Contributions

Chapitre 5

Tatouage fragile par invariants géométriques

5.1 Motivations

On peut parfois lire dans la littérature que des algorithmes de tatouage qui modifient un invariant géométrique sont robustes. Cela n'est vrai que pour la classe de transformations autorisées. Une manipulation modifiant la connexité peut également effacer la marque : un remaillage par exemple suffit à lessiver tout à fait le maillage tatoué. Cette famille d'algorithmes ne peut donc pas être employée pour des applications de type protection des droits.

Par contre, le tatouage par invariant semble être un bon choix dans des applications de type authentification. En effet, on connaît exactement la classe de manipulations qui n'entraîneront pas le lessivage de la marque : les transformations affines, projectives, etc. Ainsi en cas de perte du tatouage, on pourra affirmer que s'est produite une manipulation non autorisée dessus. Eventuellement on redemandera la transmission de l'objet. On utilise alors un tatouage fragile, *conçu* pour s'effacer à la moindre manipulation non autorisée. Une telle technique trouverait son application naturelle dans l'authentification des maillages muséologiques ou biomédicaux.

C'est sous cet angle que nous avons abordé la conception d'une méthode de tatouage pour l'authentification. Nous utilisons pour ce faire une primitive géométrique d'insertion afin de cacher les valeurs des bits. Cette primitive possède plusieurs particularités qui la rendent particulièrement flexible. Nous commençons par définir notre primitive géométrique d'insertion de l'information cachée. Ensuite, nous validons notre primitive à l'aide d'un arrangement global/local de l'information (notre primitive sert à cacher les valeurs des bits, il faut donc que l'arrangement utilisé propose une numérotation implicite des bits). Nous aborderons alors la conception de l'algorithme de tatouage fragile pour l'authentification, fondé sur un arrangement indexé. Après avoir énoncé les contraintes (sur le parcours du graphe de connexité) liées à notre primitive munie d'un arrangement indexé, nous procèderons à une évaluation des performances brutes de l'arrangement indexé. Enfin, nous proposerons un parcours du graphe de connexité qui maximise le nombre de sites pour le tatouage effectivement atteints. Nous montrerons que notre parcours s'effectue en temps linéaire pour les surfaces de genre nul, sans y être limité. Nous serons alors en mesure de proposer une

modélisation statistique de la confiance dans le résultat issu du détecteur. Nous fixerons notamment une probabilité de fausse alarme de 10^{-7}, et nous discuterons le nombre de bits effectivement utilisés pour garantir l'invisibilité statistique de la marque. Les résultats préciserons les performances globales de l'algorithme, et permettront d'évaluer empiriquement la taille de la plus petite surface protégée (MWS dans la littérature, pour Minimum Watermark Segment). En effet, nous souhaitons pouvoir retrouver la marque même si le maillage a été coupé, et nous utilisons un arrangement indexé à dessein. Seul ce type d'arrangement permet de s'affranchir, du mieux possible, de l'attaque de coupe. Nous présenterons dans la section 5.2 une primitive géométrique permettant d'insérer un bit dans une configuration géométrique locale composée d'un seul triangle. Puis nous montrerons dans la section 5.3 comment lui adjoindre un arrangement indexé. Cette primitive géométrique impose une contrainte de causalité sur l'ordre de parcours des sommets, nous verrons dans la section 5.4 comment optimiser le nombre de sites en fonction de cette contrainte de causalité (nous retrouvons ici le même problème que dans [82] - mais nous utiliserons quant à nous l'arrangement indexé précisément dans le but de s'affranchir du problème causal *à la relecture*, et par là résister à la coupe). Dans la section 5.5, nous implanterons enfin une méthode de tatouage fragile que nous pourrons caractériser finement.

5.2 Une primitive géométrique d'insertion

Nous détaillons ici la manière dont nous codons l'information de la valeur des bits à cacher. Cette insertion prend la forme d'une procédure géométrique substitutive. Cette procédure géométrique d'insertion présente les deux avantages d'être flexible et de nécessiter un support compact. Afin d'*illustrer* les performances brutes de notre primitive géométrique d'insertion de la valeur des bits, nous utilisons un arrangement global puis local. Nous avons donc cherché à cacher quelques kilobits d'information dans une dizaine de maillages afin de vérifier la pertinence de notre stratégie d'insertion de la valeur des bits.

La procédure géométrique en question est fondée sur la manipulation d'un invariant géométrique autorisant la translation, la rotation et la mise à l'échelle uniforme. Nous choisissons de modifier le rapport des longueurs de deux segments situés sur la même droite. Cette modification a lieu dans le plan du triangle considéré. Nous schématisons cette procédure sur la Fig. 5.1.

5.2.1 Description

Le principe utilisé est le suivant : le côté de référence étant noté AB, on vient perturber la projection de C sur AB de manière à ce que la projection de C sur AB tombe dans le sous-espace binaire souhaité. Pour cela, il nous faut tout d'abord préciser la manière dont sont construits les sous-espaces binaires.

On commence tout d'abord par diviser le segment $[AB]$ en un nombre pair arbitraire de parties : ce nombre est appelé la *finesse* du tatouage. Ensuite, on associe à chaque partie un symbole binaire de tatouage “0” ou “1”, en alternant la succession des symboles de manière à minimiser le nombre de transitions entre les parties étiquetées “0” et “1” (voir Fig. 5.1) : on utilisera donc par exemple la séquence “01100110” plutôt que “01010101” qui comporte 3

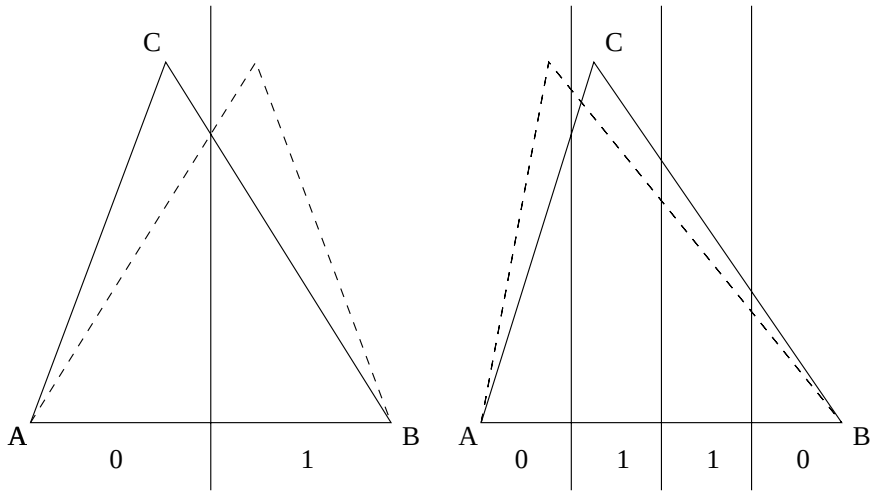

FIG. 5.1 – Procédure d'inscription de l'information cachée. La projection du point C sur la base AB du triangle sélectionne une zone "0" ou "1". Pour forcer la valeur du bit contenu dans un triangle, on effectue une simple symétrie par rapport au plan de symétrie le plus proche. A gauche : distorsion maximale, la configuration dégénère en une simple symétrie par rapport à la médiatrice de [AB]. A droite : distorsion deux fois moindre, le plan de symétrie est situé au quart de AB.

frontières de plus entre parties "0" et "1". La taille des parties est donc directement proportionnelle à la longueur du segment $[AB]$, et varie alors avec chaque triangle considéré. Cette partition du segment $[AB]$ est ensuite étendue à la droite (AB) par répétition, voir Fig. 5.2.

A présent, nous décrivons la manière dont nous tirons parti de cette partition pour insérer l'information à cacher. Nous considérons les plans 3D perpendiculaires à (AB) et passant par les points de la partition marquant une transition entre une zone étiquetée "0" et une zone étiquetée "1". Ces plans seront en réalité des plans de symétrie en fonction desquels nous déplacerons le point C. Ainsi, si la projection de C sur la droite (AB) tombe sur une partie étiquetée avec le symbole binaire à coder souhaité, nous ne faisons rien. Par contre, dans le cas où une modification du point C est nécessaire pour coder le symbole souhaité, nous procéderons à une symétrie par rapport au plan le plus proche. Nous donnons sur la Fig. 5.3 une simulation d'une telle procédure sur un maillage contenu dans le plan 2D.

Cette procédure géométrique permet donc de fixer la distorsion introduite, et de l'exprimer en fonction du nombre de parties souhaitées sur le segment $[AB]$: c'est la finesse. En outre, elle ne requiert qu'un seul triangle pour cacher un bit, elle utilise donc aussi peu de sommets que possible. Nous reproduisons sur la Fig. 5.4 l'évolution du plus grand déplacement requis pour cacher un bit en fonction du nombre de parties sur le segment $[AB]$. Comme l'on pouvait s'y attendre, l'évolution du plus grand déplacement est inversement proportionnelle à la finesse de la procédure.

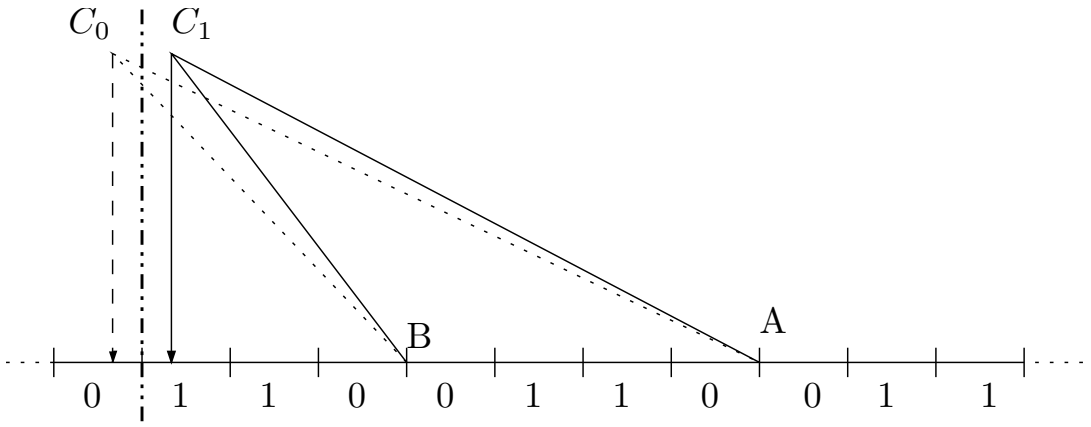

FIG. 5.2 – Répétition des sous-espaces binaires dédiés au codage de la valeur des bits.

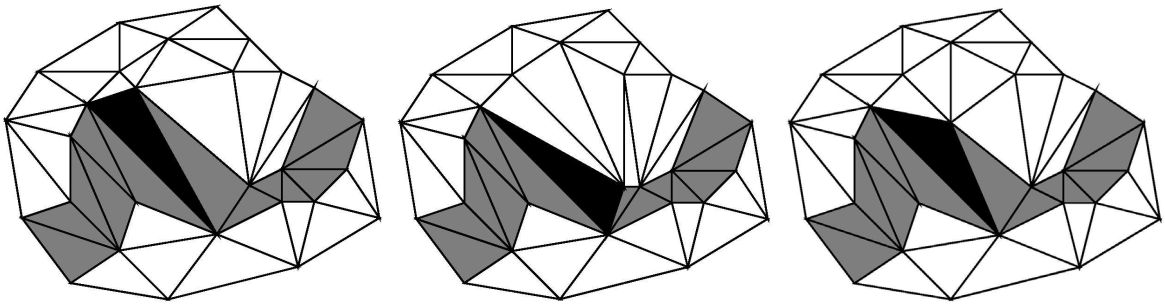

FIG. 5.3 – Simulations sur un maillage du plan 2D de la procédure géométrique d'insertion. Nous appliquons la procédure sur le triangle noir au centre du ruban gris. A gauche : le maillage original, le point C du triangle noir est celui situé le plus haut. Au centre : la partition de la procédure contient 2 parties, c'est la primitive la moins fine qui puisse être. A droite : la partition contient quatre parties, la distorsion introduite est un peu plus fine.

5.2.2 Remarques sur la sécurité

Nous discutons à présent la sécurité offerte par cette procédure géométrique d'insertion de la valeur. Un message bien caché est un message qui ne puisse pas même être seulement détecté. En tatouage substitutif, on vient en général caler une variable sur une valeur particulière, et l'espace dans lequel est calculé cette variable de tatouage est modulé par un secret. En ce qui concerne notre procédure géométrique, il faut remarquer qu'on ne vient pas caler une variable de tatouage (ici le point C) sur une valeur particulière, à considérer le tatouage comme une bijection entre un ensemble de valeurs particulières de la variable de tatouage, et celui des valeurs binaires qu'elles codent. Or, nous nous autorisons une plage de variation (continue par morceaux) déterminée par la finesse de la procédure : le sommet C peut être positionné n'importe où *pourvu* que sa projection sur AB tombe là où il faut. La marque enfouie par cette méthode est donc très probablement indétectable.

Tirant parti de ce degré de liberté supplémentaire, qui rend l'imperceptibilité computationnelle possible (en ne nécessitant pas de valeurs particulières de la variable de tatouage

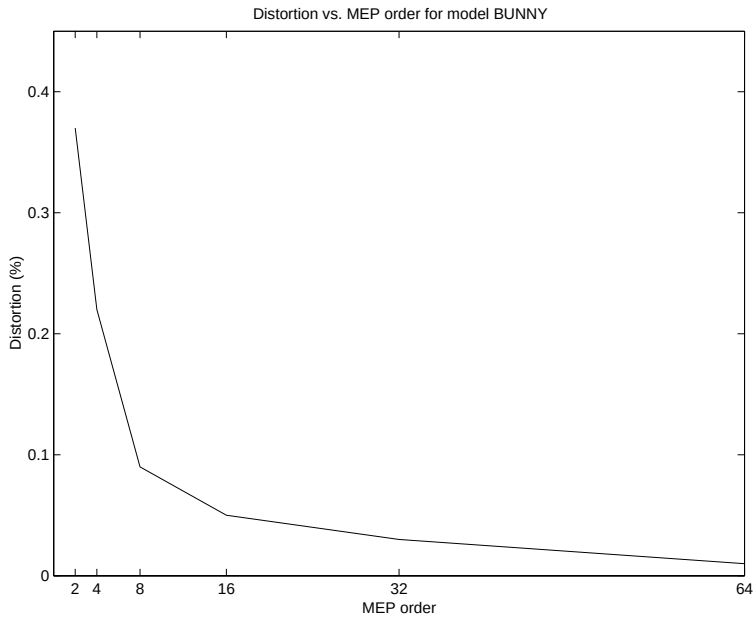

FIG. 5.4 – Evolution (par rapport à la longueur moyenne des arêtes du maillage) du plus grand déplacement de point causé par l'insertion d'information cachée en fonction de la finesse de subdivision de l'arête de référence AB (dénoté MEP order).

pour coder la valeur des bits), nous avons opté pour une simple symétrie. En effet, supposons un pirate qui souhaite retrouver la valeur des bits cachés : la première chose qu'il doit faire en examinant les triangles est de déterminer lequel des trois sommets est le point C. L'utilisation d'une symétrie ne fournit pas d'information au pirate pour détecter sur quel sommet a été appliquée la procédure géométrique. En conséquence, la sécurité de cette procédure d'inscription de la valeur des bits dépendra de la sécurité avec laquelle seront codés les *numéros* des bits, seule information permettant de reconstituer quels furent les sommets modifiés dans chaque triangle, et aussi dans quel *ordre*. C'est la raison pour laquelle nous avons souhaité donner un aperçu aussi précis que possible des différents types d'arrangement de l'information cachée que cette procédure autorise.

