UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

IMMOBILISATION DE POLYGONES AVEC LA SERRE *ÉTAU-DOIGTS-PARALLÈLE*

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

JACQUELIN CARON

AOÛT 2008

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Ce mémoire intitulé :

IMMOBILISATION DE POLYGONES AVEC LA SERRE *ÉTAU-DOIGTS-PARALLÈLE*

présenté par

Jacquelin Caron

pour l'obtention du grade de maître ès science (M.Sc.)

a été évalué par un jury composé des personnes suivantes :

Dr. Jurek Czyzowicz ..... Directeur de recherche

Dr. Marek B. Zaremba ..... President du jury

Dr. Alain Charbonneau ..... Membre du jury

Mémoire accepté le : 26 août 2008

Remerciements

À Jurek Czyzowicz pour m'avoir proposé ce sujet et m'avoir aidé à mener à terme ce travail de recherche et de rédaction.

À Marek Zaremba et Alain Charbonneau pour avoir accepté de lire et analyser ce mémoire.

À Émilie Guilbaut et Geneviève Roberge pour la correction du français et la révision du texte.

À mes parents pour m'avoir supporté et encouragé, 25 ans avant même que je commence cette maîtrise.

À mes amis pour tous les moments en leur compagnie.

Au resto-bar *Le Bop* pour m'avoir fourni un environnement de lecture lors de la recherche et de la lecture dirigée.

Table des matières

Liste des figures

Liste des tableaux

Résumé

Le domaine de la saisie d'objet est un champ important de la robotique. Un objet est saisi, ou immobilisé, par la main d'un robot afin d'y effectuer certaines opérations. Les objets peuvent être saisis avec divers types de serres. Parfois, il arrive que certains objets ne puissent pas être immobilisés par une serre en particulier.

D'un point de vue théorique, il est possible de représenter l'immobilisation d'objets avec un modèle géométrique. Ces objets sont modélisés par des polygones simples alors que les composantes des serres servant à l'immobilisation sont représentées par des points (*appelés* doigts) et des droites (*appelées* murs) en contact avec le bord des polygones.

Pour ce mémoire, nous nous sommes intéressés à la serre *Étau-Doigts-Parallèle* qui utilise deux points et un mur pour accomplir ses immobilisations. La principale contrainte de *Étau-Doigts-Parallèle* est que les deux doigts sont situés à distance égale d'un mur. Nous proposons une étude complète de cette serre.

Abstract

An important field of robotics is grasping. A typical question in this domain is how to hold an object with robot fingers to prevent any movement of it while the object is subject to some operations. Many grasping mechanisms may be used to hold an object. Sometimes, the given object cannot be immobilized by a particular grasping mechanism.

From the theoretical viewpoint, a geometric model is often used to study immobilisation. The objects to grasp are represented by simple polygons while the points (*called fingers*) and lines (*called walls*) are used as elements of the grasping mechanism.

In this text, we study the grasping mechanism *Étau-Doigts-Parallèle* (*Parallel-Vise-Fingers*) using two fingers and a wall. The fingers must be equidistant from the wall.

Table des symboles

- m : le contact *mur*
- D, D_1, D_2 : les contacts de type *doigt*
- P : un polygone
- $||P||$: le nombre de sommets dans le polygone P
- C, C_1, C_2, C_i : les côtés d'un polygone
- C_m : le côté de l'enveloppe convexe d'un polygone en contact avec le *mur* m
- n : le nombre de sommets d'un polygone P
- $\breve{n}$: le nombre de sommets concaves d'un polygone P
- e : le nombre de sommets sur l'enveloppe convexe d'un polygone
- p : une pointe (*voir définition 4.13*)
- L : le lieu d'intersection (*voir définition 4.9*)
- I : un intervalle ouvert $]a, b[$ où $a < b$
- K : le nombre de solutions ou de configurations
- $P^{(0)}$: position de référence du polygone P
- $\mathbb{R}$: l'ensemble des nombres réels
- $\mathbb{R}^{0+}$: l'ensemble des nombres réels positifs
- $\mathcal{C}_A$, $\mathcal{C}_B$: des cercles (*voir le lemme 6.10*)
- Ω, Θ, O : mesures théoriques de la complexité d'un algorithme
- Π : classe des polygones dont l'enveloppe convexe ne possède pas de côtés parallèles

Chapitre 1

Introduction

L'ordinateur d'aujourd'hui devient de plus en plus présent dans l'automatisation des procédés de fabrication. Dans ce domaine, une précision des opérations est nécessaire et cruciale pour construire un produit de qualité. Des opérations telles que percer des trous ou faire de la soudure demandent que les pièces soient immobiles.

Par conséquent, le domaine de la manipulation est devenu un champ très important de la robotique. Lorsque les objets sont tous identiques dans une chaîne d'assemblage, le problème d'immobilisation est réglé une seule fois. Par contre, si les objets sont tous de formes différentes, ou si la forme des objets à saisir est inconnue, il est important de pouvoir identifier la façon la plus utile pour les immobiliser afin d'y effectuer les opérations nécessaires. Dans ce cas, pour un processus automatisé, la forme de l'objet est tout d'abord analysée par un ordinateur. Par la suite, on recherche une ou plusieurs solutions pour immobiliser cet objet. Selon les critères d'immobilisation, l'une de ces solutions est choisie et l'objet est saisi.

Tout dépendant du contexte et des besoins, différentes serres sont utilisées pour effectuer les immobilisations. Ces serres sont caractérisées par les types de contact utilisés pour saisir les objets, le nombre de contacts et les différentes contraintes mécaniques. L'une de ces contraintes pourrait être une distance maximale entre deux contacts ou encore que les contacts doivent former une figure géométrique précise. Les caractéristiques intéressantes d'une serre sont sa capacité à immobiliser une grande classe d'objets et la facilité de sa construction.

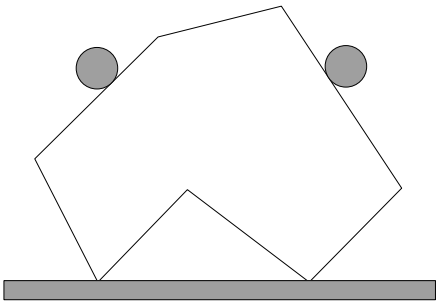


FIG. 1.1: Exemple de la serre *Étau-Doigts-Parallèle*

Dans ce travail, nous nous sommes intéressés à la serre appelée *Étau-Doigts-Parallèle*. Cette serre, constituée de trois contacts, soit un *segment* (appelé *mur*) et deux *doigts*, a comme seule contrainte que les deux *doigts* soient à distance égale du segment. Aucune contrainte n'existe sur la distance entre les deux *doigts* et sur la longueur du *segment*. Sur la figure 1.1, on peut voir un exemple de la serre *Étau-Doigts-Parallèle* immobilisant un polygone.

Nous avons choisi d'étudier cette serre principalement pour les raisons suivantes :

1. *Étau-Doigts-Parallèle* a trois degrés de liberté, soit la distance entre les deux doigts, la distance entre ces doigts et le mur et la dimension des doigts. La contrainte de la distance entre les doigts et le mur limite le nombre de possibilités pour pouvoir immobiliser certains polygones. Il est donc intéressant de connaître les limites de cette serre.
2. *Étau-Doigts-Parallèle* peut facilement être construite. Les deux premiers degrés de liberté peuvent être contrôlés à l'aide de deux vis semblables à celles que l'on retrouve sur un étau.

Pour la première partie de ce travail, le chapitre 2, nous proposons une revue des notions géométriques utilisées par la suite. Le chapitre 3 présente les différents résultats de recherche obtenus dans le domaine de l'immobilisation. Par la suite, dans le chapitre 4, on retrouve les notions spécifiques aux immobilisations.

Finalement, on retrouve aux chapitres 5 et 6 les résultats que nous avons obtenus dans notre recherche. Le premier de ces chapitres est consacré à l'étude d'une classe de polygones possédant au moins une configuration d'immobilisation alors que le second présente des algorithmes énumérant toutes les configurations d'immobilisation existantes pour un polygone donné.

Chapitre 2

Les prérequis géométriques

Nous consacrons ce chapitre à la définition les éléments mathématiques et géométriques utilisés tout au long de ce travail.

Définition 2.1. *Soit d une droite sur le plan et $O, F \in d$. Un* segment *S, dénoté $S = \overline{OF}$, de la droite d est l'ensemble de points de la droite d compris entre les points O et F de d appelée droite de segment. O et F sont les extrémités du segment.*

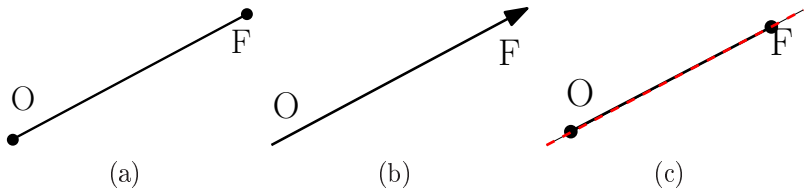


FIG. 2.1: (a) Un segment, avec l'origine O et fin F (b) L'orientation du segment vu en a (c) La droite du segment vu en a

Définition 2.2. *Un* segment orienté, *dénoté par $\overrightarrow{OF}$, avec l'origine O et la fin F est un segment OF avec un ordre (une orientation) fixé entre ses extrémités. La droite d'un segment orienté s'appelle la* droite orientée.

