersection

Entrée : contour contienne un ensemble de sommet du contour.

Sortie : un ensemble de points d'intersection du contour.

Étape 1 : calculer le nombre x-max du point extrême de la direction X et y-max de la direction Y.

Étape 2 : Si x-max <y-max, créez une liste de chaînes de balayage dans un ordre croissant de valeur X. Sinon, créez une liste de chaînes de balayage dans un ordre croissant de valeur Y.

Étape 3 : obtenir le premier bord qui n'est pas visité à partir de la liste de chaîne rapide et le définir comme bord actif. Cherchez la liste de bord actif. Si la liste des arêtes actives est nulle, le système actif bord est inséré dans la liste des bords actifs. Sinon, allez à Étape 4.

Étape 4 : rechercher dans la liste des arêtes actives et définir toutes les arêtes inactif. S'il existe un bord appartenant à la même chaine monotone en tant que bord actif, supprimez le bord de la liste de bord actif. En même temps, insérez le bord actif dans la liste des bords actifs.

Étape 5 : détectez l'intersection entre le bord actif et les bords inactifs. Si vous avez une intersection, passez à l'étape 6. Sinon, passez à l'étape 3.

Étape 6 : calculez le point d'intersection et insérez le et les points de départ des deux bords recoupés dans la liste des points d'intersection.

Étape 7 : répétez les étapes 3 à 6 jusqu'à ce que tous les bords de balayage de la liste des chaînes soient visités.

3.2.5. Détermination des boucles valides

Pour éliminer les boucles d'un contour, nous devons déterminer quelles boucles doivent être retirées. Dans cette partie, nous présentons une méthode simple pour supprimer efficacement les boucles non valides. L'algorithme suivant résume les étapes suivies :

Algorithme Détection des boucles valides :

Entrée : contour, ensemble de points d'intersection du contour.

Sortie : un ensemble de boucles valide.

Étape 1 : vérifiez si la boucle (contour) contient des points d'intersection. Si oui, alors passez à l'étape 2. Sinon, terminez le processus.

Étape 2 : obtenir le segment sortant du point d'intersection courant. Pour l'obtenir, nous avons calculé l'angle pour chaque segment sortant du point d'intersection par rapport à l'horizontal et nous prenons le segment qui a l'angle minimum. Après, nous passons à l'étape 3 (Figure 3.21).

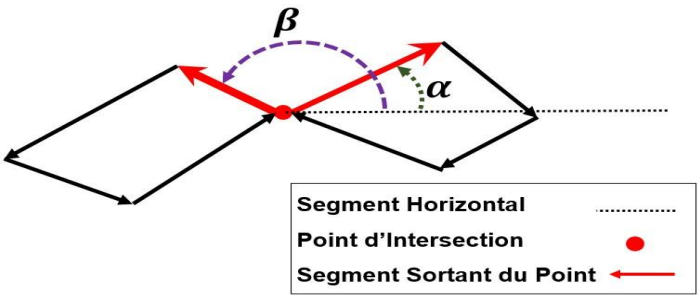


Figure 3.21. Calcul de l'angle pour les segments issus du point d'intersection.

Étape 3 : vérifier le segment choisi s'il est dans la liste spécifique. Sinon, nous prenons ce segment et nous l'enregistrons dans la liste des boucles. Si oui, nous passons à un autre segment sortant du point d'intersection. Ajouter ce segment à la liste spécifique.

Étape 4 : parcourir tous les segments en commençant du segment obtenu jusqu'à arriver à un autre point d'intersection. Ajouter ces segments parcourus à la liste boucle et choisir le segment sortant de ce dernier qui a l'angle minimum et vérifier l'étape 3.

Étape 5 : répéter l'étape 4 jusqu'à retourner au point d'intersection courant et enregistrer la première boucle dans la liste des boucles.

Étape 6 : répéter l'étape 2 jusqu'à vérifier tous les segments sortant du point d'intersection courant. Vider la liste spécifique et enregistrer les boucles restantes dans la liste des boucles.

Étape 7 : boucler sur la liste des boucles et vérifier l'étape 1.

Étape 8 : enregistrer toutes les boucles obtenues et vérifier chaque boucle si elle est valide ou non en utilisant l'Algorithme 2.

Étape 9 : supprimer toutes les boucles invalides et garder uniquement les boucles valides.

3.2.6. Décalage d'un contour

Pour créer un contour décalé « offset » d'un contour donné, il suffit de créer pour chaque segment du contour un segment qui est lui est parallèle et qui a les mêmes caractéristique que le segment original (orientation et longueur) (Figure 3.22).

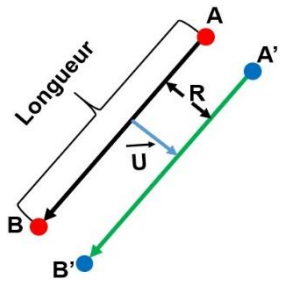


Figure 3.22. Segment décalé.

Les coordonnées des points du segment résultant sont donnés par :

Sommet A' : $XA' = XA + R \times UX$ $\quad YA' = YA + R \times UY$ (3.8)

Sommet B' : $XB' = XB + R \times UX$ $\quad YB' = YB + R \times UY$ (3.9)

La Figure 3.23 montre les segments décalés d'un contour.

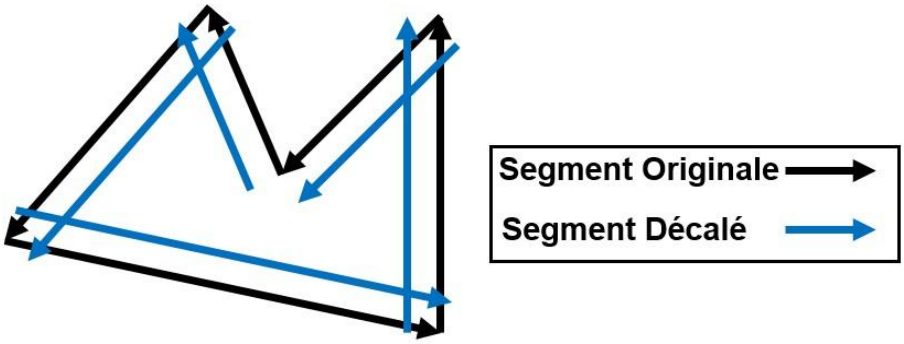


Figure 3.23. Contour décalé.

Après la création des segments décalés, nous devons passer par les traitements suivants :

- Découper chaque deux segment consécutif ayant un point d'intersection réel (Figure 3.24).

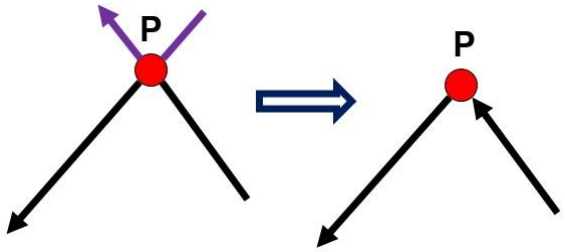


Figure 3.24. Découpage de deux segments.

Pour faire ce traitement :

1. Calculer le point d'intersection des deux segments en utilisant l'équation 3.5.

2. Le sommet de fin du premier segment devient le point d'intersection et le sommet de début du deuxième segment qui le suit devient le point d'intersection.

Si le point d'intersection est imaginaire. Dans ce cas, prolongez les deux segments décalés jusqu'au point d'intersection imaginaire (raccordement standard) (Figure 3.25).

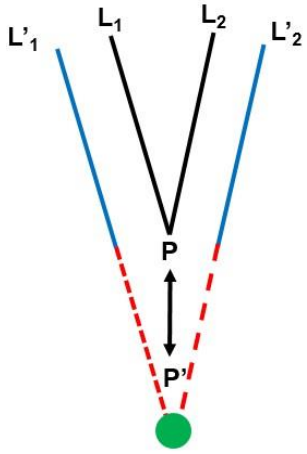


Figure 3.25. Raccordement standard.

Cette méthode n'est pas vraiment pratique car la distance entre le sommet P'et P est loin lors du déplacement du spot Laser. Donc, ce trajet est inutile. Plus l'angle entre les deux segments décalés est petit, plus la distance entre P' et P augmente. Pour la réduire, nous avons pensé à une autre méthode : c'est d'ajouter deux (02) sommets T1, T2 à la place du sommet aigu (raccordement optimisé) (Figure 3.26).

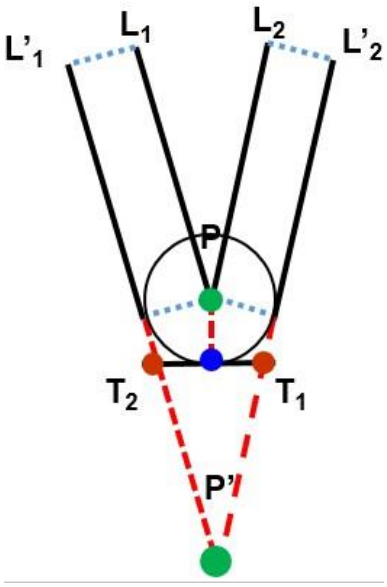


Figure 3.26. Raccordement optimisé.

La Figure 3.27 montre le contour décalé d'un contour initial.

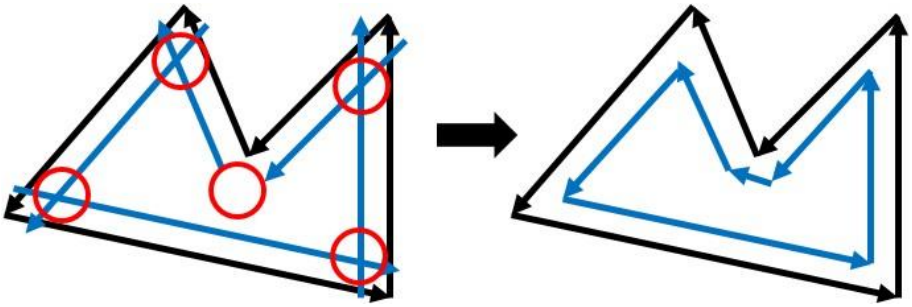


Figure 3.27. Raccordement final.

3.2.7. Génération des contours décalés d'un contour

L'algorithme suivant est utilisé pour la génération des contours décalés d'un contour donné.

Algorithme de génération des offsets :

Entrée : contour, spot (distance de décalage).

Sortie : un ensemble de contours décalé.

Int distance=0 ;

- **Étape 1 :** générer un offset pour le contour donné d'une distance de décalage et passer à l'étape 2;
- distance= distance+spot ;
- **Étape 2 :** déterminé les points extrêmes de l'offset et passer à l'étape 3 ;
- **Étape 3 :** créer les chaines monotones et passer à l'étape 4 ;
- **Étape 4 :** récupérer les points d'intersection et aller à l'étape 5 ;
- **Étape 5 :** déterminer les boucles valides et passer à l'étape 6 ;
- **Étape 6 :** répéter les étapes de 1 à 6 jusqu'à ce que l'offset change son orientation.

La Figure 3.28 montre le résultat final après l'application de l'algorithme.

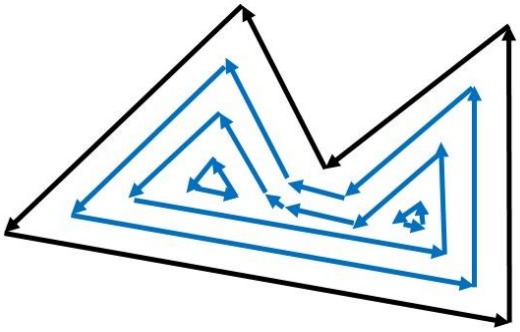


Figure 3.28. Contours décalés.

3.2.8. Nettoyages des contours décalés

Le but de cette tâche est de supprimer les contours décalés situés à l'intérieur des fils du premier niveau qui sont générés par le contour père (Figure 3.29).

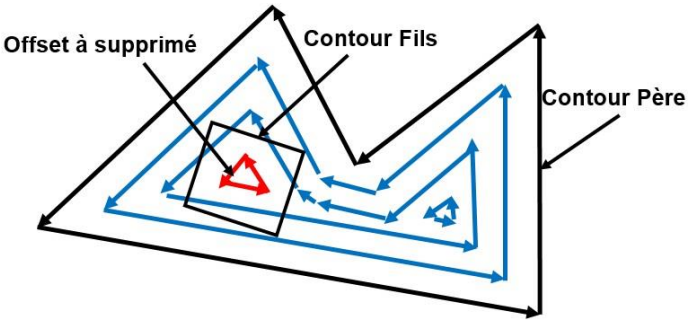


Figure 3.29. Contours décalés situés à l'intérieur des fils.

Les étapes à suivre pour réaliser cette tâche sont :

- Créer pour chaque fils du premier niveau un décalage externe.
- Vérifier pour chaque offset du contour père s'il est situé à l'intérieur du décalage externe du fils. S'il est à l'intérieur, il est supprimé (Figure 3.30).

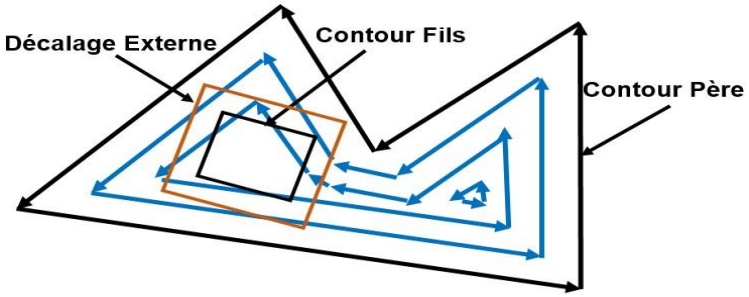


Figure 3.30. Suppression des contours décalés situés à l'intérieur des fils.

Mais il y a toujours des parties du segment qui sont à l'intérieur du décalage externe du fils qu'on doit les supprimer (Figure 3.31). Les étapes suivantes montrent la suppression :

1. Déterminer les points d'intersection entre le décalage externe et les offsets du contour père en utilisant les chaines monotones.

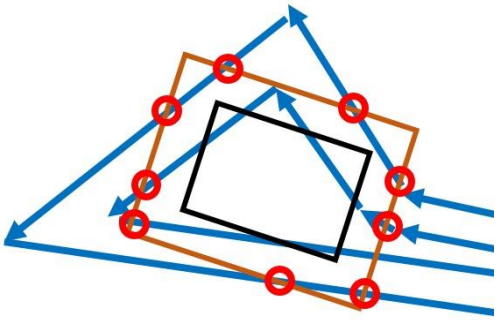


Figure 3.31. Détermination des points d'intersection.

2. Subdiviser chaque segment contenant le point d'intersection en sous-segments (Figure 3.32).

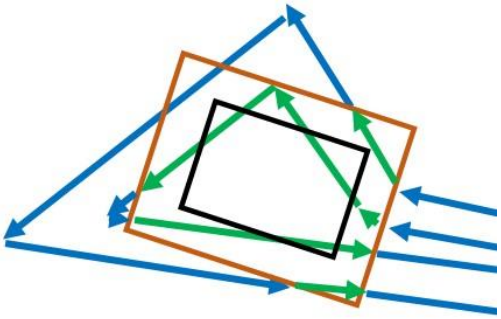


Figure 3.32. Subdivision des segments.

3. Supprimer les sous-segments situés à l'intérieur du décalage externe du contour fils (Figure 3.33).

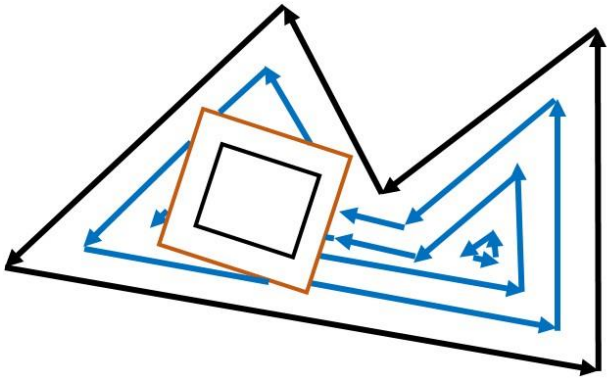


Figure 3.33. Suppression des sous segments.

Nous pouvons rencontrer le cas où les fils du premier niveau s'intersectent entre-eux (Figure 3.34).