Ces remarques sont à la base de l'approche de la sécurité que nous avons eue dans ce travail : nous souhaitons conserver le plus longtemps possible cette imperceptilibité-là à notre avantage. Dans un premier temps, nous chercherons à mieux caractériser les performances de notre procédure d'insertion de la valeur des bits de tatouage. Nous utiliserons pour cela un arrangement global que nous rendrons facilement local, et qui nous permettra de simuler les conditions d'une application de type stéganographique. Et dans un deuxième temps, nous reviendrons à des idées plus communes en tatouage pour réaliser un arrangement indexé de notre procédure. Ce sera l'occasion de développer une méthode de tatouage fragile pour l'authentification. Dans ce dernier cas, nous serons amenés à proposer une évaluation, en nombre de bits, de la sécurité offerte par notre méthode de tatouage fragile, en ce qui concerne la *détection* d'un éventuel message caché (dont la présence ne pourra donc être trahie que par le codage des *numéros* des bits).

5.2.3 Parcours en ruban pour un arrangement global

Si, dans le cas de la stéganographie LSB[77] pour l'image, on cache l'information pixel à pixel, en balayant l'image ligne par ligne (ou colonne par colonne), on peut considérer ce balayage comme définissant un ruban de pixels pris les uns après les autres. C'est sur ce ruban que l'on vient déposer l'information à cacher, définissant implicitement l'ordre dans lequel les bits à cacher sont inscrits les uns après les autres.

Nous souhaitons donc reproduire cette idée dans le cas des maillages triangulés. Nous verrons comment contourner l'absence de parcours canonique sur un maillage arbitraire, et comment en tirer parti pour une plus grande sécurité. Mais avant, il faut définir un moyen de fixer ce qui sera le premier triangle du ruban.

Début et fin du ruban

Dans [7], plusieurs solutions sont proposées, consistant à réduire drastiquement l'espace des triangles possibles pour le choix du premier dans le ruban, quitte à procéder ensuite par essai/erreur, par exemple :

- choisir le triangle de plus petite (ou plus grande) aire
- choisir le triangle de plus petit (ou plus grand) périmètre
- choisir un triangle parmi ceux ayant le plus (ou le moins) de voisins (et procéder ensuite par essai/sélection)

Dans le cas où plusieurs triangles sont éligibles, on procèdera par essai/erreur. La sélection a alors lieu en cherchant à retrouver une séquence connue d'initialisation. Pour notre part, nous faisons en sorte de ne pouvoir conserver qu'un seul triangle pour le premier dans le ruban.

Rappelons que l'inscription de l'information cachée ne modifie pas la connexité des triangles du maillage, mais se greffe uniquement sur l'information de géométrie (les coordonnées cartésiennes). En outre, nous faisons l'hypothèse vraisemblable en stéganographie qu'aucune contrainte de robustesse ne pèse sur notre méthode : nous souhaitons uniquement illustrer, et deux interlocuteurs utilisant un tel canal stéganographique peuvent facilement s'arranger pour que l'objet ne subisse pas de distorsion (en attachement d'un mail par exemple) : le tout est qu'il continue d'avoir l'air d'un objet innocent, ce qui est aussi censé le protéger d'une certaine manière. Nous pouvons donc sélectionner pour premier triangle du ruban celui qui nous agrée le plus (par exemple le plus petit). Arbitrairement, nous choisissons de prendre le tout premier triangle décrit dans l'information de connexité.

Une fois le premier triangle dans le ruban trouvé, il faut fixer le terme du ruban. Pour cela, deux solutions sont généralement proposées : soit on identifie une séquence connue de sortie, et on s'arrête dans le parcours lorsqu'on retrouve cette séquence, soit on connait la quantité d'information cachée à retrouver, et on s'arrête lorsqu'on l'a atteinte. Dans le cas présent, nous supposerons connue une telle taille, et nous arrêterons de relire l'information cachée lorsque suffisamment de bits auront été décodés. Le but étant ici l'illustration des performances de la primitive géométrique ainsi que le détail de la mise en œuvre qu'elle implique, on pourra se contenter d'un nombre faible de bits cachés (quelques kilobits). Plus loin,

[77] La stéganographie LSB consiste à remplacer les trois ou quatre plans de bits de poids faible de l'image pour y cacher l'information.

nous ajouterons d'autres briques non moins essentielles comme un arrangement indexé, et un parcours optimal du graphe de connexité au regard d'une certaine contrainte de causalité sur le parcours (comme dans [82]).

Ruban et sécurité

A présent que les deux extrémités du ruban de triangles sont déterminées, il importe de savoir comment construire ce ruban. Nous fixons les deux exigences suivantes concernant ce ruban :

- il doit définir l'ordre implicite des bits d'information cachée,
- c'est sur lui que repose le secret de l'information cachée.

La première exigence découle de la transposition des techniques de stéganographie LSB dans le cas qui nous préoccupe, tandis que la seconde compose avec l'information arbitraire de connexité. Dans le cas de la stéganographie LSB, le pixel suivant dans le ruban de pixels est implicite, alors que dans le cas des maillages triangulés, deux choix sont possibles. Nous expliquons à présent pourquoi.

La nature d'un ruban consiste à établir une suite d'éléments ordonnés entre eux : pour chaque élément du ruban, on doit pouvoir déduire quel sera l'élément suivant. Dans le cas de la stéganographie LSB, cette déduction s'appuie uniquement sur la grille d'échantillonnage des pixels, la déduction est implicite. Dans le cas des triangles, il nous faut préciser ses attributs vis-à-vis de leur appartenance au ruban. Nous considérons un triangle comme ayant un *côté d'entrée*, et également un *côté de sortie*. Ainsi, si le côté d'entrée est connu, il reste deux possibilités pour le côté de sortie. Afin de fixer le côté de sortie pour chaque triangle dans le ruban, nous faisons appel à un générateur de nombres binaires pseudo-aléatoires.

Un triangle étant donné avec son côté d'entrée, un appel au générateur binaire pseudo-aléatoire déterminera lequel des deux autres côtés utiliser pour fixer le côté de sortie. Naturellement, nous avons besoin pour cela de fixer un sens d'orientation. Pour cela, nous nous appuyons sur l'orientation horaire autour de la normale au triangle. Ainsi, par rapport au côté d'entrée, nous savons parmi les deux côtés de sortie possibles lequel sera le premier, et lequel sera le second. Nous récapitulons cette manière de voir dans la Fig 5.5. Naturellement, le côté de sortie d'un triangle devient le côté d'entrée du prochain triangle dans le ruban, et on construit ainsi un ruban aussi long que l'objet le permet. Si un bord se présente, on détecte un triangle n'ayant alors qu'un seul côté possible de sortie, que nous empruntons sans incrémenter notre compteur de numéros de bits.

De cette manière, le ruban est parfaitement déterminé, et donc l'ordre dans lequel les bits sont cachés. En outre, puisque le choix du triangle suivant le ruban dépend d'un générateur pseudo-aléatoire, on est en mesure de garantir une sécurité accrue, puisque reposant sur les numéros de bits cachés. De plus, il suffit qu'un attaquant commette une seule erreur dans le choix du triangle suivant pour que tout le reste du ruban qu'il tente de constituer soit erronné. Bien entendu, afin de rendre la stéganalyse encore plus difficile, il est nécessaire de crypter l'information à cacher de manière à ce qu'un stéganalyste ne puisse s'aider de la sémantique de l'information cachée. Nous représentons sur la Fig 5.6 un ruban d'une capacité de 4Kbits sur le modèle *bunny*.

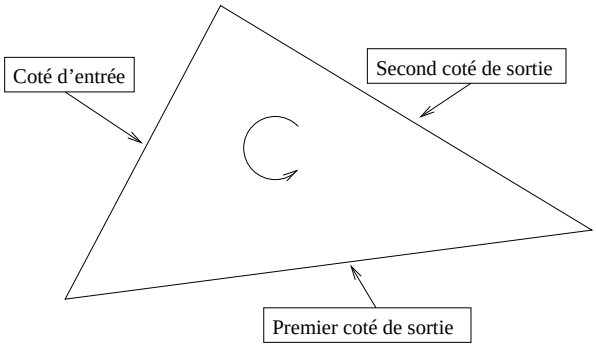

FIG. 5.5 – Comment déterminer le triangle suivant dans le ruban, étant donné le côté d'entrée ? L'ordre des deux côtés de sortie est déterminé par rapport à l'orientation de la normale au triangle. Sur la figure, la normale est dirigée vers le lecteur.

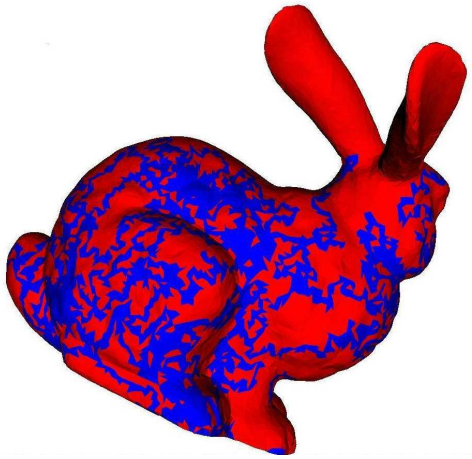

FIG. 5.6 – Exemple de ruban d'information cachée. Le ruban contient 4096 triangles, soit 4Kbits d'information cachée.

Implantation

La seule précaution à prendre désormais est de garantir que la sélection pseudo-aléatoire des triangles du ruban ne fasse pas en sorte qu'on vienne effacer ce que l'on a déjà écrit. Pour cela, il faut s'arranger pour qu'aucun point déjà dans le ruban ne puisse être modifié lors d'une itération ultérieure. Cela est possible de deux façons, voir Fig. 5.7 :

- Cas 1 : Tout d'abord, un sommet déjà tatoué lors de l'inscription d'un bit i pourrait à nouveau être candidat à un déplacement pour un bit $i + k$ si le parcours le ramenait en position d'être choisi. Le bit i serait alors potentiellement erroné après l'écriture du bit $i + k$,
- Cas 2 : Mais aussi un sommet ayant servi de base AB au tatouage du bit i pourrait être déplacé lors du tatouage du bit $i + k$. A la relecture le bit serait potentiellement erroné puisque la base servant au calcul de la projection de C sur AB serait modifiée.

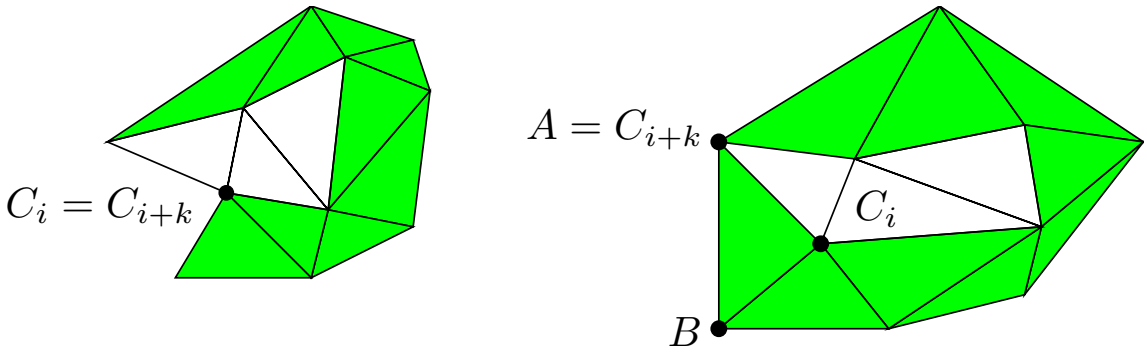

FIG. 5.7 – Les deux cas pouvant entrainer l'effacement du tatouage. A gauche : cas 1. A droite : cas 2.

Au fur et à mesure que l'on crée le ruban, on crée une liste des points interdits pour une future modification. Si un futur site nécessite de modifier les coordonnées d'un sommet déjà interdit, on le déclare invalide, et on poursuit la construction du ruban jusqu'à ce que l'on trouve un triangle valide. Chaque triangle valide pour l'insertion de données cachées voit son point C ajouté à la liste des points interdits : comme A et B appartenaient au triangle précédent, ils étaient invalidés auparavant et les trois sommets A, B et C sont dorénavant invalidés. De cette manière, tous les points du rubans sont interdits. Bien évidemment, on constitue la même liste à la relecture afin de garantir la synchronisation entre l'insertion et l'extraction. En procédant ainsi, la taille du ruban produit peut excéder la taille du message à cacher. Nous donnons ci-dessous le pseudo-code de l'algorithme :

```
Initialisation de l'arête initiale (marquer A et B interdits)

TantQue il existe encore un bit à écrire ou à relire
  Si C est un voisin non interdit de A et B
    Écrire ou relire la valeur du bit dans ABC
    Marquer C comme interdit
  Fin
Décider pseudo-aléatoirement si B ← C ou A ← C
FinTantQue
```

La remarque la plus importante est que si un sommet dans lequel on tente de cacher de l'information est déjà interdit, on passe au triangle suivant sans rien faire. Ainsi, la longueur du ruban peut très bien excèder, de beaucoup, la longueur en bits de la marque que l'on souhaite insérer. La différence est le nombre de triangle qu'il a fallu parcourir afin de trouver à nouveau un sommet non encore interdit.

