UNIVERSITE DE GENEVE

FACULTE DES SCIENCES

Département d'informatique — Professeur T. Pun

FACULTÉ DES SCIENCES ÉCONOMIQUES ET SOCIALES

Département de systèmes d'information — Professeur N. Magnenat-Thalmann

---

Feature-Based Approach on Animatable Virtual Human Cloning

THÈSE

présentée à la Faculté des sciences de l'Université de Genève pour obtenir le grade de Docteur ès sciences, mention informatique

par

WonSook LEE

de

CheongJoo (South Korea)

Thèse N° 3190

GENÉVE

Atelier de reproduction de la Section de Physique

2000

La Faculté des sciences, sur le préavis de Monsieur T. PUN, professeur ordinaire et codirecteur de thèse (Département d'informatique), Madame N. MAGNENAT-THALMANN, professeur et directrice de thèse (Faculté des sciences économiques et sociales), Messieurs M. GROSS, professeur (ETH Zürich - Département d'informatique), A. CHALMERS, professeur (Université de Bristol, Département d'informatique et multimédia - Angleterre) et F.-E. WOLTER, professeur (Université de Hanovre, Département d'informatique - Allemagne), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 21 août 2000

Thèse - 3190 -

[illegible]

Le Doyen, Jacques WEBER

Acknowledgement

I would like to express my special thanks to the director of my thesis, Prof. Nadia Magnenat-Thalmann for her guidance, support and encouragement. I appreciate the time and effort of the jury members, Prof. T. Pun (University of Geneva), Prof. M. Gross (ETH Zurich), Prof. A. Chalmers (University of Bristol) and Prof. F.-E. Wolter (University of Hanover).

I would like to give my thanks to Prem Kalra to give me the first advice to start the face cloning program and give me a serious review of this thesis, to Laurent Moccozet for the early development of the DFFD method, to Pierre Beylot for a face animation program, to Hyewon Seo to port the SGI version of face cloning program to PC, to Jin Gu for the development of the body cloning program together.

I extend my thanks to all members of MIRALab, University of Geneva, both past and present, for their cooperation and discussion during this work.

Finally I appreciate my parents for the endless love and worry to their daughter in a foreign country far from Korea.

Abstract

6,073,677,405! It is the total population of the World, projected to 5/31/00 at 12:34:28 GMT according to the International Programs Center, U.S. Bureau of the Census[1]. The world is full of people, but how many virtual humans do we have? Most digital cities and buildings such as virtual museums do not have any people around. We have some virtual representative for us, but most of them have either a very simple shape or aliens and exotic animals. If there are some virtual humans, they are specially selected by their fame. In the future, virtual twins of us will populate the virtual worlds. These virtual twins will act in different situations, where they will play our role in telecommunications processes, where we could join virtual ballet or concert in entertainment, and where we can simulate new hairstyle or new dress style before applying them really. It will be as if we would like to deal with kind, affordable and smart humans that will be at our disposal at any time with a lot of knowledge and help us to understand, to learn and to experience situations.

In this thesis we describe the feature-based modeling of animatable human body, so called virtual cloning or virtual-twins. The basic idea is to consider a virtual actor as a combined set of data, including the 3D shape but also the structure to animate the shape inside a Virtual Environment. The human modeling approach described starts from default virtual human templates, including shape and animation structures, and modifies the shape to create a new virtual actor. Shape modifications are transmitted to the attached animation structures in order to keep the set of data consistent. As a result of the modeling process, we get "ready to animate" virtual humans. Our approach optimizes the modeling of virtual humans in order to animate him/her. Feature based approach is useful to make easy input and to take the animation structure while the shape resembles the input shape.

The methodology is composed of two major components: face-cloning and body-cloning. Two views photographs are used for face cloning program and three photographs taken from the frontal, side and back of a person are used for body cloning program. The input is taken in any given imaging environment without requiring a special background or a controlled illuminating condition. The characteristics of these systems are that it utilizes the generic model with the animation information and deform the model to adapt to the individuals. A generic model specified in the VRML H-Anim 1.1 format is used to generate an individualized virtual human. The face-cloning component uses feature points detection on the frontal and side images and then uses Dirichlet Free-Form Deformations (DFFD) for shape modification. Next a fully automatic seamless texture mapping is generated for 360^o coloring on a 3D polygonal model. The body cloning component has two steps: (i) feature points specification, which enables automatic silhouette detection in an arbitrary background (ii) two-stage body modification by using feature points and body silhouette respectively. The final integrated human model has photograph-realistic animatable face, hands, feet and body. The result can be visualized in any VRML compliant browser.

As applications, we explain our three extensions of the face cloning methodology. They are (i) giving animation structure on range data, (ii) a real-time 3D-face morphing system used both for the real-time visualization and also for the generation of new photo-realistic faces, (iii) adaptation of face cloning to the standard MPEG-4.

[1] http://www.census.gov/cgi-bin/ipc/popclockw

Résumé

6 073 677 405 ! C'est la population totale du monde estimée le 31 Mai 2000 à 12 heures 34 minutes et 28 secondes GMT selon le bureau du recensement[2] de l'International Programs Center, US. Le monde est peuplé d'humains, mais combien d'humains virtuels existe t'il ? La plupart des bâtiments et des cités digitaux, comme les musées virtuels, ne sont parcourus par aucun humain virtuel. Nous disposons de représentations virtuelles de nous-mêmes, mais la plupart sont soit de forme très simple, soit des aliens ou des animaux exotiques. S'il existe des humains virtuels, ils ont été spécifiquement choisis en fonction de leur renommée. Dans le futur, nos jumeaux virtuels peupleront les mondes virtuels. Ces jumeaux agiront dans différentes situations où ils joueront notre rôle dans des processus de télécommunication. Nous pourrons ainsi nous joindre à un ballet ou à un concert virtuel, ou encore essayer un nouveau style de coiffure ou de robe avant de les adopter. Ce sera comme si nous avions à faire à des humains sympathiques, intelligents et qui seront à notre disposition à tout moment avec d'énormes connaissances pour nous aider à comprendre, apprendre et expérimenter toutes sortes de situations.

Dans cette thèse, nous présentons une approche de modélisation de corps humains animables à partir de caractéristiques, connue sous le nom de “clonage virtuel”(virtual cloning) ou “jumeau virtuel”(virtual twins). L'idée de base est de considérer l'acteur virtuel comme la combinaison d'un ensemble de données comprenant la forme tri-dimensionnelle, mais aussi la structure d'animation de cette forme dans l'environnement virtuel. La technique de modélisation proposée utilise des modèles prédéfinis d'humains virtuels, comprenant la forme et la structure d'animation, et dont la forme est modifiée pour produire un nouvel acteur. Les modifications de la forme sont transmises à la structure d'animation associée afin de maintenir la consistence de l'ensemble des données. Le résultat du processus de modélisation est un humain virtuel “prêt pour l'animation”. Notre approche optimise la modélisation des humains virtuels pour l'animation. L'approche basée sur le contrôle d'un modèle générique par des caractéristiques permet d'utiliser des données d'entrée simples et de maintenir la structure d'animation quand la forme à produire est suffisamment proche de la forme de base du modèle générique.

La méthodologie se décompose en deux parties principales: le clônage du visage et le clonâge du corps. Deux prises de vue sont utilisées par le progamme de clônage du visage (de face et de profil) et trois prises de vue (de face, de profil et de dos) par le programme de clônage du corps. Les prises de vue peuvent être réalisées avec n'importe quel équipement photographique, sans nécessiter l'utilisation d'un arrière-plan spécifique ni des conditions d'illumination particulières. La caractéristique du système présenté est qu'il utilise un modèle générique intégrant la structure d'animation et le déforme pour s'adapter aux individus. Un modèle générique spécifié dans le format VRML H-ANIM 1.1 est utilisé pour générer un humain virtuel individualisé. Le module de clônage du visage utilise la détection de points caractéristiques sur les prises de vue frontale (face) et latérale (profil) et applique des déformations libres pour la modification de la forme. Une image de texture d'une seule pièce est ensuite générée automatiquement pour être appliquée à 360 degrés sur un modèle de peau polygonal. Le module de clônage du corps comprend deux étapes: (i)la spécification des points caractéristiques, qui permet la détection automatique de la silouhette quelque soit le fond de l'image, et (ii), une modification du corps en deux parties, utilisant respectivement les points caractéristiques et la silhouette du corps. Le modèle final intégré d'humain a un visage, des mains, des pieds et un corps directement animables et photoréalistes. Le résultat peut-être visualisé dans n'importe quel visualiseur compatible VRML.

Nous avons appliqué la méthodologie de clônage du visage dans trois extensions, qui sont (i) donner une structure d'animation à partir de cartes de profondeur, (ii), un système temps-réel de morphing 3D pour la visualisation et la génération de nouveaux modèles photoréalistes de visages, et (iii), l'adaptation du clônage du visage à la norme MPEG-4.

[2] http://www.census.gov/cgi-bin/ipc/popclockw

1. Les humains virtuels pour des applications temps-réel

Notre objectif est d'obtenir un modèle animable d'humain virtuel qui présente une apparence réaliste. Les aspects impliqués dans la modélisation réaliste d'humains virtuels pour les environnements temps-réels sont les suivants:

- l'acquisition des données de la forme du corps.
- la production d'une texture haute-résolution réaliste du corps.
- l'obtention d'informations fonctionnelles pour l'animation du corps (à la fois pour le visage et pour le corps).

Pour ce faire, nous avons développé une technique qui permet une acquisition simple de ce type de modèle d'avatar, avec la capacité d'être produit à bas coût et d'être animable correctement. Dans le contexte de cette problématique, nous proposons les questions suivantes et suggérons nos solutions:

Que produire ?

Nous nous intéressons à la modélisation d'humains virtuels avec leur structure d'animation. Le corps humain est composé de plusieurs parties comme le visage, le corps, les mains et les pieds. Les vêtements sont aussi un élément très important de l'aspect visuel extérieur d'un humain. Nous produisons un visage et un corps habillé animables et photoréalistes.

Quelles type de données en entrée ?

Il existe de nombreuses méthodes d'acquisition pour la modélisation, allant des scanners à laser aux appareils photographiques. Chacune présente ses avantages et inconvénients propres. Nous avons choisi d'utiliser des photographies comme données d'entrée. Nous utilisons des prises de vue frontale et latérale pour le visage et des prises de vue frontale, latérale et dorsale (dos) pour le corps. Nous avons aussi expérimenté les cartes de profondeur pour leur donner une structure d'animation.

Quel environnement pour obtenir les données ?

Il est bien sûr possible de construire une cabine dans une pièce spacieuse avec une configuration adaptée. Mais cela présente l'inconvénient majeur de restreindre grandement les utilisateurs potentiels ainsi que les localisations possibles du système. Nous utilisons de simple prises de vue faites avec des appareils disponibles dans le commerce et sans configuration particulière de l'environnement de prise de vue. Au lieu de rechercher une solution limitant l'environnement utilisable, nous fournissons à l'utilisateur une interface conviviale, qui permet même aux novices de fournir interactivement les informations nécessaires concernant le corps et le visage.

Quel degré d'automatisme du processus pour l'utilisateur ?

Des approches automatiques, semi-automatiques, et interactives offrent des compromis entre robustesse, durée du processus et difficulté pour l'utilisateur d'obtenir des résultats satisfaisants. Nous proposons un système automatique excepté pour quelques opérations d'initialisation réalisées interactivement, afin d'assurer la robustesse du système.

Quel niveau d'animation pour l'humain virtuel ?

Nous fournissons un humain virtuel ayant des capacités complètes d'animation du visage, du corps et des mains. Nous présentons notre propre système d'animation ainsi que ceux définis selon les méthodes standardisées H-ANIM et MPEG-4.

2. Clonage du visage

Nous décrivons ici notre système de clonage virtuel d'une personne. Etant donné que notre objectif est de fournir un humain virtuel réaliste pour l'animation temps-réel, tel qu'un humain dans la réalité, le critère le plus important est de créer un modèle du visage photoréaliste dont la géométrie - la répartition des sommets 3D - est optimisée pour l'animation et l'affichage en temps réel. Nous utilisons des photos comme données d'entrée et appliquons des modifications de forme sur un modèle générique dont les sommets de la surface sont répartis de façon optimale. Le visage 3D ainsi reconstruit est directement animable à l'aide de n'importe quel paramètres décrivant des expressions du visage. Tout visage 3D produit avec cette technique est “morphable” avec un autre permettant de produire de grandes populations de visages, ainsi qu'il sera décrit plus loin.

La méthodologie de clonage du visage comprend l'initialisation (seulement une fois pour l'ensemble des clonages), la détection des caractéristiques, la modification de la forme, la génération de la texture et sa projection. L'initialisation consiste tout d'abord a créer un modèle générique pour la forme du visage, la structure d'animation, les caractéristiques génériques et les polylignes caractéristiques. Ensuite, nous calculons les relations locales pour les déformations libres, qui seront utilisées pour la modification de forme. La base de données d'expressions est elle aussi préparée à l'avance.

2.1 Détection des caractéristiques

La détection de caractéristiques à partir d'images 2D (photographies) est une des étapes principales. Quand nous faisons référence à la détection de caractéristiques, nous voulons signifier la détermination de la position 2D ou 3D des points caractérisant les parties les plus visibles d'un visage comme les sourcils, le contour des yeux, le nez, les lèvres, la limite entre les cheveux et le visage, la ligne du menton etc. Certaines zones comme le front et les joues ne sont pas toujours facilement localisables dans une photographie, et nous les référençons comme points non-caractéristiques. Pour la détection des caractéristiques, nous procédons en trois étapes. Tout d'abord, nous localisons quelques points (caractéristiques-clés) interactivement sur la photographie. Ensuite, des transformations affines par morceaux sont appliquées pour déplacer les autres caractéristiques automatiquement. Enfin des contours actifs (snakes) structurés sont utilisés pour l'adaptation finale.

1. Les transformations affines par morceaux sont une sorte de déformations libres combinées avec différentes projections affines. Les points de contrôle de la projection affine sont localisés le long de deux polylignes caractéristiques continues déterminées par les points caractéristiques génériques. Ainsi, la continuité de la surface ou de la courbe est préservée quand les positions des points de contrôle sont modifiées. Après la localisation des caractéristiques-clés, les “piecewise affine transformations” sont utilisées pour assigner la position des autres caractéristiques au plus près de leur destination.
2. La méthode de contour actif ou “snakes”, est largement utilisée pour s'adapter à un contour dans une image donnée. Nous avons ajouté trois fonctionnalités supplémentaires au “snake” conventionnel. Tout d'abord, nous ancrons des points (ici, les points caractéristiques-clés) enfin de conserver la structure des points quand des “snakes” sont impliqués. Etant donné que la correspondance entre les points de contrôle du modèle générique et les points caractéristiques de la photographie doit être établie, la détection d'arêtes n'est pas suffisante. L'ancrage de certains points afin d'obtenir une position donnée précise contribue à établir cette correspondance. Ensuite, nos contours étant modélisés par des polylignes, nous utilisons un modèle de “snakes” discrets, avec des forces d'élasticité, de rigidité et d'“image” qui agissent sur chaque pixel ayant la couleur d'intérêt. Nous définissons différents ensembles de paramètres pour les cheveux et le visages selon les caractéristiques de leurs couleurs. Enfin, dans les situations où la couleur et l'opérateur de Sobel ne permettent pas d'obtenir une détection d'arête satisfaisante, nous utilisons alors une technique de multi-résolution permettant d'obtenir des arêtes fortes.

2.2 Reconstruction de la forme

Nous avons maintenant détecté les points caractéristiques sur les photographies. Nous disposons d'un ensemble de caractéristiques 3D correspondantes. La question est donc de savoir comment modifier le modèle générique qui contient plus d'un millier de points pour construire une surface individualisée lisse. Pour cela nous devons tout d'abord décider si nous faisons les modifications en 2D et combinons deux ensembles de modifications en 2D pour passer en 3D ou si nous faisons la modification directement en 3D. Il s'agit d'une question critique pour la manipulation des données en terme de tolérance d'erreur. Nous devons appliquer une méthode qui diminue la dépendance par rapport à des prises de vue parfaitement orthogonales, étant donné que dans la plupart des cas, il est très difficile d'obtenir des photographies parfaitement orthogonales.

Nous avons choisi d'utiliser l'algorithme de modification en deux dimensions le plus approprié pour limiter la dépendance à la contrainte d'orthogonalité des prises de vue et de reporter les modifications en 3D. Etant donné que nous utilisons seulement des points caractéristiques depuis les photographies, il est possible de les gérer de façon à réduire la dépendance par rapport à la contrainte d'orthogonalité. Nous proposons un algorithme qui combine les deux ensembles de caractéristiques 2D des vues frontale et latérale pour construire des caractéristiques 3D en privilégiant la vue frontale par rapport à la vue latérale.

Puis nous modifions le modèle générique à l'aide des caractéristiques 3D. Des déformations de forme libre (Dirichlet Free-Form Deformations) sont utilisées pour modifier la forme du modèle générique en utilisant les points caractéristiques 3D comme des points de contrôle pour la déformation. Les calculs importants pour obtenir les coefficients de déplacement des points du modèle à partir des points de contrôle sont effectués seulement une fois pour le modèle générique dans la phase d'initialisation. Les coefficients de déplacements sont sauvegardés et réutilisés à chaque nouvelle modification de forme. Il s'agit d'un processus d'approximation où seuls les points caractéristiques du modèle générique sont retrouvés précisément et tous les autres sont approchés. La forme initiale du modèle générique est donc très importante pour la qualité du résultat.

2.3 Application de texture automatique

Afin d'améliorer l'aspect photoréaliste des objets virtuels, nous utilisons la projection de texture générée à partir des photographies des objets réels. Pour les visages virtuels, la texture ajoute le grain à la peau ainsi que tous les détails de couleur du nez, des lèvres, etc. L'application de texture nécessite à la fois une image de texture et des coordonnées de texture. Les images d'entrée n'étant pas utilisables telles quelles pour l'application de texture, l'image de texture doit être générée.

Si deux photographies pour des vues de face et de profil sont prises simultanément dans un environnement avec deux appareils photos correctement installés pour obtenir des vues orthogonales, il est possible de créer la projection de texture en déterminant les coordonnées de textures appropriées dans chacune des deux images. Quoiqu'il en soit, notre objectif est d'ôter, autant que faire se peut, les restrictions par rapport à la prise de vue. Nous voulons pouvoir utiliser des photos prises avec un seul appareil en demandant au sujet de tourner sur lui-même de 90 degrés, ce qui change les conditions d'éclairage et la luminosité. Dans ce contexte, une simple combinaison de la vue de face et de la vue de profil ne fournit pas une texture continue à cause de la non-orthogonalité parfaite de la prise de vue et de la non-cohérence de la luminosité.

La génération de l'image de texture se fait en deux étapes:

1. Déformations géométriques pour compenser la non-orthogonalité des prises de vue: nous considérons que la prise de vue de face est plus importante que la prise de vue de profil. La vue frontale est donc conservée telle quelle et la vue latérale est déformée pour se conformer à la vue frontale. Deux sous-ensembles de points caractéristiques sont utilisés pour définir deux lignes (une pour la partie gauche et une autre pour la partie droite) dans la vue frontale de façon à conserver la partie la plus importante et la haute résolution de la vue frontale. Pour chaque ligne de l'image frontale, est associée une ligne caractéristique correspondante sur l'image latérale pour le “collage”.
2. Lissage des frontières entre les images pour compenser la non-cohérence de la luminosité: les trois images résultantes après le processus de déformation sont fusionnées avec la méthode de décomposition pyramidale. L'opérateur de Gauss est utilisé pour lisser les frontières causées par la non-cohérence de luminosité des deux prises de vue. Cette technique de multi-résolution est très utile pour retirer les frontières entre des images.

Pour assigner des coordonnées appropriées à chaque point de la tête dans l'image de la texture, nous projetons tout d'abord la tête sur trois plans, le plan frontal (x,y), le plan gauche (y,z) et le plan droit (y,z). Nous déterminons sur quel plan le point 3D est projeté à partir des lignes caractéristiques. Les projetés des points sur l'un des trois plans sont alors transférés dans l'un des espaces des caractéristiques 2D de l'image frontale ou latérale. Puis ils sont transférés dans l'espace 2D de l'image en utilisant l'inverse de la tranformation de normalisation. Enfin, ils sont encore une fois transformés dans l'espace de l'image combinée par la même transformation utilisée pour la déformation géométrique de l'image.

3. Clonage du corps

Nous présentons la méthode de clonage du corps complet. Ce système utilise des photos prise de face, de profil et de dos d'une personne avec n'importe quel système photographique sans nécessiter un fond particulier ou un contrôle des conditions de luminosité. Un corps générique d'une seule pièce spécifié au format VRML H-ANIM 1.1 est utilisé pour générer un humain virtuel individualisé. Le corps générique a le niveau d'articulation 3 avec la hiérarchie H-ANIM 1.1 complète comprenant 94 articulations et 12 sections de peau incluant la tête, les mains et les pieds. Chaque section de peau est associée à une articulation et est exprimée dans le repère local correspondant en tant que segment. Les sections de peau sont associées à une articulation du squelette correspondante qui transforme les coordonnées locales de chaque section i en coordonnées globales par la matrice homogène Mi correspondante.

3.1 Modèle initial de peau

Nous décrivons maintenant l'environnement de prise de vue pour prendre des photographies avec un seul appareil. Trois photos sont prises, de face, de profil et de dos. Dans ce contexte, il n'y a pas de correspondance exacte entre la prise de face et de dos, car nous demandons à la personne photographiée de tourner sur elle-même pour prendre la vue de dos après la prise de vue de face. La taille de la personne (avec comme unité le mètre) est utilisée pour comparer les trois prises de vue et effectuer une normalisation. Etant donné que le fond de l'image des prises de vue est quelconque et que la taille de la personne photographiée varie, nous utilisons une approche interactive de localisation des points caractéristiques sur les images. Cette simple localisation de points caractéristiques est utilisée par la suite pour les modifications du squelette, une déformation grossière de la peau et la détection automatique des arêtes.

Une fois que les points caractéristiques de l'enveloppe de la peau sont obtenus (même si la personne porte des vêtements, nous considérons que les contours des vêtements sont proches des contours de la peau), nous pouvons estimer le squelette. Par exemple, le r_elbow doit être localisé a peu près au milieu entre l'extrémité droite de l'épaule et la base du poignet droit. Nous appliquons ici des transformations affines par morceaux et une interpolation barycentrique pour déterminer la localisation des articulations du squelette à partir des points caractéristiques. Les 94 articulations sont réparties entre deux sous-ensembles: les articulations “mobiles” qui dépendent des points caractéristiques et celles qui sont modifiées par des transformations affines par morceaux contrôlées par les articulations mobiles et la hiérarchie du squelette.

Chaque section de peau i est ensuite connectée à une articulation du squelette par une matrice Mi pour être exprimée en coordonnées globales. La matrice est déterminée à partir des changements d'échelle, des translations et des rotations de l'articulation et de ses enfants dans le squelette. A cette étape, la continuité des sections de peau n'est pas garantie. Nous appliquons alors une déformation de forme libre à partir de points caractéristiques pour assurer approximativement la continuité entre les sections. Les points de contrôle sont disposés à des positions prédéfinies caractérisant la forme du corps. Ainsi la surface de la peau peut-être modifiée en déplaçant les points de contrôle qui sont définis soit à partir de points caractéristiques sur les prises de vue frontale ou dorsale, soit à partir de relations. Par exemple, le bord entre le “front_torso” et le “right_upper_arm” peut être déterminé à partir des point caractéristiques en bas à droite du cou et de l'extrémité de l'épaule droite sur les deux images. En outre, plusieurs points de contrôle sont localisés aux frontières entre deux sections, de sorte à préserver la continuité quand la posture du corps générique est modifiée. Ces points de contrôle sont utilisés par les transformations affines par morceaux. Nous appliquons les déformations séparément par section de peau.

3.2 Modifications fines à l'aide des arêtes

La précision du modèle de peau initial n'est pas suffisante, et nous devons la modifier à partir de la silhouette extraite depuis les photographies. Nous détectons la silhouette du corps humain dans les photographies et nous adaptons la peau à la silhouette. Nous proposons un algorithme simple utilisant des points caractéristiques sur le corps qui servent de guide pour l'extraction des arêtes. Pour obtenir la silhouette, nous utilisons l'algorithme suivant:

1. Application du détecteur d'arêtes Cany et adaptation

2. Evaluation des connections de paires d'arêtes. Une fonction d'évaluation de connection est définie par des paramètres tels que l'angle entre deux segments, les magnitudes des arêtes définies par le détecteur d'arête de Cany, et les points caractéristiques localisés sur les images.

3. Extraction de la silhouette. Le problème peut être ramené à un problème de recherche de chemin. Une fonction d'évaluation Ep est définie afin de permettre de quantifier la pertinence d'un chemin. Les procédures suivantes sont utilisées:

a) choisir un segment d'arête.

b) trouver un segment d'arête approprié et s'y déplacer.

c) répéter l'étape 2) jusqu'à ce que le segment d'arête atteigne un point caractéristique signalant la fin d'un chemin.

d) Estimer la pertinence du chemin en évaluant la fonction Ep

e) Répéter les étapes 1) à 4) pour chaque segment d'arête. Le bon chemin est finalement déterminé en trouvant celui pour lequel Ep est maximum.

Une silhouette exacte est détectée grâce à la méthode proposée. Après l'ajustement du squelette, la modification grossière de la peau dans les étapes précédentes, chaque section de peau est ajustée selon l'orientation et la taille correcte. La silhouette obtenue est utilisée pour modifier les contours qui définissent la surface de la peau. Un corps humain photoréaliste peut ainsi être construit grâce à notre méthode sans nécessiter d'environnement ou d'équipement particulier.

3.3 Application de texture

Nous utilisons seulement les vues de face et de dos pour l'application de la texture, ces deux vues étant suffisantes pour couvrir l'ensemble du corps (excepté la tête). Pour l'application de texture, nous assignons des coordonnées de texture au point de la surface de la peau. Etant donné que deux images sont utilisées, il faut partitionner les polygones de la surface de la peau pour les répartir entre la vue de face et la vue de dos. La partition se détermine à partir du produit vectoriel entre le vecteur de vue et le vecteur normal au point de la surface. Pour déterminer les coordonnées de texture, nous utilisons une projection sur le plan (x,y) dans l'espace de l'image. Nous appliquons alors la procédure suivante:

1. Déformation du corps à partir des prises de vue frontale et latérale.
2. Projection inverse des points de vue frontal et dorsal dans le plan de la prise de vue dorsale pour déterminer les coordonnées de texture.
3. Déformation du corps à partir des prises de vue dorsale et latérale.
4. Projection inverse des points de vue frontal et dorsal dans le plan de la pris e de vue frontale pour déterminer les coordonnées de texture.

Le corps individualisé dispose alors de la projection de texture correcte pour les prises de vue frontale et dorsale. Cependant, il reste des erreurs au niveau de la frontière entre les prises de vue frontale et dorsale dues au processus de digitalisation et aux conditions d'illumination différentes entre les deux prises de vue. Pour les corriger, nous modifions tout d'abord la couleur dans le voisinage de tous les pixels des arêtes. Grâce au précédent processus de détection des arêtes, nous savons de quel côté du contour se trouve l'arrière-plan. Nous cherchons alors le long de la perpendiculaire à l'arête pour trouver un pixel du premier plan et utiliser sa couleur pour l'assigner aux pixels dans le voisinage de l'arête. Comme le montre les résultats, cette méthode simple permet de retirer les effets de perturbation dus à la digitalisation. Ensuite, nous lissons les textures frontale et dorsale pour éliminer les différences entre les deux images. Grâce aux points caractéristiques définis dans une précédente étape, nous pouvons reconnaître sémantiquement les différentes parties du corps humain et ainsi établir les correspondances entre les deux images. La correspondance entre les pixels est déterminée grâce à la longueur des bords. Dans le voisinage des deux pixels des vues frontale et dorsale, nous utilisons une combinaison linéaire.

4. Un humain virtuel avec un visage et un corps animables

Le visage et le corps sont construits séparément. En effet, même si la taille du visage est petite comparée au corps, nous devons la représenter avec un maximum de détails et une haute résolution, car il y a souvent besoin de zoomer sur le visage pour visualiser l'animation et la communication faciale. Nous utilisons donc des méthodologies et des images différentes pour le visage et le corps. Une fois le corps et le visage construits séparément, ils doivent être connectés ensemble correctement afin d'obtenir une surface contiguë représentant le corps entier. La position et la taille du visage sont contrôlées sur les prises de vue frontale et dorsale à partir de points caractéristiques associés au visage et définis à cet effet. La tête est ainsi déplacée sur le corps et mise à l'échelle pour s'y adapter. Nous utilisons enfin une méthode de “collage” automatique où les points du cou et de la tête les plus proches sont déterminés en utilisant les données du corps générique.

La posture par défaut du corps pour H-ANIM est définie pour l'animation. La posture sur nos photographies est légèrement différente pour permettre la détection des arêtes. Une fois le corps construit, il faut donc corriger sa posture en modifiant les angles des articulations des bras et des jambes.

5. Construire une structure d'animation à partir de carte de profondeur (range data)

Nous proposons une extension de la méthode de clonage du visage permettant de construire une structure d'animation à partir de cartes de profondeurs statiques, qui contiennent les informations de forme et de texture. Cette extension suit globalement les mêmes étapes que le clonage du visage: détection des caractéristiques, modification de la forme d'un modèle de visage générique et application de la texture. Il existe cependant quelques différences entre les types de données d'entrée, photographies ou cartes de profondeurs. La principale d'entre elles réside dans le mode de représentation de l'information de profondeur entre les deux types de données.

Nous procédons tout d'abord à l'extraction et à la détection des caractéristiques 2D dans l'image de texture (la vue frontale seulement) de la carte de profondeur. Cette procédure est réalisée en deux temps. Tout d'abord des transformations affines par morceaux contrôlées par des caractéristiques clés sont utilisées pour placer les autres caractéristiques de façon automatique. Ensuite des “snakes” structurés sont appliqués pour l'adaptation finale. Chaque fois qu'un point de la vue frontale est détecté, sa profondeur z est calculée automatiquement. Tout d'abord, nous collectons un nombre n de points les plus proches dans le plan (x,y) et calculons la profondeur du point par sommation en utilisant différents poids calculés à partir de la distance inverse. Certains points caractéristiques sont définis interactivement sur les vues latérales car ils ne sont pas fournies par les cartes de profondeur. D'autre part, la plupart des méthodes pour extraire des cartes de profondeur ne permettent pas d'obtenir la forme des cheveux à cause de la réflexion très élevée de leur structure.

Les modifications de forme à appliquer au modèle de visage générique sont déterminées à l'aide de déformations de forme libre (Dirichlet Free-Form Deformations) en utilisant les points caractéristiques comme points de contrôle. Les modifications par déformations libres sont uniquement basées sur des points caractéristiques, ce qui implique que seules les caractéristiques sont utilisées pour la modification. L'information de profondeur est disponible pour les points caractéristiques et pour tous les points de la carte de profondeur. Il est donc possible d'utiliser d'autres points que les points caractéristiques. Nous appliquons une étape supplémentaire pour obtenir aussi la position précise pour d'autres points de la carte de profondeur:

1. Certains point caractéristiques sont choisis pour réaliser une modification précise.
2. La triangulation de Delaunay de ces points caractéristiques est construite, et tous les points non-caractéristiques qui se trouvent à l'intérieur sont collectés.
3. les points à l'intérieur de chaque triangle sont projetés sur un plan afin de déterminer leur profondeur dans la carte de profondeur.

4. A nouveau, les n plus proches points sont collectés, puis une fonction de la distance inverse est appliquée pour obtenir les coordonnées correspondante dans la carte.

Pour l'application de la texture, il est possible d'étendre certaines parties de l'image frontale. La localisation des bords entre les cheveux et le visage étant connue, une nouvelle image avec les cheveux est créée et utilisée comme une prise de vue latérale. Le même algorithme utilisant les prises de vue orthogonales peut-être utilisé.

6. Système temps-réel de morphing de visages 3D

Nous proposons une série de méthodes pour la conception d'un système de morphing 3D de visages, qui peut-être utilisé aussi bien pour la visualisation temps-réel de morphing 3D que pour générer automatiquement et massivement de grandes populations de visages photoréalistes immédiatement utilisables pour l'animation. Pour la génération simple et rapide de nouveaux humains virtuels, nous utilisons tout d'abord plusieurs clones virtuels construits à partir du même modèle générique. Nous pouvons ensuite générer une population de visages humains à partir de ces clones en utilisant une méthode de contrôle rapide et intuitive.

L'idée de combiner le morphing d'images 2D avec le morphing de surface 3D a été très peu exploitée. Il semble pourtant que ce soit une approche naturelle pour obtenir d'excellents résultats avec peu d'efforts dans certains contextes. En outre, en l'appliquant au cadre du clonage du visage, les interactions avec l'utilisateur peuvent être évitées. Quand une personne est morphée en une autre personne en 3D nous devons gérer les modifications de la forme et de la texture. Les visages créés par la méthode de donage partageant tous la même topologie, il est très facile de modifier la forme 3D des visages par une simple interpolation linéaire des coordonnées 3D. Par contre, il est beaucoup plus compliqué de réaliser le morphing entre les images de texture, ce qui nécessite d'établir des correspondances entre des caractéristiques précises de chaque image. Nous proposons une méthode simple, consistant à “dessiner” sur les surfaces 3D pour contrôler le morphing entre les deux images. Pour générer une texture intermédiaire, nous morphons triangle par triangle du visage en 3D. Les parties de l'image qui sont utilisées dans la projection de la texture sont triangulées par projection des triangles du visage 3D. La triangulation de l'image de texture est optimisée avec des triangles fins dans les zones de grande courbure et de grande variation de couleur et de triangles plus larges dans les zones plus cohérentes, ce qui facilite l'obtention d'une interpolation continue entre les images. A partir des triangles ainsi obtenus, l'interpolation des coordonnées barycentriques est utilisée pour morpher les images. Chaque pixel d'un triangle de l'image intermédiaire a une couleur qui est déterminée en combinant les couleurs des deux pixels de même coordonnées barycentriques dans les triangles correspondants dans l'image de départ et l'image de fin respectivement. Pour obtenir une image lisse, une interpolation bilinéaire entre les quatre pixels voisins est effectuée.

