Université de Montréal

Reconnaissance de visages à partir de modèles tridimensionnels

par

Etienne Beauchesne

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

décembre 2002

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

Reconnaissance de visages à partir de modèles tridimensionnels

présenté par :

Etienne Beauchesne

a été évalué par un jury composé des personnes suivantes :

Pierre Poulin
(prsident-rapporteur)

Sébastien Roy
(directeur de recherche)

Max Mignotte
(membre du jury)

Mémoire accepté le

RÉSUMÉ

Ce mémoire s'intéresse à deux aspects de la reconnaissance tridimensionnelle de visages : l'ajustement des couleurs pour plusieurs acquisitions 3D ainsi que la reconnaissance proprement dite.

Lorsqu'on fait une acquisition à l'aide d'un scanner à lumière structurée, on obtient un modèle 3D de la forme de l'objet ainsi que sa texture. Dans les cas où on ne peut acquérir toute la région d'intérêt d'un seul coup, on devra fusionner plusieurs scans. Pour ce qui est de la géométrie, il n'est pas difficile de fusionner plusieurs morceaux lorsqu'on a une idée de la position relative des morceaux. Cependant, il est plus difficile de fusionner les textures lorsque celles-ci ont été acquises sous des illuminations différentes ou par des caméras différentes, comme c'est souvent le cas lors d'une acquisition 3D à lumière structurée. La méthode présentée permet de corriger les textures de manière à ce qu'elles soient compatibles, c'est-à-dire qu'elles proviennent d'une seule illumination commune, pour ensuite être fusionnées.

La deuxième partie de ce mémoire s'intéresse à la reconnaissance à partir de visages 3D. Il s'agit donc de reconnaître l'identité d'une personne dont on a acquis une modèle 3D du visage, et ce, grâce à une base de données de modèles 3D. Cette approche "3D-3D" a rarement été utilisée dans la littérature. L'opération de base dans l'approche proposée est le calcul d'une "distance" entre deux modèles 3D. Celle-ci est calculée en alignant deux modèles grâce à une variante de l'*Iterative Closest Point* (ICP), puis en calculant leur dissimilarité. Cette distance entre deux modèles sert ensuite de base dans différents scénarios de reconnaissance.

Mots-clés : vision par ordinateurs, reconnaissance de visages, modèles 3D, correction de textures, illumination, alignement de modèles 3D, couleurs, ICP

ABSTRACT

This thesis concentrates on two aspects of three-dimensional face recognition: the correction of colors for several 3D scans and the process of face recognition by 3D alignment.

When doing an acquisition with a structured-light scanner, we obtain the 3D shape of an object and its texture. In the cases where we cannot acquire all the region of interest in a single scan, we have to merge several scans. It is easy to merge several geometry parts when their relative position is known. However, it is harder to merge the textures of these parts, when they have been acquired under different illuminations or with different cameras, as is often the case for 3D structured light acquisition. The method we present allows to correct the textures so that they become compatible, by "relighting" them under a common illumination, in order to be merged later.

The second part of the thesis concerns recognition from 3D faces. The goal is to recognize someone's identity from a 3D scan of that person's face and a 3D model data base. This "3D-3D" approach has been rarely seen in the litterature. The basic operation in the proposed approach is the calculation of a "distance" between two 3D models. To calculate it, we align the two models with a modified version of Iterative Closest Point (ICP), and then calculate their dissimilarity. This distance is then used to implement different face recognition scenarios.

Key words: computer vision, face recognition, 3D models, texture correction, illumination, 3D models alignment, color, ICP

TABLE DES MATIÈRES

LISTE DES FIGURES

À Jeanne D'Arc,

et à Héloïse,

REMERCIEMENTS

Je souhaite remercier tous ceux qui m'ont apporté le support si nécessaire à la création de ce mémoire.

Chapitre 1

INTRODUCTION

La reconnaissance visuelle des visages a toujours été une activité humaine importante. Et cela pour plusieurs raisons : c'est une composante fondamentale des relations interpersonnelles, du système légal ainsi que de la protection et de la gestion des ressources. On utilise ici le mot ressource au sens très large, incluant l'information (personnelle ou autre), l'argent et les privilèges (tels que l'accès à certains lieux, régions, activités, etc.). Ce qui est nouveau aujourd'hui, c'est que la technologie permet de développer des systèmes automatiques de reconnaissance de visages, lesquels permettent des économies substantielles et une précision accrue. C'est à ces systèmes que nous nous intéresserons dans ce mémoire.

Dans un scénario typique d'utilisation, on demande à un tel système de reconnaître une personne grâce à une acquisition (la *requête*), et celui-ci cherche dans une base de données de *descriptions*, afin de retourner l'identité associée à la meilleure description, le cas échéant, sinon, le système indique qu'il ne reconnaît pas la personne.

La majorité de la recherche en reconnaissance de visages s'est faite sur des images 2D, c'est-à-dire que la requête et les descriptions sont des images 2D. Mais ce n'est qu'une possibilité parmi tant d'autres. Certains algorithmes utilisent comme descriptions des modèles 3D de visages (la forme et la couleur). Parmi ceux-ci, certains prennent comme requêtes des images 2D et d'autre des modèles 3D. Dans ce mémoire, nous nous intéresserons à ce dernier type d'algorithme, lequel on qualifie aussi de "3D-3D".

La reconnaissance "3D-3D" est plus adaptée à des applications de type "contrôle

d'accès", où le sujet à reconnaître se prête volontairement à l'acquisition 3D de son visage. Ce type de reconnaissance ne serait pas adapté, par exemple à une reconnaissance de personnes dans une foule, étant donné que les gens bougent et il peut y avoir des occlusions, etc.

1.1 La problématique

Le fait que nous, êtres humains, n'ayons pas de difficulté à reconnaître les gens ne signifie pas que la tâche soit facilement transposable à un ordinateur. En effet, l'automatisation implique la traduction du processus en une série d'instructions explicites et non ambigües. Or, ce processus est principalement inconscient et nous avons beaucoup de difficulté à l'analyser. On n'a qu'à réfléchir au problème et à faire un peu d'introspection pour comprendre le phénomène.

On comprend plus facilement la difficulté de la tâche quand on sait que les cas typiques de reconnaissance automatique impliquent des milliers ou des dizaines de milliers de personnes.

Une autre manière d'illustrer cette difficulté est d'examiner la situation des gens atteints de prosopagnosie. Ce trouble nerveux fait qu'une personne a beaucoup de difficultés à reconnaître les visages, et parfois aussi les maisons et les voitures, et ce, malgré que sa vision et ses facultés mentales ne soient pas affectées. Voici le témoignage d'une personne qui en est atteinte (traduction libre) :

> La reconnaissance ne se base pas toujours sur le souvenir du visage. Par exemple, je reconnais souvent quelqu'un qui s'éloigne de moi, plus que si je regarde ses caractéristiques. Je regarde la démarche, la posture, les manières, etc. En réalité, les visages sont très complexes, car ils diffèrent par de petites différences de distance, largeur, hauteur, etc. Les visages changent selon le mouvement, l'angle, la variation d'éclairage. Ils sont transformés par l'émotion et/ou par différents états (ébriété, fa-

> tigue, etc.) Un visage individuel n'est pas une image inchangeante sauf si c'est une photo. C'est un accomplissement incroyable que les humains arrivent à reconnaître les visages aussi facilement, étant donné la complexité de la tâche. — Cecilia Burman, *Can't remember faces? Info here*, http://prosopagnosia.homestead.com/ppg2.html

Aussi, même si notre capacité de reconnaissance démontre habituellement une très bonne performance pour l'usage qu'on en fait, il nous arrive de nous tromper sur l'identité : on a vu une personne de loin ou trop rapidement. Il nous arrive aussi de ne pas reconnaître une personne connue : changement de coiffure ou de vêtements, etc.

Malgré les limitations de la reconnaissance humaine, l'idéal d'un système automatique reste la reconnaissance parfaite : aucun faux positif (signaler la reconnaissance d'une personne alors qu'elle n'est pas connue), aucun faux négatif (ne pas reconnaître une personne alors qu'elle est connue) et aucune erreur de classification (bien reconnaître les gens connus). C'est pourquoi on compare les performances des systèmes de reconnaissance à cet idéal.

1.2 Les causes de la variabilité

Afin de mieux comprendre les causes de la variabilité du problème, on peut diviser celles-ci en deux catégories : intrinsèques et extrinsèques.

Les causes intrinsèques sont dues au fait que le visage change continuellement. En effet, la couleur, la forme et les propriétés de réflectance de notre visage ne sont jamais précisément les mêmes, malgré qu'en général elles changent très lentement ou rarement. Voici quelques facteurs qui entraînent ces variations : l'expression du visage, une blessure, une maladie, le vieillissement, le changement de poids, le changement dans la texture ou la couleur de la peau, le contenu en huile de la peau, la présence de corps étrangers (saletés, eau, etc.), le port d'une barbe, d'une moustache

ou de maquillage, etc.

On pourrait réduire l'influence de certains de ces facteurs grâce à des consignes, mais il faut toujours tenir compte de l'acceptation du système par les usagers : la perception des gens et la simplicité de mise en oeuvre jouent un rôle important dans le succès d'un système.

La deuxième catégorie contient les causes extrinsèques. Elles proviennent du fait que l'environnement autour du visage change lui aussi (on inclut dans l'environnement les senseurs utilisés pour la reconnaissance). En effet, on ne voit pas toujours une personne complètement, sous le même angle de vue ou la même illumination, et l'acquisition n'est pas parfaite.

Les systèmes qui nous intéressent sont visuels et cela implique qu'ils font face à une variabilité due à la nature de la lumière. Il faut un contact visuel : on doit au moins voir une partie du visage et cela peut être une partie différente à chaque fois. Ce qui nous amène à une des variabilités les plus significatives pour notre problème : celle due à la *pose*. La pose est une transformation rigide, elle contient donc une rotation (3 degrés de liberté) et une translation (3 degrés de liberté).

Une autre cause significative est que l'illumination peut changer. Par exemple, dans les endroits où il y a un éclairage naturel important, elle sera différente selon le moment de la journée.

Aussi, il peut y avoir de la neige ou d'autres substances gênant la vue sans nécessairement causer d'occlusion : poussière, fumée, précipitations, objets transparents clairs ou colorés. Il pourrait aussi être intéressant de pouvoir reconnaître une personne à partir de sa réflexion sur un miroir ou une surface suffisament spéculaire.

D'autres causes extrinsèques concernent l'imperfection de l'acquisition. Les divers processus d'acquisition impliquent du bruit et des erreurs. Par exemple, la caméra ne perçoit pas le monde directement. Chaque pixel de son capteur compte les photons incidents et est soumis à divers bruits : électrique, statistique (le nombre de photons suit une loi de Poisson), de corrélation (chaque pixel influence ses voisins) et autres.

Cela implique que même deux images consécutives d'une scène où rien n'a changé (incluant l'illumination) ne seront pas égales. La caméra ne perçoit donc jamais parfaitement la même chose.

Nous ne tenterons pas ici de tenir compte de toute cette variabilité, car cela est sûrement au-delà des connaissances actuelles. De toute façon, dans plusieurs cas, il y a des moyens simples et efficaces de contourner les difficultés. Par exemple, si quelqu'un a maigri au point que le système ne le reconnaît plus, on peut simplement mettre à jour son entrée dans la base de données.

1.3 Les étapes de la reconnaissance

À cause de la forme concave du visage, l'acquisition 3D d'un visage nécessite en général plus d'un scanners 3D (dans notre cas : 3 scanners de InSpeck) et procède comme suit. La personne qu'on veut scanner s'assoit sur une chaise comportant un appuie-tête. On fait ensuite l'acquisition avec les 3 scanners, chacun à son tour. Pour chaque scan, cela génère des images de franges ainsi qu'une de texture (la couleur du visage). Ensuite, on peut calculer pour chaque caméra une profondeur pour tous les pixels correspondant au visage et visibles par celle-ci, ce qui donne trois modèles 3D distincts. Pour terminer, on fusionne les trois modèles 3D en un seul, qu'on annote et insère dans la base de données. Pour reconnaître quelqu'un, il suffit de faire une acquisition du visage, puis de comparer le modèle obtenu à un ou à plusieurs descriptions de la base de données, selon le scénario de reconnaissance. Les différents scénarios seront présentés au chapitre 4.

1.4 Organisation du mémoire

La fusion des textures pose des problèmes spécifiques et fera l'objet des chapitres 2 et 3. Le chapitre 2 est une introduction à la correction de textures et le chapitre 3 est un article allant plus en détail.

La reconnaissance 3D-3D fera l'objet des chapitres 4 et 5. Le chapitre 4 est une introduction à la reconnaissance, alors que le chapitre 5 parle plus directement de notre méthode.

Chapitre 2

CORRECTION DE TEXTURES

Ce chapitre est une introduction à la correction de textures, et contient une revue de la littérature pertinente à notre méthode présentée au chapitre suivant. Cette correction est utilisé lorsqu'on fusionne ensemble des modèles partiels d'un objet. Il s'agit de corriger les textures pour qu'elles aillent bien ensemble, sachant qu'elles n'ont pas été prises dans les mêmes conditions ou avec les mêmes caméras.

2.1 Concepts de base

Parlons d'abord de *modèle 3D* d'un objet (Figure 2.1). Ce terme désigne la géométrie de cet objet (sa forme en 3D), sa texture (la couleur) et une correspondance entre les deux (à quel endroit sur la géométrie appliquer quel pixel de la texture). Les géométries, les textures et les correspondances peuvent s'exprimer sous différentes formes.

Nos modèles 3D utilisent une des formes les plus courantes pour la géométrie, soit le maillage de triangles. Chaque triangle est une surface plane délimitée par les *arêtes* sous-tendues par trois *sommets*. Le principe du maillage est simple : de la même manière qu'on peut approximer n'importe quelle courbe en 2D par une série de traits suffisament courts, on peut approximer aussi précisément qu'on le veut une surface 3D par un maillage ayant suffisament de triangles. Le nombre de triangles à utiliser peut dépendre de plusieurs facteurs : la précision désirée, la forme du modèle ou le temps de calcul disponible. Pour ce qui est de la texture, elle est représentée sous forme d'image. La correspondance est simplement une correspondance de chaque point de la géométrie vers un point de l'image représentant la texture.

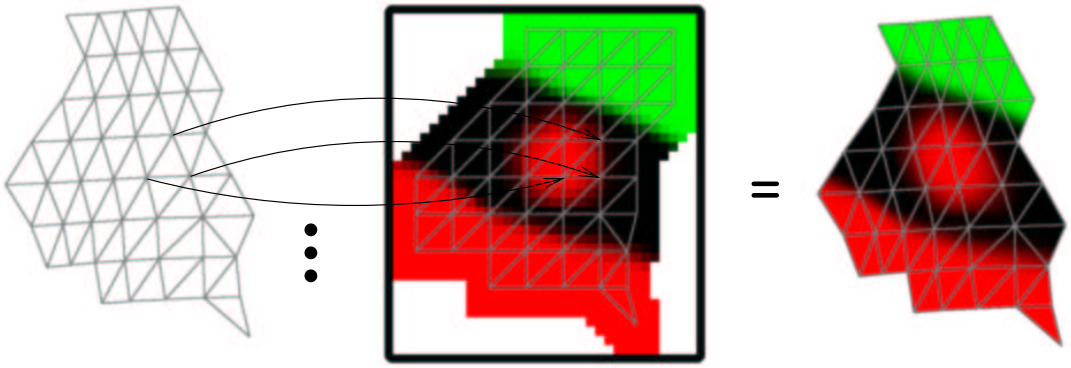

Figure 2.1. Modèle 3D d'un objet par maillage de triangles.

La *normale* à une surface en un point donné est le vecteur unitaire (dont la norme est égale à 1) perpendiculaire à celle-ci en ce point. Comme on a toujours le choix entre la normale et son opposé additif, on pourrait se demander lequel utiliser. En pratique, la convention veut que les normales pointent toutes du même coté de la surface, ce qui définit une surface orientée et facilite certains calculs.

Une *BRDF* (*Bidirectional Reflection Distribution Function*) ou fonction bidirectionnelle de réflection est une représentation des propriétés de réflectance d'une surface. Elle donne la fraction de la lumière incidente qui sera réfléchie. Le terme *bidirectionnelle* réfère au fait qu'elle est fonction de la direction d'incidence de la lumière et de la direction d'observation. Elle est aussi fonction de la longueur d'onde de la lumière, cependant, nous n'en tenons pas compte. Une *surface lambertienne* (aussi appelée *surface parfaitement matte*) est une surface pour laquelle chaque point de la surface apparaît aussi brillant peu importe le point de vue.

2.2 La problématique

La technique que nous avons développée et qui fera l'objet du prochain chapitre tente de résoudre un problème lié à l'*acquisition* de modèles 3D. L'acquisition est le processus permettant de convertir la forme d'un objet réel en modèle 3D.

Ce processus n'est pas parfait, l'idéal serait de pouvoir acquérir parfaitement la réflectance ainsi que la géométrie en une seule acquisition. Or, on ne peut acquérir la réflectance directement, puisque les images disponibles sont une combinaison de la réflectance et de l'illumination ambiante. De plus, on ne peut acquérir le modèle 3D d'un visage en un seul morceau avec des scanners à lumière structurée, parce que seul les points visibles par la caméra peuvent être numérisés. L'acquisition (illustrée à la Figure 2.2) produit donc 3 modèles 3D, un par scanner. Ceci implique qu'il faudra fusionner la géométrie et la texture des modèles partiels.

Nous nous intéresserons ici seulement aux imperfections liées à la texture puisque la fusion de maillages 3D ne présente pas de difficulté. D'ailleurs, celles-ci sont de loin les plus visibles et les plus importantes. Expliquons maintenant plus en détail une acquisition ainsi que les causes des imperfections.

Le matériel d'acquisition est placé comme dans la Figure 2.2 de manière à voir le visage au complet (grâce au positionnement des scanners) et à éviter les *spécularités* (grâce aux écrans devant les sources lumineuses).

Les spécularités sont des régions de l'image où on voit le reflet direct d'une source lumineuse, ce qui indique que la surface a agi un peu à la manière d'un miroir. Souvent une spécularité entraîne une saturation des pixels correspondants, ce qui fait qu'on perd de l'information. Les spécularités compliquent la fusion de textures. On peut les détecter facilement à l'oeil : ce sont habituellement des zones plus pâles que les autres zones qui sont de la même couleur en réalité, et en plus, les spécularités se déplacent avec l'observateur.