L'équation paramétrique d'un segment possédant comme extrémités les points O et F est : $O + t(\overrightarrow{OF})$ où $t \in [0, 1]$. L'équation d'une droite de segment est la même que celle du segment à l'exception que $t \in \mathbb{R}$.

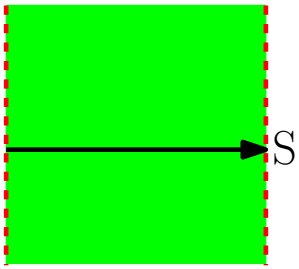


FIG. 2.2: La bande retournée par la fonction $\mathcal{B}$

Définition 2.3. *La fonction distanceDroitePoint(d, p) retourne la distance entre le point p et la droite d.*

Définition 2.4. *La fonction $\mathcal{B}(S)$ retourne une bande ouverte délimitée par des droites perpendiculaires passant par les extrémités de S (voir la fig. 2.2). Par $\overline{\mathcal{B}}(S)$, on dénote la fermeture de la région $\mathcal{B}(S)$ (la région incluant son bord).*

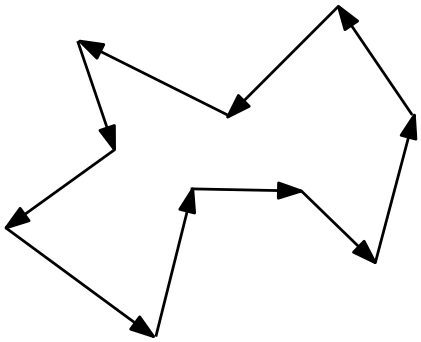


FIG. 2.3: La suite des segments déterminant un polygone

Définition 2.5. *Un polygone est une région fermée délimitée par une suite de n segments (voir la fig. 2.3), appelés côtés, où :*

- *La fin du i^e segment coïncide avec l'origine du $(i+1)^e$ segment en un point appelé* sommet du polygone
- *La fin du dernier segment coïncide avec l'origine du premier segment*
- *Aucune paire de segments ne se croise, à l'exception des paires de segments consécutifs partageant leurs extrémités*
- *L'orientation de la suite de segments est anti-horaire*

Dans les algorithmes de ce texte, un polygone est représenté par un tableau contenant les coordonnées cartésiennes de ses n sommets.

La suite des côtés d'un polygone P, appelée *bord* du polygone $bord(P)$, forme une courbe fermée qui partage le plan en deux régions ouvertes. La première région est bornée et elle s'appelle l'*intérieur* du polygone $int(P)$ et la seconde, non-bornée, l'*extérieur* du polygone $ext(P)$. Pour deux points $A \in int(P)$ et $B \in ext(P)$, chaque courbe qui joint A et B croise $bord(P)$. Pour chaque paire de points qui sont à la fois à l'intérieur ou à l'extérieur du polygone P, il existe une courbe qui les joint sans croiser $bord(P)$.

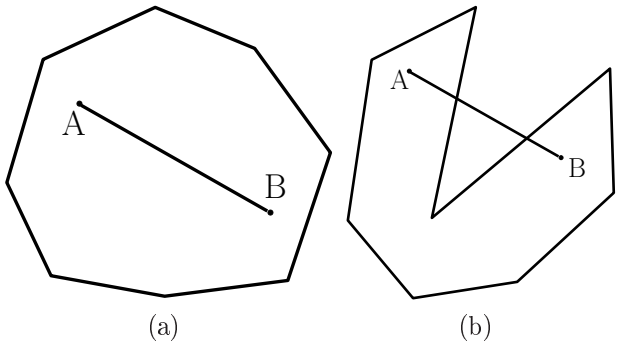


FIG. 2.4: (a) Un polygone convexe (b) Un polygone non convexe

Définition 2.6. *Un polygone P est dit* convexe *si pour chaque paire de points $A, B \in int(P)$ le segment $\overline{AB} \subset int(P)$ (voir la fig. 2.4a). Sinon, on dit que le polygone est* non convexe *(voir la fig. 2.4b).*

Définition 2.7. *On nomme* polygone arbitraire *celui qui peut être convexe ou non convexe.*

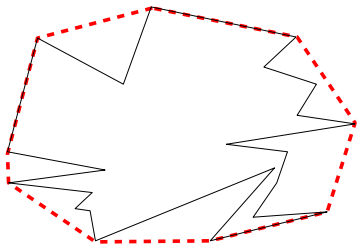


FIG. 2.5: Un polygone et son enveloppe convexe

Définition 2.8. *L'*enveloppe convexe *d'un polygone P, dénotée $env(P)$, est le plus petit polygone convexe contenant P (voir la fig. 2.5).*

Définition 2.9. *La fonction* envConvexe*(P) retourne l'enveloppe convexe du polygone P. Cette fonction s'exécute en temp* $\Theta(n)$*, voir* [4, 5].

Définition 2.10. *Trois vecteurs* $\overrightarrow{v_1}$*,* $\overrightarrow{v_2}$ *et* $\overrightarrow{v_3}$ *sur le plan* complètent $\mathbb{R}^2$ *si pour chaque vecteur* $\overrightarrow{v}$ *il existe* λ_1*,* λ_2 *et* $\lambda_3 \in \mathbb{R}^{0+}$ *tels que* $\overrightarrow{v} = \lambda_1\overrightarrow{v_1} + \lambda_2\overrightarrow{v_2} + \lambda_3\overrightarrow{v_3}$*, voir* [13].

Lemme 2.11. *Les trois vecteurs de trois demi-droites orthogonales aux côtés d'un triangle allant vers l'intérieur complètent* $\mathbb{R}^2$ *(voir fig. 2.6).*

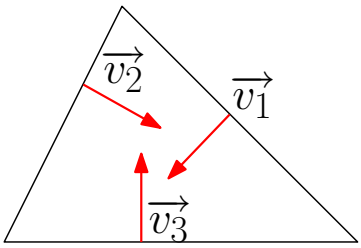


FIG. 2.6: Le triangle du lemme 2.11

Preuve :

Il faut ramener les trois vecteurs en une origine commune (*voir fig. 2.7a*). Les angles α, β et γ entre chaque paire de vecteurs consécutifs sont tous inférieurs à 180^o. Notons qu'un vecteur arbitraire $\overrightarrow{v}$ peut-être obtenu comme une combinaison linéaire avec les constantes non-négatives de deux vecteurs parmi $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$ entre lesquels il se situe (*voir fig. 2.7b*).

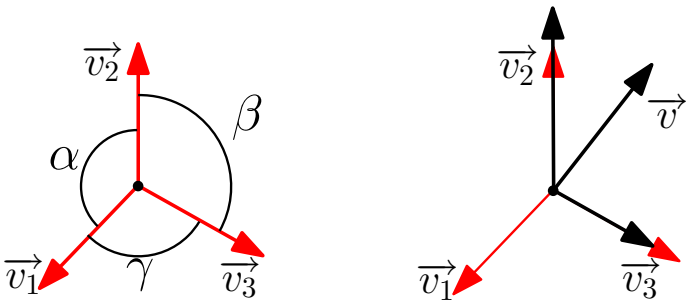


FIG. 2.7: (a) Les 3 vecteurs ramenés en une origine commune (b) Un 4^e vecteur créé

Chapitre 3

État de l'art

En 1876, Reuleaux [14] introduit le concept de *fermeture de force* (ou fermeture de force du premier ordre). Ce concept permet de vérifier si un objet est immobilisé correctement à l'aide d'un ensemble de points, appelés *doigts*, positionnés sur son périmètre. Un objet est en fermeture de force si pour n'importe quelle force ou torsion externe appliquée sur l'objet, les doigts peuvent exercer certaines forces appliquées aux points de contact perpendiculairement à la frontière de l'objet pour que celui-ci reste immobilisé.

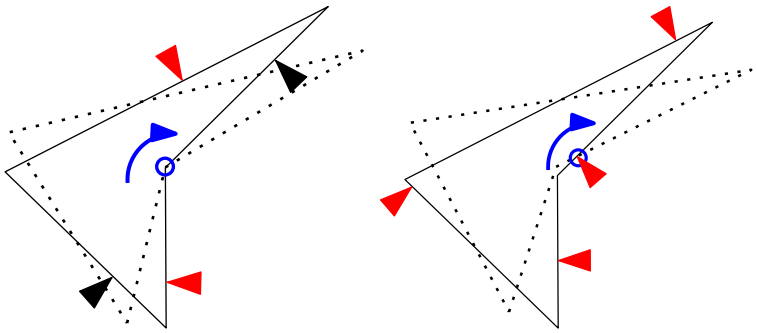


FIG. 3.1: (a) Un objet en fermeture de forme (b) Un objet pouvant bouger

Dans un contexte géométrique, ce ne sont pas les forces et la friction qui sont prises en compte mais plutôt la possibilité des déplacements de l'objet. Le concept d'*immobilisation en fermeture de forme* (ou immobilisation du premier ordre), présenté par Reuleaux [14], propose des règles permettant de positionner les doigts de façon à ce que tout mouvement soit bloqué (*voir fig. 3.1a*). Puisque tout mouvement peut être localement approximé

par une rotation (la translation est considérée comme une rotation avec son centre de rotation à l'infini) la fermeture de forme consiste à rendre impossible la présence de tout centre de rotation sur le plan. Autrement dit, pour chaque rotation de l'objet au moins un doigt doit localement pénétrer à l'intérieur de celui-ci.