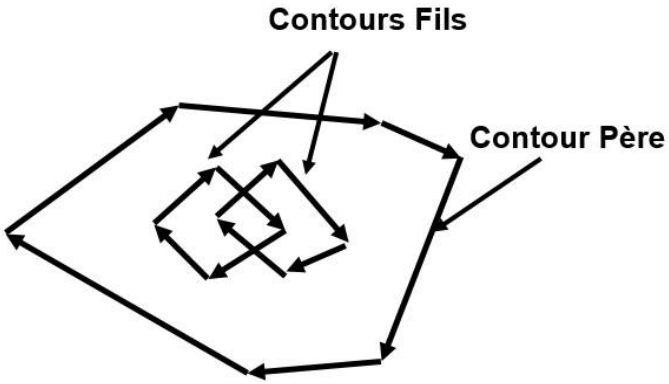
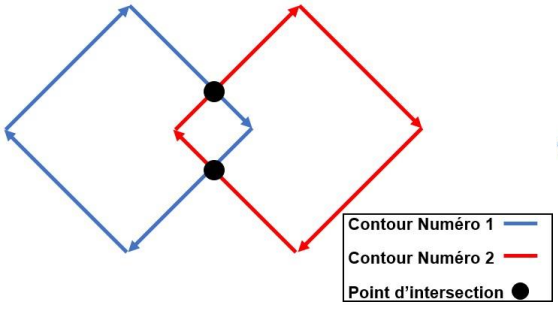


Figure 3.34. Intersection entre contours fils.

Le traitement de ce cas suit les étapes suivantes :

Étape 1 : récupérer les chaines monotones des deux contours et détecter les points d'intersection issus des deux chaines monotones. Ensuite, découper les segments qui sont intersectés en sous-segments et passer à l'étape 2.



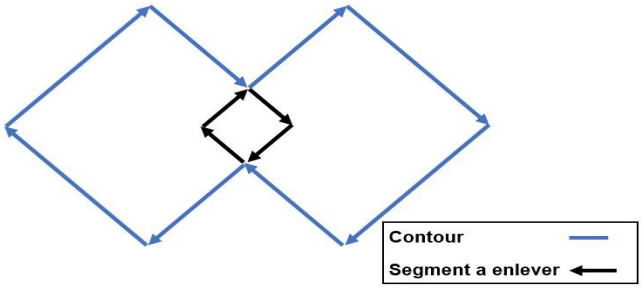


Figure 3.35. Détection des points d'intersection. **Figure 3.36.** Découpe des segments.

Étape 2 : si les sous-segments du premier contour sont à l'intérieur du deuxième contour, alors les supprimer et vice versa pour les sous-segments du deuxième contour.

Étape 3 : les deux contours ouverts vont être fusionnés pour former un contour fermé (Figure 3.37).

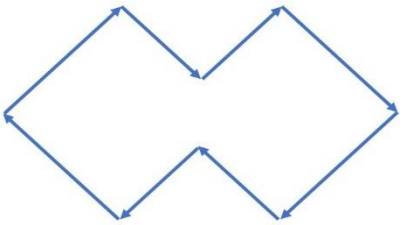


Figure 3.37. Fusion des contours fils.

Étape 4 : refaire les étapes de nettoyage vues précédemment.

3.2.9. Génération du trajet de micro-gravure

Dans cette partie, le fichier texte contenant le trajet de micro-gravure est généré. Pour atteindre cet objectif, les étapes suivantes sont nécessaires [7] :

- Détection des segments de déplacement à vide afin de mettre l'état du Laser à 0 pour ne pas marquer ces segments.
- Calcul de la vitesse de la table suivant les deux axes X et Y.
- Génération du fichier texte du trajet de micro-gravure.

3.3. Conception du système

3.3.1. Diagramme cas d'utilisation global

Dans ce qui suit, nous allons présenter une vue générale de notre système par un diagramme cas d'utilisation global (Figure 3.38). Le système est divisé en plusieurs parties :

- Création des contours.
- Récupération des paramètres des entités géométriques.
- Changement du sens des contours originaux.
- Détection des fils des contours originaux.
- Création des chaines monotones des contours originaux.
- Détermination des boucles valides et la suppression des boucles invalides pour les contours originaux.
- Génération des offsets (contours décalés).
- Génération du trajet de micro-gravure.

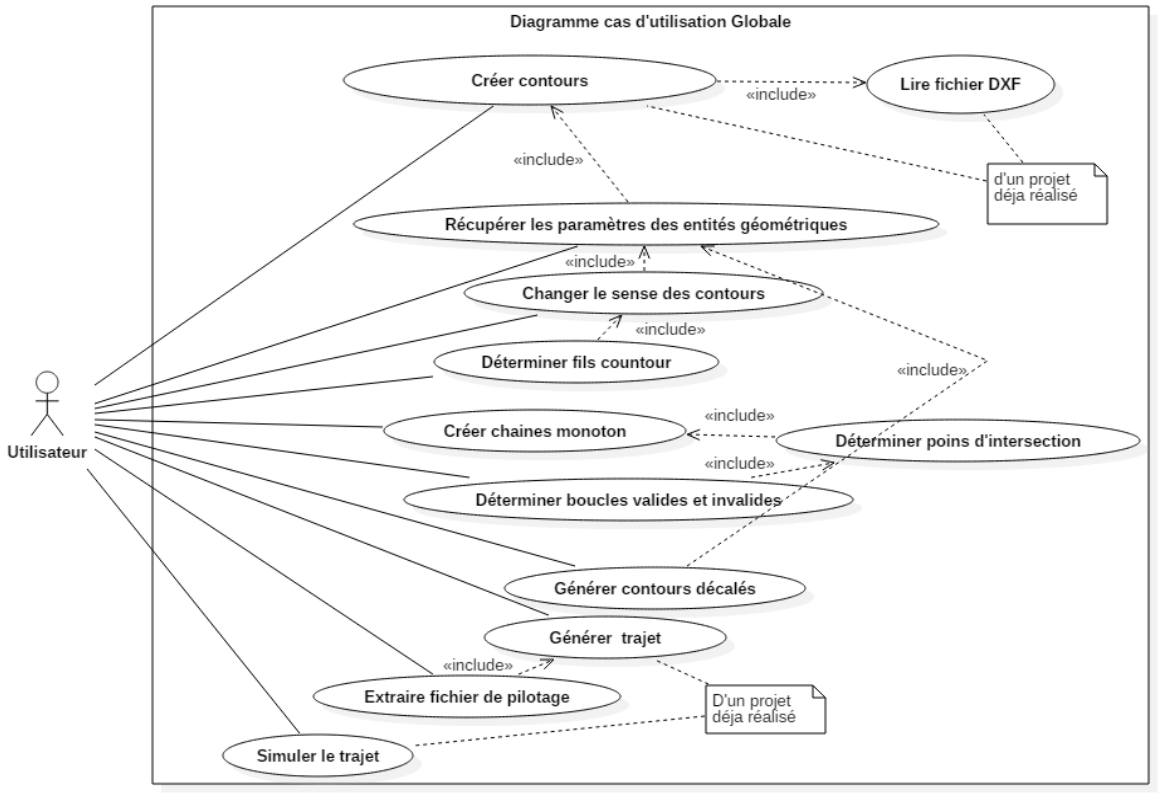


Figure 3.38. Diagramme cas d'utilisation global.

Maintenant, nous allons détailler chaque cas d'utilisation que nous venons de mettre en diagramme de séquence.

3.3.2. Diagrammes de séquence système

Le diagramme de séquence représente l'aspect chronologique de l'envoi des messages « échanges de messages ». Sans oublier que le diagramme de séquence va nous permettre de compléter et de détailler le diagramme de cas d'utilisation global en mettant en évidence les objets et leurs interactions. Sans plus tarder, nous allons présenter les différents diagrammes de séquence que regroupe notre système. La première étape du cas d'utilisation consiste en la création des contours.

La Figure 3.39 donne le diagramme de séquence relatif à la récupération des paramètres des entités géométriques.

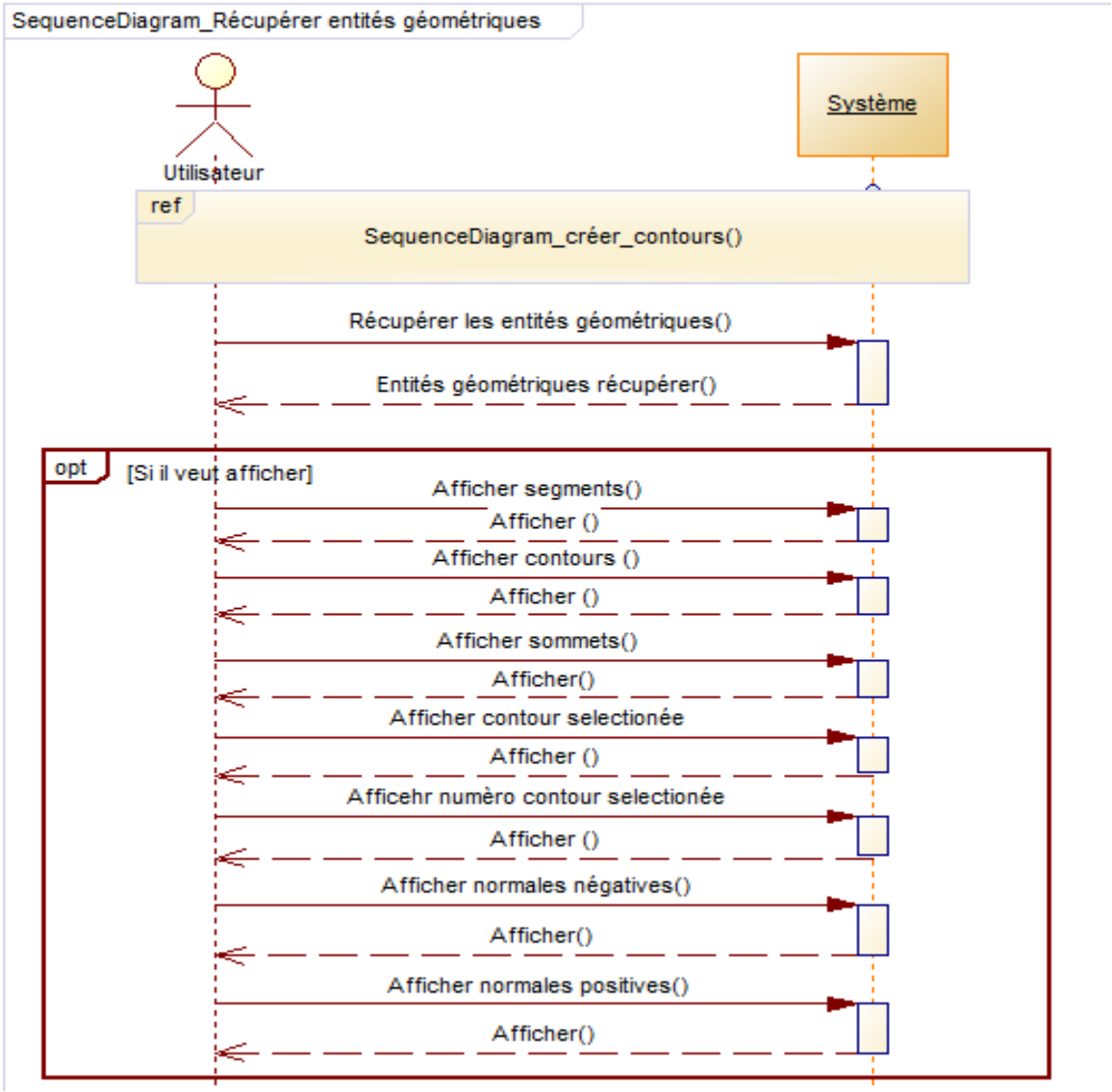


Figure 3.39. Diagramme de séquence de récupération des paramètres des entités géométriques.

La Figure 3.40 donne le diagramme de séquence relatif au changement du sens des contours originaux.

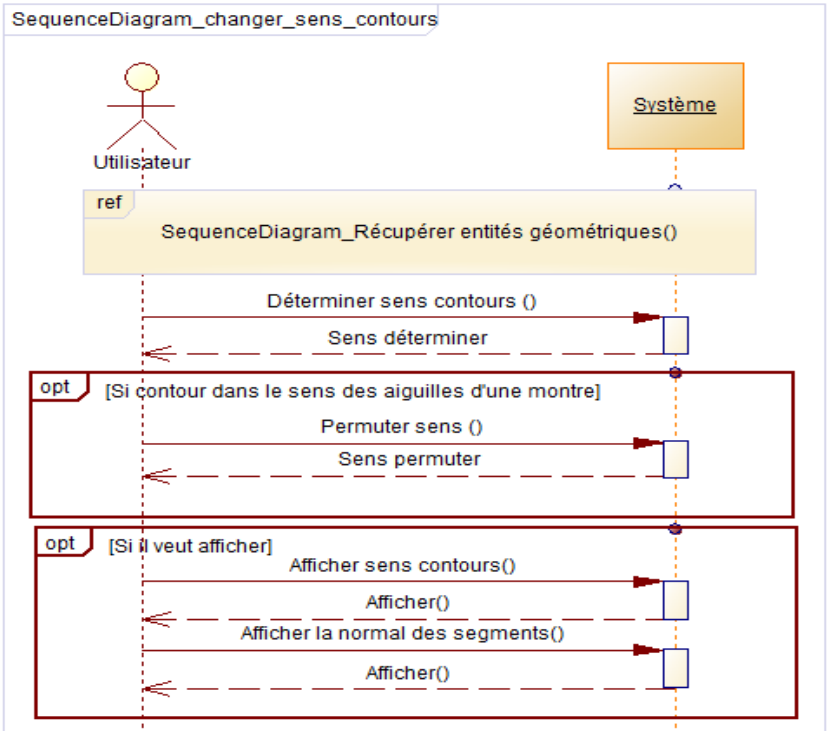


Figure 3.40. Diagramme de séquence relatif au changement du sens des contours originaux.

La Figure 3.41 donne le diagramme de séquence relatif à la détection des fils des contours originaux.

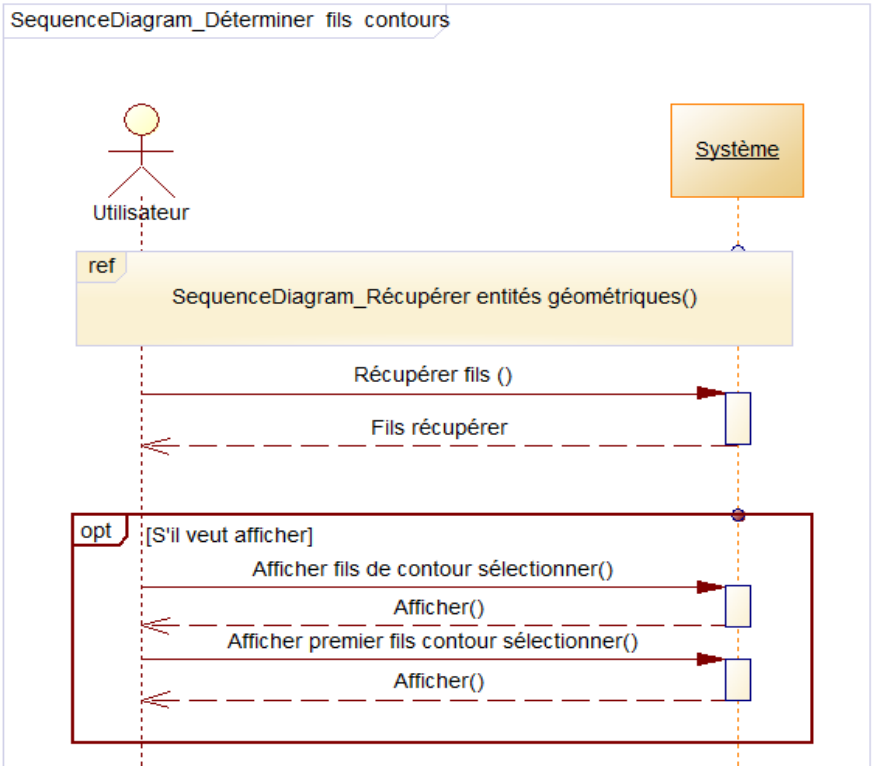


Figure 3.41. Diagramme de séquence relatif à la détection des fils des contours originaux.

La Figure 3.42 donne le diagramme de séquence relatif à la création des chaines monotones des contours.

