UNIVERSITE DE NICE-SOPHIA ANTIPOLIS - UFR Sciences
Ecole Doctorale de Sciences et Technologies de l'Information et de la Communication

T H E S E

pour obtenir le titre de
Docteur en Sciences
de l'UNIVERSITE de Nice-Sophia Antipolis

Discipline : Informatique

présentée et soutenue par
Emmanuel GARCIA

T A T O U A G E D ' O B J E T S 3 D B A S E S U R L A T E X T U R E

Thèse dirigée par ***Jean-Luc DUGELAY***
soutenue le *5 juillet 2004*

Jury :

Professeur	Attila BASKURT	président
Professeur	Gérard MEDIONI	rapporteur
Professeur	Thierry PUN	rapporteur
Docteur	Sylvain BOUGNOUX	examinateur
Professeur	Benoît MACQ	examinateur

Remerciements

Je tiens à exprimer toute ma gratitude à mon directeur de thèse Jean-Luc Dugelay pour son enseignement, pour ses conseils avisés et pour son indéfectible soutien tout au long de cette thèse. Je souhaite remercier vivement les membres de mon jury pour m'avoir fait l'honneur d'y participer et pour avoir accepté la tâche d'évaluer mon travail. J'ai une pensée particulière pour Caroline Mallauran dont le travail de stage remarquable en tatouage 3D basé sur la texture a largement contribué à initier notre travail de recherche en tatouage. Je suis également redevable à Vanessa Lopez Eslava dont le stage a permis de faire avancer la réflexion sur la stéganographie géométrie-dans-texture. Je suis reconnaissant au projet RNRT SEMANTIC-3D d'avoir soutenu une partie de mon travail, mais je voudrais surtout remercier les partenaires et membres de ce projet pour l'ambiance de travail constructive et chaleureuse. Je remercie Gwenaël Doërr pour sa relecture attentive du manuscript. Enfin je tiens à saluer mes camarades : collègues de bureau, collègues du groupe image, collègues doctorants. Je salue également Frédéric Bloch, doctorant en physique nucléaire et compagnon d'armes.

Sommaire

Avant-propos **1**

Introduction **3**

I Tatouage d'objets 3D basé sur la géométrie **7**

1 Rappels sur le tatouage de documents numériques **9**

1.1 Capacité, visibilité, robustesse 9

1.1.1 Tatouage avec ou sans capacité 9

1.1.2 Taux de fiabilité 10

1.1.3 Attaque destructive, non-destructive 10

1.1.4 Tatouage fragile, robuste 10

1.2 Protocoles de tatouage 11

1.2.1 Principe de base 11

1.2.2 Extraction aveugle et non-aveugle 11

1.2.3 Applications 11

1.2.4 Faiblesses et sécurisation des protocoles 12

2 Etat de l'art du tatouage d'objets 3D basé sur la géométrie **15**

2.1 Classification et description des objets 3D 15

2.1.1 Notion d'objet 3D 15

2.1.2 Représentation informatique des objets 3D 17

2.2 Application des principes du tatouage aux objets 3D 24

2.2.1 Tatouage de forme, de données, de texture 24

2.2.2 Attaques destructives, non-destructives 25

2.2.3 Tatouage fragile, robuste 25
2.2.4 Compromis entre capacité, visibilité, robustesse . . . 26
2.2.5 Mode d'extraction aveugle, non-aveugle 26
2.2.6 Applications possibles 27
2.3 Manipulations et attaques d'objets 3D 28
2.3.1 Renumérotation des sommets ou des triangles . . . 28
2.3.2 Similitudes, transformations affines et projectives . . 28
2.3.3 Ajout de bruit 29
2.3.4 Remaillage, compression 29
2.3.5 Découpe 29
2.3.6 Attaque par collusion, effacement, ajout de marque 30
2.3.7 Attaques d'objets texturés 30
2.4 Liste des algorithmes de tatouage d'objets 3D 30

II Tatouage d'objets 3D basé sur la texture 41

3 Principe du tatouage d'objets 3D basé sur la texture 43
3.1 Vue d'ensemble de l'algorithme 43
3.2 Parallèle avec le tatouage d'images fixes 44
3.2.1 Mode d'extraction 44
3.2.2 Visibilité du tatouage 45
3.2.3 Types d'attaques 45

4 Evaluation en environnement contrôlé 49
4.1 Modalités générales des expériences 49
4.1.1 Modèle de rendu 2D 49
4.1.2 Modèle de texturage 50
4.1.3 Modèle de projection 51
4.1.4 Modèle d'éclairage 52
4.1.5 Protocole et algorithme de tatouage d'images fixes . 52
4.1.6 Mesure des performances 53
4.2 Mise en oeuvre de la reconstruction de la texture 56
4.2.1 Principe 56
4.2.2 Utilisation d'un Z-buffer 57

4.2.3 Rééchantillonnage de la texture 58
4.3 Expérience de référence 58
4.3.1 Premier objet 59
4.3.2 Deuxième objet 62
4.4 Impact d'une compression JPEG 65
4.4.1 Compression de la texture 65
4.4.2 Compression de la vue 2D 65
4.5 Impact d'une mauvaise connaissance de la projection 66
4.6 Impact d'une simplification de maillage 68
4.7 Impact du référentiel de texture 70
4.7.1 Complétion de la texture reconstruite 70
4.7.2 Changement du mode de texturage 71
4.7.3 Utilisation d'un mode de texturage intermédiaire . . 73
4.8 Commentaires 74

5 Estimation des paramètres de rendu 79
5.1 Présentation du problème 79
5.2 Recalage projectif et vision par ordinateur 80
5.3 Estimation de la projection 81
5.3.1 Approches basées sur la texture 81
5.3.2 Notre algorithme 82
5.3.3 Appariement de blocs et flot optique 83
5.3.4 Technique d'appariement utilisée 85
5.3.5 Calcul de la projection 87
5.4 Estimation de l'illumination 89
5.4.1 Aperçu du problème 89
5.4.2 Méthodes existantes 90
5.4.3 Choix de l'algorithme 91
5.5 Résultats 92
5.5.1 Premier objet 92
5.5.2 Deuxième objet 96
5.5.3 Limitations 98

Annexe 101

A L'algorithme de tatouage Eurémark **101**
A.1 Insertion du tatouage 101
A.1.1 Formatage et cryptage de la marque 101
A.1.2 Génération du support 102
A.1.3 Combinaison de la marque au support 102
A.2 Extraction du tatouage 102

III Stéganographie **105**

6 Dissimulation d'un modèle géométrique dans une image de texture **107**
6.1 Motivation 107
6.2 Représentation des données 109
6.3 Synchronisation spatiale 110
6.4 Synchronisation fréquentielle 111
6.5 Technique de codage 114
6.5.1 Décompositions parallèles en ondelettes 114
6.5.2 Codage d'une valeur dans un bloc 114
6.6 Gestion des erreurs 116
6.7 Résultats 118
6.7.1 Protocole expérimental 118
6.7.2 Premier objet 119
6.7.3 Deuxième objet 122
6.7.4 Localisation des erreurs 125
6.8 Commentaires 127

Conclusions et perspectives **129**
8.1 Tatouage d'objets 3D basé sur la géométrie 129
8.2 Tatouage d'objets 3D basé sur la texture 130
8.3 Stéganographie 131

Bibliographie **133**

Avant-propos

Cette thèse visait initialement à poursuivre le projet de "télé-conférence virtuelle" TRAIVI, à la suite des travaux de Stéphane Valente, Katia Fintzel et Ana Cristina Andrés del Valle.

La télé-conférence virtuelle est une application ayant pour but de remplacer les séquences vidéos des intervenants par des animations de synthèse de leurs *clones* informatiques. L'intérêt est d'une part de réduire la bande passante utilisée, en ne transmettant que des paramètres d'animation au lieu de flux vidéos, et d'autre part de pouvoir créer un espace de réunion virtuel commun en jouant sur le décor et sur la position des intervenants dans les scènes de synthèse créées.

Cette application nécessite de réaliser l'analyse de la pose et des expressions faciales sur le site émetteur, et la synthèse de la scène virtuelle sur le site récepteur. Pour effectuer cette synthèse il faut disposer d'un modèle 3D réaliste et *animable* de la personne à synthétiser. En outre, un tel modèle, qu'on appelle alors clone, peut être utilisé pour l'analyse des expressions faciales au moyen d'une boucle d'analyse/synthèse.

Dans tous les cas, la possibilité de générer un clone *animable* d'un nouvel intervenant, et si possible facilement, est un prérequis à l'application de télé-conférence virtuelle.

Nous nous sommes investi sur ce problème lors du stage de DEA et au tout début de la thèse, mais il nous était difficile de rivaliser dans ce domaine avec d'autres laboratoires dont c'était l'activité principale depuis longtemps. Or à ce moment-là nous avons identifié un autre problème lié à l'emploi d'images de synthèse, que ce soit pour la télé-conférence virtuelle ou non : la possibilité que quelqu'un fasse passer une image de synthèse pour une image réelle, ou plus généralement qu'il utilise un modèle 3D sans autorisation.

L'exposé qui suit est donc exclusivement consacré au tatouage d'objets 3D ou aux méthodes de stéganographie liées aux objets 3D.

Introduction

De nos jours il est relativement aisé de produire des images de synthèse de mondes virtuels ou des images de synthèse imitant le monde physique. De nombreux outils sont disponibles pour permettre à qui veut de créer de telles images, voire même des séquences vidéos, avec une qualité pouvant parfois approcher le photo-réalisme.

Parallèlement à cette amélioration des techniques d'imagerie virtuelle, rendue possible par les progrès des performances des matériels informatiques, les objets 3D ont commencé à envahir plusieurs domaines d'activité. Dans l'industrie des jeux vidéos la modélisation 3D est devenue presque systématique. Dans le demaine du cinéma la 3D permet de créer des dessins animés entiers. Dans le domaine purement informatique, des mondes virtuels 3D sont développés avec une sémantique particulière permettant de visualiser et de manipuler de grandes quantités de données de manière conviviale, intuitive et efficace [2]. D'autres applications mêlent les images de synthèses aux images réelles pour produire ce qu'on appelle réalité augmentée [55]. Cette technique est utilisée pour la réalisation de films où des personnages de synthèse sont insérés dans des décors réels, ou pour la création d'effets spéciaux. Elle fait également l'objet d'importantes recherches dans le domaine médical où par exemple un chirurgien pourrait être assisté d'un système de vision qui mettrait artificiellement en valeur des zones d'intérêt dans une opération en cours.

Une autre application, encore au stade la recherche, fait un usage intensif de toutes les techniques 3D, que ce soit le rendu et l'animation d'objets 3D, l'analyse d'images ou de vidéos réelles, ou la mise en situation de modèles 3D dans des espaces virtuels : il s'agit de la télé-conférence virtuelle [28]. En théorie la principale caractéristique de cette application est d'utiliser des modèles 3D photo-réalistes des interlocuteurs de la télé-conférence pour recréer de manière synthétique l'animation du visage de chacun d'eux, et ce, de la manière la plus réaliste possible. Bien que cet objectif soit encore loin d'être atteint, il met en lumière un problème potentiel qui pourrait se poser, et pas uniquement dans le cadre d'une télé-conférence : l'utilisation frauduleuse du clone synthétique d'une personne pour générer une séquence vidéo photo-réaliste la représentant en train de donner un discours ou en train d'effectuer une action complètement fictifs. Dans ce cas précis on voit l'intérêt qu'il peut y avoir à sécuriser l'emploi des objets 3D, c'est-à-dire en l'occurrence à pouvoir identifier une utilisation

frauduleuse comme telle.

Malgré la généralisation du tatouage numérique, technique consistant à cacher des informations, notamment de copyright, dans un document informatique, aucune étude n'avait été consacrée à la sécurisation des représentations visuelles d'objets 3D au moyen de techniques de tatouage. Il est certes vrai que de nombreux algorithmes ont été développés pour protéger l'information géométrique des modèles 3D en elle-même — et ceci est évidemment très intéressant pour faire respecter les droits d'auteur sur une forme géométrique 3D — mais malheureusement aucun de ces algorithmes ne permet de savoir si un objet 3D apparaissant dans une image de synthèse donnée est tatoué ou non, or, en vue d'une vérification des droits d'auteur, il peut être bien plus facile de mettre la main sur une image frauduleuse produite à partir d'un objet 3D que sur les données 3D elles-mêmes. Il s'agirait donc ici de pouvoir détecter le tatouage à partir d'un rendu 2D d'un objet 3D tatoué, et non pas à partir des données 3D elles-mêmes.

Le problème de la protection de l'utilisation d'un objet 3D versus la protection de l'objet 3D lui-même est précisément le problème original que nous nous sommes posé dans cette thèse. L'idée proposée, et soumise à l'expérimentation, consiste à tatouer un objet 3D dans sa texture plutôt que dans sa géométrie, avec l'espoir que le tatouage soit toujours présent, sous une forme ou sous une autre, dans les images 2D résultantes.

Notre travail est divisé en trois parties. Dans la première partie nous rappelons les notions de base du tatouage numérique, nous détaillons les multiples aspects de la notion d'objet 3D en informatique, et nous dressons l'état de l'art du tatouage d'objets 3D (basé uniquement sur la géométrie).

Dans la deuxième partie nous présentons un algorithme de tatouage d'objets 3D basé sur la texture et censé protéger les représentations 2D d'objets 3D. Les propriétés de cet algorithme sont comparées d'une part aux propriétés des algorithmes de tatouage d'images fixes, et d'autre part, lorsque cela est approprié, aux propriétés des algorithmes de tatouage d'objets 3D basés sur la géométrie. La pierre angulaire de notre algorithme est la capacité à estimer les paramètres de rendu ayant conduit à une image 2D donnée d'un objet 3D, et ce en vue de l'inversion du rendu. Dans un premier temps nous donnons une indication des performances limites atteignables par notre algorithme au moyen d'expériences en environnement contrôlé où les conditions de rendu 2D sont connues avec exactitude. Dans un deuxième temps nous nous plaçons dans un cadre plus réaliste où ces conditions de rendu doivent être estimées. A cet effet nous proposons un algorithme de recalage projectif 2D/3D entre un objet 3D texturé et une image 2D de celui-ci.

La dernière partie de la thèse nous avons voulu la consacrer à une application de stéganographie d'objets 3D. En effet, devant les difficultés rencontrées dans l'élaboration de protocoles de tatouage réellement sécurisés [23, 5], les recherches en tatouage ont tendance à s'orienter dans deux directions : d'une part l'étude théorique des techniques de tatouage au moyen de modélisation de

type théorie de l'information [66, 73], et d'autre part le développement concret d'algorithmes de stéganographie sans impératifs de sécurité stricts. L'application que nous proposons consiste à dissimuler la description géométrique d'un objet 3D texturé dans l'image de texture qui lui est associée, de sorte qu'il suffit de transmettre l'image de texture pour pouvoir reconstruire l'objet 3D. L'originalité de cette application n'est pas tant le fait qu'elle concerne des objets 3D que le fait qu'elle offre un cas de figure où l'information dissimulée est intimement liée à l'information support. L'algorithme de stéganographie développé s'attache à conserver ce lien et pourrait être généralisé à d'autres applications.

Première partie

Tatouage d'objets 3D basé sur la géométrie

Chapitre 1

Rappels sur le tatouage de documents numériques

Le tatouage d'images est une technique relativement récente permettant de cacher de manière invisible et robuste un message dans une image ou plus généralement un document numérique multimédia [22, 54].

Selon le service visé, le message peut contenir des informations sur le propriétaire (droits d'auteur), l'image elle-même (intégrité ou indexation) ou encore l'acheteur (non-répudiation). Il est ensuite possible de récupérer le message à tout moment et ce même si l'image protégée a été volontairement ou non modifiée par une ou plusieurs attaques non-destructives.

Trois critères, a priori contradictoires, sont à considérer simultanément : la capacité (i.e. quantité d'information cachée), la visibilité (i.e. distorsion introduite par le processus de tatouage) et la robustesse.

En plus de ces trois critères, il est également important de distinguer si l'extraction est réalisée de manière aveugle ou non (i.e. besoin du document original pour récupérer la marque).

Le tatouage a été largement étudié depuis une dizaine d'années pour les images fixes principalement, mais également pour l'audio et la vidéo. Encore peu de travaux existent à ce jour pour les objets 3D qui sont classés parmi les "nouveaux objets" au même titre que les partitions musicales ou le code java par exemple.

Les principales notions du tatouage sont reprises dans ce document dans leur application aux objets 3D. Néanmoins pour plus de détails sur le tatouage en général, le lecteur peut se référer à [22, 54, 31].

1.1 Capacité, visibilité, robustesse

1.1.1 Tatouage avec ou sans capacité

La notion de tatouage "sans capacité" fait référence à un protocole de tatouage (cf. section 1.2.1) où l'extraction du tatouage consiste à vérifier qu'une

marque donnée est présente dans un document. Le résultat de cette vérification est une réponse binaire oui/non quant à la présence de la marque. Le terme "sans-capacité" caractérise cette réponse, qui représente toujours un seul bit d'information, et ce quelles que soient la taille de la marque utilisée et la manière dont elle est enfouie.

Dans d'autres cas, on ne cherche pas à vérifier qu'une marque donnée est présente ou non, mais on cherche à savoir *si* une marque est présente dans le document et si oui, *laquelle*. Cette fois le résultat de l'extraction n'est pas une réponse binaire oui/non concernant la présence ou l'absence d'un message donné, mais c'est le message lui-même, inconnu a priori. On appelle capacité du tatouage la taille du message en question.

1.1.2 Taux de fiabilité

Dans le cas "sans capacité" la réponse binaire sur la présence du tatouage est déterminée de manière probabiliste. Il y a deux taux d'erreur à considérer : la probabilité de détecter que le document est tatoué alors qu'il ne l'est pas (fausse acceptation), et la probabilité de ne pas détecter que le document est tatoué alors qu'il l'est (faux rejet). Ces deux probabilités sont liées par une courbe *ROC* [22]. L'extraction du tatouage (généralement un calcul de corrélation entre les modifications apportées à l'original par le tatouage et les modifications effectivement lues dans le document suspect) permet de calculer une estimation de la probabilité de fausse acceptation. Si celle-ci est inférieure à un seuil fixé on déclare que le document est tatoué.

Dans le cas "avec capacité" on peut éventuellement avoir aussi une indication de la fiabilité du message extrait. On peut aussi améliorer la fiabilité en diminuant la capacité, par exemple par l'emploi de codes correcteurs. Pour l'évaluation des algorithmes on mesure surtout le taux d'erreur binaire du message extrait.

1.1.3 Attaque destructive, non-destructive

Certains documents informatiques peuvent subir des modifications, malveillantes ou non, qui peuvent aussi affecter un éventuel tatouage. Lorsque ces manipulations ne changent pas de manière significative le contenu sémantique du document on parle d'attaque non-destructive, sinon on parle d'attaque destructive.

Le seuil à partir duquel une attaque est considérée comme destructive est arbitraire et dépend du type de document considéré et du contexte applicatif.

1.1.4 Tatouage fragile, robuste

On parle de tatouage fragile lorsque le tatouage est altéré, voire détruit, par une attaque non-destructive du document tatoué. Ceci permet de vérifier l'intégrité de certains documents.

Un tatouage robuste, au contraire, vise à être robuste à un maximum d'attaques, y compris certaines attaques destructives. Dans ce cas l'application privilégiée du tatouage est de pouvoir tracer la diffusion d'un document même s'il a subi de fortes modifications, éventuellement dans le but malveillant d'en effacer la marque.

Cette différenciation entre tatouage fragile et robuste repose entièrement sur la différenciation entre attaque destructive et attaque non-destructive, laquelle a un caractère arbitraire, mais le terme de tatouage robuste est plus souvent employé dans un autre sens, par rapport à une attaque précise.

On dira par exemple qu'un algorithme de tatouage est robuste à telle ou telle attaque particulière, ce qui signifie que le tatouage peut être lu à partir du document tatoué même après qu'il ait subi l'attaque en question.

1.2 Protocoles de tatouage

1.2.1 Principe de base

Il existe deux protocoles de base du tatouage. Le premier consiste à cacher une certaine quantité d'informations dans un document informatique et à pouvoir extraire ces informations du document même après qu'il ait subi certaines opérations non-destructives, et en tolérant éventuellement des pertes dans les informations récupérées.

Le second consiste à modifier de manière imperceptible un document informatique pour y laisser une empreinte. Dans ce cas l'extraction ne consiste pas à récupérer une certaine quantité d'informations mais à dire si oui ou non une empreinte donnée est présente dans le document, et de quantifier la probabilité de donner une réponse erronée (voir aussi 1.1.1).

1.2.2 Extraction aveugle et non-aveugle

Le mode d'extraction de la marque peut être aveugle ou non-aveugle suivant qu'on a besoin du document original (tatoué ou non) pour l'extraction.

Dans le cas où on cache une marque dont le procédé d'extraction et/ou la clé nécessaire à l'extraction sont connus du seul propriétaire ou auteur du document, le mode d'extraction peut être non-aveugle car celui qui va tenter d'extraire la marque possède l'original. Par contre le fait de devoir fournir des données supplémentaires à l'extraction peut rendre l'opération plus fastidieuse.

1.2.3 Applications

Parmis les applications possibles du tatouage on peut citer : la vérification des droits d'auteur, le contrôle d'intégrité, le contrôle d'accès, le suivi de copies, l'audimétrie, ou encore la stéganographie.

La vérification de l'intégrité d'un document peut être effectuée par l'emploi d'un tatouage fragile. En outre il est possible, lorsque le document tatoué a été

modifié, et en utilisant un algorithme de tatouage particulier, de savoir quelles parties ont été modifiées, voire d'obtenir une indication sur le contenu original. Cela dépend directement de la capacité du tatouage.

La vérification des droits d'auteurs peut se faire en utilisant un tatouage "sans capacité".

On peut tracer les copies d'un document en utilisant un tatouage robuste contenant des informations personnalisées sur la voie de distribution initiale (i.e. fingerprinting).

La stéganographie est quant à elle un terme générique désignant le fait de cacher un message dans un autre sans impératif de sécurité, et peut donner lieu à de multiples applications.

1.2.4 Faiblesses et sécurisation des protocoles

Indépendemment des attaques sur les données tatouées, qui peuvent mettre en défaut les techniques de tatouage, il y a des failles, ou des limitations, dans les protocoles de tatouage eux-mêmes. Nous ne présentons ici que deux exemples, mais pour plus de détails on peut se référer à [23].

Utilisation d'un détecteur de tatouage pour effacer le tatouage

Dans le cas d'un mode d'extraction aveugle, on peut imaginer qu'une personne puisse se procurer un détecteur de tatouage qui indique si un document contient ou non un tatouage et éventuellement une indication sur la fiabilité de cette réponse. Il peut alors modifier le document bit après bit, pixel après pixel, mot après mot, triangle après triangle, suivant le type de document considéré, pour voir si cela diminue la détection du tatouage. Il suffit alors d'effectuer les modifications qui diminuent le plus la détection du tatouage tout en conservant au mieux l'utilité du document. Idéalement cela peut conduire à retrouver le document original non-tatoué. Cette procédure peut mettre en danger potentiellement tous les algorithmes utilisant un mode d'extraction aveugle. Il reste qu'elle peut également être trop lourde ou complexe à mettre en oeuvre pour certains type de données tatouées, même si des stratégies génériques existent (attaques "oracle").

Revendication frauduleuse des droits d'auteur

Supposons que l'auteur A d'un document X le tatoue et distribue au public le document tatoué $X + A$. Il est facile pour A de prouver que le document $X + A$ a été crée à partir de l'original X qui est en sa possession.

Maintenant, une personne malveillante B connaissant l'algorithme d'insertion du tatouage peut tout aussi facilement créer un document "original" $X + A - B$ à partir du document $X + A$ et prétendre que $X + A$ contient la marque de B imprimée sur l'"original" $X + A - B$. Il est clair que A peut montrer que l'"original" de B, à savoir $X + A - B$, contient sa marque (le terme

$-B$ n'étant à ce point de vue que du bruit auquel on suppose le tatouage robuste), mais l'inverse est également vrai car B pourrait montrer que l'original de A, à savoir X, semble contenir la marque de B par rapport à son faux original $X + A - B$ (la marque de A jouant alors le rôle du bruit). Dans ce cas il est impossible de départager A et B. C'est ce qu'on appelle le problème de "deadlock".

Quelques solutions existent pour résoudre ce problème. La plus simple consiste à déposer l'original chez un tiers de confiance : celui qui l'y dépose le premier est alors réputé être le détenteur du véritable original. Une autre solution consiste à définir l'information cachée comme devant contenir le résultat d'une fonction non-inversible f (e.g. cryptographique) appliquée aux données originales. Dans ce cas, A peut montrer que la marque extraite de $X + A$ contient bien $f(X)$, alors que B ne peut pas montrer que la marque extraite de $X + A$ avec sa clé contient $f(X + A - B)$ car il ne peut pas trouver une marque et une clé telle que la marque extraite de $X + A$ avec la clé contienne $f(X + A - B)$.

Une dernière solution serait de développer un protocole sécurisé permettant de déterminer la date de tatouage d'un document. A ce jour aucun algorithme de ce type n'existe (si on exclut le recours à un tiers de confiance).

Chapitre 2

Etat de l'art du tatouage d'objets 3D basé sur la géométrie

2.1 Classification et description des objets 3D

Plus encore que pour les autres types de documents multimédia (images, audio, vidéo, etc.) il nous semble important, avant de parler de tatouage proprement dit, de préciser au préalable la notion d'objet 3D. Cette notion peut en effet recouvrir des concepts sensiblement différents suivant le contexte. Bien que l'on parle d'objets qui sont avant tout informatiques, ceux-ci sont susceptibles de modéliser des objets 3D utilisés dans des domaines d'application très variés (scientifiques, artistiques, industriels ou autres).

Ainsi il faut bien faire la distinction entre la nature d'un objet 3D, i.e. ce qu'il est censé représenter dans un domaine d'application donné et la manière dont de tels objets sont formalisés de manière mathématique et informatique. Comme illustré par la figure 2.1, un objet 3D est simultanément défini sur plusieurs niveaux allant de l'abstrait au concret. L'espace de tatouage d'un algorithme peut être défini par rapport à n'importe lequel de ces niveaux (comme nous le verrons en section 2.2) et dans chaque cas cela se traduit par des possibilités et des contraintes particulières pour le tatouage.

2.1.1 Notion d'objet 3D

Le but de la liste que nous établissons à présent n'est pas d'être exhaustif mais de montrer la variété des notions potentiellement recouvertes par le terme "objet 3D" et par là même la nécessité de toujours bien préciser le cadre dans lequel on se place.

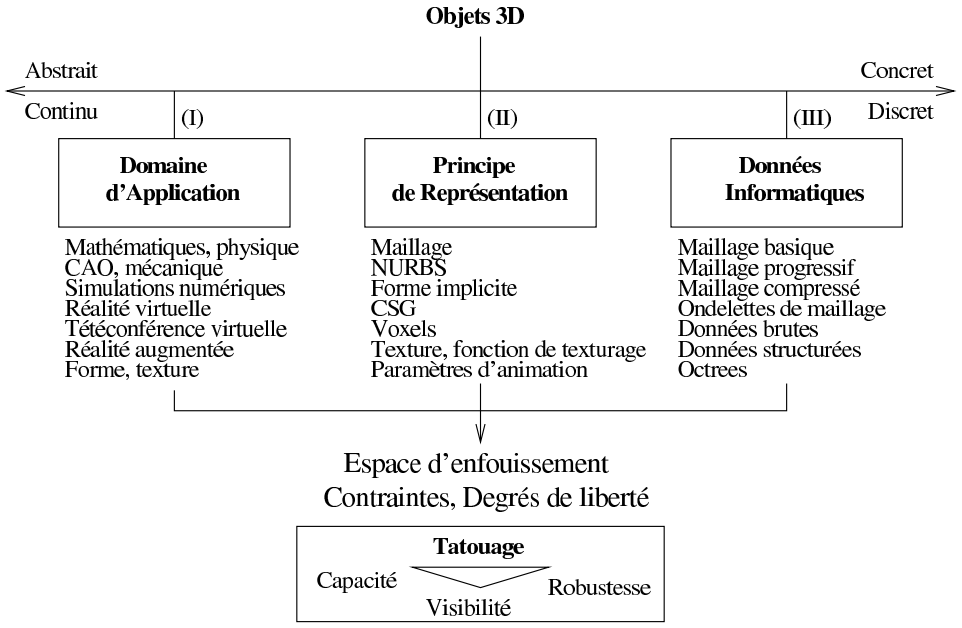


FIG. 2.1 : Notion d'objet 3D en informatique et en tatouage.

Objets 3D mathématiques

Ainsi en mathématiques, un objet 3D peut représenter un ensemble de points quelconques d'un espace vectoriel euclidien à 3 dimensions. Il peut en particulier s'agir d'un ensemble de points formant un volume, une surface, une courbe, ou d'autres entités. Ces objets sont donc définis de manière abstraite comme des "ensembles de points".

Objets 3D en physique

En physique ou en mécanique, un objet 3D consiste souvent en un volume d'un espace tridimensionnel ayant certaines propriétés (matériau, densité, température, etc.) éventuellement dépendantes de la position dans l'espace. Il n'y a pas de beaucoup de différence avec les mathématiques au niveau de l'abstraction, simplement on ne considère pas ici un ensemble de points seul, mais un ensemble de points auquel est associé un certain nombre de propriétés, modélisables mathématiquement par autant de fonctions définies sur cet ensemble de points.

Objets 3D en CAO

Les modèles physiques volumiques sont utilisés dans le cadre de simulations numériques de phénomènes physiques (paragraphe précédent), par exemple pour des calculs de résistance de matériaux, pour la météorologie, etc. Les modèles surfaciques quant à eux, sont le plus souvent utilisés à des fins de

représentation visuelle ou en CAO (Conception Assistée par Ordinateur) en vue de leur fabrication. En CAO, les objets sont décrits essentiellements par une surface géométrique précise et une nomenclature indiquant ses propriétés physiques, son emploi, sa place dans un ensemble plus général, etc.

Objets 3D en imagerie de synthèse

Il existe un autre type important d'objets 3D surfaciques dont le but premier est de permettre la création d'images de synthèse plus ou moins réalistes, que ce soit dans des jeux, des simulateurs de vol, des films, ou dans des applications encore à l'étude telles que la réalité augmentée en imagerie médicale ou en visio-conférence. Généralement, ces objets, qui peuvent représenter des objets réels ou imaginaires, n'ont pas une couleur uniforme (contrairement à la couleur donnée aux objets géométriques issus de la CAO pour leur représentation). La donnée de cette couleur, qui est fonction de la position sur la surface de l'objet fait partie intégrante de l'objet. En outre, d'autres données surfaciques sont parfois nécessaires, telles que des propriétés de matériau ou de réflectance, qui permettent de mieux simuler la manière dont la lumière est réfléchie (voire même réfractée) par la surface de l'objet.

Extensions de la notion d'objet 3D

Il est encore possible d'étendre un peu la notion d'objet 3D en fonction d'autres applications des objets 3D, bien que cela puisse être marginal et non-pertinent dans le cadre particulier du tatouage. Par exemple, dans le cas de scènes complexes qui rassemblent plusieurs objets 3D de manière organisée (que ce soit en CAO ou en images de synthèses) on peut considérer la scène globale comme étant un macro-objet 3D défini par la donnée des objets 3D élémentaires et par tous les autres paramètres nécessaires à la composition de la scène. Un autre example est celui d'un objet 3D susceptible de varier dans le temps : on peut éventuellement inclure dans sa description les paramètres décrivant son potentiel de variabilité. Nous appelons cela un modèle 3D animable lorsqu'il s'agit d'un modèle 3D de visage dont on veut manipuler l'expression faciale, mais de manière générale cela s'appelle un modèle paramétrique, paramétré, ou déformable [79].

2.1.2 Représentation informatique des objets 3D

Nous venons d'établir les différentes notions d'objet 3D rencontrées dans plusieurs domaines. Il s'agit encore de notions abstraites mais lorsqu'on doit les manipuler il faut utiliser une représentation concrète de ces notions.

La frontière que l'on franchit lorsqu'on quitte le monde abstrait des notions d'objet 3D et des purs concepts ensemblistes mathématiques pour entrer dans celui de l'informatique et du concret est la frontière continu/discret. En effet les objets 3D considérés sont des ensembles infinis de points appartenant à un

espace continu alors que l'information numérique traitée par l'informatique est discrète et finie. Ci-après nous présentons la majorité des méthodes utilisées en informatique pour représenter les objets 3D.

Forme implicite

Une des manières les plus simples de représenter une surface est de la décrire par une équation de la forme $F(x) = 0$, où F est une fonction de $\mathcal{R}^3$ dans $\mathcal{R}$, du moins lorsque cela demande peu de paramètres. C'est par exemple le cas d'un plan qui peut être défini par une équation du type $n.x + d = 0$ où n représente un vecteur normal au plan en question. C'est aussi le cas d'une sphère qui peut être définie par $(x - C)^2 = R^2$, où C en est le centre et R le rayon.

De nombreuses autres surfaces peuvent être ainsi représentées par des équations plus ou moins simples et demandant plus ou moins de paramètres, mais l'emploi d'une telle représentation "implicite" en informatique est généralement réservé aux formes géométriques les plus élémentaires. Une exception concerne les "blobs" [44] dont la représentation implicite est parfois utilisée de pair avec les techniques de rendu par lancer de rayons (ray-tracing).

Forme paramétrique

La deuxième manière de définir une surface de manière typiquement mathématique est par une représentation paramétrique de la forme $M = F(u, v)$ où F est une fonction de deux paramètres réels u et v, variant dans un domaine donné, et à valeurs dans $\mathcal{R}^3$. Des représentations paramétriques particulières sont souvent utilisées en informatique lorsqu'il s'agit de modéliser des morceaux de surfaces courbes : il s'agit des NURBS (Non-Uniform Rational B-Spline), décrites plus loin.

Une description paramétrique de la sphère parmi d'autres est donnée par :

$$M(\theta, \phi) = \begin{bmatrix} x_c + R\sin\theta\cos\phi \\ y_c + R\sin\theta\sin\phi \\ z_c + R\cos\theta \end{bmatrix} \text{ avec } \theta \in [0, \pi] \text{ et } \phi \in [0, 2\pi]. \tag{2.1}$$

Forme discrétisée volumique

Voxels Une manière de définir un volume (ou une surface) est de prendre au sens strict sa définition mathématique la plus basique en tant qu'ensemble de points et de spécifier point par point quels points appartiennent au volume (ou à la surface). Cela n'est évidemment pas possible tant qu'on se place dans un espace continu qui comporte une infinité de points, mais après discrétisation de l'espace, cela devient possible. En informatique on découpe donc une région bornée de l'espace selon une grille cubique en un nombre fini de cubes appelés voxels. Pour définir un volume il suffit alors de lister l'ensemble des voxels qui en font partie. L'inconvénient de cette méthode est que sa précision est

limitée par le degré de discrétisation de l'espace et que le nombre de voxels croît rapidement avec le degré de discrétisation. L'utilisation d'une structure hiéarchique (octrees) permet dans une certaine mesure de pallier à ce problème.

Nuages de points Les nuages de points sont en fait des ensembles finis de points 3D qui peuvent représenter un échantillonnage d'un volume ou d'une surface, mais de manière quelconque et pas nécessairement suivant une grille régulière, au contraire des voxels. Ces données sont généralement obtenues à partir du monde physique à l'aide de scanners.

Carte de profondeur

Une carte de profondeur est la spécification d'une coordonnée d'une surface en fonction des deux autres. Elle peut être donnée sous forme d'une fonction ou d'une image 2D. Si on se donne $Z = f(X, Y)$ dans un repère cartésien il s'agit d'une carte de profondeur orthographique (ou élévation) parfois utilisée pour représenter les modèles numériques de terrain [27]. Si on se donne $r = f(\theta, Z)$ en coordonnées cylindriques il s'agit d'une carte de profondeur cylindrique. Cette représentation est utilisée en sortie du scanner 3D Cyberware [1].

Forme discrétisée surfacique : maillages

Enfin, une manière très courante de représenter une surface est de la définir ou de l'approximer par un ensemble fini de polygones plans, souvent des triangles. Il s'agit de donner un ensemble de sommets ainsi qu'un ensemble de triplets d'indices de sommets qui représentent les triangles à former à partir des sommets. Cette représentation, qu'on appelle un maillage triangulaire, a l'avantage d'être flexible et simple à manipuler, notamment lorsqu'il s'agit de faire un rendu 2D de l'objet. De plus il est relativement facile de convertir n'importe quelle autre représentation en maillage. A ce titre le maillage est souvent considéré comme un "plus petit dénominateur commun" des représentations d'objets 3D et sert de base au développement de nombreux algorithmes agissant sur les objets 3D. Son inconvénient est de ne pouvoir décrire de manière parfaite que des polyèdres et de nécessiter un volume de données important pour approximer correctement les surfaces courbes.

Dans le contexte des maillages triangulaires, le terme "topologie" fait référence au graphe de connexité des sommets et à la manière dont il est décrit (e.g. un ensemble ordonné de points 3D et un ensemble ordonné de triplets d'indices de points) et non pas au type topologique de la surface de l'objet au sens mathématique/géométrique du terme.

La méthode de représentation des objets 3D par maillage fait partie d'un ensemble plus vaste appelé "B-Rep" (Boundary Representation [24]) où on cherche à décrire la frontière de l'objet 3D de manière explicite par une réunion de morceaux de surface. La représentation par NURBS fait aussi partie de cette catégorie.

Arbres CSG

La principale limitation des représentations implicites est qu'elles ne permettent de décrire que des "primitives" géométriques relativement simples. Les maillages triangulaires n'ont pas cette contrainte mais ils ne peuvent décrire certains objets, en particulier les objets courbes, qu'approximativement. Il est néanmoins possible de combiner des objets de base simples décrits sous une forme implicite au moyen d'opérations booléennes. Ces opérations agissent sur les volumes et permettent par exemple de définir une demi-sphère comme étant l'intersection de l'intérieur d'une sphère et du "dessous" d'un plan. En combinant un grand nombre d'opérations il est possible de décrire des objets très complexes bénéficiant de l'exactitude de la représentation implicite des objets de base. L'arbre qui décrit un tel ensemble d'opérations s'appelle un arbre CSG, pour Constructive Solid Geometry. Les composants de base d'un tel arbre peuvent être d'autres objets que des primitives géométriques définies de manière implicite, mais cela n'a peut-être alors plus beaucoup d'intérêt, notamment si ces objets de base sont représentés par des maillages car une représentation par maillage a déjà le potentiel de décrire un objet complexe sans recourir à la combinaison de plusieurs "sous-maillages".

NURBS

Courbes NURBS Les NURBS (Non-Uniform Rational B-Splines) représentent une classe importante de courbes ou de surfaces paramétriques. Une courbe est ainsi définie par son degré $k-1$, par $n+1$ points de contrôle $P_0, ..., P_n$ pondérés par $n+1$ poids $w_0, ..., w_n$, par une suite (non-strictement) croissante de $n+k+1$ valeurs $t_0, t_1, ..., t_{k+n}$, appelée *vecteur des noeuds* et par la formule :

$$M(u) = \frac{\sum_{i=0}^{n} w_i P_i B_{i,k}(u)}{\sum_{i=0}^{n} w_i B_{i,k}(u)} \tag{2.2}$$

pour $t_{k-1} \leq u \leq t_{n+1}$ et où les fonctions $B_{i,k}$ sont définies par récurrence à partir du vecteur des noeuds par :

$$B_{i,0}(u) = 1 \text{ si } t_i \leq u < t_{i+1}, \text{ 0 sinon} \tag{2.3}$$

et

$$B_{i,k}(u) = \frac{u - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(u) \tag{2.4}$$

Surfaces NURBS La description d'une surface NURBS utilise une famille de fonctions de base qui est le produit tensoriel de deux familles de fonctions de base à un seul paramètre $B_{i,k}(u)$ et $B'_{j,l}(v)$. Ces deux familles sont définies respectivement par deux vecteurs de noeuds $t_0, t_1, ..., t_{k+m}$ et $t'_0, t'_1, ..., t'_{l+n}$ comme dans le cas des courbes. En outre la description de la surface nécessite la donnée de $(m+1)(n+1)$ points de contrôle $P_{i,j}$ pondérés par $(m+1)(n+1)$ poids

$w_{i,j}$. La surface est alors définie par :

$$M(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}P_{i,j}B_{i,k}(u)B'_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}B_{i,k}(u)B'_{j,l}(v)} \tag{2.5}$$

Surfaces NURBS restreintes Un inconvénient des surfaces NURBS telles que décrites précédemment est que le domaine de variation des paramètres est a priori un rectangle. Pour permettre une plus grande flexibilité, par exemple la présence de trous dans la surface NURBS, on peut spécifier explicitement un domaine de variation des paramètres (u, v) qui ait une forme arbitraire. En interdisant un certain domaine de valeurs, c'est à dire en créant un trou dans le domaine de variation des paramètres (u, v) on peut ainsi créer un trou correspondant dans la surface décrite. Une telle surface est une surface NURBS restreinte.

En pratique le domaine de variation de (u, v) est défini par une ou plusieurs courbes paramétriques fermées et orientées. L'une d'entre elle est la frontière extérieure et les éventuelles autres sont les éventuels trous intérieurs. Ces courbes peuvent éventuellement être elles-mêmes représentées par des courbes NURBS.