Notre approche de morphing peut se généraliser naturellement à plusieurs visages virtuels. L'interface de l'application présente deux fenêtres, une pour contrôler le pourcentage d'entrée entre les n visages et une pour visualiser le résultat. Deux options sont possible: morphing instantané ou morphing continu. Le morphing instantané permet de sélectionner un pourcentage d'entrée dans un polygone a n sommets pour n visages virtuels donnés. Pour le morphing continu, le résultat varie continûment le long d'un chemin défini dans le n-polygône. La partie calcul du processus se limite au calcul et à l'interpolation des coordonnées barycentriques, le temps de calcul est très rapide. Il existe encore une possibilité d'activer/de désactiver les variations de la surface et de la texture séparément, ce qui permet de définir des pourcentages d'interpolation différents entre les surfaces et les images. La combinaison de différents pourcentages pour la surface, la texture et les expressions permet de générer des résultats variés et de produire de nouveaux visages de façon intuitive.

7. Adaptation à la norme MPEG-4

Nous avons adapté notre système de clonage du visage pour la norme MPEG-4 en utilisant les “Face Definition Parameters” (FDPs) comme caractéristiques pour reconstruire le visage. La méthode consiste à extraire les FDPs sur les photographies, ce qui permet de générer un visage à partir de la déformation d'un modèle générique. La projection de texture est déterminée automatiquement à partir de la combinaison des deux photographies. Le processus est le même que pour le clonage du visage. Lors de nos expérimentations, nous avons rencontré des problèmes pour détecter les FDPs sur les images.

8. Implémentation du système

Nous indiquons ici comment les systèmes décrits plus haut ont été implémentés:

Les principaux éléments de base de l'implémentation sont:

- Interface graphique: RAPIDAPP (SGI)/Visual C++ (PC)
- Librairie graphique: OpenInventor
- Langage de programmations: C++
- Liens avec d'autres librairies MIRALab: DFFD et animation faciale
- Liens avec d'autres librairies: librairie JPEG

Le système de clonage du visage comporte trois gestionnaires, chacun étant responsable d'un "viewer" OpenInventor et qui sont le gestionnaire des caractéristiques 2D, le gestionnaire des formes 3D et le gestionnaire de texture 2D. Le système commence par charger deux photos. Le gestionnaire des caractéristiques 2D permet d'effectuer la normalisation et la localisation des caractéristiques. Le gestionnaire de formes 3D est ensuite utilisé pour obtenir la forme de visage individualisée, puis le gestionnaire de texture 2D génère les images de la texture permettant d'assigner la couleur de la peau à la forme du visage. Au cours de notre expérimentation d'adaptation à MPEG-4, nous avons rencontré certaines difficultés pour obtenir des résultats satisfaisants de manière totalement automatique à partir FDPs. Ces difficultés concernent particulièrement l'application de la texture entre les cheveux et le visage et sur les ailes du nez et sont dues à l'absence de points caractéristiques. L'interface conviviale permet d'optimiser l'obtention des résultats. En mode normal, le gestionnaire de caractéristiques fonctionne en mode semi-automatique, et le gestionnaire de formes et de textures fonctionnent eux en mode automatique complet. Cependant, nous avons ajouté une méthode de feedback visuel permettant de corriger les éventuelles petites erreurs. Cette option permet aussi de créer une forme encore différente de celle des photos.

Le système de clonage du corps propose un interface utilisateur convivial organisé en deux fenêtres principales et quatre gestionnaires. Le viewer 2D est contrôlé par le gestionnaire de caractéristiques et le gestionnaire d'arêtes tandis que le viewer 3D est contrôlé par le gestionnaire de formes et le gestionnaire de texture. Le système commence par charger trois photographies. Le gestionnaire de caractéristiques permet ensuite de réaliser la normalisation et la localisation des caractéristiques avant de détecter les arêtes grâce au gestionnaire d'arêtes. Le gestionnaire de formes produit alors la forme du corps individualisé en deux étapes puis le gestionnaire de texture construit les images de texture. La partie la plus délicate du système de clonage du corps n'est pas celle des caractéristiques comme pour le visage, mais la détection automatique des arêtes. Nous avons décidé de pouvoir utiliser un fond arbitraire pour les photographies et de faciliter la détection des arêtes avec l'aide de points caractéristiques. Parfois, la détection des arêtes entre les jambes ou les bras n'est pas évidente et il n'est pas facile de déterminer où les caractéristiques sont localisées. L'interface graphique avec son feedback visuel est alors utile pour ajuster les caractéristiques afin d'obtenir les arêtes de façon satisfaisante.

9. Conclusion

Nous avons présenté la méthodologie et les algorithmes développés pour concevoir un environnement intégré de modélisation assistée par ordinateur d'humains virtuels animables. Au cours de nos recherches, nous avons développé un système de clonage du visage et un système de clonage du corps à partir de photographies. Notre système de clonage du visage a été mis en application dans plusieurs extensions telles que l'utilisation de cartes de profondeur (range data), l'adaptation à la norme MPEG-4, et un système dynamique de morphing 3D.

Les principales contributions apportées par notre recherche sont:

- . L'utilisation de données d'entrée aussi simples que des photographies a été la première priorité pour la conception du système, ce qui permet par exemple de générer des jumeaux virtuels simplement à partir de fichiers images reçus par email.
- . Une intégration complète du corps et du visage a été réalisée à partir de cinq photographies.
- . Une méthodologie permettant de travailler avec différents type de données d'entrée a été développée pour le clonage du visage.
- . Un interface utilisateur convivial permettant à tout utilisateur de mettre en oeuvre le système comme il l'a été démontré au cours de plusieurs démonstrations en public.

La dernière et aussi la plus importante contribution consiste à rendre les gens conscients de ce que les mondes virtuels sont proches d'eux. Nous pourrions alors les appeler des Narcisses[3] actifs et immortels.

[3] Narcisse: très beau jeune homme qui était tombé amoureux de son propre reflet dans une fontaine et qui est mort de sa passion pour lui-même. Une fleur a poussé à l'endroit où il est mort et il lui a été donné son nom.

Preface

This work has been done at MIRALab, University of Geneva, under the direction of Prof. Nadia Magnenat-Thalmann. Since its foundation by Prof. Nadia Magnenat-Thalmann in 1989 at the University of Geneva, MIRALab has dedicated most of its research efforts in the field of virtual human simulation. The work presented here is a contribution to the global project of the laboratory. As shown in *Figure 1-1*, the work of human modeling indicated inside colored regions is the basis for wide applications from the 3D clothes extraction to the human animation and communication.

The thesis is organized such as

- Chapter 1 introduces the aims of this thesis.
- Chapter 2 describes existing methods by comparing them to know the advantage and disadvantage.
- Chapter 3 explains our methodology for the face cloning from photograph input.
- Chapter 4 devoted to an extension of the face cloning methodology to give animation structure on range data.
- Chapter 5 describes an application of the face cloning methodology used both for the real-time visualization of 3D morphing and also for the generation of large populations of photo-realistic faces.
- Chapter 6 describes an adaptation of the face cloning methodology to the international standard MPEG-4.
- Chapter 7 explains our methodology for the body cloning from photograph input.
- Chapter 8 devotes to show how the implementation is done in practice.
- Chapter 9 finally concludes the thesis.
- Appendix A shows the feature names used throughout the thesis.
- Appendix B explains the GI menus to help how the user can control the systems.

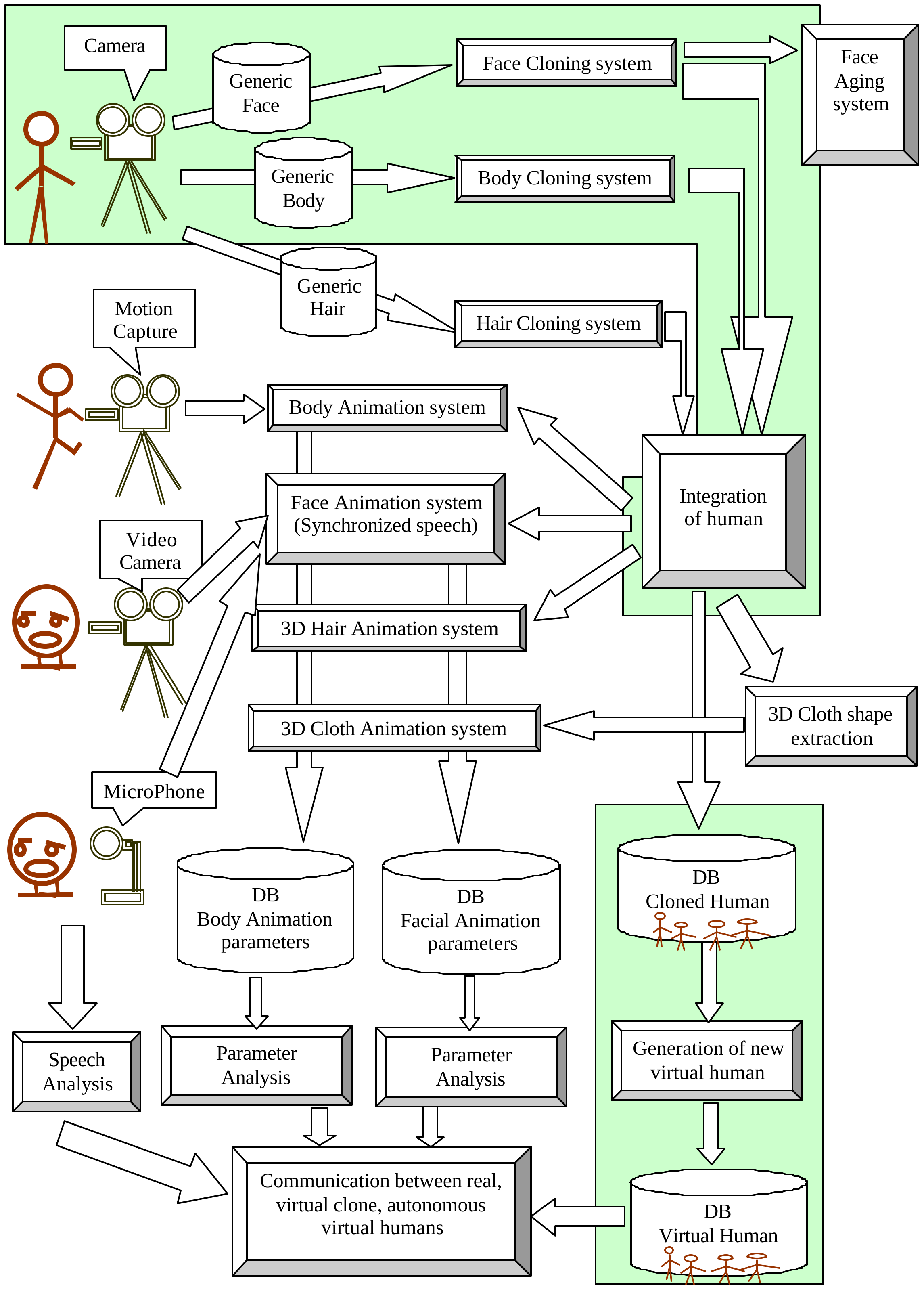


Figure 1-1: The global project of the MIRALab for virtual human simulation. The subparts in shady regions are the modeling parts covered by this thesis.

Contents

Lists of Figures

Lists of Tables

Chapter 1 Introduction

There is no landscape that we know as well as the human shape. The less than two meters high containing the features is the most intimately scrutinized piece of territory in existence, examined constantly, and carefully, with far more than an intellectual interest. Every detail of the nose, eyes, mouth, arms, legs, breast, torso, hands, and feet, every regularity in proportion, every variation from one individual to the next, are matters about which we are all authorities.

- Version 1.0 by Gary Faigin, from *The Artist's Complete Guide to Facial Expression*
- Version 2.0 by WonSook Lee

In recent years, modeling virtual human has attracted more and more attention from both the research and industrial community. 3D-modeling of human has wide applications from Virtual Reality application (requires real-time) to medical application (requires high accuracy). With the growing power of computer speed and multimedia ability, people would like to have their virtual counterpart as 3D data on the computer screen and utilize it for various applications as follows:

- human-machine interface
- advanced multimedia
- augmented reality
- immersive Virtual Reality
- simulation of human behavior with virtual human
- medical application
- communication (through network)
- multi-media games

There are two basic types of techniques for obtaining 3D human models, according to the different requirements for the models. The first techniques focuses on the accuracy and precision of the obtained object model shapes, such as those used in CAD systems for industrial purpose or medical application. The second techniques concentrate on the visual realism and speed for animation of the reconstructed models, such as those used in Virtual Reality applications.

When we have to place importance on the accuracy of the shape concerning the first type, there are various approaches either using sculpture, a laser scanner, a stereoscopic camera, an active light striper, or video stream. It concerns precision and accuracy of shapes and often produces large number of points. Most of them have limitations for accesses of often high-priced hardware and lack of functional structure for (real-time) animation.

On the other hand, systems using the second type of techniques focus on more practical aspects such as how cheap the hardware is and how easy it is to use. These techniques are usually model-based. There are several approaches to the reconstruction of either a face or a body from photographs. These approaches concern mainly visual realism using a high quality image input. Some methods take a generic model and then both structural and shape information extracted from photographs is used to modify the generic model while others use silhouette information in several views to reconstruct the shape. These approaches are simple and efficient, but the shape is not as accurate as the one from a laser scanner, for example. However for the Virtual Reality application, usually the fast animation capacity with efficient shape representation with less numbers of points compensate with the accurate shape.

In this thesis we focus more on real-time applications since it is no longer fantasy to imagine that one can see herself/himself in a virtual environment moving, talking and interacting with other virtual figures or even with real humans. By advances in algorithms and new developments in the supporting hardware, this fantasy has become a reality. In addition, newly defined MPEG-4 supports the standard parameters for communication through network in real-time.

1.1 Virtual human for the real-time application

We address how to acquire an animatable human data with a realistic appearance. The issues involved in realistic modeling a virtual human for the real-time application purposes are as follows:

- acquisition of human shape data
- realistic high-resolution texture data
- functional information for animation of the human (both face and body)

It is our goal to develop a technique that enables an easy acquisition of the avatar model having the ability to be produced at a low cost and animated properly.

We give the following questions and suggest our solutions.

What to produce?

It is more a question of subjects. We are interested in a human modeling with animation structure. The human body is composed of several parts such as a face, body, two hands and two feet. Also the clothes is a very important component of human out-looking. We produce a photo-realistic animatable face and body with clothes textured. The hands and feet will be approximated since their exact shapes are not needed to recognize people with a few exceptional cases.

What is the input data?

There are many possible methods to acquire input data for modeling from high-end laser scanner to low-end still photographs. Each of them has own advantages and disadvantages. In Table 2-2, we give detailed comparison among different equipment in the sense of cost, time, and result. We select to use photographs as input, but also use range data set if it is needed and available.

For the photograph input, we will use the frontal view and the side view of a face while the frontal view, the side view and a back view are used for a body. Photographs of the whole body cannot provide sufficiently high-resolution for facial information in order to construct a good face model and further facial animation. That is why we take two photographs for focusing on the face only besides three whole body photographs.

For the range data input, we give animation structure on the static range data for a face part, which has shape and texture information. For our experiment, we use range data produced by stereoscopic cameras.

What is the environment to get the input data?

There are possibilities to build a special booth in a spacious room with a special setting. However in this case, it gives a great limitation to the potential places and users. With the increasing popularity of Internet, we suggest to remove any kind of limitation for the environment to get input, here photographs, so that any person can produce the input data. So we use simple snapshots with commercial cameras without any special environment. Instead of seeking a solution by using special environment, we provide a user-friendly interface, which allows non-expert users to interactively hint to the system certain important information about the human face and body. In this way, with a little amount of user interaction, we achieve more flexibility in using the system.

How much automatic is the process for users?

Fully automatic, semiautomatic and interactive solutions give trade-off between how robust it is, how long it takes and how easy it is for users to get satisfactory results. We provide an automatic system except for a few interactions at the beginning to guarantee robustness of the system.

How much can we animate the virtual human?

To have a simple movement of body without any facial expressions does not interest us. We provide a virtual human, wh ich has full animation capabilities for face, body and hand parts. We discuss not only our animation systems inside MIRALab throughout the sections and also the newly standardized MPEG-4 and H-Anim for both modeling and animation.

How accurate is the result?

We do not aim to produce a virtual human with the *millimeter* accuracy. We focus more on the easy input and realistic looking with reasonably acceptable error. We discuss the error measurements in sections for cloning methodology from photographs. We perform the experiment with two kinds of input data, one from the laser scanned data and the other from the real person’s data. The amount of error is acceptable for our purpose.

Chapter 2 State of the Art

Equipment such as 3D scanners able to capture the 3D shape of human bodies seem to be an appealing way to model human bodies. Although the method seems as simple as taking a photo, the approach is not so straightforward and the resulting human body shape can not directly be used inside a Virtual Reality environment. The manipulation of such a device has various levels of difficulties to get acceptable results. For example, defining the appropriate settings of the device is generally not straightforward. The main drawback lies in the post-processing required getting a usable virtual actor. First, in order to get a proper 3D shape, the result requires a geometric modeling post-process, particularly in terms of the amount of geometric data and surface correction, such as smoothing surfaces or filling holes. In order to animate the 3D shape inside a Virtual Reality environment, it has to be combined with the animation structure. This requires additional information that the scanning process can not provide such animating the face requires having eyes independent from the rest of the head. When the face is built using a 3D scanner, first, only the visible parts of the eyes are modeled, second, they make one single surface with the rest of the face.

In this chapter, we describe the state of the art in two main categories, one for focusing on the face object and the other on the body. There are many features in common for two objects in their methodologies, so we describe more in detail for the face part and then describe things, which are especially for the body part.

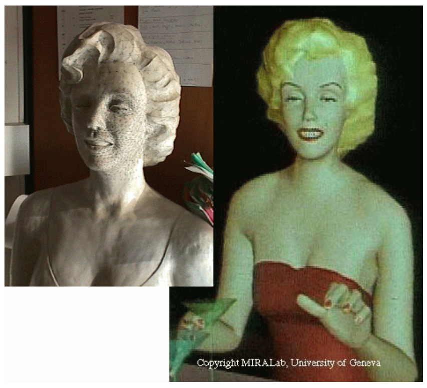
Face

3D-modeling of the human face has wide applications from videoconference to facial surgery simulation. To clone a real person has been even more interesting in today's virtual world. However, cloning a real person's face has practical limitations in the sense of how long it takes to get a result, how simple the equipment is, and how much realistic looking can be obtained. The diversity of facial forms in terms of sex, age, and race is enormous. Unlike other objects, human face modeling is considered, as one of the most difficult subjects since everyone knows how the face looks like, which makes hard to make it very realistic and believable. In addition the face modeling is a basis for the facial animation in a virtual world, in a way that three terms such as shape, texture, and animation structure be matched exactly to produce the convincing visualization.

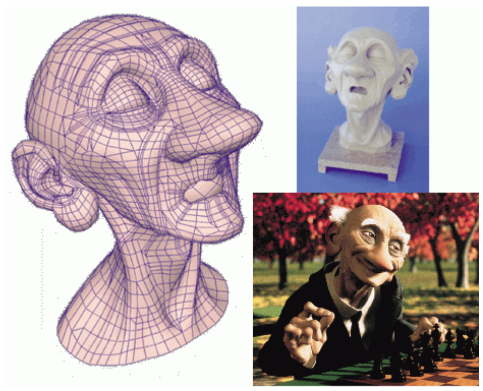
Even though many methods claim to be fully automatic, there is still a lot of user interaction for preparation (e.g. sticking markers on a face properly) and trial-error procedures for individual dependent parameters. We give the existing methods to generate a clone from a given input; such as simple equipment like cameras, more sophisticated equipment or a rather complicated algorithm. We consider practical points of view when we describe several methods one by one in detail in the next sections 2.1, 2.2, 2.3, 2.4, 2.5. The overall comparison is given in Section 2.6. Since the animation functional structure can be constructed when we get shapes or can be added as post-processing, we also discuss the existing methods in Section 2.7 to give animation structure once when we have a static shape.

Before starting, we define the terms such as "active" and "passive". When we refer that the input is achieved in an "active" way, we mean that we actively make the input such as we have the human subject in our special environment and we process to get the input data. When we refer that the input is achieved in a "passive" way, we use input data available already. When we say that the animation is done in an "active" way, it means we can generate the animation as we wish. When we say the animation is done in a "passive" way, it means we have animation, but we just play without controlling it.

2.1 Plaster model



(a) Marilyn Monroe [Magnenat-Thalmann 87]



(b) Geri [DeRose 98]

Figure 2-1: The plastic models for building 3D models in virtual worlds.

Magnenat-Thalmann et al. [Magnenat-Thalmann 87] used plaster models built in the real world. Selected facets and vertices marking on the models are photographed from various angles to be digitized. Then all the 2D coordinates are combined to produce 3D data of the face. Here the reconstruction approach requires a mesh drawn on the face and is time consuming, but can obtain high resolution in any interested area. Pixar's animation character "Geri" [DeRose 98] is also sculpted, which was then digitized and used as a basis for creating the 3D model. "Geri" is not a virtual clone of somebody in a real world. This plaster model method is also useful to create an imaginative character. It is an automatic in terms of computer process, but to make a plaster model is not the job for a non-artist. To give animation structure on the 3D-virtual model, we provide either special information as marks on the plaster model or perform the process to give animation structure as extra step. This plaster model method is still widely used in industry and it seems to reflect that people still feel more confidence to touch the objects in the real world than to use a mouse and keyboard for the virtual world.

2.2 Photographs (arbitrary)

There are many special needs to make a clone of a person who is not available to be photographed, for instance the late actress Marilyn Monroe. With given 2D photographs, we need to know the way to get 3D. The difficulty of making 3D from arbitrary photographs comes from the fact that sometimes only one photograph exists. Or several photographs are available, but they are taken at different times and environment, so that human intuition is needed to match among several photographs to get 3D data. To get a well-matched shape, we need to do a time-consuming manual job with help of a user-friendly graphical interface or heavy pixel comparison calculation with manual setting for various parameters for the photographing environment with a large set of generic head database.

2.2.1 Interactive deformation

With one or several photographs, we approach using the sculpting method in a similar way as we do in the real world. Software *Scupltor* [LeBlanc 91], dedicated to the modeling of 3D objects, is based on local and global geometric deformations. Adding, deleting, modifying, assembling triangle meshes are the basic features provided by *Sculptor*. Real-time deformations and manipulation of the surface gives the designers the same facilities as with real clay or wax sculpting. There are two ways to create a face. One way is to start from a template head, this therefore accelerates the creation process. The second method from scratch, the designer can model half of the head and use a symmetric copy for the other half. At the end, small changes should be made on the whole head because asymmetric faces look more realistic. The animation structure is given in the same time for sculpting or as a post-processing.

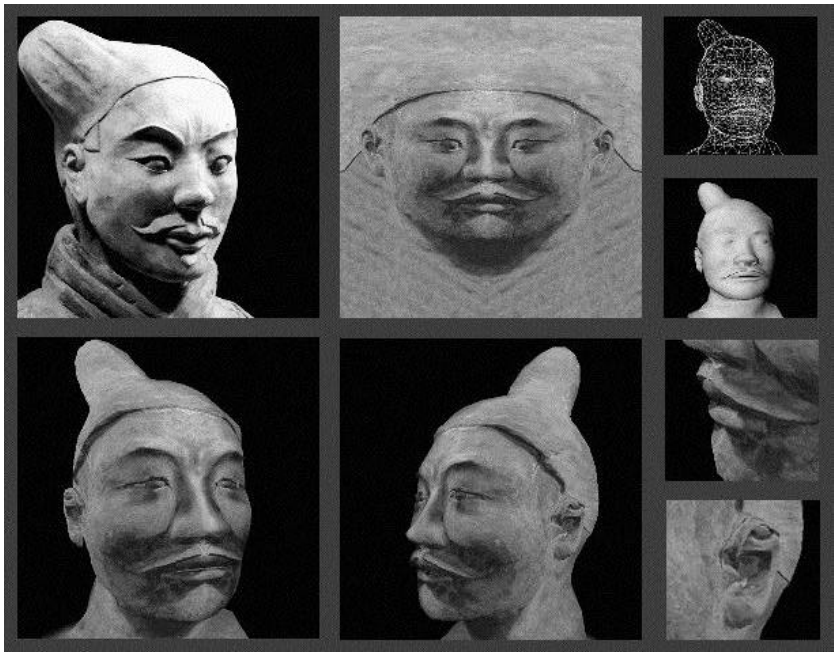


Figure 2-2: Head creation from a template in Sculptor and TextureFit program.

2.2.2 Interactive texture mapping

Texture mapping is a well-known low-cost method in computer graphics for improving the quality of virtual objects by applying real images onto them. For virtual humans, the texture can add a grain to the skin, including the color details like color variation for the hair and mouth. These features require correlation between the image and the 3D object. A simple projection is not always sufficient to realize this correlation: the object, which is designed by hand, can be slightly different from the real image. Therefore an interactive fitting of the texture is required. A program allowing the fitting of the texture according to features of the 3D object is called *TextureFit* [Sannier 97]. This enables the designer to interactively select a few 3D points (key points) on the object. These key points are then projected onto the 2D image and the designer can adjust the 2D texture coordinates on the 2D image. Then other points inside Delaunay triangles of key points are automatically calculated to get 2D texture coordinates using Barycentric coordinates. The projection can be chosen and set interactively, hence the designer is able to adjust these key points to their correct position on the image. Figure 2-2 shows an example of input photograph and the resulting head produced using *Scupltor* and *TextureFit* in MIRALab-Univ. of Geneva. The upper left image is a photograph of the real terra-cotta soldier. The upper middle image is the texture created from the photograph by a designer using a commercial program PhotoShop. The other objects are snapshots from the 3D model.

2.3 Generic database (arbitrary photographs)

Blanz and Vetter [Blanz 99] make a morphable model for the synthesis of 3D faces. They built 200 generic models using a laser scanner. Every head shares the same structure and each has 70,000 vertices. They analyze input image (one photograph or scanned data) and use statistics to find a 3D shape. This approach does not use any deformation of shape as shown in *Figure 2-3*. It treats it as a correspondence problem between a given image and generic models looking for parametric description of faces. Figure 2-3 shows the global idea of a face in the space of generic faces. Each generic head i has two kinds of vectors, shape $S_i(x, y, z)$ and texture $T_i(r,g,b)$. The problem is how to find out the linear combination of a large number of 3D face scans, i.e. to find a_i and b_i in $S_{mod} = å a_i S_i$ and $T_{mod} = å b_i T_i$.

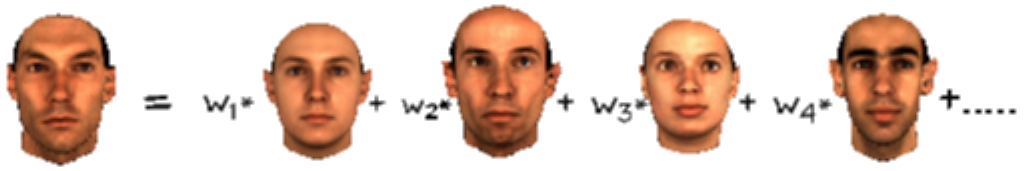


Figure 2-3: A face can be possibly found as a linear combination of generic faces.

It transforms to an orthogonal coordinate system using Principal Component Analysis (PCA). Beside shape and texture parameters, we need to have rendering parameters r_i roughly estimated by the user, which is a trial-error part to get the approximated environment for the given input image. The rendering parameters include camera position, object scale, image plane rotation and translation, intensity of ambient light, intensity of directed light. This becomes a problem how to minimize error function E of noise, position, texture, rendering parameters by iteration. In their experiment, 10,000 iteration and 50 minutes calculation time on an SGI R10000 processor are spent beside the time to build 200 generic models and trial-error parameter finding process. It is necessary to extend the dimension of faces for covering new ethnic groups by adding more faces on the generic database. The resultant faces have the same resolution as the generic faces. They use laser scanner to build the generic models with the same structure. To make all laser scanned faces morphable, which means all faces share the same structure, is not straightforward job. The most tedious job is to clean the noise on the laser scanned data. We discuss about laser scanner in later section about the advantage and drawbacks, which produce possible noise for hairy and dark area of a face. The animation structure need to be added as a post-process to the resultant model or pre-process for the every generic model as discussed in Section 2.7.

2.4 Features on photographs (organized) and a generic model

There are faster approaches to reconstruct a face shape from few photographs of a face. It utilizes a 3D existing face model and very little information (only feature points) from photographs differently from the method in Section 2.3. The input photographs are taken or selected carefully to satisfy certain demand. As one of methods in this category, a 3D-generic model with animation structure is provided in advance and two (or more) photographs are taken. Then a limited number of characteristic points to recognize people, so called feature points (eyes, nose, lips, and so on), are detected on photographs and the other points on the generic model are modified by a special function to adapt it to the photographs. Here we assume a set of 2D points provide reasonable comb ination to get 3D points. Most cases orthogonal photographs are used as input photographs.

Many use an interactive, or automatic method to detect feature points and modify a generic model. Kurihara and Arai [Kurihara 91] use an interaction to get a very few feature points. Since too small number of feature points does not guarantee various shape adaptations from a single generic head, the final shape can be different from the input face and the texture fitting is possibly failed for several areas, which is visible when we do facial animation. So Akimoto et al [Akimoto 93], and Ip and Yin [Ip 96] try to detect more feature points automatically. However the automatic methods are not robust enough to guarantee the proper detection of features since they use simple filtering using *Sobel* operator and constant threshold which must be found by trial-error for each image. Ip and Yin [Ip 96] proposes a method for automatic side feature detection using concave and convex curve, which works well with Asian looking, but not with Caucasian looking people. After modification of a generic model, they use cylindrical projection of the frontal and side views of a face to get 360^o texture mapping using linear interpolation blending with certain range of angles for overlapping area. This texture mapping usually decreases the resolution and many times the blended area shows some mismatching between frontal and side features around chin lines. The limitation of their systems is that perfectly orthogonal photographs with the very straight head position are provided. In addition to get the process automatic, the environment is limited to homogeneous background.

The problematic of these methods for the orthogonal photographs are

- Do we need only one camera to take photographs or several cameras fixed in positions ?
- How much error-tolerant is the system for input photographs?
- How automatic and how robust is the feature detection?
- How to get 3D from a set of 2D from non-perfectly-orthogonal photographs?
- How well does the modification work to get smooth 3D surfaces?
- How well done is the automatic texture image generation without any problems and how much can we keep the resolution of input photographs for the final visualization?

In Chapter 3, we answer these questions while we explain our face cloning system.

2.4.1 Modeling used for expression database

Pighin et al. [Pighin 98] also used manual indicating feature points to modify a generic head to make individualized one. First initial 13 points and then 99 additional points for shape refinement are used. The difference between their methods and the others is the aim of their reconstruction. They produce eight heads from one person with eight different expressions such as “happy”, “amused”, “angry”, “surprised”, “sad”, “sleepy”, “pained”, and “neutral”, which are used later as a database for expressions to synthesize realistic facial expressions.

2.5 Range data

The approach based on 3D digitization to get a range data often requires special purpose high-cost hardware. However it aims to produce highly matched face. These data usually provide with a large number of points for the static shape without having any functional structure for animation.

There are several existing methods using either a laser scanner, active light striper, cameras, video, or stereoscopic cameras.

2.5.1 Medical imaging

Koch et al. [Koch 96] build a human facial model that enables prediction of facial deformations after surgery. The idea is that facial and skull surfaces are extracted, registered, and then the relation between facial and skill surfaces is built using springs and Finite Element Model. When we simulate the standard procedures in craniomaxillo facial surgery, the facial surface is automatically followed and it enables us to predict deformations of the facial shape after surgical procedures. The data sources consist of laser range scans, CT volume data or, for the Visible Human Data Set™, photo slices. First of all, an initial facial surface is extracted from the data sources using edges detected using Canny edge detector. Then the first surface coordinate system is generated in cylindrical coordinate system and an adaptive surface mesh computed for further processing. The registration of facial surfaces and CT data from different sources is carried out manually by setting landmarks. The same structure of vertices on the surface and skull enables the construction of a mesh of springs between skin and skull. The experiment is done using the Visible Human Data Set™. Usually neither high resolution CT scans comparable to the VHD, nor photo slices are available. With future generations of MR and CT scanners, both facial surface and skull of the patient are expected to be available at sufficient resolutions to make range scans obsolete.

2.5.2 Laser scanner

In range image vision system some sensors, such as scanners, yield range images. For each pixel of the image, the range to the visible surface of the objects in the scene is known. Therefore, spatial location is determined for a large number of points on this surface. An example of commercial 3D digitizer based on laser-light scanning, is *Cyberware Color Digitizer™* [http:cyberscan]. Lee et al. [LeeY 95] and Guenter et al. [Guenter 98] digitized facial geometry through the use of scanning range sensors and Blanz and Vetter [Blanz 99] also make 200 generic model database using the *Cyberware™* scanner. However, the approach based on 3D digitization requires special high-cost hardware and a powerful workstation. The drawback of a laser scanner is that first the color is not very high quality and it very often produces non-smooth surfaces for the highly reflective structure of hairy parts (such as hair and eyebrows) and the shady regions such as bottom of nose and between two lips. In addition when the model is very heavy with lots of points on the surface, additional process such as mesh simplification is needed to decrease and optimize the number of points to avoid heavy calculation time to handle the shape and animation.