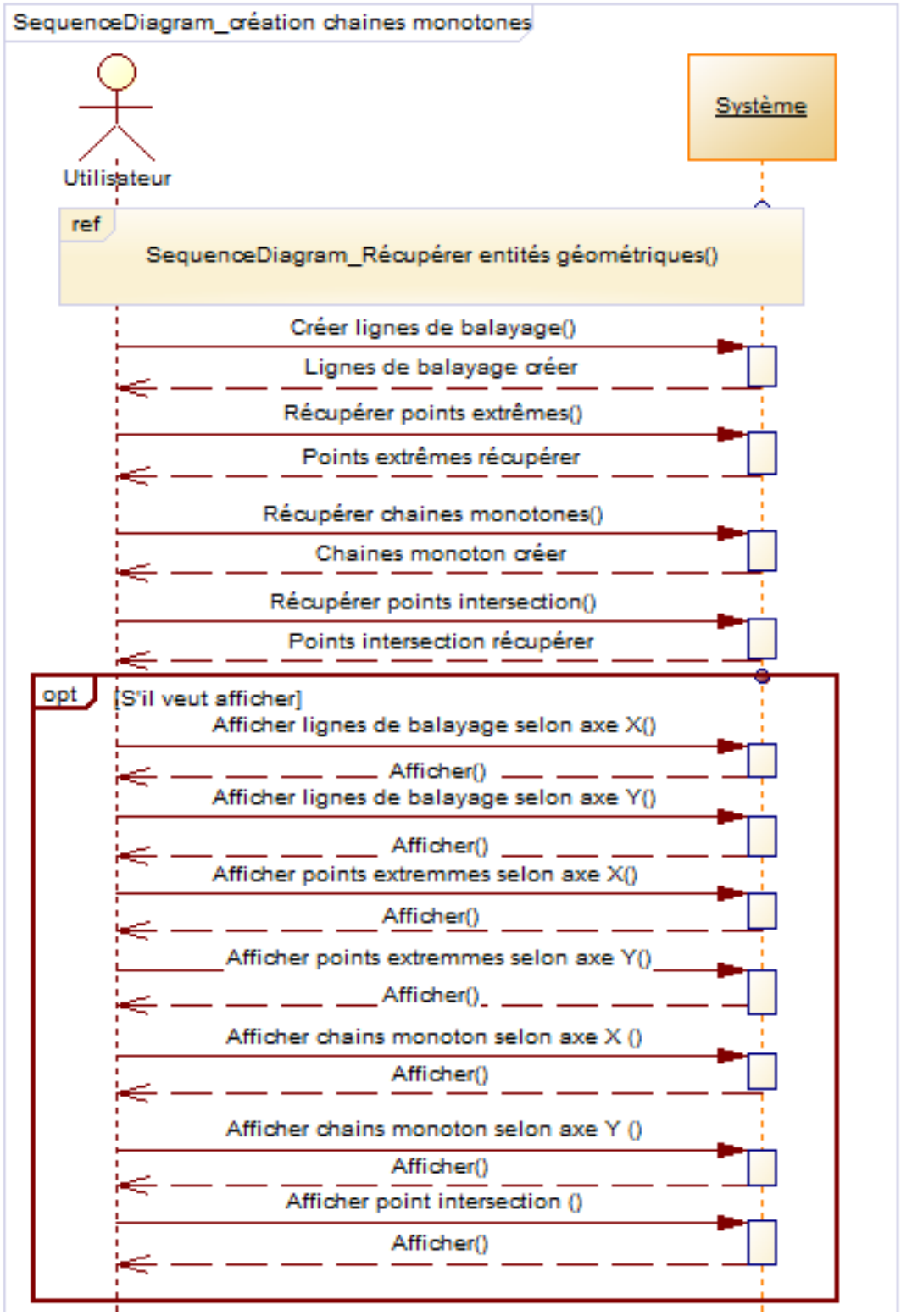


Figure 3.42. Diagramme de séquence relatif à la création des chaines monotones des contours.

La Figure 3.43 donne le diagramme de séquence relatif à la détermination des boucles valides et la suppression des boucles invalides pour les contours.

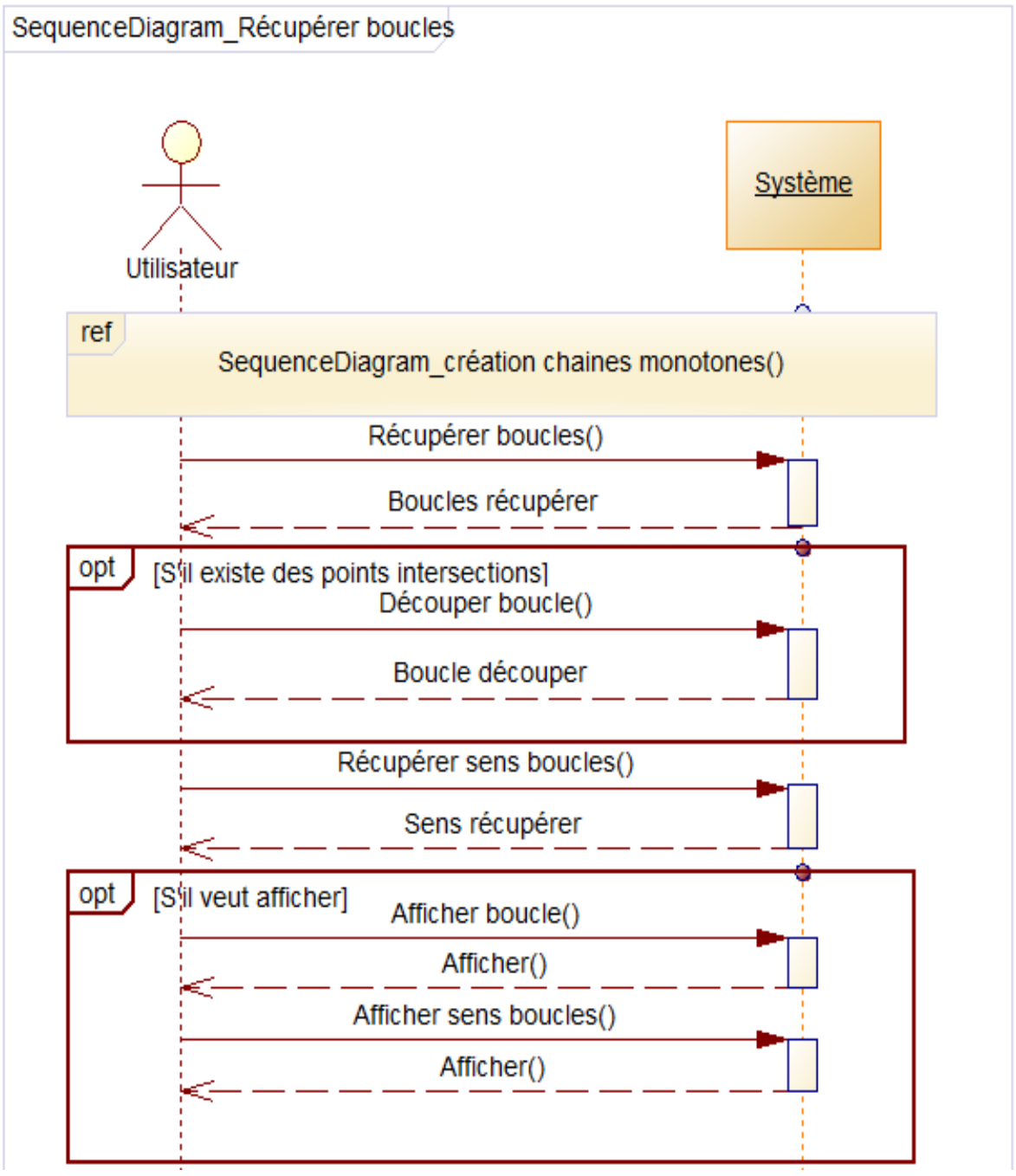


Figure 3.43. Diagramme de séquence relatif à la détermination et au traitement des boucles.

La Figure 3.44 donne le diagramme de séquence relatif à la création des contours décalés.

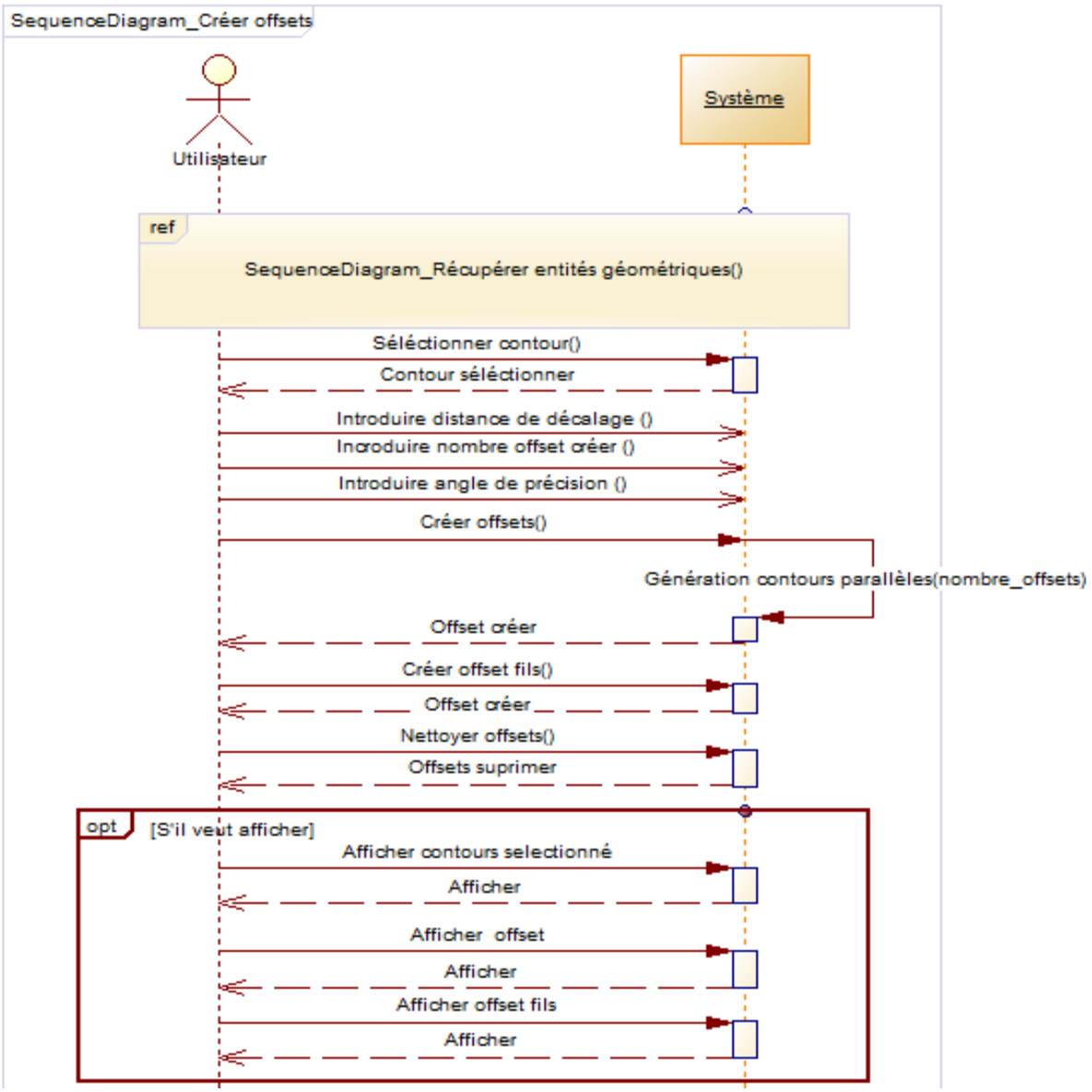


Figure 3.44. Diagramme de séquence relatif à la création des contours décalés.

3.3.3. Diagramme de classe

Après avoir défini notre problématique et avoir analysé nos besoins, nous allons montrer les solutions que nous avons proposées pour résoudre la problématique toute en prenant en compte les besoins définis précédemment et en décrivant l'architecture du système. Le diagramme de classe qui représente une vue statique du système est donné par la Figure 3.45. Les différentes classes utilisées dans notre conception sont les suivantes :

- Classe Segment_DXF
- Classe Contour_DXF

- Classe Sommet_DXF
- Classe Limites
- Classe Motif_DXF
- Classe Intersect_DXF
- Classe Boucle_DXF

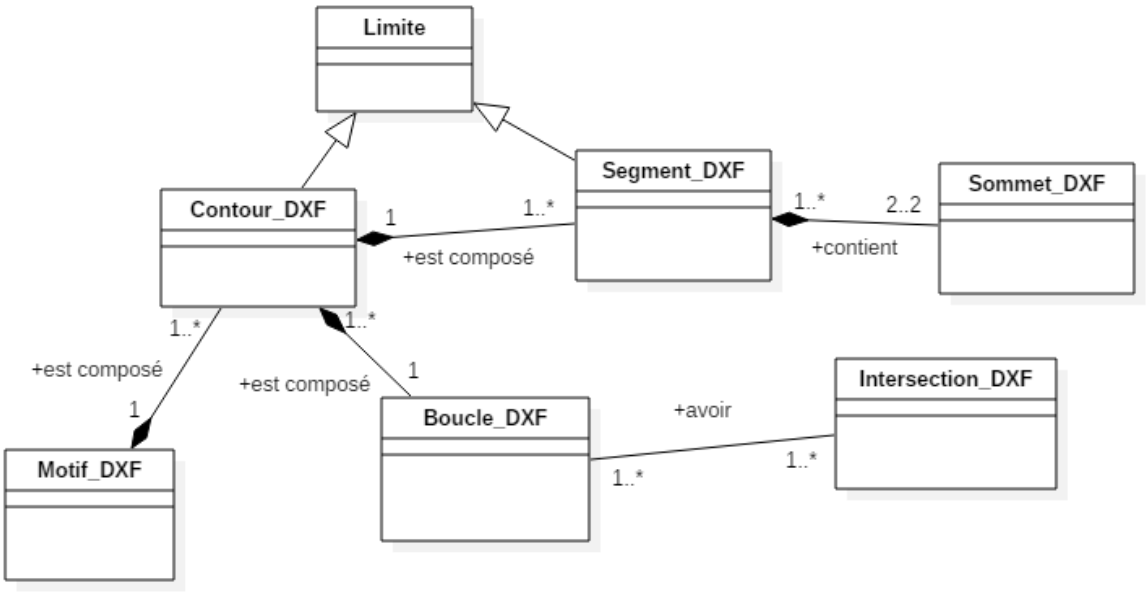


Figure 3.45. Diagramme de classe général.

Dans ce qui suit, nous allons montrer en détail les composants (attributs + fonctions) de chaque classe tout en faisant la description de chaque classe.

➢ **Classe Sommet_DXF :** c'est la classe qui va permettre de définir l'entité du point qui est représenté par les coordonnées X et Y de type double (Figure 3.46). Il y a aussi la variable distance de type double qui nous aide à calculer la distance entre deux sommets. Parmi les fonctions de la classe Sommet_DXF, nous avons :

- calculer_distance (Sommet_DXF sommet) : c'est une fonction pour calculer la distance entre deux sommets.
- check_equal_sommet (Sommet_DXF sommet) : retourne un booléen qui vérifie si deux sommets sont identiques par rapport aux coordonnées X et Y.

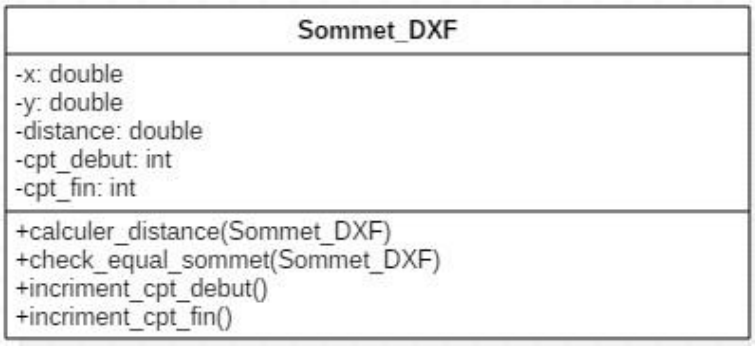


Figure 3.46. Classe Sommet_DXF.