Tel que mentionné précédemment, l'acquisition 3D d'un visage est faite grâce à

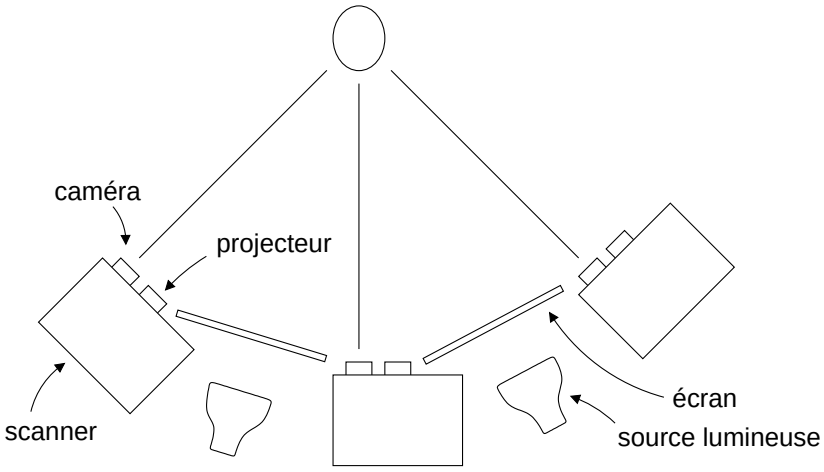

Figure 2.2. Disposition du matériel.

trois scanners 3D à lumière structurée. Elle procède comme suit. On demande à la personne qu'on veut scanner de s'asseoir dans une chaise comportant un appuie-tête, afin de réduire les mouvements de la tête (l'acquisition dure tout de même 0.4s par scanner). Lorsqu'on est prêt, on fait l'acquisition, et les 3 scanners s'activent tour à tour. Pour chaque scanner, on obtient 4 images de franges ainsi qu'une de *texture* (la couleur du visage) comme illustré à la Figure 2.3. Chaque image de franges correspond à un déplacement horizontal différent du pattern de franges. Finalement, pour chaque caméra, le logiciel FAPS de InSpeck calcule une profondeur pour chaque pixel visible par la caméra, ce qui permet de générer la géométrie. De plus, il calcule la correspondance entre la géométrie et la texture, ce qui permet d'obtenir un modèle 3D du visage. Une fois qu'on a les 3 modèles 3D, on peut utiliser un autre logiciel, EM de InSpeck, pour raffiner l'alignement puis les fusionner (géométries et textures). Malgré qu'on s'intéresse ici aux visages, la méthode s'applique aussi à n'importe quels autres objets convexes ou presque convexes.

Le problème principal avec notre procédure se situe vers la fin de celle-ci. On fusionne les textures de plusieurs modèles et ce, malgré que leurs couleurs ne soient pas nécessairement cohérentes. Cette incohérence est due principalement à deux causes.

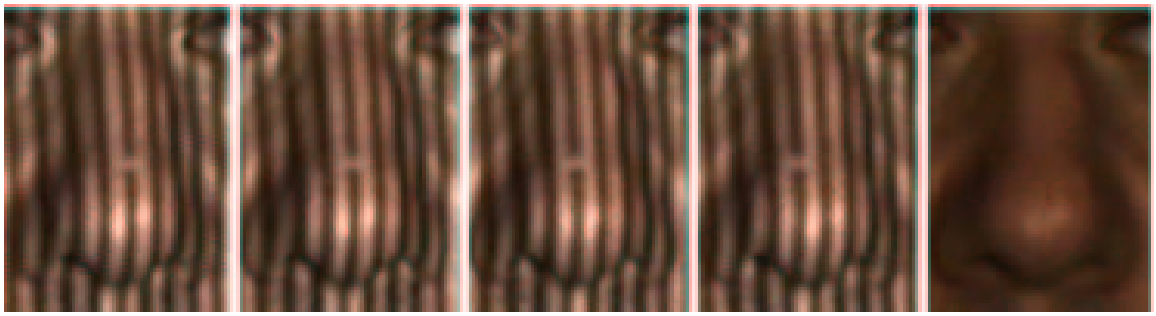

Figure 2.3. Images de sortie du scanner (agrandissement d'une partie des images). On voit, dans l'ordre, les 4 images de franges (elles diffèrent par la position horizontale des franges), suivies de l'image de texture.

La première est que chaque caméra voit les couleurs légèrement différemment, elles ne sont pas parfaitement identiques. La deuxième est que l'éclairage pour chaque acquisition est différent. En effet, l'éclairage ambiant reste constant, mais lorsqu'un scanner acquiert la texture, son projecteur est allumé alors que les autres projecteurs sont fermés. Comme l'éclairage est différent pour chaque acquisition de texture, les couleurs perçues par les caméras sont différentes, et ce suffisament pour être flagrant à l'oeil.

2.3 Solutions suggérées dans la littérature

Posons le problème plus généralement. Supposons que l'objet d'intérêt est fixe et qu'on en acquiert des images par une ou plusieurs caméras. Les points de vue et les illuminations peuvent être différents pour chaque acquisition. Chaque image est utilisée afin de créer une texture et on reçoit la correspondance entre la texture et la géométrie, laquelle est fournie. Le but est de corriger les textures obtenues afin de pouvoir les fusionner. La principale contrainte que nous imposons est le réalisme (les couleurs doivent être cohérentes et ne pas présenter de jonction visible). Le problème est illustré à la Figure 3.1.

Dans le passé, on a souvent vu le problème de la correction de textures (sans la contrainte de réalisme) comme étant celui de fusionner des textures en modifiant celles-ci le moins possible. On parlera indifféremment de correction et de fusion de textures car ces deux opérations sont proches parentes : une fois les textures bien corrigées, il est facile de compléter la fusion.

Par exemple, Milgram [19] a démontré comment la différence entre deux textures peut être distribuée dans le voisinage de la jonction de celles-ci, voir la Figure 2.4. En entrée, on reçoit deux images rectangulaires qui se chevauchent et qui sont parfaitement alignées une relativement à l'autre. Elles ont le même nombre de lignes et sont concordantes verticalement. Dans ce cas, il s'agissait d'images aériennes, mais on peut aussi bien appliquer leur technique à d'autres situations. Premièrement, on effectue un *ajustement d'histogramme d'ordre 0* (Figure 2.4a). C'est-à-dire que pour chaque composante (rouge, vert et bleu), on ajoute une valeur constante de manière à ce que la moyenne de la zone de chevauchement soit égale pour les deux images. Pour la suite, le traitement se fait indépendamment sur chaque ligne horizontale, et seulement le voisinage de la jonction sera modifiée (Figure 2.4c), à gauche de celle-ci on utilise l'image de gauche modifiée et à droite, celle de droite modifiée. Deuxièmement, on détermine à quel endroit sur la ligne on placera la jonction (Figure 2.4b). Et finalement, on ajoute une rampe linéaire (Figure 2.4c) de chaque coté de la jonction de manière à ce que les images se rencontrent à la valeur moyenne des deux pixels les plus proches de la jonction (Figure 2.4d). La pente de la rampe dépend de la largeur du voisinage en question. Il y a des problèmes avec cette approche. Le fait qu'elle dépende de la différence de deux pixels la rend vulnérable au bruit qu'ils contiennent. Aussi, cela crée des lignes horizontales d'intensités légèrement différentes qui sont facilement visibles [7]. Aussi, cette approche ne fonctionne qu'avec des images rectangulaires d'objets plats et où la région de chevauchement est rectangulaire. Dans Milgram [20], quelques améliorations sont décrites.

Rocchini et al. [23] ont développé un algorithme basé sur le partitionnement des

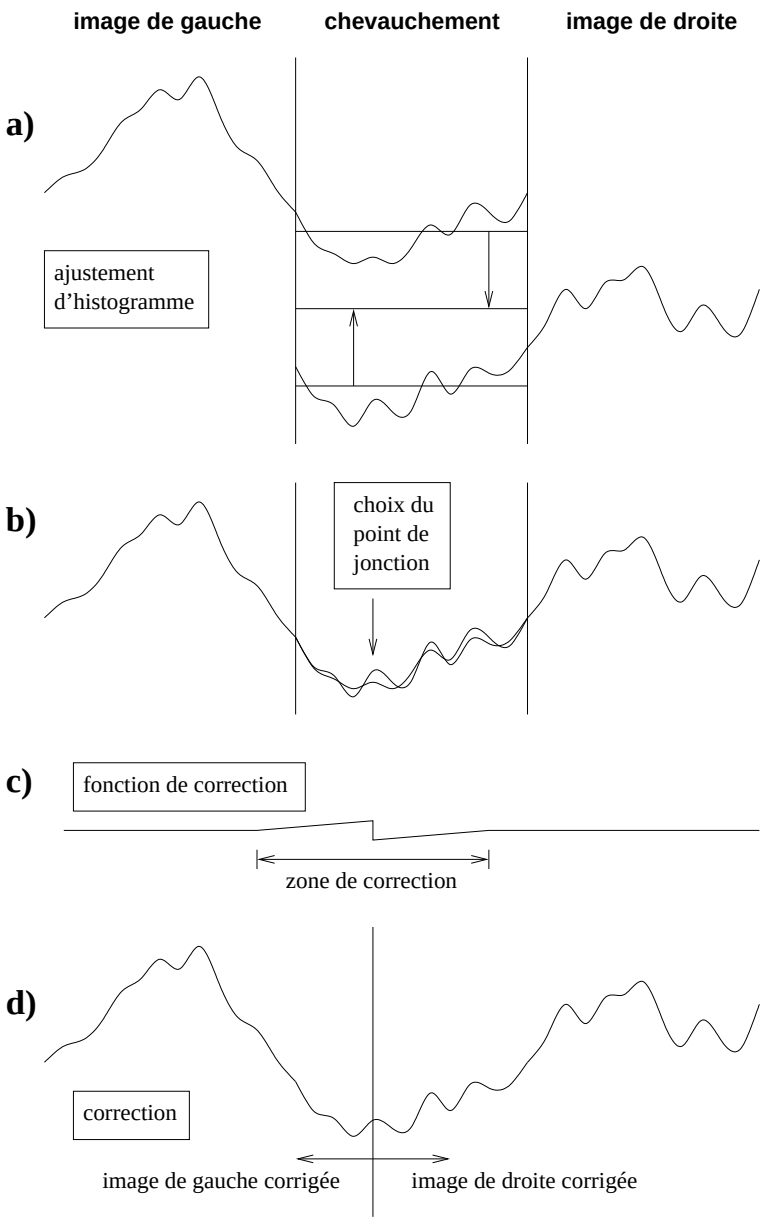

Figure 2.4. Algorithme de Milgram. Pour simplifier le diagramme, on considère deux images contenant une seule ligne et une seule composante. L'axe vertical représente l'intensité.

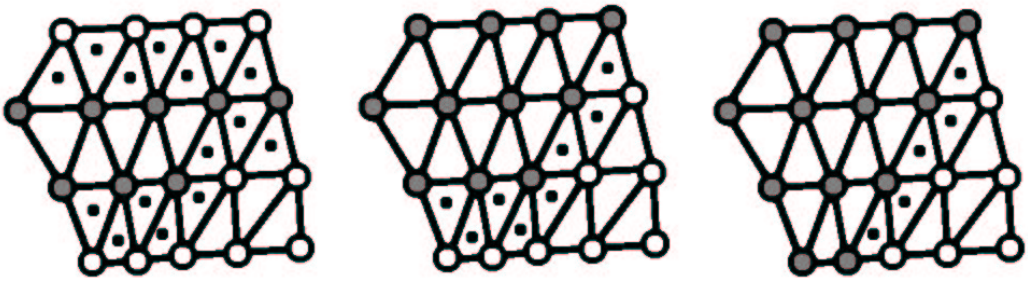

Figure 2.5. Algorithme de Rocchini et al. À gauche : une géométrie avec le partitionnement initial (blanc : texture 1; gris : texture 2). Les triangles-frontières sont marqués par un point. Au centre : après une étape de l'algorithme glouton. À droite : Après 2 étapes, nous arrivons dans un minimum local, on ne peut plus réduire le nombre de triangles-frontières.

sommets d'un maillage de triangles selon la texture de laquelle ils prennent leur couleur. Leur algorithme est illustré à la Figure 2.5. Ce partitionnement crée différentes régions et des frontières entre elles. La première étape est de faire un partitionnement initial selon un critère simple. Le partitionnement est amélioré itérativement grâce à un algorithme glouton afin de réduire le nombre de triangles-frontières, lesquels sont des triangles dont les sommets sont assignés à plus d'une texture. En d'autres mots, on fait toutes les réassignations d'un sommet à une texture qui réduiraient le nombre de triangles-frontières dans l'assignation initiale et on répète cette étape jusqu'à ce qu'on ne puisse plus diminuer ce nombre. Pour calculer la texture fusionnée, on utilise l'interpolation à l'intérieur de chaque triangle par les *coordonnées aréales* du point à interpoler.

Les coordonnées aréales sont les coordonnées barycentriques normalisées pour que leur somme soit 1. Les coordonnées barycentriques d'un point dans un triangle sont les masses qu'on devrait mettre sur chaque sommet du triangle afin que leur centre de gravité (barycentre) soit sur le point en question. On remarque que les coordonnées

barycentriques sont uniques à un facteur près, c'est-à-dire que si on utilise des masses k fois plus grandes (où $k \neq 0$), le barycentre reste le même.

Pour contraster, Pulli et al. [22] assignent plus d'une textures à chaque point de la surface. Ils font une moyenne pondérée sur toutes les régions se chevauchant. Les poids sont calculés en multipliant plusieurs facteurs : la proximité au bord de la texture, le cosinus de l'angle du rayon lumineux avec la normale, ainsi qu'un poids attribué aux trois images qui voient le point en question sous l'angle le plus semblable à la normale.

Certaines approches similaires sont encore plus élaborées : Levoy et al. [15] ont aussi utilisé une moyenne pondérée où les poids sont la confiance en chaque valeur de pixel. Cette confiance tient compte de plusieurs facteurs : l'obliquité de la surface relativement à la lumière, la surface projetée par cette surface par rapport à la caméra, la proximité à la direction miroir (pour supprimer les spécularités), la proximité de la silhouette relativement à la caméra, et celle relativement à la lumière, et finalement la proximité au bord de l'image. Pour empêcher des changements rapides dans la confiance de créer un changement rapide d'une image à l'autre, les confiances sont lissées en utilisant le voisinage sur le maillage. Aussi afin de rester conservateur, la confiance n'est jamais augmentée, seulement réduite.

Pour ce qui est de la moyenne pondérée, il y a des problèmes récurrents. Si la largeur de la transition (de la fonction de mélange) est trop grande (Figure 2.6a), une petite erreur d'alignement peut produire un effet de *double exposition* (où certains éléments apparaissent deux fois), alors que si elle est trop basse, la transition se fera en quelques pixels créant ainsi une jonction (Figure 2.6b).

Burt and Adelson [6] résolvent ces limitations avec la modélisation d'images par splines multirésolutions (*multiresolution image splining*). Ils considèrent le cas où la fonction de mélange $H(x)$ a la forme d'une sigmoïde, la première image a le poids $1-H(x)$ et la deuxième $H(x)$. La largeur de transition est un paramètre fixe de cette fonction, et il spécifie la longueur en pixels que cela prend pour aller de 0 à 1. Ils affir-

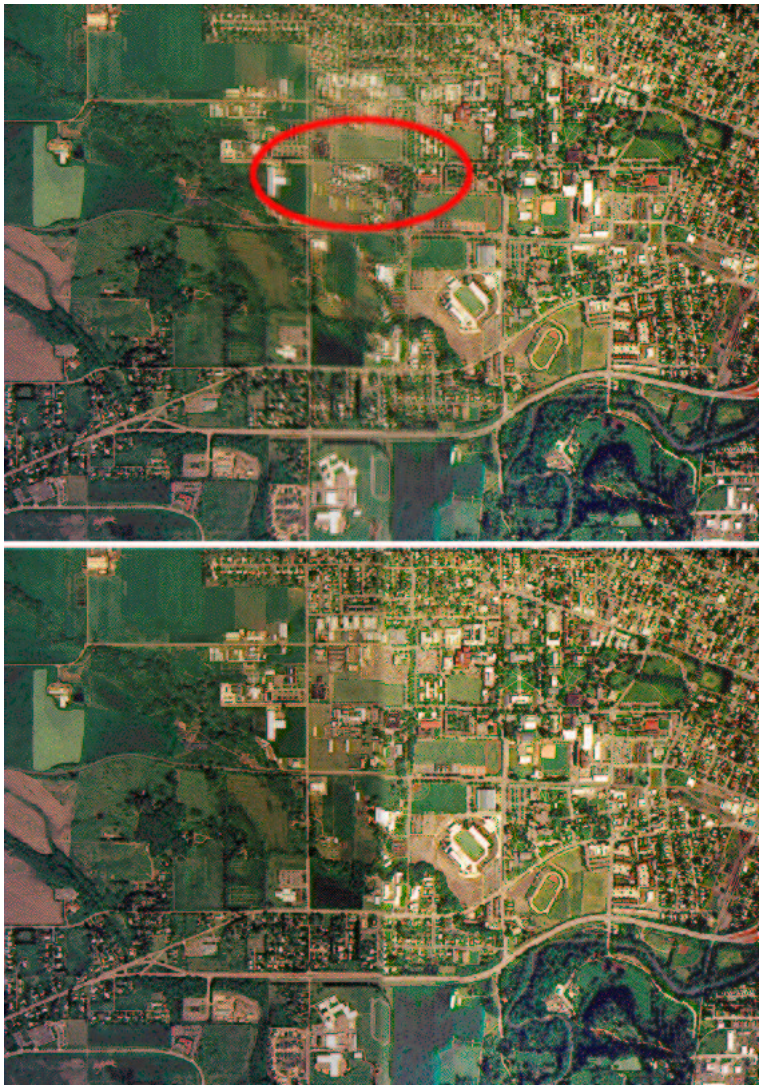

Figure 2.6. Problèmes avec la moyenne pondérée. À partir d'une image aérienne, nous en avons fabriqué une autre en la rendant plus pâle, puis on l'a légèrement translaté et fait tourné. a) Grande largeur de transition : on peut voir l'effet de "double exposition" dans la zone entourée. b) Petite largeur de transition : la transition au centre est évidente.

ment que selon la fréquence, la largeur de transition idéale variera. En conséquence, ils utilisent deux pyramides d'image filtrées passe-bande, qu'ils mélangent un niveau à la fois, pour ensuite les recombiner.