2.5.3 Stripe generator

As an example of structured light camera range digitizer, a light striper with a camera and stripe pattern generator can be used for face reconstruction with relatively cheap equipment compared to laser scanners. A slide projector displays stripe pattern on the 3D-object surface and the patterns are taken by a camera. The pattern must be projected sharply throughout the image and the projector and camera closer to each other and further from the object. A system calibration showing a box or a corner of a room is pre-processed and then with information of positions of projector and camera and stripe pattern, a 3D shape by extraction of the grid can be calculated. Proesmans et al. [Proesmans 97] shows a good dynamic 3D shape using a slide projector, by a frame-by-frame reconstruction of a video. It analyzes a grid of squares and builds a face and produces high quality shape. It reads horizontal/vertical edges using energy minimization between intensity and smoothness finding crossing contours and then nodes. Iteration procedure with two principles, square grid in the view of projector and consistent numbering to extract the depth value for 3D shape. The extraction of surface texture is automated by modeling the grid attenuation. They use active light striper method, which means it limits the input data, and it shows passive animation representing the same animation as video input since it has no structural information. When a subset of the face is deformed or shows different animation from the input expressed heads, it means a post-processing is done after reconstruction.

2.5.4 Several photographs with sequences of contours

Nagel et al. [Nagel 98] and Zheng [Zheng 94] reconstruct a 3D facial model from sequences of contours. The processes are usually as follows.

1. The "shape-from-silhouette" method is applied to obtain a volume model using the segmentation of the input images and the related camera parameters.
2. Approximating the volume model surface with a wire-frame model reduces the data required for the description of the 3D-shape.
3. Texture mapping is applied to the wire-frame model using the camera images as texture maps. This results in natural-looking textured wire-frame models.

This is a method to build an object with silhouette information in several views. For a large object, a special booth is used with homogeneous background and several cameras in fixed positions. For a small object, we can rotate the object in a fixed axis taking photographs with certain angle gaps. The process is like silhouette passing the cutting a given box. It does automatic sub-triangulation and provides a detailed shape and also it can provide a realistic texture mapping on it. However it has limitation to choose an object to get the shape unless the shape can be extracted only by silhouette. For a face, the silhouette does not provide the all shape information, specially some eye regions since some subparts are not possibly obtained. It has wide applications with various shapes, but since the result is a static shape, we need to have post-processing to give animation.

2.5.5 Stereoscopy

A distance measurement method such as stereo can establish the correspondence at certain characteristic points. The method uses the geometric relation over stereo images to recover the surface depth. It works when an object part faces two or more of the cameras but becomes unreliable where the surface slants away. C3D 2020 capture system by the Turing Institute produces many VRML models using stereoscopy method [http:turing]. Fua and Leclerc [Fua 96] created a human face from a stereoscopic camera using textured areas by weighting the stereo component most strongly for textured image areas and the shading component most strongly for texture-less areas, as shown in *Figure 2-4*. Unless it is combined with model based method with object outlines, the output is quite noisy 3D point clouds.

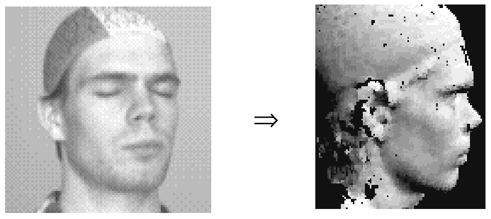


Figure 2-4: Some examples from calibrated video-stereo

2.5.6 Video camera

The method using video camera is getting more and more popular since the equipment is accessible by a non-expert user with multiple purpose.

2.5.6.1 With markers

Guenter et al. [Guenter 98] uses a laser scanner, several video cameras and several colored markers. First a laser scanner *Cyberware*^{TM} scans the person's 3D shape and some parts such as the ears, nose, the eyes and the area under the chin are corrected manually. Then a face is captured with six cameras, where 182 fluorescent colored 1/8" circular paper markers glued on it with nice arrangement with different colors neighbored. The markers after labeling generate a set of 3D points that act as control points to warp the laser scanner scan mesh of the head. Finally the texture mapping is generated frame by frame with a weighted average of the texture maps from six cameras. The result is an animation of high quality of a person. It is a very active method, where we need several equipment and it takes quite long time for preparation such as scanning, manual correcting, and gluing markers.

2.5.6.2 Uncalibrated video, using a generic face model

It acquires only a video stream, which needs an ordinary video camera. More frames there are, better the results are.

Fua [Fua 00] uses regularized bundle-adjustment to model heads from image sequences without calibration data. It is the structure-from-motion problem in the context of head modeling from video sequences for which calibration data is not available. It takes advantage of the rough knowledge of the head's shape, in the form of a generic face model. It allows us to recover relative head-motion and epipolar geometry accurately and consistently enough to exploit a stereo-based approach to head modeling by treating consecutive images in the video sequence as stereo pairs.

2.5.6.3 Optical flow without markers

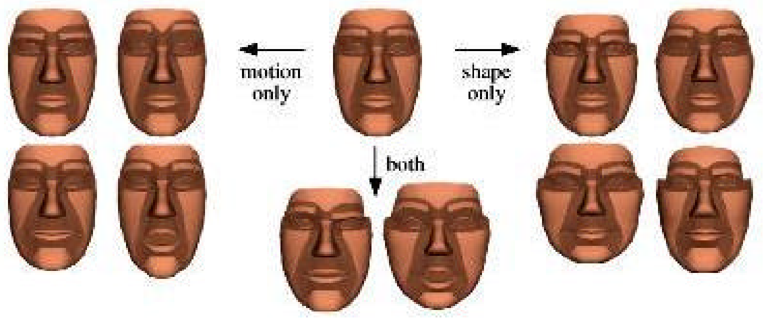


Figure 2-5: Deformable face model for modeling and tracking using optical flow

DeCarlo D. and Metaxas D. [DeCarlo 96] uses Optical flow methods for the multi purpose both to model the person's and to track the movement of the face. In *Figure 2-5*, the default model (top, center) can be made to look like specific individuals by changing shape parameters (the 4 faces on the right). The model can also display facial motions (the 4 faces on the left showing eyebrow frowns, raises, a smile, and an open mouth) by changing motion parameters. And simultaneously it changes both shape and motion parameters (bottom, center). It is able to represent any of these faces to an acceptable degree of accuracy. The benefit of this simplifying assumption is to have a fairly small set of parameters (about 100) which describe a face. This results in a more efficient and more robust system. Estimating parameters from images using a model allows using new methods for processing, which are still related to their counterparts that do not use a model. For example, computing the motion of an object using optical flow (without a model) results in a field of arrows. However, when a model is used, a set of "parameter velocities" is extracted, instead. In the face tracking experiment, the tracked subject moves their head and makes some facial expressions (currently, the system is not real-time). Identifying a small set of feature points by hand performs the initial position of the face in each sequence. In each of the following sequences, the subjects make complex facial and head motions, which are successfully tracked by their framework. Since it is mainly aimed for tracking, the resulting model shapes are not very sophisticated and the animation is passive. To give active animation, the post-processing is needed.

2.6 Comparison

We described several possible methods to create a virtual face from a real face and compare their inputs and results. Designer oriented reconstruction has a space for artistic sense, but it is time consuming. The reconstruction method modifying a generic model using two orthogonal photographs needs commercial equipment and takes just few minutes. The method using range data has the best visual result for shape, but it requires usually either expensive or sophisticated equipment and to give animation structure is another process to be done.

Table 2-1 compares three methods using rather simple equipment such as plaster models and photographs as input data while *Table 2-2* does other methods, which produce range data using more sophisticated equipment. When we say that it is not 'animatable', it means the animation is not possible as an immediate result from the reconstruction. In this case, post-processing is needed to give animation functions. The 'time' means how long it takes to get an individual face. It does not include the time to build the generic models, for example, 'detailed matching' means the result aims to match very close to the input face for every part while 'rough matching' means the result is similar to the input face, but not all the parts are matched.

Table 2-1 and Table 2-2 show practical comparison among many face-cloning methods in terms of

1. What kind of input format for a face is used?
2. What kind of hardware is used?
3. How to get the individualized head from a given input?
4. How much the result is matched with the input face?
5. How long do we expect to get the result?
6. Is the result only static shape or does it have functional structure for animation? If it has animation, is the animation passive replaying only the captured animation or active showing any kind of animation?

Input	Equipment	Acquisition Method	Result	Time	Animatable?
plaster model	mono camera	Manual marks on the model and take photographs in the several view to build 3D	detailed matching	days /weeks	yes
photographs (arbitrary)	mono camera	Manual parameter input and automatic pixel matching process of the image with large database of generic heads	detailed matching	hours	no/ active[4]
	mono camera	Manual using user friendly designing software.	detailed matching with human eyes	days /weeks	active[5]
photographs (organized)	mono camera	(semi)automatic Feature detection and a generic head modification	rough matching	minutes	active

Table 2-1: Possible ways to get a virtually cloned face. Feature part (point) means the easily recognizable parts such as nose, eyes, and lips. Passive animation means we receive animation from input. Active animation means we generate any animation.

[4] It depends on the database of the generic models. If all the generic models have animation structure, the resulting cloned head inherits the function for the animation. If the generic models have only static shape, we have to apply post-process to give animation of the cloned head.

[5] It has "no" animation unless we add the animation structure on the shape. However we put it as "active" since we usually add the animation structure together when we do shaping.

Equipment	Acquisition Method	Result	Time	Animatable?
laser scanner	automatic methods	detailed matching (noise possible)	minutes	no
a camera and pattern generator	automatic line reader	detailed matching		passive
stereoscopic camera	automatic pixel matching between two views	detailed matching (noise possible)	minutes/ hours	no
cameras with homogeneous background	automatic silhouette extraction and sub-triangulation	detailed matching	minutes	no
video camera	marks on a person's face and get positions of marks automatically	detailed matching		passive
	no marks and manual fitting (small number of features) and then automatic matching for all area.	detailed matching (noise possible)		no

Table 2-2: Typical ways to get range data

Since laser scanners and photographs are the most used methods in practice, we compare two methods in *Table 2-3* in terms of equipment, resolution and results.

Photography	Laser scanner
cheaper	expensive
very general equipment	special equipment
	output : numerous points
usually high resolution of texture mapping	usually low resolution of texture mapping
easy to catch characteristic region	often noisy result for characteristic region
difficult to catch non-characteristic points	better to catch non-characteristic points
	problems for hairy parts

Table 2-3: Comparison between photography and laser scanner

We are interested in the animatable virtual human, which are easily reconstructable with general equipment in a robust way. We are motivated to use photograph input using feature-based approach from above comparison. However if there is any way to get range data, we like to utilize the smooth and accurate surface data for the non-feature regions from the range data by combining with feature region from photograph.

2.7 How to give animation structure?

To have control animation on the virtual face, we need to give functional information on the shape. For some cases, it can be done only for a generic model and the individualized faces automatically inherit the functions from the generic model. However other cases we need to process to give animation structure for each time whenever we get a new face model.

2.7.1 Regional information

To simulate the effects of muscle actions on the skin of a virtual human face, specific regions are defined on the mesh corresponding to the anatomical regions where a muscle is desired. P. Kalra et al. [Kalra 92] employ an approach for deformation and animation of a face, based on pseudo muscle design. Animation structure depends on the structure of points. The (neutral) face model is an irregular structure defined as a polygonal mesh. The face is decomposed into regions where muscular activity is simulated using Rational Free Form Deformations. To simulate the effects of muscle actions on the skin of virtual human face, we define regions on the mesh corresponding to the anatomical descriptions of the regions where a muscle is desired. For example, regions are defined for eyebrows, cheeks, mouth, jaw, eyes, etc. as shown in *Figure 2-6*. A control lattice is then defined on the region of interest. Muscle actions to stretch, expand, and compress the inside geometry of face are simulated by displacing or changing the weight of the control points. This deformation for simulating muscle is simple and easy to perform, natural and intuitive to apply and efficient to use for real-time applications.

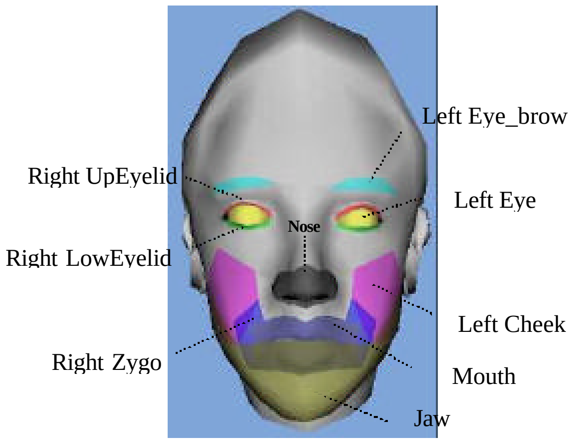


Figure 2-6: An example of regions for animation

This method is very practical to create functional information for animation from scratch.

2.7.2 Warping kernels

Williams [Williams 90] reconstructed a head using *Cyberware™* digitizer and applied warpware to animate the model. A set of warping kernels is distributed around the face, each of which is a Hanning (cosine) window, scaled to 1.0 in the center, and diminishing smoothly to 0.0 at the edge.

2.7.3 Mesh adaptation

Starting with a structured facial mesh, Lee et al. [LeeY 95] developed algorithms that automatically construct functional models of the heads of human subjects from laser-scanned range and reflection data created using *Cyberware Color Digitizer™* [http:cyberscan]. They adapt a generic face mesh to the data. After getting the large arrays of data acquired by the scanner, they reduce it into a parsimonious geometric model of the face that can eventually be animated efficiently. Once the feature based matching technique has fit the mesh, the algorithm samples the range image at the location of the nodes of the face mesh to capture the facial geometry. They use less than 10 points interactively and use an image analysis technique that searches for salient local minima and maxima in the range image of the subject to adapt the generic face mesh to the laser-scanned range and reflectance data. Their search is aimed to direct the known relative positions of the nose, eyes, chin, ears, and other facial features with respect to the generic mesh.

The node positions also provide texture map coordinates that are used to map the full resolution color image onto the triangles.

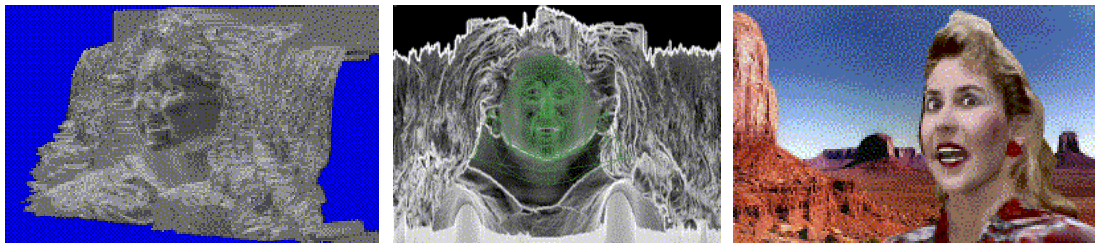


Figure 2-7: Adaptive mesh method to get an animatable individual from laser-scanned range data. (a) Range data input with texture data (b) a deformable mesh and adaptation (c) a resulting animation.

This method is useful to transfer animation structure from a given generic model to an individualized static model. *Figure 2-7* shows an example of their approach.

2.7.4 Facial motion control

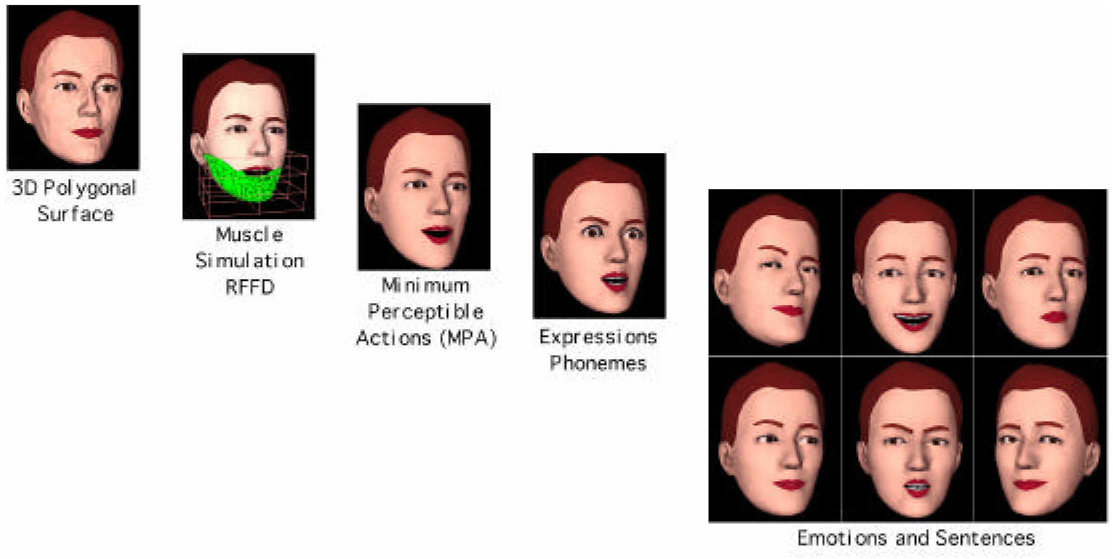


Figure 2-8: Different levels of facial motion control

Specification and parameterization of facial animation muscle actions may be a tedious task. There is a definite need for higher level specification, which would avoid setting up the parameters involved for muscular actions when producing an animation sequence. The Facial Action Coding System (FACS) [Ekman 78] has been used extensively to provide a higher level specification when generating facial expressions, particularly in nonverbal communication context. In multi-level approach [Kalra 91] as shown in Figure 2-8, motion parameters as Minimum Perceptible Action (MPA) has a corresponding set of visible features such as movement of eyebrows, jaw, or mouth and others occurring as a result of muscle contractions and pulls. The MPAs are used for defining both the facial expressions and the visemes. There are 65 MPAs used in the system, which allow users to construct practically any expression and viseme. At the highest level, a script containing speech and emotions with their duration controls animation. Depending on the type of application and input different levels of animation control can be utilized.

Body

Compared to face part, the body cloning has been less explored by researchers and is getting more and more popular now. Similarly to the face part, there are some methods that concern precision and accuracy using laser scanner [Addleman 97][Daanen 97], silhouette information [Kakadiaris 95][Kakadiaris 96][Gu 98] or video streams [Plänkers 99]. Generally, these systems are either expensive or require expert knowledge in using them and need a special environment setting. Thus, most of them have limitations when compared practically to other systems with a commercial product (such as a camera) for the input of data for reconstruction and finally animation. These approaches [Hilton 99] using photographs concern mainly the individualized shape and visual realism using a high quality image input.

2.8 Laser scanner

To capture the intricacies of the human body in one pass, the *Cyberware*^{TM} Whole Body scanner [http:cyberwareBody] uses four scanning instruments mounted on two vertical towers with laser, CCD sensor and Camera for texture capturing as well as software to merge four huge meshes from the instruments. Each tower has a linear ball-bearing rail and servo motor assembly that moves the scanning instruments vertically. With a person standing on the scanner's platform, the scanning instruments start at the person's head and move down to scan the entire body. A primary goal is to acquire as complete a model as possible in one pass. The use of multiple instruments improves accuracy on the sides of the body and in difficult-to-reach areas, such as under a person's arms. While the simple anthropometric pose gives the best results, *Cyberware*^{TM} designed the WB4 to handle many different poses for a wide range of applications. The WB4 scans a cylindrical volume 2 meters (79 inches) high with a diameter of 1.2 meters (47 inches). These dimensions accommodate the vast majority of human subjects.

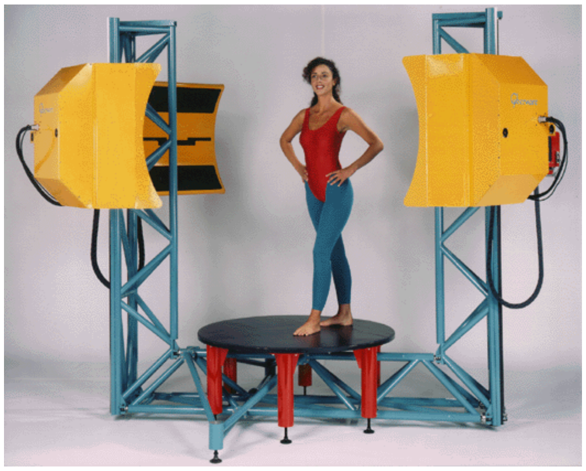
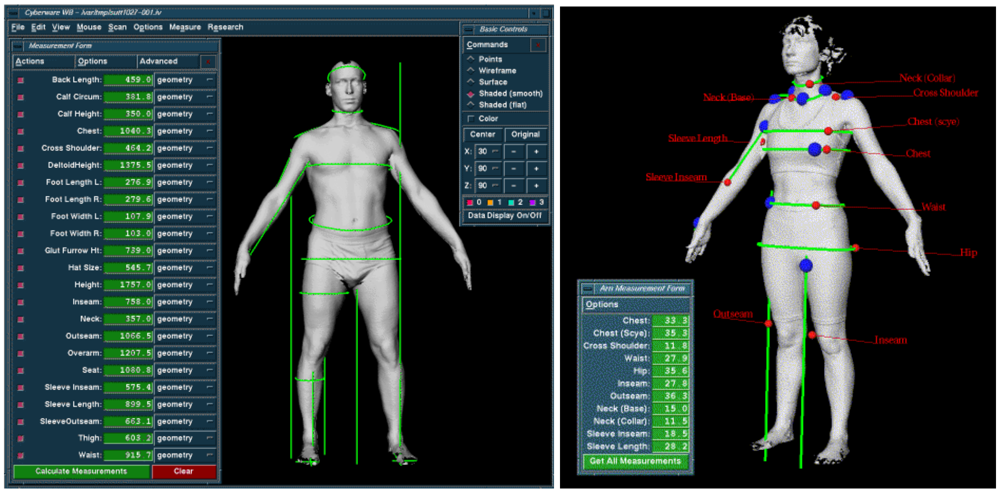


Figure 2-9: The Cyberware^{TM} *Whole Body scanner*

Since recent laser range finder (hardware) becomes very good, the key point is the software that merges the complex meshes with texture into one. Usually they sell smaller version for scan the size of human head, which provide better quality scanning because the head and shoulder is convex, having no concave part. The concave part cannot be scanned and the scanner produce holes, which causes a problem. The software to compensate the hole problem, calibration of the texture image from different camera and polygon reduction functionality is now hot area.

DigiSize as shown in *Figure 2-10* is a set of proven software tools for measuring 3D data sets from the *Cyberware*^{TM} Whole Body 3D Color Scanner and performing data analysis. *DigiSize* automatically extracts dozens of tailor and anthropological measurements. Within 30 seconds after the 17 second scan, *DigiSize* software evaluates the scan data for key reference points, automatically extracts dozens of measurements and calculates the correct clothing sizes for the subject.



Back Length:	459.0	geometry
Calf Circum:	381.8	geometry
Calf Height:	350.0	geometry
Chest:	1040.3	geometry
Cross Shoulder:	464.2	geometry
DeltoidHeight:	1375.5	geometry
Foot Length L:	276.9	geometry
Foot Length R:	279.6	geometry
Foot Width L:	107.9	geometry
Foot Width R:	103.0	geometry
Glut Furrow Ht:	739.0	geometry
Hat Size:	545.7	geometry
Height:	1757.0	geometry
Inseam:	758.0	geometry
Neck:	357.0	geometry
Outseam:	1066.5	geometry
Overarm:	1207.5	geometry
Seat:	1080.8	geometry
Sleeve Inseam:	575.4	geometry
Sleeve Length:	899.5	geometry
SleeveOutseam:	663.1	geometry
Thigh:	603.2	geometry
Waist:	915.7	geometry

Calculate Measurements | Clear

Chest:	33.3
Chest (Scye):	35.3
Cross Shoulder:	11.8
Waist:	27.9
Hip:	35.6
Inseam:	27.8
Outseam:	36.3
Neck (Base):	15.0
Neck (Collar):	11.5
Sleeve Inseam:	18.5
Sleeve Length:	28.2

Get All Measurements

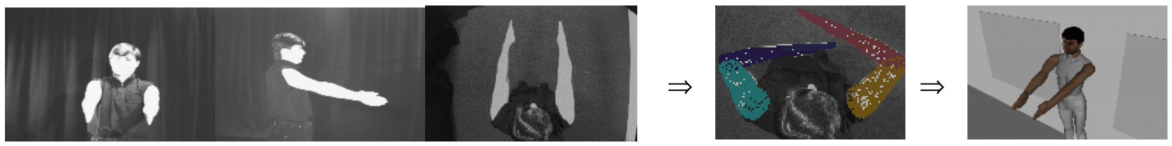
Figure 2-10: The software DigiSize for measuring 3D data sets from the Cyberware^{TM} Whole Body 3D Color Scanner and performing data analysis

2.9 Silhouette in multiple views

2.9.1 Video

Kakadiaris and Metaxas [Kakadiaris 95][Kakadiaris 96] present a novel approach to the three-dimensional human body model acquisition from three mutually orthogonal views, the shape is not their main concern though. It uses silhouette information to catch the movement of human body. Their technique is based on the spatio-temporal analysis of the deforming apparent contour of a human movement according to a protocol of movements. The technique does not use a prior model of the human body and prior body part segmentation is not assumed. Therefore, the technique applies to humans of any anthropometric dimension.

To parameterize and segment over time a deforming apparent contour, they introduce a new shape representation technique based on primitive composition. The composed deformable model allows representing large local deformations and their evolution in a compact and intuitive way. In addition, this representation allows to hypothesize an underlying part structure and test this hypothesis against the relative motion (due to forces exerted from the image data) of the defining primitives of the composed model. Furthermore, they develop a Human Body Part Decomposition Algorithm that recovers all the body parts of a subject by monitoring the changes over time to the shape of the deforming silhouette. In addition, they modularize the process of simultaneous two-dimensional part determination and shape estimation by employing the Supervisory Control Theory of Discrete Event Systems. This method is extended into three mutually orthogonal viewpoints to obtain a three-dimensional model of the subject's body parts. The effectiveness of the approach is demonstrated through a series of experiments where a subject performs a set of movements according to a protocol that reveals the structure of the human body. *Figure 2-11* shows the process of the reconstruction of the shape and motion. A generic graphical model has been customized for the subjects as follows: 1) the shape of the upper body extremities has been determined based on the data from the acquired image sequences, and 2) the model of the head has been obtained using a Cyberware scanner.



Frontal, side and top cameras Analysis Result

Figure 2-11: The input acquirement using cameras and final reconstruction of the shape and motion.

2.9.2 Photographs

Gu and et al. [Gu 98] concentrate on human body modeling for the purpose of measurement of human body for the clothes industry. They use a hexagonal supporting framework to form a closed space for imaging with 12 cameras for upper and lower body parts in six views. Additionally a slide projector with a grid pattern is used to catch the chest area using the stereo pair of the intersections of horizontal and vertical lines. The silhouette is extracted and some attached markers are reconstructed to build feature curves of the final body.

Weik et al. [Weik 98] creates flexible anthropomorphic models from multiple views by extracting silhouette. Then from different views, points of each volumetric cone are constructed, whose intersections in 3D form the final approximation of the volume model. The predefined skeleton and a flexible triangle surface mesh are adapted to the point data. The environment is again limited by homogeneous background and they treat only the upper body.

Hilton et al. [Hilton 99] proposed a method for capturing of realistic whole-body animated models of clothed people using model-based reconstruction from multi-view color images. A set of four-color images of a person is captured from the frontal, back, right and left. The images are used to morph a 3D generic humanoid model to have the shape and appearance of a specific person. Models are generated in a standard VRML Humanoid Animation format, which can be animated in any VRML browser using pre-defined animations, but without proper skin deformation. It is a low-cost whole-body capture of clothed people from digital color images. The approach is simple and efficient. However this method does not give a good reconstruction and animation for the face. It also lacks the flexibility in terms of the imaging environment since it requires a specially prepared background and properly controlled lighting when images are taken.

2.10 Stereo-video

Plänkers et al. [Plänkers 99] uses video cameras with stereo pair for the body part. A person's movements such as walking or raising arms are recorded to several video sequences and the program automatically extracts range information using stereo and tracks an outline of body. This 2-D and 3-D information is fed to an optimizer that fits to the data. The problem to be solved is twofold: First, robustly extract image information from the data; second fit the models to the extracted information. The data is used to instantiated the models and the models - augmented by our knowledge about the human body and its possible range of motions - are in turn used to constrain the feature extraction. They focus more for the movement and approximate the shape.

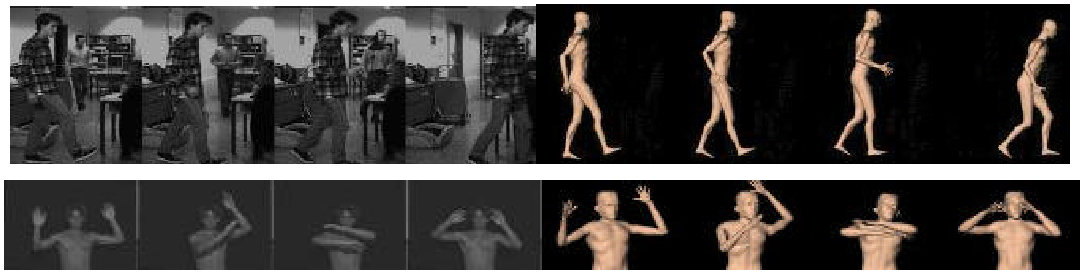


Figure 2-12: Walking sequence and upper body sequence

Chapter 3 Face Cloning

This chapter and the next chapter are devoted to describe our system to realize the virtual cloning of a real person. Since our aim is to provide a realistic virtual human for real-time animation as real human acts in the real world, the most important criteria is to make a photograph-realistic looking of a virtual face with optimized vertices for the real-time animation. So we take photographs as our input and make a shape modification using a generic model with optimally distributed vertices. The global approach belongs to a modification a generic model with Section 2.4 'Features on photographs (organized) and a generic model' category. So we propose one methodology to reconstruct 3D facial model for animation from two orthogonal photographs taken from frontal and side views. The reconstructed 3D-face can be animated immediately with any given expression parameters and any face created from this method is morphable and can be used to create a large population of faces, which is described as an application in Chapter 5.

We organize this chapter as follows. In Section 3.2, we present a fast method applied to photograph input to get an animatable cloning of a person. Semiautomatic feature detection is described to get rough shape of a given face from orthogonal photograph data. The methods to modify a generic model depending on data are followed and automatic texture mapping method is provided. Several resulting heads are illustrated in Section 3.4 while 3.5 devoted to validation of the cloned results. Finally in Section 3.6, conclusion and future research are given.

Figure 3-1 shows the global structure of face cloning from photographs.

The preprocess is done to make

1. A generic shape of the face
2. Generic animation structure
3. Definition of feature point and feature lines that show the connection of feature points
4. Definition of control points on the generic model
5. Coordinate relation calculation for Dirichlet Free-Form Deformations (DFFD) which is used for shape modification
6. Expression database

Whenever there is input for a new face data, the predefined feature points and lines need to be positioned properly. We process normalization and a semiautomatic feature positioning starting from a few key features to guide automatic feature detection.

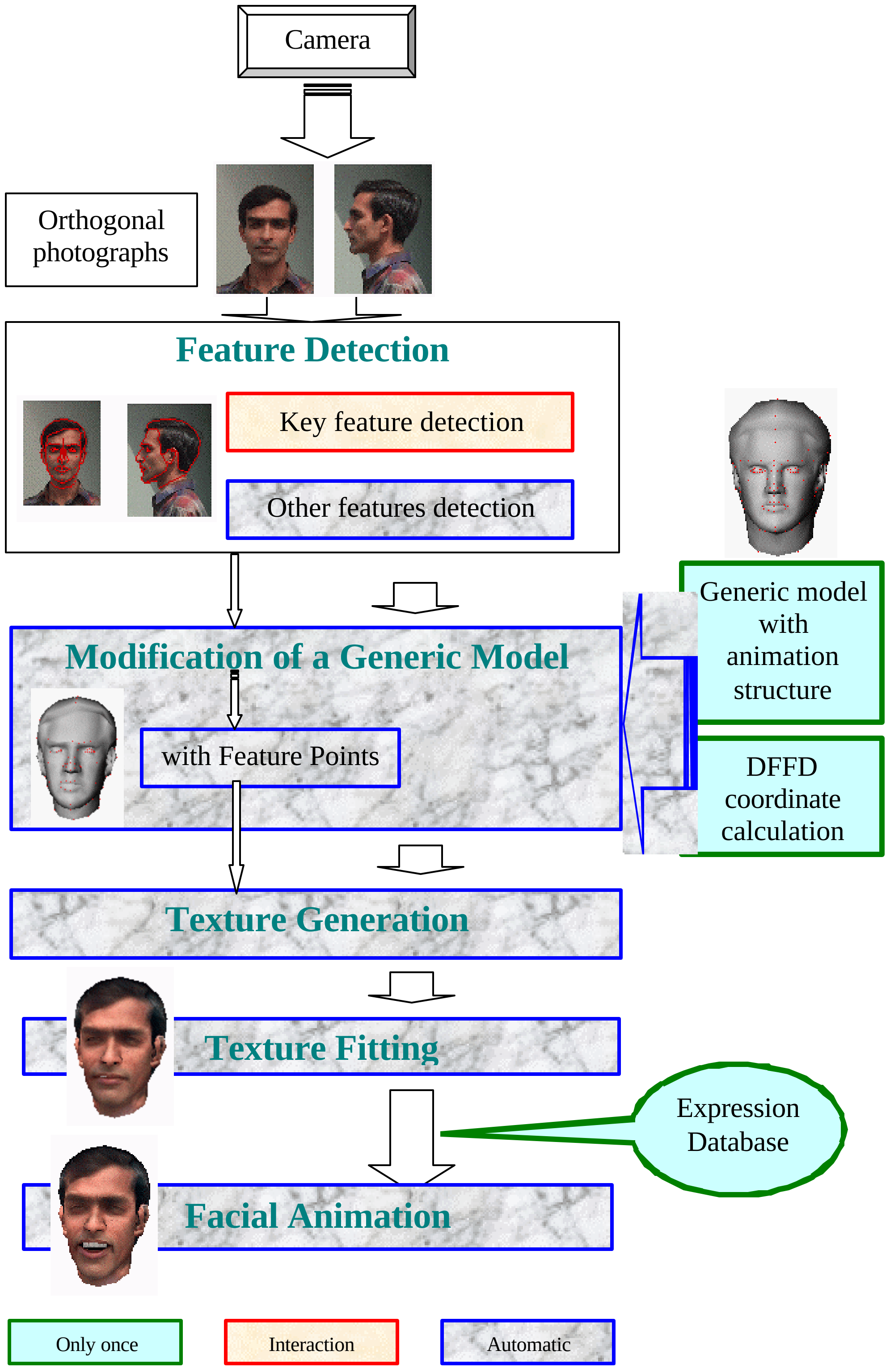


Figure 3-1: An overall flow diagram for face reconstruction from two kinds of input.