Remarque sur la sécurité

Même munie d'un module de diffusion d'erreur, la stéganographie LSB de l'image 2D produit encore des distributions suspectes de niveaux de gris jusque dans les 3ème et 4ème plans de bits, rendant l'image suspecte de contenir de l'information cachée. La méthode de

stéganographie que nous évoquons, fondé sur un parcours *pseudo-aléatoire*, ne *peut* pas présenter ce genre de faille. En effet, comme discuté précédemment, la détectabilité du message caché repose uniquement sur l'information qu'un pirate pourrait trouver sur l'ordre des bits : l'insertion géométrique de la valeur des bits par symétrie n'aide en rien le pirate. Or, si l'on rend le parcours pseudo-aléatoire, guidé par une clef secrète, et que l'on choisit une finesse qui rende l'insertion imperceptible, il devient improbable de détecter la simple présence d'un message caché.

Avec une telle approche, nous ne voyons pas comment un stéganalyste pourrait seulement déterminer si un maillage contient de l'information cachée ou pas. En effet, ne pouvant déterminer quels sommets furent modifiés, et dans quel ordre, par la procédure géométrique, il n'aurait d'autre alternative que de tenter de reconstituer un éventuel parcours (nous raisonnons ici au pire des cas : avec un message caché non crypté. Le stéganalyste pourrait ainsi s'aider d'une éventuelle sémantique en clair pour faire progresser son étude. Il va de soit que le chiffrage du contenu à cacher prive le stéganalyste d'une telle ressource, et intervient comme toujours en *complément* des techniques de tatouage). Deux composantes rentrent dans la détermination du parcours :

– la clef secrète qui initialise le générateur pseudo-aléatoire guidant le parcours en ruban,
– l'arête initiale du ruban et le premier point à tatouer,

En général, la clef secrète mesure 64 bits. Enfin, le choix de l'arête initiale est fonction de la taille du maillage : en notant e le nombre d'arêtes du maillage, cette partie du parcours secret repose sur $1 + \lfloor \log_2(e) \rfloor$ bits. La longueur de clef utilisée pour déterminer le parcours secret du ruban est donc de $65 + \lfloor \log_2(e) \rfloor$ bits. Encore cette valeur repose-t-elle sur un raisonnement supposant que le cryptanalyste dispose d'une sémantique pour s'aider.

Remarque pratique

En reconsidérant notre procédure géométrique d'insertion de la valeur, il apparaît en pratique que si le point C est déjà très proche d'un plan de symétrie, on risque l'instabilité numérique dans les calculs. Ce cas se produit toutefois rarement. En l'état, nous détectons une exception de virgule flottante et annulons la procédure : le bit n'est pas inséré. Le parcours précédent, s'il devait servir à une application de stéganographie réelle et non à une évaluation de performances, devrait proposer une répétition de chaque bit afin de tenir compte de ces exceptions.

Toutefois, nous ne devons pas perdre de vue que l'objectif de notre propos est la création d'un algorithme de tatouage fragile, pour lequel ces quelques exceptions seront tolérables statistiquement. En conséquence, il n'est pas nécessaire de s'appesantir sur cet aspect, mais il se révèlera plus profitable (voir section 5.4) de bien dégager les caractéristiques du parcours en ruban.

Densité locale d'information

Dans le meilleur des cas, un triangle peut voir ses trois sommets contribuer à l'insertion de l'information cachée. Dans le pire des cas, aucun de ses sommets ne participe au tatouage. Entre les deux extrêmes, un triangle peut voir un ou deux de ses sommets coder de l'information. En effet, nous montrons sur la Fig. 5.8 comment il est possible que les trois sommets

d'un triangle codent tous des bits du tatouage. On a donc quatre degrés possibles de densité locale d'information, suivant le nombre de sommets d'un triangle qui codent effectivement de l'information cachée.

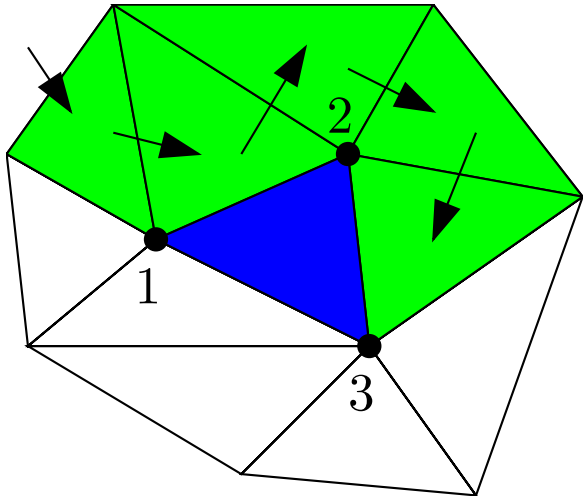

FIG. 5.8 – Exemple de parcours permettant au triangle central de voir ses trois sommets participer au codage de l'information cachée.

Nous donnons sur la Fig. 5.9 deux exemples de maillages sur lesquels un ruban d'information cachée a été inscrit : la densité locale d'information a également été peinte en différentes couleurs (quatre).

Le principal écueil de cette méthode est de ne pas pouvoir garantir le temps d'insertion du fait de la sélection pseudo-aléatoire des triangles dans le ruban, et de leur éventuelle admissibilité suivant que leur point C est déjà ou pas dans la liste de points interdits (le ruban en train de se construire). De fait, lorsque l'on cherche à cacher un nombre de bits proche du nombre de points dans le maillage, le temps de calcul explose car trouver aléatoirement un sommet non interdit devient de moins en moins fréquent au fur et à mesure que l'on a déjà caché de l'information. Nous proposons ci-dessous un moyen de corriger légèrement ce problème. En pratique, on ne peut pas atteindre la capacité maximale, mais on peut tout de même se servir d'environ 75% des points pour cacher de l'information, ce qui reste parfaitement suffisant pour les buts d'illustration que nous nous sommes fixés.

5.2.4 Un arrangement local

Une première manière de casser l'explosion du temps de recherche d'un sommet non encore interdit consiste à utiliser plusieurs rubans, avec bien entendu la même liste de points interdits pour tous. Cela revient en pratique à partitionner le maillage en sous-parties qui ne se recouvrent pas, au sens de l'insertion. Il s'agit donc d'un arrangement local. Nous choisissons de le mettre ici en œuvre essentiellement à des fins d'illustration. Nous verrons en section 5.4 une deuxième manière de parcourir le maillage en temps acceptable car linéaire. La chronologie de ces travaux a voulu que nous nous intéressions au parcours du graphe

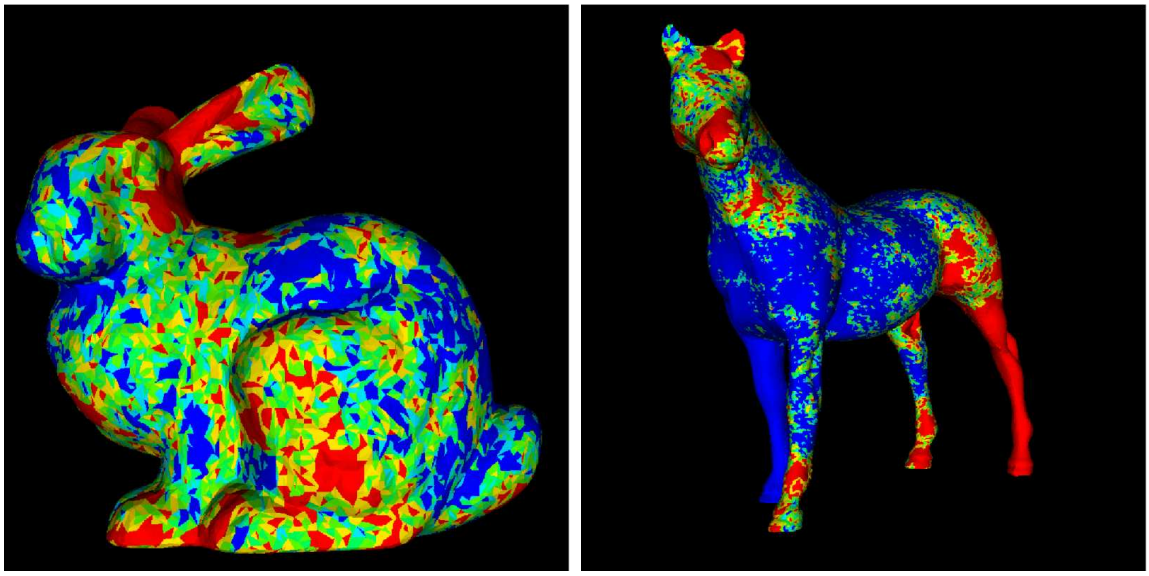

FIG. 5.9 – Maillage *bunny* contenant 12Kbits d'information cachée et maillage *horse* contenant 40Kbits d'information cachée. Chacune des quatre couleurs du ruban correspond au nombre de sommets du triangle utilisés pour coder des bits bleu=3 (saturées), vert= 2, jaune= 1, rouge= 0 (vides).

après avoir bien défini notre procédure géométrique, d'où l'utilisation de parcours naïfs pour l'illustration.

Stratégie

Nous procédons comme suit : lorsque nous détectons un trop grand nombre de triangles non admissibles à la suite, nous lançons un autre ruban sur le maillage, avec la même liste de points interdits. Cette liste matérialisera donc cette fois l'*ensemble* des rubans lancés sur le maillage. Expérimentalement, ce nombre limite de triangles a été fixé à 50. Cet ajustement ne permet pas de gagner en capacité, mais en temps de calcul. Nous représentons sur la Fig. 5.10 l'allure du temps de calcul nécessaire pour cacher 12Kbits d'information dans le modèle *bunny*.

Nous donnons sur la Fig. 5.11 le résultat de tests de capacité effectués sur onze maillages, et sur la Fig. 5.9, nous montrons deux des modèles après insertion de plusieurs Kbits d'information cachée.

Résultats et mesures

En pratique, nous avons donc utilisé un arrangement local pour cacher l'information. Nous définissons encore quelques mesures permettant de mieux appréhender le comportement de l'algorithme. On souhaite caractériser les performances de l'algorithme en termes de capacité et de distorsion, et pouvoir donner une idée de l'efficacité du codage dans le temps.

Pour caractériser les performances de capacité, nous proposons une borne supérieure.

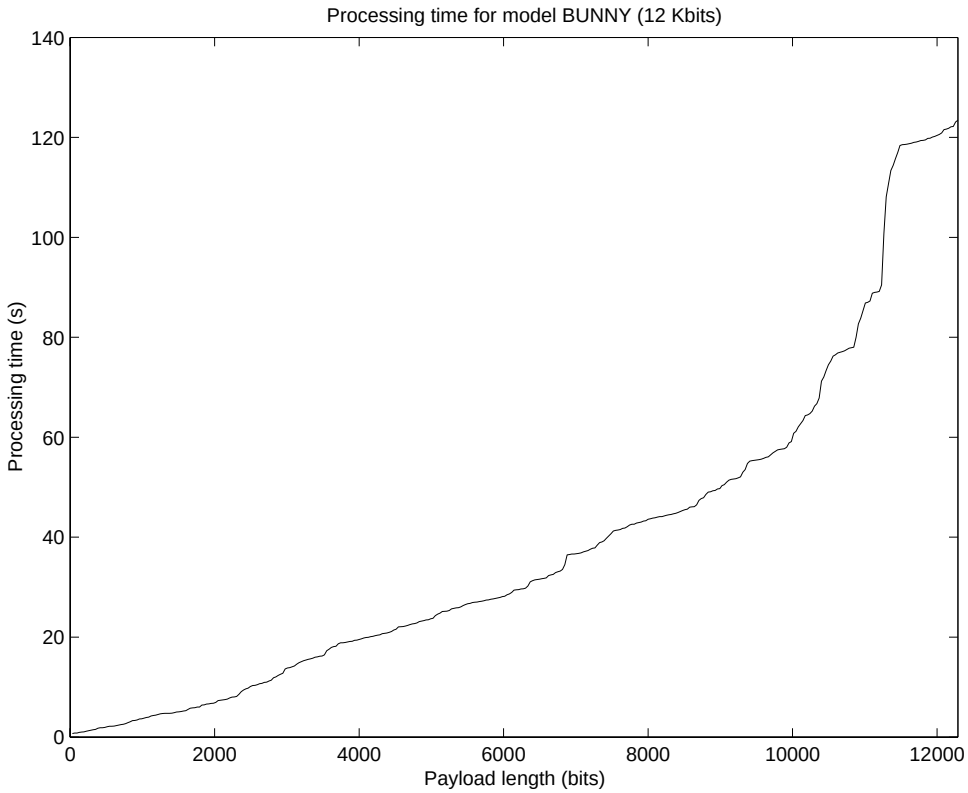

FIG. 5.10 – Temps de calcul (Pentium/800MHz) pour cacher 12Kbits dans le modèle *bunny*. Les discontinuités de la courbe correspondent au lancement d'un nouveau ruban sur le maillage. Un nouveau ruban est lancé chaque fois que 50 triangles non admissibles de suite sont visités.

Notre algorithme nécessitant de fixer les deux premiers points du ruban, ces deux points seulement ne pourront jamais être perturbés pour cacher de l'information. Si tous les sommets étaient atteignables, la capacité maximale théorique pour ce schéma serait donc de $N-2$ bits, car on vient cacher un bit dans chaque sommet libre. Toutefois, il s'agit d'une borne supérieure, jamais atteinte en pratique par un tel parcours simpliste (nous nous consacrerons par la suite à l'élaboration d'un parcours davantage adapté à notre primitive géométrique d'insertion). On quantifie le taux de remplissage du maillage par :

$$R_{capa} = \frac{N_{bits}}{N-2}, \tag{5.1}$$

où N_{bits} est le nombre de bits cachés. En pratique, nous avons procédé par sondage de la capacité pour savoir quelle quantité d'information nous pourrions cacher : nous avons cherché à cacher beaucoup car ainsi davantage de sommets contribuent à modifer le maillage, et l'évaluation de la procédure géométrique est d'autant plus valable. Nous présentons les résultats pour deux maillages à la topologie complexe (*skull* et *pelvis*, voir Fig. 5.11) : ce sont les objets les moins favorables car le ruban s'engouffre souvent dans une anse et peine à en ressortir. Nous avons arrêté le sondage de la capacité du canal lorsque le temps de calcul dépassait 3 minutes.