Rimon et Burdick [9] ont prouvé que l'objet est en fermeture de forme du premier ordre si et seulement s'il est en fermeture de force du premier ordre.

Markenscoff et al. [6], ainsi que Mishra et al. [7], ont démontré que quatre doigts peuvent être positionnés sur le bord de n'importe quel objet sur le plan en le plaçant en fermeture de forme (à l'exception du cercle où aucun nombre de doigts n'est suffisant pour l'immobiliser). Cette condition permet une certaine liberté pour placer les doigts et ce, sans enfreindre les conditions d'immobilisation. En conséquence, pour n'importe quel objet en fermeture de forme du premier ordre, il existe une valeur $\epsilon > 0$, telle que si les doigts sont déplacés sur leurs côtés respectifs d'une distance inférieure à ϵ, l'objet demeure en fermeture de forme. C'est pourquoi les immobilisations en fermeture de forme sont considérées comme étant robustes.

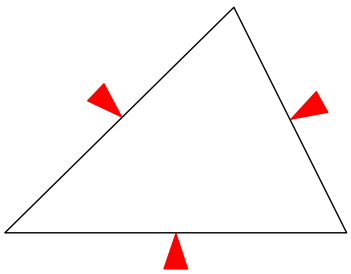


FIG. 3.2: Un triangle immobilisé mais pas en fermeture de forme

Si l'on prend un triangle et qu'il y a un doigt placé au milieu de chacun de ses côtés, celui-ci est immobilisé (*voir fig. 3.2*). Mais il n'est pas dans la fermeture de forme du premier ordre. L'un des problèmes avec les immobilisations en fermeture de forme de premier ordre est que la courbure des objets n'est pas prise en compte. Le problème est que la condition d'immobilisation pour le premier ordre est suffisante mais pas nécessaire. Czyzowicz et al. [2, 3] ont donné les conditions nécessaires et suffisantes pour qu'un objet sur le plan soit immobilisé. Ces conditions ont été étudiées par Rimon et Burdick [12, 10, 11] et ils les ont nommées *immobilisations du deuxième ordre.*

Supposons que la courbure de l'objet au point de placement d'un doigt est différente de zéro et considérons que le centre de rotation est placé sur la droite orthogonale au

bord de l'objet au point de ce doigt. Le fait que le doigt pénètre ou non à l'intérieur de l'objet dépend de la distance du centre de rotation. Prenons l'exemple de la figure 3.3. Pour étudier une position du polygone par rapport à un doigt immobile on peut, par symétrie, considérer inversement la rotation du doigt en gardant le polygone immobile. Traçons un arc de trajectoire de ce doigt. Si le rayon de cet arc est plus petit que le rayon de courbure du bord de l'objet, le doigt entre dans l'objet (*voir fig. 3.3a*). Dans le cas contraire, le doigt ne rencontre pas l'objet (*voir fig. 3.3b*).

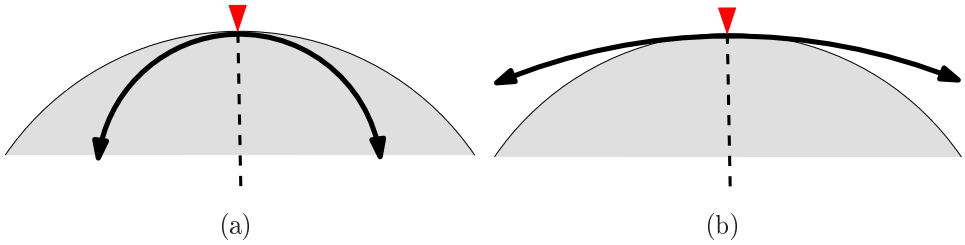


FIG. 3.3: (a) Le doigt entre dans l'objet (b) Le doigt n'offre aucune résistance

Pour qu'une immobilisation du deuxième ordre soit réussie, il faut que pour n'importe quel centre de rotation horaire ou antihoraire un doigt entre dans l'objet. Sur la figure 3.4a, l'objet n'est pas immobilisé puisqu'il existe un centre de rotation où aucun doigt ne s'oppose au mouvement de celui-ci. Par contre, avec le même objet, si les doigts sont placés au milieu des côtés, les doigts empêchent toute rotation autour du centre (*voir fig. 3.4b*).

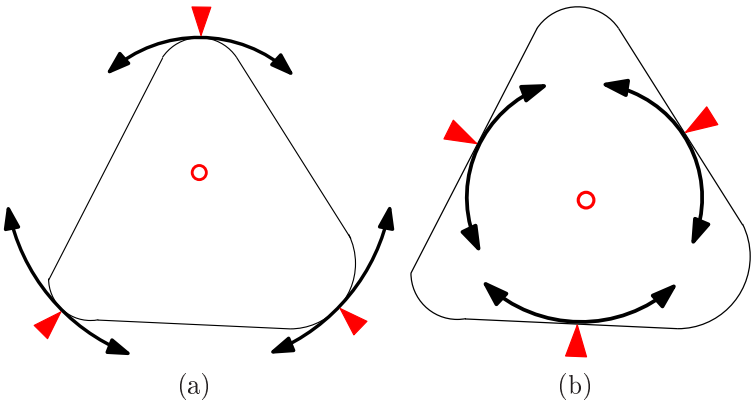


FIG. 3.4: Un objet (a) non immobilisé et (b) dans une immobilisation du 2^e ordre

Pour que les conditions d'immobilisation du deuxième ordre avec trois doigts en contact avec les côtés d'un polygone soient remplies, il faut que les doigts soient po-

sitionnés exactement. Alors, contrairement à l'immobilisation du premier ordre, un léger décalage de l'un des doigts fait qu'une rotation devient possible.

Les recherches concernant les immobilisations consistent principalement à identifier des configurations de doigts immobilisant des polygones à n côtés. Ces configurations sont trouvées par des algorithmes analysant le polygone fourni en entrée. Il existe deux classes d'algorithmes, soit ceux qui recherchent une seule configuration [2, 8] et ceux qui énumèrent toutes les configurations existantes [1, 13].

Comme résultat principal pour les immobilisations du premier ordre, van der Stappen et al. [13] ont construit un algorithme listant toutes les configurations d'immobilisation. Une configuration, dans le cas présent, est un quadruplet de côtés du polygone sur lesquels les doigts sont en contact et leur position est spécifiée sur chacun de ces côtés. L'algorithme proposé s'exécute en temps $O(n^{2+\epsilon} + K)$ où ϵ est une constante quelconque supérieure à zéro et K, le nombre de toutes les configurations.

Cheong et al. [1] ont étudié les polygones possédant des sommets concaves puisqu'un doigt placé dans un tel sommet donne l'effet de deux doigts. Un premier algorithme, s'exécutant en temps $O(\breve{n}^{\frac{4}{3}} \log^{\frac{1}{3}} \breve{n} + K)$, où $\breve{n}$ est le nombre de sommets concaves, identifie les K paires de sommets concaves plaçant le polygone en immobilisation du premier ordre. Le second algorithme recherche les K triplets de sommet concave et de deux côtés immobilisant un polygone. Celui-ci s'exécute en temps $O(n^2 \log^4 \breve{n} + K)$.

En ce qui concerne les immobilisations du deuxième ordre, Czyzowicz et al. [2] ont présenté un algorithme identifiant une configuration d'immobilisation (ici, un triplet de côtés). Cet algorithme, fonctionnant sur les polygones convexes sans côtés parallèles, trouve le plus grand cercle inscrit pour identifier l'emplacement des doigts.

Van der Stappen et al. [13] se sont intéressés à la recherche de toutes les K configurations immobilisant un polygone pour le deuxième ordre. Ils ont proposé un algorithme fonctionnant en temps $O(n^2 \log^2 n + K)$. Cheong et al. [1] ont également étudié les polygones possédant $\breve{n}$ sommets concaves pour les immobilisations du deuxième ordre. Les K paires immobilisant le polygone, dont la première composante est un sommet concave et la seconde un côté, peuvent être trouvées en utilisant leur algorithme s'exécutant en temps $\mathrm{O}(n \log^4 \breve{n} + (\breve{n}n)^{\frac{2}{3}} \log^{2+\delta} \breve{n} + K)$ où δ est une constante quelconque supérieure à zéro.

Finalement, certaines alternatives ont été explorées dans le but d'étudier d'autres types de contact pour effectuer une immobilisation. Overmars et al. [8] ont analysé un

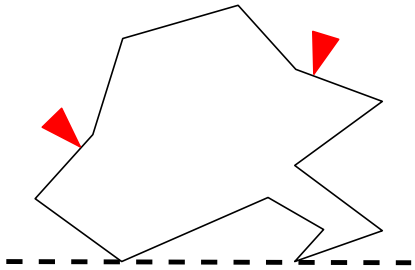


FIG. 3.5: Une immobilisation avec un mur et deux doigts

cas spécial de la fermeture de forme où deux doigts sont remplacés par une droite appelée *mur* (*voir fig. 3.5*). Ils ont aussi présenté les conditions d'immobilisation pour cette serre ainsi qu'un algorithme pour trouver une configuration d'immobilisation. Overmars et al. [13] ont proposé un algorithme, s'exécutant en temps $O(n^2 \log^2 n + K)$, listant les K configurations d'immobilisation existantes.