➢ **Classe Segment_DXF :** c'est la classe qui contient deux sommets, sommet début pour le premier point et par le sommet_fin pour le deuxième point de type (Sommet_DXF) et d'une valeur int pour l'indice du contour afin de préciser à quel contour appartient le segment. De plus, la variable longueur de type double utilisée pour calculer la longueur du segment et type_segment de type AnsiString utilisée pour déterminer les chaines monotones. Parmi les fonctions de la classe Segment_DXF, nous avons :

- cree_normal () : fonction pour créer la normale du segment.
- check_equal_segment (Segment_DXF seg) : retourne un booléen qui vérifie si un segment est identique avec le segment courant.
- calculer_angle (Segment_DXF seg) : retourne l'angle entre deux segments.

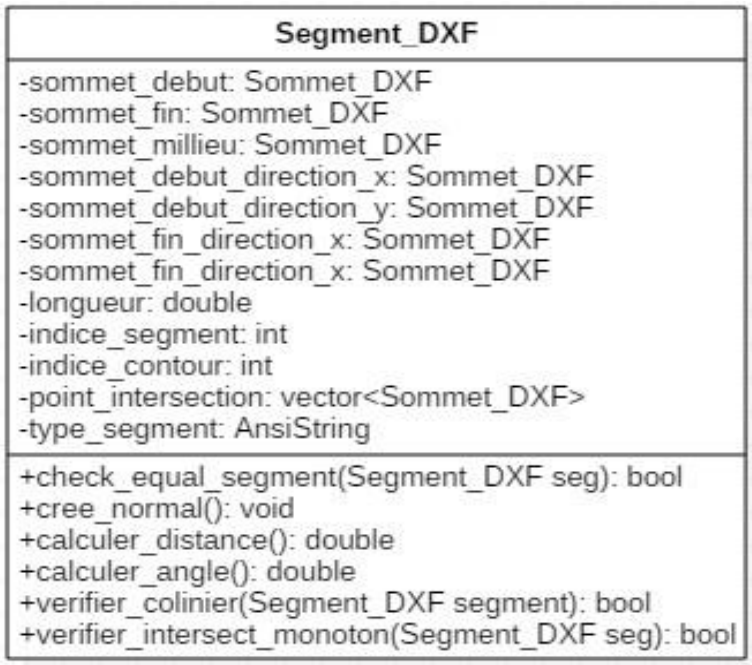


Figure 3.47. Classe Segment_DXF.

➢ **Classe Contour_DXF :** c'est la classe qui regroupe l'ensemble des segments de type (Segment_DXF) et deux variables type et sens de type AnsiString. La variable type pour

déterminer si le contour est ouvert ou fermé. La variable sens pour dire si le contour est dans le sens des aiguilles d'une montre ou dans le sens antihoraire. Parmi les fonctions de la classe Contour_DXF, nous avons :

- determiner_sens () : c'est une fonction qui détermine le sens du contour (sens des aiguilles d'une montre ou dans le sens antihoraire).
- bool checkInside (Contour_DXF contour) : retourne un booléen qui vérifie si un contour est à l'intérieur d'un contour courant.
- genere_contours_paralelles_recursive (double dist, int stop, double angle_combo, bool limit, bool invalid_contour) : c'est une fonction qui travaille d'une façon récursive pour générer les offsets d'un contour.

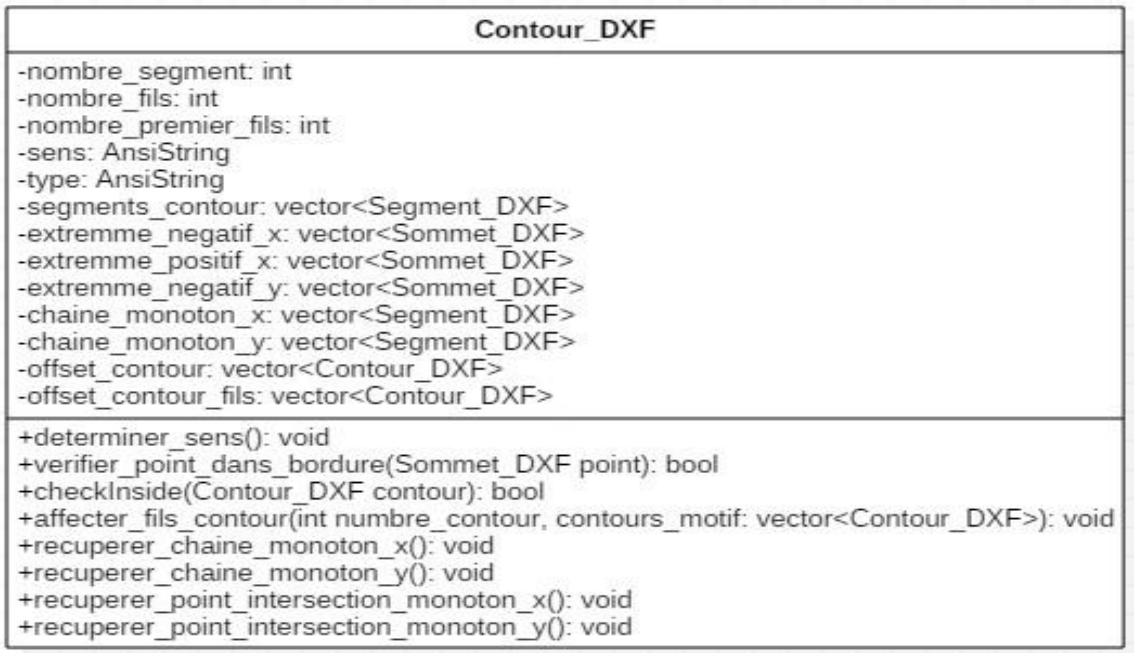


Figure 3.48. Classe Contour_DXF.

➢ **Classe Motif_DXF :** c'est la classe qui regroupe l'ensemble des contours et segments. De plus, la variable nombre_contour représente le nombre de contours. Parmi les fonctions de la classe Motif_DXF, nous avons :

- recuperer_sommet () : récupérer tous les sommets.
- recuperer_segments (vector<segment_dxf> class_sommet_dxf) : récupérer l'ensemble des segments qui se trouvent dans la classe segment_dxf.
- recuperer_contours (vector<contour_dxf> contours_motifs) : récupérer l'ensemble des segments qui se trouvent dans la classe contour_dxf.
- recuperer_monoton_contours () : récupérer l'ensemble des chaines monotones selon les deux axes X et Y pour chaque contour.

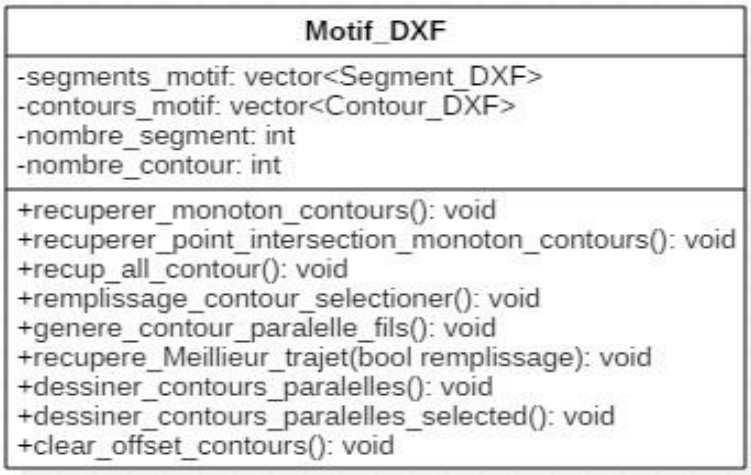


Figure 3.49. Classe Motif_DXF.

➢ **Classe Intersect_DXF :** c'est la classe qui contient un objet de type Sommet_DXF et un tableau qui contient les indices des segments qui font le point d'intersection.

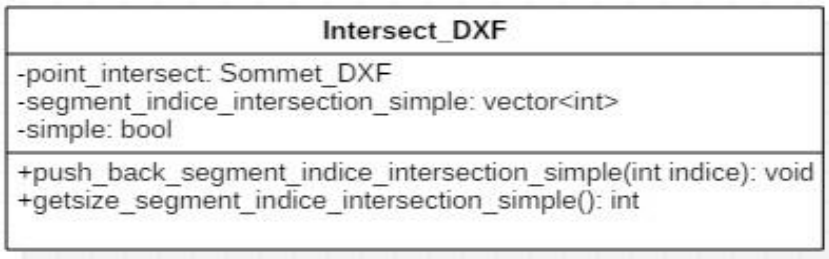


Figure 3.50. Classe Intersect_DXF.

➢ **Classe Boucle_DXF :** c'est la classe qui regroupe l'ensemble des segments et un tableau point_intersect de type Intersect_DXF. Parmi les fonctions de la classe Boucle_DXF, nous avons :

- affecter_angle_Boucle c'est une fonction qui affecte l'angle à chaque segment relié au même point d'intersection.

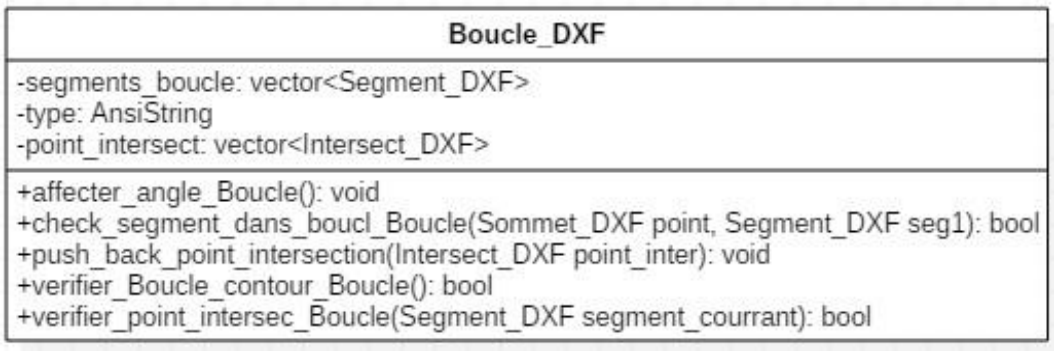


Figure 3.51. Classe Boucle_DXF.

➢ **Classe Limites :** c'est la classe qui contient quatre variables X_min, X_max, Y_max, Y_min de type double afin de déterminer la limite de chaque segment et contour.

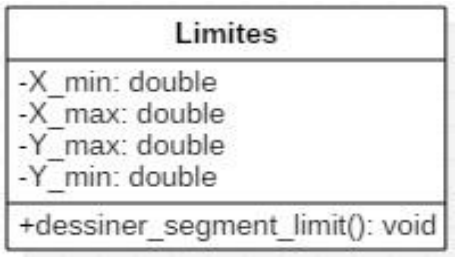


Figure 3.52. Classe Limites.

3.4. Conclusion

Dans ce chapitre, nous avons défini notre problématique et les différents objectifs à atteindre. Ensuite, nous avons présenté les solutions proposées afin d'atteindre les différents objectifs en utilisant les différents diagrammes de conception UML. Dans le chapitre suivant, nous allons présenter l'implémentation de notre application ainsi que la validation expérimentale des approches proposées et développées.

TROISIÈME PARTIE

Validation Expérimentale

Chapitre 4

Implémentation informatique et validation

1. **Introduction.**

2. **Présentation des langages utilisés.**

3. **Présentation de l’application.**

4. **Test et validation.**

5. **Conclusion.**

4.1. Introduction

La réalisation de notre travail s'achève par une présentation de l'application conçue ainsi que l'environnement de développement et les outils utilisés pour sa mise en œuvre. Il s'en suit une présentation de l'interface utilisateur et une évaluation du système à travers plusieurs exemples de validation réelle.

Ce chapitre traite les aspects techniques liés à l'implémentation et à la mise en œuvre de notre système. Nous présentons d'abord nos choix des technologies et des outils pour le développement du système. Nous montrons ensuite les principales fonctionnalités et les interfaces utilisateur offertes par notre application. Ensuite, nous passons à la partie validation en traitant un motif avec une validation expérimentale.

4.2. Implémentation

4.2.1. Présentation du langage C++

Le langage C++, inventé par Bjarne Stroustrup vers 1983 est une évolution orientée objet du langage C de Brian Kernighan et Denis Ritchie. Ce langage repose sur les mêmes mécanismes d'écriture et de génération. Il apporte notamment la gestion des exceptions, la surcharge des opérateurs et les Templates (liste non exhaustive). Enfin, une rétrocompatibilité a été gardée où les programmes en C sont compilés sans difficulté avec un compilateur C++.

Comme tout langage, C++ dispose d'une bibliothèque standard, c'est-à-dire des fonctions et de classes prédéfinies. Elle comporte notamment de nombreux patrons de classes et de fonctions permettant de mettre en œuvre les structures de données les plus importantes (vecteurs dynamiques, listes chaînées, chaînes, ...etc.) et les algorithmes les plus usuels. Parmi les environnements de développements en C++ nous citons Embarcadero C++ Builder, Code : Blocks (open-source), Dev-C++, Eclipse (open-source), Microsoft Visual C++, etc.

4.2.2. Présentation de la bibliothèque graphique OpenGL

OpenGL qui est l'acronyme d'Open Graphics Library est le premier environnement pour le développement d'applications graphiques 2D et 3D portables et interactives. Depuis son introduction en 1992, OpenGL est devenue l'interface de programmation d'applications graphiques (API) 2D et 3D la plus utilisée et la plus prise de l'industrie, apportant des milliers d'applications à une grande

variété de plates-formes informatiques. OpenGL favorise l'innovation et accélère le développement d'applications en intégrant un large éventail de fonctions de rendu, de mappage de texture, d'effets spéciaux et d'autres puissantes fonctions de visualisation. Les développeurs peuvent tirer parti de la puissance d'OpenGL sur toutes les plateformes populaires de postes de travail et de stations de travail assurant ainsi un déploiement étendu des applications.

4.2.3. Présentation d'Embarcadero C++ Builder

Embarcadero C++ Builder toute la puissance du langage C++ orienté objet, il offre la possibilité de développer rapidement des applications (RAD) sous Windows grâce à ses différentes bibliothèques. Il permet la création instantanée des interfaces utilisateurs pour les applications bureaux, applications mobiles, etc. puisqu'il offre une gestion de l'interface. Le compilateur C++ qui est inclus « Borland C++ Compiler » est un compilateur et un optimiseur de code de haut rendement et multithread qui agit en second plan. Sans arrêter le travail, les applications se compileront et s'exécuteront plus rapidement.

4.3. Présentation de l'application logicielle

Une application logicielle graphique et interactive Windows a été développée afin d'optimiser le trajet de micro-gravure par Laser. Plusieurs « Onglets » sont intégrés à cette dernière pour lancer les fonctionnalités intégrées.

4.3.1. Récupération et calcul des paramètres géométriques d'un motif

Après la première étape qui est la lecture du fichier DXF, elle vient cette étape qui consiste à récupérer les entités géométriques d'un motif (sommets, segments et contours). Par la suite, chaque ensemble de segments colinéaires sont remplacés par un seul segment. Cela est réalisé par un simple clic sur le bouton "Calcul des paramètres géométriques du motif" (Figure 4.1). Par la suite, il est possible de visualiser les entités géométriques en utilisant un des deux modes proposés :

- Visualisation globale : consiste à visualiser toutes les entités géométriques du motif en visualisant les sommets, les segments, les contours, leurs noms, les normales négatives et les normales positives.
- Visualisation locale : consiste à afficher les segments, les normales négatives et les normales positives d'un contour donné.