Dans le but de quantifier la distorsion introduite par le tatouage, nous avons choisi le rapport entre la plus grande perturbation due au tatouage (une distance d_{max}) et la moyenne des longueurs des arêtes du maillage l_{moy} :

$$R_{dist} = \frac{d_{max}}{l_{moy}}. \tag{5.2}$$

Enfin, nous voulons avoir une idée du chemin parcouru en trop. En effet, dans le meilleur des cas, il ne faudra parcourir que N_{bits} triangles pour trouver tous les sites nécessaire au codage. Or, l'aléa introduit dans le parcours fait que l'algorithme retombe sur des points invalidés car déjà porteurs d'information. Il faut donc faire un peu de chemin pour retomber sur des sites encore libres. Pour quantifier cela, nous mettons en rapport le nombre de triangles $N_{parcours}$ qu'il aura finalement fallu parcourir pour cacher toute l'information avec le nombre de bits cachés :

$$R_{cod} = \frac{N_{parcours}}{N_{bits}}. \tag{5.3}$$

Précisons que nous donnerons en section 5.4 un algorithme permettant un parcours optimal du maillage en $O(N)$. Nous donnons sur la tableau 5.11 les résultats des sondages de capacité de 11 maillages, avec les mesures introduites plus haut. On notera qu'il n'est pas rare d'obtenir des taux de remplissage voisinant les 80%. Cela se produit surtout sur les maillages avec une topologie simple, éventuellement avec bords.

Objet	N	R_{capa}	R_{cod}	R_{dist}	N_{bits} (Kbits)	Remarque
horse	48485	84.5%	6.5	0.15%	40	genre 0
bouddha	32328	79.2%	7.1	0.19%	25	trous
face	26460	77.4%	7.3	0.19%	20	1 bord
bunny	14007	87.7%	8.4	0.22%	12	1 bord
inopl	9831	83.3%	7.5	0.23%	8	1 bord
venus	8016	89.4%	8.1	0.22%	7	genre 0
hand	6650	61.6%	6.3	0.21%	4	genre 0
woman	7723	53.0%	6.2	0.23%	4	genre 0
anttoon	7082	57.8%	5.9	0.18%	4	genre 2
skull	11051	37.1%	7.8	0.24%	4	genre élevé, trous
pelvis	4189	48.9%	7.6	0.25%	2	genre élevé, trous

FIG. 5.11 – Résultats obtenus sur onze maillages en stéganographie avec arrangement local avec une finesse de 32.

Nous avons encore vérifié que les bits sont correctement relus après une rotation, une translation ou une mise à l'échelle uniforme. En effet, c'étaient les classes de transformation auxquelles notre primitive était censée résister en raison de l'invariant géométrique manipulé.

Conclusion

Nous avons donc validé notre primitive géométrique d'insertion de la valeur des bits à cacher, au moyen d'une méthode simple de stéganographie pour les maillages surfaciques

3D triangulés orientables. Cette méthode présente les avantages d'être robuste face à la rotation, la translation et la mise à l'échelle uniforme, et d'introduire le secret dans le numéro des bits cachés. Par contre, elle a l'inconvénient de nécessiter un sondage préalable de la capacité du canal stéganographique. Ce problème sera résolu dans la section 5.4. Au moins cet inconvénient met-il l'accent sur une problématique que nous aborderons plus loin lors de la conception d'un parcours adapté.

5.3 Arrangement indexé pour le tatouage fragile

A des fins d'authentification des maillages, nous avons souhaité développer une méthode de tatouage fragile basée sur la procédure géométrique décrite plus haut. La procédure géométrique fournit un certain nombre d'invariances, mais nous souhaitons pouvoir récupérer la marque y compris sur des extraits de maillage. Ainsi, nous souhaitons une robustesse face aux opérations suivantes :

- translation
- rotation
- mise à l'échelle uniforme
- coupe

Si l'invariant géométrique retenu pour notre procédure d'insertion offre naturellement la robustesse face aux trois premières opérations, il reste à mettre en oeuvre une stratégie particulière afin de s'affranchir autant que possible de la coupe. Pour cela, nous avons mis en œuvre un arrangement indexé.

5.3.1 Surcoût de codage

Nous rappelons que nous voulons cacher 64 bits (leur numéro et leur valeur). L'utilisation d'un arrangement indexé entraine donc un surcoût de codage, dû aux numéros. Si l'on veut cacher C_{utile} bits, il faudra en réalité en inscrire :

$$C_{total} = C_{utile} \times (1 + \log_2(C_{utile})), \tag{5.4}$$

sur le maillage. Dans toute la suite de ce travail, et pour conserver le point de vue généralement admis, nous continuerons de parler de capacité *utile* : lorsque nous cacherons 64 bits utiles, il faudra garder à l'esprit qu'en réalité il aura fallu en inscrire 448.

Enfin, il est une autre information à inscrire, qui ne représente pas la marque mais qui valide une configuration géométrique comme tatouée. En effet, il faut savoir reconnaître un site tatoué en tant que tel avant de chercher à y décoder quoi que ce soit. Nous verrons comment ce problème trouve naturellement sa solution.

5.3.2 Stratégie pour le choix d'un autre invariant

Nous nous proposons donc de coder à présent aussi les numéros des bits par la manipulation d'un invariant géométrique. La manipulation de cet invariant devra être plus fine que celle proposée pour le codage de la valeur des bits : on cherche ici à cacher les $\log_2(C_{utile})$ bits

qui coderont leur numéro. Ainsi, nous pouvons représenter sur la Fig. 5.12 l'allure des surfaces isovaleurs (pour les numéros) de quelques invariants. Nous n'avons souhaité considérer que des invariants pouvant être calculés dans un seul triangle.

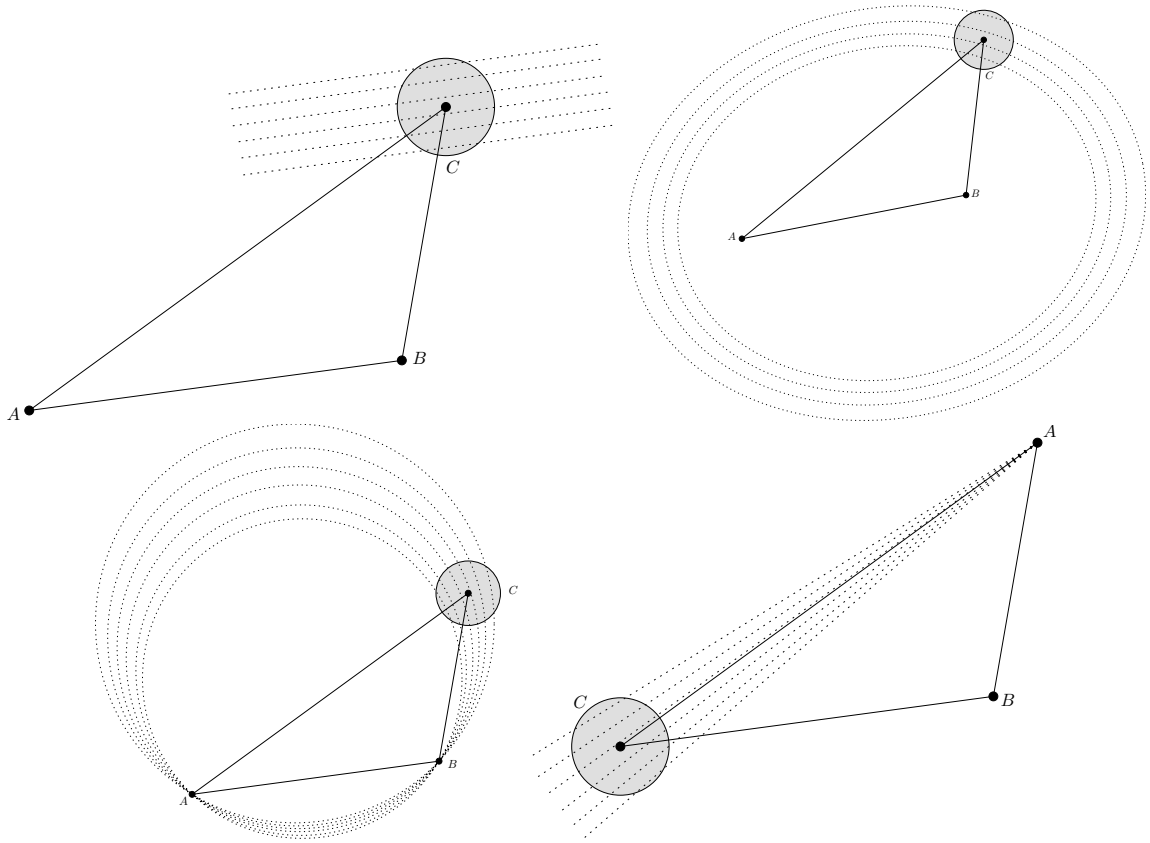

FIG. 5.12 – Allure des isovaleurs en tatouage suivant la modification de trois invariants. En haut à gauche : en modifiant l'aire en jouant sur la hauteur. En haut à droite : en modifiant le périmètre. En bas : en modifiant l'angle de valeur médiane (à gauche s'il est en C, à droite s'il est en A). Nous avons choisi de modifier des rapports d'aires (en haut à gauche).

Enfin, il faut remarquer que la procédure géométrique d'insertion de la valeur du bit modifie le point à perturber *parallèlement* aux deux points de référence de la base du triangle. Ainsi, si nous trouvons une autre procédure qui perturbe le point dans le plan *perpendiculaire* à la base du triangle et contenant le sommet à modifier, nous disposons d'un autre degré de liberté dans le codage. Manifestement, le premier invariant de la Fig. 5.12, fondé sur la manipulation de l'aire, convient parfaitement : il autorise une manipulation géométrique pour l'insertion du numéro *indépendante* de celle dédiée à l'insertion de la valeur du bit. Il s'agit en quelque sorte d'un codage CDMA (Code Division Multiple Access) géométrique de l'information.

5.3.3 Codage des numéros des bits par invariant

L'invariant que nous avons retenu est le rapport des aires de deux polygones. Les deux polygones qui nous intéressent sont le triangle (ABC) (d'aire S), et le carré imaginaire ayant $[AB]$ pour côté. Notre second invariant est donc : $\frac{AB^2}{S}$. En modifiant la hauteur du triangle (ABC) dans un plan perpendiculaire à (AB), on ne change pas l'aire du carré imaginaire, et seule l'aire du triangle s'en trouve modifiée (voir Fig. 5.13). Soit H la hauteur s'appuyant sur (AB), on utilise la relation triviale :

$$H \times AB = 2 \times S. \tag{5.5}$$

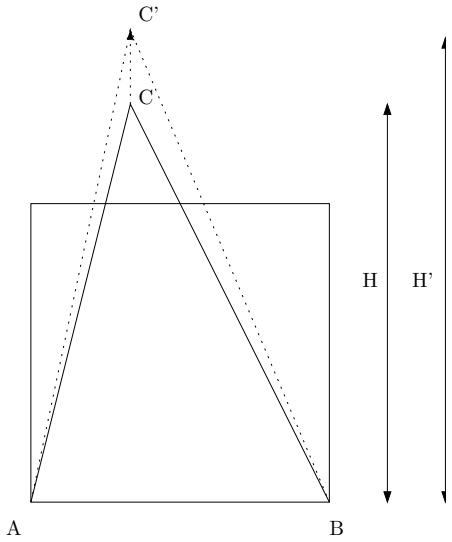

FIG. 5.13 – L'invariant utilisé pour coder le numéro des bits du message : le rapport de l'aire du carré imaginaire de côté $[AB]$ sur l'aire du triangle ABC. En modifiant la hauteur du triangle issue de $[AB]$, on ajuste l'aire du triangle sans toucher à celle du carré.

C'est donc en réalité en modifiant la hauteur du triangle s'appuyant sur AB que l'on vient perturber l'invariant $\frac{AB^2}{S}$.

Modulation secrète de l'invariant

La primitive géométrique d'insertion de la valeur ne cale pas le point C à modifier sur une valeur précise : une large plage de variation correspondant à la finesse utilisée est admise. Or, en procédant par symétrie, on ne vient pas introduire de perturbation statistique. Par contre, lorsque nous utilisons un arrangement indexé, il faut comme nous l'avons mentionné plus haut, pouvoir identifier un site tatoué comme tel.

S'il est facile à un pirate de trouver les sites tatoués, il tentera probablement ensuite d'obtenir une bonne approximation de la clef. Afin de cacher l'information de validité d'un site comme tatoué à un éventuel attaquant, nous modulons secrètement l'espace dans lequel la décision a lieu. Nous évaluerons plus loin la sécurité qu'offre ce procédé.

Soit K une clef codée sur N_K bits servant à paramétrer le processus d'insertion. Cette clef est composée par concaténation de deux clefs K_1 et K_2 codées respectivement sur N_{K_1} et N_{K_2} bits. On a également besoin de deux paramètres internes T_1 et T_2 dont nous verrons plus loin comment leur valeur a été fixée (dans la section 5.5.3). La modulation secrète en question associe une grandeur g_{ABC} au triangle (ABC) :

$$g_{ABC} = \left(T_1 + \frac{K_1}{2^{N_{K_1}}}\right)^{\left(T_2 + \frac{K_2}{2^{N_{K_2}}}\right)} \times \frac{AB^2}{S}. \tag{5.6}$$

La clef K sert à paramétrer secrètement l'étape d'écriture du numéro du bit dans le triangle. Ainsi, si le message caché contient C_{utile} bits, le numéro n du bit contenu dans le triangle est donné par :

$$n = \lfloor g_{ABC} \rfloor \% C_{utile}, \tag{5.7}$$

où % représente l'opérateur modulo.

Marquage d'un triangle comme tatoué

Pour autant, la nouvelle valeur de $g_{ABC'}$ lors de l'insertion n'est toujours pas fixée exactement. Pour lever l'indétermination, nous choisissons de faire coder à g_{ABC} l'information de validité d'un site tatoué. Un triangle sera déclaré porteur d'information cachée si g_{ABC} est placée sur l'échelle des entiers en modifiant la hauteur H telle que :

$$|g_{ABC} - \frac{1}{2} - \lfloor g_{ABC} \rfloor| < \epsilon. \tag{5.8}$$

Nous retrouvons ici un procédé classique en tatouage substitutif : caler une valeur sur la médiane d'un intervalle. En général, on procède ainsi pour augmenter la robustesse du schéma de tatouage face à la quantification. Mais même si nous pouvons éventuellement tirer parti d'une telle stratégie en terme de robustesse, il faut reconnaître que nous cherchons uniquement à marquer un triangle comme tatoué.

5.3.4 Validation de l'approche par arrangement indexé

Nous souhaitons vérifier ici la pertinence de l'arrangement indexé. Nous rappelons qu'un tel arrangement doit permettre de retrouver la marque en décodant les triangles dans n'importe quel ordre. Pour cela, chaque triangle est positionné de telle manière que AB soit son plus petit côté, ceci afin de faciliter la relecture en ne réclamant pas d'examiner les trois permutations circulaires possibles pour le choix de A, B et C. Lors de l'insertion, chaque triangle tatoué voit ses points interdits, afin de pallier les risques d'effacement de la Fig. 5.7.