Chapitre 4

Les immobilisations

Ce chapitre présente dans les deux premières sections différentes notions de l'immobilisation de polygone. Par la suite, nous introduisons formellement la serre *Étau-Doigts-Parallèle* accompagnée de concepts utilisés pour démontrer nos résultats.

4.1 Les contacts

Nous présentons ici les deux types de contact (utilisés dans ce travail), d'un point de vue géométrique, ayant pour but d'immobiliser un polygone P.

Définition 4.1. *On appelle* doigt *un point $D \in bord(P)$ (voir fig. 4.1a).*

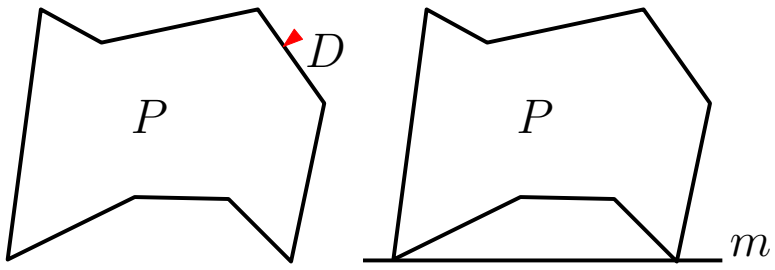


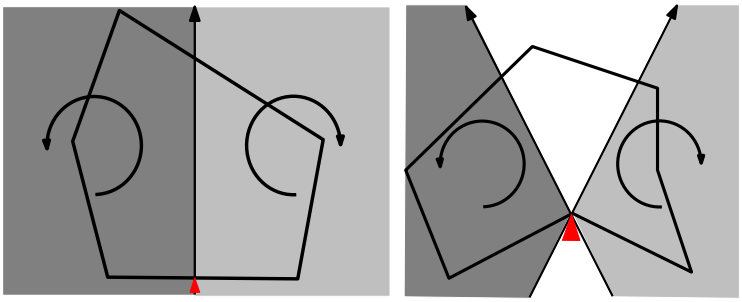
FIG. 4.1: Un polygone P (a) avec un doigt D et (b) avec un mur m

Définition 4.2. *On appelle* mur *la droite d'un segment $S \in bord(env(P))$ (voir fig. 4.1b).*

Un contact, lorsqu'il est employé seul, ne peut pas faire une immobilisation. Par contre, il peut empêcher certaines rotations.

Au point de contact d'un doigt avec le côté d'un polygone, on retrouve une droite orientée perpendiculaire au côté, orientée vers l'intérieur du polygone, appelée *normale*.

La normale au doigt sépare le plan en trois parties par rapport aux centres de rotations qui ne causent pas la pénétration de ce doigt (*voir la fig. 4.2a*) : (1) dans le demi-plan à gauche de cette normale on retrouve les centres de rotation permettant seulement des rotations positives, (2) dans le demi-plan à droite de la normale, ce sont les rotations négatives, et (3) sur la normale elle-même, les deux sens de rotations sont possibles. Ce dernier point s'applique seulement aux immobilisations du premier ordre. Pour les immobilisations du deuxième ordre, il n'y a aucun centre de rotation situé sur la portion de la normale à partir du point de contact vers l'intérieur du polygone.



(a) (b)

FIG. 4.2: Les effets des différents contacts : (a) un doigt sur un côté, (b) un doigt dans un sommet concave.

Dans le cas d'un doigt positionné dans un sommet concave, il y a deux normales, soit une par côté adjacent à ce sommet (*voir fig. 4.2b*). C'est pour cette raison qu'un doigt dans un sommet concave a l'effet de deux doigts. Avec deux doigts, ou l'effet de deux doigts, pour qu'une rotation dans un sens soit possible, il faut qu'elle soit possible pour les deux doigts à la fois.

Pour l'effet du mur, il faut prendre en compte qu'il y a deux normales, soit une à chaque extrémité du côté en contact avec le mur (*voir fig. 4.3*). Entre ces deux normales, aucune rotation n'est possible. Ces deux normales sont aussi orientées vers l'intérieur du polygone.

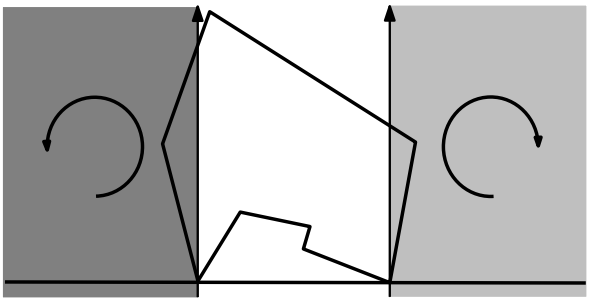


FIG. 4.3: L'effet d'un mur sur un côté

4.2 Le mouvement

Pour décrire formellement la position quelconque d'un polygone sur le plan Cartésien nous allons d'abord introduire une position de référence.

Définition 4.3. *Une* position de référence $P^{(0)}$ *d'un polygone* P *est une position où l'un de ses points, nommé* point de référence, *occupe l'origine du plan.*

Nous allons considérer ce point de référence comme un point ancré au polygone et on s'y réfère même si le polygone ne se trouve pas dans la position de référence.

Chaque isométrie positive est une composition d'une translation et d'une rotation. En ayant une position de référence $P^{(0)}$ du polygone P et une position quelconque P' du même polygone, nous pouvons déplacer P de sa position $P^{(0)}$ en P' de la façon suivante :

1. D'abord, on effectue une translation du polygone de la position $P^{(0)}$ à la position P" par un vecteur (x, y), où (x, y) sont les coordonnées du point de référence de P à la position P'.
2. Ensuite, on fait une rotation du polygone autour de son point de référence de la position P" à la position P' d'un angle θ (*voir fig. 4.4*) tel que $\theta \in [0, 360[$.

Par conséquent, nous avons :

Définition 4.4. *En ayant une position de référence* $P^{(0)}$ *du polygone donné* P *par* P'*, nous allons comprendre le triplet* $\langle x, y, \theta \rangle$*, où* $x, y \in \mathbb{R}$ *et* $\theta \in [0, 360[$*.* P' *est obtenu de* $P^{(0)}$ *en composant la translation par le vecteur* (x, y) *avec une rotation par l'angle* θ *autour du point de référence de* P*.*

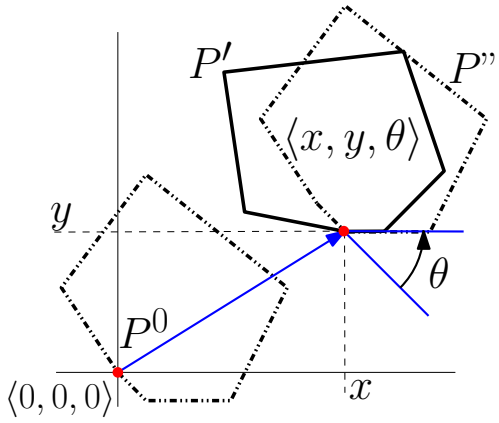


FIG. 4.4: Les coordonnées d'un point de référence après un déplacement

Définition 4.5. *Un* mouvement *du polygone P est décrit comme une fonction continue $M_P : t \to \langle x, y, \theta \rangle$ où t est le temps. Lorsque $M_P : t \to \langle 0, 0, 0 \rangle$ pour tout t, nous appelons ce mouvement le* mouvement identité.

Durant le mouvement du polygone, les contacts restent immobiles, constituant les obstacles pour ce mouvement.

Définition 4.6. *Un ensemble de contacts CS (doigt ou mur)* immobilise *un polygone P si le mouvement identité est le seul mouvement où aucun élément de CS se retrouve à l'intérieur de P, voir* [3].

Pour qu'un ensemble de contacts immobilise un polygone selon les conditions du premier ordre, il faut que ceux-ci soient positionnés de façon à annuler toutes les rotations.