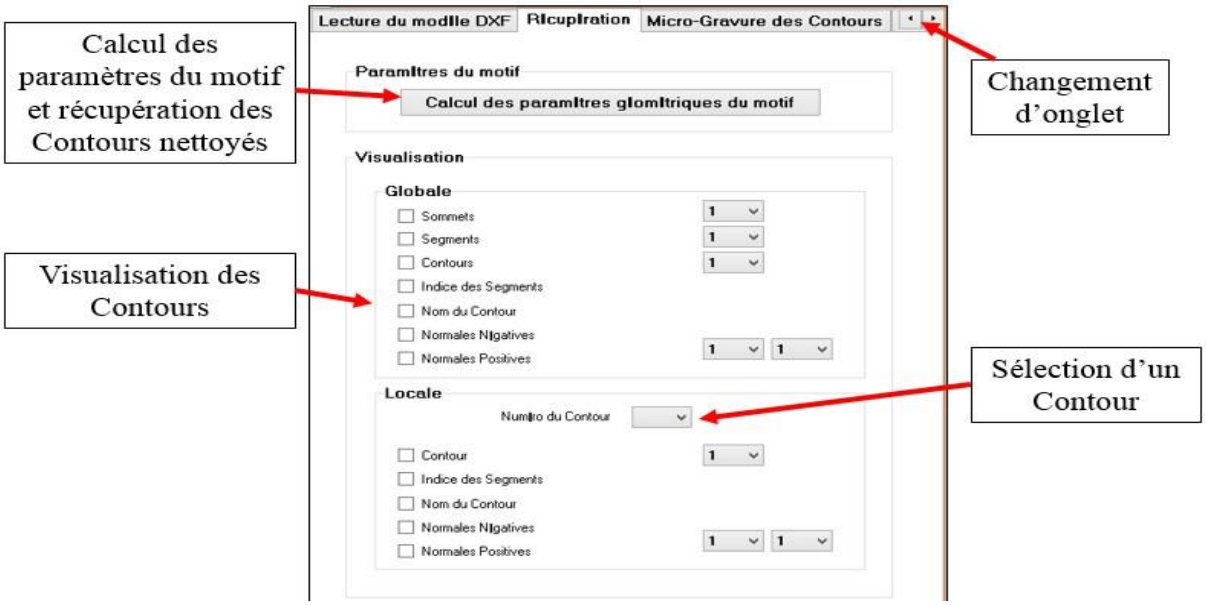


Figure 4.1. Onglet « Récupération ».

4.3.2. Orientation des Contours

Cet onglet est dédié à la détermination de l'orientation des contours pouvant être « Horaire » ou « Antihoraire » (Figure 4.2). Par un simple clic sur le bouton « Orientation Initial des Contours », le sens initial de tous les contours du motif est déterminé. Ensuite, par un clic sur le bouton "Orientation Finale des Contours", tous les contours de sens « Horaire » sont transformés en contours de sens « Antihoraire ». A ce niveau, nous pouvons :

- Visualiser le sens des contours. La couleur rouge pour le sens « Horaire » et la couleur verte pour le sens « antihoraire ».
- Visualiser les normales des segments des contours.

4.3.3. Hiérarchie des Contours

Cet onglet est divisé en deux parties (Figure 4.3). La première partie sert à tester si un point est à l'intérieur d'un contour sélectionné en déterminant le nombre d'intersection et la position du point. La deuxième partie permet de déterminer, par un clic sur le bouton "Déterminer les fils des contours" pour chaque contour, tous ses contours fils et ses contours fils du premier niveau. Une fois la détermination est effectuée, il est possible de :

- Visualiser tous les contours fils du contour sélectionné.
- Visualiser les contours fils du premier niveau du contour sélectionné.

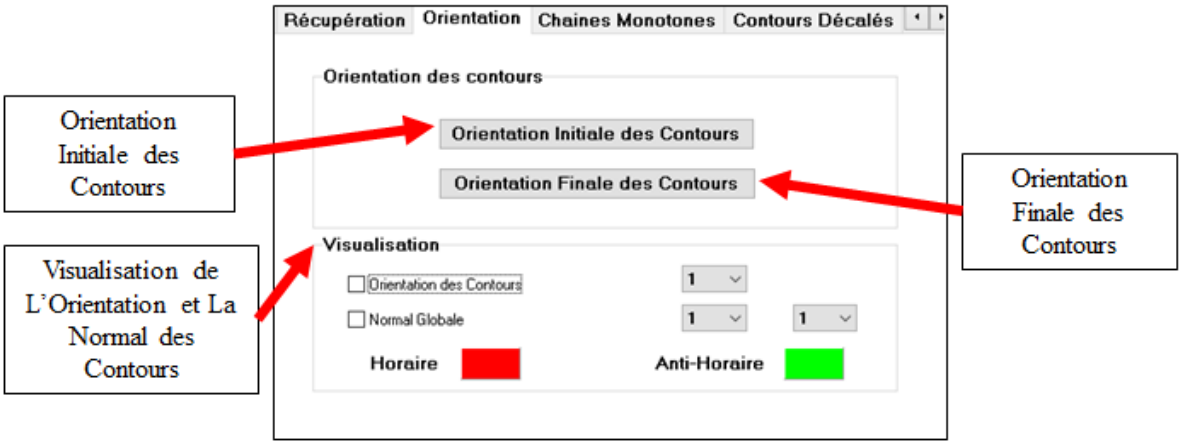


Figure 4.2. Onglet « Orientation ».

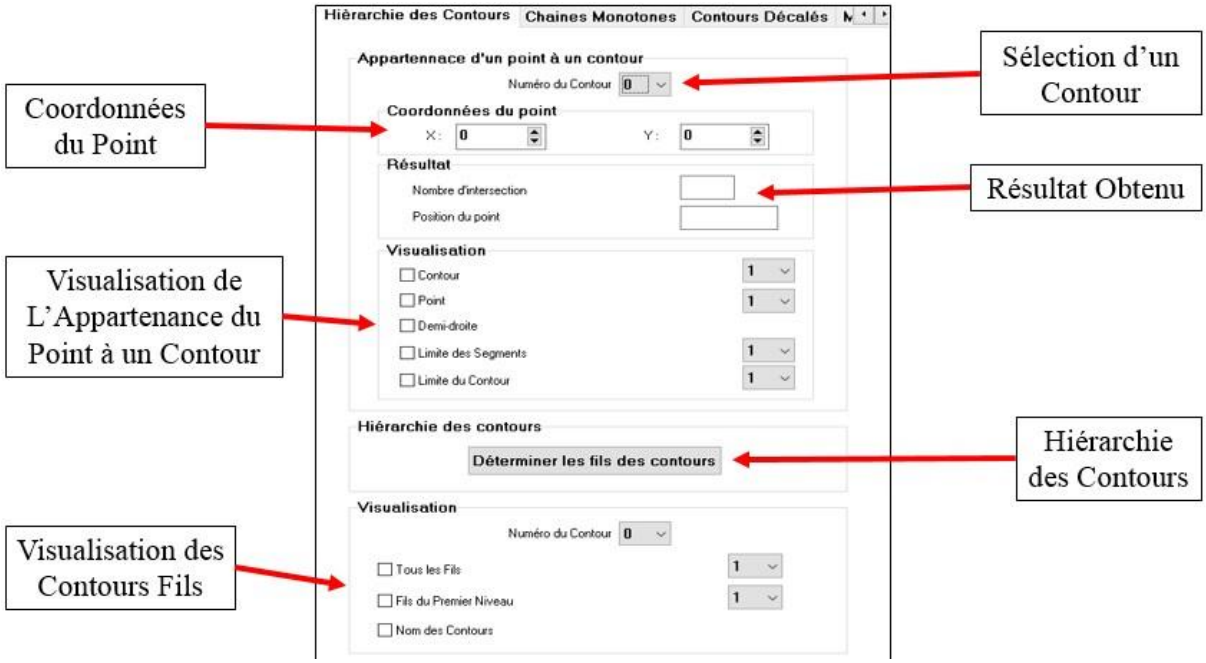


Figure 4.3. Onglet « Hiérarchie des Contours ».

4.3.4. Création des Chaines Monotones

Cet onglet est divisé en deux parties (Figure 4.4) :

- **Création des chaines monotones :** dans cette partie, l'utilisateur peut pour un contour sélectionné, créer les lignes de balayage, récupérer ses points extrêmes par simple clic sur le bouton " Récupérer les points extrêmes ". Par la suite, il peut récupérer les chaines

monotones en cliquant sur le bouton “ Récupérer les chaines monotones ”. En dernier lieu, il peut récupérer les points d’intersection par un clic sur le bouton “ Récupérer les points d’intersection ”. Une fois ces informations obtenues, il est possible de :

- Visualiser les lignes de balayage selon X et Y.
- Visualiser les points extrêmes selon X et Y.
- Visualiser les chaines monotones selon X et Y.
- Visualiser les points d’intersection selon X et Y.

➢ **Détection des boucles :** dans cette partie, l’utilisateur peut déterminer les boucles et leurs sens. Par la suite, il est possible de :

- Visualiser les boucles.
- Visualiser le sens de chaque boucle.

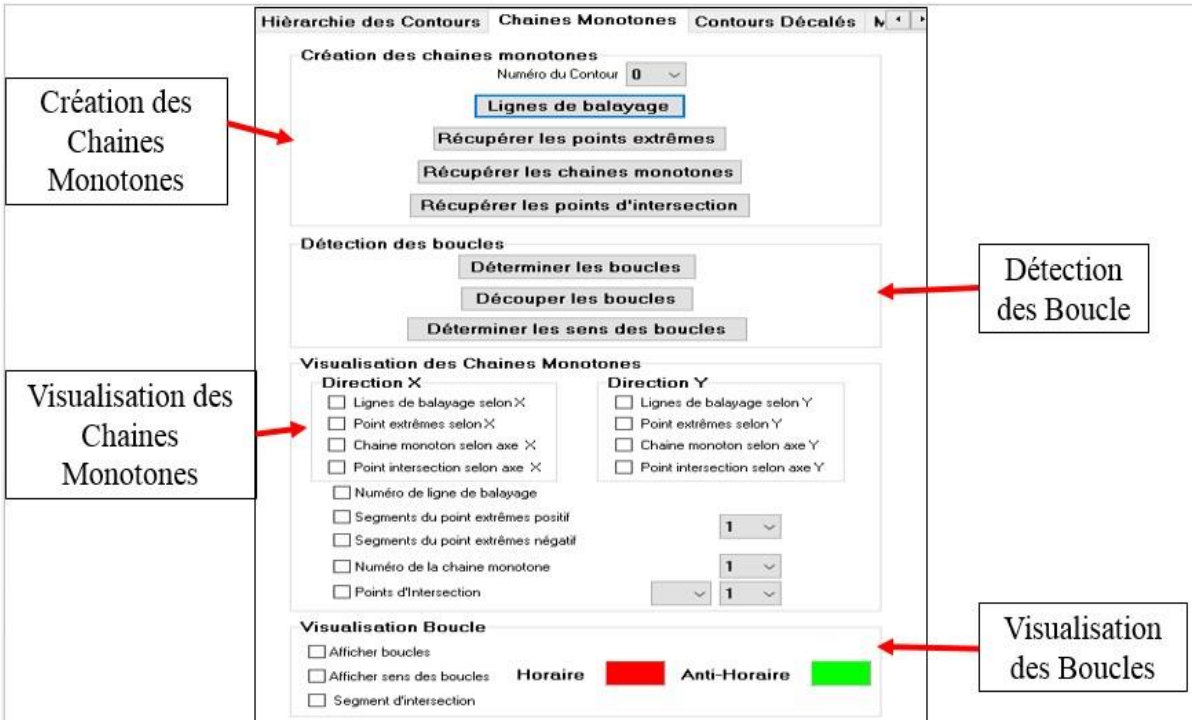


Figure 4.4. Onglet « Chaines Monotones ».

4.3.5. Génération des contours parallèles

L’intérêt principal de cet onglet est la création des contours décalés (Figure 4.5). Au début, l’utilisateur est invité à sélectionner les contours dont il veut remplir en spécifiant :

- Nombre de décalage.
- Distance de décalage.
- Angle de précision.

En cliquant sur le bouton « Créer les Contours Décalés », les offsets sons créés pour le contour sélectionné. Après, suite au clic sur le bouton « Créer les contours décalés des fils », les contours fils du contour sélectionné sont décalés vers l'extérieur. Finalement, le clic sur le bouton « Nettoyer les Contours Décalés » permet de supprimer les contours décalés qui se trouvent à l'intérieur des contours fils. Par la suite, il est possible de :

- Visualiser le temps de calcul.
- Visualiser les offsets.
- Visualiser l'offset de chaque fils.

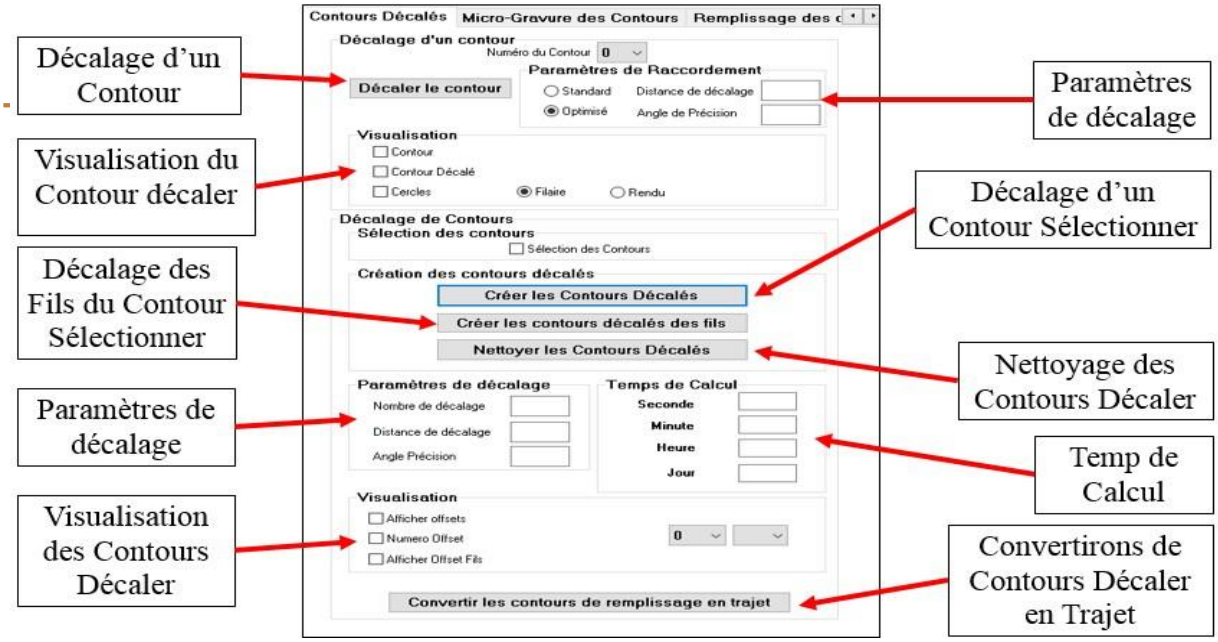


Figure 4.5. Onglet « Contours Décalés ».

A la fin, un clic sur le bouton « Convertir les contours de remplissage en trajet » permet de générer le trajet de remplissage des contours sélectionnés.

4.4. Test et validation

La validation de notre travail passe par la réalisation effective de la micro-gravure de motifs en format DXF sur des échantillons à base du verre sur lequel sont déposées des couches minces en chrome. Pour la phase de validation, plusieurs motifs sont considérés. Tous les résultats sont donnés pour un seul motif représentant une « Guitare » (Figure 4.6). Les paragraphes suivants décrivent le processus de micro-gravure de ce motif.

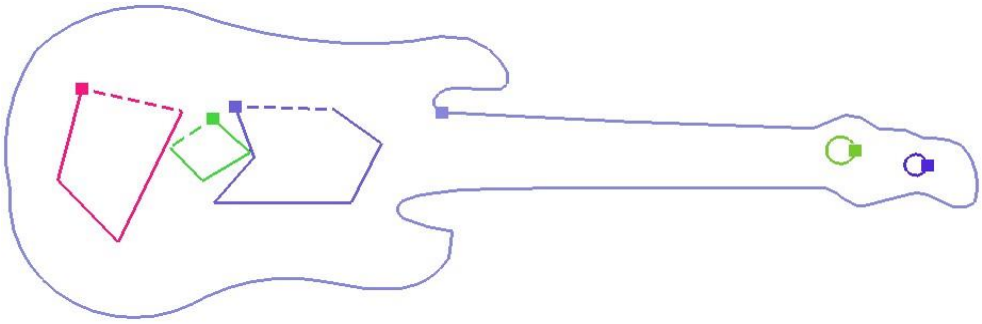


Figure 4.6. Motif pour la validation expérimentale.