A la relecture, nous énumérons encore les triangles un par un, les positionnons sur leur base de référence (AB est le côté le plus court), et si le triangle est reconnu porteur d'information on décode le contenu du bit caché (numéro et valeur). Comme on ne peut garantir le nombre de sites tatoués dès l'insertion (pour modéliser le comportement statistique du détecteur), il faut se résigner à adopter un processus de décision à la majorité pour le décodeur.

Robustesse face à l'attaque de coupe

Nous avons mis en œuvre un arrangement indexé précisément dans le but de nous affranchir de l'attaque de coupe. C'est donc face à une telle attaque de coupe que nous avons évalué les performances de l'arrangement indexé : nous avons fait varier le nombre de triangles disponibles pour la relecture et avons noté en fonction la proportion de bits correctement relus. Les résultats portent sur une moyenne de trois maillages. Il faut donc près de 40% des triangles pour espérer relire la marque. C'est encore très insuffisant - nous ferons mieux par la suite (section 5.4) - mais cela démontre la validité de l'approche par arrangement indexé. Pour la détection, nous avons requis un rapport minimum de trois des deux occurences avec lesquelles les valeurs binaires ont été détectées (rapport entre la plus grande et la plus petite occurence) pour chaque numéro.

La prochaine section concerne l'élaboration d'un parcours maximisant le nombre de sites atteints, et permettant de bien meilleures performances face à l'attaque de coupe. Mais avant, il nous faut examiner en quoi notre façon de passer en revue les triangles un par un à l'insertion est contre-productive en terme de répétition de l'information cachée.

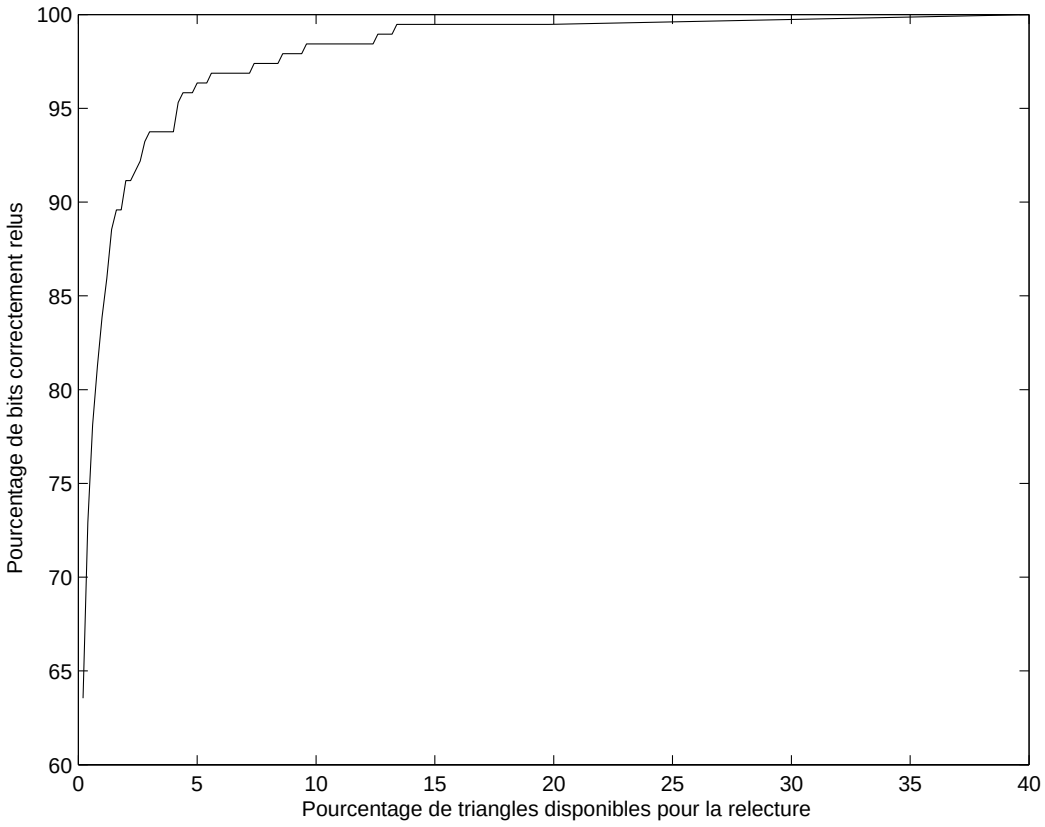

FIG. 5.14 – Attaque de coupe avec un parcours naïf du maillage : 40% du maillage original est requis afin de détecter correctement la marque.

Localisation de l'information cachée

Nous montrons sur la Fig. 5.15 la répartition des sites tatoués lorsque les triangles sont naïvement passés en revue les uns après les autres. En particulier, on constate que la surface contient encore de nombreuses zones non tatouées.

Cette mauvaise utilisation de la surface a deux origines :

- *Le passage en revue des triangles un par un à l'insertion* : en n'utilisant pas l'information de connexité pour se déplacer, nous n'optimisons pas le nombre de sites atteints. Dans la section suivante, nous développerons un parcours adapté fondé sur la croissance d'une liste active,
- *La convention qui fait de AB le plus petit côté d'un triangle* : pour faciliter la relecture, nous avons introduit une convention qui se révèle contre-productive car elle fait peser une contrainte de plus sur l'admissibilité des sites. Dans la section suivante, nous abandonnerons cette convention : le temps de relecture sera multiplié par trois (pour tester les trois permutations circulaires sur A, B et C), mais de par la nature de l'arrangement indexé, restera linéaire en nombre de triangles.

Fig. 5.15 – Répartition de l'information cachée lorsque l'on cherche les sites en énumérant les triangles un par un. Les triangles en vert sont porteurs d'information cachée, les triangles rouges ne codent pas d'information cachée. Le maillage contient encore une importante surface ne contenant pas d'information cachée. En outre, le nombre de sites utilisés est imprévisible.

5.4 Optimisation du nombre de sites

Nous nous consacrons à présent à l'élaboration d'un parcours optimal du maillage, au sens du nombre de sites atteints pour le tatouage. C'est ce parcours qui nous permettra de construire les autres éléments d'un schéma de tatouage fragile, comme une modélisation statistique de la confiance dans la sortie du détecteur. Comme nous venons de l'évoquer, nous abandonnons la contrainte qui demande que AB soit le plus petit côté du triangle (le temps de calcul restant linéaire) pour bénéficier d'un degré de liberté supplémentaire dans la diffusion de la marque.

A cette fin, nous rappelons que la seule contrainte pesant sur notre méthode d'insertion concerne la causalité : il faut s'assurer de ne jamais venir réécrire là où l'on a déjà écrit, et éviter les causes d'effacement qui sont exposées en Fig. 5.7. Le schéma de Yeo et Yeung [82] partage cette contrainte d'un parcours spécial respectant la causalité (les auteurs n'utilisent pas l'arrangement indexé pour s'affranchir de la coupe, et doivent donc, contrairement à nous, reproduire le même parcours à l'insertion et à la relecture). C'est pour cette raison que nous utilisons un jeu d'étiquettes pour marquer les sommets interdits. L'idée dont nous nous inspirons est tirée de la méthode de compression statique de Touma et Gotsman [73] (section 3.1.2). Le parcours de leur graphe de connexité est implicite, comme nous l'avons vu précédemment, et repose sur la croissance d'une région modélisée par une liste d'arêtes formant un cycle (plusieurs cycles dans le cas de surface avec genre non nul).

Nous allons présenter un parcours fondé sur la croissance d'une liste active de sommets (dont les arêtes qui les relient forment également un cycle). Nous montrerons qu'à chaque itération, on dispose d'un sommet admissible pour l'insertion. Enfin, nous donnerons le nombre de sites atteints par l'algorithme, dont nous prouverons qu'il tourne en temps linéaire sur des surfaces de genre nul, sans y être limité.

5.4.1 Algorithme

Nous exposons ici un algorithme permettant d'atteindre tous les sites possibles du maillage au regard de notre procédure géométrique d'insertion. Un point traversé (ou conquis) devient toujours *interdit*, mais nous devons encore initialiser l'algorithme en interdisant les deux sommets de la première arête choisie arbitrairement. Un triangle dont tous les sommets sont interdits est dit *tatoué*. Nous verrons qu'à proprement parler, l'implantation que nous discutons dans les résultats ne permet pas de cacher de l'information dans chacun des triangles (notre réflexion se fonde sur les sommets). Cela est dû à notre procédure géométrique, mais il n'en reste pas moins que le parcours que nous présentons ouvre tout de même la voie au tatouage de *tous* les triangles - et c'est pourquoi nous conservons cette appellation de triangle tatoué. Soient A et B les deux sommets de l'arête initiale dont on fait l'embryon de notre "liste active".

```
Initialisation de l'arête initiale (marquer A et B interdits)

TantQue il existe un point non interdit
  Si ABC est un triangle orienté de S tel que
       A et B soient interdits, mais pas C
    Insérer l'information dans ABC
    Marquer C comme interdit
  Fin
FinTantQue
```

On ne peut exprimer plus simplement la contrainte de causalité de la procédure géométrique d'insertion : dans tout triangle ABC candidat à l'insertion, il faut un point non interdit (C) et il faut, afin de maximiser le nombre de sites, que les deux autres (A et B) soient déja interdits.

5.4.2 Preuve de terminaison

Notre algorithme contient une boucle `TantQue`. Nous devons donc nous assurer qu'il termine bien. S'il termine, c'est que tous les points auront pu servir à cacher de l'information, sauf les deux premiers sommets de l'arête initiale. Nous apportons ici la preuve que notre algorithme atteint exactement $N - 2$ sites valides pour l'insertion.

Lemme

Tant qu'il existe des sommets libres, à chaque itération de l'algorithme on trouve un triangle valide pour l'insertion, c'est-à-dire avec deux points interdits et un point libre (non interdit).

Preuve

Soit q un point libre n'appartenant pas à un triangle avec deux points interdits (voir Fig. 5.16). On peut alors trouver un chemin menant de q à un point déjà interdit (avant la première itération, les deux points de l'arête initiale sont déjà interdits).

Ce chemin possède nécessairement (au moins) une arête avec un sommet interdit, q'', et l'autre encore libre, q'. En tournant autour de q'', on crée un autre chemin qui contient nécessairement un autre point interdit r, car un point n'est interdit que *par rapport à deux voisins déjà interdits*, et un point encore libre s (éventuellement $s = q'$).

Le triangle dans lequel cacher l'information est donc $sq''r$, avec s dans le rôle du sommet non interdit.

Implantation

Afin d'accélérer la recherche du triangle dans lequel insérer l'information, on maintient à jour la liste des sommets appartenant à (au moins) un triangle avec ses deux autres points interdits. On procède comme suit : lorsqu'un point est sur le point d'être interdit, on parcourt la liste de tous les triangles incidents et on détecte ceux pour lesquels un seul sommet est

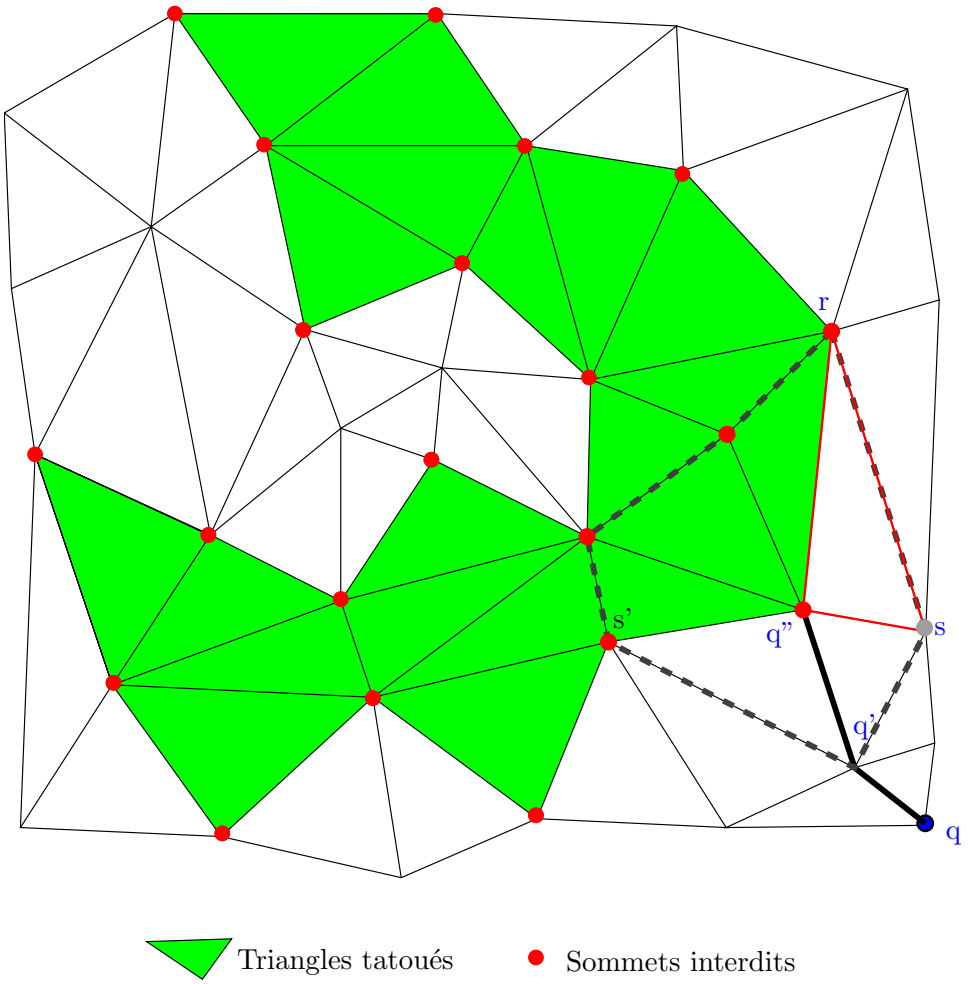

FIG. 5.16 – A chaque itération, on peut trouver un triangle valide pour l'insertion. Le chemin menant de q à q'' est représenté en gras. Le chemin (ici un cycle) autour de q'' est en pointillés. Au moins une arête (rs) sur ce chemin contient un sommet interdit (r) et l'autre libre (s). On marque alors l'information dans le site $sq''r$.

encore libre. Ce sommet est alors placé dans la liste. Nous retrouvons ici l'idée ancienne de la "liste active" que Touma et Gotsman ont utilisée [73].