4.3 *Étau-Doigts-Parallèle*

Définition 4.7. *Nous définissons la serre* Étau-Doigts-Parallèle *comme étant deux* doigts, *D_1 et D_2, situés à la même distance du* mur *m (voir fig. 4.5).*

Définition 4.8. *Soit un polygone P, avec les côtés $C_1, C_2 \in P$ et $C_m \in env(P)$, on appelle* lieu d'intersection *L le lieu géométrique des points de rencontre des normales à C_1 et C_2 lorsque les doigts sur ces côtés sont équidistants au côté C_m (voir fig. 4.6). Le lieu d'intersection est un segment de droite.*

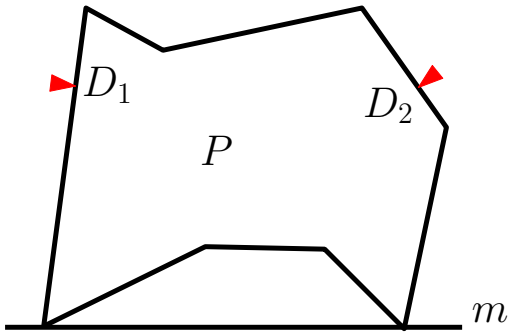


FIG. 4.5: Un polygone P immobilisé avec la serre *Étau-Doigts-Parallèle*

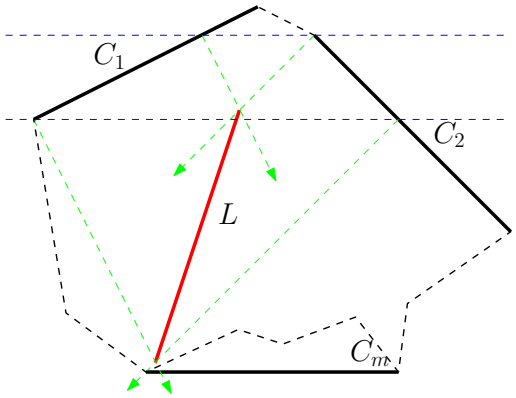


FIG. 4.6: Le *lieu d'intersection* d'un polygone.

Définition 4.9. *La fonction* lieuIntersection(C_m, C_1, C_2) *retourne le lieu d'intersection pour les doigts placés sur les côtés* C_1 *et* C_2 *et le mur sur le côté* C_m.

Théorème 4.10. Étau-Doigts-Parallèle *immobilise un polygone si et seulement si :*

1. *Le lieu d'intersection* L *croise* $\mathcal{B}(C_m)$ *où* C_m *est le côté de l'enveloppe convexe en contact avec le mur*
2. *Les vecteurs des trois normales à* C_1, C_2 *et* C_m *complètent* $\mathbb{R}^2$

Preuve :

1. **La nécessité** :

   (a) Supposons d'abord que la première condition du théorème n'est pas vérifiée (*voir fig. 4.7*). Prenons un point O dans le triangle $\triangle KMN$ déterminé par : les normales des côtés C_1 et C_2 au doigts D_1 et D_2, respectivement, ainsi que la normale au mur au point B.

Il est facile de voir qu'une légère rotation positive de P autour de O à la position P' laisse tous les points de contact, soit D_1, D_2, B et C (le second point de contact du mur), à l'extérieur de P'. En conséquence, P n'est pas immobilisé. Par symétrie, si le point X d'intersection de normales aux doigts se trouve à la droite de la normale au mur au point C, alors une rotation négative de P est possible.

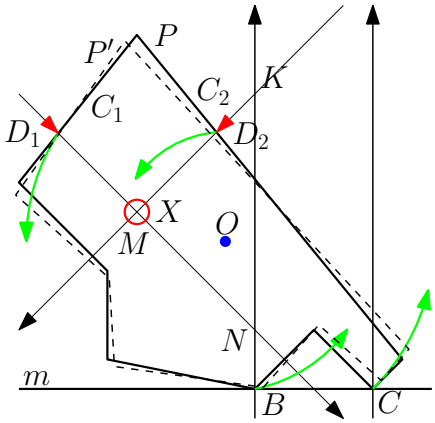


FIG. 4.7: La rotation du polygone est possible

(b) Maintenant, supposons que la seconde condition n'est pas vérifiée (*voir fig. 4.8*). Puisque les vecteurs $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$ de trois normales aux contacts D_1, D_2 et B ne complètent pas $\mathbb{R}^2$, on peut ramener ces vecteurs à la même origine O de sorte que dans une région angulaire α autour de O, tel que $\alpha > 180^o$, n'appartient aucun de ces trois vecteurs $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$. En conséquence, on peut choisir un vecteur $\overrightarrow{v}$ qui forme des angles obtus avec chacun des vecteurs $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$. ($\overrightarrow{v}$ ne peut pas être construit à l'aide d'une combinaison linéaire non-négative de $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$.) Prenons le vecteur $\overrightarrow{v}' = -\overrightarrow{v}$. Ce vecteur forme alors des angles inférieurs à 90^o avec $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ et $\overrightarrow{v_3}$. En conséquence, il est possible de faire une translation de P par $\overrightarrow{v}'$ à la nouvelle position P' où les points de contact D_1, D_2, B et C demeurent à l'extérieur. Dans ce cas, P n'est pas immobilisé.

2. **La suffisance** :

Prenons les droites orthogonales aux points A_1, A_2, B, C. On prou-

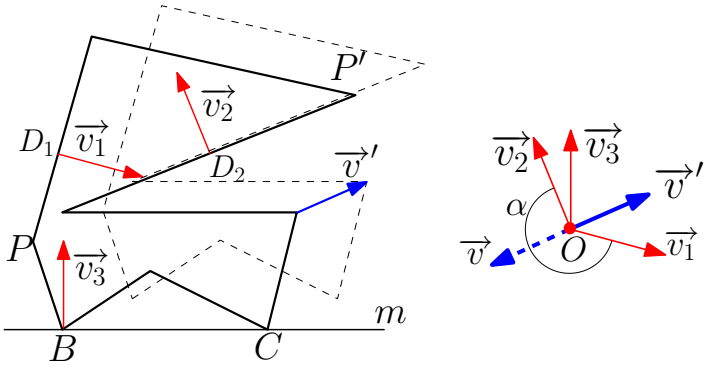


FIG. 4.8: (a) La translation du polygone (b) La direction du vecteur de translation

vera qu'aucune rotation de P n'est possible. Les normales partagent le plan en dix régions. Pour chaque région, on détermine quelles sont les rotations admissibles de sorte que le centre de rotation placé dans une région cause la pénétration de certains points de contact. Par exemple, la région marquée par $---+$ (*voir la fig. 4.9*), correspondant aux rotations possibles aux points A_1, A_2, B, C, respectivement, indique que les rotations négatives de P ne causent pas la pénétration des contacts A_1, A_2 et B mais causent la pénétration du point C. Puisque chaque région est marquée par une séquence de signes contenant au moins un $+$ et un $-$, chaque rotation fait qu'au moins un contact pénètre à l'intérieur du polygone. P est alors immobilisé selon les conditions du premier ordre. □

Définition 4.11. *On appelle* configuration *le quadruplet* $\langle C_m, C_1, C_2, I\rangle$ *où :*

- C_m *est le côté de l'enveloppe convexe en contact avec le mur*
- C_1 *et* C_2 *sont les côtés du polygone en contact avec les doigts de* Étau-Doigts-Parallèle
- I *est l'intervalle maximal* $]a, b[$ *des distances possibles entre les doigts et le mur,* a *est la plus courte distance du mur et* b *la plus longue.*

Pour chaque $i \in I$*, les doigts sur* C_1 *et* C_2 *à distance* i *avec le mur* m *immobilisent le polygone* P*.*

Une *configuration* représente une solution d'immobilisation avec *Étau-Doigts-Parallèle.*

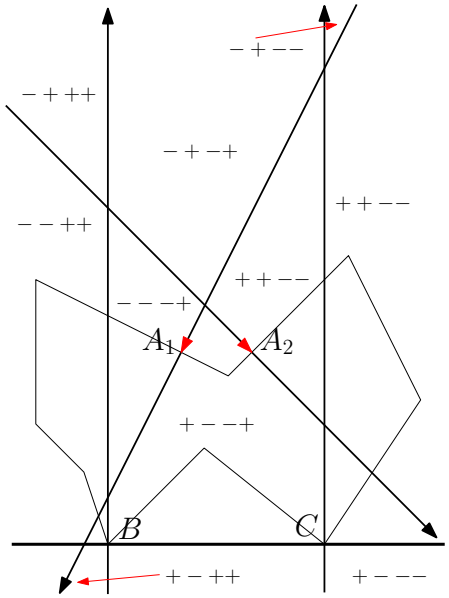


FIG. 4.9: Les régions des rotations possibles

Définition 4.12. *La fonction* distanceValide(L, C)*, où* L *est le lieu d'intersection et* C *un côté, retourne :*

1. ϕ *si* L *ne croise pas* $\mathcal{B}(C)$
2. *L'intervalle de distance* $]a, b[$ *(la distance entre les doigts et le mur) si* L *croise* $\mathcal{B}(C)$*.*

4.4 La pointe

Définition 4.13. *Soit un polygone convexe* P *et le côté* $C_m \in bord(P)$ *en contact avec un mur* m*. On appelle* pointe *le sommet* $p \in bord(P)$ *le plus éloigné de la droite* m*.*

Si deux sommets sont candidats pour être une *pointe*, c'est le sommet avec le plus petit indice (dans le tableau de n sommets décrivant le polygone) qui est la pointe.

Définition 4.14. *La fonction trouverPointe*(P) *recherche la pointe du premier côté d'un polygone convexe* P*.*

Chapitre 5

Recherche d'une configuration

Savoir qu'une classe de polygones possède au moins une configuration d'immobilisation avec la serre *Étau-Doigts-Parallèle* permet de développer une approche différente pour les analyser. Dans notre cas, avec la création d'algorithmes efficaces, cette configuration peut être repérée assez rapidement. La classe de polygone Π que nous avons identifiée est composée des polygones dont l'enveloppe convexe ne possède pas de côtés parallèles.