Représentation fréquentielle

La notion de représentation "fréquentielle" des objets 3D est proche des notions de "codage progressif" (détaillée plus loin), "niveau de détail", représentation "multi-résolution" ou encore de décomposition en "ondelettes".

Contrairement à ce qu'on pourrait a priori penser, il ne s'agit pas, du moins dans les techniques utilisées en tatouage d'objets 3D, de représenter la géométrie d'un objet 3D dans un "espace fréquentiel" "dual de l'espace euclidien", comme si on appliquait une transformée de Fourier 3D à l'ensemble des points de l'objet 3D.

Il s'agit de "décompositions en fréquence" qui s'appuient sur une représentation maillée de l'objet 3D et sur une représentation progressive de ce maillage. Nous décrivons ici trois manières de "décomposer un objet 3D en fréquences" utilisées dans les algorithmes de tatouage d'objets 3D décrits plus loin.

Décomposition en ondelettes Une décomposition en ondelettes appliquée aux objets 3D maillés, est utilisée dans l'algorithme 17 décrit en section 2.4, et est détaillée dans l'article correspondant [43]. Nous en présentons ici les principaux résultats.

On considère un objet 3D décrit de manière grossière par un maillage triangulaire de base, par exemple le maillage d'un tétraèdre pour des objets ayant le type topologique d'une sphère.

Ce maillage triangulaire de base est affiné en subdivisant chaque triangle en quatre sous-triangles, et l'opération est répétée jusqu'au niveau voulu.

Quand on passe d'un niveau de précision au suivant, on crée de nouveaux sommets qui par défaut sont les milieux des arrêtes du maillage du niveau de précision de départ (ce qui ne change pas la forme de l'objet). A ces points on associe des vecteurs de déplacement qui sont qualifiés de "coefficients d'ondelettes" (voir l'article [43] pour la justication théorique de cette appellation) et qui permettent d'améliorer le niveau de détail de l'objet 3D.

L'objet est donc décrit par un maillage de base et par un certain nombre de couches de coefficients d'ondelettes. La possibilité de représenter un objet 3D maillé suivant une telle décomposition, dépend de la possibilité de dériver son maillage d'un maillage de base très grossier par subdivision successive des triangles, ce qui n'est pas toujours le cas.

Diagonalisation du laplacien du maillage L'idée d'appliquer une décomposition fréquentielle aux maillages a d'abord été proposée par Taubin [85] qui définit une matrice "laplacien" du maillage. Cette matrice L est donnée par

$$L_{ij} = \begin{cases} 1 & \text{si } i = j \\ -d_i^{-1} & \text{si } (i,j) \text{ est une arête du maillage} \\ 0 & \text{sinon} \end{cases} \tag{2.6}$$

où d_i est la valence du sommet i (i.e. le nombre de sommets adjacents).

Cette matrice est ensuite diagonalisée pour donner n vecteurs propres unitaires w_i. A partir des coordonnées des sommets on forme trois vecteurs de n composantes : (x_i), (y_i) et (z_i). Chacun de ces trois vecteurs est projeté sur la base des n vecteurs propres w_i donnant ainsi trois vecteurs de coefficients fréquentiels (r_i), (s_i) et (t_i). Ces vecteurs peuvent vus comme n vecteurs tridimensionnels (r_i, s_i, t_i).

Ce sont les composantes fréquentielles du maillage. Celles correspondant aux vecteurs propres de plus faible (resp. forte) valeur propre correspondent aux "basse (resp. haute) fréquences" du maillage.

D'autres [72] ont repris la même idée mais en utilisant une matrice laplacien différente, en l'occurence le "laplacien combinatoire" ou matrice de Kirchhoff [16]. Cette matrice est définie par $K = D - A$ où D est une matrice diagonale dont l'élément d_{ii} est la valence du sommet numéro i du maillage et où A est la matrice d'adjacence du maillage définie par $a_{ij} = 1$ si les sommets i et j sont adjacents et $a_{ij} = 0$ sinon.

Maillage incrémental Une représentation incrémentale des maillages triangulaires est décrite dans [49] avec une application au tatouage dans [76].

Le principe consiste à inverser une suite d'opérations de simplification de maillage. Ainsi, étant donné un maillage triangulaire, on réduit le nombre de sommets un par un en effectuant une suite de contractions d'arêtes. Une telle opération consiste à choisir une arête du maillage et à remplacer ses deux extrémités par un seul nouveau sommet. A chaque pas on choisit l'arête et

la position du nouveau sommet qui minimisent la distorsion entre le nouveau maillage et le maillage original.

La mesure de distorsion choisie ainsi que le critère de fin du processus de simplification par contraction d'arêtes peuvent être choisis de diverses manières. Quoi qu'il en soit, à la fin des opérations, on obtient un maillage grossier de base ainsi que la séquence de contraction d'arêtes qui y a conduit à partir du maillage détaillé original.

On peut alors décrire le maillage original par la donnée du maillage grossier et d'une séquence d'opérations de dédoublement de sommet qui est l'inverse de la séquence d'opérations de contraction d'arêtes. En appliquant successivement les opérations de dédoublement de sommet au maillage de base, on remplace à chaque fois un sommet par deux sommets jusqu'à obtenir le maillage détaillé original.

Cette manière de représenter un maillage tient plus du concept de représentation "progressive" (voir ci-après) que du concept de représentation en fréquences. On peut quand même voir une vague analogie si on considère que les premiers dédoublements de sommets (correspondant aux dernières contraction d'arêtes) affectent surtout la forme globale de l'objet, alors que les derniers dédoublements de sommets définissent surtout des détails de manière plus localisée. En ce sens, les dédoublements d'arêtes procèdent des basses fréquences vers les hautes fréquences.

Codage progressif

Le codage progressif envisage l'information de géométrie comme étant potentiellement un flux de données entre un émetteur et un récepteur. Le but est que le récepteur puisse se faire une idée de plus en plus précise de la forme de l'objet transmis au fur et à mesure que les données lui arrivent. Au début la forme de l'objet reçu apparaît très grossière puis elle s'affine jusqu'à devenir parfaitement conforme à l'original.

Cette notion de codage progressif nécessite une organisation particulière de l'information géométrique. En effet, si on transmet des données géométriques sans une organisation spécifiquement progressive, le récepteur aura au mieux une représentation partielle de l'objet (au lieu d'une représentation grossière de l'ensemble) et au pire ne pourra pas du tout interpréter les données tant qu'elles ne sont pas complètes.

Actuellement le codage progressif d'objets 3D s'occupe surtout de maillages triangulaires. Le principe consiste alors souvent à transmettre un maillage grossier puis à envoyer un certain nombre de mises à jour plus précises (sous forme d'un différentiel avec la version qui précède). Un exemple de codage progressif peut être trouvé dans [37].

Une autre manière de voir les choses, mais qui dans le principe est assez proche, consiste à décomposer un objet 3D dans un "domaine fréquentiel" et à envoyer les composantes fréquentielles les unes après les autres. Les basses

fréquences donnent l'allure générale de l'objet et les hautes fréquences donnent les détails.

Information de texture

L'information de texture associée à un objet texturé peut être représentée de diverses manières. Dans le cas d'une texture complexe, voire photo-réaliste, cette texture est généralement représentée par une image couleur ordinaire mais ce qui peut changer est la manière de spécifier comment cette image doit être collée à la surface de l'objet 3D. D'un point de vue mathématique, il s'agit de définir une fonction de la surface de l'objet vers l'image de texture qui indique quel pixel de l'image est attribué à chaque point de la surface de l'objet.

Un moyen est d'utiliser une projection simple (sphérique, cylindrique, ou autre). Pour cela l'image de texture est plaquée sur une sphère (ou un cylindre, etc.) puis l'objet à texturer est placé au centre de la sphère et l'image de texture plaquée sur la sphère est plaquée sur l'objet par une projection centrale passant par le centre de la sphère. Cette méthode est particulièrement adaptée pour texturer des objets ayant une forme géométrique particulière, telle qu'une sphère ou un cylindre, etc., et ne dépend pas de la manière dont la géométrie est décrite.

Un autre procédé est couramment utilisé pour texturer des objets 3D décrits par un maillage triangulaire. Il s'agit d'associer explicitement à chaque sommet du maillage un point dans l'image de texture. Cet appariement entre les sommets du maillage 3D et des points de l'image de texture définit également un appariement entre les triangles du maillage et des triangles dans l'image de texture. Chaque triangle du maillage est texturé par le triangle correspondant dans l'image de texture. Pour trouver le point de texture associé à un point intérieur à un triangle 3D, on utilise ses coordonnées barycentriques par rapport aux sommets du triangle 3D, et on reporte ces coordonnées sur le triangle de texture correspondant, réalisant ainsi une interpolation linéaire des coordonnées de texture par rapport aux coordonnées 3D.

2.2 Application des principes du tatouage aux objets 3D

2.2.1 Tatouage de forme, de données, de texture

Le tatouage consiste à enfouir un message secondaire dans un message primaire, appelé support. Ici nous nous intéressons uniquement au cas où le support est en fait un objet 3D, mais il est déjà nécessaire de distinguer plusieurs interprétations possibles du tatouage d'objets 3D.

Tatouage de la forme

Premièrement, l'objet 3D peut être considéré en tant que forme géométrique indépendante d'un choix de représentation informatique particulier. Tatouer cet objet consiste dans ce cas à enfouir une information dans sa forme (ce qui implique de le déformer dans une certaine mesure) et à pouvoir récupérer cette information indépendant d'un nouveau format dans lequel l'objet pourrait avoir été converti.

Tatouage de données

Deuxièmement, l'objet 3D peut être considéré comme une donnée informatique ayant un format particulier (par exemple sous forme de maillage). Le tatouage consiste alors à enfouir de l'information dans ces données (en les modifiant) tout en préservant au mieux la forme de l'objet qu'elles décrivent.

Tatouage d'attributs

Enfin, l'objet 3D peut être plus riche qu'un objet purement géométrique s'il lui est par exemple associé une image de texture réaliste, ou d'autres informations. Le tatouage peut alors consister à enfouir de l'information dans ces données supplémentaires.

Commentaires

Chacune de ces possiblités a son intérêt et ses contraintes. Ainsi on conçoit que pour protéger la forme d'un objet 3D en tant que création artistique il est plus naturel de tatouer la forme intrinsèque d'un objet 3D indépendemment de sa représentation informatique bien que cela soit certainement plus difficile. Tatouer un objet dans son information de texture, lorsque cela est possible, permet d'autres applications que celles permises par le seul tatouage de la forme géométrique, comme par exemple la possibilité de récupérer l'information cachée à partir d'images de synthèse 2D où figure l'objet.

2.2.2 Attaques destructives, non-destructives

Le tatouage d'un objet 3D peut souffrir de certaines manipulations, malveillantes ou non. Lorsque ces manipulations ne changent pas de manière significative le contenu de l'objet (i.e. généralement sa forme et parfois d'autres caractéristiques telles que la texture ou l'apparence) on parle d'attaque non-destructive, sinon on parle d'attaque destructive.

2.2.3 Tatouage fragile, robuste

Un algorithme de tatouage peut permettre d'extraire le tatouage d'un objet 3D tatoué après que celui-ci ait subit certaines attaques non-destructives (détaillées plus loin en section 2.3).

On peut classer ces attaques en deux catégories : les attaques qui modifient la sémantique de l'objet (i.e. la forme intrinsèque) et les attaques qui n'en modifient pas la sémantique (e.g. conversion des données).

Ainsi, tronquer la géométrie d'un objet 3D modifie sa sémantique alors que convertir une description NURBS en maillage n'en modifie pas la sémantique.

Un algorithme de tatouage peut être robuste à un certain nombre d'attaques, y compris des attaques qui modifient la sémantique de l'objet, et ce afin de préserver une trace de la marque dans un objet même fortement altéré.

Un algorithme qui n'est pas robuste est fragile. Mais en tatouage le terme de tatouage fragile s'applique spécifiquement à un tatouage qui est fragile vis à vis des attaques qui modifient la sémantique de l'objet et qui par ailleurs peut ou non être robuste aux autres attaques qui ne modifient pas la sémantique de l'objet.

2.2.4 Compromis entre capacité, visibilité, robustesse

Comme dans tous les autres domaines d'application du tatouage numérique, un algorithme de tatouage d'objets 3D doit satisfaire au mieux trois containtes deux à deux antagonistes : la capacité, la visibilité, et la robustesse.

Capacité

La capacité représente la quantité d'information qu'on peut dissimuler dans l'objet 3D, que ce soit dans la forme géométrique intrinsèque de l'objet, dans une représentation informatique particulière, ou autre, selon le type d'algorithme et l'application visée.

Visibilité

La visibilité du tatouage signifie la perceptibilité du tatouage dans la forme de l'objet, dans la structure des données informatique, ou même dans des vues 2D. Idéalement le tatouage ne doit pas modifier la forme ou l'apparence d'un objet de manière perceptible.

Robustesse

La robustesse est la capacité du tatouage à résister à certaines manipulations d'objets 3D, malveillantes ou non. Il peut s'agir d'une simple rotation de l'objet, d'une mise à l'échelle, d'un remaillage, d'une conversion de format, d'une découpe, etc., et de la projection perspective dans le cas d'un objet 3D texturé. Ces attaques et manipulations sont détaillées en section 2.3.

2.2.5 Mode d'extraction aveugle, non-aveugle

Il existe, dans le cas des objets 3D comme dans le cas général, plusieurs modes d'extraction d'un tatouage. On distingue en particulier le mode d'ex-

traction aveugle et le mode d'extraction non-aveugle.

Dans le mode d'extraction aveugle, il suffit de disposer de l'objet 3D tatoué (ayant éventuellement subi de légères modifications tolérées par l'algorithme de tatouage) pour en extraire le tatouage.

Dans le mode d'extraction non-aveugle, en plus de l'objet dont on veut extraire le tatouage, il faut disposer de tout ou partie de l'objet original non-tatoué.

2.2.6 Applications possibles

On peut concevoir plusieurs protocoles d'utilisation du tatouage d'objet 3D qui sont essentiellement les mêmes que ceux déjà envisagés dans le cadre du tatouage en général. Il peut quand même y avoir quelques petites variantes en fonction de la nature exacte des objets 3D considérés.

Intégrité

La première application permet de vérifier que l'information dont on dispose est conforme à l'original, c'est à dire qu'elle n'a pas subi de modifications qui modifient son interprétation.

Cette information peut consister en la forme intrinsèque d'un objet 3D, en une représentation informatique particulière d'un tel objet, ou en une vue 2D d'un objet texturé. Pour en assurer l'intégrité il faut un algorithme de tatouage fragile (voir plus haut, section 2.2.3). Ensuite on s'assure de son intégrité en vérifiant qu'on peut extraire le tatouage, car une modification de la sémantique de l'objet est supposée détruire le tatouage.

Diffusion

Les autres applications demandent plutôt un tatouage robuste. Il peut s'agir de cacher dans un objet 3D une information qui représente un numéro de série (i.e. "fingerprint") attribué à un client pour une copie de cet objet, afin de pouvoir déterminer, au cas où on trouve une copie illégale, d'où provient la fuite. Le tatouage doit alors pouvoir résister à certaines opérations visant à le détruire.

Droits d'auteur

La protection des droits d'auteur est souvent le but principal du tatouage. A partir d'une copie illégale d'un objet 3D on cherche à extraire les informations qui y ont été enfouies en vue d'établir l'infraction. Ces informations sont essentiellement relatives à l'auteur et à la filière de distribution. Il peut être nécessaire de disposer de l'original pour extraire ces informations.

Utilisation

Le tatouage d'un objet 3D s'attache souvent à protéger cet objet en tant que tel, et pour pouvoir vérifier qu'un objet est tatoué, il faut disposer de cet objet. Or il peut arriver qu'on ne dispose pas de l'objet lui-même mais seulement d'une représentation visuelle de l'objet. Le problème est alors non pas de protéger l'objet lui-même, mais son utilisation dans des images de synthèse. Pour cela il faut pouvoir extraire le tatouage à partir d'une simple vue 2D de l'objet, au besoin en utilisant également la connaissance de l'objet 3D original.

Pour des objets 3D photo-réalistes présentant une très riche information de texture, cela est possible en plaçant le tatouage dans l'information de texture, car l'information de texture est naturellement présente dans les vues 2D de l'objet.

2.3 Manipulations et attaques d'objets 3D

2.3.1 Renumérotation des sommets ou des triangles

Généralement un maillage triangulaire est défini par une liste ordonnée de sommets et par une liste ordonnée de triangles. L'ordre choisi pour lister les sommets et triangles est arbitraire et il est toujours possible de changer cet ordre sans modifier la forme de l'objet ainsi décrit.

Cette opération presque anodine peut mettre en défaut des algorithmes de tatouage qui se basent sur une représentation maillée particulière d'un objet 3D et sur un ordre particulier des sommets et triangles.

Néanmoins, à condition de travailler en mode non-aveugle, on peut toujours espérer effectuer un recalage entre un objet 3D tatoué dont l'ordre des sommets a été modifié et sa version originale non modifiée. Une fois les sommets de l'objet tatoué réordonnés par rapport au modèle original on peut tenter d'en extraire la marque.

2.3.2 Similitudes, transformations affines et projectives

Le changement du repère orthonormé dans lequel est décrit un objet 3D, résulte en l'application à l'objet d'une translation, d'une rotation ou d'un changement d'échelle. Ces transformations ne changent pas la forme de l'objet mais peuvent détruire le tatouage s'il n'est pas marqué dans des invariants à ces transformations.

Par exemple un algorithme qui utilise la direction des normales à l'objet comme espace d'enfouissement de la marque risque d'avoir des problèmes si l'objet tatoué subit une rotation, car les directions des normales seront alors changées. Mais ce problème peut être résolu si on peut réorienter l'objet par rapport à l'objet original (c'est à dire le recaler, comme dans le cas précédent). On peut aussi envisager d'utiliser le repère formé par les axes principaux d'inertie avant le tatouage lorsque cela est possible, et en supposant que le tatouage

a un impact négligeable sur ce repère.

A ces tranformations on peut également ajouter les tranformations affines et même projectives, qui nécessitent soit une étape de recalage, soit de trouver des espaces d'enfouissement de la marque invariants par rapport à ces tranformations.

2.3.3 Ajout de bruit

L'ajout de bruit sur les coordonnées des sommets (lorsqu'il s'agit d'un maillage par exemple) ou sur les paramètres numériques décrivant l'objet 3D est une attaque non-spécifique aux objets 3D. Il peut rendre l'extraction du tatouage peu fiable, mais il dégrade d'autant la qualité de l'objet.

Certains algorithmes résistent mieux que d'autres au bruit ou à certains types de bruit, mais à partir d'un moment protéger un objet qui est trop bruité, et donc relativement peu fidèle à l'original, n'a plus beaucoup d'intérêt. On retrouve ici la notion qu'une attaque peut être destructive ou non et que le tatouage s'intéresse surtout au dernier cas.

2.3.4 Remaillage, compression

Le remaillage, c'est à dire en général le changement de représentation maillée, mais aussi la description par un maillage d'un objet qui jusqu'alors était décrit d'une autre manière, est en fait un rééchantillonnage de la surface de l'objet.

Cette opération introduit donc nécessairement un bruit de rééchantillonnage au niveau de la géométrie de l'objet (cf. attaque précédente). En outre, si l'objet était décrit précédemment par un autre maillage, l'information "topologique" (nombre et ordre des sommets, etc.) associée a ce maillage est perdue. Un algorithme de tatouage qui s'appuie sur ces données ne peut donc pas résister à une opération de remaillage, à moins de recourir au recalage et au rééchantillonnage par rapport à la représentation initiale de l'objet (ce qui implique un mode d'extraction non-aveugle).

Un cas particulier de remaillage est la simplification du maillage (réduction du nombre de sommets et de triangles) avec pour but de réduire la quantité d'information nécessaire à décrire l'objet 3D et le temps nécessaire à le transmettre sur un réseau. Il s'agit alors d'une compression avec pertes de l'objet. Suivant la force de la compression, la forme de l'objet et le tatouage sont plus ou moins altérés. Il s'agit d'une attaque analogue à l'attaque de compression d'images fixes (par exemple une conversion en JPEG de facteur de qualité 75%).

2.3.5 Découpe

La découpe d'un objet 3D peut-être interprétée à différents niveaux. Au niveau géométrique "abstrait" il s'agit d'enlever une partie de la surface de

l'objet. Il y a ensuite les répercussions de cette opération sur la description informatique de l'objet. Par exemple s'il s'agit d'un maillage triangulaire, la découpe consistera essentiellement à supprimer un certain nombre de sommets et triangles, ce qui, en plus de réduire la surface de l'objet décrit, modifie la liste (i.e. l'ordre) des sommets et triangles du maillage comme pour l'attaque de "renumérotation".

Dans tous les cas, un moyen de rendre une marque robuste à une découpe de l'objet est de cacher cette marque plusieurs fois dans plusieurs parties indépendantes de l'objet.

2.3.6 Attaque par collusion, effacement, ajout de marque

Il s'agit ici encore d'une classe d'attaques non-spécifiques aux objets 3D qui exploite les faiblesses de certains protocoles de tatouage. Notamment dans l'attaque par collusion, il s'agit de moyenner plusieurs version de l'objet tatouées différemment, ou d'assembler des morceaux pris séparément dans chaque version. Ainsi la version moyenne ou composite ne comporte qu'une moyenne de plusieurs marques ou plusieurs morceaux alternés de différentes marques, si bien qu'il est impossible d'extraire une marque en particulier.

Cette attaque est particulièrement pertinente dans le cas d'objets "composites", c'est à dire d'assemblages structurés d'objets 3D plus simples, comme en construction mécanique. On peut alors recomposer l'ensemble en prenant les différents objets de base dans plusieurs versions tatouées de l'ensemble. C'est surtout vrai si on considère que les objets simples contiennent chacun peu d'information et ne sont pas susceptibles d'être tatoués et protégés individuellement.

2.3.7 Attaques d'objets texturés

Les objets texturés sont sujets à des attaques différentes lorsque c'est la texture qui porte l'information cachée. Dans ce cas toutes les attaques précédentes sur la géométrie n'ont pas ou peu d'impact sur l'information cachée dans la texture et sur l'information qui peut être récupérée des vues 2D de l'objet.

Ceci étant dit, il y a de nombreuses attaques possibles sur la texture. Il y a d'abord toutes les attaques d'images fixes (rééchantillonnage, modification des couleurs, déformations géométriques locales, etc.). Puis il y a l'attaque induite par le rendu 2D, laquelle s'apparente à une combinaison d'attaques d'images fixes.

2.4 Liste des algorithmes de tatouage d'objets 3D

Pour terminer ce tour d'horizon du tatouage d'objets 3D nous proposons ici une liste et une brève description des algorithmes connus. Une telle liste peut être pratiquement exhaustive dans la mesure où le nombre d'algorithmes

TAB. 2.1 : Publications sur le tatouage et sur le tatouage d'objets 3D. Source : INSPEC (mars 2004).

Année	'94	'95	'96	'97	'98	'99	'00	'01	'02	'03
Tatouage	0	2	21	54	123	211	336	398	612	339
Tatouage 3D	0	0	0	2	9	10	11	11	27	17

existants à l'heure actuelle est assez limité (voir le tableau 2.1 pour avoir une idée de l'importance du tatouage 3D par rapport au tatouage en général).

Le principal critère retenu pour classer ces algorithmes est l'espace d'enfouissement utilisé. Ensuite on s'intéresse au mode d'extraction et à la robustesse à certaines opérations, par exemple au remaillage. Selon ces critères on peut distinguer plusieurs classes qui parfois ne contiennent qu'un seul algorithme.

Algorithme 1 *Ichikawa, Chiyama et Akabane [52] : maillage, permutation des sommets ou triangles*

Cet algorithme permet de coder de l'information supplémentaire dans un maillage décrit par une liste de sommets et une liste de triangles. Pour cela il modifie l'ordre des sommets, l'ordre des triangles ou l'ordre des sommets d'un triangle dans le fichier qui décrit l'objet. Cet algorithme ne modifie ni la géométrie de l'objet, ni la topologie du maillage. Par rapport à notre classification des espaces d'enfouissements (Fig. 2.1) cet algorithme se place au niveau "le plus bas" (Fig. 2.1, colonne (III) : représentation concrète des données), il n'est donc pas robuste à un changement de codage du maillage ni à un changement de format des données (e.g. NURBS) ou un remaillage ; il a néanmoins son intérêt propre pour certaines applications (e.g. stéganographie).

Algorithme 2 *Harte et Bors [45] : maillage, déplacement des sommets*

Cet algorithme de tatouage est basé sur les sommets et pas sur la connexité des triangles. Un message binaire est défini au moyen d'une clé secrète. Un ensemble de sommets éligibles pour coder chacun un bit est déterminé. A chacun de ces sommets est associé un voisinage ellipsoïdal. Pour coder un 0, un sommet est déplacé à l'extérieur de son voisinage et pour coder un 1, il est déplacé à l'intérieur, le tout en choisissant le déplacement qui minimise la distorsion introduite ainsi que la probabilité d'erreur à l'extraction. L'extraction consiste en effet à lire si un sommet se trouve à l'intérieur ou à l'intérieur du voisinage qui lui est associé. A l'extraction on doit savoir quels étaient les sommets retenus pour coder la marque.

Algorithme 3 *Yeung et Yeo [96, 95] : maillage, tatouage fragile*

Pour vérifier l'intégrité d'un maillage triangulaire, Yeung et Yeo ont développé un tatouage fragile qui perturbe légèrement chaque sommet. Cet algorithme est aveugle car il ne nécessite pas l'objet original pour l'extraction.

L'étape d'insertion procède ainsi :

1. Calcul d'un double index de position $L = (L_x, L_y)$ pour chaque sommet v
   - Définir le centroïde s de l'ensemble des sommets adjacents à v et d'indices de numérotation inférieurs à celui de v
   - Convertir les coordonnées de s en entiers
   - Combiner ces trois entiers pour obtenir L_x et L_y
2. Calcul d'un triple index de valeur $p = (p_1, p_2, p_3)$ pour chaque sommet v
   - Convertir les coordonnées de v en un triplet d'entiers
3. Générer une séquence binaire à partir d'une clé secrète K et des index de valeur $p(v)$. Chaque $p(v)$, donné en entrée d'une table de conversion paramétrée par K, fournit une séquence binaire $K(p(v))$.
4. Modifier légèrement la géométrie du modèle (i.e. les sommets v) de sorte que le bit $W(L(v))$ de la marque W (définie comme une imagette noir et blanc à deux dimensions) soit égal à $K(p(v))$.

L'étape de vérification consiste à calculer pour chaque sommet v de l'objet, les valeurs $K(p(v))$ et $W(L(v))$, à les comparer afin de tester la validité du point, puis à évaluer le taux de corrélation entre la marque insérée et la marque extraite :

$$c = \frac{|\{v : K(p(v) \neq W(L(v))\}|}{|\{v\}|} \tag{2.7}$$

Si l'objet n'a subi aucune modification, le taux de corrélation c vaut 1. Cette méthode n'est pas robuste à la renumérotation des sommets de part la définition du centroïde s lors de l'étape d'insertion.

Algorithme 4 *Wagner [90] : maillage, taille des normales*

Cet algorithme enfouit le tatouage dans les longueurs des "normales" définies aux sommets du maillage (la normale étant définie comme étant le vecteur entre le barycentre des voisins du sommet et le sommet lui-même). Modifier ces "normales" implique au bout du compte de déplacer les sommets. L'utilisation d'une norme invariante par transformation affine permet au tatouage d'être robuste aux transformations affines de l'objet 3D.

Algorithme 5 *Benedens [11, 13] : maillage, répartition des normales*

L'information support utilisée est la répartition des normales des facettes de l'objet 3D. La sphère unité, représentant l'ensemble des directions orientées, est divisée en régions. Chaque normale de facette tombe dans une de ces régions. La dispersion des directions des normales tombant dans une région donnée est utilisée pour coder un bit. Pour coder un 0 on fait par exemple en sorte de réduire cette dispersion et on fait le contraire pour coder un 1. Pour modifier ainsi la répartition des normales on joue sur la position des sommets. Les positions des régions choisies pour coder un bit sont déterminées par une clé secrète.

Algorithme 6 *Kwon et al. [59] : maillage, répartition des normales*

Cet algorithme est une amélioration du précédent. L'information support du tatouage est l'histogramme de répartition des normales sur la sphère unité. Il est rendu robuste aux découpes en cachant la marque dans plusieurs sous-parties. Il se veut aveugle et robuste aux remaillages et aux similitudes.

Algorithme 7 *Benedens [12] : maillage, "Vertex Flood"*

Cette méthode permet de cacher une marque de grande capacité dans un objet 3D représenté par un ensemble de triangles. L'application visée est la protection des droits d'auteur et le dépistage de copies illégales. Cet algorithme repose sur une modification de la distance entre les sommets du modèle et le centre de gravité d'un triangle de référence. Elle altère uniquement la géométrie du modèle.

Algorithme 8 *Yu, Ip et Kwok [98] : maillage, position des sommets*

Cet algorithme répartit les sommets du maillage dans N sous-ensembles, définis d'une manière pseudo-aléatoire paramétrée par une clé secrète. Le tatouage est codé dans la distance d'un sommet par rapport au barycentre du sous-ensemble auquel il appartient. Cette méthode est robuste à de nombreuses attaques, notamment à la découpe et à l'ajout de bruit et est rendue robuste au remaillage par une étape de recalage/rééchantillonnage par rapport à l'objet original (ce qui implique un mode d'extraction non-aveugle).

Algorithme 9 *Benedens [12] : maillage, "Triangle Flood"*

Ceci est une autre méthode de haute capacité. Les applications visées sont les mêmes que pour l'algorithme 7. L'algorithme génère un ordre de parcours des triangles et cache la marque dans les hauteurs de ces triangles en modifiant légèrement la position de leurs sommets.

Algorithme 10 *Ohbushi, Masuda et Aono [68] : maillage, "Triangle Similarity Quadruple" (TSQ)*

Cet algorithme est basé sur la modification de données géométriques d'un maillage triangulaire. Pour chaque triangle on considère un couple d'invariants géométriques, par exemple le rapport des longueurs de deux cotés et le rapport de la hauteur sur la longueur de la base. On code un entier dans un triangle en modifiant légèrement les invariants considérés. En fait l'unité de base pour coder de l'information est constituée de quatre triangles adjacents. L'entier inscrit dans l'un des triangles indique si l'information du quadruplet est une information valide. Deux autres triangles contiennent de l'information utile, et le dernier triangle contient un indice de position de manière à ordonner entre elles les informations utiles lues dans les divers quadruplets pour reconstituer un message complet.

Algorithme 11 *Ohbushi, Masuda et Aono [68] : maillage, "Tetrahedral Volume Ratio embedding" (TVR)*

Cet algorithme vise également les maillages triangulaires. L'information cachée est codée en modifiant un invariant affine qui est le rapport des volumes de deux tétraèdres donnés. Un tétraèdre est déterminé par une arrête et ses deux triangles adjacents.

Algorithme 12 *Ohbushi, Masuda et Aono [68] : maillage, "Triangle Strip Peeling Symbol sequence embedding" (TSPS)*

Cet algorithme utilise la topologie du modèle polygonal de l'objet 3D et n'en modifie pas la géométrie apparente. Certains sommets sont dupliqués pour "découper" les contours d'une bande de triangles. Cette bande est définie à partir d'un triangle initial et le parcours vers les triangles suivants est déterminé en fonction des bits du message binaire à cacher.

Algorithme 13 *Cayre et Macq [33] : maillage, modification de TSPS*

Cet algorithme considère que chaque triangle est une variable à deux états. L'état d'un triangle est déterminé par la position de la projection d'un sommet donné sur le côté opposé, et cet état peut être modifié en déplaçant légèrement le sommet concerné. L'ensemble des triangles qui codent le message binaire à cacher est une bande de triangles déterminée par une procédure analogue à celle utilisée dans l'algorithme précédent (TSPS) : on part d'un triangle initial et on se déplace vers les triangles adjacents en fonction d'une séquence binaire pseudo-aléatoire engendrée par une clé secrète. La différence par rapport à l'algorithme TSPS est que ce n'est pas la forme de la bande de triangles qui code la marque. De plus, la bande de triangle n'est pas découpée de la surface de l'objet, ce qui ne modifie pas la topologie de l'objet, et ne la rend pas topologiquement détectable.

Algorithme 14 *Ohbushi, Masuda et Aono [68] : maillage, "Polygon Stencil Pattern" (PSP)*

Etant donné un objet 3D décrit par un maillage, un motif ou un logo peut y être caché en découpant simplement un ruban de cette forme dans la surface de l'objet. Comme pour l'algorithme TSPS seule la topologie de l'objet est modifiée. L'algorithme est assez robuste aux simplifications de maillage.

Algorithme 15 *Ohbushi, Masuda et Aono [68] : maillage, "Mesh Density Pattern embedding"*

Cet algorithme joue localement sur la densité de triangles pour faire apparaître un motif dans un maillage. La détection du motif est visuelle : il suffit d'observer le modèle en mode "fil de fer". Le tatouage est par contre invisible si l'objet est représenté en mode texturé. Ce tatouage est robuste à toutes les opérations géométriques et assez résistant aux simplifications de maillage.

Algorithme 16 *Koh et Chen [57] : maillage, transmission progressive*

Cet algorithme vise à tatouer le flux de données qui sert à transmettre un maillage de manière progressive. Chaque sommet est en fait transmis l'un après l'autre dans un ordre qui justifie le terme de progressif. Le tatouage proprement

dit s'applique au signal unidimensionnel constitué par le flux de données des coordonnées des sommets. La détection de la marque a lieu sur réception du flux de données et le résultat est soit que le flux est marqué, soit qu'il est non marqué (tatouage sans capacité).

Algorithme 17 *Hanai, Date et Kishinami [43] : maillage, décomposition en ondelettes*

Cet algorithme effectue une décomposition en ondelettes du maillage. Le tatouage est effectué dans les coefficients d'ondelette et la géométrie tatouée est obtenue par reconstruction à partir des coefficients d'ondelette modifiés. Seuls les coefficients ayant une amplitude assez grande sont légèrement modifiés. Cette politique est basée sur la notion d'étalement de spectre en tatouage [21].

Algorithme 18 *Ohbuchi, Mukaiyama et Takahashi [72] : maillage, décomposition spectrale*

Cet algorithme de tatouage opère dans le domaine "spectral" obtenu d'après une diagonalisation d'une matrice "laplacien combinatoire" du maillage triangulaire (cf. section 2.1.2.0.0). La diagonalisation d'une matrice de grande dimension (égale au nombre de sommets du maillage) est une opération très lourde et numériquement instable, c'est pourquoi le maillage est divisé en régions comportant un nombre "raisonnable" de sommets et chaque région est traitée séparément. Cette méthode non-aveugle est robuste à de nombreuses attaques ; elle emploie en particulier une étape de recalage/rééchantillonnage par rapport à l'objet original pour assurer la robustesse aux remaillages.

Algorithme 19 *Cayre et al. [18] : maillage, décomposition spectrale*

Cet algorithme ressemble fortement au précédent. La matrice laplacien utilisée pour obtenir la décomposition fréquentielle du maillage est celle proposée originellement par Taubin [85] plutôt que le laplacien combinatoire. Mais l'originalité par rapport à l'algorithme 18 est que chaque région soumise à la décomposition spectrale est d'abord "rééchantillonnée" pour obtenir un graphe de connexité des sommets prédéfini et régulier (la valence de chaque sommet étant de 6 sauf aux bords). Le graphe de connexité final étant défini à l'avance, la matrice de décomposition fréquentielle (i.e. diagonalisation du laplacien) peut être précalculée, d'où un gain de temps. L'autre différence de cet algorithme par rapport au précédent est qu'il n'emploie pas de procédure de recalage/rééchantillonnage avant l'extraction du tatouage et qu'il n'est de ce fait pas robuste aux remaillages, mais ceci est uniquement un choix "stratégique" sans rapport avec la technique d'enfouissement au coeur de l'algorithme.

Algorithme 20 *Praun, Hoppe et Finkelstein [76] : maillage, représentation progressive*

Cet algorithme utilise la représentation progressive de maillage introduite dans [49] et associe une fonction de base à chaque opération de duplication de sommet. Cette fonction définit une manière de modifier la géométrie de l'objet

en déplaçant les sommets dans un certain voisinage du sommet dupliqué. Pour le tatouage, il choisit les opérations de duplication de sommets qui modifient le plus la géométrie et utilise chaque fonction de base associée pour modifier légèrement la géométrie avec une amplitude qui code de l'information. L'extraction de la marque se fait après un recalage et un rééchantillonnage de l'objet tatoué avec l'objet original.

Algorithme 21 *Yin et al. [97] : maillage, représentation multi-résolution*

Les auteurs de cet algorithme ont voulu généraliser l'idée de l'algorithme précédent pour l'appliquer à d'autres représentations multi-résolution d'un maillage triangulaire. L'exemple d'algorithme proposé est néanmoins légèrement différent de l'algorithme 20 : il construit une représentation multi-résolution d'un maillage triangulaire et choisit un maillage grossier particulier pour y insérer la marque par modification géométrique (au lieu d'utiliser un ensemble de niveaux de détail). L'extraction se fait après une étape de recalage/rééchantillonnage par rapport à l'objet original pour être robuste à de nombreuses attaques, dont les conversions de format.

Algorithme 22 *Song, Cho et Kim [82] : maillage, carte de profondeur cylindrique*

Cet algorithme calcule une carte de profondeur cylindrique associée à un maillage. Le repère cylindrique utilisé est aligné sur les axes principaux d'inertie du modèle 3D. La carte de profondeur est tatouée en utilisant un algorithme de tatouage d'images 2D a priori quelconque, puis les modifications de la carte de profondeur sont répercutées sur les coordonnées des sommets du maillage original.

Algorithme 23 *Benedens [14] : NURBS*

Cet algorithme utilise le même principe que l'algorithme 11 (Tetrahedral Volume Ratio embedding) en l'améliorant un peu. Mais le plus intéressant est qu'il utilise un algorithme de tatouage de maillages pour tatouer des NURBS. A cet effet une surface NURBS est transformée en maillage, l'algorithme original est appliqué au maillage, et les déplacements des sommets du maillage induits par le tatouage sont répercutés sur les coefficients de la description NURBS initiale pour tatouer le modèle NURBS.

Algorithme 24 *Ohbuchi, Masuda et Aono [71, 67] : NURBS, reparamétrage*

Cet algorithme permet de cacher une information dans une courbe ou une surface NURBS en faisant un changement de paramètre selon une fonction homographique. Ce changement de paramètre possède trois de degrés de liberté qui sont réduit à un seul par certaines contraintes. Ce degré de liberté code l'information cachée. Le changement de paramètre implique de modifier aussi le vecteur des noeuds et l'ensemble des poids pour laisser la géométrie inchangée. L'avantage de cette méthode est qu'elle ne modifie pas la géométrie de l'objet,

le revers de la médaille étant bien entendu que la géométrie intrinsèque de l'objet n'est pas protégée.

Algorithme 25 *Lee, Cho et Kim [61] : NURBS, cartes de profondeur orthographiques*

Cet algorithme calcule trois cartes de profondeurs orthographiques d'un modèle NURBS puis tatoue ces cartes de profondeur en utilisant un algorithme de tatouage d'images 2D. Les modifications des cartes de profondeur induites par le tatouage sont répercutées sur les coordonnées des points de contrôle du modèle NURBS. L'orientation du repère utilisé pour les projections orthographiques est définie par une clé secrète.

Algorithme 26 *Fornaro et Sanna [35] : arbres CSG*

Cet algorithme permet de dissimuler de l'information dans un modèle 3D décrit par un arbre de construction CSG. Pour cela des noeuds invisibles sont ajoutés, par exemple une sphère de rayon nul. L'information dissimulée dans ces noeuds est une valeur de hashage du modèle 3D original cryptée à l'aide de la clé secrète d'un algorithme de cryptage asymétrique. L'application visée est donc l'authentification de l'objet par comparaison de la valeur de hashage calculée sur cet objet avec la valeur de hashage dissimulée dans ce même objet (laquelle doit être décryptée au moyen de la clé publique).

Algorithme 27 *Ohbuchi, Masuda et Aono [70, 69] : divers, attributs d'objets 3D*

Ces articles indiquent la possibilité de tatouer un objet 3D dans certains attributs attachés à sa géométrie tels que des coordonnées de texture ou des paramètres de couleur ou d'opacité associés à chaque sommet, ligne, ou facette. [69] présente des résultats préliminaires d'un algorithme de tatouage dans les coordonnées de texture. Il est en outre fait mention de la possibilité d'utiliser d'autres attributs comme espace de tatouage, comme les paramètres de scènes VRML que sont les interpolateurs, caméra, etc., ainsi que la possibilité de tatouer dans les tables d'animations des modèles 3D animables.

Algorithme 28 *Hartung, Eisert et Girod [47] : divers, paramètres d'animation*

Cet algorithme est à la frontière du domaine couvert par le présent état de l'art dans la mesure où il s'intéresse à des séquences vidéo de modèles 3D de visages animés par des flux de paramètres au format MPEG-4, et où il propose de tatouer le flux de paramètres d'animation et non pas l'objet 3D lui-même.