Mais le problème principal, du point de vue du réalisme, de l'approche moyenne pondérée sans ajustement global, est que la moyenne n'influence que la région de chevauchement (cependant, dans le cas de la modélisation d'images par splines multirésolutions [6], cela pourrait ne pas être aussi sérieux). La jonction est camouflée mais les régions sans chevauchement peuvent avoir des couleurs incohérentes, par exemple une texture peut être plus rouge ou plus verte que l'autre. Ceci est dû au fait qu'on permet différentes caméras et différentes illuminations.

L'approche à la correction de textures la plus simple conceptuellement serait de "calculer la *vraie couleur* de l'objet", c'est-à-dire les propriétes de réflectance. En pratique, cela est difficile à réaliser. Sato et al. [25] ont développé une approche où ils calculent la réflectance en utilisant des images 2D et des images de profondeur de l'objet. Un désavantage majeur de cette technique est qu'elle nécessite beaucoup d'images. Pour l'exemple donné dans leur article, ils utilisent 120 images couleur de différents points de vue et 12 images de profondeur. On comprend que cela est difficilement réalisable pour de nombreux objets, notamment ceux qui bougent.

Une autre manière de corriger les textures serait de calculer l'illumination pour ré-illuminer les textures. Cependant, ce problème est sous-contraint ce qui implique qu'on doit ajouter des contraintes. Par exemple, Marshner [17] calcule l'*illumination inverse*, laquelle on pourrait aussi appeller "résoudre l'illumination". Les entrées sont la description de la scène (le modèle 3D et la réflectance) et l'image-requête. L'algorithme procède comme suit. Premièrement, la sphère de lumière de rayon infini est divisée en plusieurs morceaux appelés *sources lumineuses de base* (*basis light*). Ensuite, on fait un rendu de la scène pour chaque source lumineuse. Les résultats sont les *images de base* (*basis images*), qu'ils combinent linéairement pour obtenir l'image-requête. Les coefficients de la combinaison linéaire sont obtenus grâce

à la *décomposition en valeurs singulières généralisée*, qui est une technique d'algèbre linéaire, et à des contraintes de plausibilité, qui spécifient que les valeurs trouvées doivent être positives et semblables à leur voisines sur la sphère d'illumination. Une version modifiée de cet algorithme peut être utilisée pour calculer l'*illumination relative*, laquelle est centrale à notre méthode et sera l'objet du prochain chapitre.

La citation qui suit vient de Gumustekin [12] et explique quelques approches à la *composition d'images*. Elle est similaire à la correction de textures et fait partie de la construction de mosaïques d'images (ses étapes sont la correction géométrique, l'alignement des images, et la composition des images). L'auteur a traduit.

> Trouver la meilleure frontière entre deux images superposées [20] peut éliminer les distortions géométriques restantes. Une telle frontière passera vraisemblablement autour des objets en mouvement, ce qui empêchera la double exposition [8, 11]. Le problème de l'exposition inégale peut être résolu par une égalisation de l'histogramme [11, 16], en redistribuant itérativement l'effet de bord dans une grande région [21], ou par une fonction de mélange lisse [6].

Notre approche ne modifie pas seulement la région de chevauchement mais aussi toutes les textures afin d'obtenir une texture réaliste, c'est-à-dire telle qu'elle semblerait éclairée par une illumination donnée. Cela signifie des couleurs cohérentes partout sur les textures, pour que par exemple un visage semble avoir une couleur relativement uniforme. Le coeur de la méthode est le calcul de l'*illumination relative*, laquelle est similaire au concept de *lightsphere* de Blicher et Roy [5].

Les avantages majeurs de notre méthode sont qu'elle fait peu d'hypothèses, qu'elle donne des résultats réalistes et qu'elle n'implique aucune calibration et aucun calcul explicite de l'illumination. Elle donne des résultats plus réalistes que la moyenne pondérée, laquelle est la technique la plus utilisée. De plus, les textures peuvent avoir été prises sous différentes illuminations ou avec différentes caméras. La principale

hypothèse est que notre méthode requiert que les sources lumineuses soient à l'infini.

2.4 Hypothèses

Voici quelques hypothèses que nous faisons afin de rendre nos modèles plus simples.

Rayons parallèles

Nous supposons que toutes les sources lumineuses sont ponctuelles et sont placées infiniment loin, ce qui fait que tous les rayons incidents venant d'une même source sont parallèles proche de l'objet. Lorsqu'on les condidère comme un tout, ces sources lumineuses sont ce que l'on appelle la *sphère d'illumination.*

Modèles lisses

Nous ne considérons dans notre article que les objets ayant une surface relativement lisse, ou en d'autres mots, les objets n'ayant aucune discontinuité de la normale (sauf sur les bords du maillage). Bien entendu, comme nous utilisons des maillages de triangles, les normales ne varient pas continûment. Cette hypothèse interdit plutôt les variations trop brusques de la normale. Aussi, elle spécifie que la plupart du temps la normale varie peu d'un triangle à l'autre. On déduit que pour chaque point, on peut trouver une surface infinitésimale autour de celui-ci ainsi qu'une normale à ce point, et qu'elles sont perpendiculaires entre elles.

Facteur de forme (*view factor*)

Pour chaque point de la surface de l'objet, nous définissons le facteur de forme : c'est la fraction de la sphère d'illumination qui est visible à partir de la surface infinitésimale autour de ce point. En supposant qu'une surface infinitésimale ne peut recevoir de lumière que de l'hémisphère centré sur sa normale, le facteur de forme devrait être au plus $\frac{1}{2}$.

Convexité

Aussi, nous ne considérons que les objets convexes ou presques convexes (tels que les visages). En d'autres mots, seulement les objets où la plupart des points ont un

facteur de forme proche de $\frac{1}{2}$. Ceci n'est pas vrai en général pour les objets, parce que pour certains points dans les concavités, le facteur de forme est bien moindre. Combiné aux hypothèses précédentes, cela implique que les points ayant des normales similaires ont des illuminations similaires.

BRDF

Nous imposons peu de contraintes sur la BRDF d'un objet. En particulier, il n'est pas nécessaire qu'elle soit lambertienne. La seule chose que nous supposons est que pour les points ayant une normale similaire, la BRDF est similaire.

Observateur distant

On suppose que chaque image est prise de suffisament loin pour approximer les rayons arrivant à la caméra comme étant parallèles. En conséquence, la BRDF pour un point dépend seulement de la normale en ce point, la direction de la lumière incidente, la direction de l'observateur (qui est une constante) et linéairement sur l'albedo qui varie selon ce point.

Normales fiables

Pour les dizaines de visages que nous avons numérisés, les géométries semblaient très réalistes et précises. Nous supposons donc que nous pouvons nous fier sur celles-ci afin de déduire les normales. Cependant, nous avons utilisé le filtrage médian des normales (sur un voisinage autour du point, un composant à la fois, suivi d'une normalisation du vecteur résultant).

Chevauchement des textures

Nous supposons que, pour chaque paire de textures que l'on veut ajuster, il y a une région de chevauchement entre celles-ci. Cette région est nécessaire afin d'obtenir des estimations de l'*illumination relative*.

Chapitre 3

AUTOMATIC RELIGHTING OF OVERLAPPING TEXTURES OF A 3D MODEL

Cet article a été soumi à la conférence Computer Vision and Pattern Recognition (CVPR 2003) et est présenté ici dans sa version originale.

Abstract

This paper presents a new method to correct overlapping textures of a single 3D model where each texture was obtained under possibly different lighting conditions and color response of the camera. This situation arises frequently when a single object is digitized using multiple 3D scanners. Our goal is to remove any visible seam and improve color consistency between the textures in order to merge them afterward. To achieve this, we propose an efficient algorithm to compute the "ratio lighting" of two textures, derive a common lighting from it, and use it to "relight" each texture. We illustrate our method by correcting textures of human faces acquired with several structured-light 3D scanners. Experimental results are realistic and demonstrate how this method can reduce the need to calibrate colors or explicitly solve for the illumination.

3.1 Introduction

Lets suppose that the object of interest is fixed and images of it are acquired by one or more cameras. We are given a geometric model of the object, which may come from merging of several 3D models. For example, in the case of faces, we merge three 3D geometries. The viewpoints and the illumination may be different for each acquisition. Each image is used to create a texture and we receive the correspondence

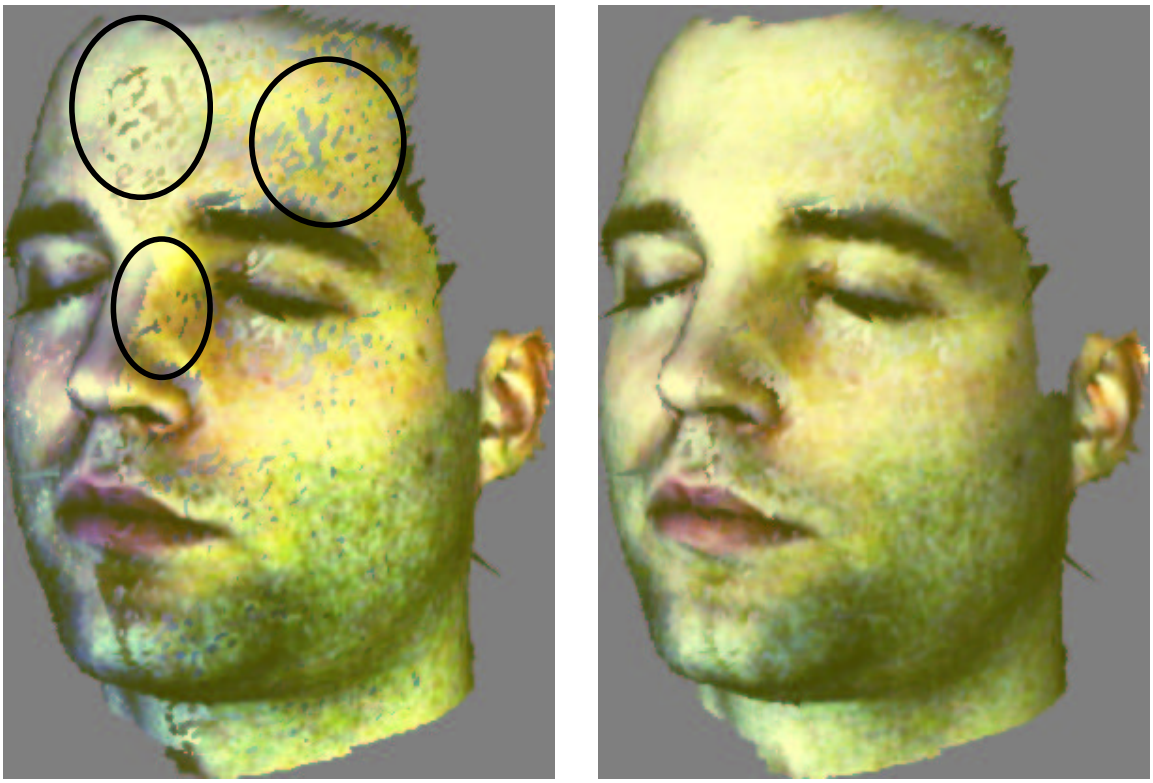

Figure 3.1. A texture correction operation. The histograms were equalized to increase contrast. *Top*. **Before: Two models with their texture shown at the same time, the speckles in the ellipses are due to interlacing of the meshes. We can see that the speckles are not the same color as the surrounding region.** *Bottom*. **After: The same two models with textures corrected. The speckles are much less apparent.**

between the texture and the 3D model from a previous step.

The problem we address here is how to correct these textures so that they can be merged. The elementary operation in our method is to take two textures and correct them relatively to each other. We consider our technique as a preprocessing step for *texture merging*. Our main concern is that the result should look as realistic as possible. See Figure 3.1. We will talk later about the constraints on textures.

A common approach to the problem of texture merging is to simply remove the

seams near texture boundaries, the constraint being that the textures be modified as little as possible. Unfortunately, it lacks realism, because the merged texture may have non-coherent colors, for example if one original texture has a green color bias relatively to the other one. We will now see several approaches to texture correction in order to understand their limitations, and the usefulness of our method.

In one of the earliest work on the subject, Milgram [19] has shown how the difference between two textures can be distributed in the neighborhood of the seam. The input is two rectangle images that overlap. Lets see how it works. First, a "zeroth-order" histogram adjustment is done: the colors of the images are modified by adding a constant value to each pixel so that overlap regions have the same intensity averages (R, G and B processed separately). In the rest of the algorithm, each row will be processed separately, and only a neighborhood of size 2N of the seam will be modified. Second, an algorithm determines on each row where the seam will be. The seam is a point between two pixels in the overlap region that specifies which image should be used for each pixel: on the left of it we use the left image and on the right of it, we use the right image. On each side of the seam, a linear ramp of length N pixels (starting at 0 and increasing in absolute value toward the seam) is added to the intensity values so they meet at the average of the two pixels nearest to the seam. There are some problems with this approach. The fact that it depends on the difference of two pixels makes it vulnerable to the noise they contain, and it creates horizontal lines of slightly different intensity which are readily visible [7]. Also it works only for rectangular images of flat objects where the overlap region is rectangular. In Milgram [20], some improvements are described related to the selection of the seam point and histogram equalization.

Rocchini et al. [23] developed an algorithm for partitioning the vertices of a triangle mesh depending on which texture each vertex gets its color from. That partitioning creates different regions and frontiers between them. The first step is to do an initial partitioning according to a simple criteria. The partitioning is refined

with a greedy algorithm to reduce the number of frontier triangles, which are triangles that have vertices assigned to more than one texture. In other words, they do all reassignments of a node to a texture that, if considered alone, would reduce the number of frontier triangles in the current assignment. The new assignment becomes the current and it continues like this until no further reduction is possible. To calculate the merged texture, interpolation is done inside all frontier triangles using areal coordinates, which are barycentric coordinates normalized so that their sum is 1. Barycentric coordinates of a point inside a triangle are the weights you must put on each of the three vertices so that their center of gravity (the barycenter) is on that point. Consequently, barycentric coordinates are unique up to scale factor.

In contrast, Pulli et al. [22] do not assign a single texture to each vertex. Instead, they do a weighted averaging over all the overlap regions. The weights are calculated by multiplying several factors: proximity to texture border, cosine of the angle of the ray with the normal, areal coordinates of the direction of the virtual view in the delaunay triangle of the directions of the real views.

Levoy et al. [15] also use a weighted average where the weights are the confidence in each pixel value. That confidence takes into account several factors: obliquity of the surface with respect to the light, projected area of the surface with respect to the camera, proximity to the mirror direction (to suppress highlights), proximity to a silhouette edge with respect to the camera, and proximity to a silhouette edge with respect to the light and finally proximity to the edge of the color image. Also to prevent rapid changes in confidence from triggering sudden switches from one color image to another, the confidences are smoothed among neighbors on the mesh. To remain conservative the confidence is never increased, only decreased.

For weighted averaging, there are some recurring problems. If the transition width (of the blending function) is too big, a slight misalignment may produce a *double exposure effect*, where some features appears twice, whereas if it is too low the transition will be made in a few pixels thereby often creating a seam.

Burt and Adelson [6] successfully address these limitations with *multiresolution image splining*. They consider the case where the blending function $H(x)$ has the shape of a sigmoid, the first image has a weight of $1 - H(x)$ and the second $H(x)$. The transition width is a fixed parameter of that function specifying how long (in pixels) it takes to go from 0 to 1. They argue that depending on the frequency, the ideal transition width will vary. So they use a pyramid of band-passed images that are blend one level at a time and then recombined.

But the main problem, from the viewpoint of realism, with the blending approaches not using global adjustment, is that the blending only affects the overlap region (however, in the case of multiresolution image splining [6], this might not be as severe). The seam is actually made invisible but the regions without overlap may have inconsistent colors, for instance one texture may be more red or green than the other. This is because we allow different cameras and different illuminations. Even for the approaches using global adjustment, the correction will be the same over all the texture, although each part of the object (because of the varying normal) is not affected by the same illumination.

The naive way to achieve global adjustment would be to "recover the *real color* of the object", that is the reflectance properties. In practice, this is hard to realize. Sato et al. [25] have developed an approach where they compute reflectance using color and range images of an object. One major disadvantage of that technique is that it needs a lot of images and ranges images. In the example given in their paper, they need 120 colors images from different angles and 12 range images. It is therefore difficult to use in certain cases, especially when scanning humans or animals.

Another way to correct textures is to recover the illumination and then relight the textures. However this problem is ill-posed so that more constraints are needed. For example, Marshner [17] calculates *inverse lighting*, which could also be called "solving for the illumination". The inputs are the scene description (3D model and reflectance) and a query image. First they divide the infinitely far sphere of light

into several pieces called *basis light*. Then they render the scene under each basis light. The results are the *basis images*, which they combine linearly to get the query image. The coefficients of the linear combination (which are also the intensity of the basis lights) are found using the Generalized Singular Value Decomposition and some plausibility constraints. A modified version of that algorithm could be used to compute the *relative illumination*, which we will talk about later.

The following paragraph is taken from Gumustekin [12] and explains some approaches to image compositing, which is related to texture correction and is a step in image mosaicking (the steps in order are geometric correction, image registration, and image compositing):

> Finding the best separation border between overlapping images [20] has the potential to eliminate remaining geometric distortions. Such a border is likely to traverse around moving objects avoiding double exposure [8, 11]. The uneven exposure problem can be solved by histogram equalization [11, 16], by iteratively distributing the edge effect on the border to a large area [21], or by a smooth blending function [6].

Our approach modifies not only the overlap region but also the rest of the textures in order to get a realistic texture, that is, as it would appear under a new illumination. That means consistent colors across textures, so that, for example, the color of the face seems the same all over. It is sometimes surprising how different the colors look when taken with different cameras and illuminations. Yet, in each image seen separately, the colors seem completely natural. The kernel of the method consists in the calculation of the "relative illumination", which is related to the *lightsphere* concept of Blicher and Roy [5].

The major advantages of our method are that it makes few hypotheses, gives realistic results and it doesn't involve any calibration or solving for the illumination. It is much more realistic than weighted averaging, which is the most commonly used

technique. It is much easier to use than the reflectance estimation techniques. Furthermore, the textures may have been taken under different illumination and with different cameras. The main constraint is that our method requires similar lighting condition for points with similar normals. This implies objects that are convex or close to convex, such as faces.

3.2 Hypotheses

Here are some hypotheses we did in order to simplify our model.

Parallel rays We suppose that the point light sources are infinitely far away, thereby making all the rays coming from a source parallel near the object of interest. When taken as a whole, these light sources are what we call the *illumination sphere*.