3.1 Feature detection

Feature detection from 2D image data is the one of the main steps. When we refer feature detection, it means to catch the 2D or 3D positioning data of most visible points on a face such as eyebrow, eye outlines, nose, lips, the boundaries between hair and face, chin lines and etc. Some parts such as forehead and cheeks are not always easy to locate exactly on 2D photographs and they are called as non-feature points.

3.1.1 Photographing

To take orthogonal photographs is not very easy in practice. There are some instructions for photographing as follows:

1. The ideal weather and place: It is best if the weather is cloudy since the skin color appears natural and no shadow casts on the face. If the photographs are taken indoors, select the lights and place that best emulate the cloudy weather condition. In this case, we recommend using at least two lights to minimize the strong reflection off the skin from a single source.
2. The ideal position of the camera: Note that if the camera shot is taken up too close to the subject, the resulting virtual human will have an overall, deformed shape. To avoid this, use the maximum zoom of the camera positioned at a distance to minimize such deformations. If the space is limited, try to do the best of the given condition. We avoid the camera calibration step that limits input equipment by specific camera whose parameters are known for the calibration calculation. So we assume that the photographs are taken in a distance.
3. The best hairstyle and face arrangement: Tuck the hair behind the person's ears so that the ears are visible, and make sure the forehead is visible. Ask the person to take off the eyeglasses and project a neutral face (no smile). Ask the person to sit up straight with the eyes open and the mouths closed. The teeth should not be visible.
4. For the side view, make the face straight so that we can see the profile properly.
5. The size of an image is not a problem, but it should not be too small or too big. We recommend 480x640.

3.1.2 Normalization

We prepare a 3D-generic model and 2D-feature point frames for both the normalization and the feature detection. Normalization is used to bring image data into feature points space.

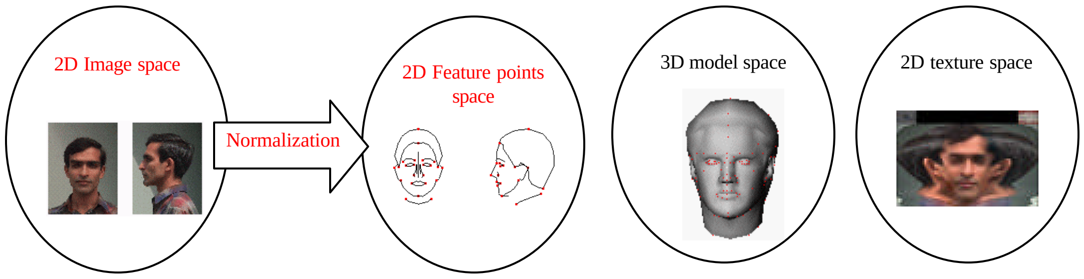


Figure 3-2: Normalization brings the input images into feature points space.

First we prepare two 2D-frames composed of feature points and feature lines with predefined relation for frontal and side views. The feature lines in the frames are designed as an initial position for the piecewise affine transformation and snake method, which will be used later. To make the head size in the frontal view and the side view the same we measure the lengths of a face in two views. Then we choose one point from each view to match them with the corresponding points in prepared frame. Then we use two transformations (scaling and translation) to bring each photograph to the feature frame coordinates, overlaying frames on photographs. Two photographs with different size and positioning are shown in Figure 3-3.

3.1.3 Structured Feature Detection

There are methods to detect feature points interactively [Arai 96] or fully automatic just using special background information, predefined threshold, or image segmentation [Akimoto 93][Ip 96]. However, they are not very reliable since some boundaries in natural environment are not easy to detect in many cases and many parameters such as threshold for edge strength are too sensitive depending on each individual's facial image.

We provide a semi-automatic feature point extraction. It has two main steps. The first uses interactive positioning for only a few key features (See brighter and bigger points in *Figure A-1*, *Figure A-2* and *Figure A-3*) and then automatic way follows to adjust others feature points. The key feature points are shown as brighter points in Figure 3-3. After applying the piecewise affine mapping to bring other feature points in relatively close to face features, we use a snake method [Kass 88], which is more robust than simple threshold method. To get correspondence between points from photographs and points on a generic model, which has a fixed number, a snake is a good candidate. Above the conventional snake, we add some more functions called as structure snake, which is useful to make correspondences between points on the frontal view and ones on the side.

Feature detection is applied for both frontal and side views. We get (x, y) from the frontal view and (y, z) from the side view. We can provide some automatic adjustment to make y coordinates on the frontal and side views share the same or at least similar value. See Figure 3-3.

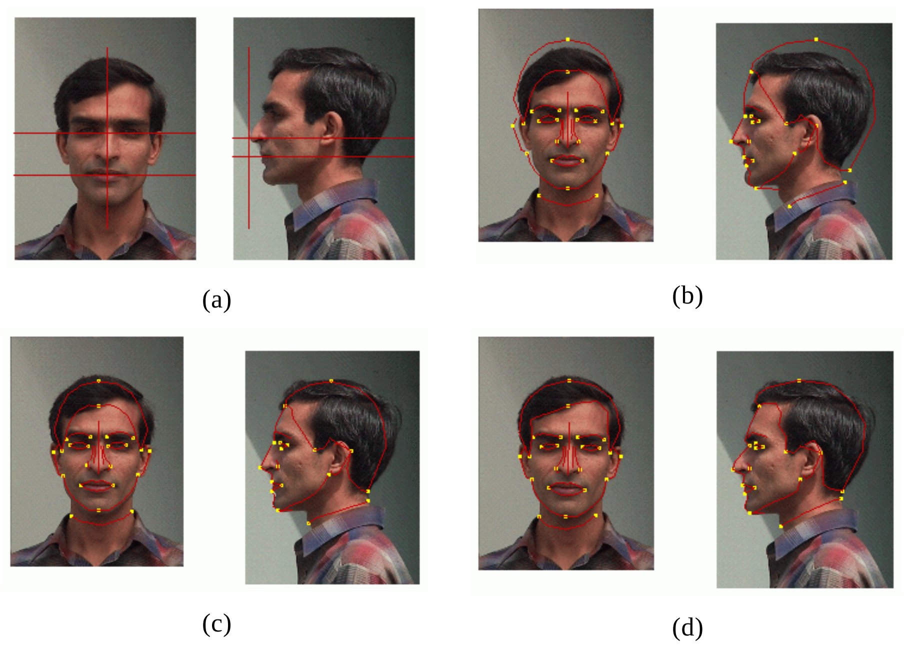


Figure 3-3: (a) normalization (b) feature points frames on the frontal and side views (c) affine mapping with key feature points (d) the final detected feature points

3.1.3.1 Piecewise Affine mapping

Piecewise affine mapping is a kind of freeform deformation combined with several affine mappings as shown in *Figure 3-4*. An affine mapping is combined with a linear matrix and translations and it is chosen to transform points when we move control points to other positions. This affine mapping is useful in 3D and a simple projection into 2D is used here. The control points for the affine mappings are located at the boundaries between two continuous parts, so that surface or curve continuity is preserved when the positions of control points are changed.

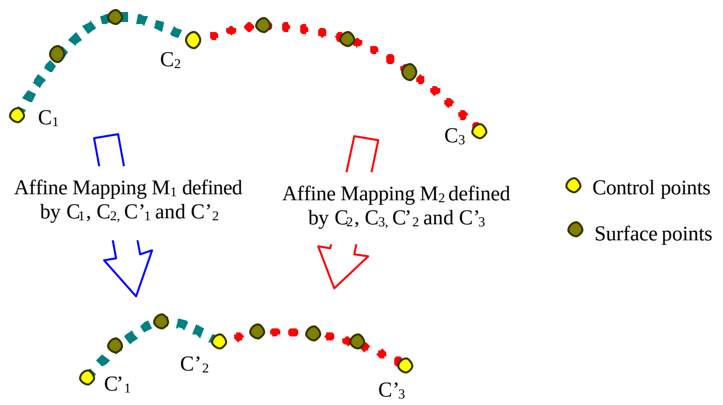

C₁
C₂
C₃
Affine Mapping M₁ defined by C₁, C₂, C'₁ and C'₂
Affine Mapping M₂ defined by C₂, C₃, C'₂ and C'₃
Control points
Surface points
C'₁
C'₂
C'₃

Figure 3-4: Piecewise affine mapping for curves

3.1.3.2 Structured Snake

First developed by Kass et al. [Kass 88], the active contour method, called snakes, is widely used to fit a contour on a given image. On the conventional snake, we add three more functions. First, we anchor some points to keep the structure of points when snakes are involved, which is also useful to get more reliable results when the edge we would like to detect is not very strong. In this case, the key feature points are anchored. Since the correspondence between the control points on a generic model, which will be used for head modification later, and the feature points on photographs have to be identified, just to have edge detection is not enough. Anchoring some points to get sure-position for certain points helps this correspondence. We have several parameters such as elasticity, rigidity, image potential and time step, to manage the movement of snake. As our feature lines (contours) are modeled as polylines, we use a discrete snake model with elastic, rigid forces and image force acting on each pixe l for color interest. We define different sets of parameters for hair and face according to their color characteristic. To adjust the snake on points of strong contrast, we consider

$$F_{ext,i} = n_i \cdot \tilde{N} E(v_i) \qquad (1)$$

where n_i is the normal to the curve at the node i, whose position is v_i and is given by

$$E(v_i) = |\tilde{N} I(v_i)|^2 \qquad (2)$$

where $I(v_i)$ represents the image itself. To estimate the gradient of the image, we use the *Sobel* operator.

We then use color blending for a special area, so that it can be attracted by a special color [Beylot 96] where the information is easily obtained by the surrounding color of key feature points. Blending of the different color channels is changed to alter the snake's color channel sensitivity. For instance, we can make snakes sensitive to the excess of dark brown color for hair. In addition, we use clamping function to emphasize special interesting range of color. The color blending and clamping function depend on the individual.

When the color is not very helpful and the *Sobel* operator is not enough to get good edge detection, we use multi-resolution techniques [Burt 83] to obtain strong edges. It has two main operators, REDUCE and EXPAND with *Gaussian* operator.

Let G_0 be the original image. Then for $0 < l < N$:

$$G_l = REDUCE\ (G_{i-1})$$

by which we mean:

$$G_l(i,\ j) = \sum_{m=1}^{5}\sum_{n=1}^{5} w(m,n) \cdot G_{l-1}(2i+m, 2j+n)$$

where the pattern of weights *w(m,n)* used to generate each pyramid level from its predecessor is called the generation kernel. These weights are chosen subject to four constrains, separable, symmetric, normalized, and each level *l* node must contribute the same total weight to level *l+1* nodes. So when $w(m,n) = \hat{w}(m)\hat{w}(n)$, $\hat{w}(0) = a$, $\hat{w}(-1) = \hat{w}(1) = b$ and $\hat{w}(-2) = \hat{w}(2) = c$ where $a + 2b + 2c = 1$ and $a + 2c = 2b$. So for a given a, $b = ¼$ and $c = ¼ - a/2$. We use $a = 0.4$ in our application since then the shape of the curve is similar to the *Gaussian* curve.

Let $G_{l,k}$ be the image obtained by expanding G_l, k times. Then $G_{l,0} = G_l$ and for $k > 0$,

$$G_{l,k} = EXPAND(G_{l,\,k-1})$$

by which we mean:

$$\mathrm{G_{l,k}(i,\ j)} = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} G_{l,k-1}\left(\frac{2i+m}{2}, \frac{2j+n}{2}\right)$$

where only terms for which $(2i + m)/2$ and $(2j + n)/2$ are integers contribute to the sum.

The subtraction of G_l by $G_{l,1}$ produces an image resembling the result after *Laplacian* operators commonly used in the image processing. More times the REDUCE operator is applied the stronger the edges become. For the better understanding, see *Figure 3-11* and *Figure 3-12*. The same method is used to remove boundary effect in texture generation in Section 3.3.1.2.

3.2 Shape reconstruction

We detected structured feature points on photograph images. There are several ways to modify a given generic model either in 2D or in 3D. We provide a 3D modification using 3D control points, which makes the condition for perfect orthogonal image less critical than in 2D modifications. Then we modify a generic model using 3D features. Dirichlet Free Form Deformation [Moccozet 97] is used to change the shape of the generic model by taking 3D feature points as control points for deformation. The heavy calculation to get coefficients of vertex movement related to control points is done only once with the generic model and the saved coefficients for vertices are applied for each new shape modification. This process is a rough matching fitting only feature points and approximating other vertices.

3.2.1 Modification in 2D or in 3D?

We have a certain set of 3D feature points. The question is how to modify a generic model, which has more than a thousand points to make an individualized smooth surface. First we have to choose if we do modification in 2D and then combine two sets of 2D to make 3D or we do modification in 3D directly. This is a critical issue for the data handling for error tolerance. We have to take a method to decrease the dependency of perfectly orthogonal photograph input.

Several kinds of distance-related functions in 2D have been employed by many researchers [Akimoto 93][Ip 96][Kurihara 91] to calculate displacement of surface points related to the feature points detected. However these methods can not recover the proper shape when the detected features are not perfectly orthogonal.

We apply an algorithm on detected 2D features to make 3D features aiming to decrease the dependency of orthogonality of photographs and then take modification in 3D. Since we know the feature point structure from photographs, there are possibilities to handle them in a proper way by filtering using the structure of feature points.

3.2.2 Asymmetric 3D features from two sets of 2D features

Our input is two views of a person. One frontal view and one side view are the only input. Since only one side view is used, one may misunderstand that the final virtual clone has a symmetric face. However the frontal view has almost all information for asymmetric information and the way to produce 3D points from two 2D points saves asymmetric information of the face. Two sets of feature points on frames have structure, which means every feature point has its own name. In our case, we use about 160 feature points. Some points among them have position values (it means they are visible on images) on both the frontal and the side views, while others have values only on the frontal view or on the side (for example, back part of a head is not visible in the frontal view). The problem is how to make (x, y, z) from a given set of 2D points. Since the perfect orthogonal pair photographs are not easy to be realized, it may occur unexpected 3D shape if we take average of y_s and y_f for y coordinate (subscripts s and f mean side and frontal view). This is our criteria to combine two sets of 2D features, say (x, y_f) and (z, y_s), to make 3D features. Figure 3-5 shows the name convention we are using such as

- FV means the feature point has value (x, y_f) on the frontal view
- SV means the feature point has value (z, y_s) on the side view
- _R_ means the right side region indicated as _R_
- _L_ means the left side region indicated as _L_
- _C_ means neither _R_ nor _L_.

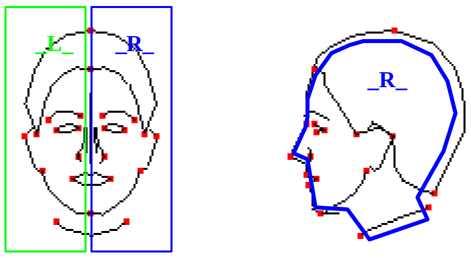


Figure 3-5: Feature points names are organized in a special way to indicate if the feature belongs to _R_, _L_ or centerline to make the algorithm to create 3D points.

1. _R_, FV, !SV : we use a predefined relation from a typical face to get z, for instance the depth value z of inner corner point of eye is calculated from middle and outer corner points of the eye.

1. FV, SV : points have x, y_f, y_s, z, and we take y_f for y value. Note that we do not take the average of y_f and y_s.

2. _R_, !FV, SV : we use a predefined relation from a typical face to get z, for instance the x value, for instance the x of neck point invisible is calculated with visible neck points of the frontal view.

3. _L_, FV, !SV, _L_ : we take z value from the corresponding point with _R_ that has the reflection counter part by vertical line, we take the depth value z from the reflection part., for example the left corner of the left side eye is corresponding to the right corner of the right side eye.

4. _L_, !FV, !SV : we use a predefined relation from a typical face to get x and we take y and z values from the corresponding point with _R_.

5. !FV, SV, _C_ : we take x value from the line between two features with FV and _C_, for example we take the feature on top of the hair and middle chin feature or simply 0. Here we set x as 0, which will be used in the texture coordinate fitting method later.

6. We take y_s for y value in 'ear' region, which is useful for 'ear' texture for texture mapping.

3.2.3 Modifying the generic model in 3D

One of the easiest modification methods of a head is to use a spherical or cylindrical projection of a head and use Delaunay triangulation and Barycentric coordinates. More sophisticated solution is to use non-linear deformations as opposed to generally applied linear interpolation, which can give a smooth resulting surface. It uses 3D feature points as a set of control points for a deformation. Then the deformation of a surface can be seen as an interpolation of the displacements of the control points. Before applying deformation method, we use global transformations (translation, and scaling) to bring obtained 3D feature points to generic model's 3D space by selecting eight feature points on a generic model and corresponding feature points on detected feature points. In general, we use the upper-most, right-most, left-most and right-most points for the translation and scaling. Here is our example of calculating the global transformations.

$$S = \frac{(Gctrl_{P1}.x - Gctrl_{P2}.x)}{(ctrl_{P1}.x - ctrl_{P2}.x)}$$

$$T.x = \frac{(Gctrl_{P3}.x + Gctrl_{P4}.x)}{2.0} - S\frac{(ctrl_{P3}.x + ctrl_{P4}.x)}{2.0}$$

$$T.y = \frac{(Gctrl_{P5}.y + Gctrl_{P6}.y)}{2.0} - S\frac{(ctrl_{P5}.y + ctrl_{P6}.y)}{2.0}$$

$$T.z = \frac{(Gctrl_{P7}.z + Gctrl_{P8}.z)}{2.0} - S\frac{(ctrl_{P7}.z + ctrl_{P8}.z)}{2.0}$$

where we use $P_1 = P_3$ = outer_most_R_EYE, $P_2 = P_4$ = outer_most_L_EYE, P_5 = middle_top_HAIR, P_6 = low_back_CHIN, P_7 = tip_NOSE, P_8 = side_outline_back_HAIR3. For the names of feature points, refer Appendix A Feature Names.

Once we transform the feature points, we define a FFD to modify the generic model to adapt to the features detected in such a way as follows:

1. Design of the control point set on the generic model
2. Calculation of the weights of a set of the control points set for each point on a surface of the generic model.
3. Deformation of the control point set to the new positions (here to the detected feature positions)
4. Deformation of the object to the new positions using the new position of control points and the weights.

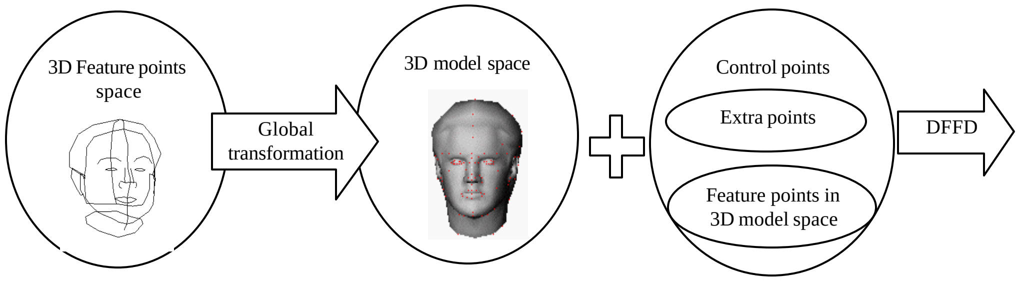


Figure 3-6: The relation between feature points and control points for DFFD

3.2.3.1 Dirichlet Free-Form Deformations (DFFD)

Free-Form Deformations (FFD)[Sederberg 86] belong to a wider class of geometric deformation tools. However FFD has a serious constraint in that control points boxes have to be rectangular, which limits the expression of any point of the surface to deform relative to the control points box. Farin [Farin 90] extends the natural neighbors' interpolant based on the natural neighbors' coordinates, the Sibson coordinate system [Sibson 80] (refer Appendix C) based on Voronoi (so called Dirichlet) and Delaunay diagrams [Aurenhammer 91] for a scattered data interpolant, using the support for a multivariate Bézier simplex [DeRose 88]. He defines a new type of surfaces with this extended interpolant called Dirichlet surfaces.

Moccozet and Magnenat-Thalmann [Moccozet 97] extended the idea of Dirichlet surface to FFD. Combining FFD's and Dirichlet surfaces leads to a generalized model of FFD's: Dirichlet FFD's or DFFD's. In the Dirichlet-based FFD approach, any point of the surface to deform located in the convex hull of a set of control points in general position, is expressed relative to a subset of the control points set with the Sibson coordinate system. One major advantage of this technique is that it removes any constraint on the position and topology of control points. One control point is defined at the position of each surface point, so that any displacement applied to the control point will also be applied to the surface point. The deformations are local, as the coordinate system on which they are based is local. An area of influence is assigned to each control point. The size of the region depends on the neighbors control points. The region of influence is defined as the union of all Delaunay spheres going through the given control point. When a control point is moved, only the points in its region of influence are affected. The method is particularly suited for irregular control point distribution, and so allows to refine the control points set at places where very fine deformation are needed without penalizing the deformation over the control points set. The continuity properties also apply to the interpolated displacement function if we consider the displacements assigned to control points as function values assigned to data nodes/control points. The derivatives of the interpolated function are continuous everywhere except at the data nodes. Sibson coordinates are continuously differentiable at all points except at data nodes and the Sibson interpolant guarantees continuity in first and second derivatives everywhere except at data nodes. With some constraints, continuity is extended to data nodes in the extended Sibson interpolant.

Moccozet and Magnenat-Thalmann [Moccozet 97] show an example of DFFD for real-time hand simulation with smoothly deformed surfaces. Once we calculate the coefficients of a subset of the control points set with the Sibson coordinate system for each point on a surface, the deformation is done in a real-time. Therefore DFFD is used for getting new geometrical coordinates for a modification of the generic head. The heavy calculation is done once with the generic model and its control points and whenever we get a new face image data, we apply the already calculated coefficients for deformation with a new set of control points. We define the control point set as the newly detected feature points. Since only the part of the object inside the convex hull of the set of control points is deformable, we have to be sure if the convex hull of the control points includes all points on the 3D head. Even though our feature points include the front and side silhouette, there are some points outside the convex hull. So we add 27 extra points on a rectangular box surrounding the 3D generic head as a part of control points for DFFD as shown in Figure 3-7. Here we adapted the routines of DFFD from a colleague, Laurent Moccozet, in MIRALab, University of Geneva, inside the modification module.

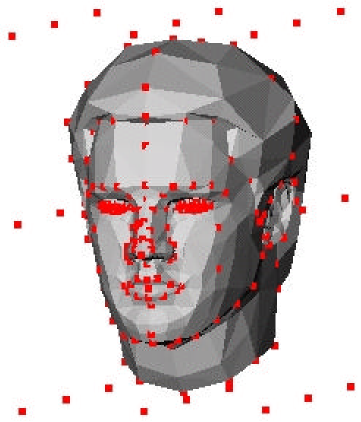


Figure 3-7: Control points and the generic model. The control points on the surface are corresponding to feature points on photographs and the other control points are set as extra.

3.2.3.1.1 Eyes and teeth

When we apply DFFD on a head, eyes and teeth are modified too and it may create unexpected deformation for them. So we take back their original shape and position and then find the correct positions of them through translation and scaling appropriate to new head. Here we can use some reference points as the lip corner feature points and eye. See Figure 3-8. This is a feature based rough matching method; it does not attempt to locate all points on the head exactly, in contrast to range data from laser scanners, which is supposed to have more accurately matching shape for overall area. The result is, however, quite respectable considering the input data (photographs from only two views).

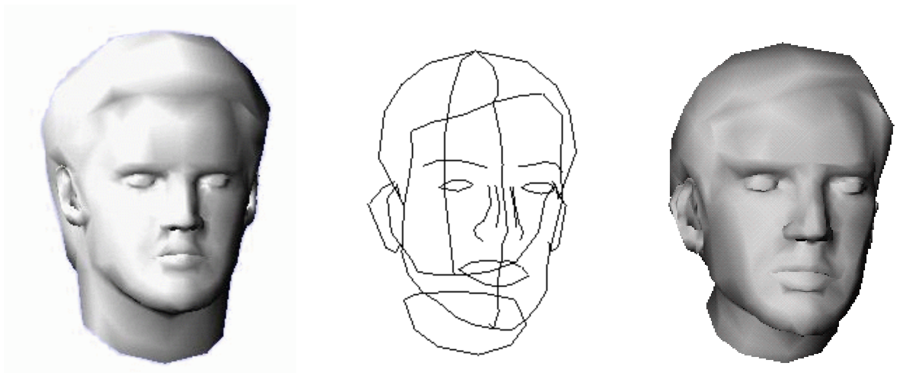

(a) The generic model (b) Features in 3D (c) After DFFD

Figure 3-8: The shape modification using DFFD.

3.3 Automatic texture mapping

To increase photo-realistic looking of virtual objects, we utilize texture mapping by applying real images onto them. For virtual faces, the texture can add a grain to the skin, including the color details for the nose, lips and etc. Texture mapping needs both a texture image and texture coordinates, where each point on a head needs one or several coordinates. The input images are not appropriate to be used for texture mapping; hence, generation of texture image is also processed in this section.

3.3.1 Texture Image Generation

If two photographs for the frontal and side views are taken simultaneously in an environment with two cameras setting on the proper positions to get orthogonal photographs, we may possibly perform texture mapping just by finding proper texture coordinates on two separate images. However our aim is to remove restriction as much as possible to get input images, so that we use two photographs taken with only one camera asking the object rotate, which changes the light condition and luminance. This case just a simple combination of the frontal view and the side view does not provide properly smooth texture mapping, caused by both non-perfect orthogonal condition and non-coherent luminance.

There are two steps for this image generation.

1. Geometrical deformation to resist non-perfect orthogonal photographs input
2. Smoothing of boundaries between different images to resist non-coherent luminance.

3.3.1.1 Geometrical Image deformation

We assume the frontal view is more important than the side view, so the frontal view is kept as it is and the side view is deformed and stuck to the frontal view. We define two subsets of feature points (one for left part and the other for right part) on the frontal view, intending to keep original (high) resolution for major part on the frontal view, which lies between two feature lines. There is a corresponding feature line on the side image.

As an example, we use feature named by Middle_top_HAIR, front_outline_R_HAIR1, face_R_HAIR3, outer_R_EYEBROW, R_CHIN5, R_CHIN4, R_CHIN3, middle_NECK for the right side feature lines for image deformation. For the left part, we use dual name policy such as face_R_HAIR3 is dual of face_L_HAIR3. This process to get position of these features on the images is automatic once we define which features are used for image deformation.

We use piecewise linear deformation for the side image to transform the feature line to the corresponding one on the frontal view as shown in Figure 3-9 where the side image is used as the right view of the face. We deform the right view of the face with transformation to match to the right feature line on the frontal image. For the left image, we flip the side image across to the vertical axis and deform it with relation of the left feature line on the frontal image. Figure 3-10 (b) shows how the side view is deformed and connected to the frontal view with input images in (a). The lines on Figure 3-10 (b) show the defined sets of feature points, which can be changed easily. The lines here are not visible in the actual process, of course. If the side image is not good enough, we take the hair, ear and neck outline feature points for the feature lines for deformation.

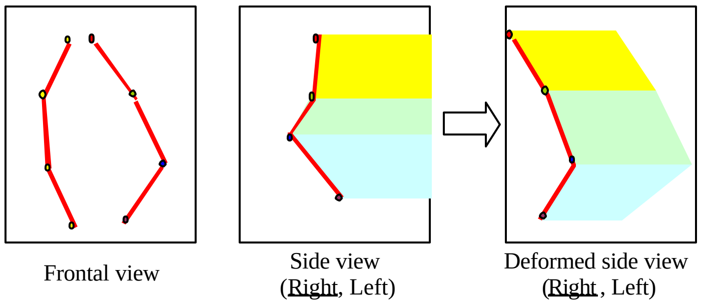


Figure 3-9: The side view is deformed to stick to the frontal view

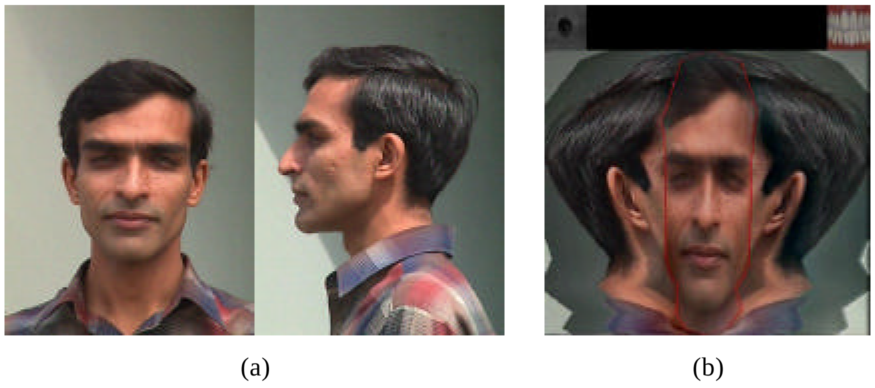
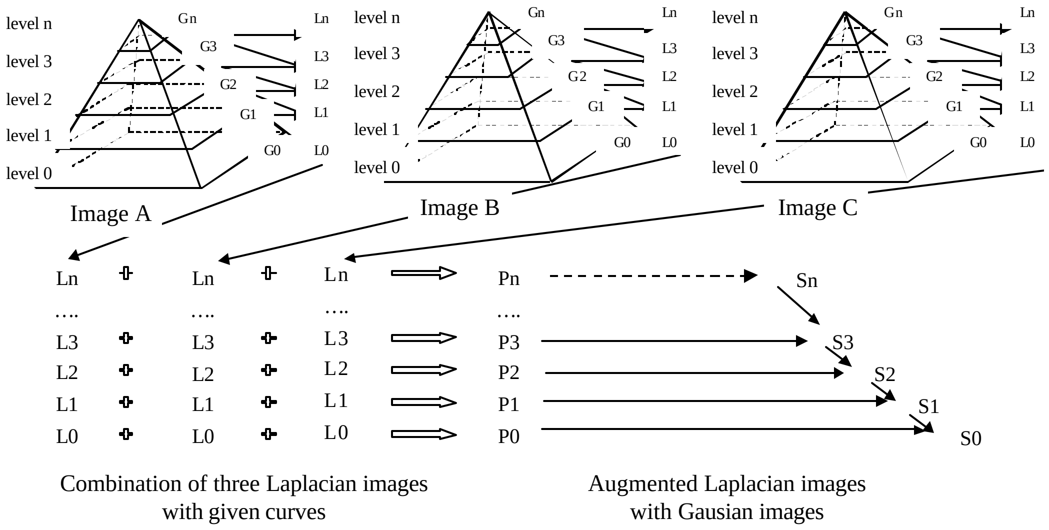


Figure 3-10: (a) Input images (b) Combination with geometrical deformation.

Using the process for geometrical deformation, the frontal and side views are stuck together without any geometrical mismatch. Look at the chin lines and face-hair lines on Figure 3-10 (b).

3.3.1.2 Multi-resolution image mosaic

The three resulting images after deformation are merged using pyramid decomposition method using the *Gaussian* operator [Burt 83] to smooth boundaries caused by non-coherent luminance for the two different photographs. We utilize REDUCE and EXPAND operators to obtain the G_k (*Gaussian* image), with $\boldsymbol{k} = \mathbf{0}, \ldots, \boldsymbol{n}$, and L_k (*Laplacian* image), which are described in detail in Section 3.1.3.2. Then we combine three L_k images on each level $\boldsymbol{k}$ on the defined curves, which are feature lines used in Section 3.3.1.1. Then the combined image P_k is augmented with S_{k+1} to get S_k, which is the result for each level. The final image is S_0. Figure 3-11 shows the whole process of the Multi-resolution technique to merge three images. Figure 3-12 shows an example from level 3 to level 2. This Multi-resolution technique is very useful to remove boundaries between the three images even though it does not provide illumination free images.



level n
level 3
level 2
level 1
level 0
Gn
G3
G2
G1
G0
Ln
L3
L2
L1
L0
Image A
Image B
Image C
Pn
P3
P2
P1
P0
Sn
S3
S2
S1
S0
Combination of three Laplacian images with given curves
Augmented Laplacian images with Gausian images

Figure 3-11: Multi-resolution technique for image mosaic using pyramid decomposition.