Conclusions Le tableau 2.2 résume quelques caractéristiques des algorithmes que nous venons de voir. On peut retenir en particulier que la plupart des algorithmes agissent sur des représentations de type maillage. Dans cette catégorie on trouve :

1. Un algorithme qui modifie l'organisation des données informatiques du maillage, sans modification de géométrie ou de topologie (algorithme 1).
2. Des algorithmes qui modifient uniquement la topologie (algorithmes 12, 14 et 15).
3. Des algorithmes qui modifient la géométrie en tenant compte des données topologiques. Parmis ceux-ci certains font le choix d'être aveugles mais pas robustes aux remaillages (algorithmes 2, 3, 4, 7, 9, 10, 11, 13, 16 et 17) et les autres font le choix d'être robustes aux remaillages via une opération de recalage/rééchantillonnage préalable à l'extraction, mais sont de ce fait non-aveugles (algorithmes 8, 18, 19, 20, 21).
4. Des algorithmes qui modifient des invariants géométriques (non liés à un maillage particulier) et qui peuvent par conséquent être théoriquement aveugles et robustes aux remaillages (mais a priori pas aux découpes). Il s'agit des algorithmes 5, 6 et 22.

Les algorithmes restants, qui ne tatouent pas des maillages, sont plus rares et représentent pratiquement chacun une catégorie différente. On peut toutefois noter que l'algorithme 1 (maillage) n'est pas le seul à enfouir les données au niveau "le plus bas" : l'algorithme 26, qui opère sur des arbres CSG, fait de même.

Une dernière remarque concerne la possibilité, au moins théorique, de convertir un algorithme de tatouage de maillage modifiant la géométrie en un algorithme de tatouage de NURBS (modifiant aussi la géométrie) et réciproquement. Cette possibilité a d'ailleurs déjà été exploitée par l'algorithme 23 : pour tatouer des NURBS, il les convertit en maillage, applique un algorithme de tatouage de maillage, et repasse au format NURBS. Il s'agit d'une généralisation de la notion selon laquelle l'espace de tatouage peut être différent de l'espace de représentation des données 3D originales.

TAB. 2.2 : Liste des algorithmes de tatouage 3D.

Algorithme	Format initial	Mode d'extraction	Modifie géométrie	Modifie topologie	Robuste remaillage	Commentaires
1	Maillage	Aveugle	Non	Non	Non	
2	Maillage	Aveugle	Oui	Non	Non	
3	Maillage	Aveugle	Oui	Non	Non	
4	Maillage	Aveugle	Oui	Non	Non	
5	Maillage	Aveugle	Oui	Non	Oui	
6	Maillage	Aveugle	Oui	Non	Oui	
7	Maillage	Aveugle	Oui	Non	Non	
8	Maillage	Non-aveugle	Oui	Non	Oui	
9	Maillage	Aveugle	Oui	Non	Non	
10	Maillage	Aveugle	Oui	Non	Non	
11	Maillage	Aveugle	Oui	Non	Non	
12	Maillage	Aveugle	Non	Oui	Non	
13	Maillage	Aveugle	Oui	Non	Non	
14	Maillage	Aveugle	Non	Oui	Oui	marque visuelle
15	Maillage	Aveugle	Non	Oui	Oui	marque visuelle
16	Maillage	Aveugle	Oui	Non	Non	
17	Maillage	Non-aveugle	Oui	Non	Non	
18	Maillage	Non-aveugle	Oui	Non	Oui	
19	Maillage	Non-aveugle	Oui	Non	Oui	
20	Maillage	Non-aveugle	Oui	Non	Oui	
21	Maillage	Non-aveugle	Oui	Non	Oui	
22	Maillage	Aveugle	Oui	Non	Oui	
23	NURBS	Aveugle	Oui	Non	-	
24	NURBS	Non-aveugle	Non	Non	-	
25	NURBS	Aveugle	Oui	Non	-	
26	CSG	Aveugle	Non	Oui	-	
27	-	-	Non	Non	-	
28	-	-	-	-	-	

Deuxième partie

Tatouage d'objets 3D basé sur la texture

Chapitre 3

Principe du tatouage d'objets 3D basé sur la texture

Nous décrivons dans ce chapitre un algorithme de tatouage d'objets 3D basé sur la texture dont le but est de protéger les représentations visuelles d'un objet 3D après que celui-ci ait été tatoué.

3.1 Vue d'ensemble de l'algorithme

Le principe de base de notre algorithme est illustré par la figure 3.1 : étant donné un objet 3D composé d'une description géométrique, d'une image de

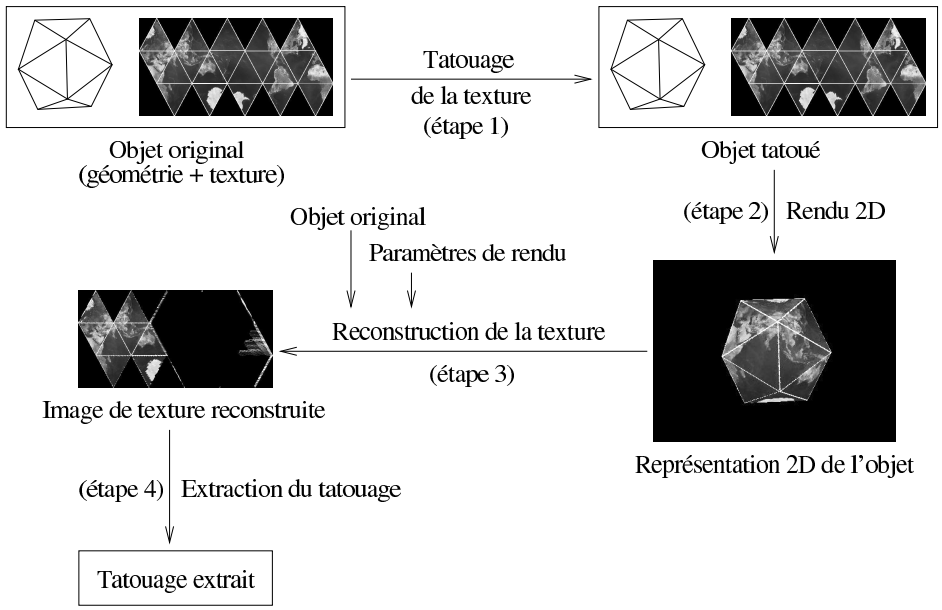


FIG. 3.1 : Principe du tatouage d'objets 3D basé sur la texture.

texture, et d'une fonction (non représentée mais implicite) de plaquage de la texture sur la géométrie, on enfouit une information dans l'objet en tatouant sa texture par un algorithme de tatouage d'images fixes (étape 1). L'objet tatoué peut alors être publié puis utilisé dans des images virtuelles (étape 2). Ensuite, on peut vérifier que l'objet représenté est protégé en reconstruisant l'image de texture tatouée (étape 3) puis en extrayant le tatouage de l'image de texture reconstruite (étape 4).

Une image d'un objet 3D n'en présente généralement qu'un seul coté si bien qu'on ne peut reconstruire à partir de cette image qu'une partie de la texture originale. Si on veut maximiser les chances d'extraire le tatouage de la texture reconstruite il peut donc être utile d'utiliser plusieurs vues différentes de l'objet 3D puis de fusionner les textures reconstruites à partir de chacune d'elles. Cela peut être possible en particulier dans le cas de séquences vidéo.

La reconstruction de l'image de texture à partir d'une vue 2D est en fait l'inversion du procédé de rendu 2D, c'est-à-dire l'inversion du plaquage de la texture et de la projection perspective. Pour réaliser cette inversion on a besoin de connaître d'une part la géométrie de l'objet original et la fonction de plaquage de texture, et d'autre part les paramètres de rendu tels que l'angle de vue et l'illumination synthétique utilisée. Nous reviendrons sur l'estimation des paramètres de rendu dans le chapitre 5.

3.2 Parallèle avec le tatouage d'images fixes

Notre schéma de tatouage d'objets 3D repose sur l'emploi d'un algorithme de tatouage d'images fixes. On peut a priori choisir parmi tous les algorithmes de tatouage d'images fixes existants ou bien en développer un qui soit spécialement conçu pour maximiser l'efficité du tatouage d'objets 3D.

Du point de vue du tatouage d'images fixes, le fait d'être employé dans un contexte de tatouage d'objets 3D impose que l'algorithme d'images fixes utilisé ait certaines propriétés. Pour mieux le comprendre nous détaillons ici certaines caractéristiques du tatouage d'objets 3D en comparaison avec le tatouage d'images fixes.

3.2.1 Mode d'extraction

Notre algorithme de tatouage d'objets 3D est non-aveugle car il a besoin de l'objet original pour reconstruire la texture préalablement à l'extraction du tatouage proprement dite. Plus précisément, si on suppose qu'on connaît les paramètres de rendu (projection perspective et illumination) on n'a besoin de connaître que la géométrie de l'objet original et la fonction de plaquage de texture pour pouvoir reconstruire la texture.

La connaissance de la texture originale n'est pas utile pour pouvoir reconstruire la texture tatouée, par contre elle peut être nécessaire si l'algorithme de tatouage d'images fixes utilisé est lui-même non-aveugle.

Enfin, comme nous le verrons dans le chapitre 5 nous avons de toute façon besoin de connaître l'image de texture originale pour pouvoir estimer les paramètres de rendu 2D (en particulier la projection perspective) au moyen d'une opération de recalage projectif entre l'objet 3D original et l'image 2D observée.

Dans la mesure où on a besoin de connaître l'image de texture originale dans une autre étape que l'extraction du tatouage d'images fixes, on peut également en faire profiter l'algorithme de tatouage d'images fixes et donc employer un algorithme de tatouage d'images fixes non-aveugle si cela présente un intérêt.

3.2.2 Visibilité du tatouage

En tatouage d'image fixes, l'impact visuel du tatouage sur l'image doit être aussi limité que possible. La visibilité est souvent exprimée numériquement en terme de rapport signal sur bruit entre l'image tatouée et l'image non tatouée. Dans le cas du tatouage d'objets 3D le problème est légèrement différent car l'image visualisée n'est pas l'image de texture qui a été tatouée mais une image déformée de celle-ci après plaquage sur l'objet 3D puis projection perspective.

Bien que la notion de visibilité du tatouage dans l'image de texture conserve sa validité, dans le cas du tatouage d'objets 3D ce qui nous intéresse avant tout c'est l'impact visuel du tatouage sur les vues 2D générées à partir de l'objet 3D texturé. En ce qui concerne le rapport signal sur bruit entre deux vues 2D identiques d'un même objet 3D mais utilisant respectivement l'object 3D tatoué et l'objet 3D non-tatoué, nous observons qu'il est pratiquement identique au rapport signal sur bruit entre l'image de texture tatouée et non tatouée.

Néanmoins, on peut imaginer que l'algorithme de tatouage d'images fixes utilisé soit adapté pour minimiser la visibilité du tatouage dans les vues 2D plutôt que dans l'image de texture. En effet, des parties différentes mais de tailles identiques de l'image de texture peuvent être plaquées sur des zones de tailles différentes de l'objet 3D. Il pourrait être intéressant d'adapter localement l'application du tatouage sur l'image de texture en fonction de l'amplitude de la déformation subie par l'image de texture lors du plaquage sur l'objet 3D.

3.2.3 Types d'attaques

Notre procédé de tatouage d'objets 3D peut subir un certain nombre d'attaques qui peuvent affecter l'extraction du tatouage de l'image de texture reconstruite. Nous répertorions ici l'ensemble de ces attaques et indiquons comment la plupart d'entre elles peuvent être perçues comme des attaques d'images fixes sur l'image de texture. Ces attaques peuvent intervenir lors des trois étapes suivantes (voir figure 3.2).

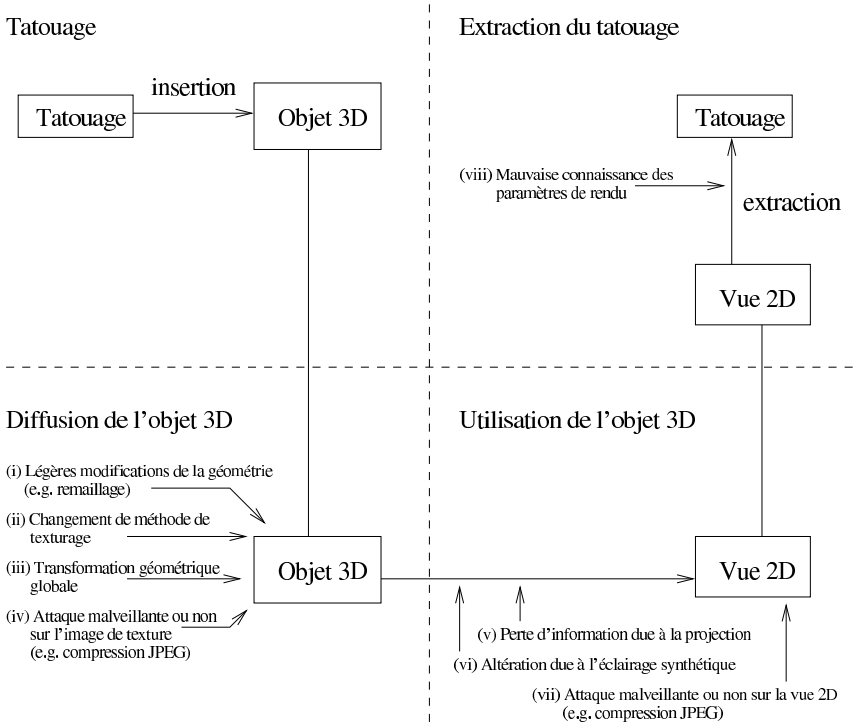


FIG. 3.2 : Inventaire des attaques pouvant affecter notre procédé de tatouage d'objets 3D.

Diffusion de l'objet

Après tatouage, l'objet 3D est distribué à des utilisateurs. A ce moment ceux-ci peuvent en modifier le format de fichier, ce qui peut entraîner de légères modifications de la géométrie (Fig. 3.2(i) et Fig. 3.2(iii)), ou de la manière de plaquer la texture (Fig. 3.2(ii)). Une caractéristique importante de notre algorithme est qu'il est en théorie insensible aux modifications de la représentation informatique tant que l'apparence de l'objet 3D reste la même. Si la géométrie et/ou la texture sont légèrement modifiés cela peut néanmoins introduire un bruit supplémentaire dans la texture reconstruite et réduire la fiabilité de l'extraction du tatouage. En dehors de l'attaque liée aux conversions de format, l'image de texture peut subir une attaque d'image fixe, par exemple une compression avec pertes (Fig. 3.2(iv)).

Utilisation de l'objet

Lors du rendu 2D de l'objet on perd de l'information (Fig. 3.2(v)). D'abord l'information de texture est tronquée de par le fait qu'une image de l'objet 3D n'en montre seulement qu'une partie. Ensuite, le plaquage de l'image de texture sur l'objet 3D est une opération qui introduit un bruit de rééchantillonnage. Enfin, la projection perspective peut diminuer la résolution de l'information

de texture visible lorsque l'objet est vu de loin ou avec un facteur d'agrandissement faible.

L'éclairage synthétique utilisé pour le rendu est également susceptible d'altérer l'information de texture (Fig. 3.2(vi)). En plus de ces "attaques" sur la texture qui sont en fait inhérentes à l'utilisation normale de l'objet 3D, il faut considérer les attaques d'images fixes que pourrait subir le rendu 2D (Fig. 3.2(vii)).

Reconstruction de la texture et extraction du tatouage

Dans cette dernière étape, l'information de texture qui était disponible au moment du tatouage de l'objet subit encore une altération du fait de l'inversion du procédé de rendu (Fig. 3.2(viii)). En effet les paramètres de rendu à inverser pour reconstruire l'image de texture à partir de la vue 2D peuvent n'être pas connus avec une grande exactitude. Dans ce cas cela peut entraîner des déformations géométriques locales dans l'image de texture reconstruite, par rapport à l'image de texture originale. Cela peut également entraîner une différence colorimétrique si l'éclairage ne peut pas être compensé. Enfin, que les paramètres de rendu soient connu avec exactitude ou non, la reconstruction de la texture depuis la vue 2D vers le référentiel de l'image de texture originale introduit toujours un bruit de rééchantillonnage.

Comparaison avec les attaques d'images fixes

Les attaques que nous venons de mentionner conduisent à des altérations de l'image de texture tatouée originale qui peuvent etre comparées à des attaques d'images fixes classiques. Ainsi, du point de vue de l'algorithme de tatouage d'images fixes utilisé, le procédé de tatouage 3D est une attaque spéciale de l'image de texture originale qui peut être vue comme une combinaison des attaques d'images fixes suivantes :

1. **Découpe.** Dans notre contexte, la reconstruction de la texture avant extraction du tatouage est généralement partielle, si bien que la texture subit une "découpe" (de forme a priori quelconque) entre l'insertion et l'extraction du tatouage.
2. **Attaque colorimétrique.** La texture reconstruite prend en compte l'éclairage synthétique utilisé pour le rendu, ce qui résulte en une différence de couleurs entre l'image de texture originale et l'image de texture recontruite.
3. **Bruit de rééchantillonnage.** Dans le cas des images fixes il s'agit du bruit introduit lorsqu'on rééchantillonne une image pour en modifier la taille. Dans le cas du procédé de tatouage 3D, un tel rééchantillonnage intervient également, mais de manière non-uniforme, lorsque la texture est plaquée sur l'objet 3D, lors de la projection 2D, et lors de l'inversion de la projection et du plaquage de texture. Par exemple, dans une vue

éloignée d'un objet 3D, la texture subit un sous-échantillonage, et lors de la reconstruction elle subit un sur-échantillonnage. Si la vue 2D est une vue très proche de l'objet, c'est le contraire. Globalement l'image de texture reconstruite est affectée par un bruit de rééchantillonnage par rapport à l'image de texture originale.

4. **Déformations géométriques locales.** De telles déformations entre l'image de texture reconstruite et l'image de texture originale peuvent apparaître si l'inversion de la projection perspective et du plaquage de la texture n'est pas exacte. Cela peut se produire si on ne parvient pas à connaître ou à calculer avec précision la projection perspective à inverser, ou si la géométrie ou la fonction de plaquage de texture (supposées connues) ont été modifiées légèrement avant le rendu 2D.

On peut voir que la combinaison de toutes ces attaques est une attaque qui peut être très forte et qui dans le cadre du tatouage d'images fixes pourrait être considérée comme une "attaque destructive". Néanmoins dans le contexte du tatouage 3D, cette attaque est une attaque non-malveillante qui est inhérente au procédé de tatouage 3D. Elle doit donc, dans ce contexte, être considérée comme une attaque non-destructive tant que l'objet 3D est représenté avec fidélité dans les images 2D. Ceci impose évidemment que l'algorithme de tatouage d'images fixes utilisé soit robuste à cette attaque composite, quand bien même une telle robustesse ne serait pas normalement recherchée s'il s'agissait uniquement de tatouage d'images fixes. La tâche pourrait être facilitée par le fait que cet algorithme de tatouage d'images fixes n'a pas besoin d'être aveugle.

Chapitre 4

Evaluation en environnement contrôlé

Notre algorithme de tatouage 3D implique de reconstruire l'image de texture d'un objet 3D tatoué à partir de la seule connaissance d'une vue 2D de cet objet. Pour cela il est nécessaire de connaître avec précision les paramètres de rendu utilisés afin de les inverser. Il s'agit en particulier de la projection perspective utilisée, c'est-à-dire la position et l'orientation de l'objet par rapport à la caméra virtuelle, et les paramètres intrinsèques de la caméra. Avant de traiter le problème de la connaissance de ces paramètres, qui sera l'objet du chapitre 5, nous allons voir dans quelle mesure il est possible d'inverser le procédé de rendu 2D et d'extraire le tatouage de l'image de texture reconstruite, dans l'hypothèse où les paramètres de rendu sont connus avec exactitude.

4.1 Modalités générales des expériences

4.1.1 Modèle de rendu 2D

Le modèle de rendu 2D comporte quatre éléments :

1. Le format de l'objet 3D et la manière de plaquer la texture sur la géométrie.
2. Le modèle de projection perspective.
3. Le modèle d'éclairage synthétique.
4. La manière d'appliquer les équations théoriques des trois modèles précédents à la nature discrète des images en informatique.

Le quatrième élément est particulièrement important lorsqu'il s'agit d'inverser tout le processus de rendu. En effet si on veut limiter les déformations géométriques locales de la texture reconstruite par rapport à la texture originale, on n'a pas le droit d'ignorer, par exemple, si les coordonnées d'un pixel d'une image représentent les coordonnées du coin supérieur gauche du pixel, ou les coordonnées du centre du pixel, ou autre. Il en va de même pour les formules

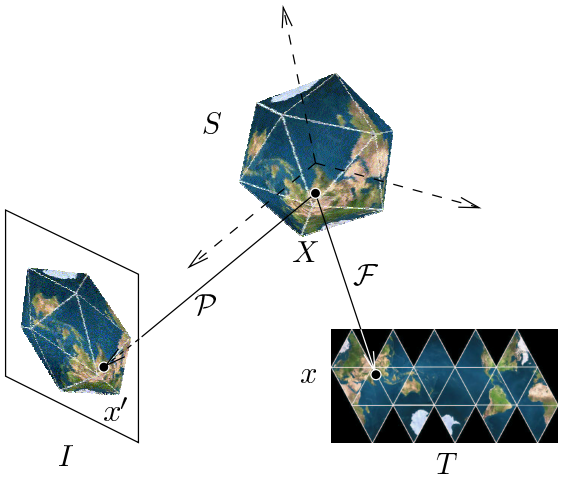


FIG. 4.1 : Modèle de rendu 2D : texturage et projection perspective.

d'interpolation utilisées lorsqu'on s'aperçoît que la couleur d'un pixel du rendu 2D doit être trouvée à une coordonnée non-entière de l'image de texture de l'objet représenté. Si pour nos simulations nous définissons nous-même ces détails et programmons nous-même le moteur 3D, le problème ne se pose pas. Si par contre nous utilisons un moteur 3D existant, par exemple OpenGL, il nous faut connaître toutes ces spécifications, et au besoin il faut procéder à un "calibrage" permettant de les déterminer.

La figure 4.1 résume l'ensemble des données du problème de rendu 2D, et de son inverse qui est la reconstruction de la texture ; on y a indiqué :

- $\mathcal{S}$, l'ensemble des points de la surface de l'objet 3D.
- I, une image représentant une vue 2D de l'objet 3D.
- T, l'image de texture de l'objet 3D.
- $\mathcal{F} : \mathcal{S} \rightarrow T$, la fonction de texturage de l'objet 3D, qui à chaque point de $\mathcal{S}$ associe sa couleur prise dans l'image T.
- $\mathcal{P}$, la projection entre le système de coordonnées de l'objet 3D et le système de coordonnées de l'image I.

4.1.2 Modèle de texturage

Le principe de texturage utilisé consiste simplement en la donnée d'une fonction $\mathcal{F} : \mathcal{S} \rightarrow T$. L'implantation pratique de cette fonction dépend du mode de représentation de la géométrie de l'objet 3D, par exemple maillage ou équation implicite, mais cela est transparent en ce qui concerne le texturage. Le seul détail d'implantation qui n'est a priori pas neutre est le type de l'interpolation réalisée pour calculer la couleur du pixel $x = \mathcal{F}(X)$ lorsque ses coordonnées sont non-entières. Comme nous le verrons, nous avons néanmoins constaté que le type d'interpolation utilisé avait peu d'influence sur les résultats de l'extraction du tatouage.

4.1.3 Modèle de projection

Le modèle de projection utilisé est le modèle de projection perspective classique où une projection perspective peut être représentée par une matrice de projection 3×4, définie à un facteur d'échelle près.

Une telle matrice de projection peut être factorisée sous la forme $P = A\begin{bmatrix} R & T \end{bmatrix}$ où

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

représente les paramètres intrinsèques de la caméra virtuelle et R et T représentent respectivement la matrice de rotation et le vecteur de translation du repère de l'objet par rapport au repère lié à la caméra.

Les paramètres α, β et γ représentent respectivement la "focale" suivant l'axe vertical et l'axe horizontal, et le facteur d'inclinaison de l'axe vertical. Les paramètres u_0 et v_0 sont les coordonnées, en pixels, de la "projection orthogonale" du "centre optique" de la caméra virtuelle dans l'image.

En réalité on utilise rarement en imagerie de synthèse des paramètres u_0 et v_0 qui ne soient pas les coordonnées du centre de l'image synthétisée, ou des paramètres α et β qui ne soient pas égaux (si leur unité est le pixel), et encore plus rarement un paramètre γ non-nul, si bien que souvent la matrice des paramètres intrinsèques peut se mettre sous la forme

$$A = \begin{bmatrix} f & 0 & \frac{W}{2} \\ 0 & f & \frac{H}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

où f est la focale en pixels, W et H les dimensions de l'image en pixels, et où l'origine des coordonnées image est située dans l'un des coins. En utilisant des coordonnées image normalisées, c'est-à-dire par exemple $[-1,1]\times[-1,1]$ comme dans OpenGL (dans d'autres systèmes on trouverait plutôt $[0,1] \times [0,1]$), la matrice A s'écrirait sous la forme

$$A = \begin{bmatrix} \frac{f}{k} & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

où f est la focale (désormais en unités image normalisées) et k est le "facteur d'aspect" $\frac{W}{H}$ entre la largeur et la hauteur de l'image.

Dans les expériences en milieu contrôlé où on suppose connue la projection à inverser, la question du choix de la modélisation des paramètres intrinsèques ne se pose pas : on se les donne, on les utilise pour créer des images expérimentales, et on les utilise à nouveau pour inverser le processus de rendu de ces images. D'un point de vue algébrique, l'inversion est exacte, quelle que soit la forme choisie pour les paramètres intrinsèques, c'est à dire la forme générale

ou la forme simplifiée. Ce sera lorsque que nous aurons à estimer la matrice de projection qu'il faudra choisir un modèle pour paramétrer cette matrice de projection et qu'on pourra déterminer les avantages et les inconvénient de chaque modèle par rapport au problème à résoudre.

4.1.4 Modèle d'éclairage

Dans les expériences en environnement contrôlé nous n'avons pas voulu altérer les couleurs de la texture. Nous n'avons donc pas utilisé d'éclairage synthétique à proprement parler. Nous avons simplement plaqué l'image de texture sur l'objet puis projeté l'objet dans l'image 2D en utilisant les couleurs de la texture telles qu'elles. Avec certains moteurs 3D cela revient à ne définir qu'une source de lumière ambiante de couleur blanche et d'intensité maximale. En pratique, comme nous n'avons pas besoin de gérer l'éclairage synthétique, nous n'avons pas utilisé un moteur 3D existant mais avons plutôt effectué le rendu (i.e. le texturage et la projection perspective) nous-même, ce qui permet d'en connaître parfaitement tous les paramètres et donc d'inverser le procédé de manière exacte pour reconstruire la texture.

La raison principale pour ne pas considérer l'impact de l'éclairage sur les résultats de l'algorithme de tatouage 3D, du moins dans un premier temps, est que la plupart des algorithmes de tatouage d'images fixes existants, et en particulier celui utilisé pour nos expériences, sont relativement robustes aux altérations colorimétriques. Nous reviendrons sur le modèle d'éclairage dans le chapitre consacré à l'estimation des paramètres de rendu.

4.1.5 Protocole et algorithme de tatouage d'images fixes

Notre schéma de tatouage d'objets 3D repose sur un algorithme de tatouage d'images fixes à choisir. Il est clair que le choix de cet algorithme a une influence sur les performances de la chaîne de tatouage 3D. Même des algorithmes qui ont des performances comparables dans le cadre du tatouage d'images fixes peuvent éventuellement conduire à des performance sensiblement différentes dans le contexte du tatouage d'objets 3D. La raison en est que l'application de tatouage d'objets 3D impose des contraintes spécifiques sur le tatouage d'images fixes, contraintes qui ne sont pas toujours considérées lors de la conception et l'évaluation des algorithmes de tatouage d'images fixes.

Comme nous l'avons vu en section 3.2.3.0 ces contraintes sont essentiellement la robustesse à la combinaison d'une altération colorimétrique, de bruit de rééchantillonnage, d'une découpe de forme irrégulière, et d'éventuelles déformations locales de faible amplitude. On doit donc en premier lieu choisir un algorithme de tatouage d'images fixes d'après ces contraintes. En fonction de l'application visée il faut également se préoccuper de certaines autres caractéristiques de l'algorithme de tatouage d'images fixes, comme par exemple le fait d'être avec ou sans capacité, et tout ce qui constitue les spécifications d'entrée/sortie de l'algorithme.

En ce qui nous concerne, et pour les besoins de l'expérimentation, nous avons utilisé l'algorithme de tatouage d'images fixes *Eurémark* développé à l'Institut Eurécom [30, 29]. Outre le fait que cet algorithme respecte les contraintes mentionnées ci-dessus, la principale raison de ce choix est que nous disposions du code source de son implantation, ce qui nous a permis de l'intégrer facilement dans notre chaîne algorithmique et d'automatiser les séries d'expériences où on fait varier un paramètre.

Nous décrivons brièvement le principe de cet algorithme en Annexe A, pour l'heure il nous suffit de savoir que cet algorithme permet d'enfouir le nombre voulu de bits dans une image puis de les extraire de l'image tatouée et éventuellement modifiée. La force de marquage (i.e. visibilité) est paramétrable mais un réglage par défaut, que nous avons utilisé, est d'avoir un PSNR de 38dB entre l'image et le bruit introduit par le tatouage. A l'extraction l'algorithme fournit le message binaire lu, lequel peut être plus ou moins erroné par rapport au message caché initialement si l'image a subi des attaques après tatouage. Un autre mode de fonctionnement permet d'obtenir en sortie des nombres variant de manière continue entre 0 et 1 (décodage "soft") ce qui permet un post-traitement différent d'un seuillage binaire strict (décodage "hard"), utile notamment si on utilise des codes correcteurs au dessus de l'algorithme de tatouage. Nous n'avons néanmoins pas utilisé de codes correcteurs et n'avons donc utilisé que le mode de décodage "hard" de l'algorithme de tatouage.

Enfin, l'algorithme utilisé effectue une extraction du tatouage en mode aveugle. Cela signifie qu'il n'a pas besoin de connaître l'image de texture originale pour l'extraction. Rappelons néanmoins qu'il n'est pas indispensable que l'algorithme de tatouage d'images fixes opère en mode aveugle dans la mesure où on a de toute façon besoin de connaître l'image de texture originale (non-tatouée) pour d'autres étapes du tatouage d'objets 3D, et qu'on pourrait donc aussi la supposer connue lors de la phase d'extraction de l'algorithme de tatouage d'images fixes sans que cette hypothèse de travail représente en réalité une contrainte supplémentaire.

4.1.6 Mesure des performances

La mesure des performances d'un algorithme de tatouage est toujours une question relativement complexe dans la mesure où il faut considérer les trois critères interdépendants de capacité, visibilité et robustesse. Et en tatouage d'objets 3D la définition de ces trois critères est de surcroît un peu plus complexe qu'en tatouage d'images fixes.

Pour simplifier le problème il est fréquent de fixer le paramètre de visibilité (c'est-à-dire la force de tatouage) et de voir l'évolution de la robustesse en fonction de la capacité (ou l'inverse). C'est ce que nous avons fait en fixant à 38dB la force de tatouage de l'algorithme de tatouage d'images fixes utilisé, avec la nuance que dans le cadre du tatouage d'objet 3D, la force de marquage de l'image de texture n'est a priori pas un indicateur immédiat de la visibilité de la marque dans les vues 2D générées à partir de l'objet 3D. Il se trouve néan-

moins que le calcul du PSNR entre l'image de texture tatouée et non tatouée ou entre les vue 2D produites à partir de l'objet 3D tatoué et non tatoué fournissent des valeurs pratiquement identiques. En dehors de ces considérations sur la pertinence de la force de marquage du tatouage d'images fixes en tant qu'indicateur de la visibilité de la marque dans les vues 2D générées à partir de l'objet 3D, ce qui importe le plus c'est qu'en fixant la force de marquage du tatouage d'images fixes on fixe un paramètre et qu'il est possible dans ces conditions d'évaluer la variation de la robustesse en fonction de la capacité.

L'algorithme Eurémark permet d'enfouir un message binaire et de le récupérer, éventuellement entaché d'erreurs. Pour une taille de message et pour des conditions expérimentales (en particulier des attaques) données on peut donc évaluer la robustesse du tatouage en comptabilisant le nombre de bits erronés dans le message recupéré.

En ce qui concerne la capacité, on peut généralement l'envisager soit comme la taille de la marque soit comme le ratio entre la taille de la marque et la taille du document original. Mais notre application est un peu particulière en ce sens que la taille du document original, en l'occurrence l'image de texture d'un objet 3D, n'a a priori pas de lien avec la quantité d'information de texture présente dans une vue 2D quelconque de l'objet 3D. En effet la projection 2D provoque une perte d'information (découpe et éventuellement sous-échantillonnage). Or c'est bien à partir de l'information récupérée dans la vue 2D qu'on tente d'extraire la marque et c'est donc bien cette information qu'il nous semble pertinent de quantifier. Au lieu de capacité au sens strict en termes de taille de message et/ou d'image de texture, nous parlerons donc de la quantité d'information présente dans la vue 2D. Il est néanmoins clair que cette quantité d'information dépend entièrement de l'utilisation particulière qui est faite de l'objet 3D, c'est-à-dire de la vue 2D considérée, alors que la notion de capacité habituelle en tatouage ne dépend que du document original.

Concrètement nous avons défini une mesure de la quantité d'information présente dans une vue 2D d'un objet 3D texturé comme étant le nombre de pixels distincts de l'image de texture originale qui sont visibles dans la vue 2D considérée. La figure 4.2 illustre le trajet suivi par l'information de texture originale jusqu'à la texture reconstruite et en passant par la projection perspective. La reconstruction est l'opération inverse du rendu 2D. Une perte d'information de texture a lieu lors du rendu 2D du fait que la surface de l'objet n'est que partiellement visible et du fait du rééchantillonnage de la texture. Si on laisse de côté la troncature de l'information de texture, il reste l'effet du rééchantillonnage. Celui-ci peut être quasiment nul si la première opération (le rendu 2D) consiste essentiellement en un sur-échantillonnage qui est suivi par un sous-échantillonnage lors de la reconstruction. Dans le cas contraire, un sous-échantillonnage suivi par un sous-échantillonnage, il y a perte d'information de texture. On peut résumer ceci en disant que c'est l'opération de rendu 2D qui représente le goulot d'étranglement, plus ou moins étroit, dans le trajet de l'information de texture.

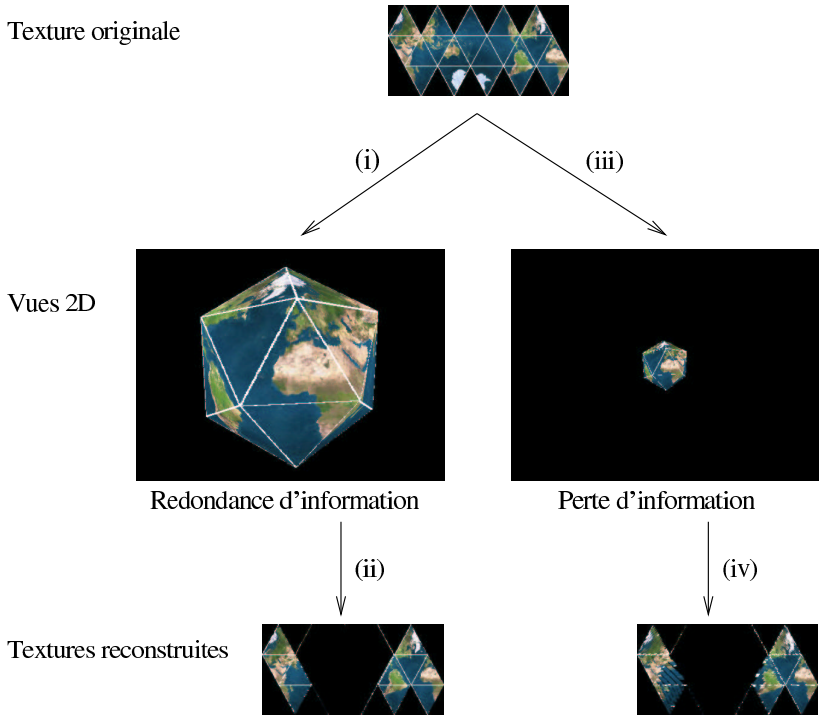


FIG. 4.2 : La quantité d'information de texture présente dans la vue 2D varie suivant que la vue est très agrandie ou non. Pour une vue très agrandie (i), on ne perd pas d'information, on peut même introduire une certaine redondance, sans toutefois créer de l'information supplémentaire. Lors de la reconstruction (ii) la redondance est éliminée mais a peu près toute l'information présente dans la texture originale est récupérée, à la troncature près (un seul côté de l'objet étant visible). Par contre si la vue implique un sous-échantillonnage de la texture (iii) il y a une perte irréversible d'information, et la reconstruction (iv), qui en l'occurrence implique un sur-échantillonnage, ne permet d'obtenir que l'information dégradée sans récupérer l'information perdue.

En fin de compte nous avons quantifié l'information réellement disponible dans la vue 2D de la manière suivante :

1. Dans l'image de texture originale, on assigne à chaque pixel une valeur unique ayant valeur d'identifiant du pixel et non plus de couleur.
2. On procède au rendu 2D avec les paramètres de rendu voulus mais en utilisant la texture modifiée comme ci-dessus. On n'utilise pas d'éclairage (la texture modifiée n'est plus une carte de couleur) et on utilise une "interpolation" au plus proche voisin : les identifiants des pixels ne sont en effet pas des valeurs modifiables, il s'agit de les faire passer de l'image de texture à la vue 2D pour comptabiliser le nombre d'identifiants distincts.
3. On effectue la reconstruction de la texture à partir de la vue 2D formée des identifiants des pixels de texture. On obtient ainsi une texture reconstruite formée d'identifiants de pixels de la texture originale. Comme pour l'opération précédente la question de l'interpolation des identifiants

n'a pas de sens : on effectue un rééchantillonnage au plus proche voisin.

4. Enfin on compte le nombre d'identifiants de pixels distincts dans la texture reconstruite.

Remarquons que si la projection perspective produit un sous-échantillonnage certains identifiants présents dans l'image de texture originale n'apparaitront pas dans la vue 2D, mais ceux qui auront été conservés apparaitront plusieurs fois dans l'image de texture reconstruite car la reconstruction implique dans ce cas un sur-échantillonnage. Néanmoins chacun de ces identifiants ne comptera qu'une fois même s'il apparaît plusieurs fois. Toujours dans le cas où le rendu 2D tend à être un sous-échantillonnage, on peut voir que le nombre de pixels visibles distincts est approximativement égal à l'aire, en pixels, de la projection de l'objet 3D. Par contre s'il s'agit d'un sur-échantillonnage certains pixels de la texture originale seront vus agrandis et apparaitrons plusieurs fois dans la vue 2D.

On pourrait aussi remarquer que dans la mesure où la perte d'information a lieu lors de la projection et pas lors de la reconstruction de la texture, il suffirait de compter le nombre de pixels distincts visibles dans la vue et pas dans l'image de texture reconstruite, car le résultat devrait être à peu près le même. L'expérience confirme que les résultats sont très proches. Bien que les résultats ne soient pas sensiblement différents, nous avons choisi de compter les pixels distincts après reconstruction de la texture. Ce décompte correspondant à la "quantité d'information disponible à l'extraction pour l'algorithme de tatouage d'images fixes". Le décompte des pixels de texture distincts dans la vue 2D concerne quant à lui "l'information disponible à l'extraction pour l'algorithme de tatouage 3D".

4.2 Mise en oeuvre de la reconstruction de la texture

4.2.1 Principe

Revenant aux notations de la figure 4.1, le problème qu'on se pose est de recréer l'image T à partir de l'image I. Pour le résoudre on suppose connus l'image I, la géométrie $\mathcal{S}$, la fonction de texturage $\mathcal{F}$ et la projection $\mathcal{P}$. L'algorithme de reconstruction de T est alors le suivant :

1. considérer chaque pixel x de T,
2. calculer $X = \mathcal{F}^{-1}(x)$ s'il existe,
3. calculer le pixel $x' = \mathcal{P}(X)$, projection de X dans l'image I,
4. vérifier que x' est réellement visible dans l'image I,
5. définir la couleur de x comme étant celle de x'.

L'étape 4 est nécessaire car si le point X se trouve sur la face cachée de l'objet 3D, alors il n'est pas visible dans l'image I, et bien que mathématiquement on puisse calculer sa projection x', le pixel x' n'aurait pas la couleur du

point X mais celle du point de la face avant de l'objet qui se projette aussi en x'. Dans ce cas le pixel de texture reconstruite x ne peut pas être renseigné.

4.2.2 Utilisation d'un Z-buffer

Pour décider si un pixel x' associé à un point X donné est visible ou non, on remplit d'abord un Z-buffer [92] associé à l'image I, c'est à dire qu'on calcule pour chaque pixel de I la distance du point de l'objet 3D le plus proche qui s'y projette. Pour cela on simule simplement un procédé de rendu 2D par Z-buffer en n'effectuant pas le rendu mais en calculant seulement le contenu du Z-buffer. Une fois que le Z-buffer est rempli, il suffit de comparer la profondeur d'un point X avec la valeur du Z-buffer en x' pour savoir si le point X est visible ou non : si les profondeurs coincident alors le point est visible, si le point X est plus éloigné que la valeur de profondeur de x' alors c'est un autre point que X qui est visible en x', quant au cas où X serait plus proche que la profondeur indiquée en x' il ne peut logiquement pas se présenter.