Smoothness In this article, we consider only objects with a relatively smooth surface, or in other words, objects having no normal discontinuity (except at the edge of the mesh).

View factor For each point on the surface of the object, we define the *view factor*: it is the fraction of the illumination sphere that is visible from the infinitesimal surface patch around this point. Combining this hypothesis with smoothness, we deduce that for all points, we can find a infinitesimal patch around it, a normal at that point, and that both perpendicular to the other. Assuming that a surface patch can only receive light from the hemisphere centered on its normal, the view factor should be at most $\frac{1}{2}$.

Convexity Also we only consider object which are convex or almost convex (like faces). In other words, only objects where most points have a view factor near $\frac{1}{2}$. That is not true in general for objects, since for some points in concavities, it is significantly lower. Combined with the previous hypotheses, it implies that all the points with similar normals have the same illumination.

BRDF We do not impose many constraints on the BRDF of the object, for example,

it doesn't need to be lambertian. The only thing we assumed is that the BRDF is similar for the points having similar normals.

Observer far away We suppose that each shot is taken from far enough to approximate all the rays as parallel. Therefore, the BRDF for a point depends only on the surface normal at that point, the incoming light direction, the direction toward the observer (which is a constant) and linearly on some albedo that may vary from point to point on the surface.

Reliable normals For the tens of faces we scanned, the geometries seemed very realistic and accurate. So we assume that we can rely on the geometries to deduce the normals. However, we used median filtering on the normals (component by component, followed by a normalization of the vector).

Texture overlap We assume that, in each pair of textures we want to adjust, there is an overlap region between them. This region is needed to provide estimates of the "relative illumination", which will be explained in the next section.

3.3 The method

We present a method to adjust the textures two-by-two, however the method could be modified to adjust n textures all at the same time, thereby making a global adjustment. In order to adjust n textures using our method, simply take two overlapping textures, adjust them, merge them into one, and continue to do that with other textures until there is only one left. We do not specify the algorithm used to merge two textures into one, because any algorithm should do the job, the textures being almost equal after the correction.

Conceptually, the elementary procedure in our method rerenders both textures, texture-1 and texture-2, as they would appear if they had been taken under a new illumination.

3.3.1 Notation

Some simplifications have been done for clarity. Instead of showing the equations for each channel (R, G and B), we show them for only one channel, but they are the same for each.

Also we adjust textures, but we don't mention indices of pixels in these textures. The reason is that we can associate each point of the surface to its corresponding pixel in the texture and vice versa. This implies that each pixel used in the texture correspond to a region on the surface of the object.

Let:

- G be the Gaussian sphere (the set of unit vectors which we use to represent directions)
- S be the surface of the object
- $A(p) : S \rightarrow \mathbb{R}^+$ be the albedo (intrinsic reflectance) function
- $N(p) : S \rightarrow G$ be the Gauss map of the surface, i.e., the function that maps each point to its normal.
- $L(g) : G \rightarrow \mathbb{R}^+$ be an illumination sphere (it can represent any number of point light sources). $L_1(g)$ and $L_2(g)$ are the ones present when the corresponding image was acquired.
- $I_i(p) : S \rightarrow \mathbb{R}^+$ be the observed image intensity for texture-i at a given point.
- $d_i \in G$ be the projector vector in the orthographic projection of image-i.
- $BRDF(v_1, v_2, n) : G^3 \rightarrow \mathbb{R}^+$ be the BRDF function for a certain normal, where v_1 is the viewpoint direction, v_2 is the incident illumination direction, and n the normal.

- $V(p,g) : S \times G \rightarrow \mathbb{R}^+$ be the visibility function of the illumination sphere from a given point p in a given direction g (0 = not visible, 1 = visible).

3.3.2 The model

In this notation, and considering an infinitesimal element of area $d\mu$ of G with position g, the general model of surface reflectance with attached shadows can be written as:

$$I_i(p) = A(p) \int\int_G V(p,g) L_i(g) BRDF(d_i, g, N(p)) d\mu$$

By applying the hypothesis that the view factor is near $\frac{1}{2}$ everywhere, we can transform it into:

$$I_i(p) = A(p) \int\int_{G_N(N(p))} L_i(g) BRDF(d_i, g, N(p)) d\mu$$

where $G_N(n) = \{g \in G | g \cdot n \geq 0\}$. Notice that this equation is of the form:

$$I_i(p) = A(p) B_i(N(p))$$

with $B_i : G \rightarrow \mathbb{R}^+$. B_i can be thought of as a brightness function for camera-i that depends on L and $N(p)$, which captures the interaction of the lighting distribution with the normal.

3.3.3 Choice of a new lighting

What we want is to rerender both textures as they would look under a different illumination L', that gives us:

$$\begin{aligned} I'_i(p) &= A(p) B'(N(p)) \\ &= \frac{B'(N(p))}{B_i(N(p))} I_i(p) \\ &= C_i(N(p)) I_i(p) \end{aligned}$$

where I'_i are the rerendered textures, C_i are the relative illuminations and B' is the desired brightness function. B' can be set to any value we want which is valid (B cannot take any arbitrary value, it must be the result of a double integral as mentionned

in the previous section). Using B_1 and B_2, a valid value would be an interpolated value or, more generally, a linear combination with non-negative coefficients. In this case, we conclude:

$$\begin{aligned} C_i(N(p)) &= \frac{I_i'(p)}{I_i(p)} \\ &= \frac{B'(p)}{B_i(p)} \\ &= \frac{k_1 B_1(p) + k_2 B_2(p)}{B_i(p)} \\ &= \frac{k_1 I_1(p) + k_2 I_2(p)}{I_i(p)} \end{aligned}$$

We can generate several interesting cases with differents values of k_1 and k_2 : $B' = B_1$, $B' = (B_1 + B_2)/2$ and $B' = B_2$. In the first and the last case, one texture is modified to use the illumination of the other texture, which is not modified.

3.3.4 Relative illumination estimation and extrapolation

We can get one estimate of $C_i(N(p))$ for each p in the overlap region. We could therefore collect all the available estimates, however a more robust way is to discretize the domain of C_i, which is G, and estimate it by voting in the bins. The result of the vote for a bin is the average of the votes that fell into it. To have more accurate results, the points for which either texture had a value too low are excluded (we used 5 as a threshold, and the intensity goes from 0 to 255). Also we didn't take into account bins that have too few votes (in our case: 10 or fewer).

In the general case, the discretization for voting could be done by partitioning G in regions of similar area and shape. However, for faces, the discretization we used is simpler: it consists in dividing evenly along the x and y components of the normal, each in the range $[-1, 1]$ to make a 20 by 20 array of bins. This works for the faces we scanned, because the face generally don't have normals with a negative z, and when they do, these normals are ignored. However, the discretization for more complicated situations would have to be more elaborate.

At this point, the method has some problems associated with the discretization, the voting, and the extension of the vote, which will be described later. The processing that is described below applies to the array representing $C_i(n)$.

First, there is noise in the pixels of the acquired images and therefore, there will be noise in the estimate of $C_i(n)$. This should be partly compensated by the averaging of the vote in each bin.

Second, in order to correct the textures, we need to have $C_i(n)$ defined over the range of the normals of the points of that texture. However, in general, it will not be the case, unless the texture to correct is entirely covered by the other, which would mean there is redundancy in the views taken.

A solution to both problems is to filter the obtained $C_i(N)$ by a weighted average of the defined values in the neighborhood. The result of this operation is to smooth the defined values and to extrapolate the undefined ones. The weights are given by a gaussian kernel 39 by 39 (with a variance of 0.5 bin widths) multiplied by the number of votes for a certain bin. For a bin more than 2 bins away from any determined one (including itself), the effect is comparable to a nearest-neighbor extrapolation, because of the radial symmetry of the gaussian and because it drops fast.

Third, what sometimes happens is that, because of the pseudo-nearest neighbor extrapolation, there are rapid fluctuations of the values, see for example Figure 3.2. To counter that, the values which were initially undefined are replaced by the result smoothed by weighted averaging (with a gaussian kernel 11 by 11 with a standard deviation of 2 bin widths).

Fourth, because of the discretization, we may see discontinuities in the result when the normal changes from one bin to another. To eliminate that, once we have processed $C_i(n)$ as mentioned, we can calculate $I_i'(p)$ using bilinear interpolation into $C_i(N(p))$.

As an example, suppose we have a model of a face from which three images have been acquired from $-45°$ (left), $0°$ (front) and $45°$ (right). We slightly modified the

Figure 3.2. Intermediate steps in the processing of a lightsphere. Black represents undefined values and other intensities represent ratios ($intensity = 100 * ratio$**). The red circle contains the valid values of the lightsphere.** *On the left*, **the result of the vote.** *In the middle*, **the lightsphere after weighted averaging. We notice the discontinuity caused by pseudo nearest neighbor extrapolation (near the lower right corner).** *On the right*, **the values initially undefined (the black pixels in the left image) are smoothed.**

procedure for n textures here in order to have symmetry between the left and the right. Lets assume there is no overlap between the left and the right textures. We receive the merged geometries from a previous step. Second, we do some texture corrections. We correct the left and the front textures together, to get a new texture called LF. Then the right and front textures, to get FR. Finally, the textures LF and FR to get the final result LFR. The coefficients k_i to calculate LF and FR are $\frac{1}{3}$ for the front texture and $\frac{2}{3}$ for the other textures. The coefficients k_i for creating LFR are both $\frac{1}{2}$. Because of the interpolation, it results in an illumination which is more uniform. Because of the coefficients, the contributions of each textures in the final result is the same.

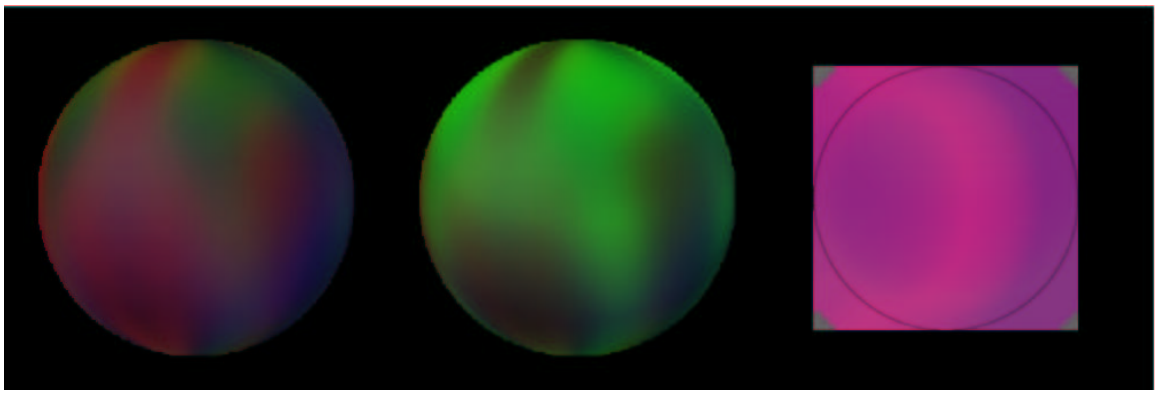

Figure 3.3. A synthetic test of our algorithm. The object is a sphere. Image-1 is on the left. Image-2 in the middle. They overlap completely. Image-2 is rerendered as if it was under the same illumination as when image-1 was taken. The result is not shown because it is equal ± 1 intensity level to image-1. On the right, we see the lightsphere (inside the circle) where each pixel is the ratio of the corresponding bin multiplied by 100.

3.4 Results and discussion

First, we tried our algorithm on synthetic data, a sphere pictured twice from the same viewpoint, but in two different illumination. The results, which were perfect except for the rounding errors, can be seen in Figure 3.3.

We also tried with real data. We used 22 faces to test our algorithm. They were acquired with three structured-light 3D scanners from InSpeck Inc. To do a multi-scanner acquisition, we first do the spatial calibration using a dodecahedron. This allows to find the position of each scanner relative to the others. Second, the scanners acquire data one after the other. Each acquisition produces one 3D model containing more than ten thousands points, each precise to 0.5mm and a 24-bits RGB color texture mapped on the model. Because of the spatial calibration, the models can be easily aligned together and each texture can be stitched onto their mesh.

Some results are shown in Figure 3.1. In Figure 3.4, we can see the errors on the corrected texture. Although there are errors (caused by violations of the hypotheses),

Figure 3.4. The absolute difference between the corrected textures (for the red chanel, it is similar for the two others channels). The completely white regions on the left and on the right are regions where there is only one texture. Black represents 0 and white, 10 or more.

the results still seem very natural. In Figure 3.5, we can see histograms of the differences of the pixels in the overlap region. In Figure 3.2, we see the result of the processing of the lightsphere (weighted averaging plus smoothing of the extrapolated undefined values). We can see the effect of interpolating between B_1 and B_2 in Figure 3.6.

3.4.1 Strengths and weeknesses

Lets now examine the strengths and the weaknesses of our approach. Before deciding to use a correction factor dependent on the normal, we tried several other models. We tried the additive model ($I_2'(p) = I_2(p)+k$), the multiplicative model ($I_2'(p) = kI_2(p)$),

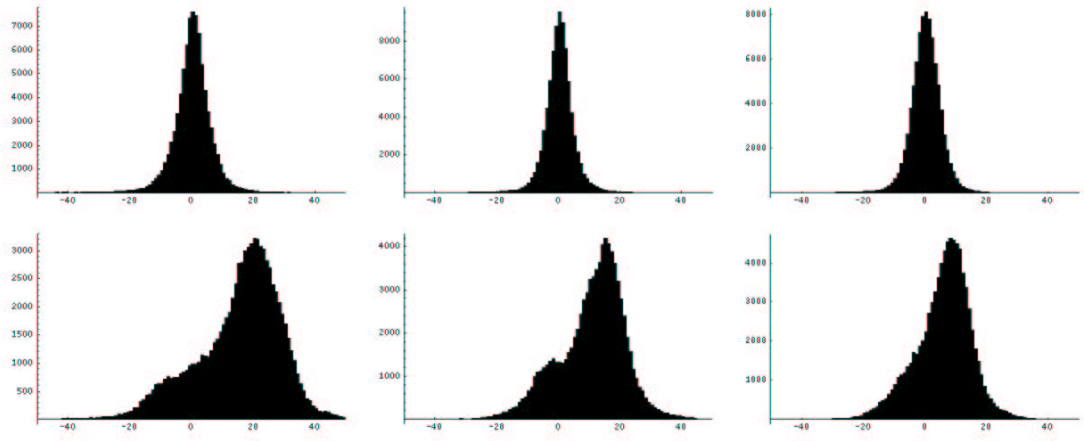

Figure 3.5. Histograms of the differences between pixels in the overlap region for a face. In the top row before the correction (red, green, and blue), and in the bottom row after the correction (red, green, and blue). We can see that the correction centers the error and reduces it significantly.

the exponential model ($I_2'(p) = I_2(p)^k$). All of them gave bad results, where we could instantly see the seam. Blending the textures would not solve our problem even if it could give acceptable results in the overlap region: the parts with no overlap would have different colors. So clearly, this is a strength of our approach, the seam is almost invisible and the color is consistent.

We could overcome the convexity limitation by allowing only certain points to vote: only the points within a certain range of view factor near $\frac{1}{2}$. We would also have to devise a way to calculate a modified correction factor for the point where the view factor is significantly less than $\frac{1}{2}$.

The constraint requiring a common BRDF for all points with the same normal was a good approximation for all the faces, and allows good recovery of specular surfaces (assuming the BRDF is equal at points with the same normals).

Even if the different images were taken with different cameras, which each have

Figure 3.6. Interpolating the illumination. The textures shown are texture-1 on the left, and texture-2 on the right. In the usual order: the original image, the results for $B' = B_2$, $B' = (B_1 + B_2)/2$ and $B' = B_1$. The dark spots on the right of each face are not errors, simply the third texture (not modified here).

their own bias, it doesn't matter because that is taken into account in the "relative illumination". If everything seen by one camera is a factor off what the other one saw, it's already included into $C_i(n)$. This can remove the effect of Automatic Gain Control.

3.4.2 Conclusion

There are some questions that we haven't considered. We haven't investigated the case where there are one or more cycles in the graph defined by the relation "texture a and texture b overlap". For example, it would be the case when one has acquired images all around an object. A simpler case: what if the intersection of 3 textures were non-null? How would we treat these cases, could we get more information?

Also we haven't taken into account the fact that some points have a view factor much less than $\frac{1}{2}$. Some possibilities are to reduce their influence during the vote, exclude them from voting, or compute their correction factor differently.

Chapitre 4

RECONNAISSANCE DE VISAGES

Ce chapitre discute de la reconnaissance automatique de visages en général, des techniques proposées dans la littérature pour la réaliser, et finalement de l'algorithme ICP qui servira au chapitre suivant pour la reconnaissance 3D-3D.

4.1 La reconnaissance de visages

Dans cette section, nous allons aborder le problème de la reconnaissance de visages et plusieurs approches à ce problème. Tout d'abord, commençons par décrire le fonctionnement de haut niveau des algorithmes de reconnaissance de visages, tel qu'illustré à la Figure 4.4.

Lorsqu'on parle de reconnaissance, on affirme implicitement qu'il y a des individus connus et d'autres inconnus. Cela signifie que le système doit pouvoir "faire la connaissance" des gens (en d'autres mots établir une correspondance entre sa perception et l'identité de la personne). De plus, il doit pouvoir le faire d'une façon fiable. On doit donc concevoir une procédure permettant un haut degré de certitude dans l'identification des gens car le système est aussi fort que le plus faible de ses maillons.

La façon la plus simple est probablement de permettre à certaines personnes en qui on a une grande confiance de "présenter" les individus au système. Pour chaque individu, le système fera une acquisition de ses senseurs, calculera une description grâce à celle-ci, puis enregistrera la description avec l'identité de la personne spécifiée par le "présentateur". Dans le cas trivial, la description peut aussi être la donnée telle quelle.