Dans ce chapitre, nous démontrons la présence d'une configuration d'immobilisation pour tous les polygones de cette classe. De plus, nous présentons deux algorithmes, un premier pour les polygones convexes et le second pour les polygones arbitraires, permettant de trouver la configuration d'immobilisation.

5.1 Les polygones convexes

5.1.1 Identification de la configuration d'immobilisation

Définition 5.1. *On appelle* largeur d'un polygone *la largeur de la plus étroite bande formée de deux droites parallèles bornant le polygone (voir fig. 5.1).*

Théorème 5.2. *Pour tout polygone convexe $P \in \Pi$, il y a au moins un côté $C_m \in bord(P)$, tel que si le mur est placé sur C_m, alors sa pointe $p \in \mathcal{B}(C_m)$.*

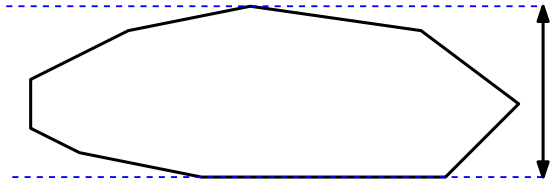


FIG. 5.1: La largeur d'un polygone

Preuve :

Les deux droites parallèles utilisées pour déterminer la largeur de P croisent ce dernier à deux endroits. La première croise un sommet et la seconde croise un côté de P. Dans le cas contraire, les deux droites croiseraient chacune un sommet, soit u et v. Ici, il serait possible de faire faire une rotation à P autour d'un centre situé au milieu du segment $\overline{uv}$ (*voir fig. 5.2a*). P serait positionné entièrement entre ces deux droites sans toutefois les toucher. Ceci contredit la définition de la largeur.

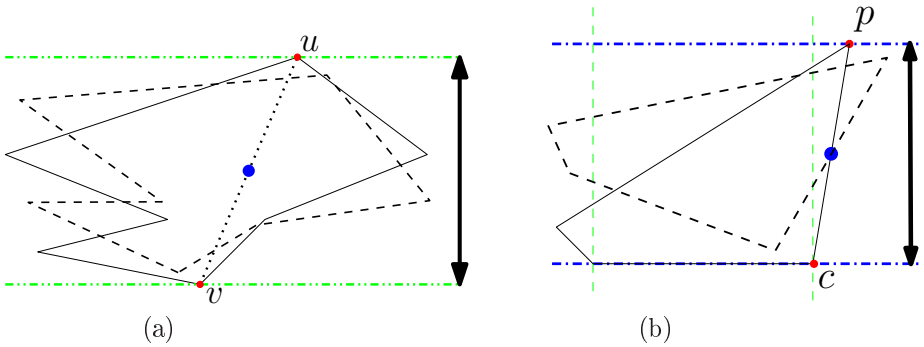


FIG. 5.2: (a) Les 2 droites doivent toucher à un côté et à un sommet (b) le sommet doit être situé dans la bande $\mathcal{B}$ du côté

Si les deux droites bornant la largeur de P sont en contact avec le sommet p et le côté C, alors $p \in \mathcal{B}(C)$. Sinon, il existerait un point $c \in C$ qui est l'extrémité de C la plus près de p. Encore une fois, il serait possible de faire une rotation de P autour d'un centre situé au milieu du segment $\overline{cp}$. Après cette rotation, P serait toujours entre les deux droites parallèles de la largeur, mais ne serait aucunement en contact avec celles-ci. Encore une fois, ceci contredit la définition de la largeur. □

Il est à noter que tous les parallélogrammes ne peuvent pas être immobilisé par *Étau-Doigts-Parallèle*.

Lemme 5.3. *Une configuration d'immobilisation est composée du quadruplet* $\langle C_m, C_1, C_2, I\rangle$ *où :*

- C_m *: le côté dont la pointe* $p \in \mathcal{B}(C_m)$
- C_1 *et* C_2 *: les deux côtés adjacents à la pointe* p
- I *: l'intervalle de distance à laquelle les doigts peuvent être situés du mur.*

Preuve :

Soit L le lieu d'intersection L de C_1 et C_2 avec le mur sur C_m. Puisque l'une des extrémités de L est la pointe p, il y a une portion de L qui croise $\mathcal{B}(C_m)$. La première condition d'immobilisation est donc respectée (*voir théorème 4.10*). Les normales des côtés C_m, C_1 et C_2 complètent $\mathbb{R}^2$ d'après le lemme 2.11.

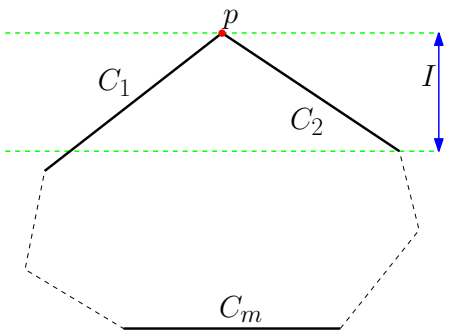


FIG. 5.3: Exemple de configuration

5.1.2 L'algorithme

Définition 5.4. *La fonction* $polyConv1Config(P)$ *recherche une configuration d'immobilisation pour le polygone convexe* $P \in \Pi$.

La stratégie de $polyConv1Config$ consiste à vérifier pour chaque côté C du polygone si sa pointe $p \in \mathcal{B}(C)$. L'exécution s'arrête dès qu'une solution est trouvée et que la configuration correspondante est créée.

La séquence d'opérations de cette fonction est la suivante :

- Pour chaque côté C du polygone :
  - Trouver la pointe p du côté C.
  - Si $p \in \mathcal{B}(C)$:
    - Trouver le lieu d'intersection des deux côtés adjacents à p avec le mur sur C.
    - Déterminer l'intervalle des distances auxquelles les doigts peuvent être situés du mur.
    - Créer la configuration d'immobilisation.

La table 5.1 contient le pseudo-code de la fonction $polyConv1Config$.

```
1.  fonction polyConv1Config(P) :
2.      pointe ← 2, i ← 0, n ← ||P||
3.      tant que i < n :
4.          cote ← créer un côté avec les sommets P[i] et P[(i + 1) mod n]
5.          d ← créer une droite de segment avec cote
6.          tant que distance(d, P[pointe]) < distance(d, P[(pointe + 1) mod n]) :
7.              pointe ← (pointe + 1) mod n
8.          si pointe ∈ B(cote) :
9.              c1 ← créer un côté avec les sommets P[(pointe − 1 + n) mod n] et P[pointe]
10.             c2 ← créer un côté avec les sommets P[pointe] et P[(pointe + 1) mod n]
11.             L ← lieuIntersection(cote, c1, c2)
12.             I ← distanceValide(L, cote)
13.             retourner ⟨cote, c1, c2, I⟩
14.         i ← i + 1
```

TAB. 5.1: Pseudo-code de la fonction $polyConv1Config$

Lemme 5.5. *Pour tout polygone convexe $P \in \Pi$, la configuration d'immobilisation est identifiée en temps linéaire par la fonction $polyConv1Config(P)$.*

Preuve :

La fonction $polyConv1Config$ utilise le théorème 5.2 et le lemme 5.3 pour construire la configuration d'immobilisation.

La recherche de la prochaine pointe s'effectue toujours dans la même direction. Au pire cas, chaque sommet du polygone est visité une fois, à l'exception du premier sommet. Le temps total pour la recherche de la pointe est $O(n)$. Puisque dans le pire cas, l'algorithme doit analyser chaque côté du polygone afin de trouver la configuration, le temps d'exécution pour notre algorithme est $O(n)$. La complexité spatiale est $O(n)$ puisque l'algorithme traite le polygone et utilise un nombre fixe de variables. □

5.2 Les polygones arbitraires

Définition 5.6. *La fonction* $polyArb1Config(P)$ *recherche une configuration d'immobilisation pour le polygone arbitraire* $P \in \Pi$.

La séquence d'opérations de cette fonction est la suivante :

- Trouver l'enveloppe convexe de P.
- Appliquer la fonction $polyConv1Config$ sur l'enveloppe convexe et obtenir la configuration d'immobilisation temporaire S_t de l'enveloppe convexe.
- À partir de S_t et de la pointe p associée à celle-ci, retrouver le sommet p' du polygone P correspondant à p.
- Trouver le lieu d'intersection des deux côtés adjacents à p' avec le mur sur le côté C_m dans la configuration S_t.
- Déterminer l'intervalle des distances auxquelles les doigts peuvent être situés du mur.
- Construire la nouvelle configuration d'immobilisation.

Lemme 5.7. *Pour tout polygone arbitraire* $P \in \Pi$*, la configuration d'immobilisation est identifiée en temps linéaire par la fonction* $polyArb1Config(P)$.

Preuve :

Le recherche de l'enveloppe convexe se fait en temps linéaire pour les polygones (*voir* [4, 5]). Le reste de l'algorithme est le même que pour les polygones convexes dont la complexité est de temps $O(n)$.

Avec les résultats précédents, nous pouvons donc en déduire le théorème suivant :

Théorème 5.8. *Tout polygone dont l'enveloppe convexe n'a pas de côtés parallèles possède une configuration d'immobilisation avec la serre* Étau-Doigts-Parallèle *pouvant être identifiée en temps et en espace mémoire linéaires.*

Chapitre 6

Recherche de toutes les configurations

Avoir la possibilité d'obtenir l'ensemble des configurations d'immobilisation pour un polygone permet de choisir la configuration optimale selon certains critères.