L'inconvénient de cette méthode est que les coordonnées des pixels du Z-buffer (les mêmes que ceux de l'image I) sont entières, or les coordonnées de la projection x' d'un point X peuvent n'être pas entières. Il faut donc d'une part savoir si on interpole la valeur du Z-buffer au point x', et si oui comment, ou si on arrondit x' aux plus proches coordonnées entières, et d'autre part savoir quelle tolérance fixer pour déclarer que deux profondeurs sont égales car quels que soient les choix faits on est amené à comparer les profondeurs de deux points certes très proches mais pas confondus.

Pour nos expériences nous avons utilisé la méthode ad hoc d'arrondir les coordonnées de x' aux coordonnées entières les plus proches et de déclarer que la profondeur de X est égale à celle de x' si la différence n'est pas supérieure à 1%. Le critère d'"égalité" est discutable en ce qu'il n'est certainement pas généralisable : il dépend en particulier du diamètre relatif de l'objet par rapport à sa distance de l'observateur. De plus, même en utilisant un critère adapté à certaines données, il y a toujours des cas (marginaux) où il peut conduire à une décision erronée, par exemple lorsque la ligne de vue est tangente à la surface de l'objet (c'est à dire aussi la plupart du temps lorsque x' se trouve à la frontière de la zone de texture reconstructible).

Néanmoins, cette méthode, qui doit être appliquée des milliers voire des millions de fois, permet d'accélérer l'exécution de nos expériences, et seuls quelques pixels de texture ne sont pas correctement reconstruits.

Si on voulait généraliser l'algorithme de reconstruction de texture on pourrait utiliser un critère dépendant des données ou choisir de ne pas arrondir les coordonnées de x'. Si on n'arrondit pas les coordonnées de x' il faut calculer la valeur du Z-buffer interpolée en x', soit calculer la valeur exacte du Z-buffer en x'. Quitte à abandonner la solution approximative que nous avions adoptée, autant calculer la valeur exacte du Z-buffer en x' plutôt que de calculer une valeur interpolée suivant une formule elle aussi approximative et choisie de manière arbitraire. Pour calculer la valeur exacte du Z-buffer en x' il suffit

de calculer les intersections de la ligne de vue passant par x' avec l'objet et de garder la profondeur de l'intersection la plus proche. Ce calcul est bien plus lourd que le remplissage d'un Z-buffer aux coordonnées entières, mais pour une application réelle où contrairement aux expériences de laboratoire il n'y a pas besoin d'effectuer de longues séries d'extraction du tatouage, le coût du calcul de la profondeur réelle serait insignifiant.

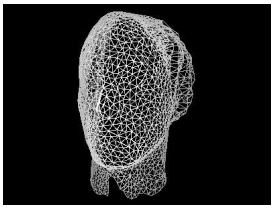
4.2.3 Rééchantillonnage de la texture

Il se pose pour le calcul de la couleur d'un pixel x' le même problème que pour le calcul de sa profondeur : les coordonnées de x' sont en général non-entières. Mais contrairement à la profondeur, qui peut être calculée de manière exacte aux coordonnées non-entières d'après la géométrie de l'objet, la couleur des pixels aux coordonnées entières (i.e. l'image I) est une donnée qui ne connaît pas de prolongement idéal aux coordonnées non-entières. Bien sûr on peut toujours interpoler la couleur aux coordonnées non-entières, mais une telle interpolation, qui dépend de la méthode d'interpolation choisie, a un caractère arbitraire.

Parmis les méthodes d'interpolation possibles on peut citer en particulier l'interpolation au plus proche voisin, l'interpolation bilinéaire ou l'interpolation B-Spline cubique. Ces méthodes, ainsi que d'autres, font l'objet d'une évaluation rigoureuse dans [87], et il en ressort que l'interpolation à base de B-Spline cubique est une méthode qui présente beaucoup d'avantages en termes de simplicité et de distorsion. Nous avons essayé les trois méthodes mentionnées pour l'interpolation de la couleur en x' et nous n'avons observé aucune différence dans la robustesse de l'extraction du tatouage en fonction de la méthode d'interpolation utilisée pour la reconstruction de la texture. Pour cette raison nous avons toujours utilisé la méthode d'interpolation au plus proche voisin, la plus simple. Nous pensons que la méthode d'interpolation, dans la mesure où elle n'est utilisée qu'une fois, introduit peu de bruit par rapport à l'information reconstruite. Par contre s'il y avait plusieurs rééchantillonnages successifs, l'information pourrait être rapidement dégradée par une interpolation au plus proche voisin et au contraire très bien conservée par une interpolation à base de B-Spline cubique comme indiqué dans [87]. C'est la raison pour laquelle dans les quelques expériences où il y a lieu de rééchantillonner plusieurs fois la texture (cf. section 4.7), et seulement dans ce cas, nous avons utilisé l'interpolation B-Spline cubique pour les rééchantillonnages distincts de la reconstruction de texture.

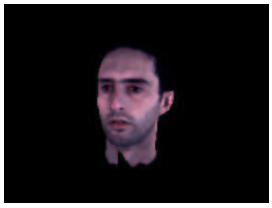
4.3 Expérience de référence

La première expérience que nous avons menée consiste à tatouer la texture d'un objet 3D, à construire une vue 2D de l'objet tatoué, à reconstruire la texture tatouée à partir de cette vue et à partir de la connaissance exacte des

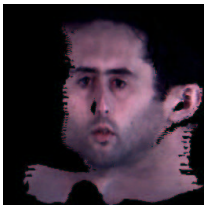


(a) Géométrie.

(b) Texture.



(c) Une vue 2D.



(d) Texture reconstruite.

FIG. 4.3 : Modèle 3D de visage.

paramètres de rendu, et enfin à extraire le tatouage de la texture reconstruite. Cette expérience a pour but de servir de point de comparaison pour toutes les expériences suivantes ainsi que de fournir une première évaluation des performances limites qu'on peut espérer atteindre par notre algorithme. En effet, dans la mesure où on reconstruit directement la texture à partir de la vue 2D, sans qu'aucune attaque ne s'intercale, et avec la connaissance parfaite des paramètres de rendu, on se place dans les conditions les meilleures possibles et les performances mesurées dans ces conditions sont réellement une borne supérieure de ce qu'on pourra obtenir dans des conditions plus réalistes.

Pour cette expérience on a utilisé deux objets 3D différents et on a fait varier les paramètres suivants :

1. Facteur d'agrandissement de la vue 2D.
2. Résolution de la texture.
3. Taille de la marque.

4.3.1 Premier objet

La figure 4.3 montre l'un des deux objets utilisés : il s'agit d'un modèle 3D de visage représenté par un maillage triangulaire (a) et une image de texture (b) dont la résolution initiale est de 512×512 pixels. On indique également le point de vue de cet objet utilisé dans notre expérience (c) et l'allure de l'image de texture reconstruite à partir de cette vue (d). La figure 4.4 illustre la variation du facteur d'agrandissement d'une vue 2D de l'objet sous un point de vue constant, et la figure 4.5 illustre la variation de la résolution de la texture de l'objet 3D.

Dans une première expérience on a utilisé le point de vue de la figure 4.3(c) et enfoui une marque de 64 bits dans l'image de texture. On a fait varier de

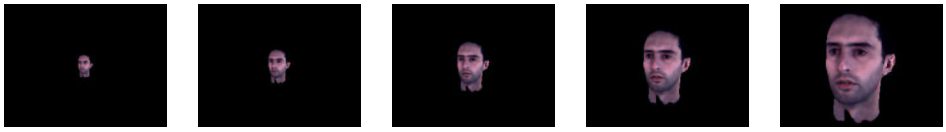


FIG. 4.4 : Variation du facteur d'agrandissement d'une vue 2D.

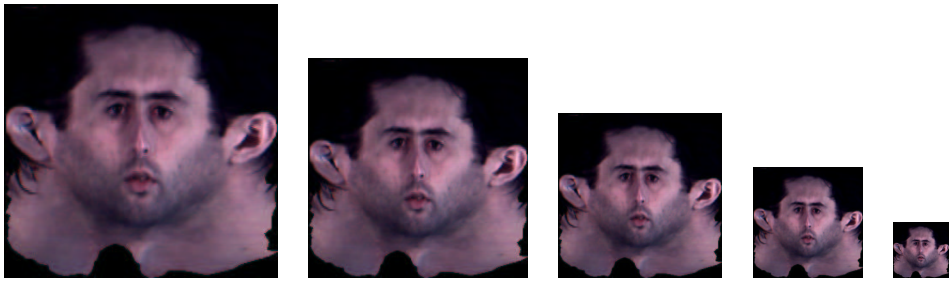


FIG. 4.5 : Variation de la résolution de la texture.

deux façons différentes le nombre de pixels visibles dans la vue 2D : soit avec une résolution de texture maximale mais en changeant le facteur d'échelle de la vue (cf. figure 4.4), soit avec un facteur d'échelle maximal mais en changeant la résolution de l'image de texture (cf. figure 4.5). Après reconstruction de la texture à partir de la vue 2D et extraction du tatouage on a compté le nombre de bits erronés. La figure 4.6 présente les résultats de cette expérience dans le cas de la variation du facteur d'échelle de la vue (a) et de variation de la résolution de la texture (b).

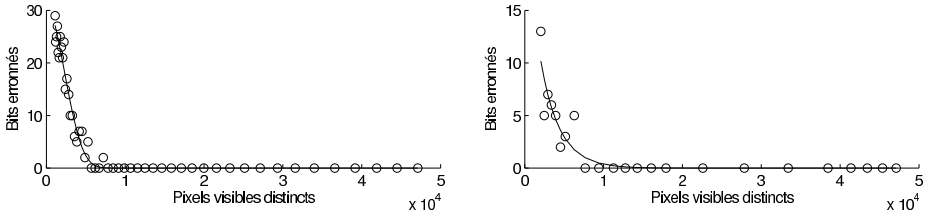
On constate que dans les deux cas le seuil de récupération correcte de la marque de 64 bits se situe un peu en dessous de 10000 pixels de texture visibles (soit l'aire d'un carré de 100×100 pixels). Ceci justifie a posteriori, et sous réserve des résultats d'expériences ultérieures, que ce qui compte pour extraire correctement la marque ce n'est pas la résolution de la texture ou le facteur d'échelle de la vue en tant que tels, mais bien le nombre de pixels de texture visibles dans la vue 2D, lequel, évidemment, dépend à la fois du facteur d'échelle de la vue et de la résolution de la texture (et des autres paramètres de rendu, qui ici sont fixés).

Un point de détail concerne le modèle de courbe que nous avons ajusté aux données dans les graphes de la figure 4.6. Cette courbe n'est là qu'à titre indicatif et le modèle utilisé n'a à ce jour pas de justification rigoureuse. Nous avons utilisé une courbe de la forme :

$$\frac{y}{64} = \frac{1}{2} erfc(\alpha x^{\beta}) \tag{4.4}$$

La fonction $erfc$ est la fonction d'erreur complémentaire définie par :

$$erfc(x) = 1 - \int_0^x \frac{2}{\sqrt{\pi}} e^{-t^2} dt \tag{4.5}$$



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.6 : Expérience de référence, premier objet. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 64 bits et pour la vue de la figure 4.3(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture.

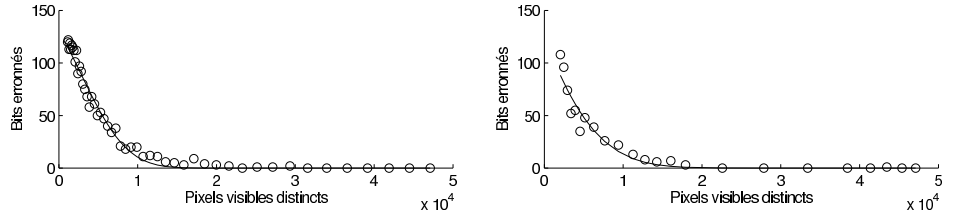
y représente le nombre de bits erronés dans le message récupéré et x le nombre de pixels de texture visibles dans la vue 2D. Quant à α et β ce sont les paramètres du modèle de courbe, lesquels sont ajustés en minimisant l'erreur quadratique moyenne entre la courbe et les données. Le choix de ce modèle est inspiré par l'analogie avec le traitement du signal. Dans le cas de base d'un canal de communication bruité, le taux d'erreur-binaire p est lié au rapport signal-sur-bruit SNR du canal par la formule :

$$p = \frac{1}{2} erfc(SNR^{\frac{1}{2}}) \tag{4.6}$$

Dans notre cas le taux d'erreur binaire est $\frac{y}{64}$, quant au rapport signal-sur-bruit nous avons voulu l'identifier avec la "quantité d'information concernant le tatouage" présente dans la vue 2D, laquelle est liée au nombre de pixels de texture visibles distincts x. De fait, ce modèle, qui ne comporte que deux paramètres, colle relativement bien aux données même, si elles sont parfois un peu éparpillées. De plus il prédit un taux d'erreur-binaire de $\frac{1}{2}$ lorsque le nombre de pixels de texture visibles est nul, ce qui est conforme au bon sens, car en l'absence d'information concernant la vue 2D tatouée, il est logique que les bits du tatouage extrait soient déterminés complètement au hasard, c'est-à-dire avec une probabilité d'erreur de $\frac{1}{2}$.

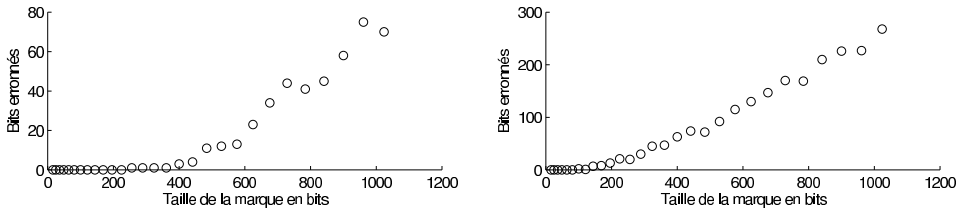
On a effectué la même expérience avec une marque de 256 bits au lieu de 64 (figure 4.7). Le modèle de courbe ajusté aux données est le même que précédemment, mais l'expression du taux d'erreur-binaire devient $\frac{y}{256}$. Là encore les résultats sont répartis de manière similaire, qu'on fasse varier le facteur d'échelle de la projection perpective (figure 4.7(a)) ou la résolution de la texture (figure 4.7(b)). Le seuil de récupération sans erreur de la marque se situe vers 20000 pixels visibles même si dans certains cas quelques erreurs sont possibles jusqu'à presque 30000 pixels visibles.

Pour clore cette première évaluation des capacités du tatouage basé sur la texture, on a effectué une dernière expérience où le modèle 3D et la vue 2D sont fixés et où seule change la taille du tatouage. La figure 4.8(a) montre les résultats obtenus en utilisant l'objet et la vue présentés dans la figure 4.3 et



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.7 : Expérience de référence, premier objet. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits et pour la vue de la figure 4.3(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture.



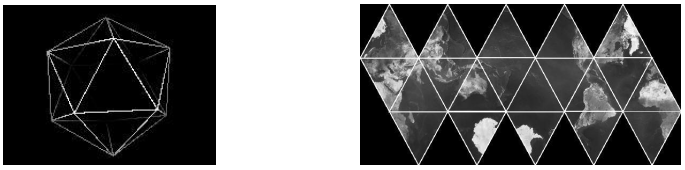
(a) 30881 pixels de texture visibles. (b) 7834 pixels de texture visibles.

FIG. 4.8 : Expérience de référence, premier objet. Robustesse de l'extraction en fonction de la taille de la marque enfouie, pour la vue de la figure 4.3(c) et lorsque le nombre de pixels de texture visibles est fixé en choisissant un facteur d'agrandissement de la vue.

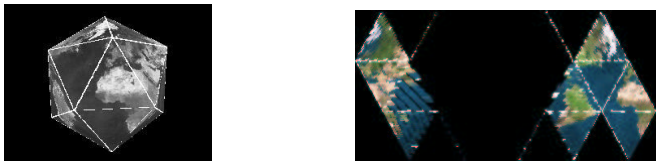
en utilisant un facteur d'échelle tel que 30881 pixels de texture distincts soient visibles. La figure 4.8(b) a été obtenue de la même manière mais en utilisant une vue plus réduite, où 7834 pixels de texture distincts sont visibles.

4.3.2 Deuxième objet

Pour vérifier la validité des résultats précédents nous avons effectué les mêmes expériences mais en utilisant un objet 3D différent : un icosaèdre régulier sur lequel est plaquée une carte du globe terrestre (figure 4.9). En comparant les résultats des figures 4.12, 4.13 et 4.14 avec ceux des figures 4.6, 4.7 et 4.8 respectivement, on s'aperçoît que les résultats sont beaucoup moins bons avec le second objet 3D. Pour expliquer cette forte baisse de performances on peut avancer l'hypothèse que les performances et la robustesse de l'algorithme de tatouage d'images fixes dépendent fortement de la géométrie de l'image de texture et/ou de la géométrie de l'information manquante après reconstruction. Premièrement, l'image de texture du premier objet est carrée alors que celle du second objet est un rectangle long et surtout toute la surface de ce rectangle n'est pas utilisée ce qui donne une forme "irrégulière" (en l'occurrence une réunion de triangles) au support de l'information utile. En outre la partie de texture reconstruite est plus "compacte" ou "regroupée" avec la vue figure



(a) Géométrie. (b) Texture.



(c) Une vue 2D. (d) Texture reconstruite.

FIG. 4.9 : Terre en forme d'icosaèdre.

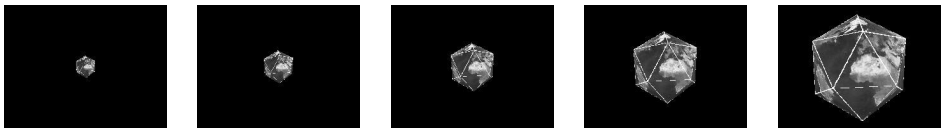
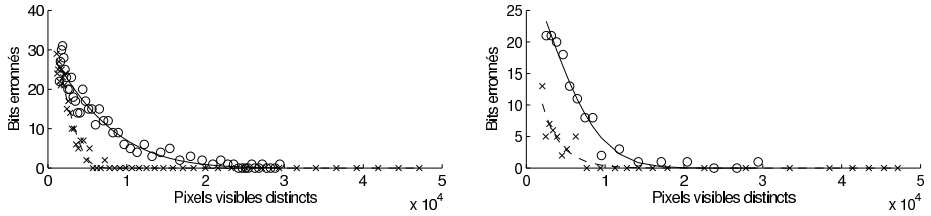


FIG. 4.10 : Variation du facteur d'agrandissement de la vue 2D.

4.3(c) qu'avec la vue de la figure 4.9(c), en effet la texture reconstruite en figure 4.9(d) présente une grande quantité d'information manquante en plein milieu de l'image alors que dans la figure 4.3(d) l'information manquante est plutôt sur le bord. En l'absence d'étude plus poussée on peut imaginer que l'algorithme de tatouage d'images fixes sous-jacent, qui n'est a priori pas conçu pour être appliqué dans le cadre du tatouage d'objets 3D ni pour être robuste à des pertes d'information relativement importantes et de forme quelconque, soit relativement sensible à la forme de la zone de texture reconstruite. Lors d'expériences ultérieures nous verrons si le fait de modifier l'image de texture et la fonction de plaquage de texture permet d'améliorer la susceptibilité de l'objet 3D à être tatoué de manière robuste (voir section 4.7).

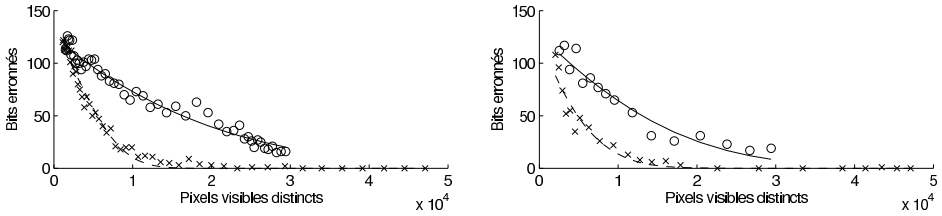
Un dernier élément de réflexion est apporté par la figure 4.15 où nous avons encore utilisé l'objet polyédral et le même point de vue, mais où nous avons remplacé la carte de la terre par l'image de texture du premier objet. On observe que les résultats restent moins bon que pour l'expérience sur le premier objet, mais qu'ils sont néanmoins bien meilleurs que lorsqu'on utilise l'image de texture de la terre. Ceci indique que c'est bien le tatouage de l'image de

FIG. 4.11 : Variation de la résolution de la texture.



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

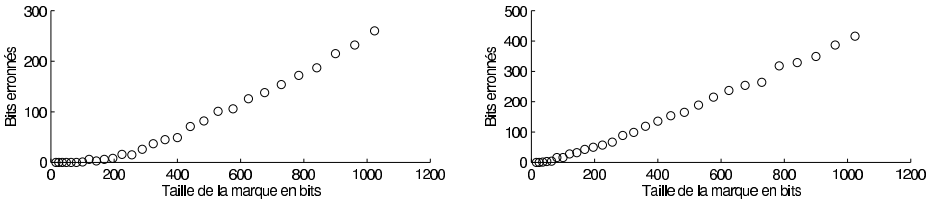
FIG. 4.12 : Expérience de référence, deuxième objet. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 64 bits et pour la vue de la figure 4.9(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.6 pour comparaison.



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

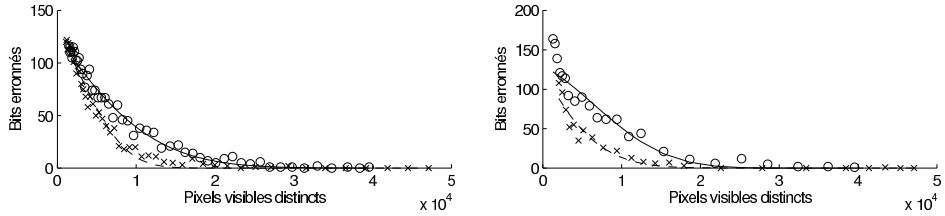
FIG. 4.13 : Expérience de référence, deuxième objet. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits et pour la vue de la figure 4.9(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.7 pour comparaison.

texture de la terre qui est la cause principale de la baisse des performances par rapport aux résultats obtenus avec le premier objet.



(a) 26120 pixels de texture visibles. (b) 10474 pixels de texture visibles.

FIG. 4.14 : Expérience de référence, deuxième objet. Robustesse de l'extraction en fonction de la taille de la marque enfouie, pour la vue de la figure 4.9(c) et lorsque le nombre de pixels de texture visibles est fixé en choisissant un facteur d'agrandissement de la vue.



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.15 : Expérience de référence, géométrie du deuxième objet et texture du premier objet. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits et pour la vue de la figure 4.9(c) (même angle de vue mais texture du premier objet) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.7 pour comparaison.

4.4 Impact d'une compression JPEG

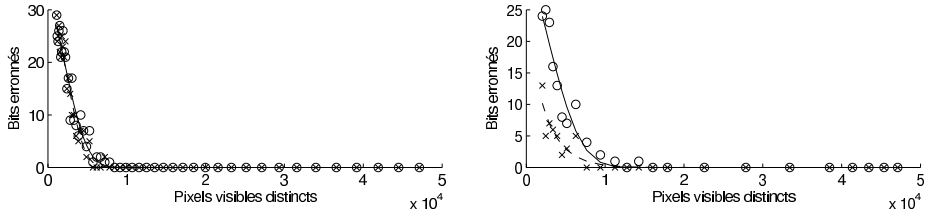
Une propriété désirable d'un algorithme de tatouage est d'être robuste à des manipulations normales et non-malveillantes qui ne modifient pas la sémantique du document. Un changement de format d'image et en particulier une compression JPEG est un exemple d'une telle manipulation. Les deux expériences suivantes examinent l'impact d'une compression JPEG de l'image de texture ou de la vue 2D de l'objet sur la robustesse de l'extraction.

4.4.1 Compression de la texture

Les résultats des figures 4.16 et 4.17 sont à comparer avec ceux des figures 4.6 et 4.7 respectivement. Il ressort de ces résultats que la compression de l'image de texture après tatouage a assez peu d'impact lorsque cette image est assez grande (cf. figures 4.16(a) et 4.17(a)). Par contre lorsque l'image de texture est petite (cf. figures 4.16(b) et 4.17(b)) la compression JPEG affecte beaucoup la robustesse. Ceci peut s'expliquer intuitivement par l'observation que dans une grande image on peut cacher de l'information avec beaucoup de redondance ce qui se traduit par une meilleure robustesse aux pertes dues à la compression JPEG, alors que dans une petite image, la même quantité d'information est enfouie au plus juste, sans redondance, et les pertes dues à la compression JPEG affectent directement l'information cachée.

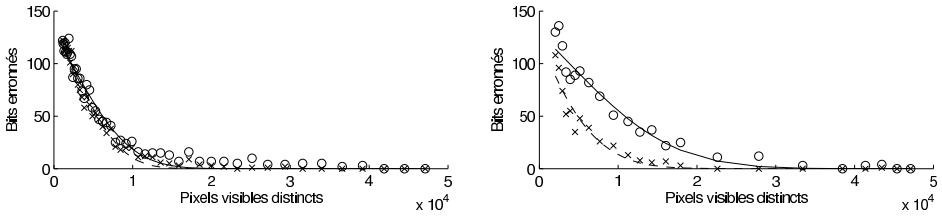
4.4.2 Compression de la vue 2D

Plutôt que de toucher à l'image de texture tatouée, on considère cette fois que c'est la vue 2D utilisée pour l'extraction du tatouage qui a subi une compression JPEG de qualité 75%. On peut s'attendre à ce que la robustesse de l'extraction soit d'autant plus affectée par la compression JPEG que l'aire de la projection 2D de l'objet est petite en raison du même principe que pour



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.16 : Impact d'une compression JPEG de la texture. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 64 bits, pour la vue de la figure 4.3(c), et avec une compression JPEG à 75% de qualité de l'image de texture, dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.6 pour comparaison.



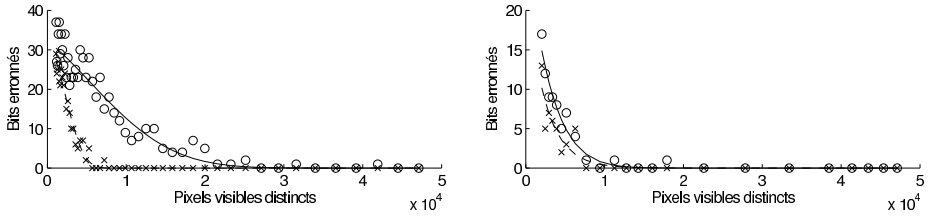
(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.17 : Impact d'une compression JPEG de la texture. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits, pour la vue de la figure 4.3(c), et avec une compression JPEG à 75% de qualité de l'image de texture, dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.7 pour comparaison.

l'expérience précédente : là où il y a redondance d'information (i.e. surface visible importante) pour cacher la marque, les pertes dues à la compression JPEG peuvent être rattrappées dans une certaine mesure. Ceci est corroboré par les résultats des figures 4.18(a) et 4.19(a). Les figures 4.18(b) et 4.19(b) quant à elles indiquent la robustesse du tatouage après compression JPEG de la vue 2D mais en utilisant une vue 2D très agrandie de l'objet ; on observe que la compression JPEG de la vue 2D affecte un peu la robustesse de l'extraction lorsque la résolution de l'image de texture utilisée est faible, surtout à haute capacité (i.e. marque de 256 bits).

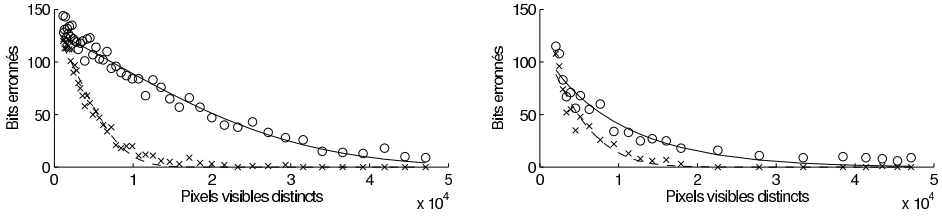
4.5 Impact d'une mauvaise connaissance de la projection

Si la projection utilisée pour reconstruire la texture n'est pas exactement la même que celle qui a servi à produire la vue 2D, alors la texture reconstruite est



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.18 : Impact d'une compression JPEG de la vue 2D. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 64 bits, pour la vue de la figure 4.3(c), et avec une compression JPEG à 75% de qualité de la vue 2D, dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.6 pour comparaison.

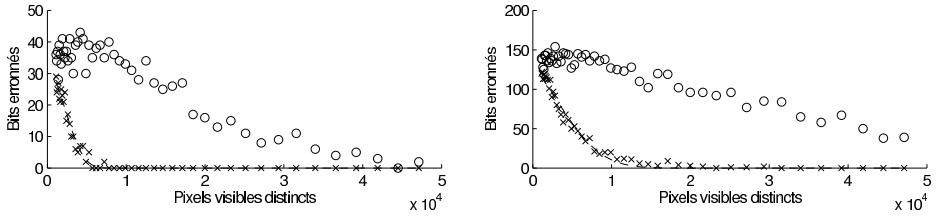


(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.19 : Impact d'une compression JPEG de la vue 2D. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits, pour la vue de la figure 4.3(c), et avec une compression JPEG à 75% de qualité de la vue 2D, dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.7 pour comparaison.

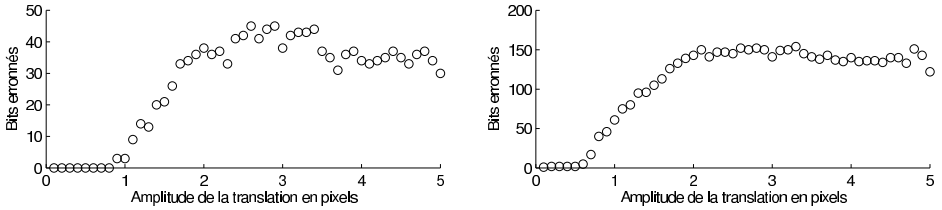
une version déformée de la texture originale, et l'extraction du tatouage en est affectée. Cette expérience vise à mesurer l'effet d'une mauvaise connaissance de la projection sur l'extraction du tatouage. Pour cela nous utilisons pour la reconstruction de la texture une projection légèrement différente de celle utilisée pour le rendu 2D. On utilise en fait la même projection que celle utilisée pour le rendu 2D mais simplement décalée dans le plan image d'une certaine quantité vers la droite. C'est-à-dire que les deux projections perspectives (celle du rendu et celle de la reconstruction) donnent des images du même objet qui sont simplement translatées l'une de l'autre.

Les figures 4.20(a) et 4.20(b) montrent la robustesse à un décalage de 1 pixel pour des marques de 64 et 256 bits. Elles sont à comparer avec les figures 4.6(a) et 4.7(a) respectivement. La figure 4.21 donne une estimation plus fine de la précision requise dans la connaissance de la projection, en faisant varier le décalage de 0 à 5 pixels. On constate que dans le meilleur des cas (message de 64 bits, et vue 2D très agrandie) les performances décrochent peu avant un décalage de 1 pixel. Ceci pourrait éventuellement être amélioré si l'algorithme de tatouage d'images fixes pouvait compenser certaines déformations locales



(a) Message de 64 bits. (b) Message de 256 bits.

FIG. 4.20 : Impact d'une mauvaise connaissance de la projection. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles (variation du facteur d'échelle de la vue). Vue de la figure 4.3(c). Erreur dans la projection utilisée pour la reconstruction : décalage de 1 pixel sur la droite. Les croix reprennent les figures 4.6(a) et 4.7(a) pour comparaison.



(a) Message de 64 bits. (b) Message de 256 bits.

FIG. 4.21 : Impact d'une mauvaise connaissance de la projection. Robustesse de l'extraction en fonction de l'amplitude de l'erreur commise sur la projection utilisée pour la reconstruction. Vue de la figure 4.3(c). Le facteur d'échelle est fixé : 30881 pixels de texture distincts sont visibles. La projection utilisée pour la reconstruction est décalée vers la droite.

dues à la mauvaise connaissance de la projection.

4.6 Impact d'une simplification de maillage

Une des propriétés de notre algorithme est qu'en principe il n'est pas affecté par un changement de format de l'objet 3D ultérieur au tatouage, du moment que l'apparence visuelle de l'objet est conservée. Pour vérifier cette propriété nous avons procédé à une simplification de maillage de l'objet 3D tatoué et tenté d'extraire la marque à partir d'une vue 2D. Plus précisément, on tatoue l'objet 3D de la figure 4.3, dont la géométrie est décrite par un maillage de près de 14000 triangles, puis on effectue une simplification de maillage [1] de l'objet 3D et on génère une vue 2D à partir de l'objet 3D tatoué simplifié. Enfin, on reconstruit la texture originale à partir de la vue 2D mais non pas en utilisant la connaissance de la géométrie 3D simplifiée, mais la connaissance de la géométrie 3D originale. On utilise donc pour la reconstruction de la texture

[1] Les opérations de simplification de maillage ont été réalisées à l'aide du logiciel QSlim [40].

un modèle 3D différent de celui effectivement utilisé pour le rendu.

Si on utilise la connaissance de la géométrie originale plutôt que la connaissance de la géométrie simplifiée utilisée pour le rendu c'est parce que la personne qui cherche la marque dans une vue 2D est censée posséder l'original mais n'est pas censée savoir si l'objet original a subi des conversions de format ou des simplifications de maillage par la personne qui a produit les vues 2D.

Tant que la simplification de maillage ne change pas l'apparence de l'objet original, la seule chose qui compte c'est d'utiliser pour l'extraction une représentation géométrique de l'objet original de pair avec le "référentiel de texture" utilisé à l'insertion. De ce point de vue l'éventuelle conversion de format avant le rendu 2D est une opération cachée sans rapport avec les données finales et utiles du problème (la vue 2D et l'objet 3D original).

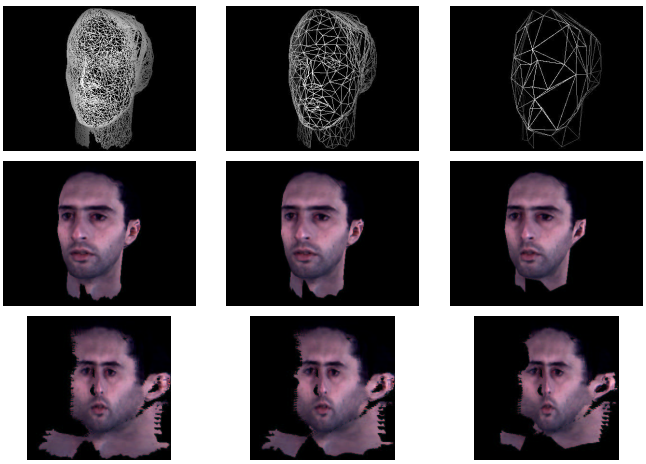
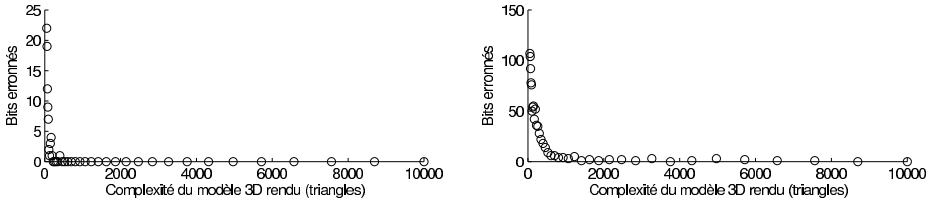


FIG. 4.22 : De gauche à droite, trois niveaux de complexité de l'objet 3D : 10000, 1000 et 100 triangles respectivement. Ligne du haut : géométrie. Ligne du milieu : vue 2D. Ligne du bas : texture reconstruite à partir de la vue 2D du modèle simplifié et de la géométrie originale (non simplifiée).

La figure 4.22 montre trois simplifications de l'objet de la figure 4.3 ainsi que la vue 2D correspondante et la texture reconstruite à partir de la vue et de *l'objet original*. On constate que lorsque la complexité de l'objet est réduite à 100 triangles, la texture reconstruire est une version sensiblement déformée de la texture originale. Ceci est dû au fait que pour une simplification de maillage aussi sévère la géométrie et l'apparence de l'objet ne sont pas conservées. Ainsi l'hypothèse selon laquelle la vue 2D est une vue de l'objet original (non simplifié) n'est plus valide, et la géométrie utilisée pour la reconstruction (géométrie originale) est différente de celle utilisée pour le rendu (géométrie trop simplifiée). En d'autres termes l'inversion de la projection perspective, qui utilise la géométrie originale, n'est plus en accord avec la projection perspective, qui utilisait la géométrie simplifiée, d'où les déformations dans la texture reconstruite.



(a) Message de 64 bits. (b) Message de 256 bits.

FIG. 4.23 : Impact d'une simplification de maillage. Robustesse de l'extraction en fonction de la complexité du maillage simplifié utilisé pour le rendu. Le modèle original est celui de la figure 4.3 avec initialement un maillage de 13957 triangles. Le nombre de pixels de texture visibles est de l'ordre de 32000.

4.7 Impact du référentiel de texture

L'expérience de référence conduite sur deux objets différents a abouti à des résultats très différents, alors qu'on pouvait raisonnablement penser (ou espérer) que la robustesse de l'extraction dépend surtout du nombre de pixels de texture visibles et est relativement indépendente de l'objet ou de la vue considérés. Il se trouve en réalité que l'objet et/ou la vue ont une influence significative. Nous avons évoqué la possibilité que cette influence puisse être réduite en dernière analyse à la variabilité des performances de l'algorithme de tatouage d'images fixe en fonction de la forme de la zone de texture reconstruite. Nous proposons ici trois expériences permettant d'examiner et d'exploiter cette hypothèse.

4.7.1 Complétion de la texture reconstruite

Le moyen le plus simple d'aider le procédé d'extraction du tatouage est de compléter les zones de la texture reconstruite non renseignées. Pour compléter ces zones il n'y a qu'une possibilité : copier l'information provenant de l'image de texture originale (figure 4.24). On suppose en effet connue cette image lors de l'extraction. Par contre on ne peut évidemment pas compléter par l'image de texture originale tatouée. Si on faisait cela on introduirait l'information de tatouage juste avant l'extraction et celle-ci serait forcément positive, or ce qu'on veut c'est tester si oui ou non le tatouage est présent dans une vue 2D donnée.

Il est clair qu'une telle complétion n'apporte aucune information supplémentaire concernant le tatouage proprement dit. Elle a seulement pour but d'aider le procédé d'extraction du tatouage d'images fixes. Il est possible que cela n'ait strictement aucun effet mais en ce qui concerne Eurémark nous pensions que cela pourrait aider car cet algorithme effectue un codage fractal de l'image or ce codage effectue un appariement de blocs. Pour que l'extraction soit au maximum en phase avec l'insertion il nous semble préférable que le codage fractal soit le même à l'insertion et à l'extraction. Or si des zones de

texture sont non renseignées à l'extraction, le procédé d'appariement de blocs ne peut qu'en être affecté, et ainsi de l'extraction du tatouage.

Les figures 4.25 et 4.26 sont à comparer respectivement avec les figures 4.12 et 4.13 de la section 4.3.2. En fait on s'aperçoît que la complétion de la texture n'apporte pratiquement rien et que même parfois les résultats sont légèrement moins bons. Il n'y a que lorsque la résolution de l'image de texture originale est faible et que la quantité d'information cachée n'est pas excessive (64 bits) qu'on observe un gain sensible (partie gauche de la figure 4.25(b)).

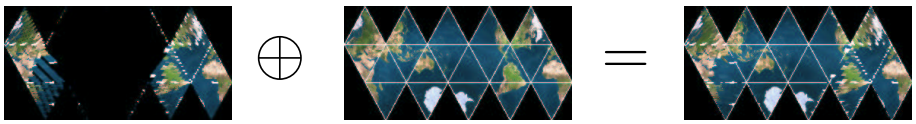
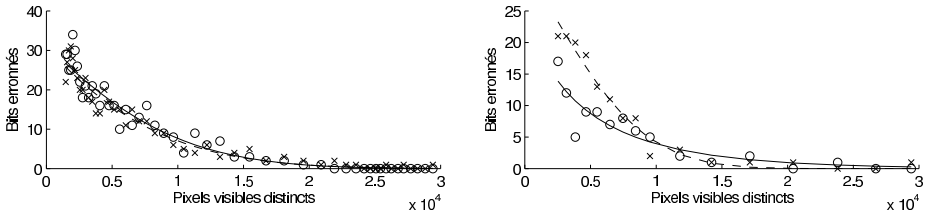


FIG. 4.24 : Complétion de la texture reconstruite. Après reconstruction de la texture on remplit les zones de texture non renseignées en recopiant l'information de texture originale (non tatouée) dans le but d'améliorer l'extraction du tatouage.