Il a aussi possibilité d'ajouter des *annotations* à chaque description. Ces an-

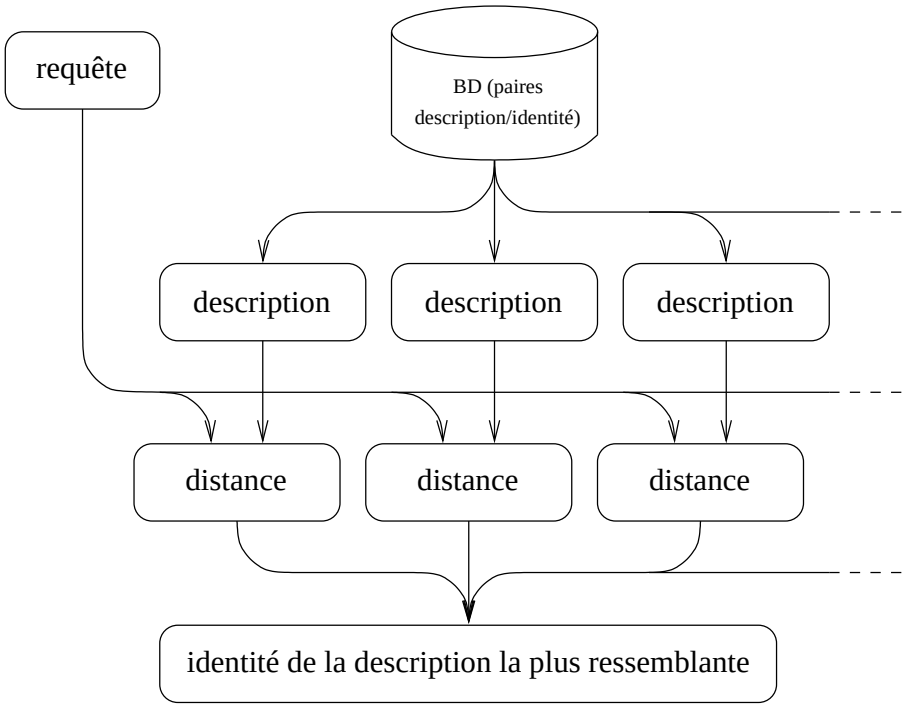

Figure 4.1. Scénario de reconnaissance : ressemblance

notations peuvent être produites manuellement ou automatiquement. Voici quelques exemples qui s'appliquent autant aux images 2D qu'aux modèles 3D. Un usager pourrait spécifier des régions sur le visage qui sont pertinentes pour l'algorithme (comme le front, les tempes, le nez). Ou encore cliquer sur des points de repère (comme le bout du nez, les extrémités des sourcils ou de la bouche). Parfois cela simplifie énormément l'algorithme, comme lorsqu'on spécifie des points de repère sur une image d'un visage pour trouver la pose du modèle 3D correspondant (qui a été annoté similairement).

Scénarios de reconnaissance Il y a au moins 3 scénarios principaux dans la reconnaissance de visages : la ressemblance, l'identification et la vérification. On peut voir un système général de reconnaissance à la Figure 4.4.

Pour la ressemblance, on cherche dans la base de données celle dont la distance à la requête est la plus faible. Cela est illustré à la figure 4.1. Une variante est de trouver les n les plus ressemblants.

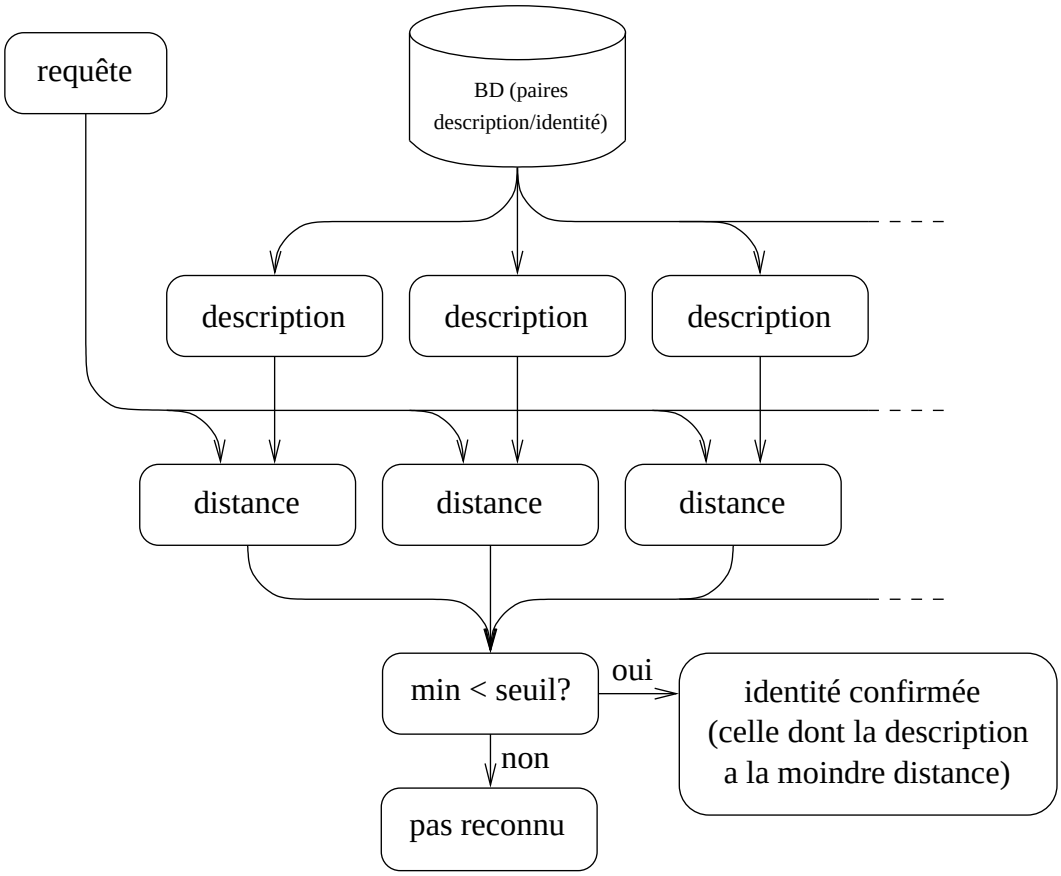

Figure 4.2. Scénario de reconnaissance : l'identification. On remarque qu'il a beaucoup en commun avec le scénario de la ressemblance.

Dans l'identification (voir Figure 4.2), on commence par trouver la description la plus ressemblante, puis on détermine si elle est suffisament ressemblante à la requête, par exemple grâce à un seuil sur la distance.

Dans la vérification (voir Figure 4.3), la personne affirme avoir une certaine identité (par exemple en entrant son numéro d'identification sur un clavier numérique) et le système vérifie cela en s'assurant que la distance est inférieure à un seuil.

On conçoit facilement que la vérification est plus facile que les deux autres car elle n'a qu'à comparer à une seule description de sa base de données, alors que les

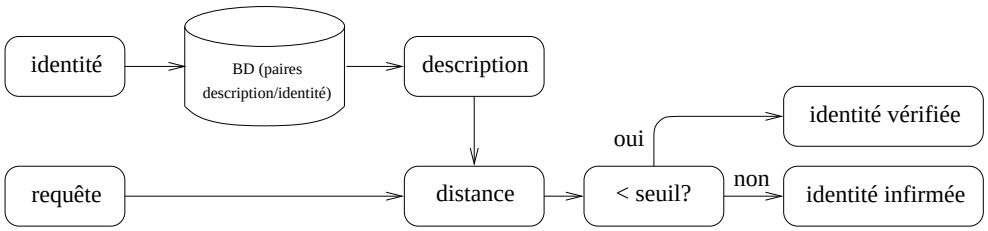

Figure 4.3. Scénario de reconnaissance : la vérification.

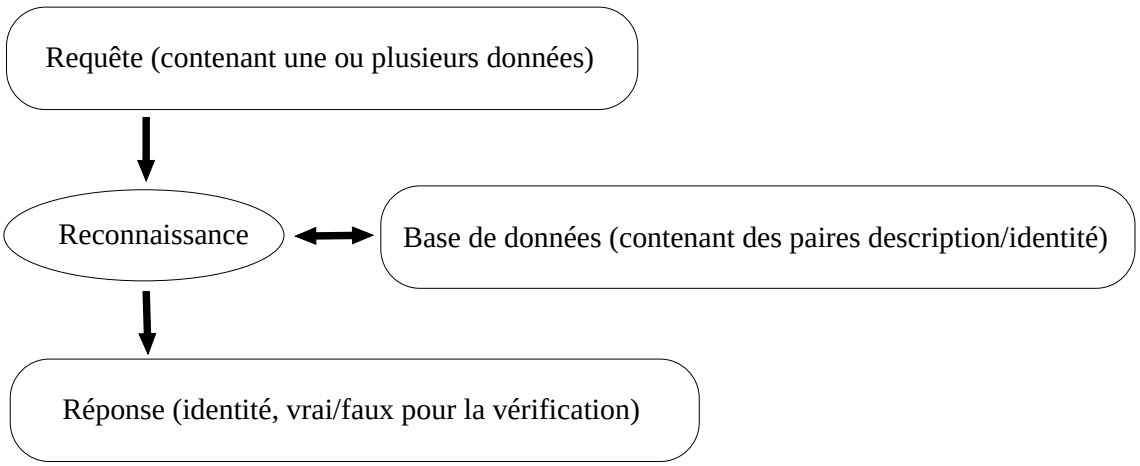

Figure 4.4. Schéma d'un système général. Les données-requêtes comme les données-descriptions peuvent être annotés. La requête peut être constituée d'images 2D, de modèles 3D, d'images de profondeur ou autres. Les descriptions aussi, mais en plus elles peuvent être le résultat d'une variété d'extraction de caractéristiques.

autres doivent comparer à toutes.

On peut implémenter ces trois scénarios en se basant sur une distance entre une requête et une description. Il faut que la distance soit définie pour toutes les requêtes possibles et toutes les descriptions possibles.

Types de données Ici, nous parlerons des différents types de données qui peuvent être utilisés comme entrées à un système de reconnaissance. Cela donne une bonne idée de la variété de la recherche dans le domaine [29]. Il est plus facile d'utiliser un certain type de données lorsque l'équipement est peu dispendieux, facilement

disponible ou encore déja en place (comme les caméras vidéos ou les webcams).

Il faut distinguer les données que l'on emmagasine dans la base de données (lorsqu'on enregistre des informations pour une reconnaissance ultérieure) et celles passées comme requête au système (lorsqu'on fait une reconnaissance). Voyons les types de données que l'on peut utiliser comme requêtes, puis de ceux qu'on peut utiliser comme descriptions. Dans les deux cas, on peut ajouter des annotations tel que mentionné précédemment.

Dans la plupart des cas, la recherche a porté sur des images 2D. C'est sûrement l'option la plus simple, car elle ne demande pas de matériel spécial, simplement une caméra ordinaire. Un des désavantages de ce type est que les données sont sensibles au changement d'illumination. Malgré la popularité des images 2D, on s'intéresse de plus en plus à d'autres types de données qui peuvent combler leurs lacunes.

Une alternative est d'utiliser des images de profondeur (appelées *range images* en anglais). Ces images ressemblent à des images normales, sauf que chaque pixel, au lieu de contenir une intensité, contient une profondeur. Par profondeur, on désigne la composante z de la surface la plus proche dans la direction d'un pixel donné. L'axe z est parallèle à l'axe optique et pointe vers l'avant. La difficulté est d'acquérir des images de profondeur sans qu'elles soient trop invasives. On peut le faire par une technique *passive*, laquelle ne modifie pas les paramètres de la caméras et n'émet pas d'énergie vers la scène. Cependant, cela implique qu'on doit résoudre un problème mal posé (*ill-posed*), ce qui nécessite des traitements et des analyses sophistiqués. De plus, elles ne sont pas encore suffisament robustes. Par ailleurs, on peut utiliser une technique *active*, dans laquelle on émet de l'énergie ou on modifie les paramètres du senseur optique. Ces techniques sont plus fiables et plus simples, parce qu'elles réussissent en général à poser le problème de manière à ce que sa solution soit sans ambiguité [9]. Par contre, elles impliquent une plus grande utilisation d'énergie et dans plusieurs cas un fonctionnement qui peut déranger les gens.

On peut aussi utiliser des vidéos, aussi appelés séquences d'images. Le désavantage

est que, si ce sont des vidéos de surveillance, probablement les plus abondants, ils sont souvent de mauvaise qualité, les visages sont petits et difficiles à identifier [29]. Par contre, l'avantage est qu'on pourrait éventuellement tirer profit de la dimension temporelle pour récolter davantage d'informations.

Sans oublier les dessins, mais ceux-ci sont peu courants. Comme par exemple les croquis ou portrait-robot, utilisés par la police pour chercher dans une base de données.

On peut supposer que plusieurs autres types de données pourraient être utilisés, étant donné la variété de senseurs existants. Par exemple, il y a eu des études sur la reconnaissance par des caméras infrarouges. L'avantage est qu'elles sont insensibles aux changements d'illumination, par contre elles sont sensibles aux changements de température.

Pour ce qui est des descriptions (enregistrées dans la base de données), le choix est beaucoup plus vaste. On peut prendre n'importe quel ensemble de données parmi les types mentionnés précédemment ou n'importe quels ensembles de caractéristiques calculés à partir de ceux-ci. Cette description sera fonction de l'algorithme. Selon les besoins, on choisira différents types de description. On pourrait par exemple rechercher la résistance au bruit, la rapidité d'exécution ou un faible coût.

4.2 Revue de littérature

Passons maintenant aux différentes approches proposées pour implémenter la reconnaissance de visages. Nous les avons divisées selon le type de données utilisé en entrée.

2D-2D Voyons une catégorie d'algorithmes de reconnaissance de visages sur laquelle on a fait beaucoup de recherche : les algorithmes *2D-2D*. Par 2D-2D, on réfère au fait qu'ils performent la reconnaissance sur une image 2D, après avoir constitué une base de données grâce à des images 2D.

Étant donné que l'on peut transformer une image $m \times n$ en un vecteur de dimension mn, on peut appliquer les techniques de l'algèbre linéaire à celle-ci. Pour la reconnaissance de visages 2D-2D, plusieurs techniques populaires ont le même principe de base. À partir des visages et possiblement de leur identité, on calcule une projection linéaire des données, et on fait la reconnaissance sur les données projetées selon cette projection. Voyons trois de ces approches : une utilisant l'*analyse en composantes principales* (ACP) et deux utilisant le *discriminant linéaire de Fisher* (DLF).

Tout d'abord, les *visages propres* (*eigenfaces*) de Turk et Pentland [28] utilisent l'ACP afin de réduire la dimensionnalité des images d'entrée et par le fait même aider à trouver les visages semblables. L'ACP est une technique permettant de réduire la dimensionnalité d'un ensemble de vecteurs en trouvant les directions de plus grande variabilité du nuage de points, appelées *composantes principales*. La plupart de la variabilité de l'ensemble se trouve expliquée par les premières composantes principales. Ils utilisent donc seulement les k premiers visages propres. Malgré l'intérêt de cette technique, elle possède des limitations sérieuses [27] et laisse plusieurs questions sans réponses, comme le nombre de visages propres (k) qu'on doit utiliser. Par exemple, elle suppose qu'on a bel et bien détecté et segmenté les visages dans l'image. Lorsque ces opérations ont mal fonctionné, les résultats en souffrent. Elle est aussi sensible aux variations d'échelle, d'illumination, d'orientation et aux occlusions.

Etemad et Chellappa [13], utilisent le DLF. On appelle aussi l'analyse par ce discriminant LDA (pour *Linear Discriminant Analysis*). Celui-ci permet de trouver le ou les meilleurs vecteurs sur lesquels projeter linéairement une représentation des données pour maximiser le pouvoir discriminant de la projection. L'étape suivante est de faire la reconnaissance sur les données projetées. On peut prendre n'importe quelle représentation de l'image, par exemple, l'image elle-même, sa transformée en ondelette ou encore un vecteur de caractéristiques déduites de l'image. On peut même en prendre plusieurs et les auteurs fournissent une méthode pour combiner les indices fournis par les discriminants des différentes représentations.

Une autre approche utilisant le DLF, est celle des *fisherfaces* de Belhumeur et al. [3]. Ils réduisent d'abord la dimensionnalité des données par une ACP, puis à nouveau grâce au DLF. Cette méthode présente l'avantage par rapport aux *eigenfaces* d'être insensible à de grandes variations de la direction d'illumination ainsi que de l'expression faciale.

Dans l'approche *Elastic Graph Modeling* de Lades et al. [14], on fait une mise en correspondance de graphes afin de reconnaître des visages. On place un sommet sur chaque point de repère du visage : nez, yeux, bouche, etc. Les sommets sont étiquettés par des caractéristiques qui décrivent bien les caractéristiques locales de l'intensité et moins bien les caractéristiques globales, ce qui donne une grande robustesse aux déformations. On utilise des filtres de Gabor pour trouver les caractéristiques en questions. Les arêtes entre ces sommets sont étiquettées par des positions relatives.

2D-3D L'approche *lightsphere* de Blicher et Roy [5] permet de faire une reconnaissance 2D-3D (Figure 4.5). On veut donc trouver parmi des modèles 3D celui qui ressemble le plus à l'image-requête. On suppose que la pose vient d'une étape précédente. Voici le traitement que l'on fera pour chaque modèle dans la base de données. La pose permet de faire un rendu du modèle et celui-ci doit se superposer parfaitement à l'image. La seule différence est que l'illumination n'est pas la même pour le rendu et pour l'image. Cela implique qu'on peut trouver l'*illumination relative*, qu'ils appelent *lightsphere*. Puis, on refait un rendu du modèle en corrigeant avec le lightsphere. Si on compare une image d'un individu avec son modèle 3D, le rendu devrait être presque identique à l'image. La moyenne du carré de l'erreur entre les deux sert de distance entre les deux visages. Le désavantage de cette technique est qu'elle a une certaine sensibilité à la pose.

Basri et Jacobs [2] décrivent les images d'une surface lambertienne ayant une pose donnée comme étant une convolution et se servent de cette description afin de faire une reconnaissance. Ils montrent que, sous certaines conditions et pour une pose donnée, l'espace des images d'un objet est un sous-espace à 9 dimensions. Ils supposent

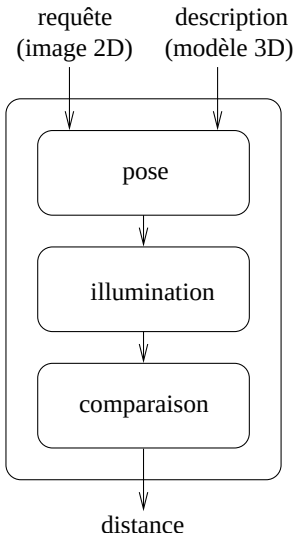

Figure 4.5. Distance pour l'algorithme de reconnaissance 2D-3D de Blicher et Roy.

que la pose de l'objet est connue, que l'objet en question est lisse et convexe, et que l'illumination est composée de sources ponctuelles loin relativement à la taille de l'objet et isotropiques (c'est-à-dire émettant uniformément dans toutes les directions). Il s'agit de trouver l'identité de la personne que l'on voit dans une image. Grâce à la base de données de modèles 3D, on peut produire, pour chaque modèle, une base pour l'espace de ses images. La distance d'une image à un modèle 3D est la distance de l'image à sa projection dans le sous-espace des images du modèle. L'approche de Basri et al., grâce à une description analytique du sous-espace des images, génère une représentation valide de l'espace des images, et qui n'est pas sujette aux particularités d'un échantillon d'images (comme dans le cas d'autres méthodes comme l'ACP).