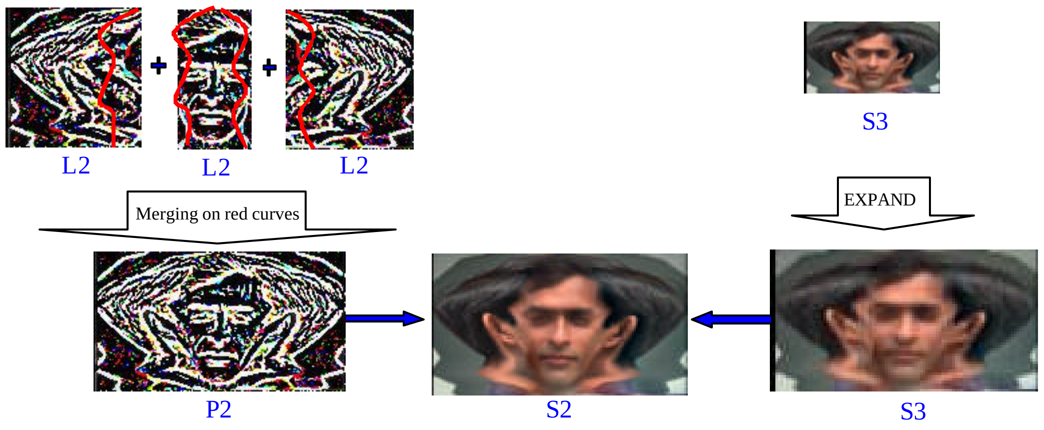


Figure 3-12: Multi-resolution techniques with an example from level 3 to level 2.

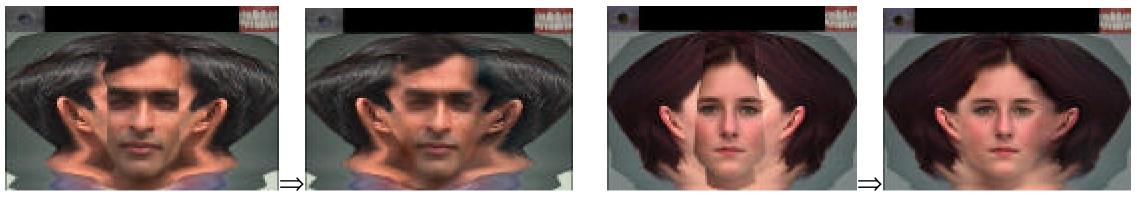


Figure 3-13: Two examples showing the effect of multiresolution.

Images of the eyes and teeth are added automatically on top of final texture images as shown in Figure 3-13, which are used to obtain full eyeball and teeth texture when we open mouth or rotate eyeballs for the facial animation.

3.3.2 Texture coordinate fitting

To give a proper coordinate on a texture image for each point on a head, we first project an individualized 3D head onto three planes, the frontal (*x*, *y*), the left (*y*, *z*) and the right (*y*, *z*) planes. With the information of feature lines used for geometrical deformation as shown in Figure 3-10, we decide on which plane a 3D-head point is projected. The projected points on one of three planes are then transferred to one of 2D-feature point's spaces such as the frontal and the side in 2D. Then they are transferred to the 2D-image space using the inverse transformation used in Section 3.1.2 for the Normalization. Then they are once more transformed to the combined image space following the same transformation used in Section 3.3.1.1 for the Geometrical Image deformation. The process is shown in *Figure 3-14*.

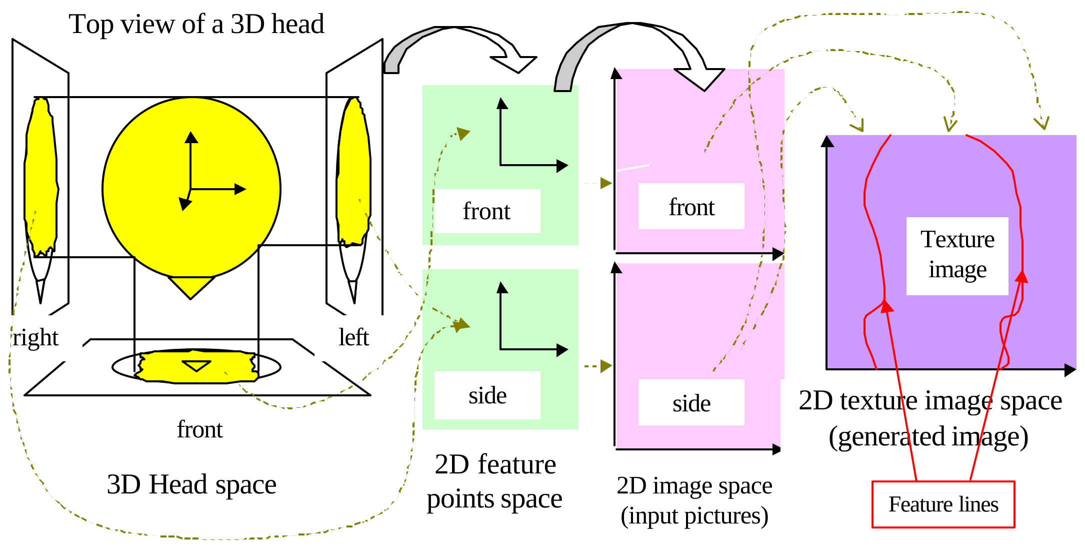


Figure 3-14: Finding texture coordinates on the texture image by projecting 3D surface points onto three planes.

To decide which plane the surface point is projected on, we follow the following criteria to find F_mask (frontal view), R_mask (right view), L_mask(left view), LR_mask(both right and left views).

1. Set a point as RFVflag if it is located on the right side of the both feature lines on the frontal view.
2. Set a point as LFVflag if it is located on the left side of the both feature lines on the frontal view.
3. Set a point as LSVflag if it is located on the left side of the feature lines on the side view.
4. if(LSVflag) mask[i] = F_mask;

   else if(!LSVflag && -0.001 < x < 0.0001) mask[i] = LR_mask;

   else if(!RFVflag) mask[i] = R_mask;

   else if(RFVflag && !LFVflag && !LSVflag && head_pt.x > 0.0) mask[i] = R_mask;

   else if(RFVflag && !LFVflag && LSVflag) mask[i] = F_mask;

   else mask[i] = L_mask;

3.3.2.1 Number of texture coordinates ^{3} number of surface points

The number of texture coordinates of the 3D object can be different from the number of surface points. *Figure 3-15* helps to understand the problem and solution.

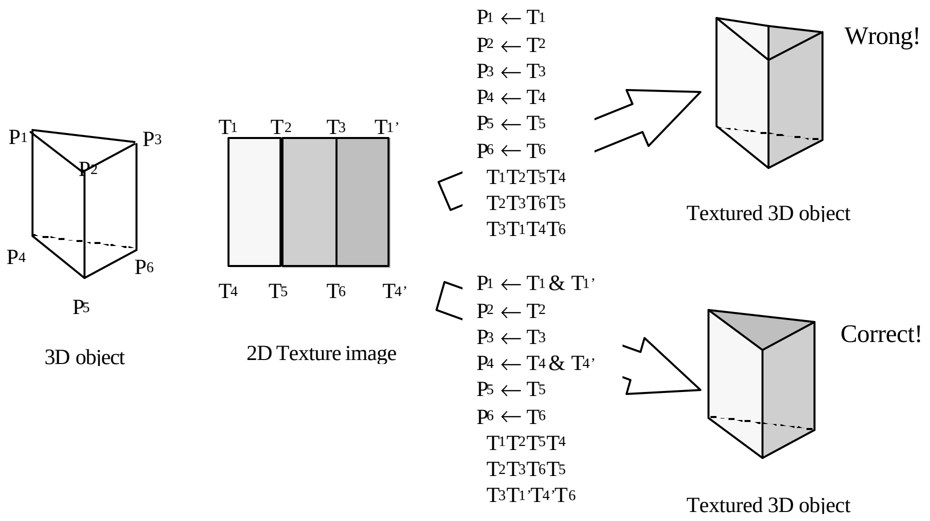
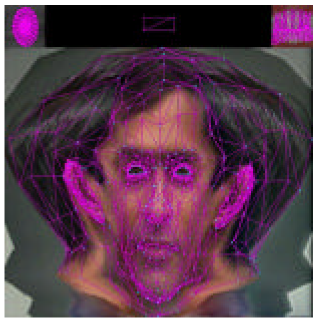


Figure 3-15: Why we need more texture coordinates then the surface point number?

When we assign only one texture coordinate for each point on the 3D surface, we may have problem in a certain region of the 3D object. To prevent this problem, we assign two coordinates for some points on the surface and give the polygon information using the texture coordinates, not using the point coordinate. For the 3D-head model, we assign two texture coordinates on the middle line of the backside view of the head. For this purpose, we force to make the middle line of the backside view of the head in a certain position (for example we set the *x* coordinate as 0) as discussed in Section 3.2.2.

1. Collect points P_k who is set as LR_mask (both right and left views) described above.
2. Collect polygons $POLY_k$ if one of whose point P_k is collected.
3. Give additional texture coordinate $T_{k'}$ on the point collected.
4. Create new polygons $POLY_k$ for the texture mapping by replacing T_k by $T_{k'}$.

Then all areas of the head have proper texture coordinates as shown in Figure 3-16 (a) where the texture coordinates laid on the texture image. It shows the central line of the backside view of the head have two texture coordinates. Figure 3-16 (b) shows how it results on a head with a piecewise line (feature points used for the front and side view image combination) on it. The left part of the overlaid line in Figure 3-10 (b) is from the frontal view image and the right part from the side (right) view image.


(a)

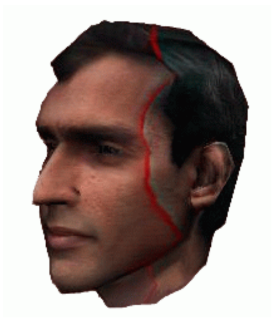

(b)

Figure 3-16: (a) Texture coordinates. (b) A textured head with curves on. The left part and the right part of the curves are the different sources of texture mapping.

The eyes and teeth coordinate fitting process are done with predefined coordinates and transformation related to the resulting texture image size, which is fully automatic after one process for a generic model.

3.4 Results and animation

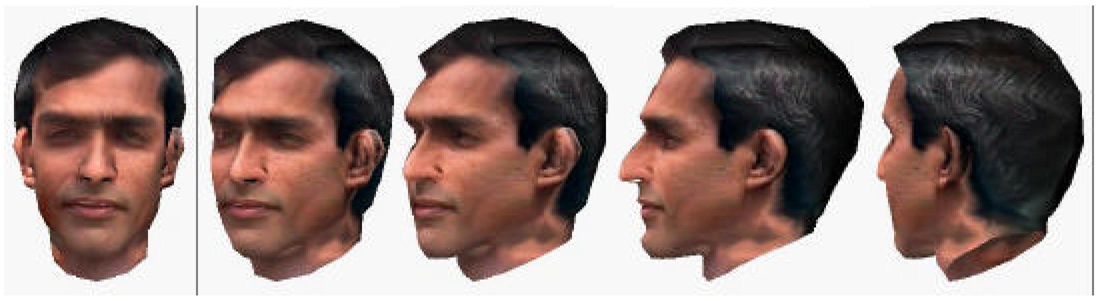


Figure 3-17: Rotation of a reconstructed head from photograph data. A backside has proper texture too.

A final textured head is shown in Figure 3-17, where input images are shown in Figure 3-10 (a). It has proper shape and texture on backside too, which makes full rotation of a head possible. Other examples from orthogonal photograph data are shown in Figure 3-18. Females and males in several different ethnic groups (Caucasian/Asian/Indian/African) covering wide range of ages are reconstructed using only two photographs and one generic model in Figure 3-8(a). For a child in Figure 3-18, it is not very easy to keep him in the same position without movement even for one second. Our algorithm allows a quite flexible solution to adapt this kind of case. As our generic model has short hairstyle and has only a few polygons on hair region, a reconstructed virtual human cannot have various hairstyle far from the one of the generic model, especially for the case of long hair. The total amount of data for the reconstructed heads in *OpenInventor* format is small considering their realistic appearance. The size of Inventor format (corresponding to *VRML* format) is about 200 KB. The texture image is stored in *JPEG* format and has sized about 5~50 KB depending on the quality of photographs; all examples shown in this thesis have size less than 45 KB.

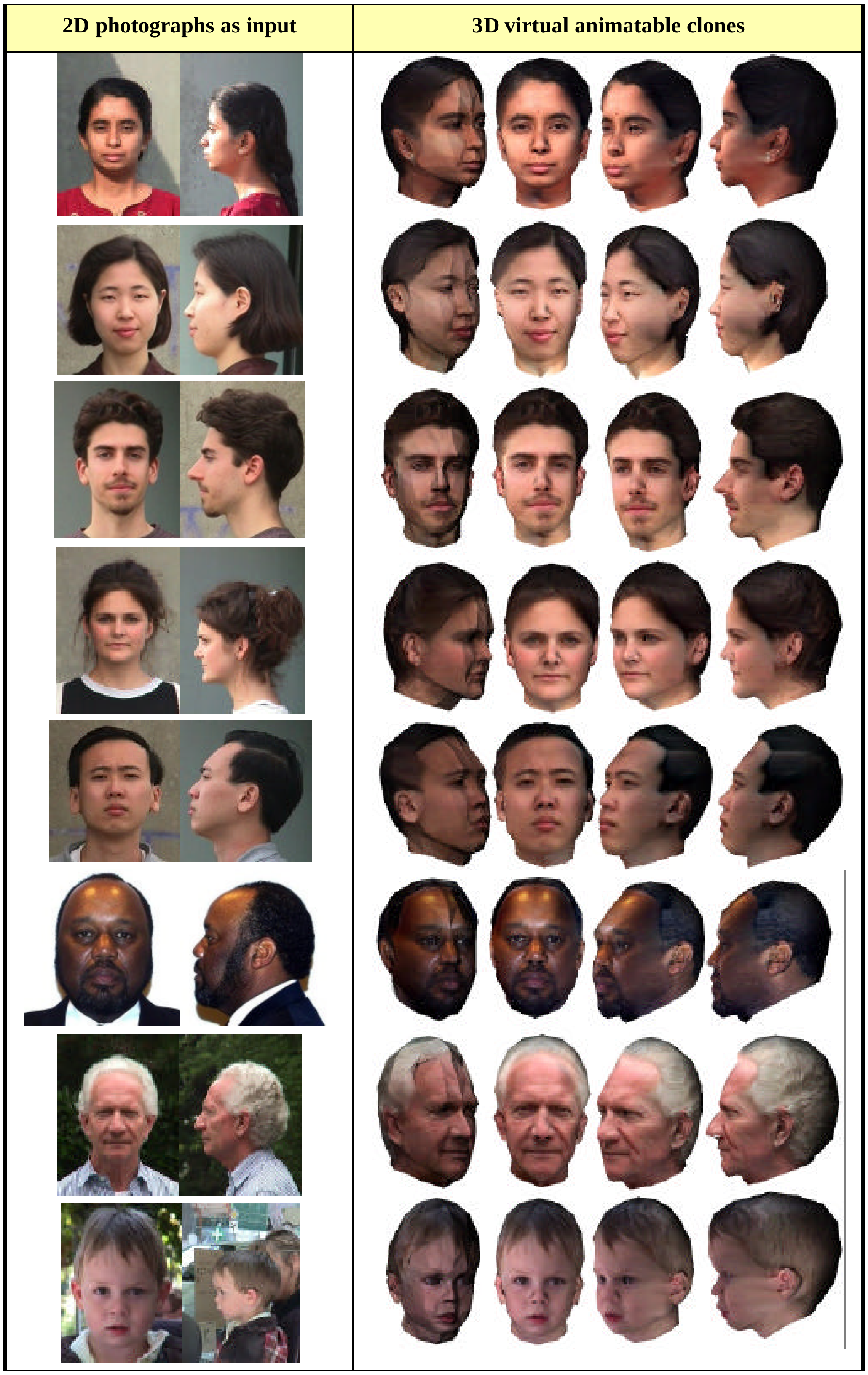


Figure 3-18: Examples of reconstructed heads from photographs. They can be animated immediately in virtual world.

3.4.1 Shape-texture separation

Good pairs of orthogonal photos are hard to come by, unless we do the photography ourselves in a controlled situation. Sometimes sets of images are available, some of which have clearer shape while others have good resolution. The frontal views in *Figure 3-20* and *Figure 3-21* are satisfactory both for shape and for texture resolution while the middle side views have better resolution but the rightmost side views have profile outline clear. For best results, we can combine shape data from the latter with texture image data from the former using our user-friendly three graphical managers in *Figure 8-3* where we show several expressions on the resulting head. The process is outlined in Figure 3-19.

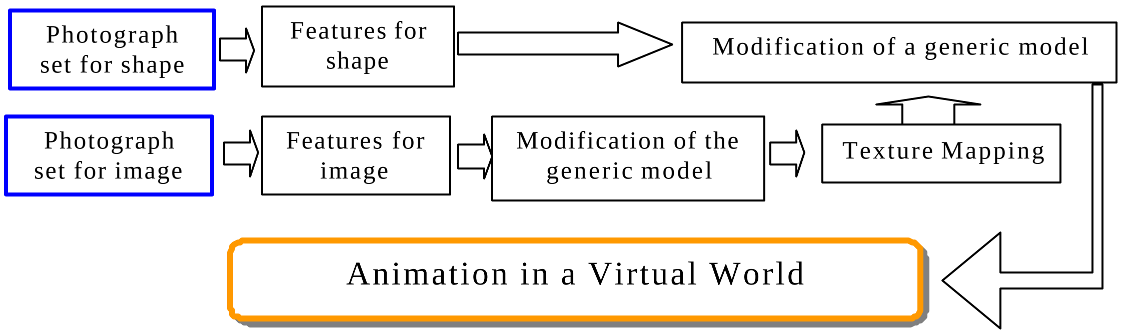


Figure 3-19: A procedure for reconstruction from shape-image-separated input photographs.

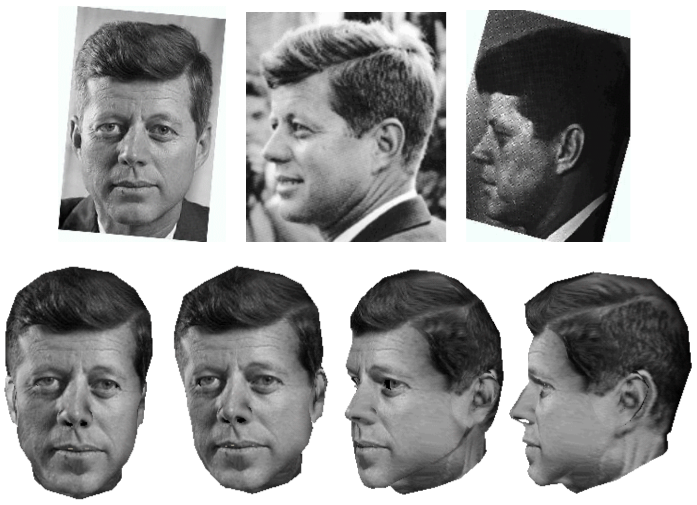
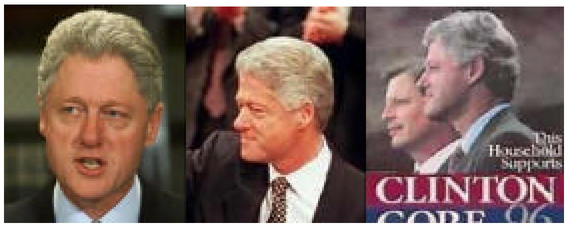


Figure 3-20: An example of a reconstructed head from three photos of John F. Kennedy.

3.4.2 Animation

We employ an approach for deformation and animation of a face, based on pseudo muscle design as described in Sections 2.7.1 and 2.7.4 based on Kalra et al.'s approach [Kalra 91][Kalra 92]. Muscle actions to stretch, expand, and compress the inside geometry of face are simulated by displacing or changing the weight of the control points. This deformation model for simulating muscle is simple and easy to perform, natural and intuitive to apply and efficient to use for real-time applications. In the multi-level approach [Kalra 91], motion parameters as Minimum Perceptible Action (MPA) are used to construct practically any expression and viseme. At the highest level, a script containing speech and emotions with their duration controls animation. An example of facial animation is shown in *Figure 3-21*.



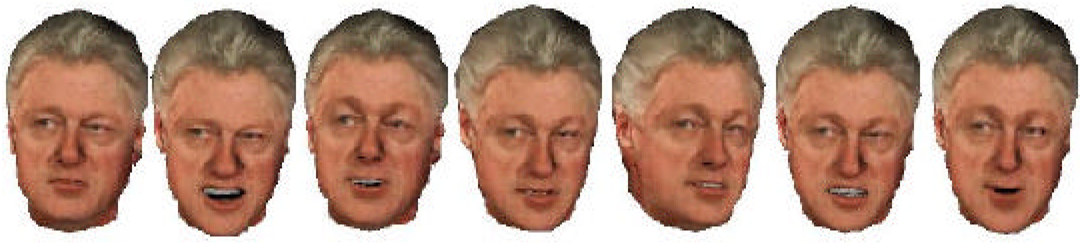


Figure 3-21: Another example of a reconstructed head from three photos of Bill Clinton and several expressions on it.

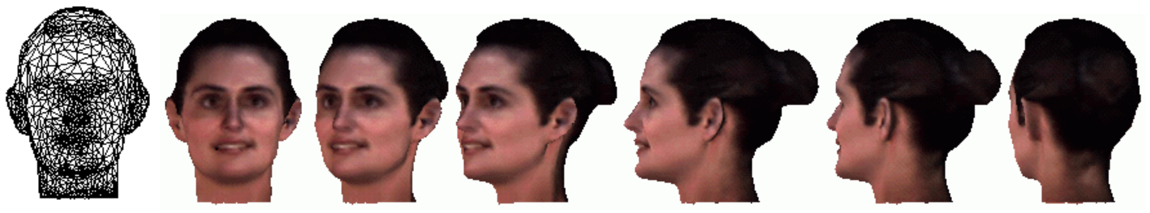
3.5 Validation of the face cloning results

As we mentioned in Chapter 1, there are two main categories for obtaining 3D human models, according to the different requirements for the output. The typical example of the first category is either CAD or medical application where the precise and exact location of the each pixel is important while the visual realism is not. The second category belongs mainly to the Virtual Reality application, where the real-time calculation and realistic and believable looking are more important than precision. Since our aim is more focused on real-time Virtual Reality application, the accuracy of the shape is less important than the visual resemblance for human cloning. Nevertheless, we present both validations to prove the efficiency of the feature-based approach.

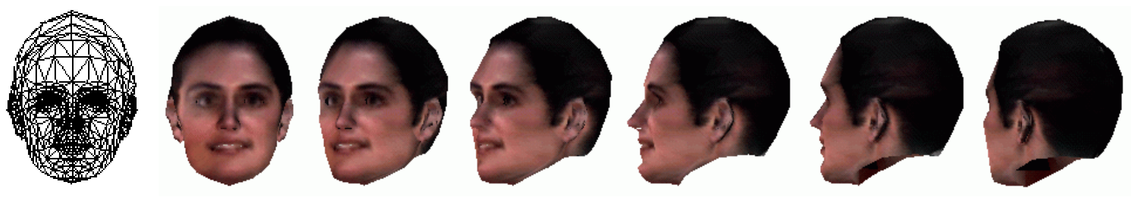
To validate the result, we use the output from a laser scanner as our input. In this case, we do neither need to worry about how to measure the real surface of the human subject nor the possible error produced by the laser scanner. We collect a model in VRML 2.0 format from a freeware web site, *http://www.cyberware.com/wb-vrml/tammy/tammy.wrz*, which was produced by Cyberware Digitizer for human body as described in Section 2.8. This *tammy.wrl* contains both the face and body parts. We use only the face part in this section and the body part is discussed separately in Section 7.7.

3.5.1 Visual validation

We use snapshot to take two orthogonal photos of the face part after loading the laser-scanned model. We use the generic model in *Figure 3-7*. The first and forth snapshots in *Figure 3-22* (a) are used as the input photographs (301x309) for the face cloning. The visual comparison of the original data and reconstructed data in *Figure 3-22* shows that the result is very much acceptable and feature based approach is efficient. The number of points on a head is 1257, where 192 of them are for teeth and 282 of them are used for the two eyes. This leaves only 783 points for the individualization of the face surface aside from eyes and teeth. Beside the small number of points, the output surface has animation information inside.



(a) snapshots of the laser-scanned model "Tammy"



(b) snapshots of the reconstructed model using feature-based face cloning program.

Figure 3-22: Compare snapshots in (a) and (b) taken in the same angles. The neck in (b) is shortened to make connection between the face and the body (refer Section 7.6.2).

3.5.2 Error measurement

The input surface (from the laser scanner) is a set of 3D points and so is the output surface (from the cloning program). However there the surfaces have different structure, which means the resulting head has inside structure such as eyes and teeth differently from the laser-scanned head. So we have to make error measurement after taking out the inside structures. Of course we perform this measurement after aligning the output surface in the same space of the input surface using scaling and translation. *Figure 3-23* shows two heads in the same space after translation and scaling. Two surfaces show some mismatch, which is the error, for example, the back part of the hair shows only the input surface where the various hairstyles are not included in this face cloning system. So the points on the left side of the vertical line on the last image Figure 3-23 are used for error measurement.

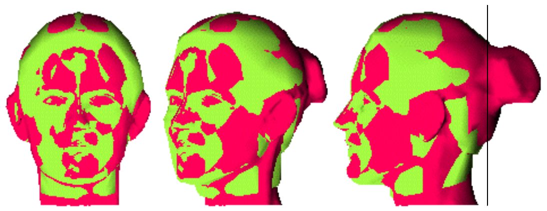


Figure 3-23: Two surfaces in different colors in the same space. The points on the left side of the vertical line on the last image are used for error measurement.

In order to compute the error between two surfaces, we use distance measurement between two surfaces. For each point P_i on the output surface, we calculate the normal vector n_i by taking the average of normal vectors of connected triangle faces on the surface. Then we make the projection of n_i to the input surface and get the intersection Q_i. To calculate the projection of $n_{i,}$, we first calculate the intersection point between the normal vector and each extended plane of a triangle face on the input surface. Then we can decide if the intersection is inside the triangle face or not. Usually there are one or two intersection points of the normal vector on the input surface. We select the nearest intersection point, Q_i, and calculate the distance between the each point P_i on the output surface and the intersection point Q_i. The distance between the point P_i and Q_i is the error E_i of P_i. *Figure 3-24* shows the 2D case.

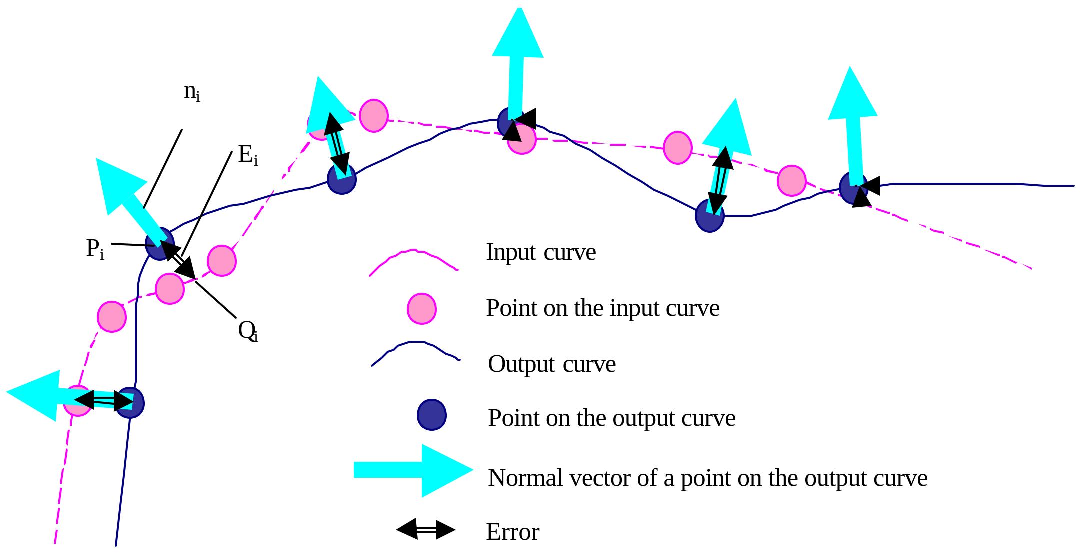


Figure 3-24: 2D case to show the calculation of the distance for each point on the reconstructed curve to the original input curve by calculating normal vector.

The surface error *E* is a summation of each error *Ei* and the error % is calculated as follows where output surface has *n* points and *Ri* is the distance of the distance between *Qi* and the center *O* of the surface.

$$Error\% = 100.0\frac{\sum_{i=1}^{n}\frac{Ei}{Ri}}{n}$$

See *Figure 3-25* for better understanding of the notes.

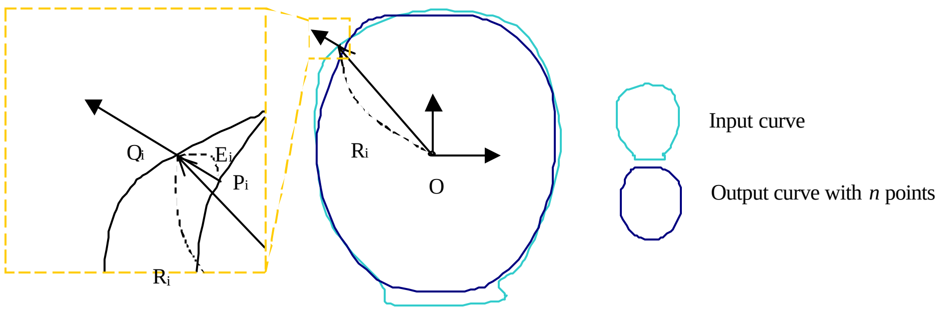


Figure 3-25: 2D case to get error % between 2D curves.

When the bounding Box of points on the output surface which are calculated for errors has the size *167.2 X 236.651 X 171.379 where x(-84.1756 , 83.0243), y(-112.715, 123.936)* and *z(-99.808, 71.5709)*, the average error is 3.55236. Here total 756 points are used for error measurement, where the eyes, teeth and some of back hair parts (indicated in *Figure 3-23*) are exempted.

The errors are distributed as shown in *Table 3-1* with minimum error = 0.00139932 and maximum error = 29.6801. *Figure 3-26* shows where the error comes. The biggest error between two surfaces comes from ear region as shown in *Figure 3-26* (a). Since the structure of ears is so complicated and it is not possible to adapt accurately with feature based method and even with range data equipment. The hairlines between the hair and face regions create the next error sources a shown in *Figure 3-26* (b) and (c). This error comes from the special hairstyle of the generic model.

The final error is 2.84306 % of the surface.

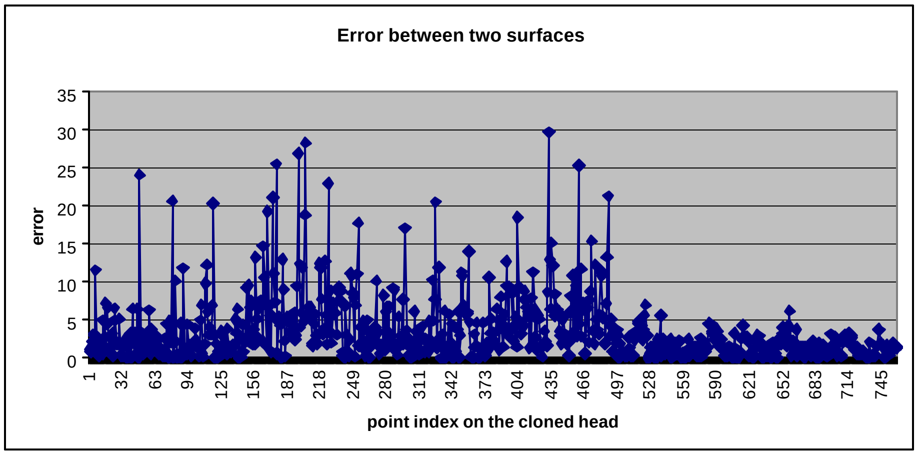


Table 3-1: Error distribution for the face cloning from orthogonal photograph where the bounding box of the head has the size 167.2 X 236.651 X 171.379.

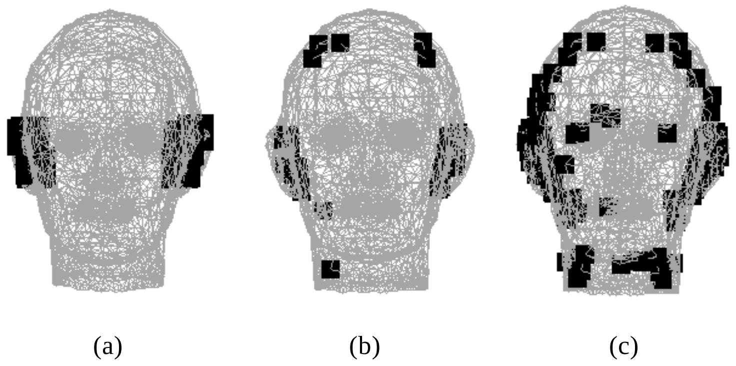


Figure 3-26: Points with error bigger than (a) 15 (b) 10 and (c) 5.

3.6 Summary

We described a method to create virtual animatable faces from orthogonal photographs. One generic model is prepared in advance to have optimized polygon distribution and animation structure. The heavy calculation for Sibson coordinates for shape modification is done only once with the generic model. The generic model is converted to any individualized head with feature-based approach, whose modification is processed with characteristic points (features). Using DFFD, the modified shape has smooth surface. A seamless texture image generation combining two separate images is presented. The experiments on the several ethnic groups in various aging show the feature-based approach is very convincing methodology to approximate human face in a fast and efficient way.

As we mentioned in Section 2.4, there are the problematic of the orthogonal photographs are:

- How much error-tolerant is the system for a given input photographs?
- How to get 3D from a set of 2D from non-perfectly-orthogonal photographs?

  Section 3.2.1 proposes the solution to handle non-perfectly-orthogonal photograph input. The graphical interface described in Section 8.1.1 shows there is the possibility for users to handle quite wide range of images.

- How much automatic and how much robust is the feature detection?

  Section 3.1.3 proposes the semi-automatic solution.

- How well is the modification working for the smooth surface?

Section 3.2.3.1 proposes more sophisticated Free Form Deformation.

- How well done is the automatic texture image generation without any problems and how much can we keep the resolution of input photographs for the final visualization?