➢ **Étape 1 :** récupérer et calculer les paramètres de toutes les entités géométriques (Figure 4.7). Les paramètres de ce motif sont les suivants :

Nombre de contours = 6

Nombre de segments = 357

Longueur = 91.7452 mm

Largeur = 31.4112 mm

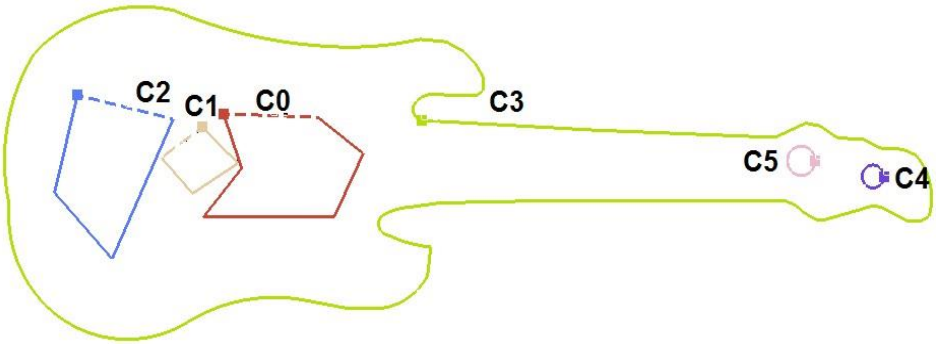
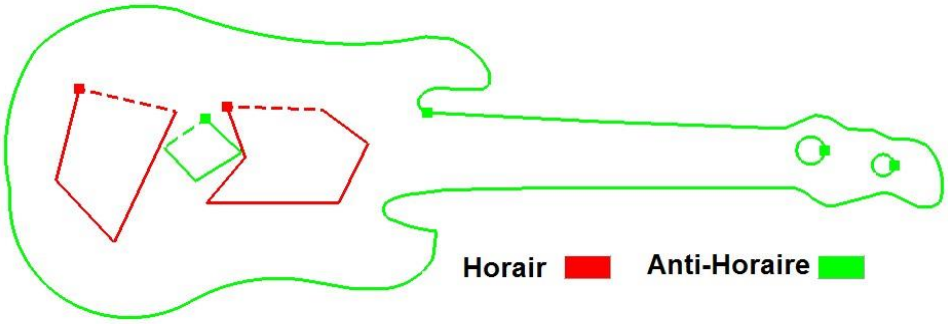
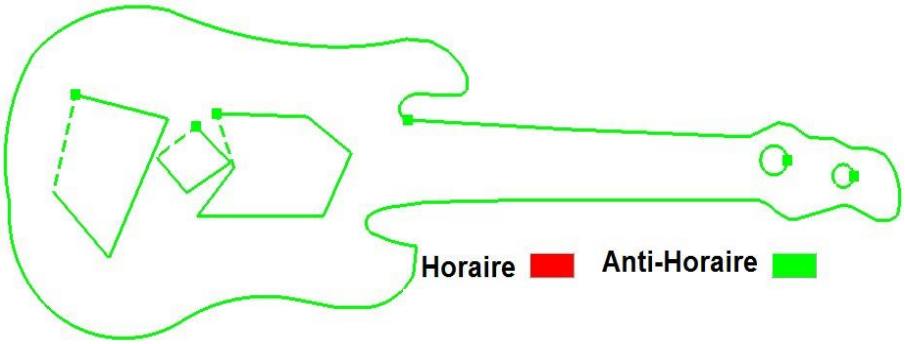


Figure 4.7. Récupération des entités géométriques.

➢ **Étape 2 :** récupérer les orientations initiales des contours (Figure 4.8.a) et rendre tous les contours dans le sens « Antihoraire » (Figure 4.8.b).



a. Orientations initiales.



b. Orientations finales.

Figure 4.8. Orientation des contours.

La Figure 4.9 montre la normale de chaque orientation. Pour le sens « Horaire » la direction de la normale est vers à l'extérieur tandis que pour le sens « Antihoraire » est vers l'intérieur.

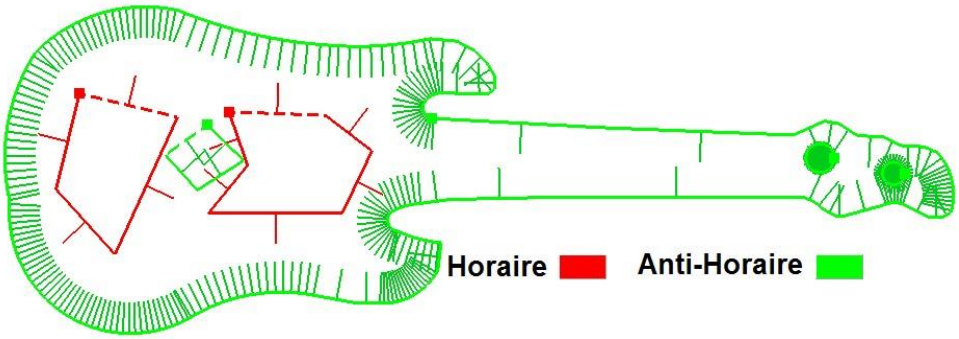


Figure 4.9. Normales des segments.

- **<u>Étape 3 :</u>** déterminer les contours fils du contour sélectionné (Figure 4.10).

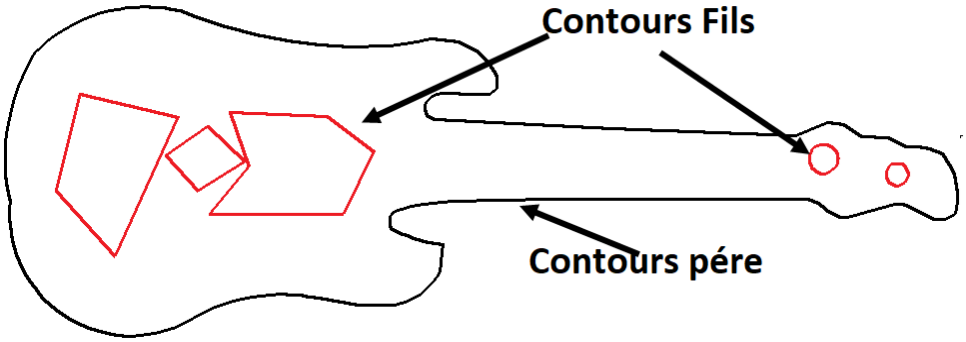


Figure 4.10. Hiérarchie des contours.

- **<u>Étape 4 :</u>** déterminer les boucles valides et les boucles non valides. Pour ce test, nous avons pris un contour décalé comme exemple avec une distance de décalage égale à 2mm et un angle de précision égal à 45°. Deux types de décalages sont possibles :

  - Décalage avec un raccordement standard (Figure 4.11).
  - Décalage avec un raccordement optimisé (Figure 4.12).

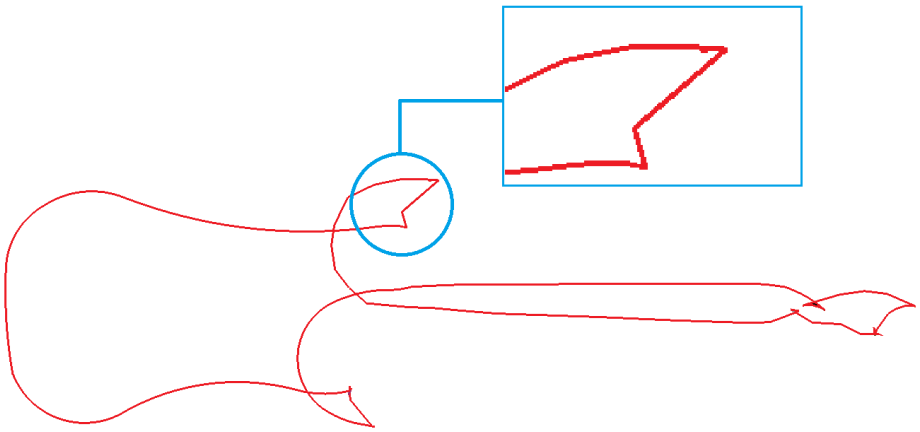


Figure 4.11. Décalage avec un raccordement standard.

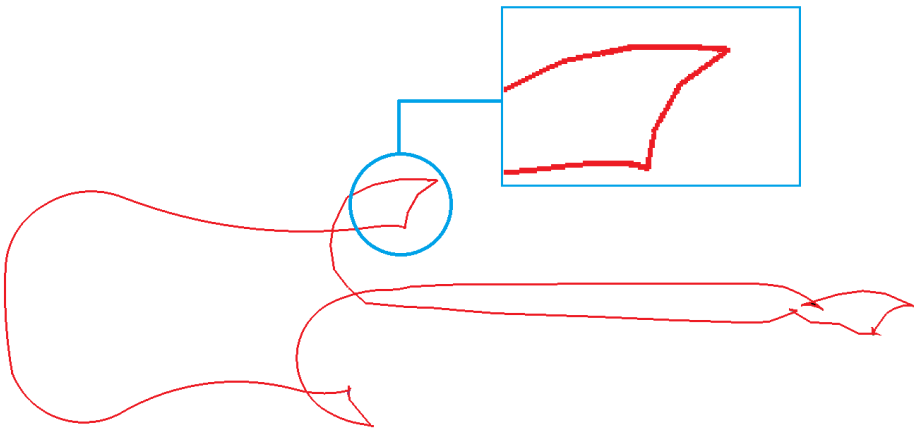
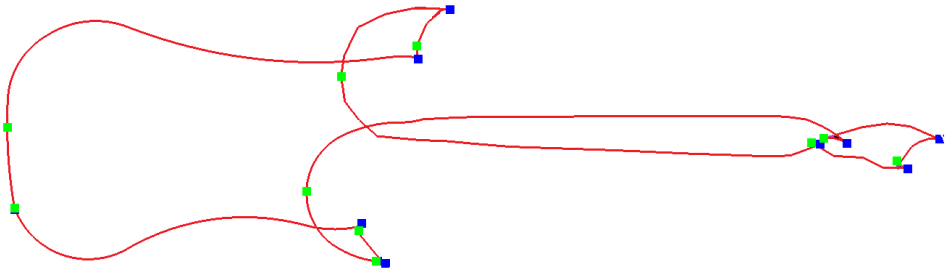


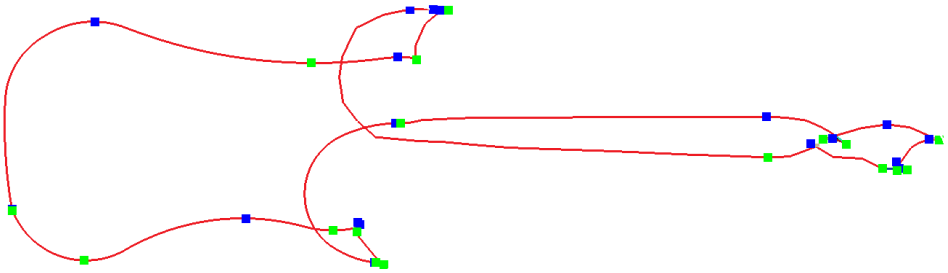
Figure 4.12. Décalage avec un raccordement optimisé.

La détermination des boucles valides passe par les étapes suivantes :

1. Récupérer les points extrêmes positifs et négatifs (Figure 4.13).



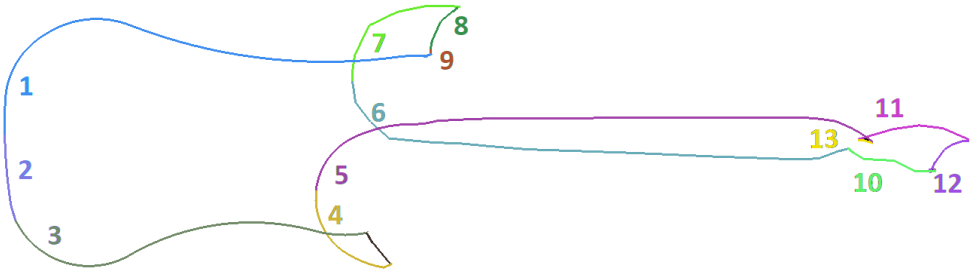
a. Points extrêmes suivant l'axe X.



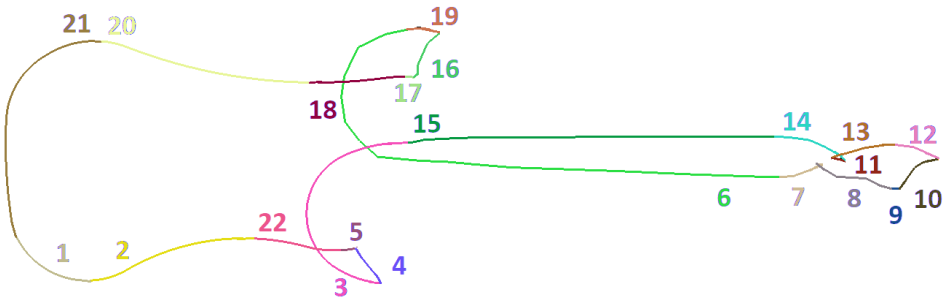
b. Points extrêmes selon l'axe Y.

Figure 4.13. Récupération points extrêmes.

2. Récupérer les chaines monotones. Ces dernières sont récupérées suivant l'axe X (Figure 4.14.a) et suivant l'axe Y (Figure 4.14.b).



a. Chaines monotones suivant l'axe X.



b. Chaines monotones suivant l'axe Y.

Figure 4.14. Récupération chaines monotones.

3. Récupérer les points d'intersections. La récupération dépend du nombre minimal de chaines monotones suivant les axes X et Y (Figure 4.15).

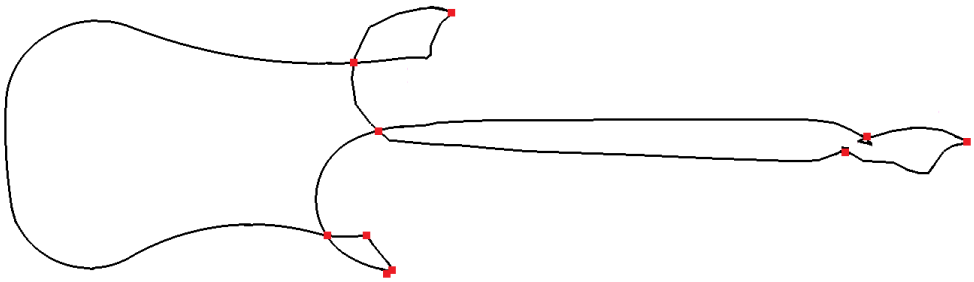
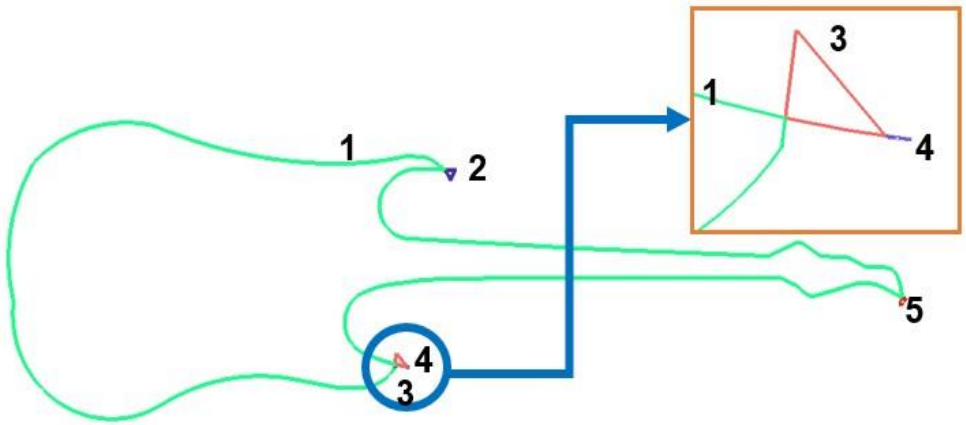
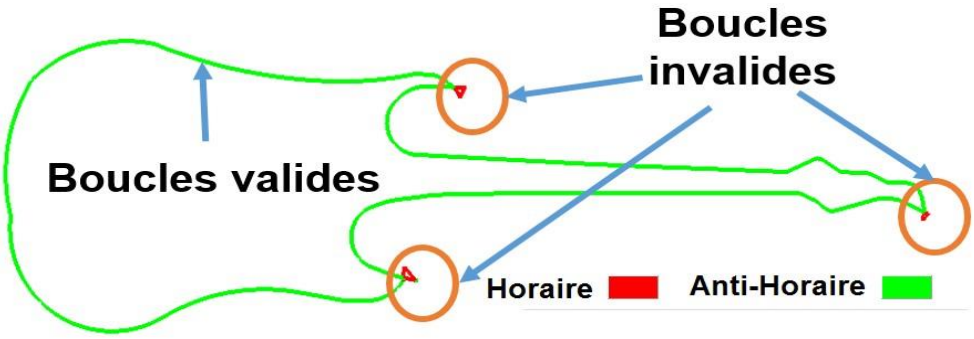


Figure 4.15. Points intersection.