Complexité

L'algorithme, du point de vue de la complexité, est en réalité constitué d'une boucle `Pour` (puisque chaque itération de la boucle `TantQue` interdit un sommet et que tous les sommets deviennent interdits), et d'une boucle imbriquée itérant sur toutes les arêtes adjacentes au sommet en cours de traitement. La complexité est donc la somme des valences de tous les points, ce qui est moins que $6N$ (par relation d'Euler). Notre algorithme permettant de maximiser le nombre de sites a donc une complexité linéaire en nombre de sommets.

5.4.3 Discussion

Regardons à présent un peu plus attentivement notre algorithme. Lorsque nous interdisons un point p, c'est relativement aux deux seuls points précédemment interdits q et r avec lesquels il forme un triangle.

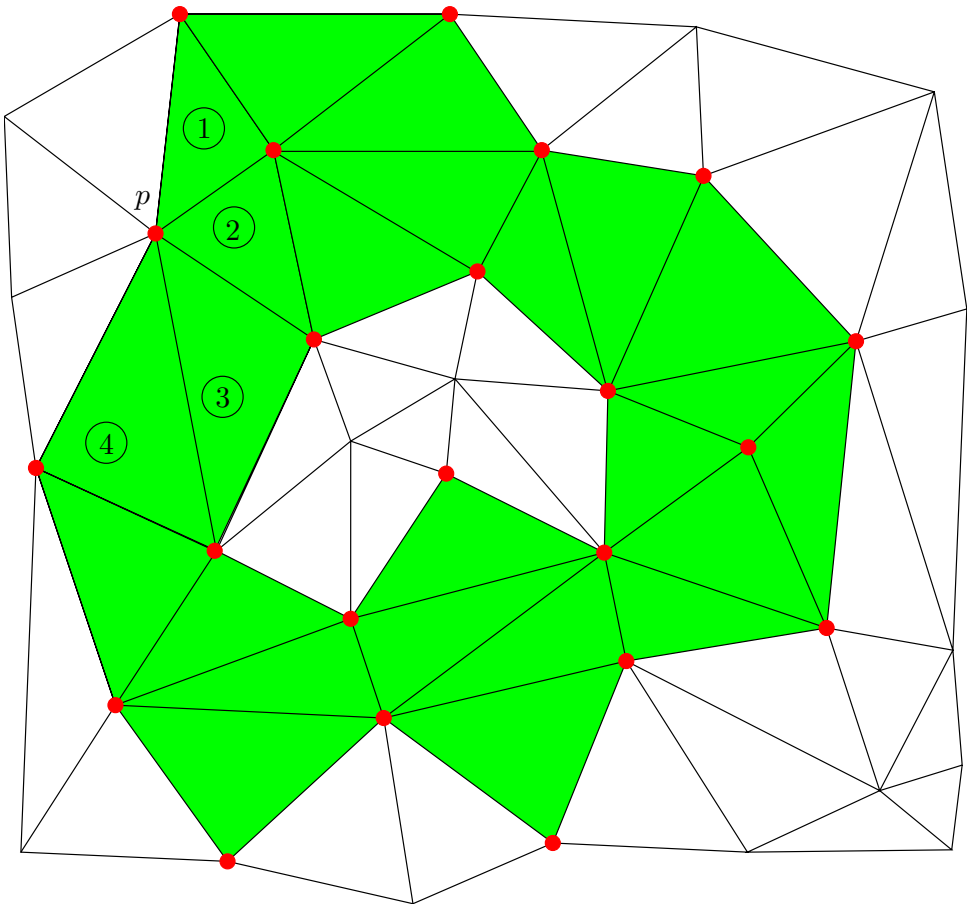

FIG. 5.17 – Nombre de sites : par rapport à l'itération de la Fig. 5.16, le sommet p ci-dessus permet de marquer en réalité plusieurs triangles (ici quatre).

Si l'on regarde la Fig. 5.16, en imaginant que l'algorithme soit au début de l'itération suivant celle ayant vu l'interdiction de s, et que le point q' soit tiré de la "liste active", il est effectivement valide au regard du triangle $q'q''s$, mais également au regard du triangle $q'q''s'$.

Ce qui revient à dire qu'à chaque fois que nous interdisons un point par rapport à *un seul* triangle, nous perdons en réalité substantiellement au regard de tous les triangles qui pourraient alors être utilisés pour coder un bit. Ainsi, le nombre de sites atteignables passerait-il tout simplement de $N-2$ à f (le nombre de triangles dans le maillage), et on sait par relation d'Euler que l'on a $f \simeq 2N$.

Mais pour cela, il nous faut sans doute une procédure d'insertion différente de celle que nous avons développée. En effet, il s'agit maintenant d'affiner la position d'un point par rapport à plusieurs triangles parmi ceux qui lui sont adjacents. La méthode d'insertion que

nous avons présentée a l'avantage certain d'être déterministe, mais si on voulait coder un bit dans chaque triangle, il faudrait procéder par perturbation des sommets (en partant par exemple d'un schéma à la Yeo et Yeung [82]). Le temps de calcul ne serait plus garanti.

Par contre, en marquant tous les triangles, on ne serait plus obligé de marquer aussi le fait qu'un site est tatoué : cette distinction n'aurait plus lieu d'être puisque tous le seraient. On se souvient que le marquage de cette information permettait également (implicitement) de retrouver la bonne orientation du triangle lors de l'insertion : quel sommet avait été potentiellement modifié. C'est pour cette raison qu'en l'état, notre procédure géométrique ne nous permet pas de marquer de l'information dans tous les triangles (seulement dans $N-2$ sommets), alors que notre parcours du graphe de connexité nous l'autorise pourtant.

Toutefois, nous risquons une piste pour tourner cette nouvelle difficulté. Puisqu'on ne pourrait plus se servir de l'information d'orientation du triangle à l'insertion, il faudrait privilégier une sorte plus spécifique d'invariants : les invariants "invariants par permutation circulaire" des sommets lors du calcul. Par exemple, le rapport de l'aire sur le périmètre élevé au carré est un tel invariant. Notre stratégie de CDMA géométrique aurait vécu, car il faudrait concaténer le numéro du bit et sa valeur associée, et cacher ce message dans un site. Notons qu'il reste en théorie possible d'organiser une imperceptibilité computationnelle (ou calculatoire) en marquant tous les triangles : il suffit que chaque sommet se trouve dans un volume, et non sur une position particulière. Notre nouvel algorithme s'écrit alors :

```
Initialisation de l'arête initiale (marquer A et B comme interdits)

TantQue il existe un point non interdit
  Si ABC est un triangle orienté de S tel que
       A et B soient interdits, mais pas C
    Trouver T l'ensemble des triangles tels que
       C est leur dernier point à interdire
    Insérer l'information dans chaque triangle de T
      en perturbant C
    Marquer C comme interdit
  Fin
FinTantQue
```

Nous n'avons pas souhaité nous consacrer à ce problème pour deux raisons :

– L'équirépartition des bits du message caché n'est plus garantie ;

– L'insertion devient un problème d'optimisation géométrique.

En effet, il n'est déjà pas acquis que l'intersection de volumes valides pour l'insertion existe, mais la trouver revient à proposer des ensembles {numéro de bit, valeur de bit} valides, et à choisir le moins mauvais au regard de l'équirépartition des bits du tatouage. Nous risquerions bien de défaire ce que nous avions voulu faire si nous persistions dans cette voie. Aussi nous faut-il nous restreindre à notre borne de $N-2$ sites atteints.

Du point de vue de la sécurité d'une application de type stéganographie, il faut souligner qu'il pourrait être utile de se servir de toute la capacité (mettre les f triangles de $\mathcal{S}$ à profit, ou un maximum d'entre eux, plutôt que les seuls $N-2$ sommets), afin que personne ne puisse développer de version "améliorée", libérant ainsi environ N bits supplémentaires (par

relation d'Euler) pour l'évasion discrète d'une clef secrète par exemple. La même remarque, en stéganographie LSB pour l'image revient à fixer une valeur sur *toute* la dynamique admissible en un pixel (sinon il resterait quelques bits non employés qu'un développeur malintentionné ou un tant soit peu malin pourrait mettre à profit).

5.5 Application au tatouage fragile

A présent, nous avons toutes les briques nécessaires au développement d'un algorithme de tatouage fragile. Nous allons utiliser une procédure géométrique de finesse 64, supportée par un arrangement indexé et un parcours optimal : nous pouvons déjà garantir que nous résisterons à la translation, la rotation et la mise à l'échelle, ainsi qu'à la coupe. Nous pouvons encore garantir le nombre de sites utilisés ($N - 2$). Jointe à notre équirépartition des bits de la marque, il devient possible de procéder à une modélisation statistique de la confiance dans la détection. Puis, sous cette contrainte, nous serons en mesure d'évaluer la taille de la plus petite surface protégée, ainsi que celle de la clef K utilisée pour brouiller les numéros des bits.

5.5.1 Contrôle de la distorsion

Au minimum, une méthode de tatouage doit permettre de régler l'ampleur de la distorsion introduite dans le média hôte. Au mieux, elle se doit de s'y adapter. Dans le domaine des surfaces, aucun modèle psycho-visuel de perception n'a encore vu le jour. Il faudra donc nous contenter de limiter la distorsion introduite : nous la contraindrons à rester dans un disque de rayon δ_{adm}, voir Fig. 5.18. On rappelle que tous les manipulations ont lieu dans le *plan* du triangle. Nous avons fixé δ_{adm} en fonction de la diagonale de la boîte englobante de l'objet d_{bbox} :

$$\delta_{adm} = 10^{-3} \times d_{bbox}. \tag{5.9}$$

On pourrait encore imaginer de contraindre la distorsion introduite à l'intérieur d'une ellipse dont les axes principaux seraient les vecteurs du champ local de courbure (rendant ainsi possible l'adaptation *de facto* aux objets de CAO, sans doute au prix d'une baisse drastique du nombre de sites car les contraintes fonctionnelles de l'objet enlèveront des degrés de liberté). Toutefois, notre stratégie simpliste permet de bien illustrer les idées que nous développons dans ce travail, et nous les appliquerons dans nos exemples à des maillages muséologiques.

5.5.2 Détection

Nous souhaitons pouvoir donner, comme seulement Praun [62] et Benedens [9] l'ont fait, une indication de la confiance dans la détection. Pour cela, nous modélisons notre méthode du point de vue statistique de la manière suivante : on va s'intéresser, pour chaque numéro de bit, à la différence signée e_{01} entre le nombre de uns relus, et celui des zéros, dans l'ordre : le signe de e_{01} est important. En effet, comme nous pouvons garantir le nombre de sites utilisés pour le tatouage, nous pouvons procéder à une modélisation statistique plus fine pour le détecteur que celle à la majorité, simpliste, que nous avons employée plus haut.

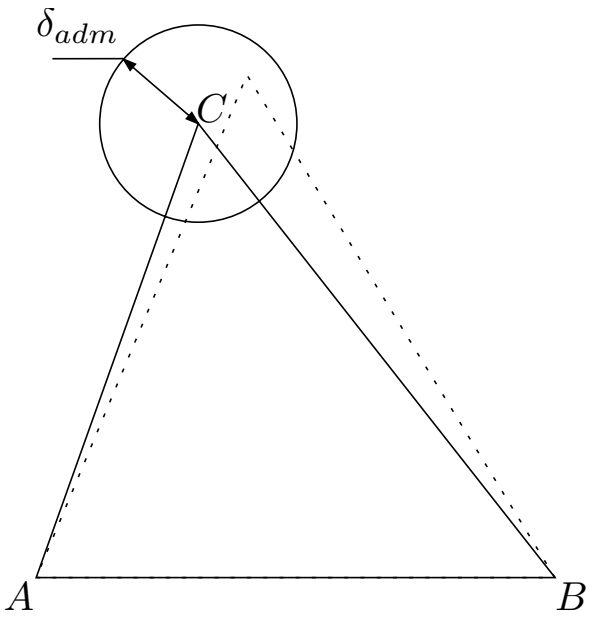

FIG. 5.18 – Contrôle a minima de la distorsion introduite : elle est contrainte de rester dans une boule de rayon δ_{adm}.

Si un maillage n'est pas tatoué avec la clef K présentée lors de la relecture, alors la distribution des e_{01} sera centrée autour de zéro. A l'inverse, si la clef K présentée est celle ayant servi à l'insertion, alors on aura, pour chaque e_{01}, soit $e_{01} << -1$ (si l'on a tatoué un 0), soit $e_{01} >> 1$ (si l'on a tatoué un 1). Nous avons donc besoin d'une distribution de référence indiquant ce qu'est statistiquement un maillage non tatoué avec K.

Distribution de référence

Nous adoptons le cadre classique en tatouage qui consiste à modéliser le canal caché par une loi gaussienne. Avant d'en estimer les paramètres, il faut disposer d'échantillons représentatifs d'une loi gaussienne non perturbée par le tatouage associé à la clef K. Deux solutions permettent de générer facilement ces échantillons :

– Soit on utilise d'autres clefs secrètes,

– Soit on va tester les valeurs des bits avec K dans un espace de numéro non tatoué

La première solution présente l'inconvénient de devoir réserver des clefs à cet usage, et de les maintenir secrètes (sinon on pourrait tenter une optimisation face à *ces* clefs-là seulement). La seconde solution, que nous avons employée, n'utilise que la clef K. On se souvient qu'un site est déclaré tatoué si :

$$|g_{ABC} - \frac{1}{2} - \lfloor g_{ABC} \rfloor| < \epsilon. \tag{5.10}$$

Ainsi, l'insertion du numéro du bit n'a fait que centrer la distribution des valeurs de tatouage autour de $\frac{1}{2}$. Si nous examinons les distributions détectées autour d'autres valeurs, on obtiendra autant d'échantillons que l'on voudra représentatifs du maillage non tatoué avec K. Nous avons pris nos échantillons autour des 8 valeurs suivantes : 0.1, 0.2, 0.3, 0.4,

0.6, 0.7, 0.8 et 0.9, avec le même ϵ. Nous reproduisons sur la Fig. 5.19 une telle distribution, pour laquelle on a tracé, après estimation des paramètres, la loi normale associée. On vient ainsi relire, à chaque valeur testée, des bits dans un espace dont on sait qu'il n'est pas tatoué.