  Section 3.3.1 described how we generate a texture image without any feature mismatch keeping resolution of the major frontal part as high as the frontal photograph input.

Our technique for the face cloning has the general approach, but this work solved some of the key problems of the approach, namely

- More robust facial feature detection;
- Modification of the generic head model in 3D using DFFD
- Techniques for producing a more realistic and natural looking texture mapped visualization of the individualized head model.

The difficulties of fully automatic feature detection come from various colors depending on each person, many shadows on face and also wrinkles and spots. Our two steps semiautomatic structured feature detection methods applied first for the key feature points and then for other feature points bring reliable results using piecewise affine mapping and structure snake. It has shown its efficiency and robustness in several public demonstrations, where hundreds people were reconstructed from orthogonal photograph data. The reconstruction from two photographs needs low-cost commercial equipment and takes just few minutes.

The result from the range data has the better visual result for output, especially non-featured area such as cheeks, forehead and chin. However it has not only some difficulty for featured area such as eyes and lips, but also some limitation due to equipment since it requires usually either expensive or sophisticated equipment. The best result can be obtained when we combine two input data, photographs for feature parts and also for the high quality texture image and range data for non-feature parts. The extension of the face cloning system to treat range data is described in the Chapter 4.

Chapter 4 Giving Animation to Range Data

In Chapter 3, we have shown how to make a virtual cloned face using two photographs and this chapter is devoted to virtual face cloning using range data. Even though the face cloning using only photographs shows reasonably good fitting to the person, it is feature based, so that other points are approximated using the predefined generic shape and feature points detected. As Table 2-3 shows, the usage of range data gives useful precise shape on overall points. So when there is a source to provide a good surface data (range data), we combine the photographs data and range data utilizing feature points from photographs input and non-feature points from range data to get the accurate face cloning for animation. The global approach belongs to a modification a generic model with Section 2.4 'Features on photographs (organized) and a generic model' category, but also we generalize this method to range data input to give animation structure.

Since our aim is to provide a realistic virtual human for real-time animation as real human acts in the real world, the most important criteria is to make a photograph-realistic looking virtual face with optimized vertices for the real-time animation. Our methodology in this chapter has two aims such as firstly to combine photograph input and range data shape input to improve the given range data surface shape and secondly to give animation structure on the static range data shape. The range data used here can be obtained from any available resources such as stereoscopic camera or laser scanner (VRML format for our experiment). The reconstructed 3D-face can be animated immediately with any given expression parameters and any face created from this method is morphable and can be used to create a large population of faces, which is described as an application system in Chapter 5.

Figure 4-1 shows the overall flow of how to give animation structure on the range data. The flow shows very similar structure as the one in *Figure 3-1* for the face cloning using orthogonal photographs. The difference between them is that here we do not use feature detection for the side view since the input range data provides the depth value. And also we use two steps modification to approximate the shape such as rough shape modification first with feature points and then fine shape modification with non-feature points.

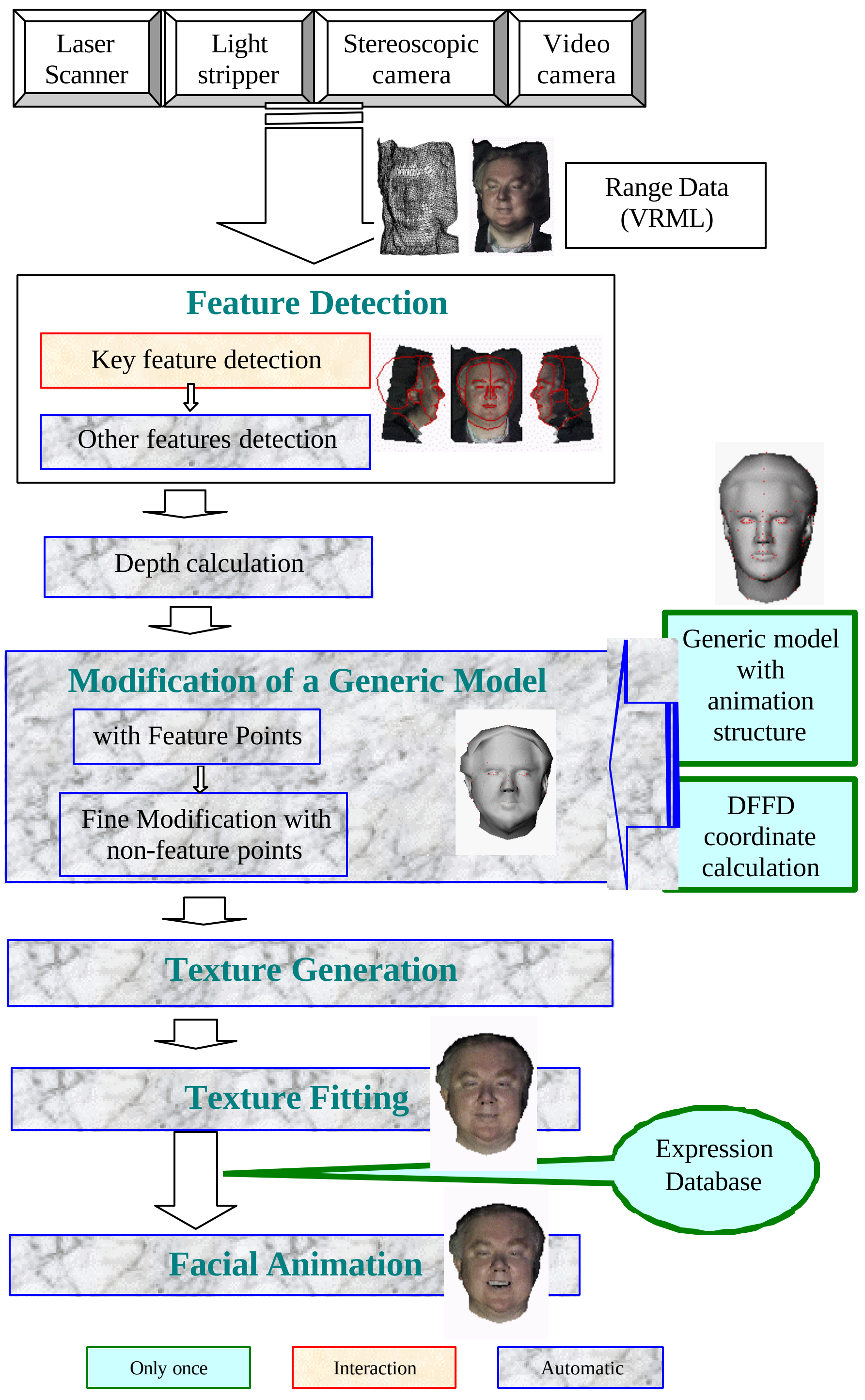


Figure 4-1: An overall flow diagram for face reconstruction from range data input.

4.1 Feature detection

Feature detection from 2D-texture image for range data is one of the main steps. When we refer to feature detection, it means to catch the 2D positioning data of most visible points on a face such as eyebrow, eye outlines, nose, lips, the boundaries between hair and face, chin lines and etc.

For our experiment, we utilized data copied from the Turing Institute [http:turing] where stereoscopic camera is used to generate range data of faces. We can convert some part of the range data in VRML format to appropriate to our need as input if necessary.

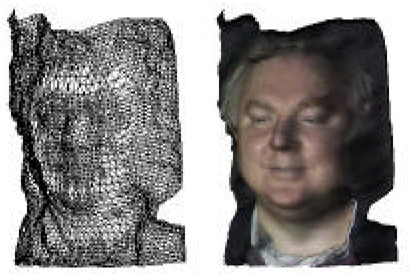
We visualize the range data in frontal view as it is and rotate and translate to show the right and left views together as shown in *Figure 4-2* (a). The height checked in the frontal view is used to place three 2D frames for frontal, right and left views automatically. Then like the case for the orthogonal photographs, normalization parameters (scaling and translation) bring the range data into feature points space.

Feature detection is applied only for frontal view. However right and left views are shown together in *Figure 4-2* (b). Whenever a point on the frontal view is detected, its depth *z* is calculated automatically and it visualizes the *z* value on a window. First we collect a certain number *n* (user's definition) of nearest points in *(x, y)* plane. Then apply

$$z = \sum_{i=1}^{n} \frac{\frac{1}{d^2(\vec{p}, \vec{p}_i)}}{\sum_{i=1}^{n} \frac{1}{d^2(\vec{p}, \vec{p}_i)}} \cdot z_i$$

where $\vec{p} = (x, y)$, $\vec{p}_i = (x_i, y_i)$ and $d^2(\vec{p}, \vec{p}_i) = (x_i - x)^2 + (y_i - y)^2$.

Some feature points on side views are defined interactively since they are not provided in range data and most range data methods have problems obtaining proper hair shape because of the characteristic of high reflection of hair structure.


(a)

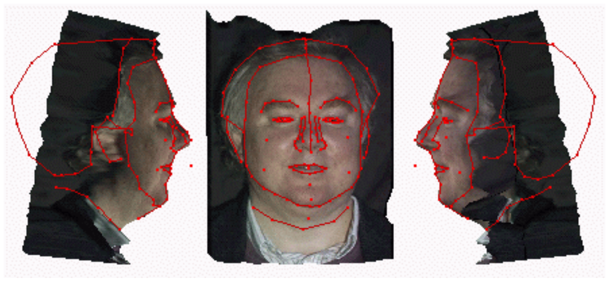
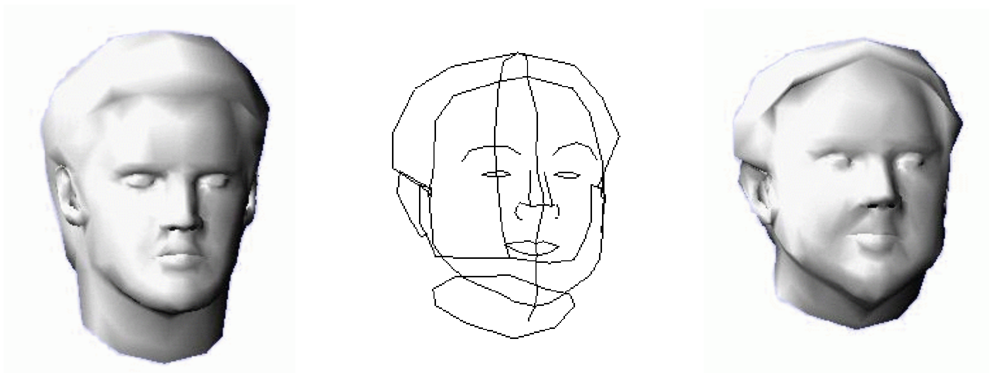

(b)

Figure 4-2: (a) Input range data with details but without functional structure and backside head. (b) Feature detection on the frontal view and automatic depth calculation on side view

4.2 Shape reconstruction

We detected feature points on photograph images. Using the 3D features after depth calculation using the 3D position data on the range data, we apply DFFD for the head by utilizing the features as control points.

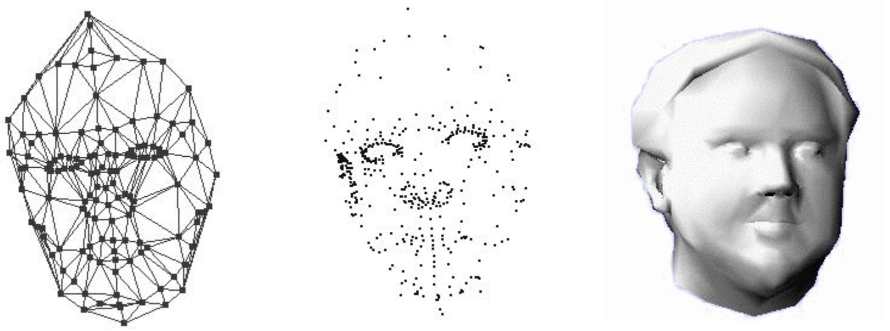
As shown in *Figure 4-3*, this is a feature based rough matching method; it does not attempt to locate all points on the head exactly. It is in contrast to range data from laser scanners, which is supposed to have more accurately matching shape for overall area.



(a) The generic model (b) Features in 3D (c) After DFFD

Figure 4-3: The shape modification using DFFD.

4.2.1 Fine modification for range data



(a) (b) (c)

Figure 4-4: (a) Delaunay triangles created with feature points. (b) Points collected for fine modification using Delaunay triangles. (c) The resulting head after the fine modification. Compare the chin lines in Figure 4-3 (c).

The modification by DFFD is feature points based, which means only features are used for modification. When range data is available from any possible source such as las er scanner or stereoscopic camera, we are able to utilize not only feature points but also other points from the range data. So we apply one more step to get accurate positions for not only feature points but also other points from the range data.

1. Some of feature points are chosen for a fine modification. Not every feature point is candidate for fine modification since some points in range data, where for example hair and ear regions have non-proper position value with high error. We collect feature points only when their positions on a modified head are inside certain limitation of corresponding points on original range data.
2. We calculate Delaunay triangles of chosen feature points and collect non-feature points inside Delaunay triangles. We check it using Barycentric coordinate. The Delaunay triangles and collected points on a surface are shown in *Figure 4-4* (a) and (b).
3. Project the points inside each triangle onto a certain plane to find the corresponding depth in the range data. We use three kinds of projection of points depending on a normal vector of a corresponding triangle. It can be projected on either *(x,y), (y,z),* or *(z,x)* plane.
4. Again the *n* nearest points in 2D or 3D are collected and then an inverse distance function for depth calculation is applied to get the corresponding coordinate in a range data such as

$z = \sum_{i=1}^{n} \frac{\frac{1}{d^2(\vec{p}, \vec{p}_i)}}{\sum_{i=1}^{n} \frac{1}{d^2(\vec{p}, \vec{p}_i)}} \cdot z_i$ where $\vec{p} = (x, y)$, $\vec{p}_i = (x_i, y_i)$ and $d^2(\vec{p}, \vec{p}_i) = (x_i - x)^2 + (y_i - y)^2$ for the case of (x, y) plane. Here we apply only reasonably big triangles (for example we consider a triangle is reasonably big if the area of it is above average of area of every triangle) for fine modification since if we apply projection to every triangles, it sometimes creates irregular shape, specially in the eye region.

Figure 4-4 (c) is the final result after fine modification. Compare it with *Figure 4-3* (c), especially in the parts of the chin lines. It recovered the original shape while the modification with only defined feature points is not able to do it due to lack of feature lines on the region. *Figure 4-4* (c) has also a smoother surface. Without the first modification step using DFFD with feature points, it is difficult or takes much longer time to get detailed matching. Two steps modification approach, first rough matching with FFD and then fine matching with simple projections, obtains the result easier and guarantees the convergence for the fine shape.

This method using two-steps modification on the range data is a corresponding method to the "Adaptive mesh " methods by Lee et al. [LeeY 95] (Refer Section 2.7.3). Their image analysis technique that searches for salient local minima and maxima in the range image of the subject to adapt the generic face mesh to the laser-scanned range and reflectance data is not always reliable due to the noise. Our two step approach guarantees more robustness and rapid result.

4.3 Results and animation

For the texture mapping, we can extend some part of the frontal image. Since we know the location of boundary between hair and face from feature detection step, we produce new image with hair image and use it as side image like orthogonal photograph input. Then we can apply the same algorithm as orthogonal photograph input case.

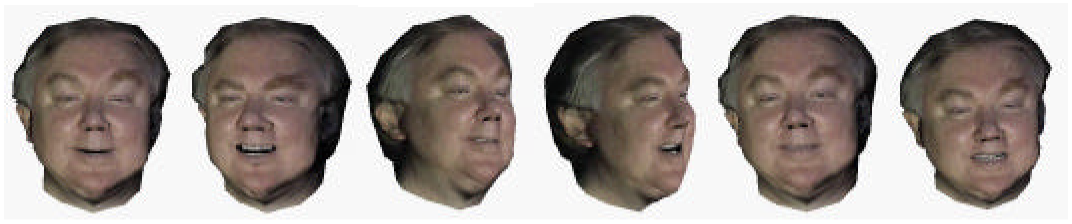


Figure 4-5: The result of giving functional structure on a range data. This reconstructed head is able to animate with given expression parameters.

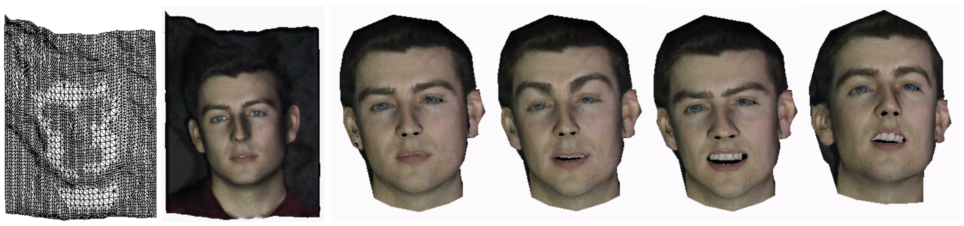


Figure 4-6: Another example of giving functional structure on a range data.

Figure 4-5 shows a result with range data shown in *Figure 4-2* (a). It shows several animated faces with given expression parameters. The head in *Figure 4-2* (a) has a nice shape on the frontal view, but not have a good shape for back part and ear part. In addition, it does not have any functional information how to animate. Through our process using two-steps modification of the generic model (adapting the shape and passing an existing animation structure) to the given range data, it keeps nice shape and enables to animate with given expression. *Figure 4-6* is showing another example of range data.

4.4 Conclusion

We described an extension of the face cloning methodology to treat range data. There are two main problems with range data. First the result of the range data does not have a smooth surface for all regions. It has serious noise especially on feature area such as hairy or shady parts. Second the result of range data has only static shape without any animation structure. Our methodology solves these problems by combining strong points only from photograph and range data to make a better surface for overall area. It also adds the animation structure on the static range data input by adjust the generic head to the range data. The resultant head has optimally distributed triangulation and animation structure inherited from the generic head and the surface has best fitting to the subject inherited from the input range data.

The difference between photograph input and range data is first range data uses feature detection only for the frontal view and second we use two steps shape modification for the range data. The two steps shape modification is first based on features and second based on the overall points matching, guarantees the convergence of a generic model to match the data to get higher accuracy.

The result from the range data has the better visual result for output, especially non-featured area such as cheeks, forehead and chin. However it has not only some difficulty for featured area such as eyes and lips, but also some limitation due to equipment since it requires usually either expensive or sophisticated equipment. The best result can be obtained when we combine two input data, photographs for feature parts and also for high quality texture image and range data for non-feature parts. Our system dealing with the range data provide the possibility to get the results with highly realistic animatable virtual faces with small number of polygons with high accuracy.

Chapter 5 Real-time 3D Face Morphing System

This chapter is devoted to a 3D-face morphing application of cloned faces. This is used for two kinds of applications. One is for the real-time morphing among several faces and the other for creation of new faces from existing virtual faces.

5.1 Review

5.1.1 Morphing

The term morph, short for metamorphosis, has been applied to various computer graphics methods for smoothly transforming geometric models or images. Morphing techniques can be classified into image based methods and object-space methods. In most cases, the image-based methods are used for 2D morphing and the object-space methods for 3D morphing.

5.1.1.1 Image morphing

Image morphing has three terms to be considered, which are feature specification, warp generation methods, and transition control [Wolberg 98]. These areas relate to the ease of use and quality of results. Feature specification is the most tedious aspect of morphing. Although the choice of allowable primitives may vary, all morphing approaches require careful attention to the precise placement of primitives. Given feature correspondence constraints between both images, a warp function over the whole image plane must be derived. This process, which we refer to as warp generation, is essentially an interpolation problem. Another important problem in image morphing is transition control. If transition rates are allowed to vary locally across in-between images, more interesting animations are possible. The main methods are based on mesh warping [Wolberg 90], field morphing [Beier 92], radial based functions [Arad 94], thin plate splines [Litwinowicz 94], energy minimization [LeeS 96], work minimization [Gao 98] and multilevel free-form deformations [LeeS 95]. A tradeoff exists between the complexity of feature specification and warp generation. As feature specification becomes more convenient, warp generation becomes more formidable. Some methods do not guarantee the one-to-one property of the generated warp functions while others derive one-to-one warp functions, but the performance is hampered by its high computational cost. Most of the morphing methods and applications require manual user intervention to specify matches between given objects. In many cases, significant user involvement is required to achieve a pleasing visual result. It is desirable to eliminate, or at least to limit, this involvement in morphing design, and to have a more automatic process. In addition the speed is quite low, requiring at least several seconds due to the heavy calculation.

The traditional formulation for image morphing considers only two input images at a time, i.e. the source and target images. Morphing among multiple images, referred as polymorph, is understood to mean a seamless blend of several images at once.

5.1.1.2 3D-shape morphing

A metamorphosis or a 3D morphing, for example topology independent morphing in a short animation film “Galaxy Sweetheart” [Magnenat-Thalmann 89] in late ’80, is the process of continuously transforming one object into another [Kent 92][Lerios 95][Arai 96]. Since there is no intrinsic solution to the morphing problem, user interaction can be a key component of morphing software if two objects do not share the same topology. Most efforts have been dedicated to the correspondence problem, and we still lack intuitive solutions to control the shape interpolation.

5.1.2 Generation of new population

Techniques for generating virtual populations have been investigated by DeCarlo et al. [DeCarlo 98]. They vary a geometric human face model using random distances between facial points sampled from distributions of anthropometric statistics, but they do not consider a real-life face data as an input and realistic texture variation.

5.2 Morphing between two faces in 3D

The idea of combining 2D-image morphing and 3D-shape morphing has been little exploited. Nevertheless, this seems a natural approach to better results with less effort in some cases. Furthermore, were this to be combined with face cloning methods, user interactions could be avoided.

In this chapter, we vary aspects of both 2D and 3D representations in creating new virtual faces, showing how easy and fast the smooth morphing can be achieved. When we morph one person to another in 3D, we must deal with alteration in both shape and texture. Since every face created through the methods described in Chapter 3 shares the same topology, it is relatively easy to vary head shapes in 3D just using a simple linear interpolation of 3D coordinates if they are created from one generic model. On the other hand, it is less straightforward to carry out 2D morphing of textures since this requires some feature information specifying correspondences between specific points in the two images. Here we use a simple method, drawing on 3D information to facilitate 2D morphing among texture images.

5.2.1 Interpolation in 3D-shape

Every head generated from one generic model shares the same topology with the generic model. Then the resultant 3D several shapes are easily interpolated. An interpolated point P between P_L and P_R is found using a simple linear interpolation such as $P = aP_L + bP_R$. Figure 5-1 shows a head after interpolation where $a = b = 0.5$.

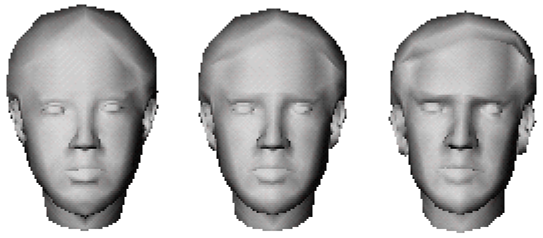


Figure 5-1: Shape interpolation

5.2.2 Image morphing

Beside shape interpolation, we need two items such as new texture coordinates and new texture image to obtain intermediate texture mapping. First texture coordinate interpolation is performed and then 2D texture image morphing follows.

5.2.2.1 2D interpolation of texture coordinate

It is as straightforward as 3D-shape interpolation. An interpolated texture coordinate C between C_L and C_R is found using a simple linear interpolation in 2D such as $P = aC_L + bC_R$.

5.2.2.2 2D-image metamorphosis based on triangulation

To create a new texture image and coordinates, we use texture morph with a given set of cloned heads with textures. As it was mentioned in the previous sections, image morphing normally considers feature specification, warp generation, and transition control. When we, however, use texture image with 3D head information, feature specification and warp generations are automatically given by the same structure of 3D head topology and only a simple calculation based on triangulation with given ratios among input texture images is performed.

We morph two images with a given ratio using texture coordinates and the triangular face information of the texture mapping; we first interpolate every 3D vertex on the two heads. Then to generate new intermediate texture image, we morph triangle by triangle for every face on a 3D head. Parts of image, which are used for the texture mapping, are triangulated by projection of triangular faces of 3D heads since the generic head is a triangular mesh. See Figure 3-16 (a) and note that the triangulation on the texture image is optimized with finer triangles for highly curvature/color-varied region and bigger triangles for smoother region, which results a very smooth texture morphing. With this information for triangles, Barycentric coordinate interpolation is employed for image morphing. Figure 5-2 shows that each pixel of a triangle of an intermediate image has a color value decided by mixing color values of two corresponding pixels on two images. Three vertices of each triangle are interpolated and pixel values inside triangles are obtained from interpolation between two pixels in two triangles with the same Barycentric coordinate. To obtain smooth image pixels, bilinear interpolation among four neighboring pixels is processed.

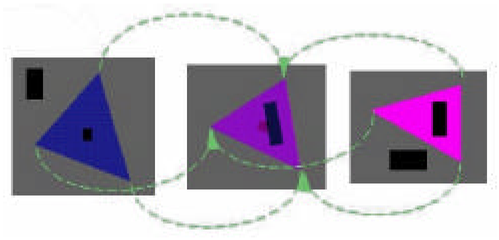
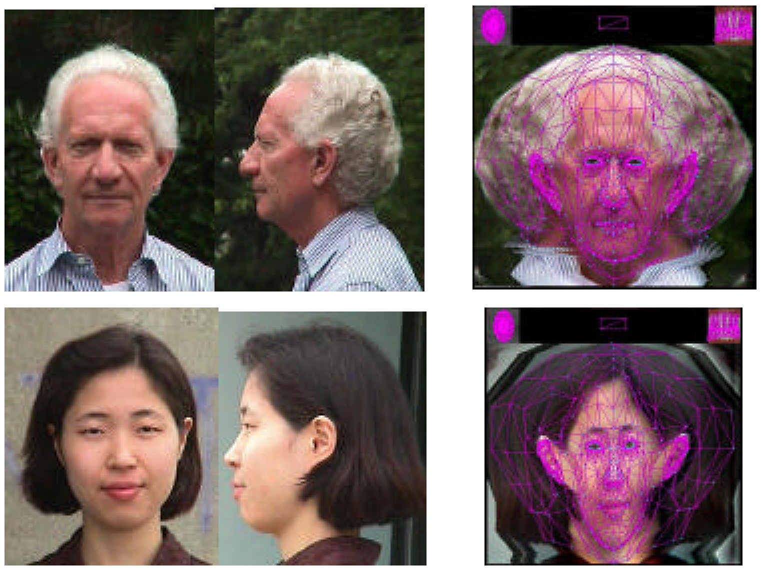


Figure 5-2: Image transformation using Barycentric coordinates.

We process the triangles on the image one by one. The new problem is how to fill the triangle in an efficient way. For the three vertices P_1, P_2, and P_3 of a triangle, we rename them according to the x coordinate in descending order like $P_1 \leq P_2 \leq P_3$. Then a line between P_1 and P_2 and one between P_1 and P_3 are compared to fill the triangle to find the range of y for a given x where $P_1 \leq x \leq P_2$. After similar checking is performed to find (x, y) where $P_2 \leq x \leq P_3$.



(a) (b)

Figure 5-3: (a) The input frontal and side view images of (Caucasian male, Asian female). (b) Generated texture images with texture coordinates.

For each pixel (x, y) in an intermediate image, the Barycentric coordinate (u, v, w) is found inside a triangle with three points P_1, P_2, and P_3. There are corresponding points in two input images, say P_{1L}, P_{2L}, and P_{3L}

for the first image and P_{1R}, P_{2R}, and P_{3R} for the second one. Then $uP_{1L} + vP_{2L} + wP_{3L}$ is the corresponding pixel in the first image and vice versa for the second. The color value $M(x, y)$ of a given pixel (x, y) is found from linear interpolation between the color value $M_L(x, y)$ of the first image and $M_R(x, y)$ of the second as the formula below.

$$M(x, y) = r\, M_L(uP_{1Lx} + vP_{2Lx} + wP_{3Lx}, uP_{1Ly} + vP_{2Ly} + wP_{3Ly})$$
$$+ (1 - r)\, M_R(uP_{1Rx} + vP_{2Rx} + wP_{3Rx}, uP_{1Ry} + vP_{2Ry} + wP_{3Ry})$$

To show 2D-image metamorphosis, we show two examples of faces as seen in **Figure 5-3**. The two images in **Figure 5-3** (b) shows the texture coordinates with visible triangles which are the triangular faces of a corresponding 3D head. The triangulation on two texture images shows very similar structure each other.

We vary the morphing ratio *r* to show dynamic morphing. **Figure 5-4** illustrates intermediate texture images. Pixels like eyes, mouth and some part of hair regions, which are not covered by triangles, are not calculated to get intermediate values, but are not used for texture mapping anyway. If different eye colors are used, it interpolates eye colors too.

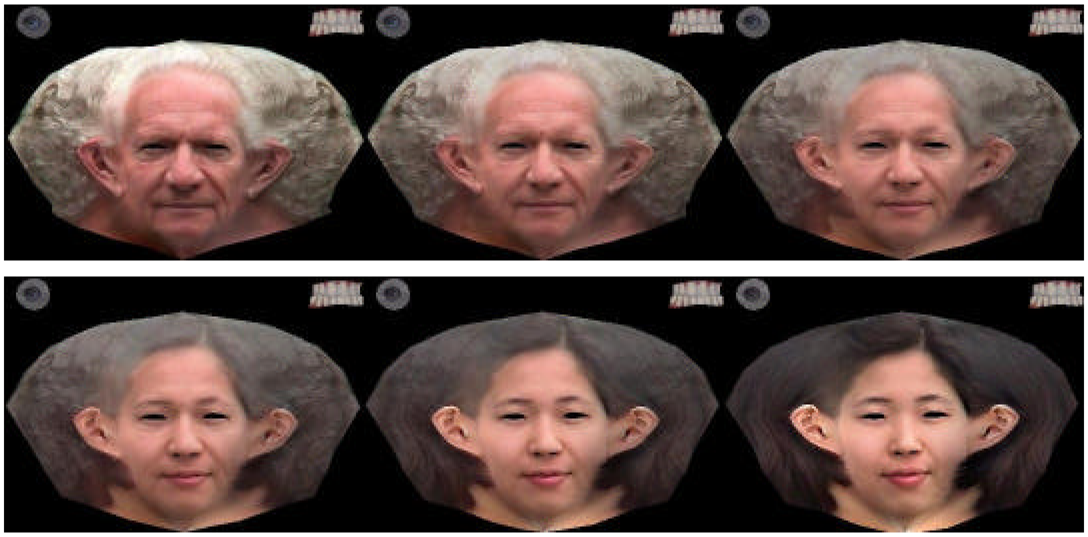


Figure 5-4: Intermediate images between two texture images in the previous figure for Caucasian male and Asian female.

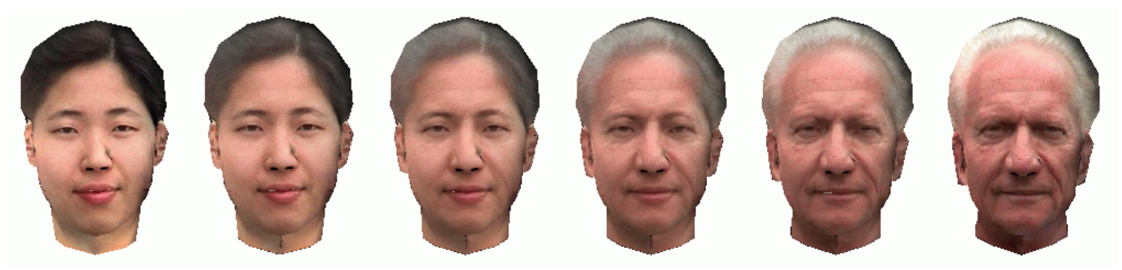


Figure 5-5: 3D Morphing steps between two heads.

The resulting head manifests very smooth images without any holes in the textured parts. Figure 5-5 shows the final result of 3D morphing. It depicts the results of interpolating the shapes, the skin and the hair colors between an aged Caucasian male and a young Asian female.

Since it is a morphing in 3D, changing the view is an easy task to enable the visualization of various profiles. **Figure 5-6** shows two examples of a Caucasian male with another male and the Caucasian male with the Asian female in profiles.

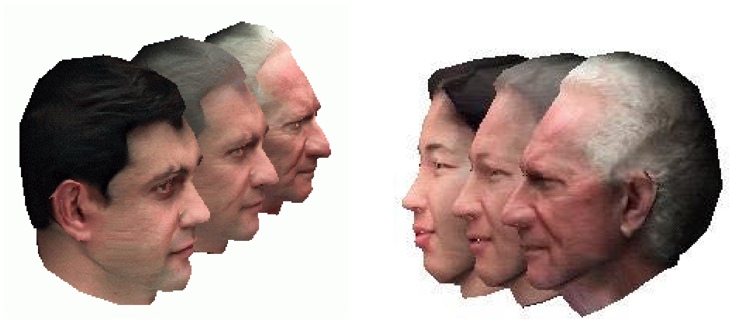


Figure 5-6: Different views of morphing between two among three people. They show the change of shape and image morphing. Each middle head is mixed with a ratio of 50% for both the shape and the image.

5.3 Variations for shape, image and expression

It is possible to set different ratios for the interpolation of the 3D shapes and for morphing images. Figure 5-7 shows different ratios in morphing for shape and texture images. The Asian female looks older with a lot of wrinkles and white hair, which are borrowed from a Caucasian male on the right side.

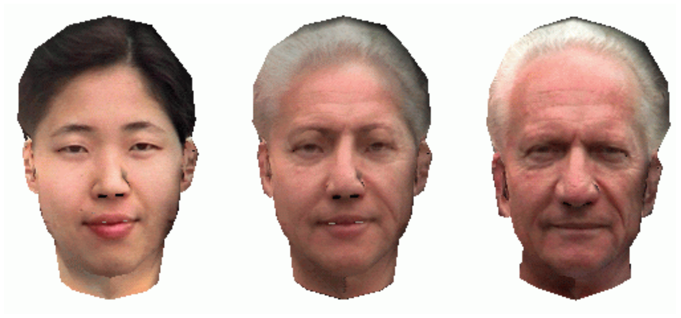


Figure 5-7: 90% of shape is from the Asian female and the image is mixed 50% for each of Caucasian male and Asian female.