4. Récupération des boucles valides et non valides. Pour ce test, nous avons trouvé cinq (05) boucles où chaque boucle est représentée avec sa propre couleur (le vert pour valide et le rouge pour non valides) (Figure 4.16.a). La Figure 4.16.b représente le sens de chaque boucle.



a. Boucles générés.



b. Sens des boucles.

Figure 4.16. Récupération des boucles.

➢ **<u>Étape 5 :</u>** remplissage du contour sélectionné. Pour ce test, la distance de décalage des contours est prise égale à 0.1mm et la précision est fixée à 45°. La Figure 4.17 montre le remplissage du contour sélectionné sans tenir compte de ses contours fils.

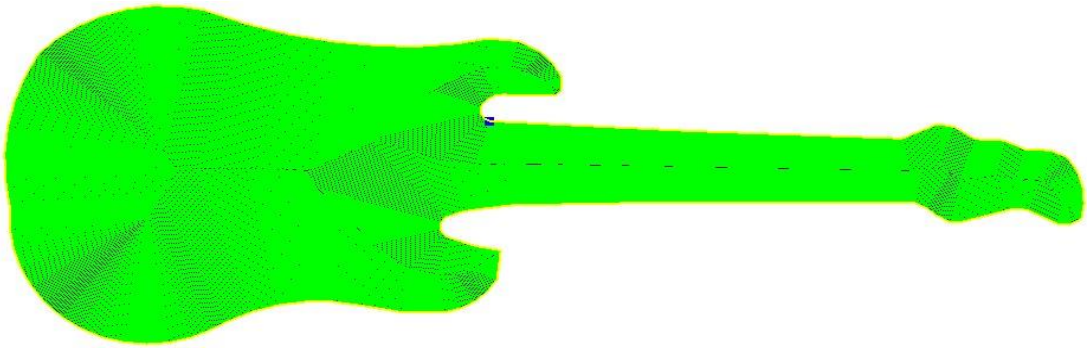
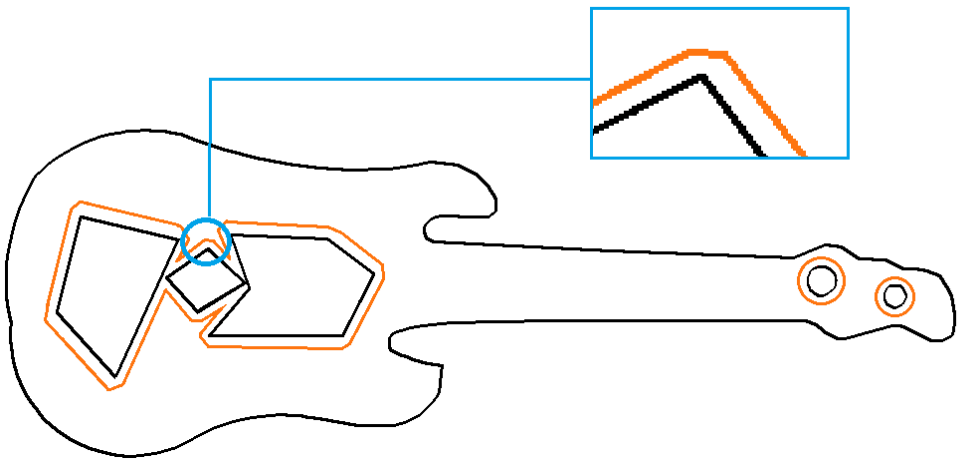
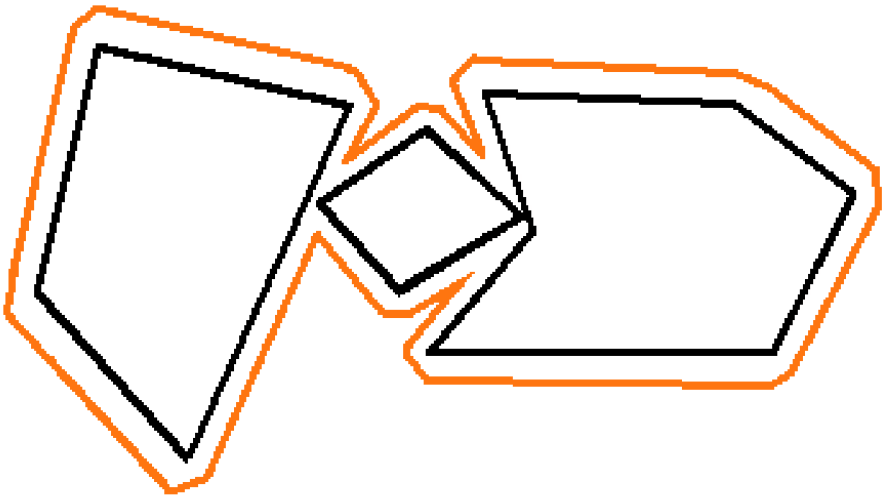


Figure 4.17. Remplissage du contour.

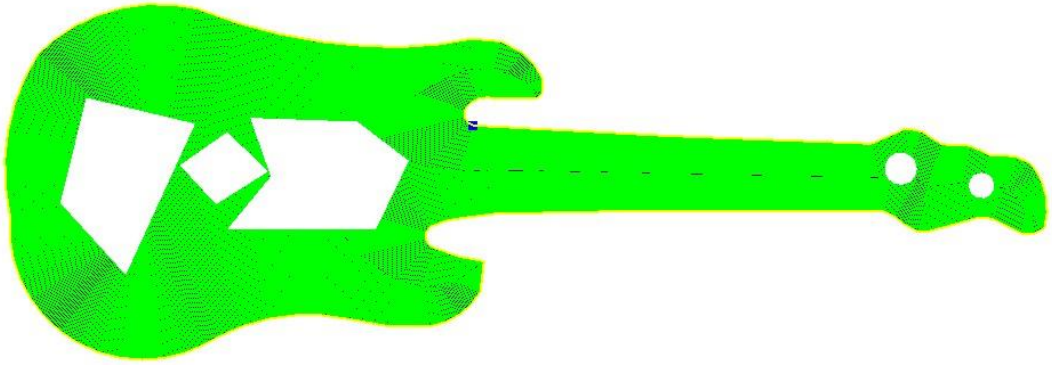
➢ **Étape 6 :** nettoyer les contours décalés. Pour cela, il faut tout d'abord créer pour chaque contour fils son contour décalé ver l'extérieur (Figure 4.18.a). Si les contours décalés s'intersectent entre eux, ils seront combinés. Pour tester le cas, nous avons pris la distance de décalage égale à 0.9mm ver l'extérieur (Figure 4.18.b) .La Figure 4.18.c montre le résultat final après le nettoyage.



a. Contours décalés ver l'extérieur.



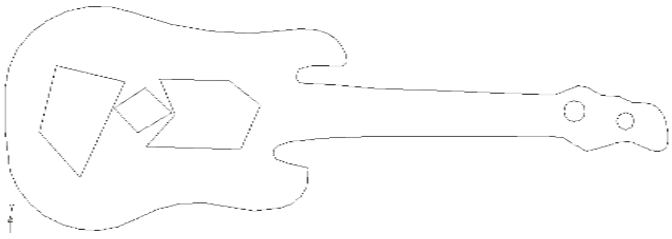
b. Combinaison des contours décalés.

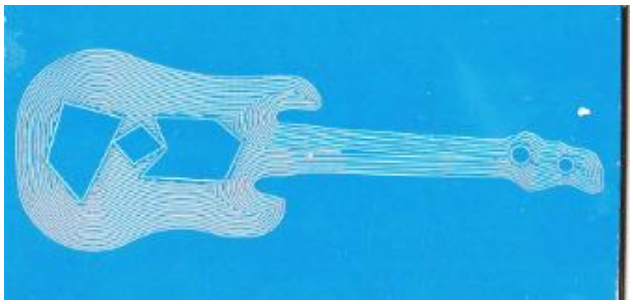


c. Nettoyage des contours.

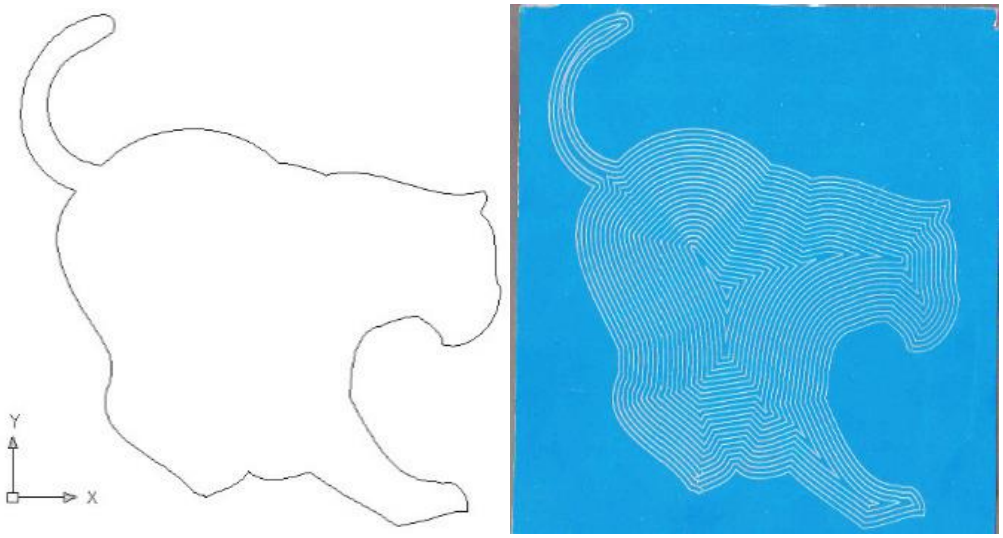
Figure 4.18. Contours décalés finaux.

Nous donnons maintenant les résultats de micro-gravure de quatre (04) autres motifs (Figure 4.19).

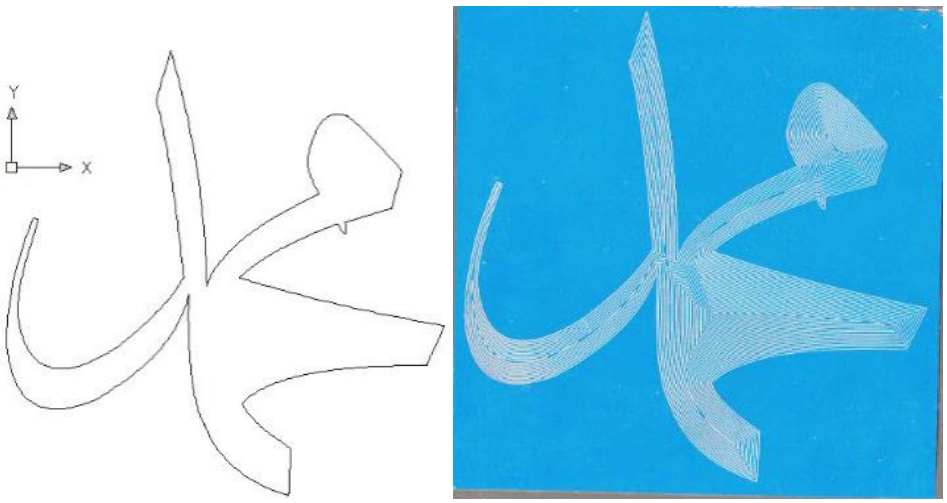




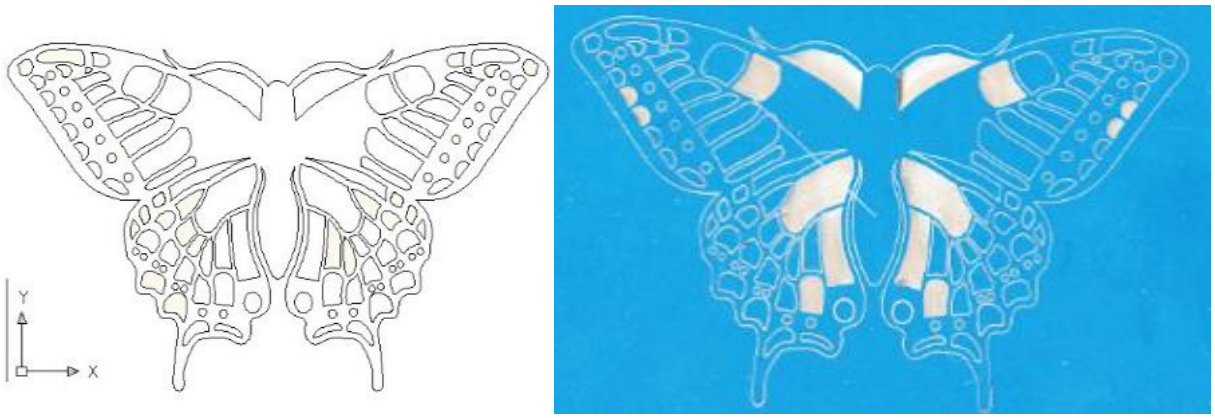
Décalage = 0.5mm, temps = 10 min 41 s



Décalage = 0.2mm, temps = 17 min 31 s



Décalage = 0.5mm, temps = 23 min 51 s



Décalage = 0.02mm, temps = 2 h 41 min 53 s

Figure 4.19. Résultats de micro-gravure des motifs considérés.

4.5. Conclusion

Dans ce chapitre, les différentes étapes de notre application logicielle ont été présentées et le travail réalisé a été validé à travers la micro-gravure d'exemples réels de motifs très complexes. L'application a été testé depuis la lecture du fichier DXF jusqu'à la génération du fichier de la trajectoire.

La validation depuis l'importation du fichier DXF jusqu'au test expérimental du trajet généré confirme la fiabilité de l'approche proposée.

Conclusion Générale et Perspectives

Le travail réalisé au sein de l'équipe « Conception et Fabrication Assistées par Ordinateur » « CFAO » de la Division Productique et Robotique du « CDTA » s'articule autour du développement d'une application logicielle graphique et interactive sous Windows ayant pour but d'automatiser et d'optimiser la micro-gravure par Laser.

Dans ce projet, nous nous sommes intéressés à la génération automatique du programme de pilotage du système de micro-gravure par Laser de motifs 2D sous le format « DXF » en utilisant la stratégie de remplissage des contours « Contours Parallèles ».

Lors de la réalisation de ce projet, nous avons commencé par une étude bibliographique sur le domaine du Laser, ses types, ses applications et les différentes techniques de marquage par Laser. Par la suite, nous avons présenté les différentes méthodes de remplissage. Après, nous avons illustré la stratégie des contours parallèles. A la fin de notre mémoire, nous avons présenté la conception, les algorithmes développés et l'implémentation informatique de notre application logicielle ainsi que des tests de validation sous Windows en utilisant le langage C++, l'environnement de développement Embarcadero et la bibliothèque graphique OpenGL.

Le résultat de l'application développée et l'intégration à l'application logicielle graphique de l'équipe « CFAO » des fonctions qui permettent de :

- ✓ Récupérer les entités géométriques.
- ✓ Déterminer l'orientation des contours.
- ✓ Déterminer la hiérarchie des contours.
- ✓ Détecter les points d'intersection.
- ✓ Déterminer les boucles valides.
- ✓ Générer les contours décalés (offset).
- ✓ Générer le trajet de micro-gravure.
- ✓ Simuler virtuellement le trajet final de micro-gravure.

En perspective de notre travail, nous recommandons de traiter les thématiques suivantes :

- Insertion de nouvelles méthodes de remplissage des contours.
- Application des méthodes d'optimisation pour réduire le temps de micro-gravure.
- Intégration de la possibilité de modifier les motifs et les formes directement sous l'application logicielle.
- Combinaison de différentes techniques de remplissage du même motif.
- Extension de l'approche pour la micro-gravure de motifs 3D.

Bibliographie

[1]. Importation et exportation de fichiers dxf.

https://knowledge.autodesk.com/fr/ support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2018/FRA/AutoCAD-Core/ files/GUID-D4242737-58BB-47A5-9B0E-1E3DE7E7D647-htm.html.

[2]. Caractéristiques d'un faisceau laser.

http://www.a2l-laser.com/index.php?page caracteristiques, 2008.