Dans ce chapitre, en premier lieu nous présentons un algorithme fonctionnant strictement sur les polygones convexes. Pour la seconde partie, un algorithme s'attaquant aux polygones arbitraires y est démontré.

Bien que les algorithmes que nous présentons donnent des résultats pour les polygones dont l'enveloppe convexe ne possède pas de côtés parallèles, ils fonctionnent pour tous les polygones, sans toutefois produire toujours des résultats.

6.1 Les polygones convexes

Définition 6.1. *La fonction* orientationNormale*(X, Y) retourne la paire* $\langle C, \overrightarrow{N_C} \rangle$ *où :*

1. *C est un segment orienté dont l'origine est le point X et la fin le point Y*
2. $\overrightarrow{N_C}$ *est le vecteur normalisé de la normale du segment C.*

Si Z est la paire retournée, alors la fonction cote*(Z) retourne le premier élément et la fonction* normale*(Z) le second.*

Définition 6.2. *Soit trois côtés* C_m, C_1 *et* C_2 *d'un polygone convexe où* C_m *est le côté en contact avec le mur m. On dit que* C_1 *et* C_2 *forment une* couche *si leur lieu d'intersection n'est pas nul et que leurs normales complètent* $\mathbb{R}^2$ *avec celle du côté* C_m *(voir fig. 6.1).*

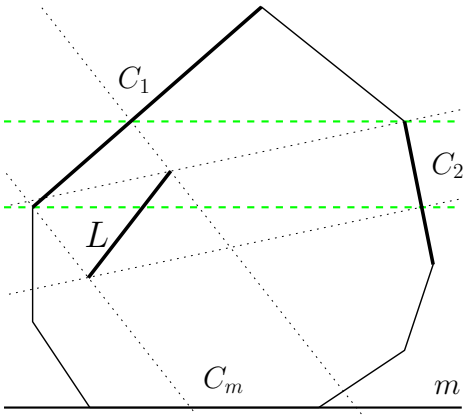


FIG. 6.1: Une couche

Lemme 6.3. *Un polygone convexe possède jusqu'à* $\Omega(n^2)$ *configurations d'immobilisation avec la serre* Étau-Doigts-Parallèle.

Preuve :

Soit un polygone régulier à n côtés, où n est pair mais n'est pas un multiple de 4. Les n côtés du polygone peuvent être en contact avec le mur. Pour chacun de ces côtés, on retrouve $\frac{n-2}{4}$ couches dont le lieu d'intersection croise $\mathcal{B}(C_m)$(voir fig. 6.2). Il y a donc $\frac{n(n-2)}{4} \in O(n^2)$ configurations d'immobilisation. □

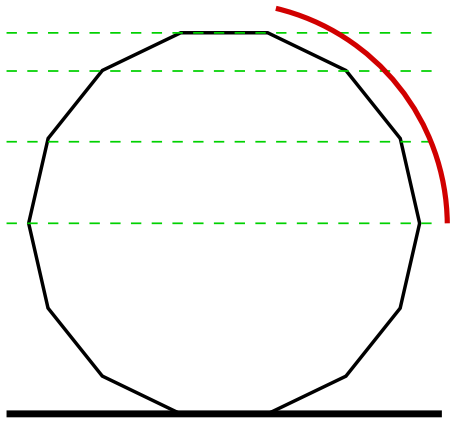


FIG. 6.2: Un polygone avec $\frac{n-2}{4}$ couches

Définition 6.4. *La fonction* $polyConvToutesConfigs\ (P)$ *trouve toutes les configurations d'immobilisation d'un polygone convexe* P.

La séquence d'opérations de cette fonction est la suivante :

- Trouver la pointe du premier côté.
- Pour chaque côté C_m du polygone allant sur le mur :
  - Pour chaque couche H, à partir de la pointe vers le mur :
    - Trouver le lieu d'intersection L des deux côtés composant la couche H.
    - Si L croise $\mathcal{B}(C_m)$:
      - Trouver l'intervalle des distances valides pour positionner les doigts.
      - Créer une configuration d'immobilisation avec C_m, les deux côtés formant la couche H et l'intervalle de distance.
      - Rapporter cette configuration.
  - Trouver la pointe du prochain côté.

La table 6.1 contient le pseudo-code de la fonction *polyConvToutesConfigs*.

Lemme 6.5. *Pour tout polygone convexe P, la fonction polyConvToutesConfigs(P) énumère toutes les configurations d'immobilisation de P avec la serre* Étau-Doigts-Parallèle.

Preuve :

Pour chaque côté, en partant de la pointe et en se dirigeant vers le mur, chaque couche est analysée. Avec une couche, la seconde condition d'immobilisation est automatiquement respectée. Il ne reste donc qu'à vérifier, avec le lieu d'intersection, si la première règle est respectée pour trouver une configuration d'immobilisation. Sur un polygone convexe, toutes les couches sont analysées par *polyConvToutesConfigs*, il est donc certain que chaque configuration d'immobilisation est trouvée. □

Lemme 6.6. *La fonction polyConvToutesConfigs possède les complexités suivantes :*

- *Un espace mémoire de $O(n)$*
- *Un temps d'exécution de $\Theta(n^2)$*

Preuve :

Comme toutes les variables de la fonction sont simples, la complexité spatiale est $O(n)$, n étant la complexité du polygone. Pour chaque côté, l'algorithme

```
1.  fonction polyConvToutesConfigs(P) :
2.      i ← 0, pointe ← trouverPointe(P)
3.      d ← créer une droite avec les sommets P[0] et P[1]
4.      tant que i < n :
5.          C_m ← orientationNormale(P[i], P[(i + 1) mod n])
6.          droite ← pointe, gauche ← pointe
7.          si distance(d, P[droite]) = distance(d, P[gauche]) :
8.              gauche ← (gauche + 1) mod n
9.          C_1 ← orientationNormale(P[gauche], P[(gauche + 1) mod n])
10.         C_2 ← orientationNormale(P[(droite − 1 + n) mod n], P[droite])
11.         tant que normale(C_m), normale(C_1) et normale(C_2) complètent ℝ² :
12.             L ← lieuIntersection(cote(C_m), cote(C_1), cote(C_2))
13.             I ← distanceValide(L, cote(C_m))
14.             si I ≠ φ :
15.                 rapporter ⟨cote(C_m), cote(C_1), cote(C_2), I⟩
16.             dg ← distance(d, P[(gauche + 1) mod n])
17.             dd ← distance(d, P[(droite − 1 + n) mod n])
18.             si dg ≥ dd :
19.                 gauche ← (gauche + 1) mod n
20.                 si dg = dd :
21.                     droite ← (droite − 1 + n) mod n
22.             sinon :
23.                 droite ← (droite − 1 + n) mod n
24.             C_1 ← orientationNormale(P[gauche], P[(gauche + 1) mod n])
25.             C_2 ← orientationNormale(P[(droite − 1 + n) mod n], P[droite])
26.         i ← i + 1
27.         d ← créer une droite avec les sommets P[i] et P[(i + 1) mod n]
28.         tant que distance(d, P[pointe]) < distance(d, P[(pointe + 1) mod n]) :
29.             pointe ← (pointe + 1) mod n
```

TAB. 6.1: Pseudo-code de la fonction $polyConvToutesConfigs$

analyse chaque couche existante, qu'il y ait une configuration d'immobilisation ou non. Dans le meilleur et dans le pire cas, chacune des $O(n)$ couches est considérée, c'est pourquoi le temps d'exécution est de $\Theta(n^2)$. □

Comme on peut avoir $\Omega(n^2)$ configurations d'immobilisation (*voir fig. 6.2*) l'algorithme les rapportant doit fonctionner en temps $\Omega(n^2)$. En conséquence, nous avons :

Théorème 6.7. *La fonction polyConvToutesConfigs décrit un algorithme optimal, fonctionnant en temps $\Theta(n^2)$, pour énumérer toutes les configurations d'immobilisation.*

6.2 Les polygones arbitraires

Définition 6.8. *La fonction polyArbToutesConfigs* (P) *trouve toutes les configurations d'immobilisation d'un polygone* P*.*

La séquence d'opérations de cette fonction est la suivante :

- Trouver l'enveloppe convexe du polygone.
- Si l'enveloppe convexe a autant de sommets que le polygone, appeler la fonction *polyConvToutesConfigs* avec le polygone.
- Pour chaque côté C_m de l'enveloppe convexe :
  - Pour chaque paire de côtés C_1 et C_2 du polygone :
    - Si C_1 ou C_2 est le côté C_m ou si les normales de C_m, C_1 et C_2 ne complètent pas $\mathbb{R}^2$, ignorer cette paire.
    - Calculer le lieu d'intersection L de C_1 et C_2 par rapport au côté C_m.
    - Si L croise $\mathcal{B}(C_m)$, calculer l'intervalle I des distances à laquelle les doigts doivent être du mur. Créer et rapporter la configuration $\langle C_m, C_1, C_2, I\rangle$.

La table 6.2 contient le pseudo-code de la fonction *polyArbToutesConfigs*.