The (generic) face model is an irregular structure defined as a polygonal mesh. As model fitting transforms the generic face without changing the underlying structure, the resulting new face can be animated. Since a reconstructed head can be animated in active way, which keeps the same topology and texture mapping during animation, it shows an easy solution for the experimentation of mixing people with different expressions. The two heads in Figure 5-8 have different expressions and head positions. The middle head with in-between shape and in-between texture shows in-between expression.

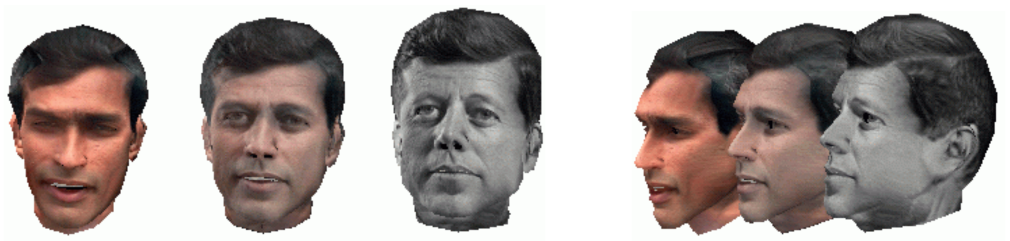


Figure 5-8: Morphing in 3D with different expressions on faces.

5.4 Dynamic 3D-morphing system – creating new heads

Techniques for generating virtual populations have been investigated by DeCarlo et al. [DeCarlo 98]**.** However, we are more interested in intuitively controlling the creation of people about the shape and skin color using 3D-morphing system. We have explained about morphing between two persons. To extend it into morphing among any given number of people is straightforward. Our approach for 3D morphing has the advantage of extending to several people in a natural way. Figure 5-9 illustrates a dynamic system for 3D morphing among a given number of people.

The interface in Figure 5-9 shows 3D morphing has two windows, the left one for controlling weights among several input faces and the upper right one containing the resulting new face. Two options are provided, one-time morphing and continuous morphing. One-time morphing provides a convenient way to select an input ratio in an *n*-polygon in the left window for *n* given virtual persons. The result appears immediately on the right window. For continuous morphing, the result varies according to the ratio points, which are chosen along a line or curve in the polygon.

We morph several images with given ratios using the information of texture coordinates and the triangulation information of the texture mapping. We first interpolate every 3D vertex on the several heads. Then to generate a new intermediate texture image, we morph triangle by triangle the entire 3D head. The parts of the image used for texture mapping are triangulated by projection of triangular faces of 3D heads since the generic head is a triangular mesh as seen in Figure 3-16(a). With this information for triangles, Barycentric coordinate interpolation can be employed for image morphing. First, the three vertices of each triangle are interpolated. Then pixel values inside triangles can be obtained from interpolation between corresponding pixels in several triangles with the same Barycentric coordinate. Since only a Barycentric calculation for texture image pixels is required, the calculation is very fast, which makes it possible to see real-time 3D morphing. There is also an option to have the shape and image variation separately to enable more various creations of new faces as shown in Figure 5-10.

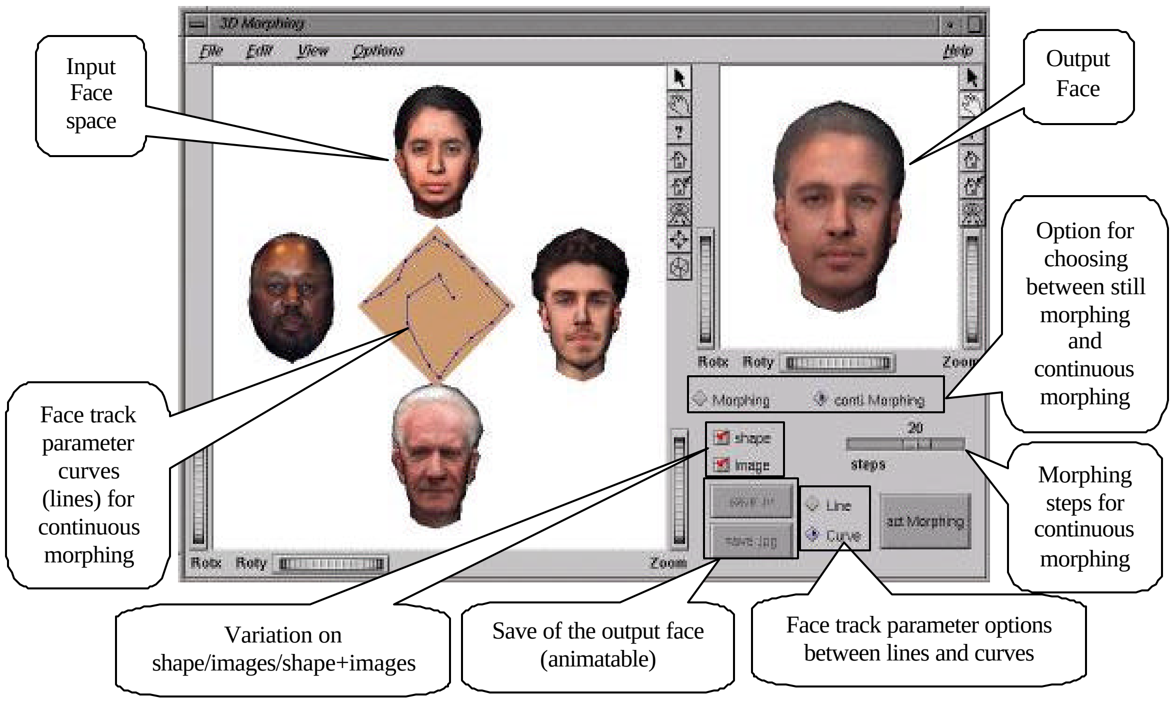

3D Morphing
File
Edit
View
Options
Help
Input Face space
Output Face
Option for choosing between still morphing and continuous morphing
Rotx
Roty
Zoom
Morphing
conti Morphing
shape
image
20
steps
Morphing steps for continuous morphing
save
save jpg
Line
Curve
act Morphing
Face track parameter curves (lines) for continuous morphing
Variation on shape/images/shape+images
Save of the output face (animatable)
Face track parameter options between lines and curves

Figure 5-9: A dynamic system for continuous 3D polymorphing or creation of new population from several people in intuitive way.

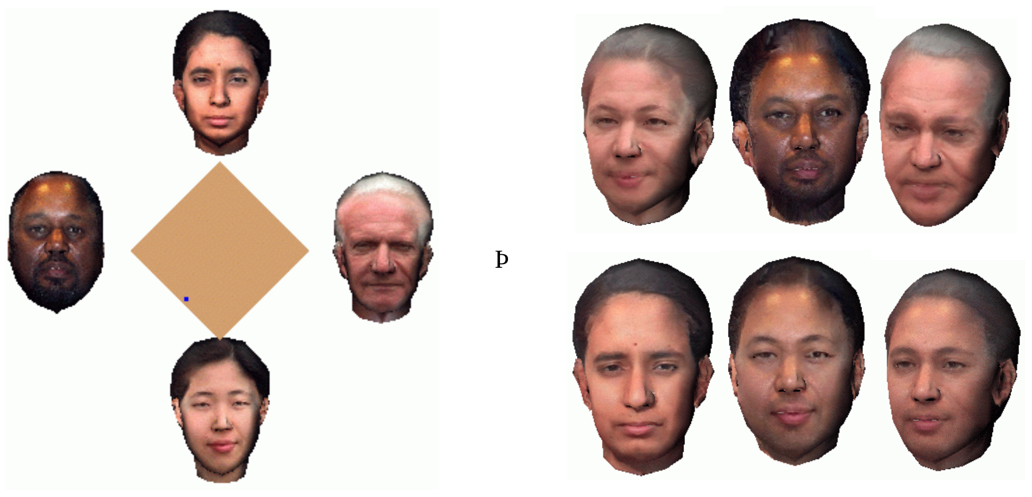
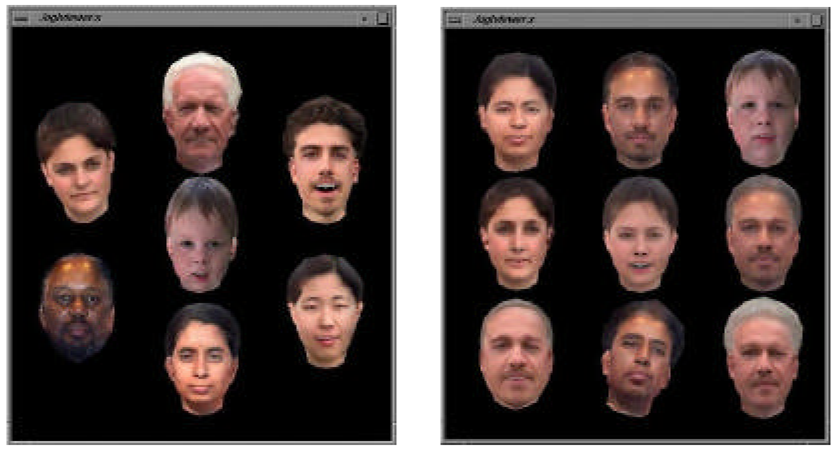


Figure 5-10: Examples of creation of new population. From four cloned faces, we create six new virtual faces, which do not have the counter parts in the real world.

When we create new virtual faces, the important point to consider is that all faces created keep functional structure for animation. Figure 5-11 (a) and (b) show snapshots of expressions on faces of cloned or created faces. The faces created using the morphing system are directly usable to do any animation.



(a) (b)

Figure 5-11: (a) a snapshot of animation of 7 virtual clones. (b) a snapshot of animation of created 9 heads from heads in (a). The animation shows the teeth and eye movements. The eyes and teeth are provided from the cloning methodology in Chapter 3.

5.4.1 Speed or flexibility?

We have shown the dynamic system for 3D morphing among faces with intuitive controlling in real-time visualization. It is rapid since the head models share the same structure. When two 3D objects have different structure such as shape morphing from a cup to a human face, the 3D morphing combining 2D images variation is more complicated [Kent 92][Lerios 95]. If two faces have different structure such as different number of points and different polygons, we have to specify feature correspondence. If we are given the same set of feature points for modeling purpose, we are still able to use the given feature correspondence. Then one of the simplest methods is using Delaunay triangles and Barycentric coordinates to find correspondence between surface points. However in this case the calculation cannot be performed in real-time since the Delaunay triangles are not yet obtainable in real-time.

Speed and flexibility for head structure are contrary terms.

5.5 Conclusion

We have introduced a method for real-time 3D face morphing system, which is used both for the real-time visualization of 3D morphing and also for the generation of large populations of photo-realistic faces, enabled for immediate real-time animation, from just a few pairs of orthogonal pictures. For the easy and fast generation of new virtual human, we first virtual-clone some people using only one generic model. Then we populate big number of virtual human based on them under an intuitive and rapid control with real-time visualization.

One key to our fas t 3D-face morphing technique is based on the efficient reconstruction of animation-ready individualized faces fitted with fully automatic seamless textures resulting in optimized triangulation. As we intend for speed purpose, all models generated in this system have the same topological structure, i.e. same triangulation, for the 3D shape and similar characteristics for the texture images, enabling real-time simulation of 3D metamorphosis among several virtual heads. Control of the mixing ratio for shape and image is intuitive and the rotation in any view allows the full appreciation of the 3D virtual heads. Various facial expressions can also be dynamically mixed in this system. The representation of a population as a probability distribution has great potential for allotting variation among face shapes into gender, age, race and residual components with eventual feedback to more realistic and efficient modeling.

Chapter 6 Adaptation to the Standard MPEG-4

6.1 What are MPEG standards?

MPEG (formally ISO/IEC JTC1/SC29/WG11) is a very large group of experts dedicated to the development of standards for digital audio and video. MPEG-4 is developed in response to the growing need for a coding method that can facilitate access to visual objects in natural and synthetic video and sound for various applications such as digital storage media, Internet, and various forms of wired or wireless communication. ISO/IEC JTC1/SC29/WG11 (Moving Pictures Expert Group - MPEG) works on this standard [SNHC1 98][SNHC2 98][Doenges 97]. In a world where audio-visual data is increasingly stored, transferred and manipulated digitally, MPEG-4 sets its objectives beyond "plain" compression. Instead of regarding video as a sequence of frames with fixed shape and size and with attached audio information, the video scene is regarded as a set of dynamic objects. Thus the background of the scene might be one object, a moving car another, the sound of the engine the third etc. The objects are spatially and temporally independent and therefore can be stored, transferred and manipulated independently. The composition of the final scene is done at the decoder, potentially allowing great manipulation freedom to the consumer of the data. MPEG-4 aims to enable integration of synthetic objects within the scene. It will provide support for 3D Graphics, synthetic sound, text to speech, as well as synthetic faces and bodies.

As the final approval is made on March 1999, Seoul, Korea, we may notice the fast convergence of many diverse domains: telecommunications, computers, movie industry, and universities. The MPEG-4 standard, initiated in 1995, aims at proposing tools for efficient coding of multimedia scenes. One important feature of MPEG-4 visual is the ability to code not just a rectangular array of pixels, but also objects in a scene. It proposes an efficient coding of diverse kind of data:

- Video Objects (extent of the notion of video defined by the previous MPEG standards by proposing the encoding of video objects having a shape),
- Face Objects (mainly devoted to teleconferencing),
- Mesh Objects (deformable nested polygons),
- StillTexture Objects (wavelet based coded large textures)

The second version adds Body Objects. For all parties involved in MPEG-4 the slogan "Develop once, play everywhere from anywhere" applies! A colleague, Igor Pandzic, in MIRALab, University of Geneva, has been from the beginning involved in the MPEG-4 facial parameters definition and MIRALab animation system has been used for early testing MPEG-4 committee.

This chapter demonstrates an experiment for a virtual cloning method and animation system, which is compatible with the MPEG-4 standard facial object specification. Our method uses orthogonal photos (frontal and side view) as input and reconstructs the 3D facial model. The method is based on extracting MPEG-4 face definition parameters (FDP) from photos, which initializes a custom face in a more capable interface, and deforming a generic model. Texture mapping is employed using an image composed of the two orthogonal images, which is done completely automatically. A reconstructed head can be animated immediately inside our animation system, which is adapted to the MPEG-4 standard specification of face animation parameters (FAP). The resulting head is exported into the format of surface with FDP and FAP point indication, which is used for other applications by colleagues in MIRALab, University of Geneva such as talking heads or virtual human director (VHD) system.

6.2 Face definition and animation in MPEG-4

The Face and Body animation Ad Hoc Group (FBA) has defined in detail the parameters for both the definition and animation of human faces and bodies. Definition parameters allow a detailed definition of body/face shape, size and texture. Animation parameters allow the definition of facial expressions and body postures. These parameters are designed to cover all natural possible expressions and postures, as well as exaggerated expressions and motions to some extent (e.g. for cartoon characters). The animation parameters are precisely defined in order to allow an accurate implementation on any facial/body model. Here we will mostly discuss facial definitions and animations based on a set of feature points located at morphological places on the face. The two following sections will shortly describe the *Face Animation Parameters* (FAP) and *the Face Definition Parameters* (FDP). Here, we use mainly FDP for face cloning and both FDP and FAP points are exported as output of the system.

6.2.1 Facial Animation Parameter set.

The FAP are encoded for low-bandwidth transmission in broadcast (one-to-many) or dedicated interactive (point-to-point) communications. FAPs manipulate key feature control points on a mesh model of the face to produce animated visemes (visual counterpart of phonemes) for the mouth (lips, tongue, teeth), as well as animation of the head and facial features like the eyes or eyebrows.

All the FAP parameters involving translational movement are expressed in terms of Facial Animation Parameter Units (FAPU). These units are defined in order to allow the interpretation of FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to fractions of distances between some essential facial features (e.g. eye distance) (Figure 6-1). The fractional units used are chosen to allow enough accuracy.

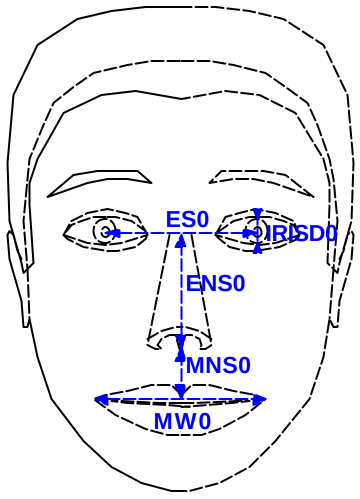


Figure 6-1: The Facial Animation Parameter Units

6.2.2 Facial Definition Parameter set.

An MPEG-4 decoder supporting the Facial Animation must have a generic facial model capable of interpreting FAPs. This insures that it can reproduce facial expressions and speech pronunciation. When it is desired to modify the shape and appearance of the face and make it look like a particular person/character, FDPs are necessary. The FDPs are used to personalize the generic face model to a particular face. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs.

The FDP fields are as follows:

- **FeaturePointsCoord –** it specifies feature points for the calibration of the proprietary face.

- **TextureCoords –** it specifies texture coordinates for the feature points.
- **TextureType –** it contains a hint to the decoder on the type of texture image, in order to allow better interpolation of texture coordinates for the vertices that are not feature points.
- **FaceDefTables -** this field describes the behavior of FAPs for the face in the **FaceSceneGraph**.
- **FaceSceneGraph -** this node can be used as a container of the texture image as explained above or the grouping node for the face model rendered in the compositor and therefore it has to contain the face model.

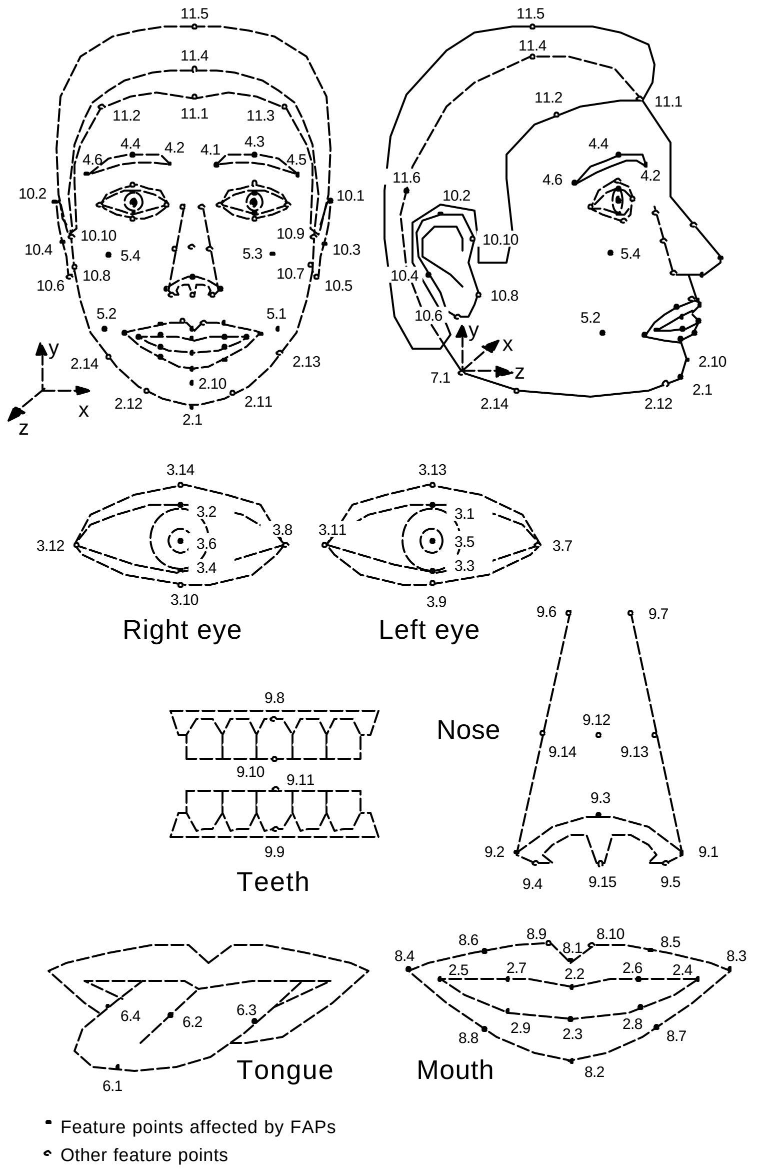
We do not deal with **FaceDefTables** in this thesis.

6.3 FDP cloning from two orthogonal photographs

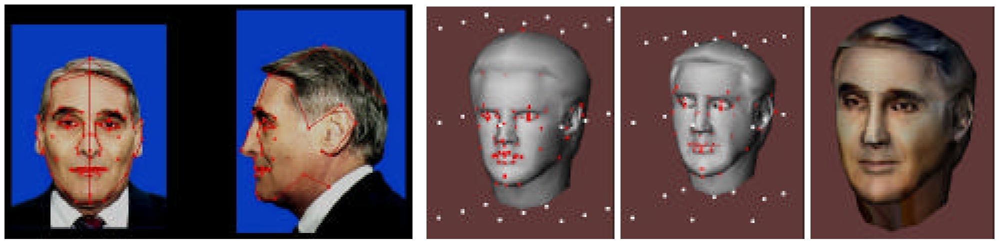
There was an approach to make an individualized head by modifying a generic model using MPEG-4 parameters, but it used range data as input, such as laser scanner [Escher 98][Escher 97], instead of easy-and-cheap-device like a camera. In addition it assumes that the FDP points are provided.

Our methods use orthogonal photos (frontal and side views) as input to reconstruct 3D facial models. The difference between the face cloning in Chapter 3 and this chapter is feature points definition. The features used in Chapter 3 are our own definition, and here we will use FDPs. The method is based on extracting FDPs on photos, which initializes a custom face in a user-friendly interface, and deforming a generic model. Texture mapping is employed using a composed image from the two images in a fully automatic way.

FDPs contain features of eyes, eyelids, eyebrow, lips, nose, chin, cheek, tongue, head rotation point, teeth, ear and some of face and hair outlines as shown in Figure 6-2(a). The FDP points on a pair of orthogonal photos are shown in Figure 6-2(b), which has some line segment to help the user to detect FDPs. The model in Figure 6-2(c) shows a previously constructed, animation-ready generic model, which is transformed to an individualized head, based on the FDPs shown in Figure 6-2(d). The two heads in Figure 6-2(c) and Figure 6-2(d) have red points on them, which are FDPs in 3D composed of two FDP sets, red points in Figure 6-2(b). The final cloned model shown in Figure 6-2(e) is ready to be animated immediately.



(a)



(b) (c) (d) (e)

Figure 6-2: (a) FDP feature point set and feature points affected by FAPs. (b) FDPs on input photographs. (c) A generic model (d) An individualized shape (e) A final virtual human for animation.

6.3.1 Some problems with FDP extraction on photos.

There are 84 parameters in FDPs. Some of them are not visible on photos (for example, 4 points on teeth, 4 points on tongue). In our animated model, since points 3.6 and 3.5 can be extracted approximately from 3.12, 3.8, 3.11 and 3.7 (See Figure 6-2(a)), we do not use them. There are also some difficulties to identify features on photos such as 11.4 and 11.6. Although they are not visible on photos, we need some points to define hair shape. So we use 11.4 and 11.6 as boundary of head instead of 11.5. Finally we make use of only 84 – 4 – 4 – 2 – 1 = 73 points.

Another difficulty also comes from locating some points when they are too near each other, such as (3.4, 3.10), (3.3, 3.9), (2.5, 8.4), and (2.4, 8.3). Points (2.7, 2.9), (2.2, 2.3), and (2.6, 2.8) are sets, whose positions are the same when a person has a closed mouth. In addition, some important points to characterize a person are not defined in FDPs, which causes some problem later. There are too few points on nose, especially on side profiles. There is no parameter defined on central line on nose between two eyes and between two eyebrows, which limits the generation of various shapes on side profiles. There are not enough points on eyes and nosewings and it results in some texture fitting error later. If there are some more points on outlines of a face and hair, the characterizing could have given better results.

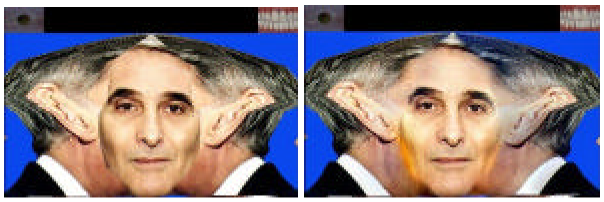
6.3.2 Head modification.

We deform a generic model using FDP points extracted from photos as control points. The deformation of a surface can be seen as an interpolation of the displacements of the control points. DFFD [Moccozet 97] process gets new geometrical coordinates for a modification of the generic head on which are situated the newly detected feature points. Refer Section 3.2 for more detail.

6.3.3 Automatic Texture Mapping

Texture mapping serves not only to disguise the roughness of shape matching determined by feature point matching, but also to imbue the face with more realistic complexion and tint. We use information from FDP points detected to generate texture automatically, based on the two views. We then obtain appropriate texture coordinates for every point on the head using the same image transformation.

The main criterion is to obtain the highest resolution possible for the most detailed portions. We first connect two photographs along predefined feature lines using geometrical deformations and, to avoid visible boundary effects, a multiresolution technique is used. The feature lines are defined as a piecewise lines of 8 points such as for example (11.4, 11.3, 4.5, 10.9, 10.7, 2.13, 2.11, 2.1) on the left side, which follows from a point on top skull, a point between hair and forehead, an outer point of left eyebrow, an upper point between left ear and face, a lower point between left ear and face, a left point on jaw, a left point on chin, a middle point on bottom of chin. The counter feature piecewise line of the right side is also defined in similar way. We keep the frontal face and deform the side face to match the feature line on the side face to the feature line on the frontal face. The left view is a flip image of right view. Therefore, we use the side photo for both the right and the left views. Figure 6-3(a) shows the result of geometric deformation.



(a) (b)

Figure 6-3: (a) A texture integrated after geometric deformation. (b) A texture with multiresolution technique.

To correct visible boundary problem among images merged, we apply multiresolution techniques. As in Figure 6-3(a), skin colors are not continuous when the multiresolution technique is not applied. The image in Figure 6-3(b) shows the results with the technique, which has smoothed the connection between the images without visible boundaries. Then the texture coordinates are found. More detail is found in Section 3.3.

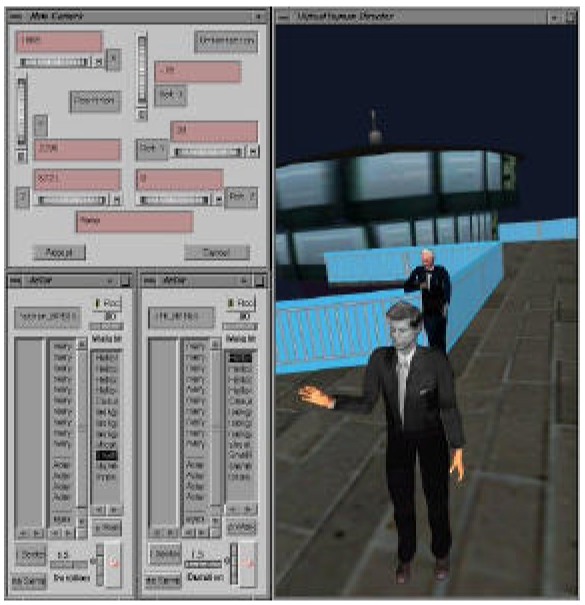
The final textured head is shown in Figure 6-2(e). The eye and teeth fitting process is done with predefined coordinates and transformations related to the texture image size. After doing this once for a generic model, it is fully automated.

As mentioned in Section 6.3.1, there are some difficulties in obtaining satisfactory results using MPEG-4 FDPs completely automatically, especially for texture fitting between the hair and the face region and nosewings due to the lack of feature points. We provide a user-friendly interface to accelerate the results, whose diagram is shown in *Figure 8-3*. Normally the 2D-feature detection manager is processed in an interactive way and then the 3D-modification manager and 2D-texture manager follow in a fully automatic way. However, we introduce a visual feedback method to correct some minor errors.

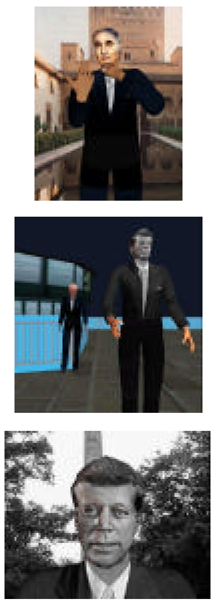
6.4 Results

The resulting head produces two kinds of output. The first is for the head surface information such as the 3D geometrical surface data, texture data. The second is the animation information in either regional information or in FDP and FAP points' information, which is used by colleagues in MIRALab, University of Geneva, to animate the faces and bodies. One of the examples for rendering systems in MIRALab is VHD [Sannier 99], which provides a range of virtual human animation with talking capacity and interaction capabilities integrated into one single environment.

To animate a virtual head in VHD, the only sets of data needed are the topology and the facial-animation feature points. The heads created here provide this information, so the heads are directly usable in VHD. Figure 6-4 shows the various snapshots of these three virtual clones acting in 3D virtual world. In order to animate the virtual humans using the MPEG-4 parameters, we provide a set of predefined animations to each virtual human.



(a) A user interface VHD



(b) Some snapshots

Figure 6-4: Real-time animated virtual humans in VHD.

6.5 Conclusion

We have applied our face cloning method from photographs to be compatible with the newly standardized MPEG-4 specification of the facial object in object based coding syntax. This application is based on extracting MPEG-4 *face definition parameters* (FDP), of animation-ready individualized faces fitted with seamless textures. Some problems for FDPs are discussed. A reconstructed head can be animated immediately inside MIRALab's animation systems, which are adapted to MPEG-4 standard specification of *face animation parameters* (FAP).

As the goal of MPEG-4 standard is the coding definition for communication on the network, we expect applications in the future when everybody follows the standard for their virtual humans.

- Virtual Reality experiences on the Web

  The high-compression capability of MPEG-4 and the ability to download portions of the world only when they are actually needed makes MPEG-4 a very interesting tool to enjoy virtual worlds with streaming content at the typical bitrates of the Web today;

- Virtual sites on the network

  The synthetic virtual human of MPEG-4 makes it possible to create virtual spaces for new exciting experiences on the network.

The real-time facial communication between different sites connected by only network can be performed. Two different sites will have the same generic model and cloning system. The calculation to make each clone for each site will be done only once and then the texture image and coordinates and head data will be transferred to other sites. Then the same data in several sites will allow only a small number of parameters will be transmitted via network in real-time, which makes possible the real-time application and the same visualization in several sites.

Chapter 7 Body Cloning

In this chapter, we present a full body cloning methodology. This system utilizes photos taken from the frontal, side and back of a person in any given imaging environment without requiring a special background or a controlled illuminating condition. A seamless generic body specified in the VRML H-Anim 1.1 format is used to generate an individualized virtual human. The body cloning component has three steps: (i) feature points specification, which enables automatic silhouette detection in an arbitrary background (ii) two-stage body modification by using feature points and body silhouette respectively (iii) texture mapping using two views. The resulting body is integrated with the result from the face cloning system. The final integrated human body with face together has photo-realistic animatable face, hands, feet and body. The VRML Humanoid Animation Working Group (H-Anim) exists for the main purpose of creating a standard VRML representation for humanoids. We produce the final bodies in VRML H-Anim 1.1 format [http:H-Anim] and they can be visualized by any web browsers, such as Netscape and animated by a JAVA program if available. The outline of the algorithm for body is shown in Figure 7-1.

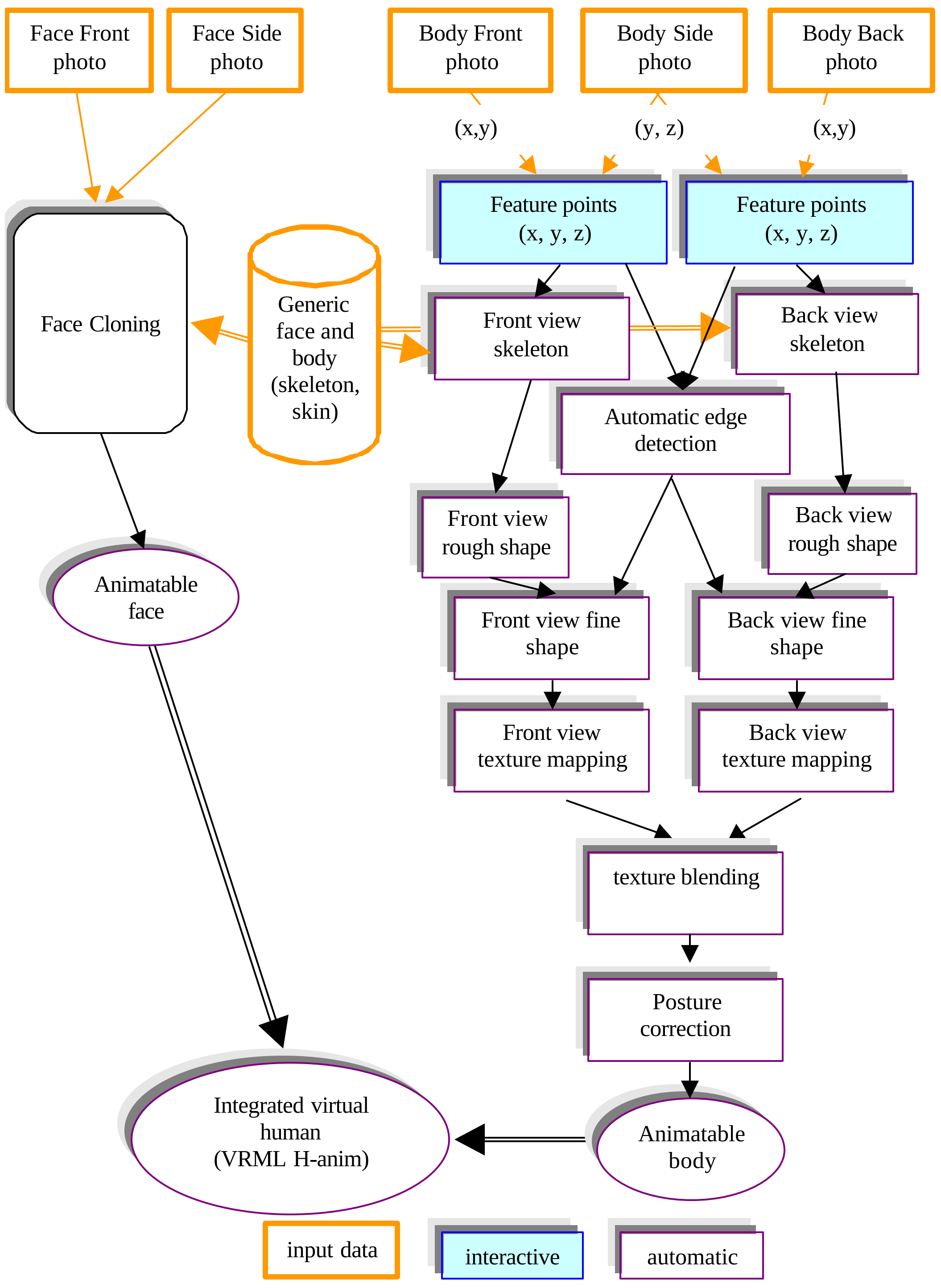


Figure 7-1: Overview of body cloning.