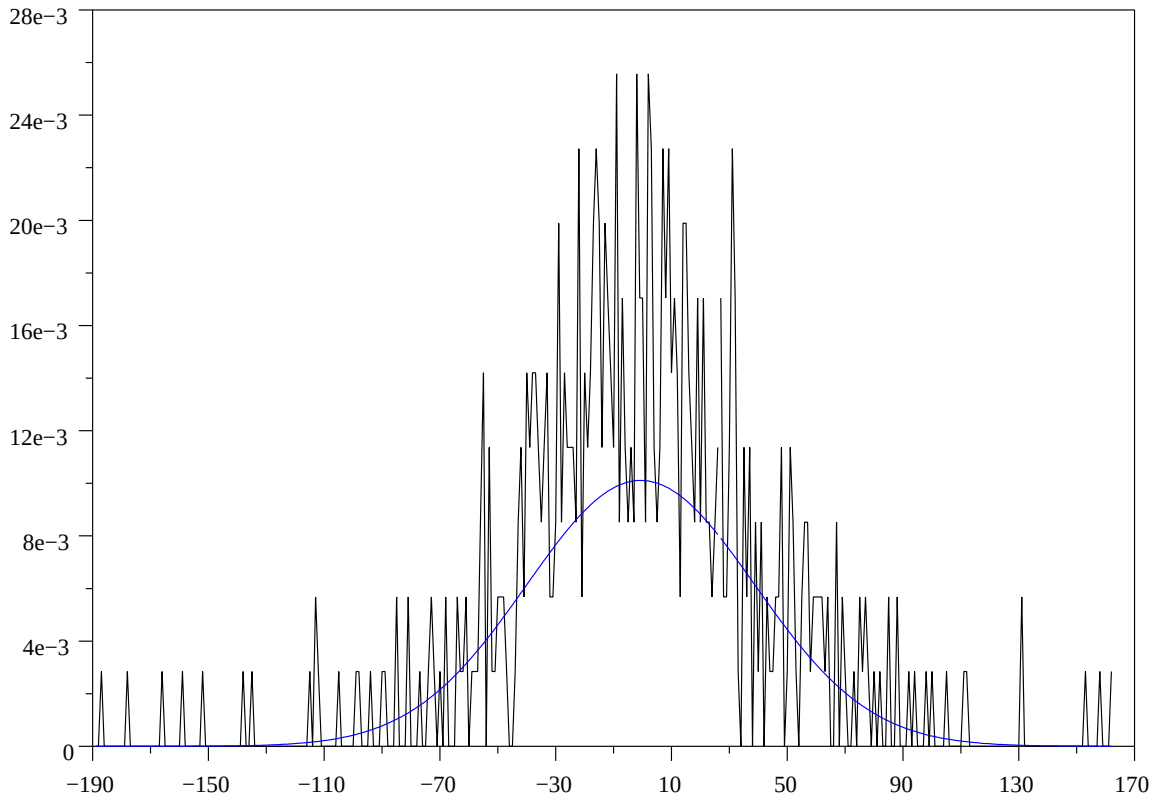

FIG. 5.19 – Distribution de référence des différences signées e_{01} des valeurs des bits. On a mesuré $8 \times 64 = 512$ différences e_{01} (8 valeurs de tests 0.1, 0.2, 0.3, etc. pour chacun des 64 numéros de bits). On modélise ainsi l'écart normal entre le nombre de 1 et de 0 relus sur un maillage non tatoué avec K. Le maillage d'exemple comporte 45906 sommets (BigHead_2). La distribution est centrée près de zéro : on trouve environ autant de un que de zéros sur un maillage non tatoué avec K.

Seuil de fausse alarme

Nous donnons sur la Fig. 5.20 l'allure de la distribution des différences des valeurs des bits pour un tatouage ne contenant que des "1" (les valeurs des différences signées sont toutes positives). Une telle distribution indique que pour un maillage tatoué avec K, la distribution des e_{01} pour chaque numéro est très fortement perturbée par le tatouage. En particulier,

on peut tester la déviation de chacun des e_{01} (pour chaque numéro de bit) par rapport à la distribution de référence, et fixer en conséquence un seuil paramétré par une probabilité de fausse alarme P_{fa}. Nous avons fixé $P_{fa} = 10^{-7}$ pour notre implantation. Nous en avons déduit un seuil de 3.7665 pour la loi $\mathcal{N}(0,1)$, en pratique le seuil est fixé à $3.7665 \times (\mu + \sigma)$ où μ et σ sont les paramètres estimés de la loi normale de référence.

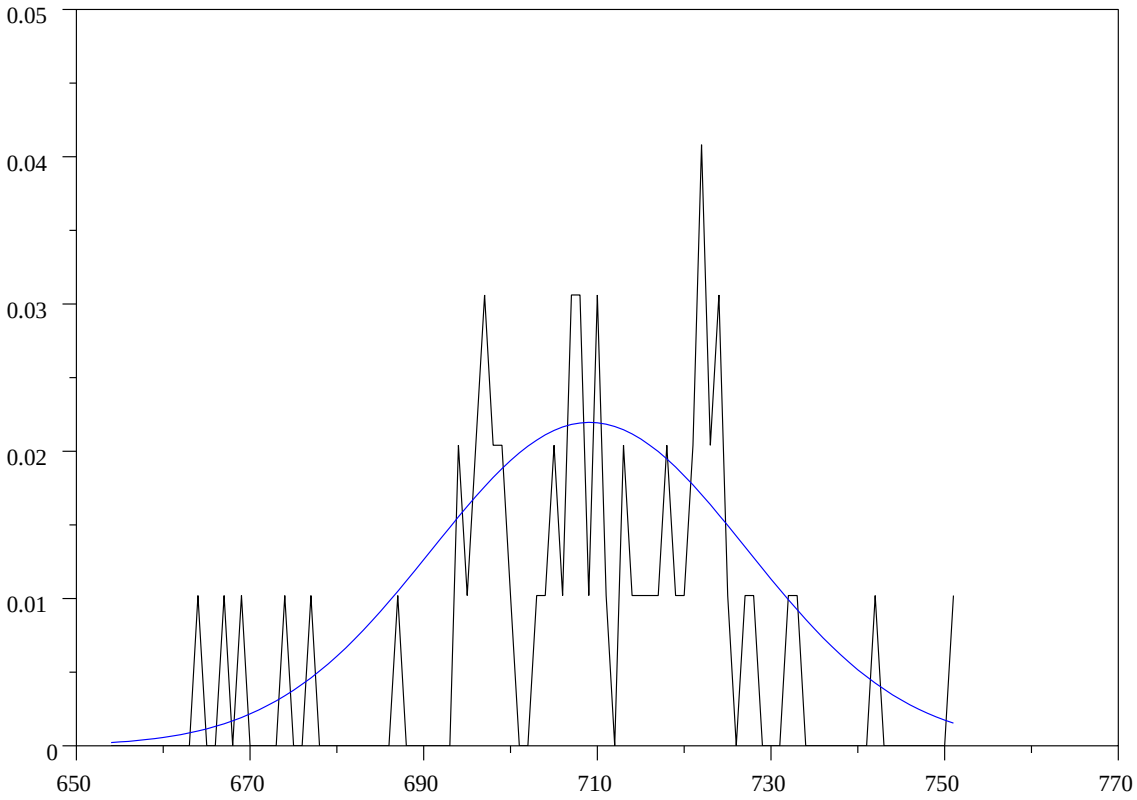

FIG. 5.20 – Distribution (pour les 64 numéros) des différences des valeurs des bits relus pour un tatouage ne contenant que des "1" (tous les e_{01} sont positifs). Par simplicité lors de la détection, nous faisons comme si nous ne cherchions à relire que des "1" : nous testons en réalité les $|e_{01}|$ (une distribution tatouée n'en reste pas moins suspecte). Le maillage d'exemple comporte 45906 sommets (BigHead_2). La distribution est centré près de $45906/64 = 717.3$: les sommets sont utilisés au maximum pour cacher de l'information.

Nous procédons en pratique à un décodage dur de l'information tatouée : le plus petit des $|e_{01}|$ valide ou non la détection entière. On pourrait encore mettre en œuvre un décodage doux en assignant à chaque symbole une probabilité propre. Nous représentons sur la Fig. 5.21 le seuil de détection, et les sorties du détecteur pour 500 clefs dont celle (numéro 250) qui a été effectivement utilisée pour cacher un message. Nous décrivons dans la section suivante

comment nous fixons la taille de l'espace des clefs, que nous évaluons à 2^{24}.

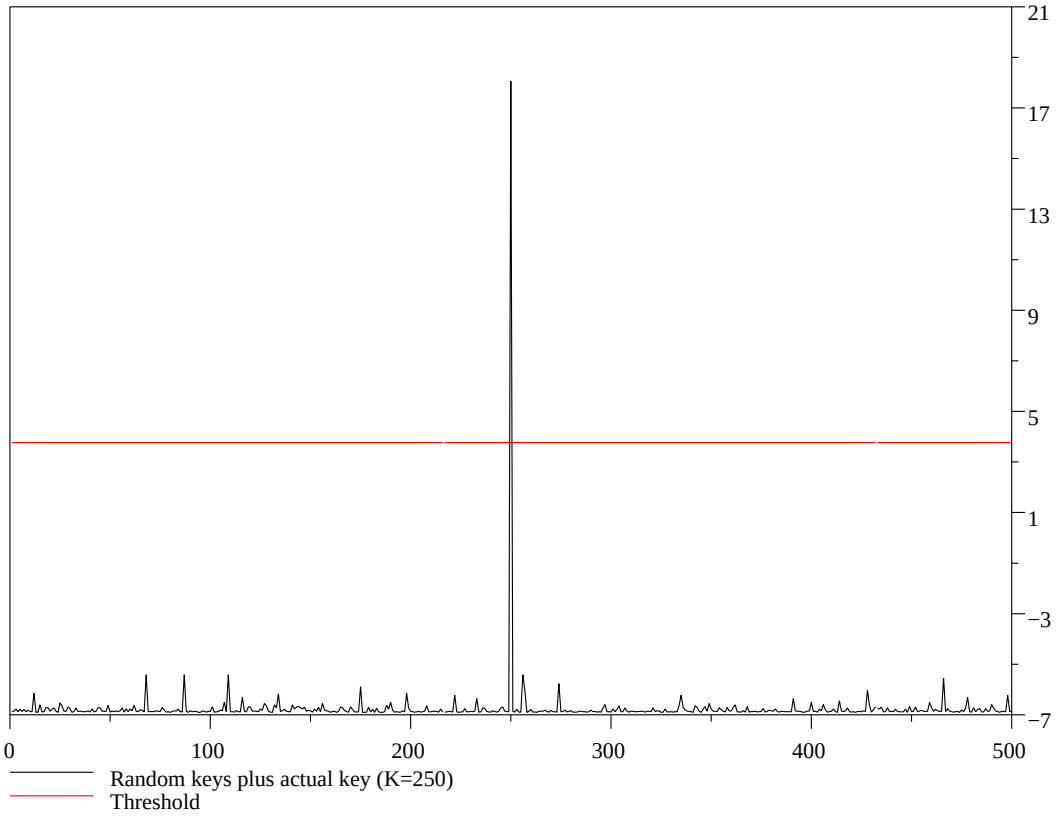

FIG. 5.21 – Sortie du détecteur pour 500 clefs. Seule la clef 250 a été effectivement utilisée pour insérer un message de 64 bits. Le seuil vaut 3.7665, et la sortie de la clef 250 vaut 18.0722.

5.5.3 Sécurité des clefs

On lit souvent dans la littérature du tatouage que la clef secrète mesure 64 bits, comme le message, sous-entendu : *tous* les 64 bits de la clef contribuent à la sécurité du tatouage. En cryptographie, les 64 bits participent effectivement à la sécurité (sous réserve d'absence de canal subliminal) car on vient tester le résultat (binaire) d'un calcul.

En tatouage, on vient tester une corrélation. On peut donc imaginer deux clefs (l'une des deux étant "la bonne") donnant un motif pseudo-aléatoire de 64 bits identique à quelques valeurs près. La corrélation sera élevée même dans le cas de la mauvaise clef, et un tatouage (faux) sera détecté, mais le pirate aura trouvé une bonne approximation. Nous cherchons à quantifier la taille de l'espace des clefs effectivement utilisables dans notre schéma.

Il faut à présent remarquer que le calcul de g_{ABC} fait intervenir deux formes linéaires en K_1 et en K_2. Ainsi, le résultat du calcul de g_{ABC} sera-t-il potentiellement proche pour deux clefs proches en valeurs : c'est précisément ce que nous voulons éviter. Nous avons choisi les valeurs T_1, T_2, et les longueurs de clefs de manière à ce que la clef la plus proche en valeur de la bonne clef donne une probabilité de fausse alarme de 10^{-1}. Autrement dit, nous organisons l'imperceptibilité statistique du tatouage et nous en déduisons expérimentalement les longueurs de clefs et les paramètres internes de l'algorithme. Nous avons fixé dans un premier temps T_1 et T_2 de manière à obtenir dès l'origine une bonne équirépartition des numéros des bits avant tatouage. Ensuite, nous avons augmenté la longueur des clefs jusqu'à ce qu'elles permettent encore de bien discerner deux clefs proches en valeur numérique.

Nous avons fixé les paramètres internes à $T_1 = 10$ et $T_2 = 4$. Les longueurs de clefs utilisables ont été déterminées expérimentalement et fixées à $N_{K_1} = 14$ et $N_{K_2} = 10$ bits. Ainsi, la sécurité de notre schéma, au sens de l'imperceptibilité statistique est-elle seulement de 24 bits.

Cela veut dire que si le message caché est effectivement crypté avec 64 bits, son imperceptibilité ne repose en réalité que sur 24 bits. D'un point de vue de pirate, il faudra une recherche brute parmi 2^{24} attaques pour tomber sur une clef donnant une réponse suffisante à la détection. Sur un PC basique, un test de clef prend environ 6 secondes (sur 100K triangles). Détecter si un tatouage est présent, sans connaître la clef secrète, prendra donc un peu plus de 3 ans, sur une seule de ces machines.

5.5.4 Résultats

Nous avons testé notre méthode sur plusieurs maillages et avons déterminé le nombre de triangles nécessaires à la détection correcte du tatouage, voir Tab. 5.22. Ce nombre est aussi appelé *Minimum Watermark Segment* dans la littérature. Il s'agit de la plus petite quantité de média protégée. Pour notre schéma, il faut 600 triangles au maximum pour retrouver la marque de 64 bits. Ce résultat plutôt satisfaisant tient au fait que peu de sites sont rejetés à cause des contraintes de distorsion ou d'exception en virgule flottante. L'apport du parcours optimal est assez saisissant lorsque l'on compare ces 600 triangles aux 40% du maillage nécessaires pour retrouver la marque avec un parcours naïf (voir le maillage Twins sur la Fig. 5.14).