Lemme 6.9. *La fonction polyArbToutesConfigs*(P) *trouve toutes les configurations d'immobilisation sur le polygone* P*.*

Preuve :

La définition 4.11 décrit une configuration d'immobilisation comme étant un quadruplet composé du côté de l'enveloppe convexe en contact avec le mur, des deux côtés en contact avec les deux doigts et l'intervalle des distances entre les doigts et le mur.

Pour chaque côté de l'enveloppe convexe, la fonction *polyArbToutesConfigs* analyse chaque paire de côtés afin de savoir s'ils peuvent former une configuration d'immobilisation. Chaque possibilité est donc prise en compte. □

Lemme 6.10. *Il existe des polygones possédant* $\Omega(n^3)$ *configurations d'immobilisation avec la serre* Étau-Doigts-Parallèle*.*

```
1.  fonction polyArbToutesConfigs(P) :
2.      env ← envConvexe(P)
3.      e ← ||env||, n ← ||P||
4.      si e = n :
5.          polyConvToutesConfigs(P)
6.      sinon :
7.          i ← 0
8.          tant que i < e :
9.              C_m ← orientationNormale(env[i], env[(i + 1) mod e])
10.             j ← 0
11.             tant que j < (n − 1) :
12.                 C_1 ← orientationNormale(P[j], P[j + 1])
13.                 k ← j + 1
14.                 tant que k < n :
15.                     C_2 ← orientationNormale(P[k], P[(k + 1) mod n])
16.                     si cote(C_1) ≠ cote(C_m) et cote(C_2) ≠ cote(C_m) :
17.                         si normale(C_m), normale(C_1) et normale(C_2) complètent ℝ² :
18.                             L ← lieuIntersection(cote(C_m), cote(C_1), cote(C_2))
19.                             I ← distanceValide(L, cote(C_m))
20.                             si I ≠ φ :
21.                                 rapporter ⟨cote(C_m), cote(C_1), cote(C_2), I⟩
22.                     k ← k + 1
23.                 j ← j + 1
24.             i ← i + 1
```

TAB. 6.2: Pseudo-code de la fonction *polyArbToutesConfigs*

Preuve :

Nous allons construire un polygone à l'aide de deux cercles $\mathcal{C}_A$ et $\mathcal{C}_B$ concentriques à l'origine du plan cartésien où le rayon du premier cercle est très petit (légèrement supérieur à zéro). Sur le périmètre de $\mathcal{C}_A$, plaçons n points, A_1 à A_n, formant les sommets d'un polygone régulier. De la même façon, sur le cercle $\mathcal{C}_B$ plaçons, respectivement, les points B_1 à B_n. Construisons le polygone en forme d'étoile $P = A_1B_2A_3 \ldots A_{n-1}B_n$ avec le mur m, en contact avec les deux sommets de P les plus bas (*voir fig. 6.3*).

Soit une paire de côté $C_1 = \overline{A_{2k-1}B_{2k}}$ et $C_2 = \overline{B_{2j}A_{2j+1}} \in bord(P)$ en contact avec les doigts, où C_1 appartient au troisième quadrant du plan et C_2 au quatrième, et $C_m = \overline{B_{2i}B_{2i+2}}$, le côté en contact avec le mur. Puisque le cercle $\mathcal{C}_A$ est très petit, le lieu d'intersection L des côtés C_1 et C_2 a une de ses extrémités arbitrairement proche de l'origine du plan. En conséquence, L croise $\mathcal{B}(C_m)$ et les deux conditions d'immobilisation du théorème 4.10 sont vérifiées. Puisqu'il y a $O(n)$ choix pour placer le mur avec $O(n)$ côtés C_1 dans

le troisième quadrant ainsi que $O(n)$ côtés C_2 dans le quatrième quadrant, le polygone construit admet $\Omega(n^3)$ configurations d'immobilisation pour la serre *Étau-Doigts-Parallèle*. □

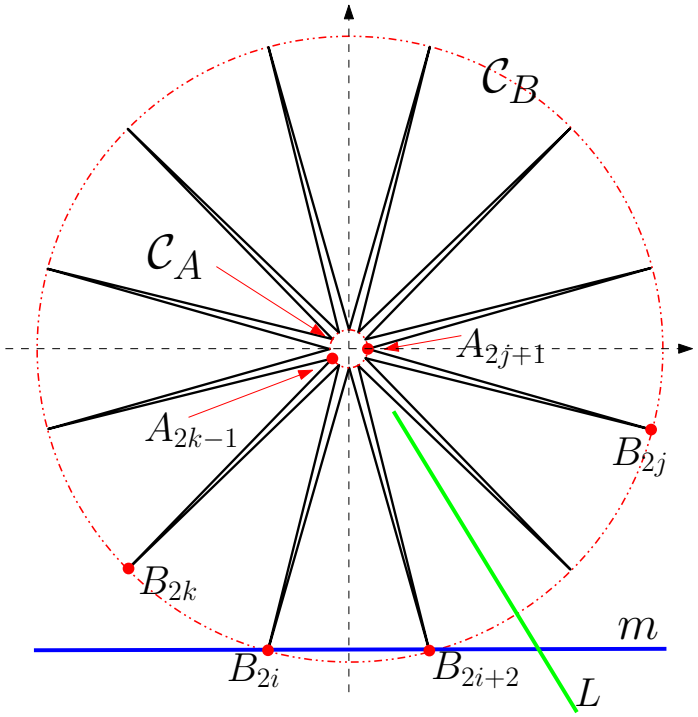


FIG. 6.3: Polygone de la preuve du lemme 6.10

On peut conclure par le théorème suivant :

Théorème 6.11. *La fonction polyArbToutesConfigs est optimale puisqu'elle énumère toutes les configurations d'immobilisation en* $\Theta(n^3)$.

Chapitre 7

Conclusion

Dans ce travail, nous avons fait une analyse complète de la serre *Étau-Doigts-Parallèle*. Nous avons démontré que pour les polygones avec une enveloppe convexe sans côtés parallèles, il est toujours possible de les immobiliser avec notre serre. De plus, nous avons développé des algorithmes fonctionnant en temps linéaire, par rapport au nombre de côtés du polygone, pour trouver cette configuration d'immobilisation.

Pour la seconde partie de nos résultats, nous nous sommes intéressés à la recherche de toutes les configurations d'immobilisation existantes sur différents polygones. En premier, il y a eu la construction d'un algorithme spécialisé pour les polygones convexes trouvant toutes les configurations en temps quadratique. En deuxième, un algorithme dit général fonctionne sur n'importe quel polygone et énumère toutes les configurations en $O(n^3)$ où n est le nombre de côtés sur le polygones. Nous avons montré que les deux algorithmes sont optimaux en prouvant que $\Omega(n^2)$ et $\Omega(n^3)$ sont les bornes inférieures des algorithmes pour les polygones convexes et arbitraires, respectivement.

Bibliographie

[1] Cheong, J.-S., Haverkort, H. J., and van der Stappen, A. F. Computing all immobilizing grasps of a simple polygon with few contacts. *Algorithmica 44*, 2 (2006), 117–136.

[2] Czyzowicz, J., Stojmenovic, I., and Urrutia, J. Immobilizing a polytope. In *WADS* (1991), pp. 214–227.

[3] Czyzowicz, J., Stojmenovic, I., and Urrutia, J. Immobilizing a shape. *Int. J. Comput. Geometry Appl. 9*, 2 (1999), 181–206.

[4] Graham, R. L., and Yao, F. F. Finding the convex hull of a simple polygon. *J. Algorithms 4*, 4 (1983), 324–331.

[5] Lee, D. T. On finding the convex hull of a simple polygon. *International Journal of Computer and Information Sciences 12*, 2 (1983), 87–98.

[6] Markenscoff, X., Ni, L., and Papadimitriou, C. H. The geometry of grasping. *Int. J. Rob. Res. 9*, 1 (1990), 61–74.

[7] Mishra, B., Schwartz, J. T., and Sharir, M. On the existence and synthesis of multifinger positive grips. *Algorithmica 2* (1987), 541–558.

[8] Overmars, M. H., Rao, A. S., Schwarzkopf, O., and Wentink, C. Immobilizing polygons against a wall. In *Symposium on Computational Geometry* (1995), pp. 29–38.

[9] Rimon, E., and Burdick, J. On force and form closure for multiple finger grasps. In *Proceedings of IEEE International Conference on Robotics and Automation* (1996), pp. 1795–1800.

[10] Rimon, E., and Burdick, J. Mobility of bodies in contact - part i : A second-order mobility index for multiple-finger graps. *IEEE Transactions on Robotics and Automation 14* (1998), 696–708.

[11] Rimon, E., and Burdick, J. Mobility of bodies in contact - part ii : How forces are generated by curvature effects. *IEEE Transactions on Robotics and Automation 14* (1998), 709–717.

[12] Rimon, E., and Burdick, J. W. New bounds on the number of frictionless fingers required to immobilize 2d objects. In *ICRA* (1995), pp. 751–757.

[13] van der Stappen, A. F., Wentink, C., and Overmars, M. H. Computing immobilizing grasps of polygonal parts. *I. J. Robotic Res. 19*, 5 (2000), 467–479.

[14] von Reuleaux, F. *The Kinematics of Machinery.* Dover, 1876.