3D-3D Pour ce qui est de la reconnaissance 3D-3D, à la connaissance de l'auteur, il n'existe pas d'approche spécifiquement pour les visages.

Il existe une approche simple à cette reconnaissance (voir Figure 4.6) : on aligne d'abord les visages, puis on calcule une distance mesurant leur distance (ou dissimi-

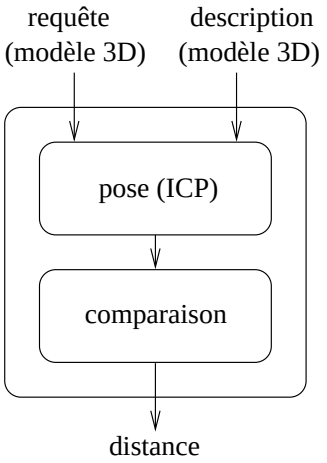

Figure 4.6. Distance pour la reconnaissance 3D-3D

larité).

Vidéo Un vidéo, tel que mentionné plus tôt, permet en général d'avoir accès à plus d'information qu'une simple image. Cependant, selon Zhou et al. [30], la plupart du temps les algorithmes traitant ces données font la détection et le suivi du visage, choisissent une ou plusieurs images dans laquelle la dimension et la pose du visage sont adéquates, puis soumettent ces images à un algorithme de reconnaissance d'images 2D. Ils les qualifient d'algorithmes de suivi-puis-reconnaissance (*tracking-then-recognition*), ce qui les amène à proposer un algorithme de suivi-et-reconnaissance (*tracking-and-recognition*), où les deux processus se déroulent simultanément, ce qui permet de tirer profit de la dimension temporelle des données et de résoudre d'autres difficultés comme le choix d'une bonne image et l'estimation de paramètres pour l'alignement (*registration*).

Chapitre 5

RECONNAISSANCE 3D-3D

Au cours de ce chapitre, nous décrirons le fonctionnement de notre méthode ainsi que les modifications que nous avons fait à l'algorithme ICP pour mieux l'adapter à la reconnaissance de visages (trouver le plus ressemblant).

5.1 Acquisition d'un modèle 3D et construction de la base de données

Les données sont acquises par trois scanners 3D à lumière structurée de type Capturor SF de la compagnie montréalaise InSpeck. On peut voir la disposition du matériel à la Figure 2.2.

Processus d'acquisition d'un modèle 3D L'acquisition des scanners se fait tel que décrit au chapitre 2. Les scanners produisent 4 images de franges ainsi qu'une de texture. Ces images sont reçues par le logiciel FAPS de InSpeck, et une intervention manuelle est nécessaire tout au long du processus. La plupart des opérations de ce processus ont un mode automatique, cependant, ils ne sont pas tellement fiables.

Voici les étapes de la partie logiciel du processus : le prétraitement, la sélection de la zone d'intérêt, le traitement et le post-traitement. Au cours du prétraitement, le logiciel calcule la phase de chaque pixel dans l'image. Il s'agit de la phase en degrés du pixel par rapport aux franges (qui sont une sinusoïde en réalité) projetées sur le visage. Ensuite, on spécifie la région d'intérêt en dessinant un polygone autour de celle-ci. Les autres opérations se feront seulement sur cette région. Le traitement est l'étape où la phase est "déroulée", en ajoutant un multiple d'un tour complet à la phase de chaque pixel pour se débarasser des discontinuités : au lieu de passer de 0 à 359 degrés, on passe plutôt, par exemple, de 360 à 359 ou de 0 à -1. La dernière étape

est le post-traitement, pendant lequel on convertit la phase déroulée en coordonnées 3D, et c'est là que le modèle 3D est créé.

Le processus d'acquisition d'un modèle de visage est le même que ce soit pour l'insertion dans la base de données ou pour la reconnaissance. Ce processus prend beaucoup de temps (si on inclut le traitement manuel), c'est-à-dire au moins 30 minutes par modèle. Il serait possible sans trop de difficulté de rendre l'acquisition de modèles 3D plus automatique. Par exemple, la segmentation pourrait utiliser la technique de l'écran bleu (*blue keying*) où l'on segmente selon la couleur, simplement en mettant un fond bleu derrière la personne, ou un algorithme de détection de mouvements.

Quelques instructions pour les sujets Pour obtenir des conditions relativement uniformes pour tous les modèles, nous avons établi quelques instructions. Tout d'abord, le sujet doit enlever ses lunettes et remonter ses cheveux de manière à dégager son front le plus possible, pour qu'on voit la ligne des cheveux, si possible.

L'acquisition des scanners prend environ 1 seconde, mais si on ne fait pas attention, on aura bougé pendant ce temps, ce qui donne de moins bons résultats. On demande donc au sujet de prendre une grande respiration, puis de s'accoter sur l'appuie-tête.

On demande aussi de fermer les yeux, parce que le projecteur laisse le sujet ébloui pendant plusieurs minutes. De toute façon, la surface de l'oeil n'est généralement pas bien reconstruite, parce qu'elle est luisante pour la majeure partie, ce qui viole les hypothèses de base de la numérisation par lumière structurée. Pour ce qui est de la pupille, elle absorbe la lumière, on ne peut donc pas y observer les franges.

Notre base de données Nous avons acquis 26 modèles dont 9 de femmes et 17 d'hommes. Ceux-ci proviennent de 16 personnes, dont 5 femmes et 11 hommes. Il y a une personne dont on a 4 modèles, 1 dont on en a 3, 5 dont on en a 2, et 9 dont on en a un seul. Pour la personne dont on a 4 modèles et celle dont on en a 3, les modèles ont été acquis en trois jours différents ainsi qu'avec une calibration légèrement différente pour chacun (on a réaligné/recalibré les scanners à chaque fois).

Figure 5.1. Une texture de poids. Le noir représente un poids faible et le blanc un poids fort. La partie manquante sur le nez est due à un trou dans le modèle.

Cela a permis de tester la robustesse de l'algorithme.

Augmenter la robustesse Un visage peut prendre un nombre infini de formes, il convient donc de spécifier les parties du visage que l'on pourra considérer comme rigides et sur lesquelles on pourra prendre des mesures de ressemblance. On verra à la section 5.3 comment ceux-ci sont aussi utilisés pour faciliter l'alignement.

Nous avons utilisé une technique manuelle pour simplifier l'implémentation. Sur chaque modèle, on a peint une texture de poids, comme dans la Figure 5.1. Nous attribuons un poids fort aux zones "rigides" : le front, les tempes, en avant des favoris et le nez. Les autres régions obtiennent un poids faible. Ce choix a été fait en tentant de trouver les zones du visages qui ne varient pas tellement malgré le changement d'expression. Par exemple, il aurait été inapproprié de choisir les joues ou la bouche.

L'utilisation de poids comme nous le faisons permet d'avoir une insensibilité aux mouvements dans les zones plus flexibles (comme le cou). Cela devrait aussi perme-

ttre une insensibilité à l'expression, mais nous ne l'avons pas vérifié dans nos tests. Par ailleurs, les zones que nous considérons comme rigides sont des zones où l'on a généralement pas de poils, ce qui rend l'algorithme insensible à la barbe et à la moustache.

5.2 L'algorithme ICP

L'algorithme ICP, élaboré par Besl et McKay [4], est un algorithme très utilisé pour l'alignement de modèles 3D lorsqu'une *pose* approximative est disponible. Dans le cas présent, la pose sert à représenter le déplacement qu'il faut faire subir à un objet pour le superposer à un autre.

Le principe de base de l'ICP est très simple. L'entrée de l'algorithme consiste en deux modèles 3D, et une pose approximative. Un des modèles est le modèle-données tandis que l'autre est le modèle-référence. Le but de l'algorithme est de raffiner la pose en déplaçant le modèle-données de manière à l'aligner le plus parfaitement possible sur le modèle-référence. Une limitation de l'algorithme est que le modèle-données doit être un sous-ensemble du modèle-référence. En d'autres mots, chaque point du modèle-données doit avoir un point correspondant sur le modèle-référence. Cela pose problème lorsqu'on veut aligner deux modèles dont aucun n'est sous-ensemble de l'autre. On verra plus loin la technique utilisée pour résoudre cette difficulté que nous avons rencontrée.

Chacun des deux modèles 3D peut être représenté sous plusieurs formes : ensemble de points, ensemble de segments, ensemble de courbes paramétriques, ensemble de courbes implicites, ensemble de triangles, ensemble de surfaces paramétriques, ensemble de surfaces implicites. Ces quelques formes englobent la grande majorité des représentations couramment utilisées. La seule contrainte sur la représentation est que l'on doit être capable de calculer, sur le modèle, le point le plus proche d'un point donné. Cette contrainte est facile à satisfaire en pratique.

ICP signifie *Iterative Closest Point* ou "point le plus proche itératif" en anglais, c'est donc une technique qui fonctionne par approximations successives. La première approximation de la pose étant la pose donnée en entrée.

Une itération de l'algorithme fonctionne comme suit (voir la Figure 5.2). Pour chaque point du modèle-données, on créera une paire en lui adjoignant le point le plus proche sur le modèle-référence. Puis, on calculera la transformation rigide (rotation + translation) minimisant la *fonction d'erreur*. La dernière étape d'une itération est d'appliquer cette transformation au modèle-données. La fonction d'erreur est la moyenne, sur toutes les paires de points, des carrés des distances entre les points d'une paire.

Chaque itération fournit un raffinement de la pose. On itère jusqu'à ce qu'on atteigne le critère d'arrêt. Celui-ci dépendra de l'usage qu'on veut faire de l'alignement trouvé. Par exemple, on pourrait vouloir exécuter un certain nombre d'itérations et pas plus ou encore s'arrêter lorsque l'erreur ne s'améliore presque plus (ce qui signifie qu'on est proche d'un minimum local).

L'algorithme convergera dans tous les cas vers un minimum local de l'erreur, cependant, on ne peut garantir que ce sera le minimum global. Pour cette raison, ils suggèrent de répéter l'algorithme en commençant avec différentes poses initiales.

5.3 Modifications à l'algorithme ICP

Afin de pouvoir utiliser l'algorithme à nos fins, nous avons dû utiliser certaines variantes de l'algorithme ICP rapportées par Rusinkiewicz et Levoy [24].

Pose initiale des modèles L'algorithme ICP est conçu pour commencer à partir d'une pose approximative. Autrement, en général, il va immédiatement tomber dans un minimum local et ce ne sera pas un bon alignement.

Au lieu d'utiliser un algorithme d'alignement approximatif, nous déduisons l'alignement d'une contrainte que nous imposons sur l'acquisition : les visages sont habituellement

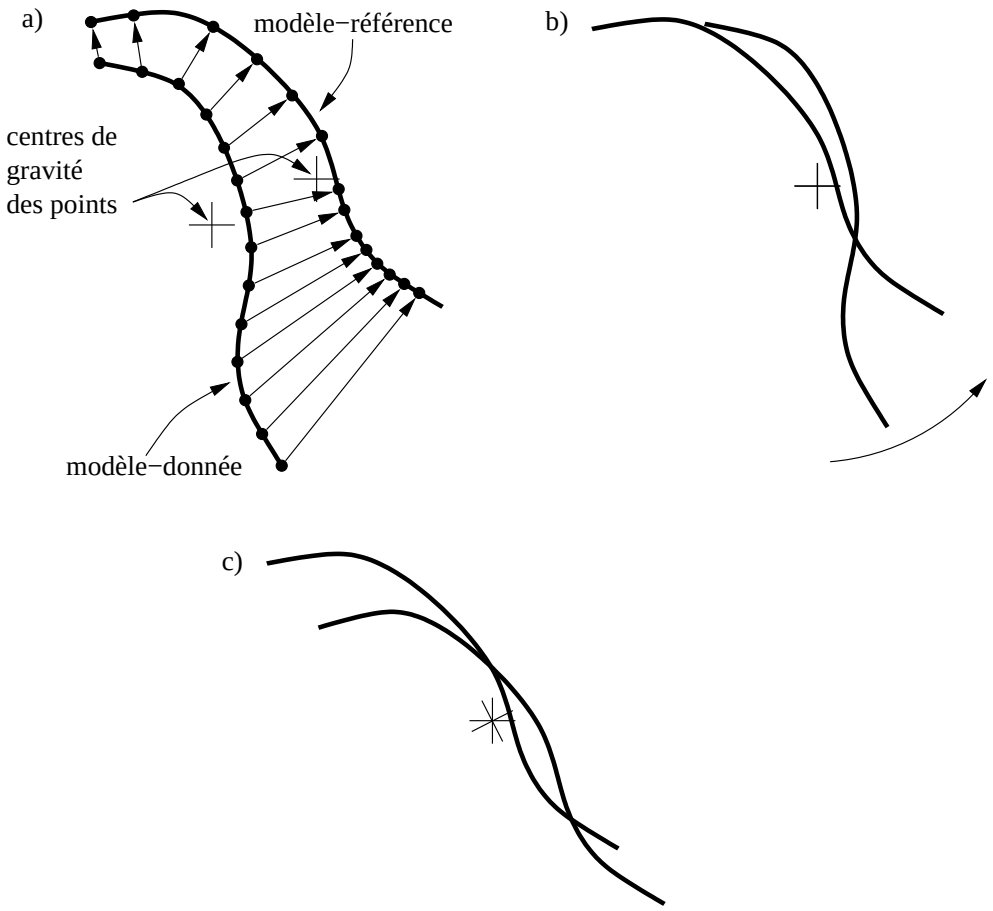

Figure 5.2. Une itération de l'algorithme ICP. Ceci est une analogie en 2D. *a)* **On prend des points sur le modèle-données et on trouve pour chacun le point le plus proche sur le modèle référence. Grâce aux paires, on peut calculer une translation et une rotation.** *b)* **Après avoir appliqué la translation sur le modèle-donnée.** *c)* **Après lui avoir appliqué la rotation.**

tous scannés de la même manière. C'est-à-dire qu'on voit le coté gauche du visage autant que le coté droit. Aussi, les visages sont tous segmentés d'une manière similaire. Malgré que les visages n'ont pas toujours eu la même calibration, tous les modèles examinés avaient l'axe y bien orienté (soit vers le haut).

La première étape est d'aligner les centres de gravité des deux modèles, qu'on calcule en faisant la moyenne des positions des sommets. Ensuite, on fait pivoter les visages autour de leur centre de gravité selon l'axe y en prennant comme référence le vecteur "avant" de chacun. Le vecteur avant d'un modèle est calculé en faisant la moyenne des normales de tous les sommets. L'alignement ainsi trouvé approxime bien l'alignement final.

Cette technique est suffisamment robuste pour nos fins car l'utilisation des moyennes tendent à réduire le bruit. Aussi, même s'il manque une partie importante au modèle, la perturbation ne sera pas trop importante. Prenons comme exemple le cas où l'on enlève 10% des points sur le coté gauche du modèle, et que ces points ont des normales pointant précisément vers la gauche. Dans ce cas, on observe que le vecteur avant serait dévié d'environ 7° à droite de l'avant et le centroïde serait déplacé d'environ 1 cm. Un étude de l'impact du manque de points est faite à la section 5.6.

Afin d'avoir plus de chance de trouver un bon alignement, pour chaque essai, on ajoute à la transformation initiale trois rotations aléatoires autour du centre de gravité du modèle. D'abord autour de l'axe z, puis de l'axe y, puis de l'axe x. Chaque rotation est choisie aléatoirement dans l'intervalle $[-7.5°, 7.5°]$.

Utilisation d'une fraction des points Dans l'algorithme original, on forme une paire pour chaque point du modèle-données (Figure 5.2.a)). Cela est hautement redondant, parce qu'en général un point affectera l'alignement de la même manière que ses voisins. Les modèles qui nous intéressent contiennent environ 10^4 sommets. On peut donc réaliser une accélération considérable en ne prenant qu'un certain nombre de points à chaque étape. Dans le cas présent, on prend 100 points, ce nombre de points garantit que les points sont suffisament bien répartis sur tout le

visage. Les points sont déterminés aléatoirement à chaque pas. Cette modification a déjà été proposée par Masuda et al. [18].

Le désavantage de cette variante est qu'elle transforme l'algorithme original qui est déterministe, en un algorithme non-déterministe. Dans l'algorithme original, chaque pas fait diminuer l'erreur, alors qu'ici, étant donné qu'on n'utilise pas les mêmes points à chaque pas, l'alignement peut s'améliorer alors que l'erreur s'empire. Il faut donc tenir compte de cela dans le design d'un critère d'arrêt dont nous parlerons dans la prochaine section.

Essais Multiples Pour maximiser les chances de trouver un bon minimum local, nous avons décidé d'effectuer plusieurs essais. Chaque essai se déroule comme suit. On commence avec la position initiale décrite précédemment (section 5.3). Puis on itère l'algorithme. Lorsque cela fait τ itérations que l'erreur n'améliore pas la plus basse erreur à ce point dans l'essai, on considère être pris dans un minimum local et on abandonne l'essai courant. On appelle τ le paramètre de tolérance.

On fait au maximum ξ pas au total, ce qui peut impliquer l'interruption d'un essai. Une fois que ce nombre de pas est terminé, on utilise la position ayant l'erreur la plus basse comme solution de l'algorithme. Dans notre cas, nous avons expérimenté un peu et fixé $\xi = 300$. C'est une valeur pour laquelle, lors de l'alignement de tous les modèles sur tous les modèles, tous les alignements ont fonctionné.

Nous examinons ici l'influence de τ, le paramètre de tolérance, afin de trouver la meilleure valeur. Étant donné le nombre limité de pas, on comprend qu'on ne doit pas fixer $\tau = \xi$ - 1 (sa valeur maximale). Cela impliquerait que si on tombe sur un mauvais minimum local, ce qui est probable, on ne peut plus améliorer le résultat. D'un autre coté, si on met $\tau = 1$ (sa valeur minimale), on ne tient pas suffisament compte de la nature non-déterministe de l'algorithme et on interrompt rapidement les essais. La Figure 5.3 illustre les distances et montre que ce paramètre n'a pas une grande influence sur les distances trouvées. On a donc choisi $\tau = 3$.