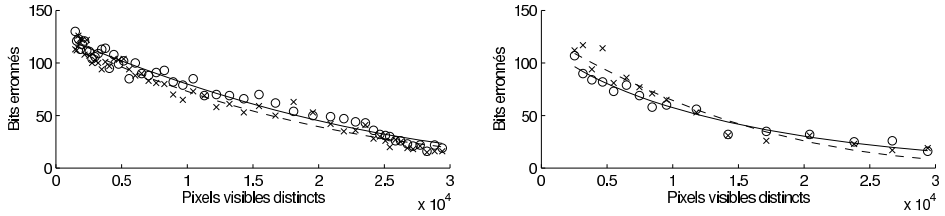
4.7.2 Changement du mode de texturage

Dans l'expérience de référence nous avons utilisé deux objets dont les fonctions de plaquage de texture sont très différentes. L'image de texture du premier est objet est similaire à une carte de texture cylindrique à ceci près que seule la moitié de l'amplitude angulaire est couverte. L'image de texture du second objet n'est pas une projection de la surface texturée de l'objet mais un dépliage de la surface polyédrique de l'objet (qui dans ce cas particulier est possible). Néanmoins rien n'impose d'utiliser une fonction de texturage plutôt qu'une autre. Au lieu d'une projection en coordonnées cylindriques on pourrait utiliser une projection en coordonnées sphériques, ou une projection sur les six faces d'un cube englobant qui seraient dépliées dans l'image de texture.



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.25 : Complétion de la texture reconstruite. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 64 bits et pour la vue de la figure 4.9(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.6 pour comparaison.



(a) Variation du facteur d'échelle de la vue. (b) Variation de la résolution de la texture.

FIG. 4.26 : Complétion de la texture reconstruite. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles distincts pour une marque de 256 bits et pour la vue de la figure 4.9(c) dans les deux cas suivants : variation du facteur d'échelle et variation de la résolution de la texture. Les croix reprennent la figure 4.7 pour comparaison.

Au lieu de déplier les faces d'un polyèdre on pourrait aussi bien utiliser une représentation cylindrique de sa texture (figure 4.27).

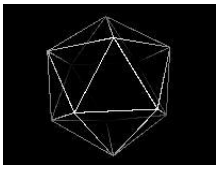
Etant donné que nous avons une certaine liberté dans le choix de la méthode de texturage, on peut se poser la question de savoir s'il y'en a une qui se prête mieux qu'une autre au tatouage d'objets 3D par notre algorithme. Si la réponse à cette question est positive, il y a alors deux moyens de l'exploiter :

1. Imposer que le modèle 3D à tatouer et à diffuser soit texturé suivant la méthode la plus adaptée pour le tatouage, au besoin en convertissant un objet qui au départ ne serait pas dans le bon format et en ne tenant plus compte de son ancien format (figure 4.28). Ceci est discuté dans cette section.
2. Permettre que le modèle 3D à tatouer soit texturé suivant une méthode qui n'est pas la plus adaptée pour le tatouage, mais effectuer une conversion directe avant tatouage et une conversion inverse après tatouage pour effectuer le tatouage proprement dit dans le référentiel de texture le plus adapté (figure 4.30). Cette possibilité est discutée dans la section suivante.

Les résultats des figures 4.29(a) et 4.29(b) sont à comparer avec l'expérience de référence (figures 4.12(a) et 4.13(a)). Nous avons utilisé l'objet et la vue décrits en figure 4.27. La vue est la même que pour l'expérience de référence et le protocole de conversion/tatouage utilisé est celui illustré par la figure 4.28. On note une légère amélioration des résultats lorsque nous utilisons un mode de texturage cylindrique pour l'objet polyédral. Ces résultats restent néanmoins bien inférieurs à ceux obtenus dans l'expérience de référence avec le modèle 3D de visage.

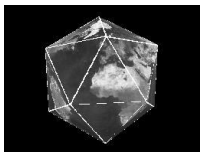
Il est vraisemblable que cette légère amélioration soit due à la forme plus régulière de la texture reconstruite lorsque celle-ci est cylindrique au lieu d'être un dépliage. Mais si cette hypothèse est vraie on ne peut pas en conclure directement que la représentation cylindrique de la texture est meilleure que le dépliage. En effet, en utilisant un point de vue différent de l'objet 3D il est

possible que la texture reconstruite soit composée de deux partie disjointes et séparées par une grande partie centrale noire. Dans ce cas peut-être que l'extraction serait aussi mauvaise que dans l'expérience de référence. Il reste que la représentation cylindrique de la texture est certainement plus apte, en moyenne sur tous les angles de vue possibles, à conduire à une texture reconstruite de forme plus compacte et de contour plus régulier, que des représentations de texture de type dépliage. Dans la mesure où l'algorithme de tatouage d'images fixes sous-jacent serait dépendant de la qualité de la forme de la texture reconstruite, et dans cette mesure seulement, il faudrait penser à utiliser un mode de texturage qui en moyenne conduise le plus souvent à une forme de texture reconstruite favorable.



(a) Géométrie.

(b) Texture.



(c) Une vue 2D.

(d) Texture reconstruite.

FIG. 4.27 : Terre en forme d'icosaèdre utilisant un mode de texturage par projection cylindrique.

4.7.3 Utilisation d'un mode de texturage intermédiaire

Le principe d'imposer un mode de texturage donné pour favoriser le tatouage peut n'être pas applicable si pour certaines raisons on veut conserver le mode de texturage original puis continuer à diffuser et utiliser l'objet selon ce format. Dans ce cas il reste possible de n'envisager la conversion de format que comme une étape intermédiaire et réversible. On se place alors dans un autre format pour tatouer puis on revient au format original (figure 4.30) pour diffuser l'objet. Le tatouage reste présent après conversion inverse, mais pour le détecter il faut se replacer dans l'espace de représentation de la texture où il a été enfoui. En pratique les opérations de conversion peuvent n'être pas parfaitement réversibles dans la mesure où elles impliquent un rééchantillonnage suivant une grille discrète et où les données rééchantillonnées sont elles-même quantifiées. En conséquence il est possible que le gain éventuel obtenu par le passage à une représentation de la texture plus favorable au tatouage soit contrebalancé par des pertes dues aux changements de format. Ceci est confirmé par la figure 4.31 où l'on observe que le gain de performances

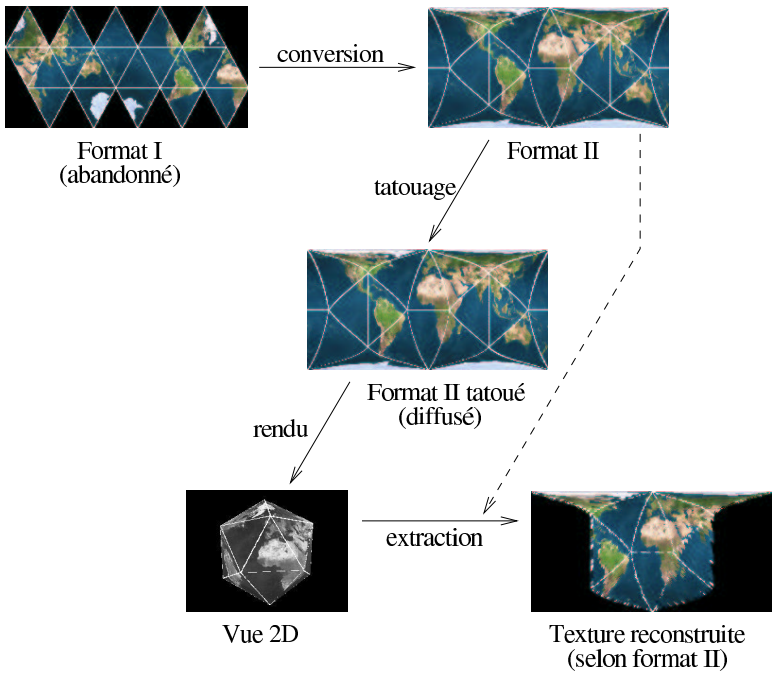
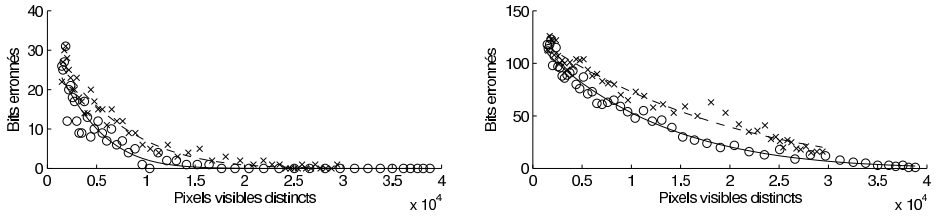


FIG. 4.28 : Changement du mode de texturage.



(a) Message de 64 bits.

(b) Message de 256 bits.

FIG. 4.29 : Impact du référentiel de texture, changement du mode de texturage. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles (variation du facteur d'échelle de la vue). Vue de la figure 4.27(c). Les croix reprennent les figures 4.6(a) et 4.7(a) pour comparaison.

est moindre que celui apporté par l'expérience précédente.

4.8 Commentaires

Le tableau 4.1 récapitule l'ensemble des expériences menées dans ce chapitre. Pour conclure sur l'ensemble des résultats de ces expériences nous ne nous attarderons pas à commenter le détail des chiffres obtenus. Il suffit de constater que des choses sont faisables dans certaines limites : un point de repère est le fait de pouvoir cacher 64 bits dans un objet 3D et de récupérer la marque sans erreur à partir d'une vue 2D qui montre au moins 100×100

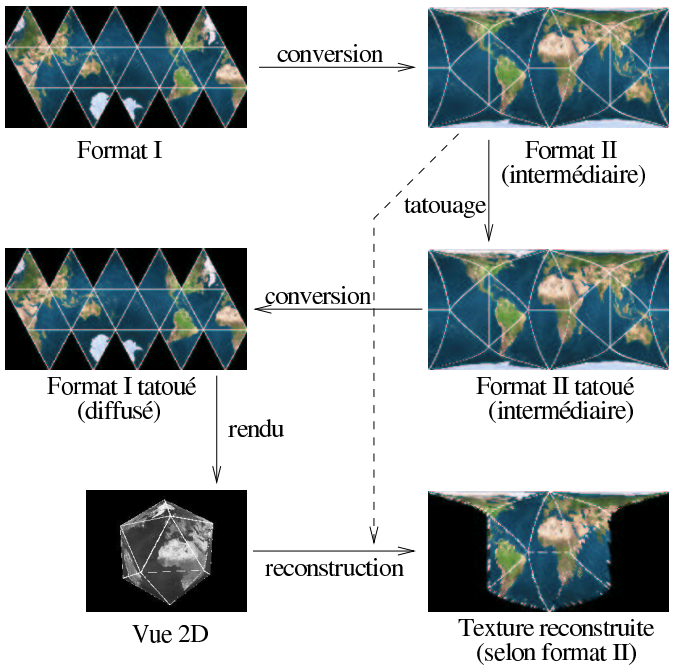
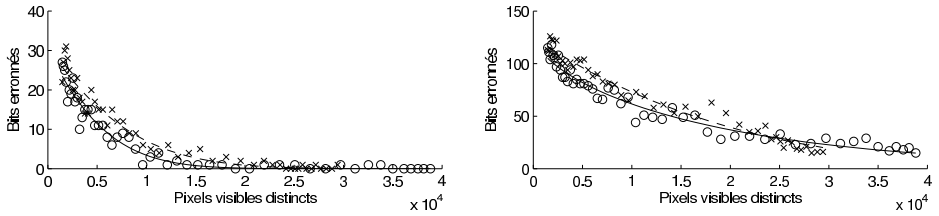


FIG. 4.30 : Utilisation d'un mode de texturage intermédiaire.



(a) Message de 64 bits.

(b) Message de 256 bits.

FIG. 4.31 : Impact du référentiel de texture, mode de texturage intermédiaire. Robustesse de l'extraction en fonction du nombre de pixels de texture visibles (variation du facteur d'échelle de la vue). Vue de la figure 4.27(c). Les croix reprennent les figures 4.6(a) et 4.7(a) pour comparaison.

pixels de texture distincts, et ce en l'absence d'attaques. On remarquera également que la robustesse aux conversions de format de géométrie ou de texture, inhérente au principe de notre algorithme, est vérifiée par l'expérience.

Par ailleurs on notera que la diminution des performances due à un décalage de 1 pixel dans la projection à inverser est plus importante que la diminution des performances due à une compression avec pertes de l'image de texture tatouée ou de la vue 2D ; néanmoins la diminution des performances est comparable dans les deux cas, et bien qu'il soit toujours important de développer un algorithme de recalage permettant d'estimer la projection perspective avec une précision très inférieure au pixel (ce qui est l'objet du chapitre suivant), il

se peut que les performances soient de toute façon limitées par des opérations communes telles qu'une compression JPEG.

Soulignons à nouveau le caractère indicatif des résultats obtenus : il serait intéressant d'effectuer les mêmes expériences avec d'autres algorithmes de tatouage d'images fixes qu'Eurémark, ou même simplement en jouant sur certains paramètres ce cet algorithme. De plus un algorithme de tatouage d'images fixes pourrait être spécialement adapté à l'application de tatouage 3D. Il pourrait par exemple tirer parti de la connaissance de l'agrandissement relatif entre différentes zones de la texture dû au plaquage de la texture sur l'objet (cf. section 3.2.2). L'algorithme pourrait également tirer parti du fait que deux points de normales opposées ne sont généralement pas visibles simultanément dans une vue 2D, et ainsi enfouir la même information dans une zone de texture et dans la zone de texture d'orientation opposée sur l'objet 3D.

TAB. 4.1 : Récapitulation des expériences.

Section	Description	Figures
4.3.1	**Référence.** Expérience de "référence"; tatouage d'un modèle 3D de visage, production d'une vue 2D, reconstruction de la texture à partir de la vue 2D, extraction du tatouage à partir de la vue 2D reconstruite. On fait varier la résolution de la texture, la résolution de la vue 2D, ou le nombre de bits cachés.	4.6, 4.7, 4.8
4.3.2	**Deuxième objet.** Même expérience que l'expérience de référence mais utilisant un objet 3D très différent : modèle de la Terre sous forme d'icosaèdre, image de texture obtenue par "dépliage".	4.12, 4.13, 4.14
4.4.1	**Compression de la texture.** Même expérience que l'expérience de référence, mais après tatouage on effectue une compression JPEG de facteur de qualité 75% de la texture et on regarde l'impact sur les résultats de l'extraction.	4.16, 4.17
4.4.2	**Compression de la vue 2D.** Même expérience que l'expérience de référence mais on regarde l'impact d'une compression JPEG de facteur de qualité 75% de la vue 2D sur l'extraction du tatouage.	4.18, 4.19
4.5	**Projection mal connue.** Même expérience que l'expérience de référence mais on regarde l'impact d'une connaissance imprécise de la projection perspective à inverser sur les performances de l'extraction du tatouage.	4.20, 4.21
4.6	**Simplification de maillage.** Même expérience que l'expérience de référence mais on effectue une simplification de maillage de l'objet 3D après tatouage et avant le rendu 2D.	4.23
4.7.1	**Complétion de la texture reconstruite.** On regarde si l'extraction du tatouage par l'algorithme de tatouage d'images fixes peut être aidé en complétant par l'information de texture originale non-tatouée les zones de textures qui n'ont pas pu être reconstruites. On utilise le second objet de test.	4.25, 4.26
4.7.2	**Changement de mode de texturage.** On regarde si le mode de texturage a un impact sur les performances du tatouage. On remplace le mode de texturage par "dépliage" de la surface polyédrale du second objet par un mode de texturage cylindrique.	4.29
4.7.3	**Mode de texturage intermédiaire.** Comme pour l'expérience précédente on procède au tatouage dans la représentation cylindrique de la texture, mais on diffuse l'objet tatoué en ramenant l'image de texture tatouée dans le format "déplié" original. Le rendu 2D utilise ce format "déplié". La reconstruction de la texture à partir de la vue 2D se fait en considérant le référentiel de texture cylindrique dans lequel l'insertion du tatouage a eu lieu.	4.31

Chapitre 5

Estimation des paramètres de rendu

5.1 Présentation du problème

Dans une application réaliste de notre algorithme on doit être capable d'estimer les paramètres de rendu 2D à inverser pour la reconstruction de la texture. Ces paramètres ne sont pas connus a priori, bien qu'ils aient été supposés connus dans les simulations du chapitre précédent. Néanmoins on peut espérer pouvoir les estimer à partir de la connaissance de l'image 2D et de l'objet 3D. Il s'agit principalement d'estimer l'illumination de l'objet 3D apparaissant dans la vue 2D et la projection perspective entre l'objet 3D et le plan image.

L'estimation de la projection perspective entre des données 3D et des données 2D, autrement appelée recalage projectif 3D/2D, est un problème de base de la vision par ordinateur. Il se présente directement ou indirectement dans tous les problèmes de calibrage de caméra, de suivi d'objet 3D dans une vidéo, de reconstruction 3D, ou de texturage d'objets 3D, pour ne citer que quelques exemples. Toutes ces applications agissent comme un lien entre le monde physique et une représentation informatique de celui-ci. Leur limitation fondamentale tient au fait qu'il est difficile, voire impossible, de modéliser le monde physique avec une précision arbitraire ou avec un modèle se prêtant bien aux calculs.

Par rapport à ces applications notre algorithme a la particularité de s'appliquer uniquement à des images de synthèse et à des objets 3D informatiques, sans référence au monde physique. Nous ne connaissons pas d'autres applications qui nécessitent d'effectuer un recalage projectif entre un objet 3D et une vue 2D synthétique de celui-ci. Il est clair que tous les algorithmes de vision par ordinateur utilisant des images réelles peuvent utiliser des images synthétiques pour leur évaluation ou leur mise au point dans un environnement contrôlé, mais leur objectif final étant d'être applicables à des images réelles, il ne peuvent tirer parti des avantages qu'il y a à ne travailler qu'avec des images de synthèse.

Le principal avantage d'une image de synthèse est que par hypothèse il existe un modèle 3D et un modèle de projection exacts et exploitables conduisant à cette image, et que par conséquent le problème du recalage projectif a une solution exacte, alors que dans le cas d'images réelles il n'est pas certain qu'un modèle 3D et un modèle de projection exacts existent ou du moins que des modèles suffisament précis puissent être employés pour obtenir une solution satisfaisante.

Dans le cadre de notre algorithme, en plus de savoir qu'il existe un modèle 3D et un modèle de projection exacts ayant conduit a une image donnée, nous *connaissons* ces modèles, ce qui serait en toute rigueur impossible avec des images réelles. Seuls les paramètres de la projection sont inconnus et font l'objet du recalage projectif. En plus du recalage proprement dit il faut également estimer l'illumination de la scène 3D pour l'inverser si cela peut aider au recalage et/ou à l'extraction du tatouage dans la texture reconstruite.

5.2 Recalage projectif et vision par ordinateur

En vision par ordinateur le recalage projectif consiste à aligner un modèle 3D informatique sur une image 2D réelle prise par une caméra ou un appareil photo, l'image étant censée représenter le modèle 3D. La relation entre le modèle 3D et l'image peut être approximée par une projection perspective, mais cette approximation n'est pas très bonne si l'objectif de la caméra qui a servi a prendre l'image a une forte distorsion. Le modèle de projection peut être amélioré si on ajoute un ou plusieurs paramètres pour modéliser la distorsion produite par l'objectif. Il faut alors estimer ces paramètres du modèle de projection de pair avec les autres paramètres du recalage. C'est ce qui est fait par exemple dans l'algorithme de calibrage de caméra par un objet plan décrit en [99].

La qualité du recalage dépend aussi de la qualité du modèle 3D employé. Tout comme pour le modèle de projection, si le modèle 3D de l'objet à recaler n'est pas connu à l'avance avec assez de précision on peut lui ajouter des paramètres pour mieux ajuster sa forme à l'objet effectivement observé dans l'image. Ces paramètres doivent aussi être estimés simultanément avec les autres paramètres du recalage [38].

Enfin, la même remarque s'applique, le cas échéant, à la modélisation de l'éclairage auquel est soumis l'objet observé dans l'image. La modélisation de l'éclairage est possiblement le problème le plus difficile étant données la complexité et la diversité des états de surface d'un objet réel et des modes selon lesquels la lumière incidente est propagée, sans oublier l'interaction entre l'objet, son entourage et les sources de lumières [32].

D'un point de vue pratique, le recalage projectif procède la plupart du temps par la mise en correspondance de caractéristiques entre le modèle 3D, la vue 2D et/ou d'autres vues 2D, l'autre possibilité étant la minimisation "brutale" (par descente de gradient par exemple) de la différence photométrique

entre l'image synthétisée et l'image de référence par rapport aux paramètres du modèle 3D et aux paramètres du modèle de rendu.

Les caractéristiques d'objets 3D, dont toutes ne sont pas exploitables pour le recalage, sont soit des caractéristiques purement géométriques — par exemple les lignes de crête, les points de contact des plans bitangents [93], des morceaux de surface ou des points donnés a priori — soit des morceaux de surface texturés ou des points d'intérêts de la surface texturée.

De la même façon les caractéristiques dans une image 2D peuvent être soit des points d'intérêt — par exemple contours apparents, coins ou bords de motifs géométriques — soit des morceaux d'image (imagettes).

Naturellement les caractéristiques géométriques des objets 3D ainsi que les points d'intérêts de la surface texturée sont à mettre en correspondance avec les points d'intérêt des images, et les morceaux de surface 3D texturés sont à mettre en correspondance avec les imagettes 2D. En réalité la mise en correspondance de caractéristiques peut se révéler problématique : soit les caractéristiques ne sont pas localisées avec une précision suffisante, soit les caractéristiques d'un domaine n'ont pas de pendant dans l'autre domaine. Par exemple, bien qu'il soit possible de définir des caractéristiques purement géométriques dans un objet 3D *texturé*, il est impossible de définir des points d'intérêt correspondants dans une image de cet objet, à part le contour apparent. Dans ce cas il vaut mieux mettre en correspondance des points d'intérêt de la texture de la surface de l'objet, ou des morceaux de surface texturés, avec des points d'intérêts dans l'image 2D ou avec des imagettes 2D ; l'utilisation de caractéristiques géométriques 3D est surtout pertinente dans le recalage 3D/3D.

L'utilisation de la texture n'est néanmoins possible que si la texture synthétique permet de modéliser avec assez de fidélité l'objet observé dans l'image 2D. Nous avons déjà fait remarquer que c'est de ce point précis qu'on peut tirer parti dans le cadre du tatouage d'objets 3D basé sur la texture, ce qui n'est habituellement pas possible en vision par ordinateur. En effet, s'il est difficile de connaître ou de modéliser la texture d'un objet réel, la modélisation exacte de la texture d'un objet synthétique est possible, et dans notre cas on suppose même que la texture est connue a priori.

5.3 Estimation de la projection

5.3.1 Approches basées sur la texture

Les objets que nous manipulons étant texturés, on n'a pas d'autre choix que d'utiliser l'information de texture pour le recalage, à part à utiliser le contour apparent, mais nous sommes dubitatif quant à la possibilité de baser un algorithme de recalage précis uniquement sur le recalage du contour apparent d'un objet texturé.

Plusieurs possibilités s'offrent à nous pour exploiter l'information de tex-

ture :

1. Minimiser la différence photométrique entre l'image de référence et l'image estimée par rapport aux paramètres de rendu estimés.
2. Mettre en correspondance des points caractéristiques de la texture.
3. Mettre en correspondance des zones caractéristiques de la texture.

La première possibilité a été exploitée avec succès dans [75] pour des images réelles dans le but d'analyser des expression faciales de visage. Néanmoins cette méthode n'a pas fait l'objet d'une évaluation pour des images synthétiques, et nous ne connaissons pas sa précision absolue en terme de recalage.

La deuxième possibilité est couramment utilisée en vision par ordinateur pour faire du calibrage du caméra. Il s'agit d'avoir un objet de géométrie connue sur lequel sont dessinés des motifs géométriques simples tels que des croix, des carrés, des cercles [38] ou autres, puis de mettre en correspondance les points caractéristiques du modèles 3D avec les points caractéristiques correspondants dans l'image 2D (par exemple les centres des croix détectés par traitement d'image). Cette solution n'est pas très adaptée à notre problème car les objets que nous utilisons peuvent consister en des objets "naturels" dont la texture ne présente pas des motifs géométriques desquels ont peut extraire des points caractéristiques.

Enfin, nous ne connaissons pas de travaux utilisant la troisième approche pour le recalage projectif 3D/2D. Par contre cette approche est très utilisée pour la mise en correspondance d'images 2D [41], que ce soit pour l'estimation de mouvement en codage vidéo, ou pour la construction de la carte de disparité en stéréo-vision, ou autres. Ce manque est peut-être dû au fait que, comme nous le soulignions, en vision par ordinateur classique on ne connaît généralement pas avec précision la texture de l'objet 3D qui est à recaler avec une image 2D réelle.

Cette dernière approche est celle que nous avons retenue en premier, en raison des réserves exprimées vis à vis des deux autres et parcequ'elle nous paraît particulièrement exploitable, compte tenu que nous avons connaissance de l'information de texture de l'objet 3D à recaler.

5.3.2 Notre algorithme

La figure 5.1 illustre le problème du recalage projectif : il s'agit de trouver la projection exacte entre le modèle 3D et la vue 2D cible. Une estimation imprécise de cette projection conduit à une vue déformée par rapport à la vue 2D cible. Le procédé de recalage que nous proposons s'appuie néanmoins sur la connaissance d'une telle vue estimée pour améliorer l'estimation de la projection de manière itérative selon l'algorithme suivant :

1. Calcul de la projection effectué au moyen de correspondances de points 2D-3D. Des appariements initiaux sont fournis manuellement et de meilleurs appariements sont disponibles à chaque itération.

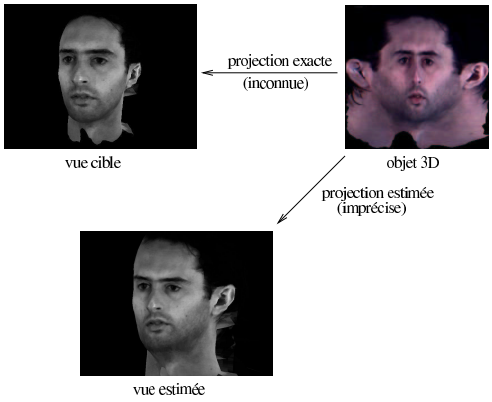


FIG. 5.1 : Le problème du recalage projectif. La déformation de la vue estimée est due à la très forte imprécision de la projection estimée qui ici a été calculée à partir de six appariements manuels. Le but est d'améliorer l'estimation jusqu'à faire coincider les deux vues.

2. Compensation de l'illumination synthétique pour favoriser l'étape d'appariement de blocs qui suit.
3. Appariement de blocs entre l'image cible — sur laquelle l'objet 3D doit etre recalé — et l'image construite par l'estimation courante de la projection. Ces appariements permettent d'obtenir de nouveaux appariements 2D-3D qui seront utilisés à l'itération suivante pour une nouvelle estimation de la projection (voir figure 5.2).

5.3.3 Appariement de blocs et flot optique

Une application fondamentale en vision par ordinateur est le calcul du champ de déplacement, ou flot optique [7, 62, 9], entre deux ou plusieurs images. Ces deux images peuvent être soit des images d'une scène 3D prises au même instant mais de deux points de vue différents, soit des images d'une scène 3D prises d'un même point de vue mais à des instants différents, soit plus généralement des images 2D qui se ressemblent mais pas nécessairement en rapport avec une scène 3D.

Parmis les méthodes de calcul du flot optique on distingue en particulier les méthodes différentielles et les méthodes par appariement de blocs.

Méthodes différentielles

Les méthodes différentielles sont basées sur l'équation de conservation de l'illumination des pixels dans leur mouvement d'une image à l'autre, qui met en relation le champ de déplacement avec les dérivées spatiales (intra-image) et temporelles (inter-image) de l'intensité des pixels.

Ces méthodes sont confrontées au "problème d'ouverture" c'est-à-dire à une indétermination locale de la direction du champ de déplacement qui ne peut

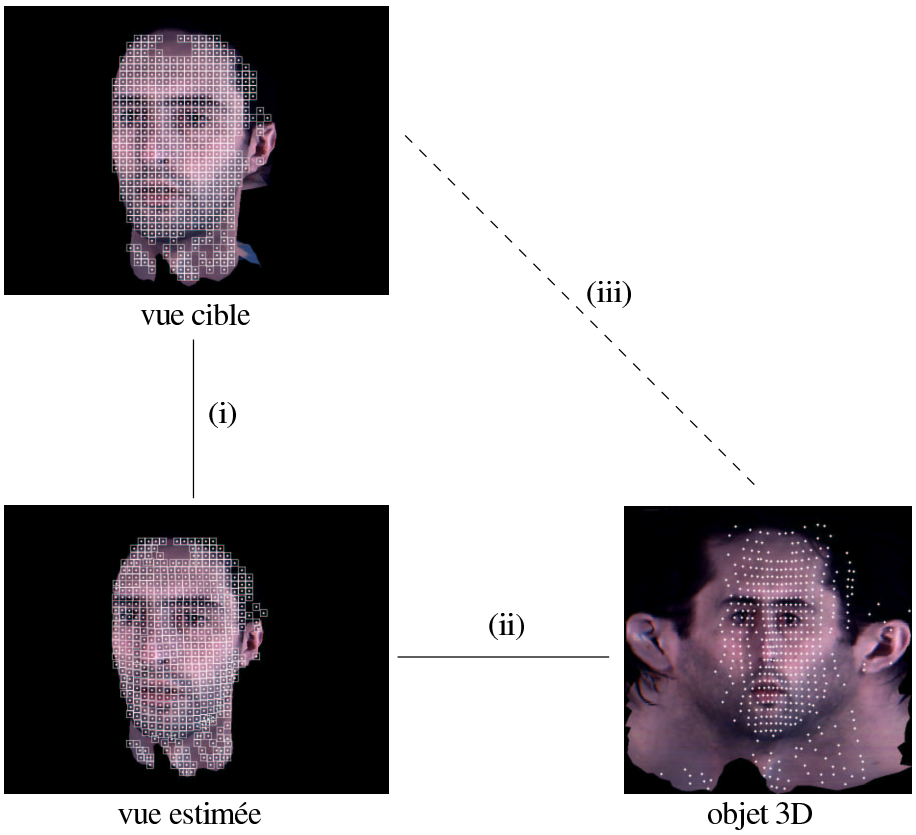


FIG. 5.2 : Amélioration de la projection perspective estimée : des appariements (iii) sont utilisés pour calculer une meilleure estimation de la projection. Les appariements 2D-2D (i) sont obtenus par appariement de blocs. Les appariements 3D-2D (ii) sont obtenus de manière exacte via la projection estimée qui a servi à construire la vue estimée. Enfin, les appariements 3D-2D (iii) sont obtenus par transitivité des appariements (i) et (ii).

être levée que par une hypothèse de régularité globale du champ de déplacement [50].

A la base les méthodes différentielles ne sont applicables qu'à des déplacements de faible amplitude (de l'ordre du pixel), mais sont extensibles à de grands déplacement en adoptant une approche multi-échelle [63].

Appariement de blocs

L'appariement de blocs est la principale alternative aux méthodes différentielles pour le calcul du champ de déplacement d'une image à l'autre, mais cette technique n'est pas limitée au calcul du champ déplacement. Elle est employée notamment en stéréo-vision [58], pour le codage fractal des images [91], ou encore pour l'estimation de mouvement en codage vidéo [94].

Sous sa forme la plus simple, l'appariement consiste à trouver le bloc d'une

image (destination) qui ressemble le plus à un bloc donné dans une autre image (source). Les paramètres d'une telle recherche sont essentiellement la taille des blocs, la taille de la fênetre de recherche, et le critère de similarité entre deux blocs, mais il existe d'innombrables variantes et dérivations de l'algorithme d'appariement de base. Parmi les critères de similarité possibles on peut citer l'erreur quadratique moyenne, l'erreur en valeur absolue ou encore le coefficient de corrélation.

En codage fractal des images, les images source et destination sont les mêmes mais un bloc source est plus grand que le bloc destination correspondant. De plus on considère qu'un bloc destination peut correspondre à un bloc source à via une rotation ou une symétrie (en plus de l'homothétie) ou via un changement d'échelle de l'intensité des pixels (voir Annexe A).

En stéréo-vision, la fenêtre de recherche des blocs appariés est limitée par la géométrie épipolaire (le correspondant d'un bloc ne peut se trouver que sur une certaine droite). Dans certains cas la géométrie (a priori inconnue) de l'objet observé par stéréo-vision est indirectement prise en compte pour modéliser la déformation subie par un bloc entre les deux images et pour en estimer les paramètres en même temps que l'appariement de blocs [26].

En codage vidéo on veut approximer une image à partir de blocs de l'image précédente. Le codage consiste alors à spécifier quels blocs de l'image précédente sont utilisés ainsi que l'erreur commise par l'approximation. La principale difficulté si on veut coder en temps réel est de trouver un bon compromis entre l'exhaustivité de la recherche d'appariements — qui en fournissant potentiellement de meilleurs appariements diminue l'erreur résiduelle et améliore le taux de compression — et la rapidité des calculs, si bien qu'une grande partie de la litérature concernant l'appariement de blocs est consacrée à la diminution du temps de calcul des algorithmes [60] ou à leur parallélisation et leur implantation sur des circuits intégrés dédiés.

Plusieurs techniques existent pour améliorer la précision de l'appariement de blocs, c'est-à-dire pour estimer le mouvement d'un bloc entre deux images avec une précision inférieure au pixel. Une première méthode consiste à suréchantillonner les images et à interpoler les intensités des pixels [10]. Un autre type de méthode utilise une décomposition spectrale des blocs, par exemple une décomposition DCT [56] ou une transformée de Fourier [36]. L'approche fréquentielle permet d'obtenir une précision de $1/N$ pixel, N étant la largeur des blocs.

Pour des déplacements de l'ordre du pixel, les méthodes d'appariement de blocs de précision inférieure au pixel sont équivalentes aux méthodes différentielles de calcul du flot optique [25].

5.3.4 Technique d'appariement utilisée

L'algorithme d'appariement de blocs que nous avons développé pour le recalage projectif possède les caractéristiques suivantes. Des blocs de tailles fixe (par exemple 15×15 pixels) sont définis selon un quadrillage dans l'image

cible. Ils doivent être appariés à des blocs de l'image estimée. On cherche le bloc correspondant dans une fenêtre de recherche de taille fixe (par exemple 15×15 pixels).

Le critère de similarité de deux blocs est la somme des erreurs en valeur absolue entre les pixels correspondants. Nous avons essayé aussi l'erreur quadratique moyenne mais les résultats sont généralement moins bons, i.e. après 4 itérations l'erreur de reprojection moyenne de la projection estimée est environ deux fois plus grande (voir section 5.5 pour l'explication des résultats). D'autres mesures pourraient être pertinentes, notamment le taux de corrélation.

On utilise plusieurs critères pour éliminer des appariements présumés incorrects (*outliers*). On élimine les appariement qui se trouvent à la frontière de la fenêtre de recherche. On élimine les appariements dont l'erreur résiduelle est supérieure à la moyenne des erreurs résiduelles. On élimine les appariements dont l'amplitude du déplacement est supérieure à la moyenne des déplacements. Ces critères sont drastiques mais il faut considérer qu'on dispose de beaucoup d'appariements et qu'il n'est pas nécessaire de retenir tous les appariements valides, par contre il est important que tous les appariements qui sont retenus soient valides. Une poignée d'appariements ayant des déplacements incorrects, même faibles, peut diminuer très significativement la qualité de la projection estimée (en terme d'erreur de reprojection).

On pourrait aussi utiliser d'autres critères ou ajouter des contraintes de cohérence locale [65] ou globale du champ de déplacement ou même prendre en compte la contrainte épipolaire. En effet, s'agissant d'un problème de stéréo-vision détourné, les deux vues 2D de l'objet 3D qu'on se propose de mettre en correspondance sont liées par la contrainte épipolaire. La différence avec le problème de stéréo-vision classique est qu'on ne connait pas la géométrie épipolaire, laquelle est liée à la projection qu'on cherche précisément à estimer. Si on veut tout de même imposer la contrainte épipolaire cela pourrait se faire en ajoutant un facteur de régularisation au champ de déplacement qui exprimerait son respect d'une géométrie épipolaire a priori inconnue. En pratique on peut imaginer procéder itérativement pour imposer la contrainte épipolaire : d'abord estimer la géométrie épipolaire qui colle le mieux au champ de déplacement estimé (voir par exemple [89] pour le calcul de l'épipole d'après le champ de déplacement) puis réestimer le champ de déplacement en prenant en compte le critère de similarité entre blocs *et* le critère d'écart à la géométrie épipolaire calculée précédemment.

La dernière caractéristique de notre algorithme est de rechercher une précision inférieure au pixel lorsque le déplacement d'un appariement est inférieur ou égal au pixel dans chaque direction. Pour cela on considère l'erreur résiduelle associée à l'appariement s'il était décalé de un pixel, dans les quatre directions respectivement. Le décalage inférieur au pixel à apporter à l'appariement dans la direction horizontale est calculé comme le minimum de l'interpolation quadratique de l'erreur résiduelle suivant la direction horizontale. De même pour la direction verticale. Ce calcul simple a permis d'améliorer sensiblement les ré-

sultats du recalage (passage de 0.2 à 0.1 de l'erreur de reprojection moyenne). D'autre méthodes d'appariement de précision inférieure au pixel pourraient etre utilisées — notamment des méthodes différentelles fournissant une expression algébrique "exacte" du déplacement dans le cas d'images couleurs [6] — mais notre principale préoccupation n'est pas de trouver l'algorithme le plus performant sur ce point précis : c'est avant tout de démontrer la faisabilité du recalage pour le tatouage, et la précision obtenue par la présente méthode est tout à fait suffisante à cet égard.

5.3.5 Calcul de la projection

Etant donné un ensemble de couples de points 3D-2D $(X_i, x_i)_{i \in I}$ le recalage consiste à trouver une projection P telle que chaque X_i se projette en le x_i correspondant, c'est-à-dire telle que

$$\forall i \in I, P\tilde{X}_i \equiv \tilde{x}_i \tag{5.1}$$

où P est une matrice 3×4, où

$$\tilde{X}_i = \begin{bmatrix} X_i \\ 1 \end{bmatrix} \tag{5.2}$$

et

$$\tilde{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \tag{5.3}$$

(i.e. représentations en coordonnées homogènes de X_i et x_i), et où le symbole $\equiv$ exprime la relation de collinéarité.

En notant

$$P = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \tag{5.4}$$

et

$$x_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \end{bmatrix} \tag{5.5}$$

le système d'équations précédent peut se mettre sous la forme

$$\forall i \in I, \begin{cases} l_1\tilde{X}_i = (l_3\tilde{X}_i)x_{i,1} \\ l_2\tilde{X}_i = (l_3\tilde{X}_i)x_{i,2} \end{cases} \tag{5.6}$$

Ce système est linéaire en les coefficients inconnus de la matrice P. Si on suppose que les données sont telles qu'il existe une solution exacte, alors celle-ci peut-être simplement trouvée en calculant la solution de ce système linéaire. Etant donné que la solution n'est définie qu'à un facteur d'échelle près (si

P est solution alors λP est aussi solution) la matrice du système n'est pas inversible, mais en supposant qu'on dispose d'assez d'équations pour que le rang de la matrice du système soit déficient seulement de 1, on trouve une solution en prenant le vecteur propre associé à la valeur singulière nulle de la décomposition en valeurs singulières de la matrice du système. Pour que le rang de la matrice du système soit de 11 (nombre de degrés de liberté de P, $3 \times 4 - 1$) il faut au minimum 6 appariements de points (fournissant chacun deux équations).

Si les données dont on dispose sont imprécises alors il n'existe généralement pas de solution exacte au système linéaire précédent. Ce qu'on peut faire est alors de calculer la solution aux moindres de carrés, ce qui revient à prendre pour solution le vecteur propre associé à la plus petite valeur singulière dans la décomposition en valeurs singulières de la matrice du système. Le problème est que les "moindres carrés" dont il s'agit n'ont rien à voir avec la distance entre les PX_i calculés avec la projection P estimée, et les x_i correspondants dans l'image. Or ce qu'on veut c'est minimiser la distance, ou la distance quadratique moyenne, dans le plan image, entre les $P\tilde{X}_i$ et les x_i, c'est-à-dire

$$\sum_{i \in I} \left(\frac{l_1 \tilde{X}_i}{l_3 \tilde{X}_i} - x_{i,1} \right)^2 + \left(\frac{l_2 \tilde{X}_i}{l_3 \tilde{X}_i} - x_{i,2} \right)^2 \tag{5.7}$$

Ce critère est appelée l'erreur de reprojection, et il n'est pas linéaire en les éléments de P. Une méthode de référence (voir par exemple [34]) pour l'estimation de P procède en deux étapes : d'abord une estimation linéaire "aux moindre carrés" de P, puis une minimisation de l'erreur de reprojection par descente de gradient. La descente de gradient demande elle aussi des efforts particuliers car on peut paramétrer les 11 degrés de liberté de P de plusieurs manières mais tous ne permettent pas d'effectuer la descente de gradient dans de bonnes conditions.