Notre méthode de tatouage a été conçue pour résister à la translation, la rotation, et à la mise à l'échelle : nous l'avons vérifié en relisant la marque sur un maillage tatoué ayant subi une combinaison aléatoire des trois (ligne RST dans le Tab.5.22). La ligne MWS donne le nombre minimal de triangles nécessaires pour relire la marque (chaque triangle est permuté trois fois pour retrouver son orientation à l'insertion). Nous avons encore vérifié que d'autres manipulations effacent bien la marque : un lissage barycentrique de Taubin (seulement une itération suffit), une compression MPEG-4 3D Mesh Coding (standard : 12 bits de quantification des erreurs de prédiction géométrique), et un bruit faible sur les coordonnées (d'amplitude égale à $d_{bbox} \times 10^{-7}$). Il y a tout lieu de penser qu'à chaque fois que la marque est effacée, c'est parce que beaucoup de sites tatoués ne sont plus reconnus comme tels, et qu'ensuite éventuellement le numéro de bit qu'ils codent est faussé. En effet, les modifications géométriques codant les numéros des bits sont plus fines que celles codant les valeurs.

Objet	Horse	BigHead_1	BigHead_2	African	Twins
Sommets	48485	21777	45906	57639	83241
Triangles	96966	43550	91808	115282	166482
Genre	0	0	0	2	1
Sites tatoués	47131	21683	45726	57611	83212
Utilisation des sommets	97.21%	99.57%	99.61%	99.95%	99.97%
Répétition moyenne	736.4	338.8	714.1	900.2	1300.2
RST	Oui	Oui	Oui	Oui	Oui
MWS (triangles)	>600	>580	>590	>560	>550
Bruit faible	Non	Non	Non	Non	Non
Compression MPEG-4	Non	Non	Non	Non	Non
Lissage (2 itérations)	Non	Non	Non	Non	Non

FIG. 5.22 – Résultats obtenus en tatouage fragile.

Nous sommes maintenant en mesure de chiffrer la densité locale d'information utile que nous insérons, exprimée comme le rapport de la capacité utile sur le MWS : nous cachons 0.11 bit utile par triangle (soit 0.75 bit réel par triangle à cause de l'arrangement indexé). En effet, nous considérons qu'au pire il nous faut 600 triangles pour cacher 64 bits sous les contraintes que nous avons fixées. Par la relation d'Euler, on peut multiplier ces quantités par deux pour obtenir la densité par sommet. Notre schéma de tatouage fragile propose finalement les caractéristiques suivantes :

- Tatouage *expressément* robuste à la translation, la rotation et la mise à l'échelle uniforme,
- Parcours des $N-2$ sites utilisables en temps $O(N)$,
- Arrangement indexé permettant de pallier la coupe,
- Capacité de 64 bits utiles (448 bits réels),
- Sécurité sur la relecture du tatouage : 64 bits,
- Sécurité sur la détection du tatouage : 24 bits,
- Plus petite surface protégée : 600 triangles (déterminé expérimentalement),
- Densité minimum d'information : 0.2 bit/sommet,
- Probabilité de fausse alarme $P_{fa} = 10^{-7}$.

Nous voulons voir ici un avantage de notre méthodologie : la caractérisation des performances est plus complète que pour la plupart des méthodes à base d'invariants géométriques. A notre connaissance, nous sommes les seuls à proposer à la fois une estimation de la longueur de clef utile et de la densité d'information utile. Toutefois, tout au long de la conception de notre méthode, nous avons fait apparaître des remarques qui méritent d'être détaillées au titre des perspectives ouvertes par cette étude[78].

[78]"On commence en tout genre par le simple, ensuite vient le composé, et souvent enfin on revient au simple par des lumières supérieures. Telle est la marche de l'esprit humain.", Voltaire, *op. cit.*, p. 447. Puisse cette forte pensée nous consoler de n'avoir pas été davantage éclairé plus tôt !

5.6 Perspectives

Nous avons conçu, séparément, trois briques permettant de réaliser diverses variations autour du tatouage par invariant géométrique : une primitive géométrique d'insertion de la valeur des bits, un arrangement indexé pour leur numéro, et un parcours performant du graphe de connexité. Nous avons voulu détailler soigneusement les caractéristiques de chacune d'entre elles, car il nous sera plus facile de spéculer sur les limites de l'approche par invariants géométriques. L'application que nous visions d'emblée était le tatouage fragile, mais nous pouvons à présent discuter, par exemple, de l'application du parcours optimal à la stéganographie. Comme nous l'avons présenté dans les résultats, la marque que nous insérons est extrêmement fragile - l'étude de la robustesse face à la quantification pourrait nécessiter de trop baisser la capacité utile ou la longueur des clefs utilisées contre la détection, et c'est la raison pour laquelle nous ne voyons pas d'autre application que le tatouage fragile et la stéganographie pour l'approche par invariant. Toutefois, il reste possible d'esquisser les nouvelles caractéristiques d'un schéma optimisé pour la stéganographie, d'un autre pour l'intégrité, mais aussi d'améliorer notre méthode de tatouage fragile, et de soulever les nouvelles difficultés techniques qui y sont liées.

5.6.1 Application du parcours optimal à la stéganographie

Nous avons vu dans le cadre du tatouage par arrangement indexé que nous cachons en réalité $1 + \log_2(C_{utile}) = 7$ bits par site. Puisque dans le cadre de la stéganographie nous considérons un arrangement global (i.e : les numéros des bits cachés sont implicites), nous pouvons libérer les $\log_2(C_{utile})$ bits de codage du numéro des bits pour augmenter la capacité utile. Soit N_{bps} le nombre de bits cachés par site, nous répétons R fois chaque bit afin de pallier les inévitables erreurs de transmission dans le canal stéganographique (dues par exemple aux exceptions en virgule flottante). La capacité C_{stego} en bits de notre schéma de stéganographie s'écrit donc :

$$C_{stego} = \lfloor (N-2) \times N_{bps} \times \frac{1}{R} \rfloor. \tag{5.11}$$

Une autre modification à apporter concerne le parcours, et nécessiterait de choisir le sommet suivant de manière pseudo-aléatoire dans la liste active : cela rendrait la seule reconstitution de la *séquence* des bits cachés improbable. Enfin, puisque l'on se servirait du parcours pour orienter implicitement les triangles à la relecture, on pourrait se dispenser de marquer un triangle comme tatoué, et laisser flotter indifféremment g_{ABC} dans $[\lfloor g_{ABC} \rfloor ; \lfloor g_{ABC} \rfloor + 1[$, cela afin de reconquérir l'imperceptibilité computationnelle perdue. Avec cette technique, nous ne voyons pas même comment on pourrait suspecter seulement l'existence d'un message caché.

5.6.2 Application de l'arrangement indexé à l'intégrité

Si un pirate modifie une partie seulement du maillage, on aimerait pouvoir en être informé. Pour cela, sans disposer encore d'outil fiable pour quantifier une telle modification, on pourrait toutefois dans un premier temps mettre en œuvre une stratégie d'inspection visuelle. En effet, les $N-2$ sites étant atteints (et très peu rejetés), on peut considérer que

la dispersion de l'information couvre tout le maillage. L'arrangement indexé autorise la mise en évidence des zones ayant éventuellement été malicieusement modifiées. En effet, dans la zone incriminée, plus aucun sommet ne sera reconnu comme tatoué. Notons cependant qu'il conviendrait sans doute de développer ici des techniques de morphologie mathématiques pour mieux agréger entre elles les parties dégradées.

Il faut reconnaître que cette application est cruciale : si une partie du maillage seulement avait été modifiée, supposée petite, la détection du tatouage eût tout de même été probablement suffisante pour relire la marque. La modification malicieuse serait passée inaperçue. On pourrait imaginer de donner l'alarme lorsque la plus petite surface connexe non tatouée excède un seuil significatif, ou de noter l'ampleur des modifications par rapport à $N - 2$, pour indiquer à l'utilisateur s'il doit procéder à une inspection visuelle. Pour cela, on peut commencer par mettre en valeur les triangles porteurs d'information cachée détectés comme tels, et les autres, comme illustré sur la Fig. 5.23 (où le maillage tatoué n'a subi aucune manipulation).

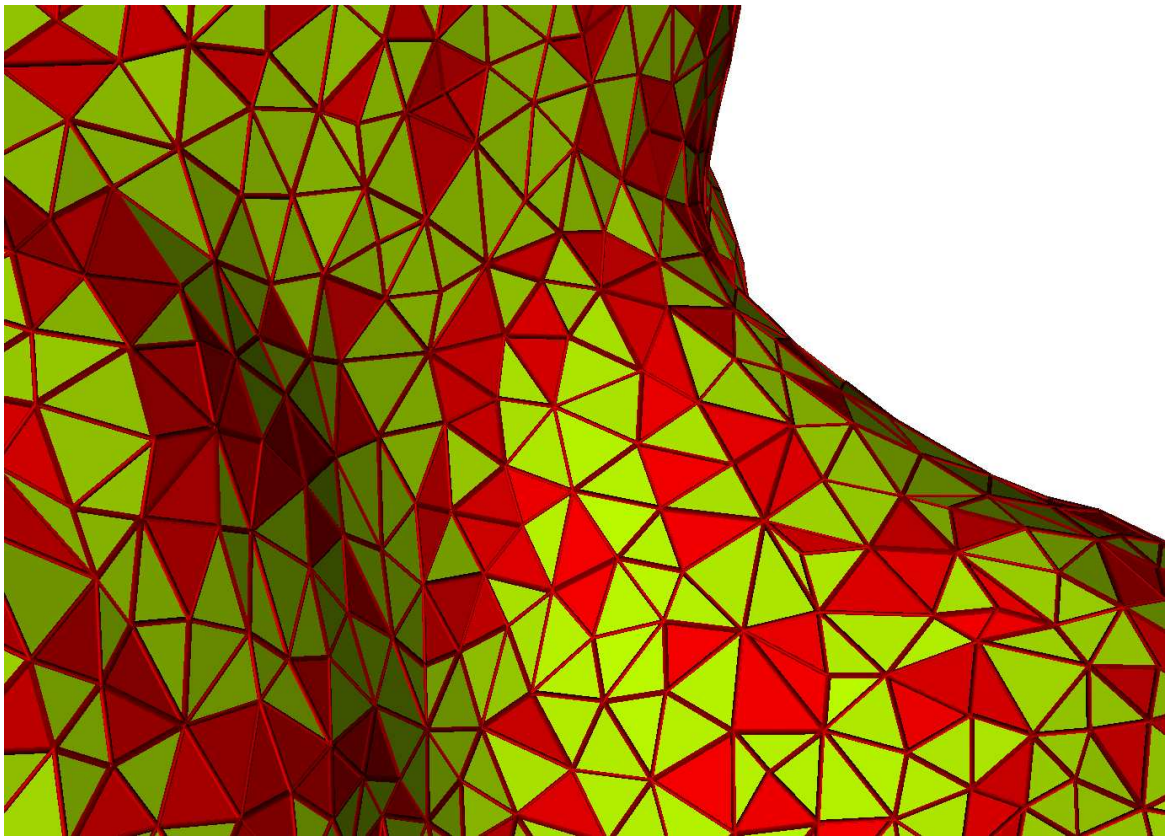

FIG. 5.23 – Localisation de l'information tatouée : les triangles porteurs d'information sont en vert, les autres en rouges. Comme attendu, environ un triangle sur deux code de l'information (soit le nombre de sommets à peu près). Des techniques de morphologie mathématique permettraient d'agréger ensemble les zones dont la densité de triangle porteurs d'information serait inférieure à un certain seuil, afin de visualiser les zones modifiées.

5.6.3 Tatouage de tous les triangles

Nous l'avons détaillé plus haut, notre parcours permettrait de tatouer tous les triangles (et non plus seulement $N-2$ sommets), au prix d'une modification substantielle de la procédure entière d'inscription géométrique de l'information. Nous avons suggéré l'utilisation d'invariants dont le calcul est invariant par permutation circulaire des sommets, comme le rapport de l'aire sur le périmère au carré.

Toutefois, il n'est pas évident de concevoir une procédure d'inscription qui autorise ce mode de relecture. En effet, on devra fixer la valeur de cet invariant dans un nombre variable (parfois élevé) de triangles connexes. Nous laissons ce problème à la sagacité du lecteur intéressé. La résolution de ce problème serait bienvenue car n'ayant plus besoin d'orienter les triangles, le tatouage fragile réellement imperceptible deviendrait possible car la présence du tatouage pour un pirate ne serait plus trahie par l'Eq. 5.8. Enfin, il serait encore envisageable de bénéficier d'un module d'inspection visuelle de l'intégrité (même si elle serait sans doute de moins bonne qualité qu'avec l'information d'orientation des triangles disponible). Comme un maillage contient environ deux fois plus de triangles que de sommets, la taille du MWS diminuerait en rapport (la densité des sites serait multipliée par deux pour un même nombre de bits).

5.6.4 Allocation adaptative des bits

Dans la présente implantation, nous forçons les bits à être inscrits les uns après les autres, dans l'ordre des numéros de la marque. Une telle stratégie peut sans doute être améliorée en plaçant les numéros des bits dans une file de priorité et en déterminant dynamiquement le bit à inscrire (et donc son numéro) en fonction de la géométrie, sous contrainte d'équirépartition des numéros.

Les avantages attendus sont de deux ordres :

- *Minimisation de la distorsion* On favoriserait les numéros entraînant la plus petite distorsion,
- *Optimisation du nombre de sites tatoués* Il devient possible de se servir de quasiment tous les $N-2$ sites atteints, car on pourra sans doute trouver à chaque itération un bit à inscrire qui ne génère pas d'erreur de traitement (distorsion géométrique ou exception en virgule flottante).

5.7 Conclusion

Nous avons proposé, autour du tatouage fragile, un tour d'horizon des possibles, tant en stéganographie qu'en protection de l'intégrité, de notre approche par invariants géométriques. Nous avons souhaité insister sur l'aspect modulaire de notre raisonnement, afin de bien séparer ce qui peut encore porter à amélioration. Au final, nous avons développé une méthode de tatouage fragile permettant de déterminer nombre de caractéristiques utiles au tatoueur, comme la probabilité de fausse alarme, la taille de la plus petite surface protégée (MWS), la classe de robustesse admissible, la sécurité face à la détection pirate, le tout avec un parcours optimal du graphe en $O(N)$.