Rejet de certaines paires L'algorithme ICP de base suppose que le modèle-

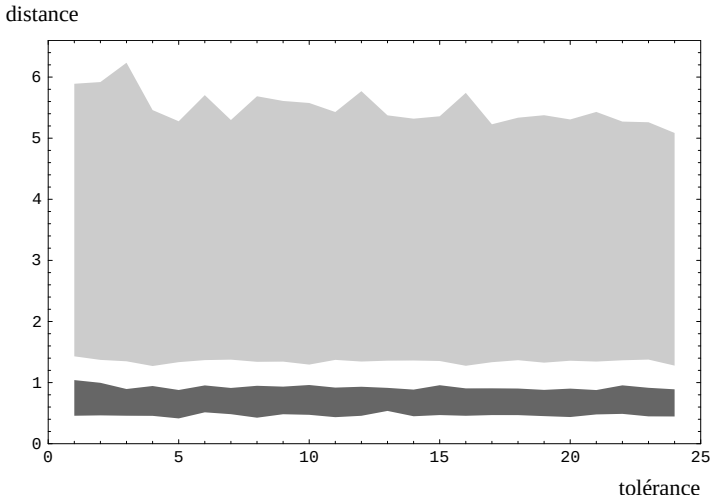

Figure 5.3. Distance entre deux modèles versus τ (paramètre de tolérance) selon qu'ils sont de la même personne ou pas. L'intervalle en gris foncé représente le minimum et le maximum des distances entre deux modèles de la même personne. L'intervalle en gris pâle représente le minimum et le maximum des distances entre deux modèles de personnes différentes.

référence doit contenir tous les points du modèle-données. Dans le cas contraire, il n'y a aucune garantie, comme c'est le cas pour les modèles 3D qu'on a acquis.

Nous avons donc testé l'algorithme de base (en alignant tous les modèles sur tous les modèles) pour voir si, malgré tout, cela fonctionnait suffisament bien. Il s'est avéré que cela fonctionnait en partie : les visages étaient bien alignés en général (Figure 5.4a). Cependant, il y a une fraction non-négligeable où les visages n'étaient pas bien alignés du tout (Figure 5.4b).

Le problème est que l'on ne peut pas dire, en général, qu'un modèle est un sous-ensemble de l'autre. Ce qu'on peut dire, par contre, c'est que les visages ont été segmentés manuellement, et par le fait même, différemment. Cependant, en gros, la segmentation est similaire pour tous les modèles à cause de la procédure que nous utilisons.

Pour cette raison, la variante relative au rejet de paires amenée par Turk [26]

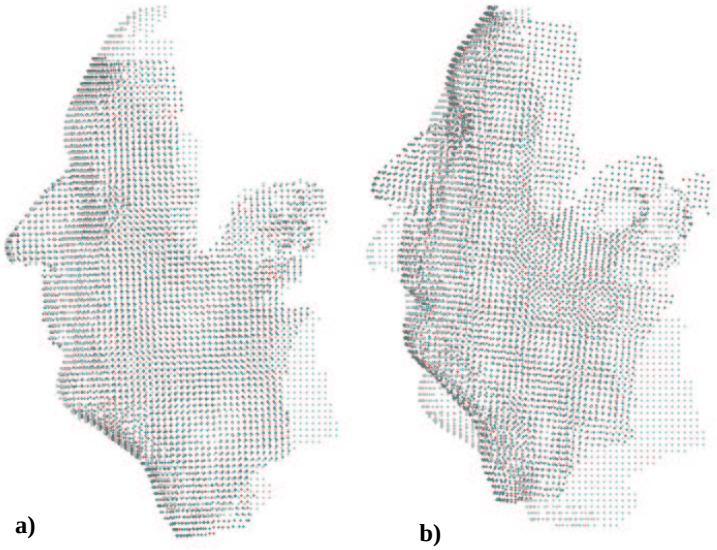

Figure 5.4. Deux résultats de l'alignement (vue de profil d'un visage) : a) un alignement valide et b) un alignement invalide.

convient à nos besoins. Ils proposent qu'une fois les paires de points formées, on rejette les paires dans lesquelles un des deux points est sur le bord du modèle.

Étant donné que les modèles qu'on utilise ne sont pas parfaits et contiennent des triangles dont une des hauteurs est nulle, nous avons implémenté une approximation de cette variante. Premièrement, on trouve le sommet le plus proche du point en question. Si et seulement si ce sommet fait partie d'un triangle touchant au bord du modèle, on considère le point initial comme faisant partie du bord. L'approximation est valide parce que les triangles sont relativement petits. En effet, à cause d'un paramètre donné à l'algorithme de fabrication du modèle, on sait que la longueur maximale d'un coté de triangle est de 10 mm. Toutefois, dans le cas général, les cotés sont beaucoup plus courts, soit en moyenne 3.8 mm.

Il y a souvent des trous dans les modèles, ce qui fait que plusieurs paires contenant un point qui n'est pas proche du contour du modèle seront rejetées quand même. Cependant ceci n'est pas un problème car il n'y en a pas beaucoup et ils sont petits.

Mesure de la distance entre deux points S'il était possible de déterminer la correspondance entre les points du modèle-données et ceux du modèle-référence, le problème de l'ICP serait résolu, car il ne resterait qu'à appliquer la translation et la rotation calculés facilement grâce à la technique mentionnée dans l'article de Besl et McKay [4]. Cela serait possible si on pouvait créer une mesure de distance entre deux points qui donnerait une valeur faible pour les points qui se correspondent et plus élévée pour les autres points. Par contre, cette distance est difficile à définir.

Nous nous sommes donc inspiré de la distance de Feldmar et Ayache [10] : ceux-ci ajoutent la normale du point ainsi que ses courbures principales dans le calcul du point le plus proche. Dans notre cas, nous ajoutons seulement la normale du point. La technique que nous utilisons pour accélérer la recherche du point le plus proche (et dont nous parlerons plus tard) permet des recherches en dimension arbitraire, on peut donc augmenter chaque point avec n'importe quelle information. Pour un point $P = \{x, y, z\}$ sur la surface, ayant une normale $\{N_x, N_y, N_z\}$, on insère donc le point $\{x, y, z, I_N * N_x, I_N * N_y, I_N * N_z\}$ où I_N est l'importance relative de la normale. Après quelques essais, nous avons choisi $I_N \approx D/15$ où D est la longueur de la diagonale de la boîte englobante (*bounding box*). La boîte englobante est le plus petit parallélépipède rectangle contenant le modèle tout en étant aligné sur les axes.

Pondération des paires Il est préférable de moins tenir compte des zones souples du visages, étant donné que nous faisons un alignement rigide. Pour tenir compte des poids de chaque paire, il y a seulement quelques petites modifications simples à faire aux équations. On peut voir sur la Figure 5.1 les poids que l'on mets sur les visages. La zone blanche représente des poids forts (valant P_{fort}) et la noire, des poids faibles (valant P_{faible}). Nous avons déterminé des poids tels que les parties rigides aient une influence beaucoup plus grande que le reste : $P_{fort} = 1$ et $P_{faible} \approx \frac{1}{50}$.

Le poids que l'on attribue à une paire de points est le maximum des poids attribués à ces points (chaque modèle est pondéré). Le maximum convient parce qu'il donne

la même importance à toutes les paires qui ont un point ou plus à poids fort, mais tient quand même compte des paires où les deux points sont à poids faibles.

5.4 Optimisations

Point le plus proche Le calcul du point le plus proche à un point donné sur un modèle est tellement central à l'algorithme ICP qu'il doit être optimisé. Il y a deux cas où nous l'avons optimisé.

Voyons tout d'abord le cas de base. L'approche naïve calcule, pour chaque triangle du modèle, le point le plus proche sur celui-ci et sa distance, et retourne le point dont la distance est la moindre.

Cependant cette technique est lente, car le point le plus proche sur un triangle est une opération relativement chère. Si on arrive à trouver une opération permettant de déterminer qu'un triangle est trop loin pour améliorer la distance minimale observée, et faire cela de manière économique, on peut accélérer significativement le processus.

On peut facilement vérifier que le point le plus proche sur un triangle ne peut être à une distance moindre que la distance à un des sommets du triangle moins la longueur du coté le plus long contenant ce sommet. En utilisant ce fait on peut rapidement écarter les triangles qui sont trop loin de manière certaine, et on ne teste que les triangles qui ont des chances d'être suffisament proches. Comme on utilise toujours le premier sommet du triangle, on peut précalculer la longueur du plus long coté l'incluant.

Passons maintenant à l'autre cas : utiliser seulement les sommets. Grâce aux *arbres k-d*, et en acceptant une légere pénalité, on peut accélérer d'au moins un ordre de grandeur l'opération du point le plus proche. Nous avons donc utilisé la librairie *Approximate Nearest Neighbor* (ANN) de Arya et al. [1]. Celle-ci implémente entre autres une recherche approximative des k plus proches voisins d'un point donné en utilisant un arbre k-d. Cette structure de données permet une recherche en temps

$O(logN)$ où N est le nombre de points dans l'arbre.

Le petit désavantage de cette optimisation est que le résultat de cet algorithme est toujours un sommet du modèle-référence, alors qu'en réalité le point le plus proche peut aussi être sur une arête ou en dedans d'un triangle. Cela implique une perte de précision.

Cependant, cette approche est acceptable parce que les points sont rapprochés (en moyenne à 3.8 mm l'un de l'autre). Cela fait que l'erreur causée par un point est habituellement compensée par celle d'un autre et que l'alignement global est acceptable.

Pour compenser cette perte de précision, à la toute fin, on fait un pas déterministe de l'ICP (c'est-à-dire avec tous les points) en partant du meilleur alignement trouvé dans la partie non-déterministe. Cependant, il y a une petite différence : la distance pour trouver le point le plus proche tient compte seulement de la position des points, et non de leur normale.

Transformations La recherche dans un arbre k-d est en temps $O(logN)$, cependant la construction de celui-ci est en temps O(N log N), où N est le nombre de points. Il faut donc éviter le plus possible la construction, et les opérations telles que la transformation rigide qui nécessite une reconstruction de l'arbre. Une autre raison est qu'il y a des valeurs calculées pour chaque sommet et chaque triangle, et si on déplace les points du modèle, il faudrait en recalculer une bonne partie. C'est ce que nous évitons en simulant la transformation.

Les transformations rigides sont représentées par une matrice 4×4. Au lieu de déplacer le modèle-données, lorsqu'on veut trouver le point le plus proche sur le modèle-référence, on fait une recherche dans l'arbre du modèle-référence avec le point-requête transformé.

5.5 *Fonctionnement de la méthode*

Nous avons implémenté le scénario du "plus ressemblant". Se référer au chapitre précédant pour implémenter les autres scénarios.

L'opération de base de notre méthode est l'alignement, grâce à l'algorithme ICP, de deux modèles 3D de visages. Cet alignement permet de déterminer la distance entre deux modèles 3D, et donc de trouver dans la base de données le modèle le plus proche (le plus similaire).

5.6 *Résultats et discussion*

Pour vérifier la validité de l'algorithme, nous avons fait un alignement de tous les modèles sur tous tous les modèles, et cela donne la matrice de distances illustrée à la Figure 5.5. Les entrées sur la diagonale principale ($i = j$) sont très proches de zéro, étant donné qu'elles viennent de l'alignement d'un modèle avec lui-même. Lorsqu'il y a plusieurs modèles de la même personne, ceux-ci sont consécutifs en indice, ce qui crée des carrés de distances basses sur la diagonale principale.

Comme l'algorithme est non-déterministe, chaque exécution produira une matrice de distances différentes. Pour la matrice présentée, dans tous les cas, si l'on trouve le modèle le plus ressemblant à un modèle donné, on obtiendra un autre de la même personne (si l'on en a un). Évidemment, pour cette opération, on exclut de la base de donnée le modèle-requête, sinon on l'obtiendrait toujours comme résultat. On peut donc dire que la reconnaissance a fonctionné selon cette expérience.

Peut-on pour autant affirmer que l'algorithme marcherait à chaque exécutions? Non, car on ne peut pas garantir que deux modèles de la même personne vont à tout coup bien s'aligner, et ce, à cause de la nature non-déterministe de l'algorithme.

Cependant, on peut vérifier grâce à plusieurs exécutions (dans notre cas 16), si l'algorithme fonctionne avec la même expérience. Pour simuler le calcul de la distance entre toute les paires de modèles possibles, on n'a qu'à piger aléatoirement

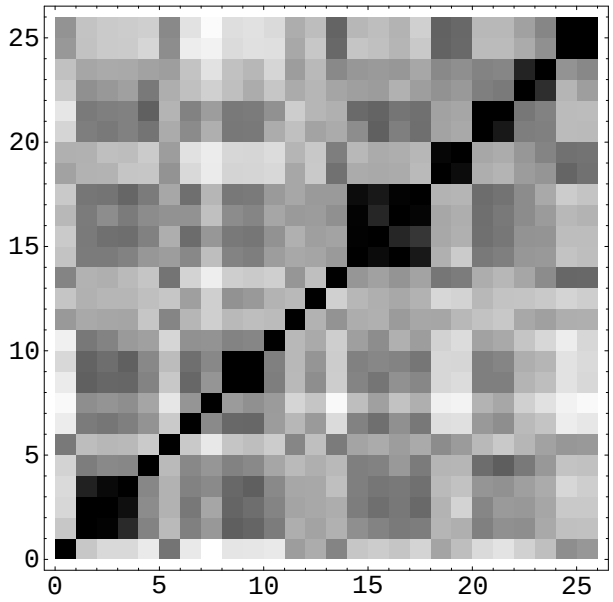

Figure 5.5. Matrice des distances entre deux modèles. On a pris le logarithme des valeurs pour mettre en évidence les différences entre les coefficients. La ligne est le numéro du modèle-données et la colonne celui du modèle-référence. Les carrés foncés représentent des valeurs faibles et les carrés pâles des valeurs élevées.

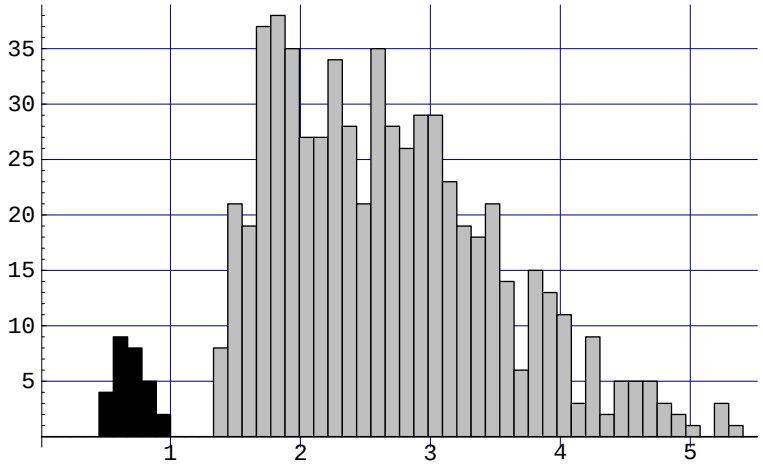

Figure 5.6. Histogramme des distances entre deux modèles selon qu'ils proviennent de la même personne (barres en noir à gauche) ou non (barres en gris à droite).

parmi les valeurs trouvées pour la même paire de modèles lors des exécutions. Le but est de calculer quelle proportion des exécutions du scénario "le plus ressemblant" retournent un autre modèle de la même personne. Lorsqu'on fait 10000 fois cette expérience sur tous les modèles non uniques, on constate que l'expérience fonctionne dans exactement 100% des essais. On peut donc avoir une certaine confiance dans l'algorithme, malgré qu'on ne peut prouver qu'il fonctionnera dans tous les cas.

Pour mieux comprendre ce résultat, considérons la Figure 5.6. Elle représente la distibution des distances entre deux modèles pour toutes les paires de modèles selon qu'ils proviennent de la même personne ou pas. On peut remarquer que les paires de modèles sont très bien séparés selon ce critère, ce qui explique le résultat. On pourrait d'ailleurs trouver facilement un seuil pour implémenter l'identification et la vérification

Robustesse de l'alignement Comme la reconnaissance repose sur un bon alignement, il est important que l'algorithme soit résistant à des données incomplètes, par exemple dû à une occlusion. Pour vérifier que c'est bien le cas, nous prenons

le même modèle comme données et comme référence, sauf que nous tronquons du modèle-données tous les points qui sont d'un coté donné d'un plan. Si l'alignement fonctionne, on trouvera la transformation identité (aux imprécisions numériques près).

Nous faisons deux expériences semblables. Dans les deux cas, la troncation se fait par un plan perpendiculaire à un axe donné. Dans la première expérience, cet axe pointe vers l'avant (ce qui génère des troncations approximativement symétriques), et dans la deuxième l'axe pointe vers la droite (troncation asymétrique).

Pour chaque axe, on peut définir deux plans passant par les points extrêmes. Ces points sont ceux dont le produit scalaire avec l'axe est extrémal. Le paramètre de troncation exprime la fraction où l'on place le plan entre ces deux plans. Ainsi, lorsqu'il vaut 0, aucun point n'est tronqué, et lorsqu'il vaut 1, tous les points sont tronqués. Nous limitons les tests à des valeurs inférieures ou égales à 0.8, car autrement, il ne reste pas assez de points pour aligner. Nous faisons 100 essais pour chaque valeur du paramètre.

La troncation symétrique n'a en aucun cas empêché la convergence vers le bon alignement, même pour les valeurs les plus élevées. Les tests ont été fait sur 4 modèles. Étant donné qu'on tronque symétriquement, la pose initiale reste fiable, et l'algorithme peut converger.

Pour ce qui est de la troncation asymétrique, on s'attend que le problème principal soit la pose initiale : l'algorithme de pose initiale donne des résultats moins fiables lorsqu'on tronque asymétriquement. Si la pose initiale est trop erronée, l'alignement ne convergera jamais (par exemple pour une valeur de troncation de 0.8).

Dans la Figure 5.7a), nous illustrons la robustesse à la troncation asymétrique pour plusieurs visages. On peut voir que certains visages semblent plus faciles à aligner que d'autres mais que dans presque tous les cas, lorsque la fraction de troncation est à 0.2 ou moins, on réussira à l'aligner et à 0.7 ou plus, on ne réussira pas. En général, on réussira à aligner les modèles pour une valeur de 0.3.