S'il est incontestable qu'une résolution linéaire sans minimisation de l'erreur de reprojection est imprécise et instable, il existe un prétraitement simple des données qui peut améliorer la qualité de la solution linéaire à tel point qu'une minimisation supplémentaire de l'erreur de reprojection devient pratiquement superflue. C'est cette approche qui est défendue dans [46]. Le titre "In Defense of the Eight-Point Algorithm" fait référence au problème de la stéréovision où il s'agit d'estimer la "matrice fondamentale" entre deux images d'une même scène prises sous deux points de vue différents, et où un minimum de huit appariemments 2D-2D est nécessaire pour calculer les huit degrés de libertés de la matrice en question. Ce problème présente beaucoup de similitudes avec le nôtre et l'idée mise en avant dans [46] est a priori transposable à des appariements 2D-3D.

Cette idée consiste à normaliser les coordonnées d'espace et les coordonnées d'image de sorte que la moyenne des x_i soit nulle et que leur matrice de covariance soit la matrice unité, et de même pour les X_i. Si on note (M, KK^T) la moyenne et la covariance des X_i et (m, kk^t) la moyenne et la covariance des

x_i, on calcule donc $X'_i = K^{-1}(X_i - M_i)$ et $x'_i = k^{-1}(x_i - m_i)$. En coordonnées homogènes cela donne

$$\tilde{X}'_i = \begin{bmatrix} K^{-1} & -K^{-1}M \\ \mathbf{0} & 1 \end{bmatrix} \tilde{X}_i \tag{5.8}$$

et

$$\tilde{x}'_i = \begin{bmatrix} k^{-1} & -k^{-1}m \\ \mathbf{0} & 1 \end{bmatrix} \tilde{x}_i \tag{5.9}$$

Ensuite on estime la projection P' entre les X'_i et x'_i en résolvant le système 5.6 aux moindres carrés. Enfin, on prend pour estimation de la projection entre les X_i et les x_i la valeur

$$P = \begin{bmatrix} k & m \\ \mathbf{0} & 1 \end{bmatrix} P' \begin{bmatrix} K^{-1} & -K^{-1}M \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.10}$$

Cette solution est bien meilleure, que si on résolvait directement le système 5.6 sans normalisation des coordonnées. La raison en est que la normalisation des coordonnées donne aux coefficients du système linéaire des ordres de grandeur comparables, ce qui en améliore le conditionnement. Ceci est démontré plus en détail dans [46].

En conclusion, lorsque nous avons à calculer la projection à partir d'appariements 3D-2D nous utilisons la méthode linéaire avec normalisation des coordonnées. A l'appui de ce choix il faut souligner que lorsqu'on dispose de nombreux appariements cohérents, ce qui est le cas après la première itération de notre algorithme, la solution du système est d'autant plus stable.

5.4 Estimation de l'illumination

5.4.1 Aperçu du problème

Lorsqu'on veut mettre en correspondance deux images d'une même scène, obtenues soit de points de vue différents (par exemple en stéréo-vision) soit à des instants différents (par exemple pour l'estimation de mouvement), on est confronté au problème de la disparité photométrique entre ces deux images, autrement dit, un point d'un objet n'a généralement pas la même couleur ou la même intensité lumineuse dans les deux images. Le même problème peut se présenter pour le calcul de flot optique entre deux images successives d'une séquence vidéo où l'hypothèse fondamentale est que dans leur mouvement les pixels gardent une intensité lumineuse constante, hypothèse qui est rarement strictement vérifiée. Dans ces conditions, il peut être intéressant de modéliser la disparité photométrique [48] et de l'estimer en même temps ou avant d'effectuer la mise en correspondance des deux images.

Dans notre cas, il ne s'agit pas seulement d'images 2D, mais d'images 2D construites à partir d'une représentation 3D dont on connaît certains éléments

(la géométrie et la texture de l'objet 3D principal). Ce qu'on veut c'est réduire (ou compenser) la disparité photométrique entre l'image cible (vue 2D dans des conditions inconnues) et l'image estimée (vue 2D dans conditions maîtrisées) avant de procéder à l'appariement de blocs entre ces deux images. Pour ce faire, le plus naturel nous semble être d'estimer le modèle d'éclairage ayant contribué au rendu de l'image 2D cible, de manière à synthétiser dans les mêmes conditions l'image 2D estimée, mais voyons les méthodes existantes.

5.4.2 Méthodes existantes

On distingue trois types de méthodes pour compenser l'illumination observée dans la vue 2D d'un objet 3D. Les premières sont basées sur la répartition des intensités lumineuses dans une image sans référence à un modèle 3D extérieur. Cela peut consister en la normalisation de l'histogramme de luminance [53], ou par le moyennage de plusieurs images similaires mais sous un éclairage différent, notamment pour l'acquisition de la texture d'un objet [39]. Néanmoins ces approches sont peu pertinentes pour ramener dans un "référentiel de luminance" commun des images similaires mais éclairées différemment lorsqu'elles présentent de fortes différences de luminance locales dues soit à un fort éclairage directionnel, soit à des réflexions spéculaires.

Ensuite on trouve les techniques qui visent à modéliser l'illumination de la scène 3D observée. Ce problème est d'autant plus difficile s'il y a d'autres inconnues à estimer en même temps, telles que la couleur de la texture et la BRDF [2] de l'objet 3D [77] (rappelons que dans notre cas on peut supposer connues la couleur et la BRDF, qui sont des données de l'objet 3D original). Dans [79] le modèle d'éclairage est défini par un certain nombre de paramètres (couleur de l'éclairage ambiant, direction et couleur d'une source de lumière), lesquels sont estimés en minimisant la différence entre l'image cible et l'image rendue d'après ces paramètres.

Les algorithmes de la dernière catégorie ne cherchent pas à modéliser l'illumination de la scène 3D d'un point de vue géométrique (i.e. en cherchant la position des sources de lumières) mais à décomposer l'image 2D cible sur une base d'images obtenues à partir d'éclairages prédéfinis. Ainsi [88] utilise sept sources de lumière à des positions prédéfinies par rapport à l'objet 3D principal ou par rapport à la caméra virtuelle. L'objet 3D est éclairé successivement par chacune de ces sept sources de lumière pour produire une base de sept images. L'image cible est alors considérée comme une combinaison linéaire des sept images de base, dont les coefficients sont calculés par simple inversion d'un système linéaire. Ainsi une ou plusieurs sources de directions inconnues sont approximées par la somme des contributions de quelques sources prédéfinies.

Des travaux récents [8] montrent que l'ensemble des images d'un objet 3D de réflectance lambertienne obtenues sous un point de vue constant mais sous un éclairage quelconque peut être approximé par un espace linéaire de dimen-

[2] Bidirectional Reflectance Distribution Function, [64]

sion neuf. Comme base de cet espace les auteurs de [8] ont choisi les images produites à partir de modèles d'éclairages où la distribution de l'intensité lumineuse reçue (de l'infini) en fonction de la direction sont les neuf premières harmoniques sphériques.

5.4.3 Choix de l'algorithme

Parmis les méthodes que nous venons d'évoquer il nous semble que la dernière présente le meilleur compromis entre simplicité, efficacité, généralité, et justification théorique. En effet elle peut traiter indifféremment des situations où il y a une seule ou un nombre quelconque de sources de lumière distantes, alors que dans le cas de l'estimation du modèle par descente de gradient, la complexité du modèle choisi et de sa paramétrisation (par exemple une seule source de lumière directionnelle) a un impact direct sur les performances de l'algorithme. En outre la descente de gradient présente tous les inconvénients habituels concernant la définition d'une estimation initiale permettant de converger vers la solution optimale.

De plus les auteurs de [8] suggèrent que leur modélisation et leur méthode puissent être utiles pour traiter le cas de surfaces spéculaires en augmentant l'ordre des harmoniques sphériques utilisées. La seule chose qu'il nous semble a priori difficile à modéliser par cette méthode sont les ombres portées d'un objet non-convexe sur lui-même. La forme de ces ombres dépend de la position précise des sources de lumières et l'image qu'elles forment ne peut être approximée comme la combinaison linéaire d'une poignée d'images de base. Dans ce cas seulement une modélisation "géométrique" des sources de lumières et une résolution par descente de gradient peut-elle être utile.

Idéalement nous voudrions pouvoir compenser tous les types d'illumination synthétique qui préservent l'apparence de l'objet 3D tatoué. Mais la première préoccupation est de valider le principe de la chaîne algorithmique complète du tatouage d'objets 3D basé sur la texture, sans nécessairement utiliser des algorithmes optimaux à chaque étape. La première étape était de lever l'hypothèse de la projection connue, en effectuant un recalage projectif. Désormais il d'agit de lever l'hypothèse que l'illumination doit être connue. Un premier pas est de considérer que la surface de l'objet a une réflectance lambertienne, et d'estimer l'illumination à partir d'une des méthode évoquées.

On ne peut toutefois pas employer ces méthodes directement sans aucune réserve car elles présupposent que le modèle 3D est déjà aligné avec l'image 2D et elles procèdent par la minimisation pixel à pixel de l'erreur entre l'image 2D cible sous éclairage inconnu et l'image 2D resynthétisée à partir du modèle 3D sous éclairage estimé. Or au début de la procédure itérative de recalage, le modèle 3D n'est pas parfaitement aligné avec l'image 2D cible, ce qui signifie que les deux images 2D à comparer pour estimer l'illumination ne représentent pas l'objet 3D sous le même point de vue et ne peuvent théoriquement pas être comparées pixel à pixel. C'est d'ailleurs le genre de problèmes qui pourraient justifier l'emploi de techniques de normalisation des histogrammes de

luminances, car elles s'appliquent à chaque image de manière indépendante. La seule chose qui importe est qu'elles représentent les mêmes surfaces de l'objet dans les mêmes proportions. Cependant, comme nous l'avons déjà dit, cette approche globale n'est d'aucune utilité pour compenser les effets plus ou moins localisés de l'éclairage synthétique, à moins de développer une technique de normalisation localisée. Un filtrage passe-haut pourrait aussi contribuer à estomper l'effet de l'éclairage et à conserver les traits caractéristiques de la texture.

Ces réserves étant faites, on peut espérer que les méthodes d'estimation de l'éclairage basées sur l'alignement du modèle 3D et de l'image cible puissent donner des résultats exploitables même si l'alignement n'est pas très bon. Par résultat exploitable nous entendons une compensation d'illumination qui, même imparfaite, favorise l'appariement de blocs. Finalement nous avons donc employé la méthode de [8].

5.5 Résultats

Nous présentons ici quelques résultats obtenus par l'application de notre méthode de recalage projectif comprenant une étape de compensation d'illumination et une étape d'appariement de blocs. Tout d'abord faisons remarquer que le problème de la compensation de l'illumination synthétique, préalable à l'appariement de blocs, est presque parfaitement résolu par l'algorithme de [8] (du moins sous l'hypothèse que nous faisons que l'objet à une réflectance uniquement lambertienne), et ce même lors de l'itération initiale de notre algorithme où les deux vues ne sont pas très bien recalées. Dans ces conditions, qu'on simule une expérience réaliste avec éclairage synthétique, ou une expérience moins réaliste — sans éclairage synthétique — n'a pas une incidence significative sur les performances de la procédure d'appariement de blocs ou sur les résultats du recalage projectif. C'est pourquoi les résultats qui suivent traduisent uniquement l'efficacité de la procédure d'appariement de blocs et du calcul de la projection basé sur les appariements.

5.5.1 Premier objet

Dans une première expérience nous avons repris le modèle 3D de visage et la vue 2D utilisés pour l'expérience de référence en tatouage (voir section 4.3). La vue 2D étant donnée il s'agit donc de retrouver la projection perspective qui la relie au modèle 3D. Pour initialiser notre algorithme de recalage nous avons sélectionné manuellement 10 paires de points entre la vue 2D et le modèle 3D (voir figure 5.3).

Le calcul de la projection perspective à partir de ces appariements manuels est déjà très bon (voir figure 5.4), du moins suffisament pour permettre une bonne estimation de l'illumination et pour permettre à notre algorithme itératif d'appariement de blocs et de calcul de projection de converger.

(a) Vue 2D cible. (b) Texture de l'objet 3D.

FIG. 5.3 : Appariements manuels initiaux pour le recalage projectif. La différence de couleurs entre la vue 2D et la texture de l'objet 3D est due à l'éclairage synthétique utilisé dans la vue 2D. L'estimation de cet éclairage fait partie intégrante du procédé de recalage.

(a) Vue 2D estimée. (b) Différence avec la vue cible.

FIG. 5.4 : Première estimation de la vue 2D à partir des appariements manuels et avec estimation de l'illumination. Le recalage est déjà visuellement très bon, mais les écarts de l'ordre de plusieurs pixels ne sont pas tolérables pour l'application de tatouage 3D.

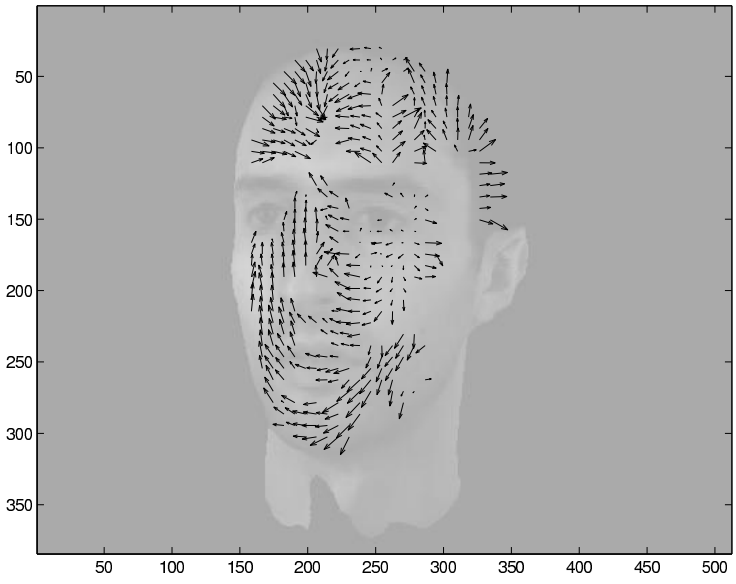
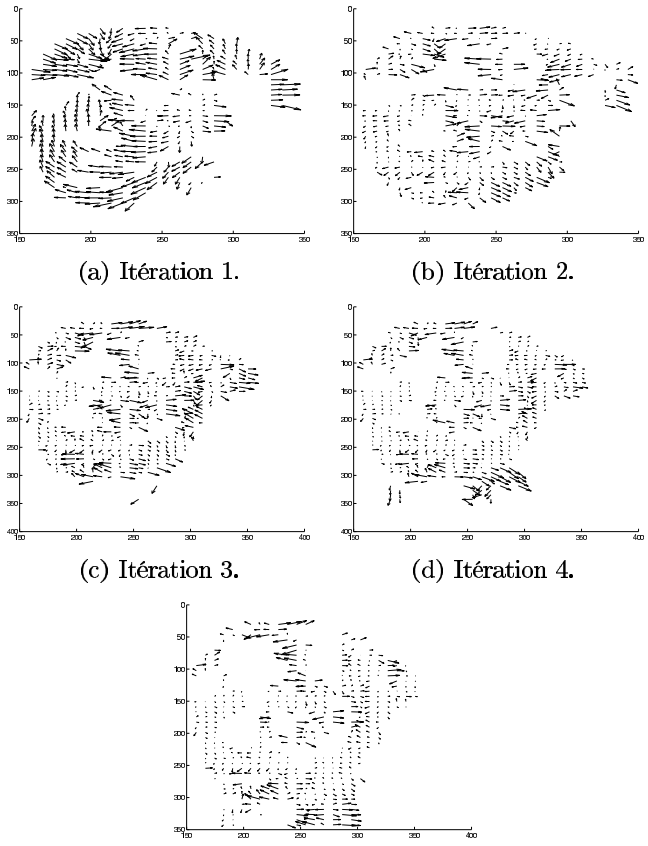


FIG. 5.5 : Champ de déplacement calculé par appariement de blocs entre la vue 2D cible (image de fond) et la vue 2D estimée.



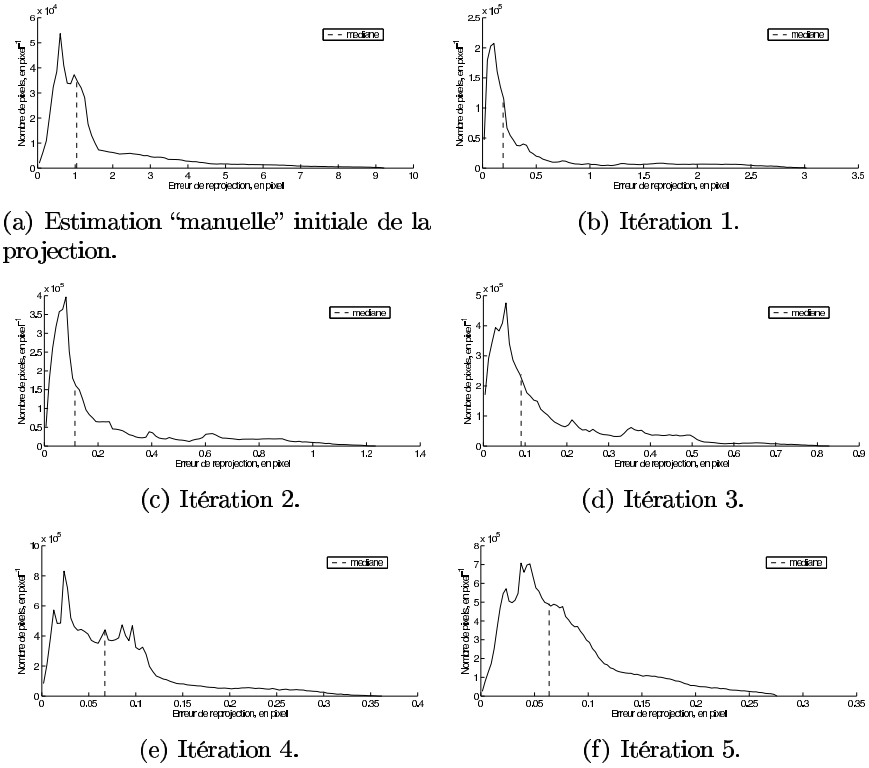
(a) Itération 1. (b) Itération 2.

(c) Itération 3. (d) Itération 4.

(e) Itération 5.

FIG. 5.6 : Champ de déplacement calculé dans les cinq premières itérations. Le premier champ est le même qu'en figure 5.5. Concernant l'échelle des vecteurs, dans la première itération l'amplitude maximum des vecteurs de déplacement est de l'ordre de 1.7 pixel et dans la cinquième itération elle est de l'ordre de 0.5 pixel.

La figure 5.5 montre le résultat de l'appariement de blocs entre l'image 2D cible et l'image 2D destination. Tout le visage n'est pas couvert, les "trous" sont dus à l'élimination des appariements jugés peu fiables. Les vecteurs restants ont une amplitude maximum de l'ordre de 1.7 pixel. Les appariements 2D-3D qu'ils permettent d'obtenir, et qui sont incomparablement plus nombreux et précis que les 10 appariements manuels initiaux, permettent de calculer une estimation plus précise de la projection perspective, ce qui conduit à une vue 2D estimée encore plus proche de la vue 2D cible, et à un champ de déplacement de moindre amplitude à l'itération suivante. La figure 5.6 montre justement le champ de déplacement calculé lors des cinq premières itérations. On constate que dès la deuxième itération de nombreux blocs coincident entre l'image cible et l'image estimée, et à chaque itération leur nombre augmente. Ces blocs sont caractérisés par un vecteur de déplacement d'amplitude très inférieure au pixel mais pas tout à fait nulle à cause du traitement effectué sur les vecteurs de



(a) Estimation "manuelle" initiale de la projection.

(b) Itération 1.

(c) Itération 2.

(d) Itération 3.

(e) Itération 4.

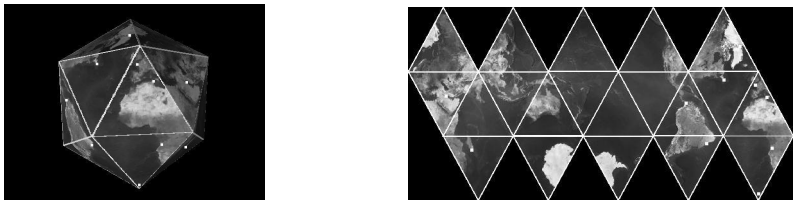
(f) Itération 5.

FIG. 5.7 : Histogrammes de répartition de l'erreur de reprojection pour la projection estimée manuellement et pour la projection estimée lors des cinq premières itérations de la procédure de recalage.

faible amplitude (déplacement de 0 ou 1 pixel dans chaque direction) pour obtenir une précision inférieure au pixel.

On s'arrête à la cinquième itération car au-delà le critère de qualité du recalage cesse de s'améliorer de façon monotone et commence à fluctuer. Le critère que nous avons considéré est l'erreur de reprojection, c'est-à-dire l'erreur commise sur la position de la projection des points de l'objet 3D lorsqu'on utilise la projection estimée au lieu de la vraie projection. Notons que ce critère n'est calculable que lorsqu'on connaît la vraie projection, c'est-à-dire uniquement dans le cadre d'expérience contrôlées, et son seul intérêt est d'évaluer la précision de la procédure itérative de recalage mais évidemment pas de servir de critère d'arrêt.

Pour le calcul de l'erreur de reprojection nous considérons tous les pixels de l'image 2D cible qui représentent un point de l'objet 3D, nous calculons le point correspondant de l'objet 3D (grace à la connaissance exacte de la projection) puis nous le reprojetons dans l'image par la projection estimée, et enfin on regarde la distance entre le point de départ et le point d'arrivée. On fait donc ceci pour chaque pixel de la vue 2D cible et on présente le résultat sous forme d'un histogramme de répartition (nombre de pixels par unité d'erreur



(a) Vue 2D cible. (b) Texture de l'objet 3D.

FIG. 5.8 : Appariements manuels initiaux pour le recalage projectif de l'objet polyédral.

de reprojection).

La figure 5.7 présente la répartition de l'erreur de reprojection pour la projection calculée d'après les appariements manuels et pour la projection calculée lors des cinq premières itérations de la procédure de recalage. On a encore la confirmation que la projection initiale est très bonne pour une projection obtenue par des appariements manuels, en effet, bien qu'un nombre marginal de pixels se trouvent éloignés de plus de 5 pixels de leur position de référence, l'erreur de reprojection se trouve quand même majoritairement regroupée autour d'une erreur de 1 pixel. L'erreur de reprojection médiane est améliorée significativement lors des quatre premières itérations du recalage.

La vue 2D utilisée dans cette expérience a une taille de 512×384 pixels. L'erreur de reprojection médiane obtenue à la cinquième itération est de 0.064 pixel. En utilisant exactement la même vue mais augmentée ou diminuée de taille on obtient des résultats du même ordre de grandeur mais tout de même assez variables. Par exemple si la vue 2D a une taille de 800×600 pixels, l'erreur de reprojection médiane minimum est de 0.080 et est obtenue à la septième itération. Si la taille est de 400×300 l'erreur de reprojection médiane est de 0.099 et est obtenue à la neuvième itération. Dans ce dernier cas, on peut conjecturer que le moins bon résultat est dû à la diminution du nombre de blocs utilisés pour estimer le champ de déplacement entre la vue 2D estimée et la vue 2D cible. En effet nous avons gardé la même politique de définition des blocs à apparier (blocs équirépartis selon une grille carrée dont le pas est de 15 pixels) et si la taille de l'image diminue cela signifie moins blocs, et donc une estimation du champ de déplacement globalement moins précise.

Il serait intéressant de voir comment l'erreur de reprojection varie en fonction de la densité du champ de déplacement estimé, qui à la limite pourrait être estimé en chaque pixel.

5.5.2 Deuxième objet

Pour vérifier la validité des résultats Nous avons mené la même expérience sur notre deuxième objet (figure 5.8). Là encore les 10 appariements manuels que nous avons choisis ont donné un très bon recalage initial (figure 5.9). La figure 5.10 montre le champ de déplacement obtenu par appariement de blocs

(a) Vue 2D estimée. (b) Différence avec la vue cible.

FIG. 5.9 : Première estimation de la vue 2D à partir des appariements manuels et avec estimation de l'illumination.

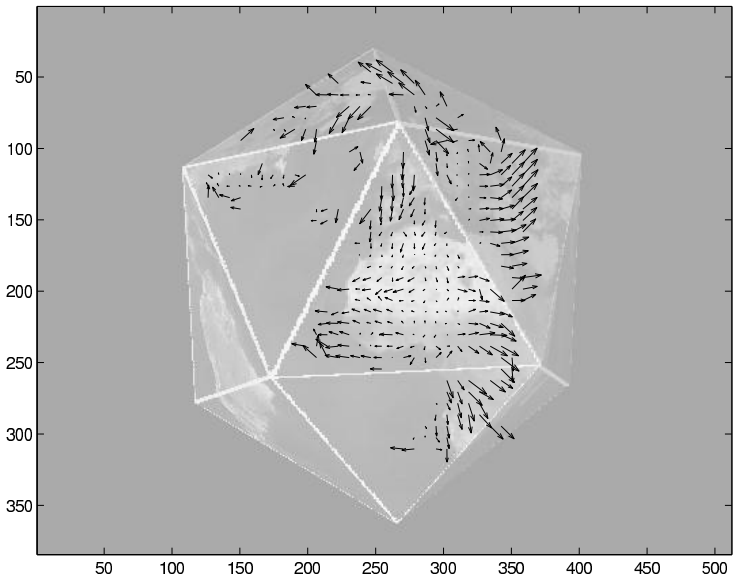
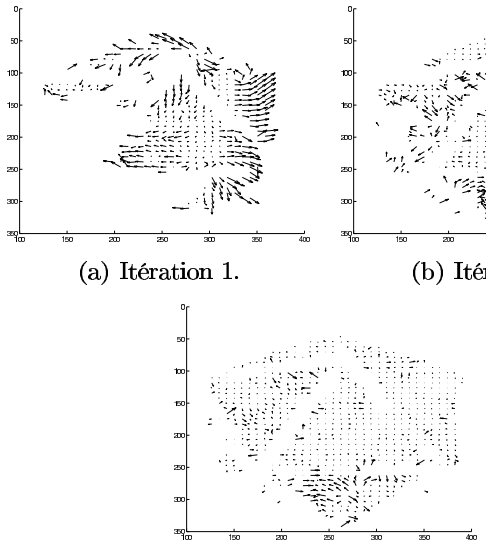


FIG. 5.10 : Champ de déplacement calculé par appariement de blocs entre la vue 2D cible (image de fond) et la vue 2D estimée.

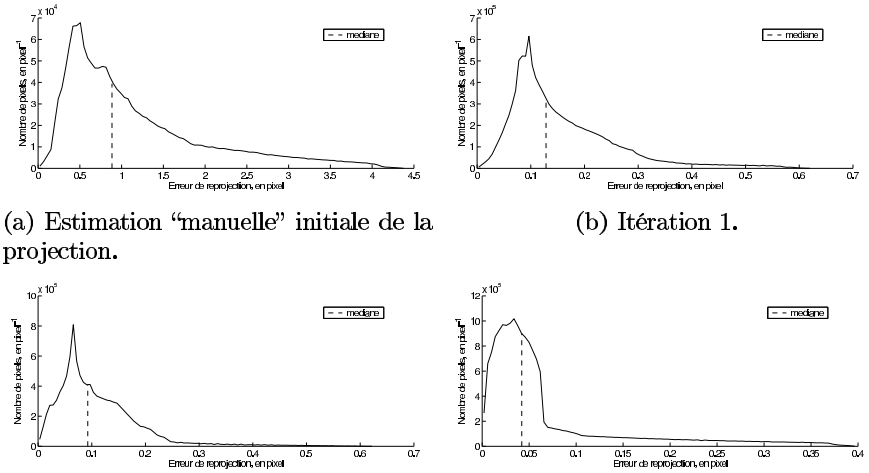
entre la vue 2D cible et la vue 2D estimée par les appariements manuels. Il présente de nombreuses lacunes, mais ce qui est important est que les vecteurs de déplacement retenus soit corrects afin d'améliorer l'estimation de la projection. Avec une meilleure projection la fiabilité des appariements est meilleure car les blocs sont moins déformés entre les deux images, et par suite le nombre de vecteurs retenus comme fiables augmente, ce que confirme la figure 5.11 : il y a moins de lacunes dans le champ de déplacement après la première itération. Enfin, la figure 5.12 montre l'évolution de l'erreur de reprojection. La convergence est moins rapide que la première expérience (c'est pourquoi nous n'avons pas montré toutes les itérations) mais il se trouve que l'erreur de reprojection médiane finale y est moindre (meilleure).



(a) Itération 1. (b) Itération 2.

(c) Itération 8.

FIG. 5.11 : Champ de déplacement calculé lors de trois (sur huit) itérations. Le premier champ est le même qu'en figure 5.10.



(a) Estimation "manuelle" initiale de la projection.

(b) Itération 1.

(c) Itération 2.

(d) Itération 8.

FIG. 5.12 : Histogrammes de répartition de l'erreur de reprojection pour la projection estimée manuellement et pour la projection estimée lors de trois (sur huit) itérations de la procédure de recalage.

5.5.3 Limitations

Le comportement de l'algorithme de recalage que nous avons proposé repose entièrement sur la qualité de la procédure d'appariement de blocs. Dans nos expériences nous avons réalisé cet appariement de blocs en recherchant le correspondant d'un bloc dans une fenêtre autour de celui-ci, indépendem-

ment pour chaque bloc, et avec une préoccupation minimum pour la cohérence globale du champ de déplacement obtenu. On espère assurer l'absence d'appariements incorrects par un élagage "brutal" des vecteurs du champ de déplacement.

Une approche aussi "simpliste" n'a pu donner de bons résultats que parce que nous avons travaillé avec des images en couleur, c'est-à-dire des images dont les pixels ne sont pas représentés par des scalaires mais par des vecteurs de dimension trois. Notre méthode n'est pas directement transposable au cas d'images en noir et blanc, comme nous l'avons vérifié expérimentalement.

Pour le calcul du flot optique dans une image en niveaux de gris on est confronté au problème d'ouverture, ce qui n'est en général pas le cas avec des images couleurs (cf. [6]). Théoriquement ce problème se pose avant tout pour les méthodes différentielles de calcul du flot optique, mais il se pose également pour la méthode par appariement de blocs lorsque les blocs sont trop petits. Malheureusement on ne peut pas augmenter arbitrairement la taille des blocs sans risquer d'invalider l'approximation selon laquelle les blocs correspondants se correspondent par une simple translation. Dans ces conditions il faudrait imposer une contrainte de régularité au champ de déplacement, par exemple la contrainte épipolaire.

Une autre voie à explorer serait de relier la "projection différentielle" (différence entre la projection estimée courante et la projection exacte inconnue) et le champ de déplacement entre l'image estimée et l'image cible et de voir dans quelle mesure cela permet d'imposer une contrainte de régularité au champ de déplacement (dont la forme globale serait paramétrée par les onze paramètres de la projection différentielle). La difficulté est vraisemblablement de trouver le bon paramétrage pour la projection différentielle.

Un paramétrage peut-être un peu naïf serait d'exprimer P sous forme matricielle

$$P = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \tag{5.11}$$

et de prendre la différentielle

$$dP = \begin{bmatrix} dl_1 \\ dl_2 \\ dl_3 \end{bmatrix} \tag{5.12}$$

où on fixe la dernière composante de dl_3 à 0 (ce qui revient à dire que la composante correspondante de P est constante, et fixée par exemple à 1).

Si on se donne alors un point

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.13}$$

de l'image estimée et X son point correspondant sur l'objet 3D ($\tilde{X}$ en coordonnées homogènes) on a

$$\begin{cases} x_1 = \frac{l_1\tilde{X}}{l_3\tilde{X}} \\ x_2 = \frac{l_2\tilde{X}}{l_2\tilde{X}} \end{cases} \tag{5.14}$$

et

$$\begin{cases} dx_1 = \frac{(dl_1\tilde{X})(l_3\tilde{X}) - (l_1\tilde{X})(dl_3\tilde{X})}{(l_3\tilde{X})^2} \\ dx_2 = \frac{(dl_2\tilde{X})(l_3\tilde{X}) - (l_2\tilde{X})(dl_3\tilde{X})}{(l_3\tilde{X})^2} \end{cases} \tag{5.15}$$

et on observe que le champ de déplacement global $(dx_1, dx_2)_{i,i\in I}$ s'exprime linéairement en fonction de (dl_1, dl_2, dl_3). Comme par ailleurs P et les $\tilde{X}_i$ sont connus et fixés, cela permet de dire que le champ de déplacement global est à chercher dans un espace vectoriel de dimension onze (4 dimensions pour dl_1 et dl_2 et 3 seulement pour dl_3, dont la quatrième composante est fixée à 0). On peut alors soit tenir compte de cette contrainte lors de l'estimation du champ de déplacement, soit l'imposer après avoir calculé le champ de déplacement. Dans ce dernier cas, une résolution aux moindres carrés du système 5.15 permettrait d'estimer dP à partir de $(dx_1, dx_2)_{i,i\in I}$, et de calculer la nouvelle estimation $P + dP$ de la projection.

Notons que d'autres paramétrages de P et de dP pourraient être plus appropriées, par exemple faisant intervenir la décomposition de P sous forme de rotation, translation et projection perspective. De plus dP pourrait être exprimé sous la forme d'une composition de transformations plutôt que sous la forme d'une addition de matrices.

Annexe A

L'algorithme de tatouage Eurémark

Le logiciel de tatouage d'images fixes utilisé dans nos expériences est un algorithme de tatouage aveugle documenté dans [29] et dont le principe est inspiré du codage fractal des images et de la notion d'auto-similarité (illustrée par la figure A.1). L'idée principale est de tirer parti de certaines propriétés d'invariance du codage fractal, telles que l'invariance aux transformations affines géométriques ou photométriques, pour assurer la robustesse du tatouage.

A.1 Insertion du tatouage

Le procédé d'insertion peut être décomposé en trois étapes : formatage et cryptage du message à dissimuler, génération du support, et combinaison du tatouage au support.

A.1.1 Formatage et cryptage de la marque

Les bits du message à cacher sont répartis de manière redondante, à la fois par duplication et par sur-échantillonnage du message, de manière à obtenir une marque ayant la taille de l'image. Cette redondance est nécessaire pour améliorer la robustesse, notamment à la découpe. Ensuite la marque est cryptée par un XOR avec une séquence binaire pseudo-aléatoire générée par une clé secrète, ce qui donne la marque cryptée W. Le XOR permet d'une part de sécuriser le message caché et d'autre part de supprimer les motifs réguliers dus à la duplication et au sur-échantillonnage du message, ce qui en diminue l'impact psycho-visuel. Pour améliorer la robustesse il est possible d'utiliser des codes correcteurs avant la duplication et le sur-échantillonnage du message.

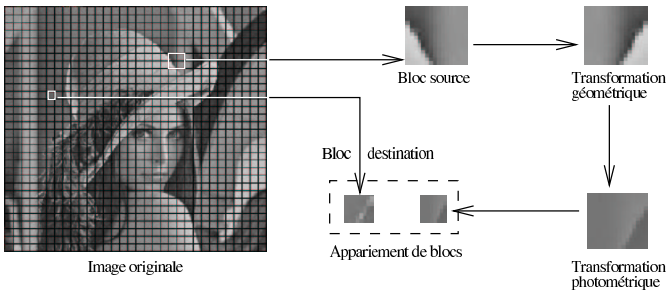


FIG. A.1 : Codage fractal des images.

A.1.2 Génération du support

Premièrement, une approximation fractale I_{approx} est calculée à partir de l'image originale I_{orig}. Le support $I_{support}$ correspond à l'image d'erreur, c'est-à-dire la différence signée entre l'image originale et son approximation fractale.

$$I_{support} = I_{orig} - I_{approx} \tag{A.1}$$

A.1.3 Combinaison de la marque au support

La dernière étape du tatouage est la modulation du support $I_{support}$ par la marque W. Cette modulation consiste à mettre à zéro certains pixels du support en fonction de leur signe et du bit à coder. Pour des raisons de visibilité seuls les pixels de $I_{support}$ de faible valeur sont marqués. Enfin, le support modulé $\hat{I}_{support}$ est ajouté à l'approximation fractale I_{approx} pour obtenir l'image tatouée $I_{tatouée}$:

$$I_{tatouée} = I_{approx} + \hat{I}_{support} \tag{A.2}$$

A.2 Extraction du tatouage

L'extraction du tatouage est analogue à son enfouissement. D'abord l'approximation fractale de l'image tatouée est calculée ce qui permet de générer une image support proche de l'image support originale. Ensuite le support est décodé d'après les règles de modulation (e.g. un pixel positif devrait représenter un bit à 1 et un pixel négatif un bit à 0). Un point important est que la plupart des opérations géométriques sur l'image tatouée sont transmises à l'image support : la marque n'est pas perdue mais la séquence binaire pseudo-aléatoire d'encryptage soit être repositionnée correctement par rapport à l'image support avant d'appliquer le XOR. A cet effet des bits supplémentaires, appelés bits de resynchronisation, sont ajoutés au message utile pour permettre une resynchronisation en mode aveugle des échantillons, et cela suivant deux procédures : l'une pour gérer les déformations géométriques globales (rotation et changement d'échelle), basée sur les propriété de la transformée de Fourier des

signaux périodiques (que représente l'ensemble des bits de synchronisation) et sur la transformée de Hough, et l'autre pour gérer les déformation géométriques locales par un algorithme d'appariement de blocs. Le tatouage peut alors être décrypté et le message reconstruit.

Troisième partie

Stéganographie

Chapitre 6

Dissimulation d'un modèle géométrique dans une image de texture

6.1 Motivation

La stéganographie a pour but de cacher de l'information dans un document avec le même compromis que le tatouage en termes de capacité, visibilité et robustesse. La différence avec le tatouage est que le but n'est pas de marquer un document pour le protéger, mais d'utiliser le document comme moyen pour transmettre une information supplémentaire [74, 17].

Il n'y a a priori pas de restrictions sur le type de documents qui peuvent servir de support pour cacher de l'information. Parmis les documents multimédia il s'agit surtout d'images [3], de vidéos [19] ou de flux audio [20].

Il n'y a également pas de restriction sur le type des données qui peuvent être dissimulées, mais généralement ce qui importe c'est de savoir quelle quantité d'information on peut dissimuler, libre à l'utilisateur d'utiliser ensuite cette capacité pour cacher un logo, un texte, un message audio ou autre.

Certains algorithmes ont néanmoins des applications privilégiées. Dans [20] on propose de dissimuler deux bandes audio supplémentaires dans un signal audio stéréo. Les auteurs de [81, 83] proposent de dissimuler une image dans une image ou une vidéo dans une vidéo. De même dans [100] il s'agit de cacher une sous-bande d'une image dans l'image elle-même à des fins de compression. Plus exactement l'image est séparée en deux bandes (par exemple hautes et basses fréquences) et le résidu (e.g. hautes fréquences) est dissimulé dans la bande principale (e.g. basses fréquences).

Dans ces algorithmes conçus spécialement pour enfouir un type d'information particulier, il peut y avoir une cohérence spatiale (ou temporelle) entre l'information cachée et l'information support. C'est par exemple le cas dans [20] où la bande son supplémentaire est cachée de manière synchronisée avec la bande son support, de sorte qu'un segment de la bande son support contient

cachée l'information de la bande son supplémentaire correspondant au même intervalle de temps.

Par contre dans l'algorithme de [100], qui sépare une image en deux sous-bandes, il n'y a pas nécessairement corrélation spatiale entre la bande cachée et la bande support, c'est-à-dire que l'information cachée dans un bloc donné de la bande support n'est pas forcément l'information manquante correspondant à ce même bloc. En fait cela dépend de la manière dont est compressée la bande à cacher (car elle est compressée) et d'autres détails qui ne sont pas précisés.

Outre l'éventuelle corrélation spatiale entre le support et l'information cachée il peut y avoir une corrélation en termes de dégradation, autrement dit l'information cachée et récupérée peut être d'autant plus dégradée que l'information support a été dégradée. Un algorithme pour cacher une image dans une image ayant cette propriété a récemment été développé [81], mais il n'a pas la propriété de synchronisation spatiale entre l'image support l'image cachée, bien que celle-ci puisse vraisemblablement être assurée en modifiant légèrement l'algorithme.

Dans ce chapitre nous proposons un algorithme ayant les deux propriétés précédentes, c'est-à-dire synchronisation spatiale entre le support et l'information cachée, et dégradation progressive de l'information cachée en fonction de la dégradation de l'information hôte. Et surtout, nous proposons une application qui justifie qu'on souhaite développer ces deux propriétés.

Dans cette application l'information support est une image de texture associée à un objet 3D texturé, et l'information cachée est la géométrie de l'objet 3D. Le but est de transmettre un objet 3D complet, géométrie et texture, en n'envoyant qu'une simple image dans un format standard quelconque. Après extraction de la géométrie dans l'image, l'image peut être plaquée sur la géométrie pour obtenir l'objet 3D texturé complet (voir figure 6.1).

Comme on le voit, l'information cachée est liée à l'information support dans le sens où un point de la géométrie correspond à un point de l'image de texture. Nous voulons préserver cette relation lors de la dissimulation de la géométrie dans la texture, de sorte qu'un bloc de la texture cache la géométrie de l'objet associée à ce bloc.

En outre on veut que la géométrie extraite de l'image ne soit pas brutalement affectée si l'image subit une dégradation "raisonnable" (e.g. compression JPEG). On veut au contraire que la géométrie devienne de plus en plus grossière ou imprécise en rapport avec la dégradation de l'image.