[3]. Continuous wave laser action on vibrational-rotational transitions of co2. http//link.aps.org/doi/10.1103/PhysRev.136.A1187, 2009.

[4]. Histoire du laser.

http://tpelaser2014-larochefoucauld.e-monsite.com/pages/caractéristique-du-laser/histoire-du-laser.html, 2010.

[5]. Principe et fonctionnement du laser.

http://tpelaser2014-larochefoucauld.e-monsite. com/pages/caracteristique-du-laser/principe-et-fonctionnement-du-laser.html, 2010.

[6]. Marquage laser et gravure laser.

http://www.eslaser.com/applications-lasers/marquage-laser/, 2018.

[7]. Khelili Akram et Akkouche Abderrahmane. *Conception et Développement d'un Logiciel pour le Pilotage Automatique d'un Système de Micro-Gravure par Laser*. Mémoire de Licence, Université Benyoucef BENKHEDDA - Alger1, 2017.

[8]. Marie-Christine Artru. Laser et maser.

http://culturesciencesphysique.ens-lyon.fr/Ressource/laser-maser.xml, 2011.

[9]. Autodesk. *DXF Reference*. Autodesk, Inc.111 McInnis Parkway San Rafael, CA 94903, USA, 2010.

[10]. Jon Louis Bentley and Thomas A Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on computers*, (9):643–647, 1979.

[11]. Giacomo Benvenuti. *Chemical beam deposition of titanium dioxide thin films*. PhD Thesis, Verlag nicht ermittelbar, 2003.

[12]. Philippe Binant. Une petite histoire du laser. Pages 9–13. Commission Supérieure Technique, Paris, 2017.

[13]. Xiaorui Chen and Sara McMains. Polygon offsetting by computing winding numbers. In *ASME 2005 international design engineering technical conferences and computers and information in engineering conference*, pages 565–575. American Society of Mechanical Engineers, 2005.

[14]. Byoung Kyu Choi and Sang C Park. A pair-wise offset algorithm for 2d point- sequence curve. *Computer-Aided Design*, 31(12):735–745, 1999.

[15]. Jin Jacob Chou and E Cohen. Computing offsets and tool paths with voronoi diagrams. In *Technical Report UUCS-89-017*. University of Utah USA, 1989.

[16]. Martin Held. Voronoi diagrams and offset curves of curvilinear polygons. *Computer-Aided Design*, 30(4):287–300, 1998.

[17]. Zhao Jibin, Liu Weijun, and Wang Yuechao. Study on algorithm of contours path generation for robotic prototyping. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 181–186. IEEE, 2004.

[18]. Thomas J Kane and Robert L Byer. Monolithic, unidirectional single-mode nd : Yag ring laser. *Optics letters*, 10(2):65–67, 1985.

[19]. Krati Katyal. *Polygon Fil ling*. Mohanlal Sukhadia University,

[20]. Deok-Soo Kim. Polygon offsetting using a voronoi diagram and two stacks. *Computer-Aided Design*, 30(14):1069–1076, 1998.

[21]. Yuan-Shin Lee and Bahattin Koc. Ellipse-offset approach and inclined zig-zag method for multi-axis roughing of ruled surface pockets. *Computer-Aided Design*, 30(12):957–971, 1998.

[22]. Xu-Zheng Liu, Jun-Hai Yong, Guo-Qin Zheng, and Jia-Guang Sun. An offset algorithm for polyline curves. *Computers in Industry*, 58(3):240–254, 2007.

[23]. T.H. Maiman. Stimulated optical radiation in ruby masers. *Nature vol 187*, 1960.

[24]. Nisha Nisha and Sapna Varshney. A review: Polygon filling algorithms using inside-outside test. International Journal of Advanced Engineering Research and Science, 4(2).

[25]. David Sands. *Diode lasers*. CRC Press, 2004.

[26]. Jerome Swartz, Howard M Shepard, Eric F Barkan, Mark J Krichever, Boris Met- litsky, Edward Barkan, and Alexander M Adelson. Portable laser diode scanning head, July 26 1988. US Patent 4,760,248.

[27]. David Terr and Eric W Weisstein. Concave polygon.

http: //math- world.wolfram.com/ConcavePolygon.html.

[28]. David Terr and Eric W Weisstein. Convex polygon.

http: //math-world.wolfram.com/ConvexPolygon.html.

[29]. Wayne Tiller and Eric G Hanson. Offsets of two-dimensional profiles. *IEEE Computer Graphics and Applications*, 4(9):36–46, 1984.

[30]. V.padovani. *Remplisage*. Equipe Preuves,Programmes et Systemes,, 2010.

[31]. JL Zyskind, V Mizrahi, DJ DiGiovanni, and JW Sulhoff. Short single frequency erbium-doped fibre laser. *Electronics Letters*, 28(15):1385–1387, 1992.

Annexe A

Format d'échange de données DXF

DXF est un acronyme signifiant **D**rawing e**X**change **F**ormat. C'est un format développé par Autodesk pour permettre l'interopérabilité des données entre le logiciel « AutoCAD » et d'autres programmes et de rendre clair (ASCII) [1] le contenu binaire des fichiers « DWG » pour passer d'un logiciel de DAO à un autre. Les fichiers DXF permettent aux utilisateurs d'ouvrir des fichiers en utilisant des programmes et des applications de base d'édition de textes. Les fichiers se caractérisent par des informations de mise en forme comprenant différentes sections et communes à d'autres fichiers connexes. Ces éléments de mise en forme comprennent l'entête (informations de propriété), les classes, les tableaux, les entités, les blocs et les images miniatures. Les données balisées dans le fichier DXF permettent l'identification des méthodes d'interprétation des données. Le format des données balisées signifie également que chaque élément de données est précédé d'un nombre entier appelé code de groupe, indiquant l'ordre des données et la signification des éléments pour certains objets. La structure du fichier DXF [9] est illustrée par la Figure B.1.

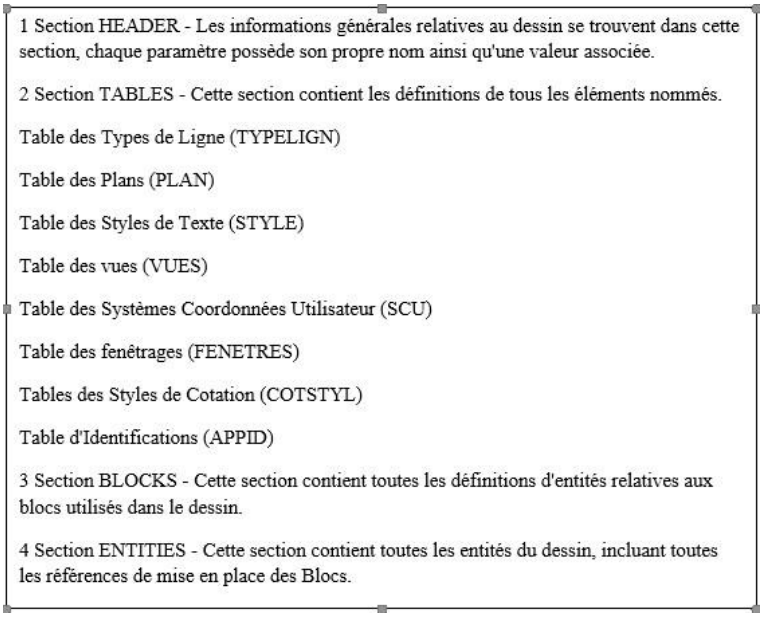


Figure B.1. Structure d'un fichier DXF.

Il est important de noter que dans le fichier « DXF » chaque entité est définie par une structure de données particulière. Donc, la récupération des paramètres des différentes entités doit se faire sur la base de ces structures de données. A titre d'exemple, les paramètres de l'ellipse sont définis par la Figure B.1. Le pseudocode permettant de récupérer les paramètres de l'ellipse est donné par la Figure B.2.

```
ELLIPSE
#
AcDbEntity
 8
0
100      //Marquage de sous-classe « AcDbEllipse »
AcDbEllipse
10       // Valeur X du point du centre
2292.36988089708
20        // Valeur Y du point du centre
1667.663620396206
30        // Valeur Z du point du centre
0.0
11       // Valeur X du point d'arrivée du grand rayon par rapport au centre
326.9184535729491
 21      // Valeur Y du point d'arrivée du grand rayon par rapport au centre
0.0
 31      // Valeur Z du point d'arrivée du grand rayon par rapport au centre
0.0
210      // Valeur X de la direction d'extrusion (facultatif, par défaut = 0, 0,
1)
0.0
220      // Valeur Y de la direction d'extrusion (facultatif, par défaut = 0, 0,
1)
```

Figure B.2. Paramètres de l'ellipse dans le fichier « DXF ».

Annexe B

Diagrammes de séquence

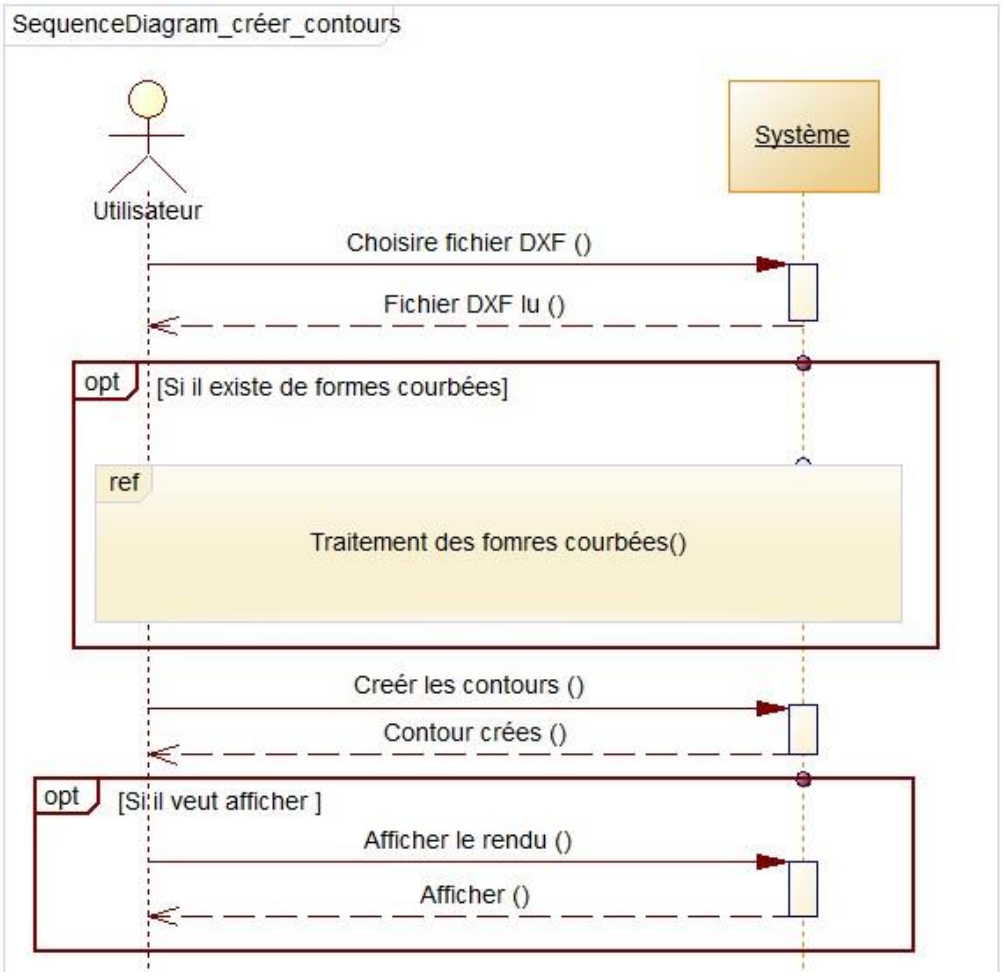


Figure C.1. Diagramme de séquence relatif à la création des contours [7].

La Figure C.2 donne le diagramme de séquence relatif au traitement des formes courbées.

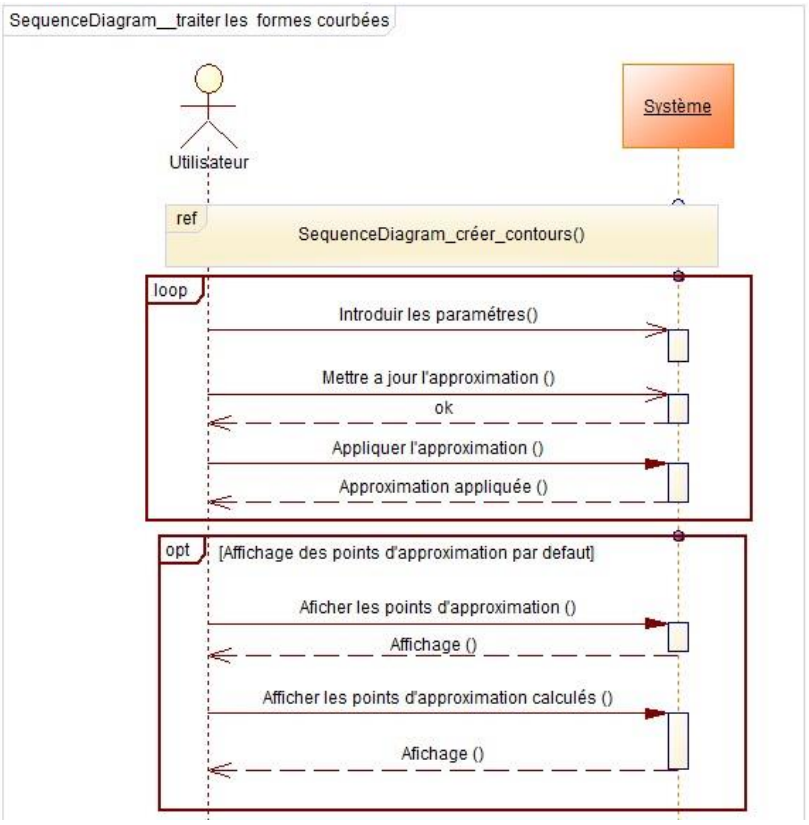


Figure C.2. Diagramme de séquence relatif au traitement des formes courbées [7].

La Figure **C.3** donne le diagramme de séquence relatif à la simulation de trajet.

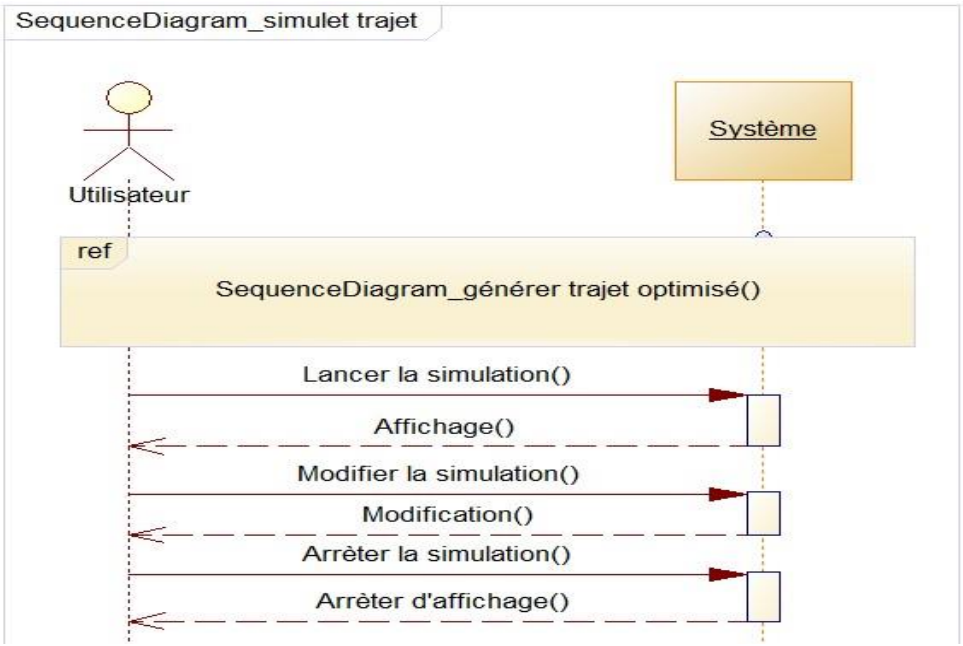


Figure C.3. Diagramme de séquence relatif à la simulation du trajet [7].