Nous avons aussi essayé d'enlever certaines étapes de notre méthode pour voir

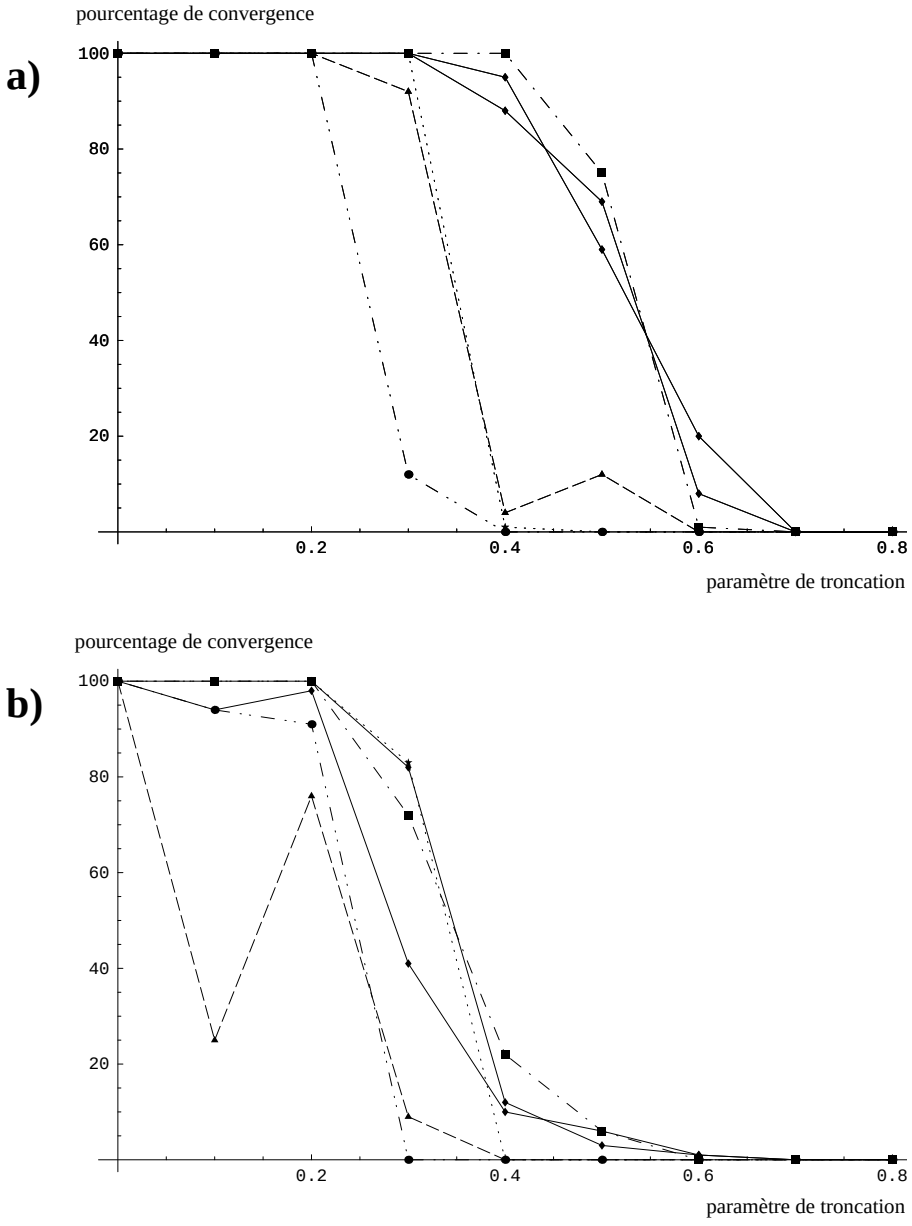

Figure 5.7. Robustesse de l'alignement pour une troncation asymétrique. En abscisse, le paramètre de troncation. En ordonnée, la fraction des 100 essais qui ont réussi. Chaque courbe représente un modèle différent (6 au total). $a)$ **Notre technique.** $b)$ **Notre technique sans la perturbation de la position initiale et utilisant un seul essai d'alignement.**

l'effet. On peut voir à la Figure 5.7b) la robustesse de notre algorithme lorsqu'on enlève la perturbation de la position initiale et en utilisant un seul alignement. On peut voir une différence majeure avec la technique complète. Cela est une justification de ces étapes.

5.7 Conclusion

On a décrit une technique de reconnaissance de visages 3D-3D. Après que les modèles aient été alignés grâce à une variante de l'ICP, on peut mesurer la distance entre ceux-ci. Cette distance est définie comme l'erreur d'alignement lorsqu'ils sont alignés.

Notre système fonctionne bien pour un petit nombre de visages mais pourrait être amélioré en l'automatisant davantage. Présentement, l'intervention humaine est nécessaire et cela prends beaucoup de temps. Nous devons peinturer manuellement les modèles, les segmenter, vérifier les résultats, etc.

Chapitre 6

CONCLUSION

Ce mémoire s'est intéressé à deux aspects de la reconnaissance de visages, soit la correction de textures pour la fusion de modèles 3D et la reconnaissance de visages 3D-3D.

Pour la correction de textures, on propose une nouvelle approche qui ne requiert aucun calcul de l'illumination ambiante et aucune calibration. Pour ce faire nous calculons une correction pour chaque normale.

Pour la reconnaissance de visages 3D-3D, nous avons montré comment un algorithme simple d'alignement (l'ICP) combiné à une distance peut servir à implémenter la reconnaissance 3D-3D.

Comme investigations futures, pour ce qui est de la correction de textures, il serait approprié de tenir compte dans le vote du fait que certains points ont un facteur de forme clairement inférieur à $\frac{1}{2}$. Ces points sont strictement à l'intérieur de l'enveloppe convexe (*convex hull*) de l'objet, par exemple dans une concavité. Même si, pour le cas général, un point dans une concavité ne voit pas toute les sources lumineuses à cause d'auto-occlusions (ombre portée par l'objet sur lui-même), il pourrait arriver qu'il les voit toutes. Cependant, nous sommes principalement intéressés par le cas général. Il serait aussi intéressant de considérer une représentation par harmoniques sphériques de l'illumination relative.

On pourrait tester la résistance de la reconnaissance à une plus grande base de données. Cela permettrait d'avoir une meilleure idée du nombre de personnes qu'elle peut distinguer de manière fiable. En se basant sur l'histogramme des distances (Figure 5.6), on se doute qu'à partir d'un certain nombre de visages, les deux parties de l'histogramme vont se superposer, sauf qu'on ne peut pas le prévoir.

Pour faciliter la création de modèles, il serait intéressant d'automatiser la fabrication des modèles 3D (incluant la création de la texture de poids et possiblement une forme d'annotation).

Finalement, l'alignement pourrait sûrement se faire de manière plus efficace en tenant compte des caractéristiques typiques des visages, comme par exemple la forme généralement convexe, le nez, le menton ou le front.

RÉFÉRENCES

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, et A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[2] R. Basri et D. Jacobs. Lambertian reflectance and linear subspaces. Rapport technique, NEC Research Institute, 2000.

[3] Peter N. Belhumeur, Joao Hespanha, et David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Dans *The Fourth European Conference on Computer Vision (Vol I)*, pages 45–58, 1996.

[4] P.J. Besl et N.D. McKay. A method for registration of 3-d shapes. *IEEE. Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[5] Albert Peter Blicher et Sébastien Roy. Fast lighting/rendering solution for matching a 2d image to a database of 3d models: Lightsphere. *IEICE Transactions on Information and Systems*, E84-D(12):1722–1727, December 2001.

[6] Peter J. Burt et Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, November 1983.

[7] Chia-Yen Chen. Image stitching - comparisons and new techniques. Rapport Technique CITR-TR-30, Tamaki Campus, University of Auckland, October 1998.

[8] J. Davis. Mosaics of scenes with moving objects. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[9] Fred W. DePiero et Mohan M. Trivedi. 3-d computer vision using structured light: Design, calibration, and implementation issues. *Advances in Computers*, 43:243–278, 1996.

[10] J. Feldmar et N.J. Ayache. Rigid, affine and locally affine registration of free-form surfaces. *International Journal of Computer Vision*, 18(2):99–119, 1996.

[11] S. Gumustekin et R.W. Hall. Mosaic image generation on a flattened gaussian sphere. Dans *Proc. of IEEE Workshop on Applications of Computer Vision*, pages 50–55, 1996.

[12] Sevket Gumustekin. An introduction to image mosaicing. http://likya.iyte.edu.tr/eee/sevgum/research/mosaicing99/, July 1999.

[13] K.Etemad et R.Chellappa. Discriminant analysis for recognition of human face images. *J.Opt. Soc. Am.*, 14(8), 1997.

[14] Martin Lades, Jan C. Vorbrüggen, Joachim M. Buhmann, Jörg Lange, Christoph von der Malsburg, Rolf P. Würtz, et Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[15] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, et D. Fulk. The digital michelangelo project: 3d scanning of large statues. Dans *Proc. of SIGGRAPH 2000*, New Orleans, Lousiana, USA, July 2000.

[16] J. Lim. *Two Dimensional Signal and Image Processing.* Prentice Hall, 1990.

[17] S.R. Marshner. *Inverse rendering for computer graphics.* PhD thesis, Cornelle Univ., 1998.

[18] T. Masuda, K. Sakaue, et N. Yokoya. Registration and integration of multiple range images for 3-d model construction. Dans *Proceedings of the 13th International Conference on Pattern Recognition*, 1996.

[19] D. L. Milgram. Computer methods for creating photomosaics. *IEEE Trans. on Computers*, C-24:1113–1119, November 1975.

[20] D. L. Milgram. Adaptive techniques for photomosaicking. *IEEE Trans. on Computers*, C-26:1175–1180, November 1977.

[21] S. Peleg. Elimination of seams from photomosaics. *Computer Graphics and Image Processing*, 16:90–94, May 1981.

[22] K. Pulli, H. Abi-Rached, T. Duchamp, L. Shapiro, et W. Stuetzle. Acquisition and visualization of colored 3d objects. Dans *Proc. of Int. Conf. on Pattern Recognition*, pages 11–15, 1998.

[23] Claudio Rocchini, Paolo Cignomi, Claudio Montani, et Roberto Scopigno. Multiple textures stitching and blending on 3d objects. Dans *Proc. of Eurographics Rendering Workshop 1999*, pages 173–180, Granada, Spain, June 1999.

[24] Szymon Rusinkiewicz et Marc Levoy. Efficient variants of the icp algorithm. Dans *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling*, 2001.

[25] Yoichi Sato, M. Wheeler, et Katsushi Ikeuchi. Object shape and reflectance modeling from observation. Dans *Proc. of ACM SIGGRAPH 97, in Computer Graphics Proceedings, Annual Conference Series 1997, ACM SIGGRAPH*, pages 379–387, August 1997.

[26] G. Turk et M. Levoy. Zippered polygon meshes from range images. Dans *Proc. SIGGRAPH*, 1994.

[27] M. Turk. A random walk through eigenspace. *IEICE Trans Information and Systems*, E84-D(12):1586–1595, 2001.

[28] M. Turk et A. Pentland. Eigenfaces for recognition. *Journal of Neuroscience*, 3(1):71–86, 1991.

[29] W. Zhao, R. Chellappa, A. Rosenfeld, et P. Phillips. Face recognition: A literature survey. Rapport technique, University of Maryland, 2000.

[30] Shaohua Zhou et Rama Chellappa. Probabilistic recognition of human faces from video. Submitted for publication to Computer Vision and Image Understanding (CVIU).

Annexe A

INSTRUCTIONS RELATIVES À LA CRÉATION DE MODÈLES

Les guides de InSpeck à propos de FAPS (User's Guide et Tutorial) et EM sont très utiles, une lecture attentive en est donc recommandée. Ces guides sont completés par les directives suivantes.

A.1 Chaise de dentiste

- Il y a deux boutons placés directement sous l'appuie-tête, un blanc et l'autre noir. Le blanc place automatiquement la chaise en position couchée et le noir, en position assise (aussi appelée de base). ATTENTION : Chacun de ces deux boutons initie un mouvement automatique qui s'arrête seulement lorsque completé ou lorsque qu'on appuie sur un des 6 boutons latéraux (sur le coté du dossier), il faut donc vérifier qu'il y a assez d'espace libre.

- Pour calibrer avec le dodécaèdre, il faut faire pivoter la chaise de 90 degrés autour de son axe vertical (pour cela il faut décoincer le levier au pied) et baisser le dossier. Le but est de placer le dodécaèdre au même endroit que le visage serait. Il est mieux de ne pas déplacer le socle de la chaise pour garder les conditions les plus constantes possibles. Il serait bon d'avoir un point de référence afin de pouvoir ramener la chaise dans la direction initiale.

- Pour monter et descendre la chaise, il y a deux boutons latéraux ainsi qu'une pédale sur le coté droit.

A.2 Installation du matériel

- Placer la chaise pour scanner.

- Ajuster la hauteur des scanners de manière qu'ils soient à la hauteur du visage pour une personne de taille moyenne et à la position de base de la chaise.

- On doit ensuite placer les scanners autour de la chaise. Ils doivent être approximativement à la distance optimale de scan et aux angles suivants (l'ouverture de la caméra est le point de référence pour les angles) (les angles sont en coordonnées polaires dont le centre est le visage à scanner et le 0 degrés est la direction où le visage regarde) :

  - le scanner 1 à -45 degrés (numéro de série 20010092)
  - le scanner 2 à 0 degrés (numéro de série 20010091)
  - le scanner 3 à 45 degrés (numéro de série 20010084)

- En tentant de garder l'angle constant, ajuster la position des scanners à la distance de scan optimale (c'est un facteur qui joue assez sur la qualité du résultat). On peut déterminer cette distance expérimentalement à l'aide d'une feuille de papier : c'est l'endroit où les franges sont au focus sur la feuille. Les points de référence pour la distance sont le devant du scanner et la profondeur moyenne du visage vu par le scanner en question. Pour garder cette distance la plus constante possible, on ne doit pas changer l'inclinaison de la chaise, mais seulement la hauteur de la chaise et possiblement la hauteur de l'appuie-tête.

- Mettre au niveau les scanners (grace aux niveaux à bulle et aux manettes dans la partie supérieure des trépieds).

- Brancher les fils et démarrer les scanners, puis orienter les scanners vers l'appuie-tête avec FAPS (on peut faire "switch mode" apres avoir montré le "live video" dans l'*Acquisition Control Center*).

- Placer les 2 écrans entre les scanners et devant ceux-ci. Il est mieux de mettre les écrans le plus proche du visage possible sans obstruer les scanners (caméra et projecteur).

- Placer les lumières (un trépied par écran, chacun portant deux lumières).

  - Derrière les écrans, elles ne doivent pas etre dirigées directement vers le visage, mais plutôt vers un point entre le scanner du centre et le visage. Le but est d'éviter que les rayons soient dirigés directement vers le visage, ce qui créerait des spécularités.

  - Sur chaque trépied, une des deux lumières porte un gel bleu, c'est pour rendre la lumière plus blanche (au sens où une feuille blanche sera vue environ comme un ton de gris avec les caméras que l'on a).

- Placer les fils pour qu'on ne puisse pas s'enfarger dans ceux-ci. Utiliser du "duct tape" s'il le faut.

- Calibrer les scanners dans FAPS avec le dodécaèdre. Lorsque la correspondance des plans est bonne, les scores sont inférieurs à 0.5. Après chaque calibration, on doit sauver les paramètres hardware (Tab Setup->Hardware->Save Configuration, le nom du fichier est la date). Il est important de ne pas toucher ou accrocher les scanners une fois la calibration faite. Si cela arrive, on doit refaire la calibration.

A.3 Acquisition d'un visage

- Faire signer la formule d'acceptation et y inscrire le numéro d'identification de la personne (dont on parle plus loin).

- Prendre des informations sur la personne et l'entrer dans la base de données. Le fichier se nomme liste_des_gens.txt et contient un enregistrement par ligne sous le format suivant :

```
xxxx|<nom de la personne>|<email>|<commentaires :
    numero de tél, etc...>
```

  Le premier champ (le numéro d'identification de la personne) ainsi que les 3 barres verticales sont obligatoires, le reste est souhaitable.

- Traitement avec FAPS

  - Acquisition

    * Demander d'enlever les lunettes et de dégager les oreilles et le front.
    * Ajuster la hauteur de la chaise de facon à bien voir la ligne des cheveux en haut et en bas du menton. Ajuster la hauteur de l'appuie-tête pour que la personne ait la tête bien accotée.
    * Ajuster le diaphragme de la caméra de chaque scanner de manière à éviter les pixels saturés sur le visage (cela va donc dépendre de la couleur de la peau). Un pixel peut etre saturé sans avoir une valeur au-dessus de 250. Pour voir s'il y a saturation en un point, on doit regarder le graphique RGB sur une ligne passant pas le point. Si la couleur sur le visage devrait etre la même tout le long de cette ligne et que le rapport entre les composantes R, G et B ne sont pas les mêmes

tout le long, c'est qu'il y a saturation. Il peut y avoir saturation pour des valeurs aussi basses que 220.

* Sauvegarde des images (Save (.seq)), entrer le nom du fichier est xxxx_y où xxxx est le code de 4 chiffres attribué a la personne, et y est le numéro du scan, la plupart du temps ca sera le premier donc 0.

A.4 Construction du modèle 3D

On peut éxécuter cette étape à n'importe quel moment après l'acquisition.

- Traitement avec FAPS
  - Prétraitement (Preprocess All)
  - Sélection de la zone d'intérêt (Select Interest Area)
    * Ne pas inclure de cheveux, ni d'autre surface que le visage.
    * Inclure les oreilles et une partie du cou si possible.
    * Pour les vues de coté, couper sur le plan de symétrie du visage.
  - Traitement (Process All)
  - Post-traitement (Postprocess All)
  - Exportation (Export All)
    * 1 sample over 3
    * Data Type : 3D Model : NET
    * Coordinate system : World : Digitizer : 2
    * Nom du fichier : id_numeroscan (les noms sont générés automatiquement à partir de cette information).

Nous avons maintenant le modèle d'un visage mais en plusieurs morceaux, il faut donc les fusionner.

- Traitement avec le programme de correction de texture pour corriger les textures.

- Traitement avec EM

  - Fusion afin de créer un seul modèle.
  - Exportation vers le format SZE (format de Inspeck)
  - Peinture des poids sur le modèle tel qu'illustré à la Figure 5.1

A.5 Divers

- On enregistre tous les fichiers résultants sur c:\scans sur la machine de scan.

- IMPORTANT : Il faut faire une copie de sécurité des scans régulièrement sur une autre machine (par exemple sur un compte UNIX).