On peut considérer que la géométrie constitue une bande supplémentaire de l'image et qu'on veut réduire de un le nombre de bandes qui constituent l'objet texturé, tout en faisant en sorte que les bandes purement image ne soient pas (ou peu) visuellement affectées et tout en assurant la corrélation spatiale et en terme de qualité entre les bandes d'image et la bande géométrique cachée.

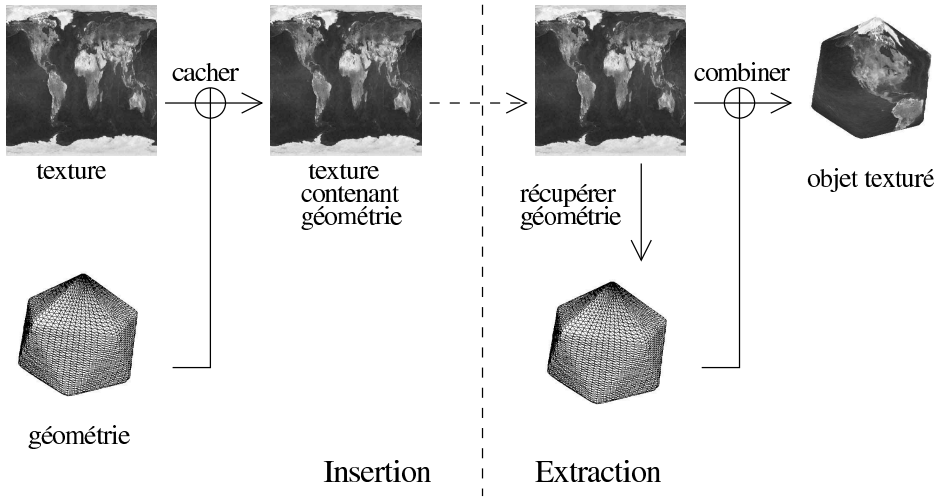


FIG. 6.1 : Utilisation de l'image de texture d'un objet 3D pour véhiculer sa géométrie de manière transparente.

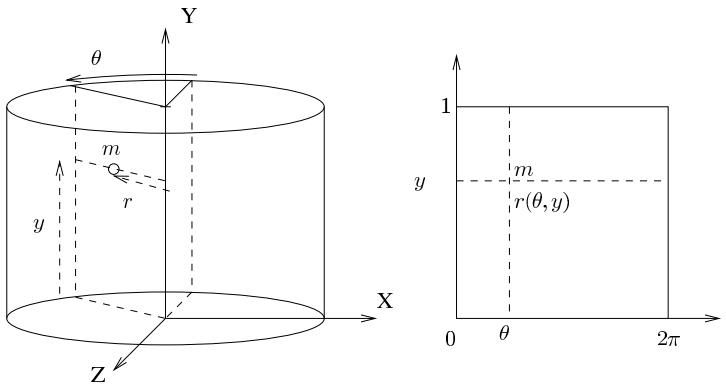
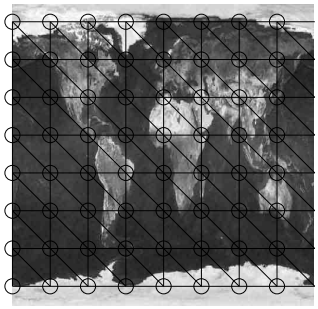


FIG. 6.2 : Passage des coordonnées cartésiennes (X, Y, Z) aux coordonnées cylindriques (θ, y, r) et à la carte de profondeur bidimensionnelle $r(\theta, y)$.

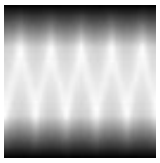
6.2 Représentation des données

Parmis les différentes représentations géométriques possibles nous avons adopté la représentation sous forme de carte de profondeur cylindrique (Fig. 6.2). On suppose également que l'image de texture est une carte de texture cylindrique synchronisée avec la carte de profondeur. Etant donné un objet 3D quelconque on peut aisément construire la carte de profondeur cylindrique et la texture cylindrique en considérant un axe "vertical" qui traverse l'objet et en projetant la surface de l'objet sur un cylindre autour de cet axe. Il y a cependant une restriction sur la forme des objets qui peuvent être décrits ainsi.

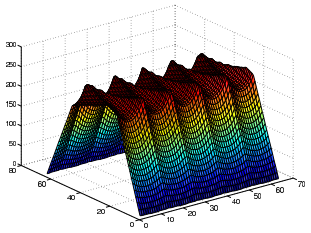
Le fait de représenter un objet 3D par une texture et une carte de profondeur définis dans le même repère cylindrique permet de considérer l'enfouissement de la géométrie dans la texture de la manière évoquée à la section



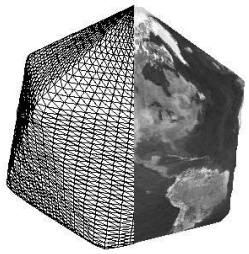
(a) Texture cylindrique.



(b) Carte de profondeur.



(c) Carte de profondeur.



(d) Objet 3D texturé.

FIG. 6.3 : (a) Points d'échantillonnage de la profondeur (cercles) et triangulation de la géométrie associée. En réalité il y a 64×64 points d´échantillonnage. Il n'y a pas de points d'échantillonnage sur le bord droit car ils sont identifiés à ceux du bord gauche via la topologie cylindrique. Le maillage ne permet pas de prendre en compte les demi-bandes supérieure et inférieure. (b) Profondeur échantillonnée dans le repère cylindrique de la texture. Les pixels blancs représentent des points 3D éloignés de l'axe du repère cylindrique. (c) Carte de profondeur représentée différemment. (d) Objet 3D reconstruit et maillé d'après la carte de profondeur cylindrique.

précédente : c'est-à-dire qu'on a un signal bidimensionnel multi-bande et qu'il s'agit de cacher une bande dans les autres.

6.3 Synchronisation spatiale

Pour assurer la synchronisation spatiale entre la texture et la géométrie il faut faire en sorte qu'un bloc de la carte de profondeur soit caché dans le bloc correspondant de l'image de texture. A la limite, si les deux images ont la même échelle cela signifie cacher un pixel de géométrie dans le pixel de texture correspondant. Ceci n'est évidemment pas possible si on veut que la profondeur ait une certaine résolution (par exemple 8 bits par point). Un compromis serait de remplacer les quatres bits de poids faible d'un pixel de texture par les quatre bits de poids fort du pixel de géométrie correspondant, mais cela introduit une forte distorsion sur l'image de texture, et la résolution de la géométrie n'est

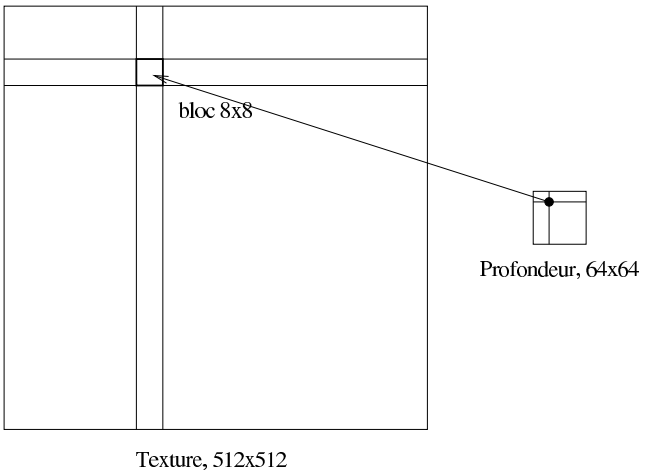


FIG. 6.4 : La carte de profondeur cylindrique étant sous-échantillonnée d'un facteur 8 par rapport à l'image de texture, chaque valeur de profondeur est en correspondance avec un bloc 8×8 de la texture. On pourrait s'attacher à enfouir chaque valeur dans le bloc correspondant pour assurer la cohérence spatiale entre l'information support et l'information cachée.

pas très bonne.

Notons que le problème a une solution si on n'impose pas la cohérence spatiale entre la géométrie et la texture : il suffit de considérer l'ensemble des bits de poids faible de la texture comme étant l'espace d'enfouissement, et de compresser (avec un taux de 1/8) la carte de profondeur par un algorithme de compression d'image pour obtenir un flux binaire qui tient dans l'ensemble bits de poids faible de la texture. C'est ce qui est fait dans [100], si ce n'est que l'espace d'enfouissement est plus sophistiqué que les bits de poids faible de l'image hôte.

Pour avoir la cohérence spatiale nous n'avons d'autre choix que de diminuer la résolution de la carte de profondeur par rapport à la résolution de la texture cylindrique. Ainsi, par un facteur de sous-échantillonnage de 8 dans chaque direction, un pixel de profondeur correspond à un bloc 8×8 de la texture (voir figure 6.4). Si on utilise les bits de poids faible de la texture pour cacher de l'information on dispose alors de 64 bits pour cacher un pixel de profondeur (qui peut être codé sur 8 bits par exemple).

6.4 Synchronisation fréquentielle

Outre la cohérence spatiale entre la texture et la géométrie cachée nous voulons qu'une dégradation de la texture s'accompagne d'une dégradation proportionnée de la géométrie récupérée. Un algorithme simple basé sur les bits de poids faible de la texture, comme évoqué dans la section précédente, n'a pas cette propriété car dès lors que le bruit sur la texture devient supérieur

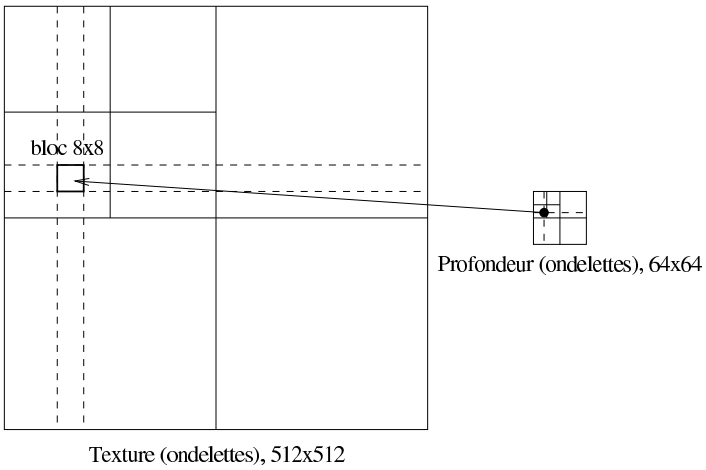


FIG. 6.5 : Après décomposition en ondelettes de la texture et de la carte de profondeur, chaque valeur de profondeur est enfouie dans le bloc de texture correspondant.

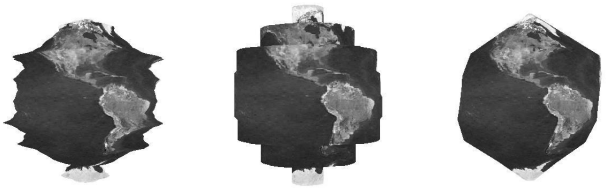
à l'amplitude du bit le moins significatif, la géométrie cachée est entièrement détruite alors même que le bruit est de très faible amplitude.

Un principe pour obtenir la proportionnalité de la dégradation de l'image de texture support et de l'image de profondeur cachée consiste à considérer une décomposition fréquentielle de ces deux images et à cacher les coefficients haute fréquence de la profondeur dans les coefficients haute fréquence de la texture et les coefficients basse fréquence dans les coefficients basse fréquence. Ainsi, la perte des hautes fréquences de l'image de texture implique la perte des hautes fréquences de la géométrie cachée, mais pas la perte des basses fréquences. Ce principe peut être utilisé si on suppose que la dégradation progressive d'une image est une dégradation progressive du spectre à partir des hautes fréquences et vers les basses fréquences. C'est le cas notamment pour des compressions JPEG de facteur de qualité décroissant.

Un moyen d'obtenir une décomposition fréquentielle serait la transformée de Fourier, mais on perdrait alors toute notion de synchronisation spatiale. Un compromis peut être trouvé par l'emploi d'une décomposition DCT par blocs ou par une décomposition spatio-fréquentielle de type ondelettes (voir figure 6.5). Si cela se révèle approprié, on peut également envisager d'utiliser deux types de décomposition différents pour les deux images.

En ce qui nous concerne nous avons utilisé une décomposition en ondelettes des deux images avec le même niveau de profondeur. Pour fixer les choses, nos expériences portent sur une image de texture de taille 512×512, une carte de profondeur associée de taille 64×64, et chaque image subit trois niveaux de décomposition en ondelette, si bien que la composante d'ondelette de base de l'image de texture a une taille de 64×64 et celle de la carte de profondeur a une taille de 8×8.

Plusieurs ondelettes existent en traitement d'image [4] et il est aisé de



(a) Filtre 5/3. (b) Haar. (c) Linéaire.

FIG. 6.6 : Géométrie reconstruite uniquement à partir de la bande de base (8×8 coefficients) de la carte de profondeur cylindrique pour trois bases d'ondelettes différentes.

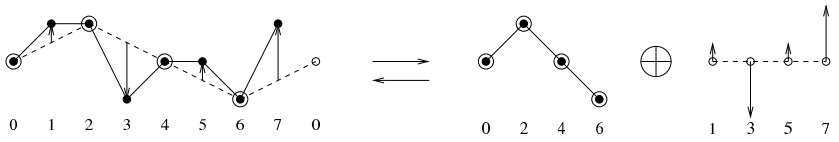


FIG. 6.7 : Ondelettes d'interpolation linéaire.

développer une base d'ondelette spécifique [84]. En ce qui concerne la décomposition de l'image de texture nous avons testé deux bases d'ondelettes : les ondelette de Haar et les ondelettes basées sur le banc de filtres "5/3" retenu pour l'élaboration du standard JPEG 2000 [86].

Par contre pour la décomposition de la géométrie nous avons défini une base d'ondelette spécifique : nous voulons que si les hautes fréquences manquent (ou sont proches de zéro) la profondeur reconstruite soit interpolée linéairement, pour mieux rendre compte des surfaces planes, au lieu d'être interpolée par une fonction de forme vaguement sinusoïdale ou de forme carrée et discontinue (comme pour Haar), ce qui engendrerait des ondulations dans la forme de l'objet ou des arêtes vives là où il ne devrait pas y en avoir (voir figure 6.6). La formule de décomposition en ondelettes, illustrée par la figure 6.7, est simplissime :

$$\begin{cases} v_{2k} = u_{2k} \\ v_{2k+1} = u_{2k+1} - \frac{1}{2}(u_{2k} + u_{2k+2}) \end{cases} \tag{6.1}$$

où (u_k) est le signal original et (v_k) le signal décomposé. Sur la figure on voit que si les hautes fréquences (symbolisées par les flèches verticales) manquent (i.e. sont mises à zéro), le signal reconstruit est interpolé linéairement entre les points de la bande de base. Pour conditions aux bords on choisit la périodicité. C'est naturellement justifié dans la décomposition en ondelettes horizontale du fait de la topologie cylindrique de la carte de profondeur. Quant au sens vertical on peut adopter la périodicité, la symétrie ou la nullité car la profondeur tend généralement vers zéro en haut *et* en bas. Dans tous les cas on n'introduit donc pas de discontinuité parasite.

6.5 Technique de codage

6.5.1 Décompositions parallèles en ondelettes

De l'image de texture on ne considère pour l'enfouissement que la luminance définie par la formule $Y = 0.3R + 0.59G + 0.11B$, ou $Y = 0.3R + 0.6G + 0.1B$ si on veut simplifier.

On effectue la décomposition en ondelette de la carte de profondeur et de la luminance puis on code une valeur de la décomposition en ondelettes de la carte de profondeur dans le bloc 8×8 correspondant de la décomposition en ondelettes de la luminance (voir figure 6.5).

6.5.2 Codage d'une valeur dans un bloc

Une *valeur à coder* (en l'occurence *à cacher*) peut être considérée comme une valeur entière ou réelle en tant que telle ou comme sa représentation numérique sous forme de chiffres (par exemple bits). Dans le premier cas on peut définir une *valeur scalaire* sur l'*information hôte* (par exemple pour un bloc de pixels, sa moyenne), quantifier cette *valeur scalaire* avec un pas Δ, ramener la *valeur à coder* dans l'intervalle $[0, \Delta[$, et enfin ajouter la valeur à coder à la valeur scalaire hôte quantifiée. L'extraction consiste à re-quantifier la valeur scalaire hôte, à récupérer le résidu de quantification de l'information hôte (qui correspond à l'information cachée) et à le remettre à la bonne échelle.

Dans le second cas, on ne code pas la valeur en tant que telle, mais les bits de sa représentation binaire (lesquels peuvent d'ailleurs être considérés individuellement comme des valeurs à coder au sens précédent).

C'est une solution de ce type que nous avons utilisée. Etant donnée une valeur x de la décomposition en ondelettes de la carte de profondeur, on la quantifie sur 8 bits. L'intervalle de quantification est choisi en fonction de la bande de fréquence où se trouve la valeur. La représentation binaire obtenue est $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$.

On dispose d'un bloc de 8×8 coefficients d'ondelette de luminance pour ces 8 bits. On choisit de cacher un bit b_i dans "le bit de poids faible" d'un coefficient d'ondelette. Concrètement, pour cacher b_i dans le coefficient de luminance y on choisit d'abord un pas δ, lequel dépend de la bande de fréquence où se trouve y et de la distorsion tolérée pour cacher les données. Ensuite on calcule

$$n = \left\lfloor \frac{y}{\delta} \right\rfloor \tag{6.2}$$

Le "bit de poids faible" de y, déterminé par le pas de quantification δ, est le bit de poids faible de n ; on le note b. Le codage de b_i dans y consiste à modifier y pour que son bit de poids faible soit égal à b_i. On pose donc

$$\hat{y} = \begin{cases} (n + \frac{1}{2})\delta & \text{si } b = b_i \\ (n + \frac{1}{2})\delta + \delta & \text{si } b \neq b_i \text{ et } y \geq (n + \frac{1}{2})\delta \\ (n + \frac{1}{2})\delta - \delta & \text{si } b \neq b_i \text{ et } y < (n + \frac{1}{2})\delta \end{cases} \tag{6.3}$$

Notons qu'on modifie y même lorsque $b = b_i$ pour que son bit de poids faible soit le plus stable possible par rapport à de petites variations. De plus lorsque b doit être inversé, on a deux possibilités : diminuer ou augmenter y. On choisit ce qui modifie le moins y.

Comme on dispose de 64 valeurs de luminance pour cacher 8 bits, on peut cacher chaque bit plusieurs fois. On pourrait par exemple cacher chaque bit dans huit valeurs de luminance distinctes. A l'extraction on peut ainsi vérifier que les huits valeurs lues pour un même bit sont égales, on peut aussi faire un vote majoritaire pour déterminer la valeur du bit.

En fait, dans la mesure où une erreur sur les bits de poids fort serait plus grave qu'une erreur sur les bits de poids faible, nous avons décidé d'enfouir les bits de poids fort avec une plus grande redondance. Ainsi on code chacun des bits b_7 et b_6 dans 16 valeurs de luminance distinces, les bits b_5 et b_4 dans 8 valeurs de luminance distinctes, les bits b_3, b_2, b_1 et b_0 dans 4 valeurs distinctes. On peut évidemment imaginer d'autres répartitions.

Après, il s'agit de définir quelles valeurs du bloc de texture 8×8 vont permettre de coder quels bits de l'octet de géométrie. Là encore il y a de nombreuses possibilités. On peut prendre un sous-bloc de valeurs de taille 4×4 pour coder 16 fois le bit b_7 et ainsi de suite pour les autres bits. Mais on peut aussi choisir une répartition où les valeurs choisies pour coder un bit donné sont réparties plus uniformément sur tout le bloc, comme de la manière suivante :

b_7	b_5	b_7	b_5	b_7	b_5	b_7	b_5
b_3	b_6	b_2	b_6	b_3	b_6	b_2	b_6
b_7	b_4	b_7	b_4	b_7	b_4	b_7	b_4
b_1	b_6	b_0	b_6	b_1	b_6	b_0	b_6
b_7	b_5	b_7	b_5	b_7	b_5	b_7	b_5
b_3	b_6	b_2	b_6	b_3	b_6	b_2	b_6
b_7	b_4	b_7	b_4	b_7	b_4	b_7	b_4
b_1	b_6	b_0	b_6	b_1	b_6	b_0	b_6

Par rapport à d'autres répartitions que nous avons testées c'est cette dernière qui s'est révélé la plus "satisfaisante".

Mentionnons un dernier détail sur l'utilisation de la luminance pour coder les données de profondeur. Après avoir modifié les bits de poids faible de la luminance dans sa décomposition en ondelettes, on revient du domaine ondelettes au domaine spatial. Il s'agit alors de transmettre les modifications de la luminance aux bandes de couleur initiales.

On calcule donc la différence $d = \hat{y} - y$ entre une valeur de luminance modifiée et sa valeur originale. Ensuite il faut modifier les valeur des composantes de couleur R, G et B pour traduire la différence de luminance. Il y a deux degrés de liberté sur les accroissement qu'on peut appliquer aux composantes de couleurs : on peut fixer arbitrairement les accroissements sur deux composantes et trouver un accroissement de la troisième qui fait que l'accroissement de la luminance est l'accroissement voulu d. Plusieurs solutions naturelles ou

justifiables existent. On peut par exemple multipler R, G et B par un même facteur, pour obtenir l'accroissement de luminance voulu (sauf si R, G et B sont tous nuls). Une autre solution, que nous avons retenue, est de poser

$$\begin{cases} \hat{R} &= R + d \\ \hat{G} &= G + d \\ \hat{B} &= B + d \end{cases} \tag{6.4}$$

Cette solution est en accord avec la première colonne de la matrice de passage d'espaces colorimétriques $YUV \rightarrow RGB$ [15], composée de trois 1, et suppose les composantes U et V constantes.

A ce stade il ne faut pas perdre de vue qu'on revient dans un domaine où les nombres doivent être entiers (R, G et B). Ceci implique qu'il faut arrondir d à une valeur entière. On peut quand même coder d avec une précision de 10^{-1} en tirant parti des degrés de libertés dont on dispose dans la manipulation de R, G et B pour obtenir une luminance donnée. Sans perte de généralité on peut supposer que d est dans l'intervalle $[-0.5, 0.5[$ (après avoir appliqué aux composantes R, G et B la modification correspondant à l'arrondi entier de d).

En considérant l'expression simplifiée de la luminance

$$Y = 0.3R + 0.6G + 0.1B \tag{6.5}$$

on peut tenir compte de la partie fractionnaire de d en modifiant R, G et B de la façon suivante :

intervalle	valeur codée	modification
$d \in [-0.5, -0.45[$	-0.5	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (0, -1, +1)$
$d \in [-0.45, -0.35[$	-0.4	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (-1, 0, -1)$
$d \in [-0.35, -0.25[$	-0.3	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (-1, 0, 0)$
$d \in [-0.25, -0.15[$	-0.2	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (-1, 0, +1)$
$d \in [-0.15, -0.05[$	-0.1	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (0, 0, -1)$
$d \in [-0.05, 0.05[$	0	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (0, 0, 0)$
$d \in [0.05, 0.15[$	0.1	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (0, 0, +1)$
$d \in [0.15, 0.25[$	0.2	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (+1, 0, -1)$
$d \in [0.25, 0.35[$	0.3	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (+1, 0, 0)$
$d \in [0.35, 0.45[$	0.4	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (+1, 0, +1)$
$d \in [0.45, 0.5[$	0.5	$(\hat{R}, \hat{G}, \hat{B}) = (R, G, B) + (0, +1, -1)$

6.6 Gestion des erreurs

Lors de l'extraction des bits (réalisée en regardant simplement le bit de poids faible de $\lfloor \frac{y}{\delta} \rfloor$) il faut gérer les éventuelles erreurs signalées par une non-uniformité de l'ensemble des valeurs extraites pour un bit donné : si les 16 valeurs où est codé le bit b_7 ne donnent pas 16 fois 0 ou 16 fois 1 à l'extraction c'est qu'il y a une erreur. La manière la plus simple de gérer cette erreur est

de réaliser un vote majoritaire, c'est d'ailleurs dans cette optique qu'on a codé les bits de manière redondante.

Une autre solution est de déclarer que la valeur n'est pas valide et de l'ignorer (si possible) pour la reconstruction de la carte de profondeur, ce qui peut éventuellement laisser des trous dans celle-ci. Evidemment on peut définir un seuil pour déclarer qu'un bit est non-valide, par exemple si la différence entre les nombres de 0 et de 1 est inférieure à 8 pour le bit b_7. Si un bit est non-valide on peut invalider l'octet entier ou bien ne pas considérer les bits de poids faible suivants mais néanmoins reconstruire une valeur basée sur les bits de poids fort précédents s'ils sont valides.

Une autre manière d'exploiter la redondance des bits cachés consiste à considérer les bits non plus comme des valeurs binaires mais comme des valeurs continues entre 0 et 1. Ainsi, par défaut tous les bits valent 0.5 mais au fur et à mesure du décodage (à partir des bit de poids fort) les bits acquièrent la valeur 0 ou 1. Si on détecte une erreur dans un bit on peut arrêter le décodage et garder la valeur 0.5 pour ce bit et pour les suivants. La valeur reconstruite à partir de ces bits est simplement donnée par l'expression $\sum_{i=0}^{7} b_i 2^i$, qui peut être évaluée même pour des bits b_i "continus".

Pour exploiter cette idée jusqu'au bout, on peut attribuer à un bit une valeur pondérée par le nombre de 0 et de 1 extraits pour ce bit : si on extrait N fois 1 pour le bit b_7 (et donc $16 - N$ fois 0) on peut lui attribuer la valeur $b_7 = \frac{N}{16}$. Par défaut, la valeur reconstruite pour l'octet de géométrie est donc la valeur moyenne son intervalle de quantification, et par la suite chaque bit tire cette valeur dans sa direction dans la mesure de son "degré de validité".

Dans la section suivante nous verrons que cette méthode a pour caractéristique de permettre une plus grande progressivité de la dégradation des valeurs reconstruites en fonction de la dégradation de l'information support, ce qui est un avantage ou un inconvénient suivant les propriétés recherchées.

On peut imaginer des formules de pondérations non-linéaires, par exemple de la forme :

$$b_7 = \begin{cases} \frac{1}{2} + \frac{1}{2}\left(\frac{N-8}{8}\right)^{\alpha} & \text{si } N > 8 \\ \frac{1}{2} - \frac{1}{2}\left(\frac{8-N}{8}\right)^{\alpha} & \text{si } N < 8 \\ \frac{1}{2} \text{ si } N = 8 \end{cases} \tag{6.6}$$

Pour $\alpha = 0$ on retrouve la formule du vote majoritaire, pour $\alpha = +\infty$ la valeur n'est valide (différente de 0.5) que si tous les bits sont identiques, et pour $\alpha = 1$ il s'agit de la formule linéaire proposée plus haut, toutes les valeurs intermédiaires de α restant également envisageables. En plus de cela on pourrait utiliser une politique différente (en l'occurrence un α différent) pour chacun des b_i. En effet on peut supposer que s'il y a une incertitude sur un bit de poids fort il vaut mieux déclarer la donnée invalide ($\alpha = +\infty$) que de l'utiliser, alors que s'il y a une incertitude sur un bit de poids faible on ne risque pas grand chose à utiliser une valeur de compromis calculée avec $\alpha = 0$ ou 1. Enfin, tous ces paramètres peuvent également dépendre de la bande de fréquence où se trouve

l'information extraite : on peut par exemple décider de déclarer invalides les données basse fréquence peu fiables (qui déterminent l'allure générale de l'objet 3D) alors qu'on se contenterait de minimiser l'impact des données haute fréquence peu fiables mais sans les déclarer invalides. Evidemment toutes ces considérations devraient faire l'objet d'une évaluation expérimentale ou théorique précise.

6.7 Résultats

Les expériences qui suivent donnent un aperçu limité de la pertinence de notre algorithme pour atteindre les objectifs que nous nous sommes fixés (cohérence entre l'information support et l'information cachée). En effet, l'algorithme a de nombreux paramètres sur lesquels on peut agir, et de plus la "cohérence" souhaitée est susceptible d'être interprétée de différentes manières. Trouver expérimentalement les meilleurs paramètres en fonction du critère retenu et de l'ensemble des données de test utilisées nous mènerait trop loin. Nous nous bornons ici à présenter des résultats obtenus en ne faisant varier que quelques paramètres et dans le but de fournir une illustration concrète de notre algorithme.

6.7.1 Protocole expérimental

Notre expérience consiste à cacher la carte de profondeur d'un objet 3D dans son image de texture, à appliquer une compression/décompression JPEG de force variable à la texture tatouée, et à récupérer la géométrie de la texture modifiée par JPEG. On mesure le PSNR entre la carte de profondeur extraite et la carte de profondeur originale, et on compare l'évolution de ce PSNR à l'évolution du PSNR entre la texture compressée et non compressée par JPEG.

La plupart des paramètres utilisés sont ceux donnés en exemple dans les sections 6.4 et 6.5 :

1. Taille d'image de texture de 512×512.
2. Taille de carte de profondeur de 64×64.
3. Décomposition en ondelette à trois niveaux.
4. Filtre "5/3" pour les ondelettes de texture.
5. Ondelettes d'interpolation linéaire pour la profondeur.
6. Quantification des coefficients de profondeur sur 8 bits.
7. Répartition de chacun des 8 bits de profondeur à cacher dans les 64 pixels du bloc de texture correspondant : 16 fois les bits 7 et 6, 8 fois les bits 5 et 4, et 4 fois les quatre bits de poids faible.

Il reste à préciser les tables de quantification pour les coefficients de profondeur et de texture, ainsi que la politique de gestion des erreurs dans la reconstruction des valeurs de profondeur.

La table de quantification des coefficients de texture spécifie les différents pas de quantification utilisés pour les dix bandes de fréquences de la décomposition en ondelette de la luminance. Le pas de quantification définit ce qu'est le "bit de poids faible" à modifier (sachant qu'on ne se préoccupe pas des autres bits, ni même de leur nombre). Les pas de quantification doivent être adaptés à la dynamique de chaque bande de fréquence (qui dépend de la base d'ondelettes utilisée) ainsi qu'à la distorsion qu'on tolère pour chaque bande. La table de quantification définit entièrement la force de marquage, c'est à dire la distorsion moyenne induite par l'enfouissement des données géométriques dans la texture. Plusieurs tables de quantification permettent d'obtenir la même force de marquage : il suffit de compenser la diminution de certains pas de quantification par l'augmentation de certains autres. Mais ceci a une influence sur la forme de la courbe caractéristique de dégradation progressive de l'information cachée en fonction de la dégradation de la texture. Quoi qu'il en soit, dans nos expériences nous avons utilisé un pas de quantification de 1.8 pour la composante de base (dont les valeurs vont de 0 à 255), et de 1.0 pour les autres composantes d'ondelette, ce qui au final donne une force de marquage de 38dB.

Les coefficients d'ondelettes de profondeur doivent également être quantifiés par bande de fréquence et à cet effet on définit une table de quantification. La différence par rapport à la quantification de la texture est que pour la profondeur on a besoin de quantifier sur un certain nombre de bits (et pas seulement définir un bit de poids faible), en l'occurence 8. La valeur du pas de quantification n'a aucune influence sur la force de marquage mais seulement sur la fidélité de l'information cachée par rapport à l'original de la carte de profondeur. Il faut donc trouver le bon compromis entre un pas de quantification petit ou grand. S'il est grand, les coefficients sont quantifiés avec peu de précision mais on peut couvrir tout l'intervalle des valeurs. S'il est petit la plupart des valeurs sont quantifiées avec précision mais on risque d'écrêter les valeurs un peu grandes. Le cas échéant on pourrait augmenter la résolution (nombre de bits) ou effectuer une quantification non-linéaire pour mieux s'adapter à la répartition hautement non-uniforme de certains coefficients d'ondelettes. Là encore la table de quantification utilisée dépend de la base d'ondelette. Sachant qu'on définit les valeurs de profondeur sur une échelle de 0 à 256, on a utilisé la table de quantification 1, .2, .1, .2, .1, .05, .1, .05, .025, .05 pour les coefficients d'ondelette de profondeur (1 étant pour la composante de base). Ces valeurs ont été définies en observant la dynamique des coefficients sur le premier objet de test et en ajoutant une certaine marge.

6.7.2 Premier objet

Pour obtenir les résultats de la figure 6.8 on a utilisé la géométrie polyédrale et la carte du globe terrestre vus précédemment (figure 6.3). On a simplement fait varier la politique de gestion des erreurs basée sur la formule 6.6 en prenant des valeurs extrêmes, et significatives, du paramètre α.

On a pris le facteur de qualité de la compression JPEG comme indice de la dégradation de l'image de texture. La force de marquage est représentée par la ligne pointillée constante à 38dB. L'autre courbe pointillée représente le bruit introduit par la compression JPEG. Enfin les trois autres courbes représentent le bruit sur la carte de profondeur récupérée par rapport à l'original. La limite autour de 50 à 60dB (pour la géométrie) obtenue pour une compression JPEG de facteur de qualité 100 (ce qui est très légèrement différent d'une absence de compression) est due au bruit introduit par la quantification de la carte de profondeur avant son enfouissement dans la texture.

Pour le reste on constate que la dégradation de la profondeur récupérée décroît bien en même temps que la dégradation de la texture, mais avec des caractéristiques différentes dans chacun des trois cas. Tentons une interprétation de la position relative de ces courbes.

$\alpha = +\infty$ catastrophique vers la fin. Ce n'est pas une surprise car ce paramètre signifie que le moindre désaccord dans les valeurs (redondantes) lues pour un bit donné implique que le bit soit purement et simplement "ignoré" (i.e. mis à 0.5). C'est la pire politique de gestion des erreurs.

$\alpha = +\infty$ légèrement meilleur que $\alpha = 0$ au début. Malgré la remarque précédente, il se trouve que la politique $\alpha = +\infty$ n'est pas la pire lorsque la dégradation est faible. A notre sens cela ne peut s'expliquer que si seules les composantes de plus haute fréquence de la texture sont affectées par la compression et que cela détériore complètement (au point de les rendre quasiment aléatoires) les hautes fréquences de la carte de profondeur cachée. Dans ce cas on peut effectivement envisager qu'il soit préférable de mettre ces hautes fréquences à zéro (en mettant chaque bit à 0.5, politique $\alpha = +\infty$) plutôt que d'utiliser le vote majoritaire (politique $\alpha = 0$) qui ne peut que déterminer les bits au hasard si les bits du votes sont eux-mêmes aléatoires.

$\alpha = 0$, évolution par paliers. On constate que le vote majoritaire dans la récupération des erreurs ($\alpha = 0$) donne une courbe décroissant très lentement. Cette robustesse prolongée est certainement due au fait que le vote majoritaire est une bonne solution pour récupérer la plupart du temps la valeur correcte des bits de profondeur. En fait on peut s'attendre à ce que les résultats soient à peu près constants (car le bit correct, 0 ou 1, est récupéré, et non pas une valeur intermédiaire fluctuant autour de 0.5) jusqu'à ce que les bits du vote majoritaire approchent d'une distribution aléatoire, auquel cas il devrait y avoir un décrochage des performances. Le graphe permet d'appuyer (modérément) cette hypothèse car on peut distinguer un premier décrochage qui a lieu dès le début (JPEG 100 à JPEG 95) et qu'on peut associer à la perte du premier niveau de hautes fréquences. Ensuite jusqu'à JPEG 80 la courbe est à peu près constante puis on observe un nouveau mini décrochage jusqu'à JPEG 70, qu'on peut attribuer à la perte du deuxième niveau de hautes fréquences. Suit

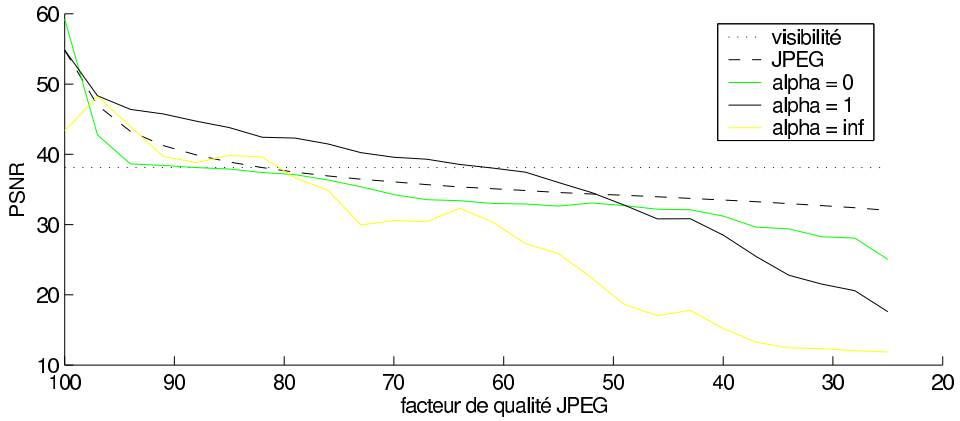
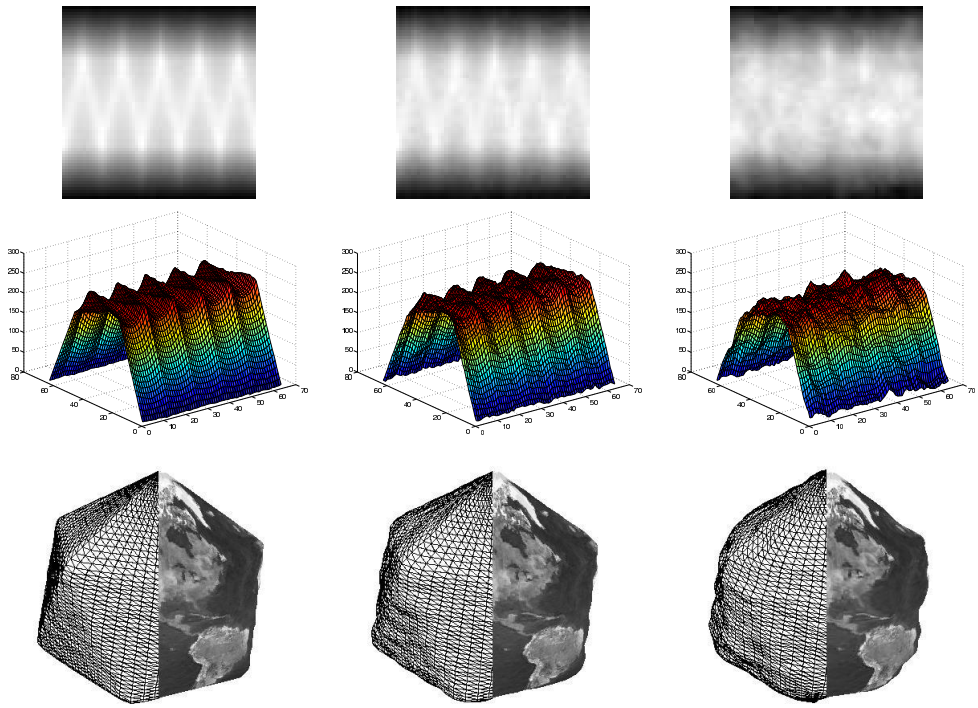


FIG. 6.8 : Dégradation de l'information de profondeur cachée en fonction du facteur de compression JPEG de l'image de texture pour trois valeurs du paramètre de reconstruction α. On a également indiqué la force de marquage (visibilité) et la force de la compression JPEG.

un nouveau palier jusqu'à JPEG 50 et un dernier décrochage correspondant à la perte du dernier niveau de hautes fréquences. Ne reste ensuite que la bande de base.

$\alpha = 1$ meilleur que $\alpha = 0$ au début. La solution $\alpha = 1$ décroît plus rapidement que le vote majoritaire mais lui reste meilleure jusqu'à JPEG 50. On peut supposer que jusqu'à JPEG 50 il se trouve toujours certaines fréquences de la profondeur qui sont détruites et que pour ces fréquences, un vote majoritaire sur un ensemble de bits aléatoire fournit un bit aléatoire 0 ou 1 qui présente une plus forte erreur quadratique moyenne qu'un bit déterminé par pondération linéaire et dont la valeur se situe plutôt autour de 0.5. Après JPEG 50 l'ordre des courbes s'inverse, mais il ne s'agit plus de la même échelle : c'est la bande de base qui devient sujette à la correction d'erreur et non plus seulement les autres bandes de fréquence. La bande de base décrit la géométrie dans ses grandes lignes alors que les bandes de plus hautes fréquences ne font que définir des variations locales de relativement faible amplitude. Dans la mesure où l'importante bande de base est récupérable sans trop d'erreur par vote majoritaire il est normal qu'utiliser $\alpha = 1$ soit sous-optimal. Si elle était entièrement détruite, comme les autres fréquences, alors les deux solutions se rejoindraient sans doute à nouveau.

On a représenté figure 6.9 quelques cartes de profondeur reconstruites après compression de l'image de texture ainsi que la géométrie associée. A supposer que la forme initiale est un polyèdre composé d'un nombre limité de faces planes, comme c'est le cas dans cet exemple, et à supposer que la géomé-



JPEG 100 JPEG 85 JPEG 40

FIG. 6.9 : Géométrie extraite (avec $\alpha = 1$) de la texture compressée : carte de profondeur (lignes du haut et du milieu) et aperçu de la géométrie 3D (ligne du bas).

trie n'est pas trop détériorée, un post-traitement possible à l'extraction serait d'identifier les surfaces planes (avec une certaine tolérance) dans la représentation par carte de profondeur puis de remplacer la description sous forme de carte de profondeur par une représentation polyédrale, qui dans le cas présent pourrait conduire à une représentation compacte de la géométrie sous forme de 20 triangles.

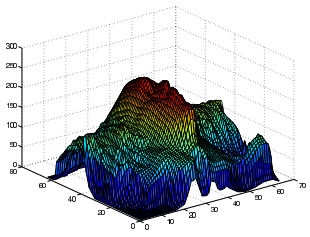
6.7.3 Deuxième objet

Pour voir dans quelle mesure on peut généraliser les résultats précédents nous avons appliqué notre algorithme à un modèle 3D de visage (figure 6.10). Nous avons été d'emblée confronté à deux difficultés mineures concernant cet objet 3D.

La première difficulté est que la profondeur n'est pas définie sur la totalité de la bande cylindrique. Les valeurs manquantes sont arbitrairement mises à zéro, ce qui correspond aux zones noires sur la figure 6.10(c). La difficulté ne vient pas du fait qu'une partie de la géométrie ait une profondeur de zéro, mais du fait qu'il y a "discontinuité" (si on peut parler ainsi pour une carte échantillonnée de manière discrète) entre la zone de profondeur renseignée et la zone

(a) Texture cylindrique 512×512.

(b) Carte de profondeur 64×64.



(c) Carte de profondeur.

(d) Objet 3D texturé.

FIG. 6.10 : Deuxième objet de test.

de profondeur nulle. Ces points de discontinuité entraînent des valeurs anormalement élevées dans les hautes fréquences de la décomposition en ondelette, et ces valeurs sont sévèrement écrétées par la quantification sur 8 bits. Ceci biaise la valeur du PSNR entre la géométrie initiale et la géométrie extraite. Pour cette raison on a modifié légèrement la carte de profondeur avant l'expérience en calculant la décomposition en ondelettes, en écrétant les valeurs trop grandes, puis en effectuant la transformée inverse (mais le tout sans effectuer de quantification). Dans le domaine spatial original cela a pour effet de flouter un peu les contours de discontinuité.

La seconde difficulté vient de ce que la texture n'est pas définie là où la profondeur ne l'est pas non plus. Comme pour la profondeur on la complète par la valeur zéro, en l'occurence la couleur noire. Il y a également d'autres zones de la texture qui dans ce cas particulier sont naturellement noires (cheveux à l'arrière de la tête). Les pixels de couleur noire posent problème pour cacher de l'information. En effet, après décomposition en ondelette, quantification et modification des bits de poids de faible, puis recomposition de la texture, les pixels se trouvent modifiés d'une valeur positive ou négative dont le signe est imprévisible a priori. Si la modification est négative alors que la couleur est noire, on est obligé de la tronquer à zéro : la modification est donc ignorée. Il est possible que cela n'ait pas de conséquence si l'erreur ainsi commise sur certains coefficients de texture reste en dessous du pas de quantification de

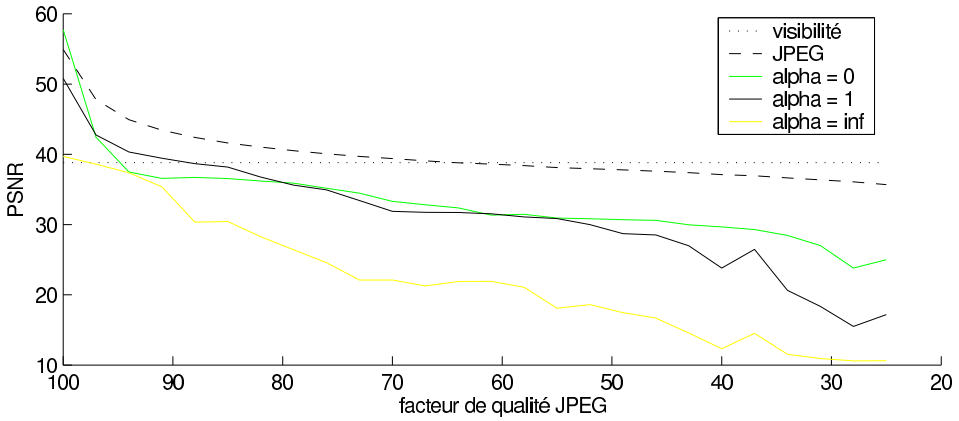
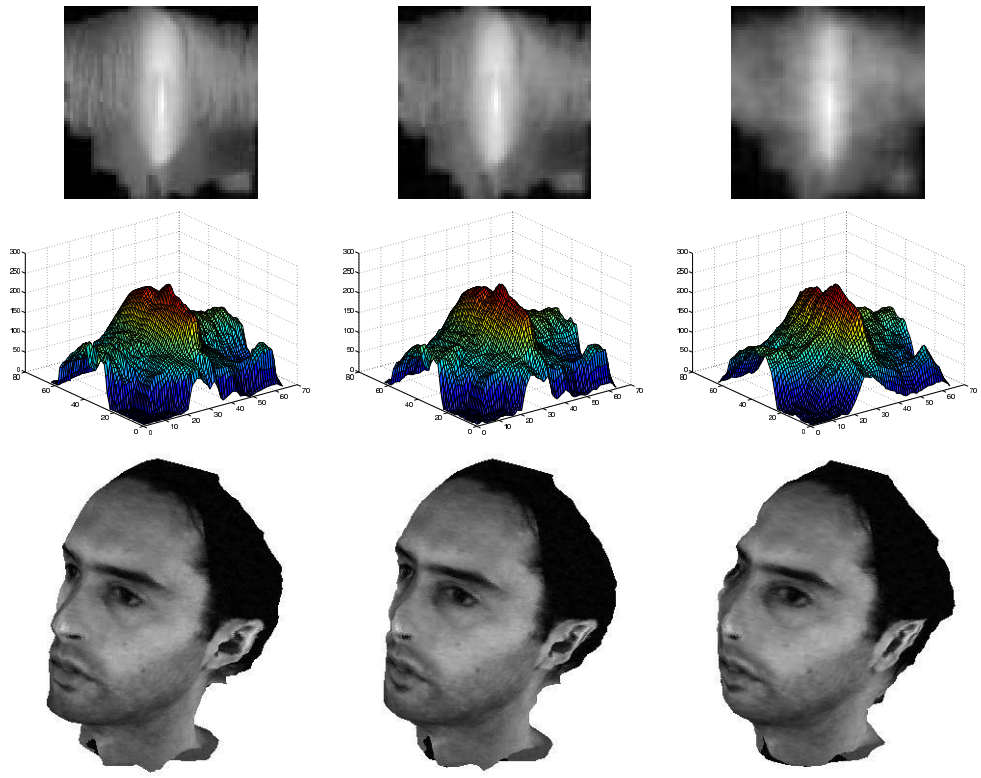


FIG. 6.11 : Dégradation de l'information de profondeur cachée en fonction du facteur de compression JPEG de l'image de texture pour trois valeurs du paramètre de reconstruction α. On a également indiqué la force de marquage (visibilité) et la force de la compression JPEG. Les données utilisées sont celles de la figure 6.9.

la texture, ou si l'erreur est rattrapée par le procédé de correction d'erreur (e.g. vote majoritaire). Dans le cas présent beaucoup de pixels sont noirs ce qui entraîne potentiellement beaucoup d'erreurs. Concrètement, si on laisse les choses en l'état les courbes de PSNR de géométrie sont plafonnées par une valeur anormalement basse de l'ordre de 20dB ou 30dB, à cause de la géométrie des zones noires qui est récupérée de manière pratiquement aléatoire. Dans l'optique de pouvoir comparer les résultats des deux objets nous avons donc imposé que l'intensité des pixels de texture soit comprise entre 5 et 250 avant tout autre traitement ce qui permet de coder la géométrie dans la texture par de légères modifications positives ou négatives sans sortir trop souvent de l'intervalle $[0, 255]$.

Les résultats obtenus avec le modèle de visage 3D (figure 6.11) sont un peu différents de ceux obtenus avec le modèle terrestre polyédral. La différence n'est pas telle que l'interprétation des résultats de la première expérience soit vraiment remise en cause, mais cette interprétation doit être plus nuancée pour être appliquée à la seconde expérience. En particulier on constate que contrairement à l'expérience précédente, la courbe $\alpha = 1$ n'est plus sensiblement meilleure que la courbe $\alpha = 0$ pour les taux de dégration moyens et faibles (quoique en restreignant les valeurs de l'image de texture originale à l'intervalle $[10, 245]$ au lieu de $[5, 250]$ nous avons constaté que les courbes se séparent un peu à nouveau). Quant aux paliers de la courbe $\alpha = 0$ il faut peut-être un peu plus d'imagination pour les voir. Enfin, pour terminer la comparaison, on a représenté (figure 6.12) quelques cartes de profondeur reconstruites après compression de l'image de texture.



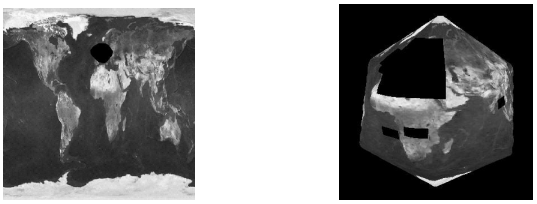
JPEG 100 JPEG 85 JPEG 40

FIG. 6.12 : Géométrie extraite (avec $\alpha = 1$) de la texture compressée : carte de profondeur (ligne du haut et du milieu) et aperçu de la géométrie 3D (ligne du bas).

6.7.4 Localisation des erreurs

Nous venons d'illustrer ce que nous pourrions qualifier de "gestion passive" des erreurs : l'application d'une formule de reconstruction de la géométrie prenant en compte l'ensemble des bits extraits, qu'ils soient cohérents entre eux ou non. La géométrie résultante est d'autant moins fidèle à l'original que la texture a été dégradée.

Une deuxième étape pourrait être une "gestion active" des erreurs dont on estime qu'elle n'ont pas pu être corrigées ou atténuées de manière satisfaisante par la formule de reconstruction. Cela consiste à se poser les questions suivantes et à y répondre : "y a-t-il une très forte probabilité que telle ou telle partie de la géométrie récupérée soit erronée ?" et si oui, "que faire ?". Nous avons déjà donné des éléments de réponse concernant la seconde question : le minimum à faire est de signaler le caractère incertain de la zone géométrique concernée et de laisser l'utilisateur prendre les mesures appropriées, sinon on peut entre autres laisser un trou dans la surface de l'objet, correspondant à la zone supposée erronée, ou alors remplacer la surface erronée par une autre surface s'appuyant sur son contour, par exemple la surface de courbure moyenne nulle en tout



(a) Texture tronquée. (b) Géométrie reconstruite.

FIG. 6.13 : Détection des zones de texture manquantes ou trop altérées pour une extraction correcte de la géométrie. Cohérence spatiale entre texture support et géométrie enfouie.

point (i.e. la surface d'un film d'eau savonneuse s'appuyant sur le contour en question).

Quant à la première question, c'est le problème de définir une probabilité qu'une valeur soit correcte, et de définir un seuil sur cette probabilité pour accepter la valeur. Il est impossible de calculer cette probabilité sans connaître le taux d'erreur binaire des bits récupérés, or cette connaissance dépend des attaques auxquelles peut être soumis le support de l'information cachée. De plus les bits extraits pourraient présenter des erreurs plus ou moins corrélées en fonction de l'attaque subie. Pour des raisons pratiques on peut toujours faire l'hypothèse simplifiée que les bits ont chacun et indépendemment l'un de l'autre une probabilité *a priori* d'être erronés de $\frac{1}{2}$.

Quoi qu'il en soit, tous les paramètres et seuils à choisir doivent l'être indépendemment pour chaque bande de fréquence d'ondelette, car les coefficients de géométrie ont des ordres de grandeurs différents d'une bande à l'autre, et leur impact visuel sur la géométrie est différent d'une bande à l'autre. On peut par exemple décider d'ignorer de considérer les erreurs pour les coefficients de plus haute fréquence en arguant que même s'ils sont erronés cela n'a pas beaucoup d'influence sur la forme et l'apparence de l'objet 3D. Par contre on peut choisir d'être très strict concernant la bande de base sachant que c'est elle qui définit la forme générale de l'objet : si elle n'est déjà pas correcte les autres fréquences peuvent n'avoir plus aucun intérêt (mais cela dépend de l'application). Pour des raisons analogues, au sein d'une bande donnée on peut choisir de tester uniquement la validité des bits de poid fort mais pas les bits de poids faible.

Enfin, précisons que si une valeur de la bande de base est déclarée erronée, c'est un gros bloc dans le domaine spatial qui est invalidé, même si les coefficients haute fréquence de ce bloc ne sont pas déclarés erronés. Par contre si un cofficient des plus hautes fréquences est erroné, ce n'est qu'un petit bloc qui est invalidé dans le domaine spatial.

La figure 6.13 illustre la possibilité de localiser des zones de géométrie qui ne peuvent être extraites avec fiabilité, ainsi que la propriété de synchronisation spatiale entre la géométrie et la texture : la géométrie manquante correspond à la texture manquante. On remarque quelques blocs de géométrie manquants là

où la texture était intacte (e.g. au sud du continent africain). Ceci est un effet de bord dû à la décomposition en ondelette de la texture : la zone manquante de la texture influe sur des coefficients d'ondelette de texture moyennement distants, ce qui peut affecter l'information de géométrie que ces coefficients distants sont censés coder. L'importance d'utiliser une base d'ondelettes à support aussi compact que possible est donc confirmé.

6.8 Commentaires

Nous avons pris pour prétexte les objets 3D texturés pour développer une méthode de stéganographie répondant à un cahier des charges original : préserver simultanément la synchronisation spatiale et le rapport de qualité entre l'information cachée et l'information support. La méthode présentée considère un objet 3D texturé comme un ensemble de deux images : une image de texture cylindrique et une image de profondeur cylindrique associée. Cette manière de voir les choses est discutable à deux points de vue : d'abord, imposer qu'un objet 3D soit représentable par une carte de profondeur cylindrique est une limitation importante sur la forme possible de l'objet 3D, et de plus l'image de texture doit elle-même être cylindrique, ce qui n'est pas toujours la représentation la plus naturelle ou la plus esthétique ; mais surtout, le fait que le problème se réduise à l'enfouissement d'une image (de profondeur) dans une autre image (de texture) peut faire paraître un peu artificiel le contexte de stéganographie d'objets 3D.

S'il est vrai que l'algorithme présenté se réduit à de la stéganographie image-dans-image, rappelons qu'il est rare de trouver une application où les deux images sont liées entre elles et où on voudrait préserver ce lien : les objets 3D fournissent justement une telle application. Il est clair que notre algorithme pourrait être appliqué à d'autres cas qui se réduisent à de la stéganographie image-dans-image et où il y a un lien entre les deux images : une idée parmi d'autres serait d'enfouir la carte de disparité d'un couple d'images stéréo dans l'une des deux images, ou alors de coder une quatrième bande de couleur (par exemple la transparence) dans un format d'image couleur qui ne peut représenter que trois bandes de couleur.

Si maintenant on considère l'application "stéganographie 3D" sérieusement, et non plus comme un simple prétexte, il serait souhaitable de relacher un peu la contrainte sur la forme des objets 3D (imposée par la représentation cylindrique). Dans [42] est présentée une méthode pour représenter la géométrie de n'importe quel objet 3D (ou presque) par une simple image 2D. Pour cela, la surface de l'objet 3D est paramétrée par les coordonnées 2D de l'image [80], et chaque pixel contient les trois coordonnées (x, y, z) du point correspondant de l'objet 3D, ou, dans une autre version de l'algorithme, les trois coordonnées de la normale en ce point. Cela permettrait d'étendre notre méthode de stéganographie à toutes les formes d'objets, à ceci près que pour une reconstruction parfaite de la topologie de l'objet 3D à partir de l'image 2D, quelques informa-

tions supplémentaires sont nécessaires pour recoller les points de la frontière de l'image 2D entre eux (e.g. dans le cas particulier de la représentation cylindrique le bord gauche est à coller au bord droit).

Le dernier commentaire que nous pourrions faire sur notre méthode de stéganographie concerne les choix techniques effectués. Le principe de décomposition en ondelettes nous a paru naturel pour préserver la relation entre l'image cachée et l'image support, mais d'autres espaces de représentation peuvent être également valides. Ainsi une autre idée simple serait d'utiliser la décomposition DCT par blocs (pas forcément de taille 8×8) de la texture et de la géométrie, de quantifier chaque coefficient DCT sur un certain nombre de bits, de trier les bits de géométrie par ordre d'importance, de trier les bits de texture par ordre de robustesse, et de coder chaque bit de géométrie dans le bit de texture correspondant. Des types de décomposition différents pour la texture et la géométrie pourraient aussi être utilisés.

Conclusions et perspectives

8.1 Tatouage d'objets 3D basé sur la géométrie

Le postulat de départ de cette thèse était que l'état de l'art du tatouage d'objets 3D avait pour but de protéger les données informatiques 3D mais pas leurs représentations visuelles. L'étude préalable de l'ensemble de ces algorithmes nous a permis de clarifier certaines notions utiles pour leur classification.

Tous ces algorithmes enfouissent le tatouage à un niveau plus ou moins abstrait de la représentation géométrique. Le niveau le plus abstrait est la forme intrinsèque de l'objet 3D, un niveau intermédiaire est le type de représentation mathématique utilisé (paramétrique, maillage, etc.), et le niveau le plus bas est le format de fichier ou la manière de coder les données. Si on considère que ce qui doit être protégé c'est la forme intrinsèque de l'objet 3D, alors le tatouage est d'autant plus robuste qu'il est effectué à un "haut niveau". Le tatouage à un "bas niveau" est approprié pour des applications sans robustesse ou sans notion de sécurité, telle que la stéganographie.

L'obervation de certains algorithmes [76, 72] nous a fait remarquer que tous les algorithmes qui modifient la géométrie (même si leur but n'est pas de tatouer la forme intrinsèque mais simplement une représentation mathématique donnée) ont le potentiel de tatouer la forme intrinsèque si on accepte qu'ils deviennent non-aveugles et si on leur ajoute une opération de recalage/rééchantillonnage par rapport à l'objet original (au cas où l'objet tatoué ait subi une conversion de format). Inversement, les algorithmes qui se veulent robustes aux conversions de format grace à l'opération de recalage/rééchantillonnage, mais qui sont de ce fait non-aveugles, peuvent être réduits à une utilisation en mode aveugle mais non-robuste aux conversions de format.

Il y'a également des algorithmes [59] qui se veulent à la fois aveugles et robustes aux conversions de format (ou du moins aux remaillages). Le moyen d'y parvenir est d'utiliser comme espace d'enfouissement un invariant géométrique, calculable quel que soit le mode de représentation de l'objet. Un tel invariant, utilisé dans [59] est l'image gaussienne généralisée [51] (distribution de la surface d'un objet en fonction de l'orientation de la normale).

Un autre moyen de trouver des algorithmes aveugles de tatouage de la forme intrinsèque serait de définir une représentation informatique de référence

de n'importe quel objet 3D (voxels ou carte de profondeur orthographique par exemple). Pour l'insertion du tatouage il suffirait de convertir l'objet dans ce format, de tatouer, puis de revenir au format initial, et de même pour l'extraction du tatouage. Une telle méthode serait applicable pourvu que la représentation informatique "de référence" soit stable par rapport à de petites modifications de la forme de l'objet (induites par le tatouage). Par contre il y a une attaque qu'il serait difficile de contrer : la découpe de l'objet.

8.2 Tatouage d'objets 3D basé sur la texture

Les algorithmes de tatouage basés sur la géométrie permettent de protéger un objet dans la mesure où on a accès aux données 3D tatouées suspectes. Or dans certains cas (par exemple sur des pages web) il est plus facile de repérer et de récupérer des images 2D suspectes d'objets 3D protégés que les fichiers 3D eux-mêmes. Les méthodes précédentes ne permettent pas d'extraire le tatouage à partir d'une vue 2D de l'objet 3D protégé. L'approche que nous avons développée dans cette thèse enfouit le tatouage dans l'information de texture de l'objet 3D, ce qui suppose qu'il s'agisse d'un objet 3D réaliste ou du moins riche en texture, contrairement aux objets purement géométriques issus de la CAO. Le tatouage peut alors être extrait de vues 2D de l'objet dans lesquelles l'information de texture tatouée est présente, bien que ce soit de manière déformée.

De nombreuses expériences ont permis d'avoir une estimation de la faisabilité et des limites de cette approche. Sur le plan purement technique, on peut cacher une marque de 64 bits dans la texture d'un objet 3D et l'extraire sans erreur à partir d'une vue 2D qui montre entre 10000 et 20000 pixels de texture distincts, ceci en l'absence d'attaques et en utilisant l'algorithme de tatouage d'images fixes Eurémark avec un paramétrage "par défaut". D'autres algorithmes de tatouage d'images fixes pourraient être testés et adaptés au tatouage d'objets 3D.

Le principe de tatouage d'objets 3D basé sur la texture proposé est non-aveugle dans la mesure où il faut transformer la texture observée dans une vue 2D pour l'exprimer dans l'espace de tatouage d'origine (qui est une donnée de l'objet 3D). En outre cet algorithme nécessite une inversion des paramètres de rendu ayant conduit à une vue 2D donnée. Pour cela un algorithme de recalage projectif adapté au recalage d'un objet synthétique texturé avec une vue 2D de synthèse a été proposé. D'autres méthodes pour revenir de la vue 2D à l'image de texture originale pourraient être envisagées, par exemple un recalage non-rigide direct entre la vue 2D et l'image de texture ne faisant pas intervenir le modèle de rendu 2D (en particulier le modèle de projection). A supposer que cela soit faisable, ce serait particulièrement pertinent dans le cas de modèles 3D déformables. Dans tous les cas, l'algorithme resterait non-aveugle. Sans aller chercher si loin, il est encore possible d'améliorer l'algorithme actuel en cherchant quel est l'algorithme de tatouage d'images fixes le plus adapté.

La considération de certaines attaques pourrait invalider notre algorithme.

Nous avons par exemple constaté la baisse de performances due à une simple compression JPEG de la texture ou de la vue 2D. Mais l'attaque la plus radicale consisterait à re-texturer l'objet 3D : si on possède un modèle 3D tatoué et photo-réaliste d'une personne, et si on dispose d'une photo indépendante et non tatouée de la même personne, il serait possible de re-texturer l'objet 3D par cette photo (au moyen de techniques de vision par ordinateur certes sophistiquées) et de se débarasser de l'ancienne texture tatouée. Cette attaque est spécifique au tatouage d'objets 3D basé sur la texture, mais les attaques d'images fixes consistant à "lessiver" ou supprimer le tatouage [78], lorsque cela est possible, auraient le même effet. Dans le cas où ces attaques ne serait pas envisagées, une amélioration de l'algorithme serait de le rendre aveugle.

Une extension intéressante, mais ambitieuse, de nos travaux serait la protection des vues 2D d'objets 3D sans texture. On pourrait s'appuyer pour cela sur les contours apparents des objets 3D ou sur l'illumination synthétique des surfaces délimitées par les contours apparents. Evidemment cela n'aurait alors plus rien à voir avec du "tatouage basé sur la texture", mais l'idée sous-jacente de protéger l'utilisation de l'objet plutôt que l'objet lui-même, cette idée demeure.

8.3 Stéganographie

Le tatouage ne se limitant pas aux aspects de protection des droits d'auteur et de sécurité, nous avons proposé un algorithme de stéganographie dont le but est de dissimuler la géométrie d'un objet 3D dans l'image de texture qui lui est associée. Ceci permet de véhiculer une information de géométrie de manière transparente dans un format d'image standard.

Dans cette application de stéganographie, l'information cachée et l'information support sont considérées comme deux bandes (respectivement la géométrie et la texture) d'un même signal. Nous voulons conserver le lien entre ces deux bandes en terme de synchronisation spatiale et en même nous voulons que lorsque l'information support est dégradée, l'information cachée subisse une dégradation progressive et proportionnelle.

Ces contraintes originales ainsi que la solution proposée pourraient être pertinentes dans d'autres contextes que les objets 3D texturés, c'est à dire dans tous les cas où l'information cachée et l'information support peuvent être considérées comme deux bandes synchronisées d'un même signal.

Bibliographie

[1] Cyberware home page. `http://www.cyberware.com/`.

[2] P. Abel, D. Loisel, J.-P. Paris, and C. Russo Dos Santos. Automated construction of dynamic 3D metaphoric worlds for network management information visualization. In *SPIE Electronic Imaging*, San Jose, California, January 2000.

[3] N.K. Adbulaziz and K.K. Pang. Robust data hiding for images. In *International Conference on Communication Technology*, 2000.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Debauchies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2), April 1992.

[5] M. Barni and F. Bartolini. Data hiding for fighting piracy. *IEEE Signal Processing Magazine*, 21(2) :28–39, March 2004.

[6] J. Barron and R. Klette. Quantitative color optical flow. In *International Conference on Pattern Recognition*, volume 4, pages 251–255, August 2002.

[7] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performances of optical flow techniques. In *IEEE International Conference on Pattern Recognition*, 1992.

[8] R. Basri and D. W. Jacobs. Lambertian reflectance properties and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2), February 2003.

[9] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27 :433–466, September 1995.

[10] F. Bellifemine, M. Celidonio, A. Chimienti, G. Marcone, and R. Picco. Subpixel accuracy in motion estimation for video coding. *SAIEE Transactions, Special Issue on Signal Processing and Information Theory*, 84(2), 1993.

[11] O. Benedens. Geometry-based watermarking of 3D models. *IEEE Computer Graphics and Applications*, 19(1) :46–55, 1999.

[12] O. Benedens. Two high capacity methods for embedding public watermarks into 3D polygonal models. In *ACM Multimedia and Security Workshop*, pages 95–99, Orlando, Florida, 1999.

[13] O. Benedens. Watermarking of 3D polygon based models with robustness against mesh simplification. In *SPIE Security and Watermarking of Multimedia Content*, pages 329–340, 1999.

[14] O. Benedens. Affine invariant watermarks for 3D polygonal and NURBS based models. In *Third Information Security Workshop*, Wollongong, Australia, December 2000.

[15] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards*. Kluwer Academic Publishers, second edition, 1997.

[16] B. Bollabás. *Modern Graph Theory*. Springer, 1998.

[17] S. Cacciaguerra and S. Ferreti. Data hiding : Steganography and copyright marking.

[18] F. Cayre, P. Rondao-Alface, F. Schmitt, B. Macq, and H. Maître. Application of spectral decomposition to compression and watermarking of 3D triangle mesh geometry. *Image Communications - Special issue on Image Security*, 18(4) :309–319, April 2003.

[19] J.J. Chae and B.S. Manjunath. Data hiding in video. In *IEEE International Conference on Image Processing*, 1999.

[20] J. Chou, K. Ramchamdran, D. Sachs, and D. Jones. Audio data hiding with application to surround sound. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[21] I. J. Cox, J. Killian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 1997.

[22] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2002.

[23] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung. Resolving rightful ownerships with invisible watermarking techniques : Limitations, attacks, and implications. *IEEE Journal on Selected Areas in Communications*, 16(4) :573–586, 1998.

[24] G.A. Crocker and W.F. Reinke. An editable nonmanifold boundary representation. *IEEE Computer Graphics and Applications*, 11(2) :39–51, March 1991.

[25] C. Q. Davis, Z. Z. Karu, and D. M. Freeman. Equivalence of subpixel motion estimators based on optical flow and block matching. In *International Symposium on Computer Vision*, Coral Gables, Florida, November 1995.

[26] F. Devernay and O. Faugeras. Computing differential properties of 3-D shapes from stereoscopic images without 3-D models. Research Report 2304, INRIA, July 1994.

[27] J. Dozier and J. Frew. Rapid calculation of terrain parameters for radiation modeling from digital elevation data. In *Geoscience and Remote Sensing Symposium*, pages 1769–1774, July 1989.

[28] J.-L. Dugelay, K. Fintzel, and S. Valente. Synthetic natural hybrid video processings for virtual teleconferencing systems. In *Picture Coding Symposium*, Portland, Oregon, April 1999.

[29] J.-L. Dugelay and C. Rey. Method of marking a multimedia document having improved robustness, May 2001.

[30] J.-L. Dugelay and C. Rey. Image watermarking for owner and content authentication. In *ACM Multimedia*, Los Angeles, California, November 2002.

[31] J.-L. Dugelay and S. Roche. Introduction au tatouage d'images. In *Annales des télécommunications*, volume 54 of *9–10*, pages 427–437. Groupe des écoles des télécommunications, September–October 1999.

[32] F. Durand, G. Drettakis, and C. Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics*, April 1999.

[33] Cayre F. and Macq B. Data hiding on 3D triangle meshes. *IEEE Transactions on Signal Processing, Special Issue on Data Hiding and Secure Delivery of Multimedia Content*, 2003.

[34] O. Faugeras. *Three-Dimensional Computer Vision : a Geometric Viewpoint.* MIT Press, 1993.

[35] C. Fornaro and A. Sanna. Public key watermarking for authentication of CSG models. *Computer Aided Design*, 32(12) :727–735, 2000.

[36] H. Foroosh, J. B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Transactions on Image Processing*, 11(3) :188–200, March 2002.

[37] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial meshes. In *ACM SIGGRAPH*, volume 21, pages 372–379, 2002.

[38] D. Garcia, J.-J. Orteu, and M. Devy. Calibrage précis d'un capteur de vision stéréoscopique. In *ORASIS*, Cahors, France, June 2001.

[39] E. Garcia and J.-L. Dugelay. Acquisition à faible coût de modèles 3D réalistes de visages. In *ORASIS*, Cahors, France, June 2001.

[40] M. Garland. QSlim mesh simplification software.

[41] M. Gleicher. Projective registration with difference decomposition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 1997.

[42] X. Gu, S. Gortler, and H. Hoppe. Geometry images. In *ACM SIGGRAPH*, pages 355–361, 2002.

[43] S. Hanai, H. Date, and T. Kishinami. Digital watermarking for 3D polygons using multiresolution wavelet decomposition. In *6th IFIP 5.2 International Workshop on Geometric Modeling : Fundamentals and Applications (GEO-6)*, pages 296–307, Tokyo, Japan, December 1998.

[44] J. Hart. Ray tracing implicit surfaces. *SIGGRAPH Course Notes*, 1993.

[45] T. Harte and A. G. Bors. Watermarking 3D models. In *IEEE International Conference on Image Processing*, Rochester, NY, USA, September 2002.

[46] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 19(6) :580–593, June 1997.

[47] F. Hartung, P. Eisert, and B. Girod. Digital watermarking of MPEG-4 facial animation parameters. *Computers & Graphics*, 22(3), 1998.

[48] H. W. Haussecker and D. J. Fleet. Computing optical flow with physical models of brightness variation. volume 2, pages 760–767, 2000.

[49] H. Hoppe. Progressive meshes. In *ACM SIGGRAPH*, pages 99–108, 1996.

[50] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17 :185–204, 1981.

[51] B.K.P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72 :1671–1686, 1984.

[52] S. Ichikawa, H. Chiyama, and K. Akabane. Redundancy in 3D polygon models and its application to digital signature. *Journal of WSCG*, 10(1) :225–232, 2002.

[53] T. Jebara and A. Pentland. Parametrized structure from motion for 3D adaptative feedback tracking of faces. *IEEE International Conference on Computer Vision and Pattern Recognition*, 1997.

[54] S. Katzenbeisser and F. A. P. Petitcolas. *Information Hiding Techniques for Steganography and Digital Watermarking.* Artech House, 2000.

[55] I. Kitahara, H. Saito, S. Akimichi, T. Ono, Y. Ohta, and T. Kanade. Large-scale virtualized reality. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[56] U.-V. Koc and K. J. R. Liu. Interpolation-free subpixel motion estimation techniques in DCT domain. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(4) :460–487, August 1998.

[57] B. Koh and T. Chen. Progressive browsing of 3D models. In *IEEE Workshop on Multimedia Signal Processing*, Copenhagen, Denmark, September 1999.

[58] A. Koschan, V. Rodehorst, and K. Spiller. Color stereo vision using hierarchical block matching and active color illumination. In *IEEE International Conference on Pattern Recognition*, volume 1, pages 835–839, August 1996.

[59] K.R. Kwon, S.G. Kwon, S.H. Lee, T.S. Kim, and K.I. Lee. Watermarking for 3d polygonal meshes using normal vector distributions of each patch. In *IEEE International Conference on Image Processing*, pages II : 499–502, 2003.

[60] C.-H. Lee and L.-H. Chen. A fast motion estimation algorithm based on the block sum pyramid. *IEEE Transactions on Image Processing*, 6(11) :1587–1591, November 1997.

[61] J. J. Lee, N. I. Cho, and J. W. Kim. Watermarking for 3D NURBS graphic data. In *IEEE International Workshop on MultiMedia Signal Processing*, December 2002.

[62] J. J. Little and A. Verri. Analysis of differential and matching methods for optical flow. In *Workshop on Visual Motion*, pages 173–180, March 1989.

[63] M. R. Luettgen, W. C. Karl, and A S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *IEEE Transactions on Image Processing*, 3(1) :41–64, January 1994.

[64] S. R. Marschner, B. Guenter, and S. Raghupathy. Modeling and rendering for realistic facial animation. In *11th Eurographics Workshop on Rendering*, Brno, Czech Republic, June 2000.

[65] A. Mitiche, R. Grisell, and J.K. Aggarwal. On smoothness of a vector field – application to optical flow. *IEEE PAMI*, 10(6) :943–949, November 1988.

[66] P. Moulin and J.A. O'Sullivan. Information-theoretic analysis of information hiding. *IEEE Transactions on Information Theory*, 49(3) :563–593, March 2003.

[67] R. Ohbuchi and H. Masuda. Managing cad data as a multimedia data type using digital watermarking.

[68] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models. In *ACM Multimedia*, pages 261–272, Seattle, Washington, November 1997.

[69] R. Ohbuchi, H. Masuda, and M. Aono. Geometrical and non-geometrical targets for data embedding in three-dimensional polygonal models. *Computer Communications*, August 1998.

[70] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking multiple object types in three-dimensional models. In *Multimedia and Security Workshop at ACM Multimedia*, Bristol, U.K., September 1998.

[71] R. Ohbuchi, H. Masuda, and M. Aono. A shape-preserving data embedding algorithm for NURBS curves and surfaces. In *Computer Graphics International*, pages 170–177, June 1999.

[72] R. Ohbuchi, A. Mukaiyama, and S. Takahashi. A frequency-domain approach to watermarking 3D shapes. In *Eurographics 2002*, Saarbrucken, Germany, September 2002.

[73] F. Perez-Gonzalez, F. Balado, and J.R.H. Martin. Performance analysis of existing and new methods for data hiding with known-host information in additive channels. *IEEE Transactions on Signal Processing*, 51(4) :960–980, April 2003.

[74] F. A.P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding — a survey. *Proceedings of the IEEE*, 87(7) :1062–1078, July 1999.

[75] F. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *International Conference on Computer Vision*, pages 143–150, Corfu, Greece, September 1999.

[76] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *ACM SIGGRAPH*, Los Angeles, California, August 1999.

[77] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *28th annual conference on Computer graphics and interactive techniques*, pages 117–128. ACM Press, 2001.

[78] C. Rey, G. Doërr, J.-L. Dugelay, and G. Csurka. Toward generic image dewatermarking ? In *IEEE International Conference on Image Processing*, Rochester, NY, September 2002.

[79] S. Romdhani, V. Blanz, and T. Vetter. Face identification by fitting a 3D morphable model using linear shape and texture error functions. In *IEEE European Conference on Computer Vision*, 2002.

[80] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *ACM SIGGRAPH*, pages 409–416, 2001.

[81] K. Solanki, O. Dabeer, B.S. Manjunath, U. Madhow, and S. Chandrasekaran. A joint source-channel coding scheme for image-in-image data hiding. In *IEEE International Conference on Image Processing*, pages II : 743–746, 2003.

[82] H. S. Song, N. I. Cho, and J. W. Kim. Robust watermarking of 3D mesh models. In *IEEE International Workshop on MultiMedia Signal Processing*, December 2002.

[83] M. D. Swanson, B. Zhu, and A. H. Tewfik. Data hiding for video-in-video. In *IEEE International Conference on Image Processing*, volume 2, pages 676–679, Santa Barbara, CA, October 1997.

[84] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes*, pages 15–87, 1996.

[85] G. Taubin, T. Zhang, and G. Golub. Optimal surface smoothing as filter design. Technical Report RC-2040, IBM, March 1996.

[86] D. S. Taubman and M. W. Marcellin. *JPEG 2000*. Kluwer Academic Publishers, 2002.

[87] P. Thévenaz, T. Blu, and M. Unser. Image interpolation and resampling. In I.N. Bankman, editor, *Handbook of Medical Imaging, Processing and Analysis*, chapter 25, pages 393–420. Academic Press, San Diego CA, USA, 2000.

[88] S. Valente and J.-L. Dugelay. A visual analysis/synthesis feedback loop for accurate face tracking. *Image Processing Image Communication*, January 2001.

[89] A. Verri and E. Trucco. Finding the epipole from uncalibrated optical flow. In *IEEE International Conference on Computer Vision*, pages 967–972, 1998.

[90] M. G. Wagner. Robust watermarking of polygonal meshes. In *Geometric Modeling and Processing*, Hong Kong, China, April 2000.

[91] B. Wohlberg and G. de Jager. A review of the fractal image coding literature. *IEEE Transactions on Image Processing*, 8(12) :1716–1729, December 1999.

[92] M. Woo, J. Neider, T. Davis, and D. Shreiner. *The OpenGL programming guide*. Addison-Wesley, third edition, 1999.

[93] J.V. Wyngaerd, L. Van Gool, R. Kock, and M. Proesmans. Invariant-based registration of surface patches. In *Seventh IEEE International Conference on Computer Vision*, volume 1, pages 301–306, September 1999.

[94] J.-B. Xu, L.-M. Po, and C.-K. Cheung. Adaptative motion tracking block matching algorithms for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7) :1025–1029, October 1999.

[95] B.-L. Yeo and M. M. Yeung. Watermarking 3D objects for verification. *IEEE Computer Graphics and Applications*, pages 36–45, January/February 1999.

[96] M. Yeung and B.-L. Yeo. Fragile watermarking of three-dimensional objects. In *IEEE International Conference on Image Processing*, volume 2, pages 442–446, 1998.

[97] K. Yin, Z. Pan, J. Shi, and D. Zhang. Robust mesh watermarking based on multiresolution processing. *Computers & Graphics*, 25 :409–420, 2001.

[98] Z. Yu, H.H.S. Ip, and L.F. Kwok. A robust watermarking scheme for 3D triangular mesh models. *Pattern Recognition*, 36(11) :2603–2614, November 2003.

[99] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11) :1330–1334, 2000.

[100] B. Zhu, M. D. Swanson, and A. H. Tewfik. Image coding by folding. In *IEEE International Conference on Image Processing*, 1997.

Résumé

Tous les algorithmes de tatouage d'objets 3D développés à ce jour ont pour but de protéger soit les données informatiques décrivant la géométrie d'un objet 3D, soit la forme géométrique intrinsèque d'un objet 3D. Partant de l'observation que ces méthodes ne peuvent protéger l'utilisation des objets 3D en terme de représentation visuelle, nous proposons une approche radicalement différente, basée sur la texture. L'idée est de tatouer la texture d'un objet 3D et d'être capable d'extraire le tatouage de n'importe quelle représentation 2D de l'objet. Cette méthode procède par la reconstruction de l'image de texture de l'objet 3D à partir d'une vue 2D donnée. A cet effet les paramètres de rendu doivent être connus ou estimés. Après l'évaluation de notre algorithme en environnement contrôlé nous proposons donc un algorithme de recalage projectif permettant d'estimer avec précision la projection utilisée pour produire une vue 2D donnée. La dernière partie de la thèse présente un algorithme de dissimulation des données géométriques d'un objet 3D dans l'image de texture associée à cet objet. L'originalité de cet algorithme par rapport aux autres algorithmes de stéganographie est que l'information à cacher est intimement liée à l'information support. Nous voulons conserver ce lien de cohérence spatiale, en même temps qu'assurer une dégradation progressive de l'information cachée lorsque l'information support se dégrade. Ceci est réalisé par l'intermédiaire d'une représentation spatio-fréquentielle de la géométrie et de la texture, lesquelles sont toutes deux représentées par une fonction des coordonnées cylindriques.

Abstract

The goal of all current 3D object watermarking algorithms is to protect either the computer data representing an object's geometry or the intrinsic shape of a 3D object. Observing that these methods cannot protect the use of 3D objects in terms of visual representation, we propose a completely different approach, based on the texture of 3D objects. The idea is to watermark the texture of a 3D object and then be able to retrieve the watermark from any 2D representation of the object. This method works by reconstructing the object's texture from a given 2D view of it. For this, the rendering parameters must be known or estimated. After evaluating our algorithm in a controlled environment we thus propose a projective registration algorithm allowing us to accurately estimate the perspective projection used to produce a given 2D view. The last part of this thesis presents an algorithm for hiding the geometry of a 3D object into its texture image. The originality of this data-hiding algorithm is that the hidden information is intimately linked to the host information. We want to preserve this link of spatial coherence as well as ensure that the retrieved data degrades progressively when the host information is degraded. This is achieved through a spatio-frequential representation of geometry and texture, which are both represented as a function of cylindrical coordinates.