7.1 H-Anim body

As the 3D Internet continues to grow, there will be an increasing need to represent human beings in online virtual environments. Achieving that goal will require the creation of libraries of interchangeable humanoids, as well as authoring tools that make it easy to create new humanoids and animate them in various ways. H-Anim [http:H-Anim] specifies a standard way of representing humanoids in VRML97. This standard will allow humanoids created using authoring tools from one vendor to be animated using tools from another.

H-Anim humanoids can be animated using keyframes, inverse kinematics, performance animation systems and other techniques. The goals are as follows:

- Compatibility: Humanoids should work in any VRML97 compliant browser.
- Flexibility: No assumptions are made about the types of applications that will use humanoids. It allows direct access to the joints of the humanoid's body and the vertices of the individual body segments.
- Simplicity: When in doubt, we leave it out. The specification can always be extended later.

7.1.1 Nodes

The human body consists of a number of Segments (such as the forearm, hand and foot) which are connected to each other by Joints (such as the elbow, wrist and ankle). In order for an application to animate a humanoid, it needs to obtain access to the joints and alter the joint angles. The application may also need to retrieve information about such things as joint limits and segment masses. A mesh of polygons will typically define each segment of the body, and an application may need to alter the locations of the vertices in that mesh. The application may also need to obtain information about which vertices should be treated as a group for the purpose of deformation.

In order to simplify the creation of humanoids, several new node types are introduced. Each node is defined by a PROTO. The basic implementation of all the nodes is very straightforward, yet each provides enough flexibility to allow more advanced techniques to be used. See *Figure 7-2* to see the node hierarchy.

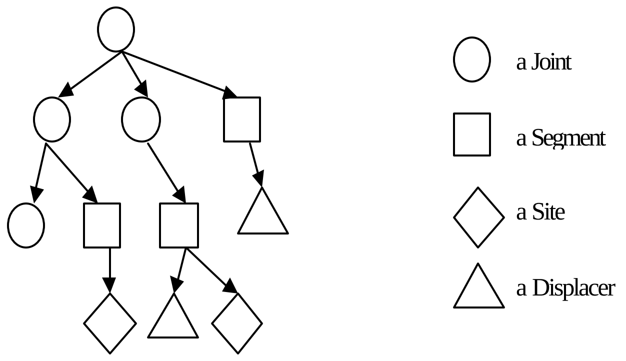


Figure 7-2: Node Hierarchy

7.1.1.1 Joint

A Joint has a name, a center of rotation, and a rotation relative to its parent Joint. Joints also define a LOCAL frame in which upper and lower rotational limits can be applied. Joint includes a stiffness value to weight the influence of the joint for a potential IK engine. An H-Anim file contains a set of Joint nodes that are arranged to form a hierarchy.

7.1.1.2 Segment

Segments define geometry. A polygonal mesh typically defines the shape of a segment. A segment also includes mass, center of mass, and moments of inertia (though in practice, these are rarely used).

7.1.1.3 Site

It specifies a location and orientation of a point relative to a Segment. It is useful for attaching accessories (glasses, hats, etc) and also as end effector locations for Inverse Kinematics (IK).

7.1.1.4 Displacer

It specifies how the geometry of a Segment can be deformed. It assumes a polygonal mesh for the geometry and stores references to vertices, along with displacement vectors. It is useful for morph targets and muscle actions

7.1.2 Coordinate system

The humanoid should be modeled in a standing position as shown in *Figure 7-3*.

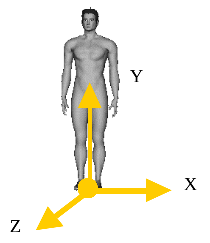
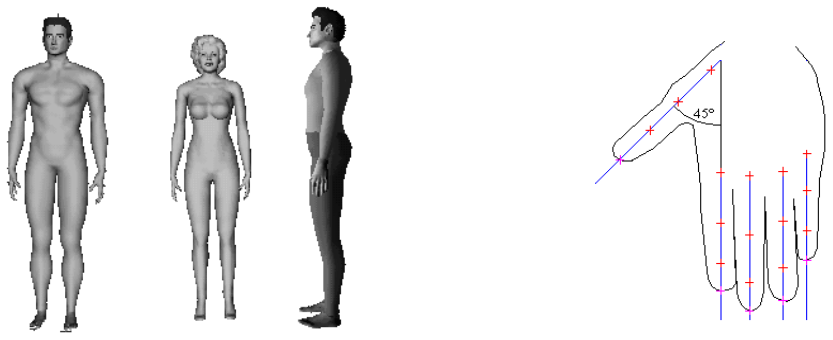
• Y up • X left • Z front • The origin (0,0,0) of the coordinate system "should" be located at ground level between the feet. The bottom of the feet "should" be at Y=0. • The orientation of the joints frames and the sites frames in the default posture is the same as the one of the HUMANOID coordinate system	

Figure 7-3: A HUMANOID Coordinate system

The humanoid should be built with actual human size ranges in mind. All dimensions are in meters. A typical human is roughly 1.75 meters tall. The default position of the humanoid body is shown in Figure 7-4 (a). The feet should be flat on the ground, spaced apart about the same distance as the width of the hips. The arms should be straight and parallel to the sides of the body with the palms of the hands facing inwards towards the thighs. The hands should be flat as shown in *Figure 7-4* (b). Note that the coordinate system for each joint in the thumb is still oriented to align with that of the overall humanoid. The face should be modeled with the eyebrows at rest, the mouth closed and the eyes wide open.



(a) (b)

Figure 7-4: Default humanoid. Illustration courtesy of the SNHC group, where the basic mesh and software was provided by MIRALab-University of Geneva and LIG-EPFL. (a) Default body posture. Note that the thumb is shown in a slightly different position than required by this specification. (b) Default hand.

In this position, all the joint angles should be zero. In other words, all the rotation fields in all the Joint nodes should be left at their default value of (0, 0, 1, 0). In addition, the translation fields should be left at their default value of (0, 0, 0) and the scale factors should be left at their default value of (1, 1, 1). The only field which should have a non-default value is center, which is used to specify the point around which the joint (and its attached children and body segment if any) will rotate. Sending the default values for translation, rotation and scaling to all the Joints in the body must return the body to the neutral position described above.

It is suggested, but not required, that all the body segments should be built in place. That is, they should require no translation, rotation, or scaling to be connected with their neighbors. For example, the hand should be built so that it is in the correct position in relation to the forearm. The forearm should be built so that it is in the correct position in relation to the upper arm, and so on. All the body's coordinates will share a common origin, which is that of the humanoid itself. If this proves difficult for an authoring tool to implement, it is acceptable to use a Transform node inside each Segment, or even several Transforms, in order to position the geometry for that segment correctly.

Note that the coordinate system for each Joint is oriented to align with that of the overall humanoid. Only humanoids sharing the same segment lengths can share animations successfully. Otherwise, collision with the environment, self-collisions or missed targets are "genetically programmed" to happen. In case two humanoids have the same segment proportions and the same skin proportions, there is no risk of self-collisions or missed target.

7.1.3 The Joint Hierarchy

The body is typically built as a series of nested Joints, each of which may have a Segment associated with it. Figure 7-5 shows the hierarchy of the full set of Joints and Segments, which are our interests for our body cloning.

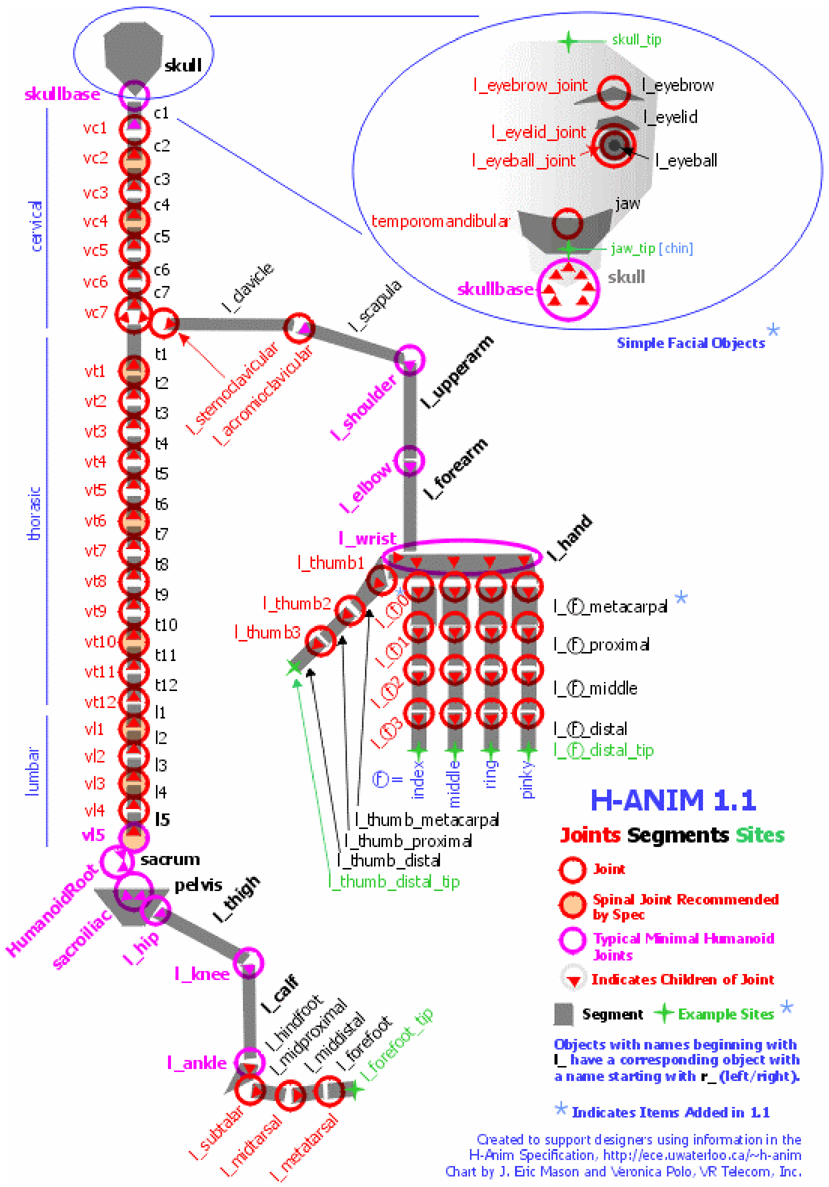


Figure 7-5: The standard Joints and Segments for a version 1.1 Humanoid. Illustration courtesy of the SNHC group [SNHC1 98][SNHC2 98].

7.2 The generic body – continuous mesh humanoids

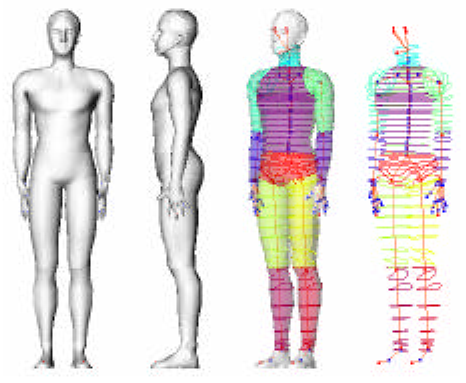


Figure 7-6: The seamless generic body with H-Anim 1.1 skeleton and several skin parts

The generic body is in MPEG-4 compatible H-Anim 1.1 formats [http:H-Anim]. The skeleton and several skin parts displayed with several colors are shown in the right side of *Figure 7-6*, where the skin parts are smoothly connected. It has the Level of articulation three with full H-Anim hierarchy with 94 skeleton joints and 12 skin parts including *head*, hands and feet. An option in H-Anim 1.1 format definition allows to select a subset of 94 skeleton joints. The first version of generic body we are using is collected from a public domain [Babski 99] and modified for our usage.

Each skin part is saved in local coordinates and are related to a skeleton joint as a segment node as shown in *Figure 7-7*, where the skeleton location is indicated by arrows and related skin parts are written inside (). The right side skin parts are not shown, but it is easy to guess from the left side. The skin parts have the corresponding skeleton joints, which transform the local coordinates of the skin part ***i*** to global coordinate by 4x4 matrix $\boldsymbol{M_i}$.

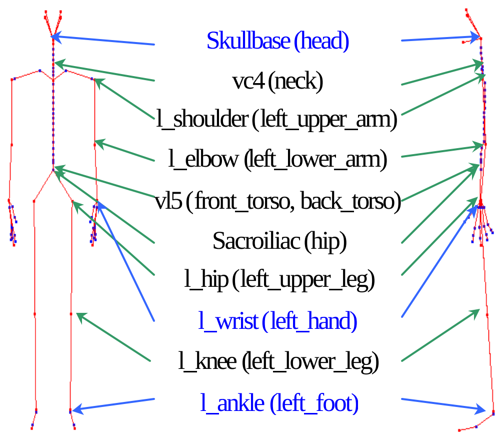


Figure 7-7: The H-Anim joints related to skin parts

The 12 skin parts beside head, hands and feet are designed to have real-time deformation for animation [Babski 99]. The good points of these skin parts are that first they compose a seamless skin envelope, so that the texture mapping will be smoothly connected while animation deforms both skeleton and skin. The second point is the point structure of each skin part, which is designed in a special way such that each skin part is composed of several slices and each slice in a skin part has the same number of points. For example the '*hip*' has 6 slices and 26 points on each slice (the total point number on '*hip*' is 6x26 = 156). We call it as the *grid structure*, which makes the *piecewise affine transformation* possible in later sections.

The *head*, *hands* and *feet* are separate objects with different structures from the *grid structure*. They are also animated in different ways.

7.2.1 Data structure

When we load a generic body, there is information to be included as input.

1. Default Hierarchy of joints. We use H-Anim hierarchy.
2. Default location of joints.
3. Default skin parts coordinates in local coordinates as mentioned above.
4. Default feature points for three images. See Figure 7-8.
5. Define movable joints as a subset of skeleton joints. Other skeleton joints are moved automatically by affine transformation defined by the hierarchy of joints. One example of defined *movable* joints are as follows.
    - HumanoidRoot
    - vt1
    - vc4
    - temporomandibular
    - skullbase
    - l_hip, r_hip
    - l_knee, r_knee
    - l_ankle, r_ankle
    - l_metatarsal, r_ metatarsal
    - l_shoulder, r_ shoulder
    - l_elbow, r_ elbow
    - l_wrist, r_ wrist
    - l_middle3, r_ middle3

    There are end joints on feet and hands such as *l_metatarsal, r_metatarsal, l_middle3, r_middle3*. They are used for global translation and scaling for hands and feet.
6. Correspondence of each skin part to a skeleton joint
7. The adjacency of neighboring skin parts such as, for instance,

    '*left_upper_leg*' has '*hip*' as 1 pre_skin_part (adjacent skin of parents joints)

    '*hip*' has '*front_torso*', '*back_torso*' as 2 pre_skin_part

With the information given, the following items are constructed when we load the generic body.

1. Database of coordinate connection between adjacent skin parts of the generic body including head, hands and feet part.
2. Find the skeleton joint connected to pre_skin_part for each skin part

Then the generic body has properly connected skin envelope and can keep the connection always.

7.3 Feature-based modification

7.3.1 Taking photographs and initialization

We focus on the simplest environment to take photos with only one camera. We take three photos, from the frontal, the side and back. In this case, the frontal and back views are not exact reflections of each other since we asked the person to rotate for the back view after taking the frontal view.

We input the height (a unit - the meter), of the person and the image body heights are checked on the three images for normalization.

Figure 7-8 shows normalized images. Since we use arbitrary background to take photos and the size of the person is not fixed, we use interactive feature point localization on images as shown as small points in Figure 7-8. The names of the features are given in *Figure A-4*. This simple feature point localization is used for skeleton modification, rough skin deformation and fully automatic edge detection in the next sections.

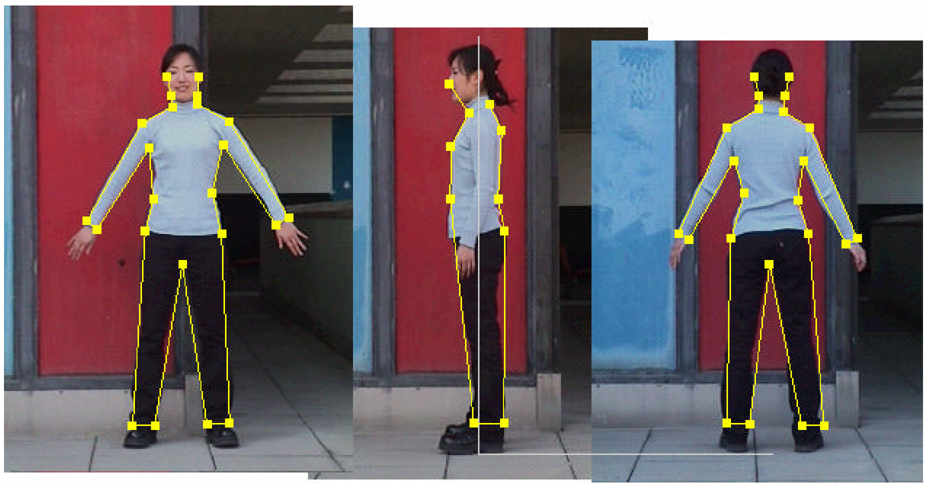


Figure 7-8: Feature points on three images.

7.3.2 Skeleton modification

When we have feature points for the skin envelope (even though the person put on clothes, we assume that the clothes outlines are close to the skin outlines), we can have an estimation of the skeleton. For example, for the *r_elbow* must be located around middle position between the right end shoulder point and outer end point of the right wrist. Here we applied *affine transformations* and Barycentric interpolation to find the typical skeleton joints from feature points. Since there are 94 skeleton joints, we make a subset of joints as *movable* joints as mentioned in Section 7.2.1, which are modified by feature points while others are modified by *piecewise affine transformations* defined by the *movable* joints and the skeleton hierarchy.

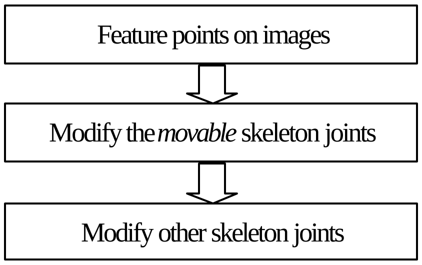

Feature points on images
Modify the *movable* skeleton joints
Modify other skeleton joints

Figure 7-9: From feature points on images to skeleton modification

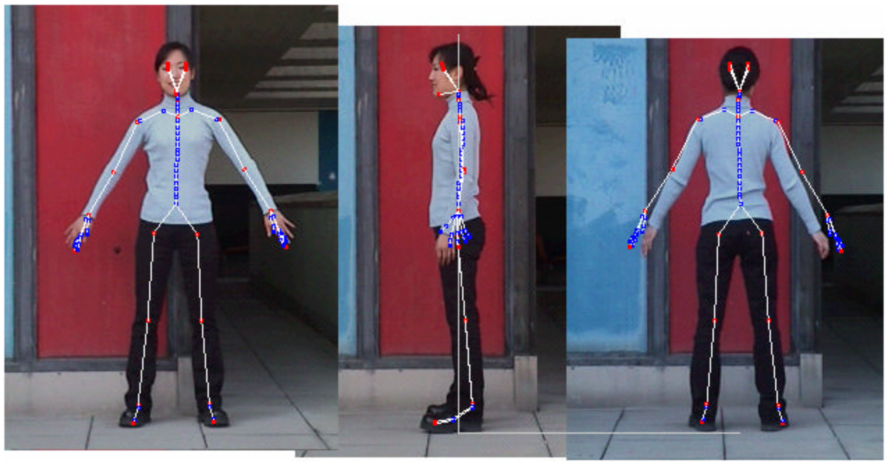


Figure 7-10: Automatic Skeleton fitting with feature points.

The skeleton joints of the *head*, *hands* and *feet* are simply scaled and translated with the factor between the generic body's skeleton and the person's skeleton, for example *vt1* and *HumanoidRoot* or *l_elbow* and *l_wrist* can be used to find the transformation. Figure 7-10 shows the skeleton modified by the frontal and the side views. Since the frontal and back views do not have the exactly same pose, the skeleton modified by the frontal view does not match on the back view.

7.3.3 Rough skin modification

As we mentioned in Section 7.1, each skin part ***i*** is connected to a skeleton joint by a matrix $\boldsymbol{M_i}$ to be in global coordinates. We update the matrix by scaling, translation and rotation defined by the corresponding skeleton joint and the child skeleton joints. As we see in Figure 7-11, adjacent skin parts are not guaranteed to be continuous. The shoulder parts are overlapping with torso parts.

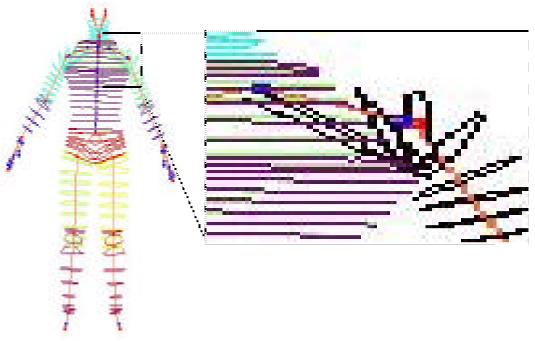


Figure 7-11: updated skin parts by skeleton joints and related matrix.

A simple linear transformation by a matrix $\boldsymbol{M_I}$ does not solve the overlapping or separating problems. So we need a special transformation to solve the overlapping (or separating) problem and integrate the skin parts properly with an approximated shape to the person.

Here we define a freeform deformation to make a rough matched continuous body with feature points information. The control points are placed at certain required positions to represent the shape characteristics. Hence the skin model can be deformed by moving these control points, which is designed such as they have corresponding points on the frontal (or back) and the side view images, or the locations are found with certain relations. For example the boundary between the '*front_torso*' and '*right_upper_arm*' can be found from the feature points on the bottom-right corner point of the *neck* and on the right end shoulder point on the images. Furthermore, several control points are located at the boundaries between two parts, so that surface continuity is preserved when the posture of the generic body is changed.

These control points are used for the *piecewise affine transformation*. We apply separate deformations for each skin part. We show some examples of the control points in Figure 7-12.

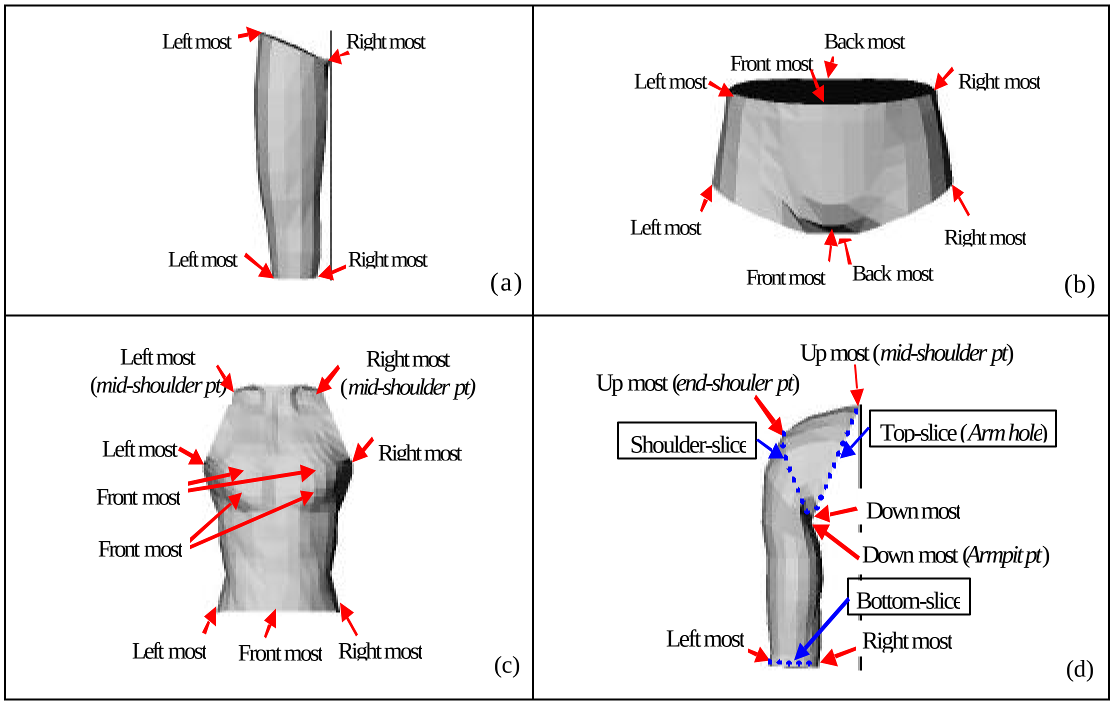


Figure 7-12: The control points on right_upper_leg, hip, front_torso and right_upper_arm parts.

We apply first *piecewise affine transformation* where each *affine transformation* is defined on each segment on a curve. Thanks to the *grid structure* of skin parts, we apply the *piecewise affine transformation* horizontally first on slices that have control points and then vertically on points which have the same index on each slice. For example for the '*hip*' part, we define the first *affine transformation* with the left-most point on the top-slice and the front-most point on the top-slice and corresponding control points on images. Also we apply the *affine transformation* on points between the left-most point and the front-most point. Then we define the next *affine transformation* for front-most point and right-most point and apply it to points in between. We define and apply the third and forth *affine transformations* on the other two pieces on the top-slice. Then we get a new top-slice from the generic body's slice, which fits to the person's top-slice now. Then we apply the similar process for the bottom-slice that has control points too. After getting the new bottom-slice, we define an *affine transformation* in the vertical direction using two points with the same index on the top-slice and the bottom-slice. Figure 7-13 shows the steps for the '*hip*' part. For the '**_upper_arm*' parts, we apply *piecewise affine transformation* for three slices such as the top-slice, the shoulder-slice, and the bottom-slice. For '*front_torso*', we have four slices with control points such as top-slice, upper-breast-slice, lower-breast-slice, bottom-slice, as we can see in Figure 7-12 (c).

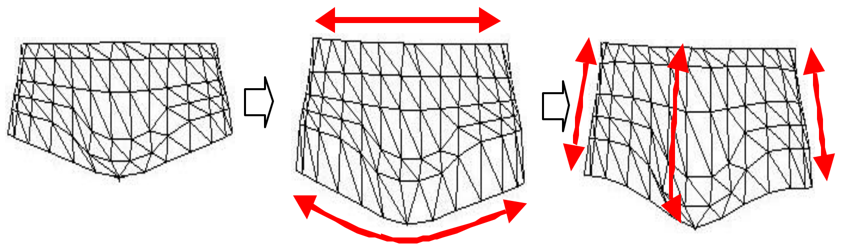


Figure 7-13: The process of the affine transformations for the hip.

After applying the *piecewise affine transformation*, we make one more process to stick adjacent skin parts properly. When the generic body is loaded, the connection database is built automatically and it is used for the skin parts connection. The database contains which point on which skin part is connected to which point on which skin. The *neck* is not deformed using the feature points from images since it is too small on the images. The *piecewise affine transformation*s are defined from adjacent points on the *'head'* with the top-slice on *'neck'* and from adjacent points on the *'front_torso'* and *'back_torso'* with the bottom-slice on *'neck'*.

It is a rough deformation since we deform the generic body only with a few feature points. So it does not match exactly for the outfit on the images. However it serves as an initialization of skin for the shape and catches a proper functional approximation for animation such as having a proper skeleton and skin parts localization.

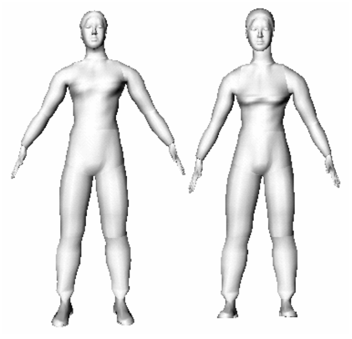


Figure 7-14: two bodies (front+side/back+side) with rough skin deformation

Since the frontal and back views were taken in different positions, we produce two bodies as seen in Figure 7-14. The left one obtained from the frontal and the side images and the right body from the back and the side images. The reason to produce two bodies is because of two sources for texture mapping, which will be described later.

7.4 Edge-based modification

A colleague, Jin Gu, in MIRALab is the main implementer and the writer [EG 00] on this section. The contribution on this section of the thesis author is the proposal of the idea to use feature-driven edge detection and body modification on the arbitrary background and the discussion of the implementation especially at the beginning of the body-cloning project. The thesis author takes and shortens this part to explain the overall steps for the body cloning. This section is for the fine modification of the body. The silhouettes are extracted from the images with feature point information and they are used for shape modification.

7.4.1 Heuristic based silhouette extraction with any background

There are plenty of literatures available about boundary extraction or edge linking [Ballard 96] [Lai 95]. It can be treated as a graph-searching problem, as an optimization problem, or as an energy minimization and regularization procedure. However, these algorithms are usually inefficient due to the need for backtracking or exhaustive search. Or the algorithms need time to reach convergence or stable result, such as the snake algorithm. We design a simple algorithm by making use of the feature points on the body. In this section these feature points serve as the heuristics for the body silhouette extraction.

First, Canny edge detector is applied to every image. Then a coloring-like linking algorithm is used to link the edgels (edge pixels) into connected segments. Due to possible noise caused by the background, the edgels generated by the background sometimes are connected to the body edgels. To avoid this potential wrong connection from occurring, we split the segments into short line segments.

To make the following discussion easier, let us call the line segments formed by consecutive feature points as *feature segments*, while the short line segments formed by linking edge pixels, as *edge segments*. Each *feature segment* indicates the vicinity and approximate direction of the boundary to be found. From the Canny edge detector and linking step, we obtain the *edge segments* generated by the object as well as by the background. We first throw away those lying outside the vicinity of any *feature segments*. The goal now is, for each *feature segment*, to find a path that is formed by an ordered set of *edge segments* within its vicinity.

To link the *edge segments* into meaningful boundaries, we first look for the admissible connection for each *edge segment*. We define a connection between *edge segment* $\mathbf{S_1}$= P_{12}-P_{11} and $\mathbf{S_2}$= P_{22}-P_{21} as *admissible* if (see Figure 7-15):

$$\boldsymbol{S_1} \cdot \boldsymbol{S_2} >= 0.0,\ a_1 < T_{sm} \text{ and } a_2 < T_{sm}$$

where P_{11}, P_{12}, P_{21} and P_{22} are the ends of the two segments under consideration; α_1 is the angle between $\mathbf{S_1}$ and the potential connecting segment $\mathbf{C_{12}}$ =P_{21}-P_{12}, α_2 is the angle between $\mathbf{C_{12}}$ and $\mathbf{S_2}$, T_{sm} is the maximum angle allowed for the connection. Next in order to select the most desired connection, we define a function G^L to evaluate the "goodness" of the potential connection (see Figure 7-15):

$$G^L(\boldsymbol{S_1},\boldsymbol{S_2}) = cos(a_1)*cos(a_2)*(1+cos(a_1)*cos(a_2)*log(M_2))/(w*D_1+(1-w)*D_2)$$

where D_1 is the length of the $\mathbf{C_{12}}$ and D_2 is the distance from the P_{22} to the *feature segment* **L**, M_2 is the edge magnitude of edge segment S_2, w is the weight of D_1 taken as a constant in our experiments. The rationale behind this definition is that we always favor a connection that contributes to a smooth path running along **L,** formed by segments with strong edge magnitude.

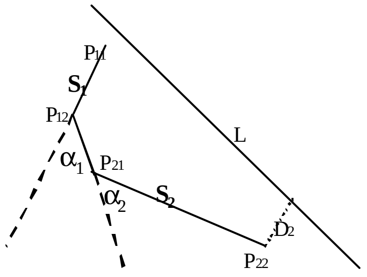


Figure 7-15: Link two edge segments

Thus based on G^L, we find, for the two ends of each *edge segment*, its best connection that maximizes G^L among all of its neighboring *edge segments*. Now we are ready to build the path. Starting from each *edge segment*, we connect it to its two best connections computed by G^L to form a partial path. For the *edge segments* sitting on the *head* and tail of this partial path, we connect their open ends to their respective best connections. This procedure is repeated until there is no further connection possible. So a path **P** consists of a sequence of *edge segments* $\mathbf{S_i}$, i=1,2,..., N. Now we need another evaluation function to assess the "goodness" of a path. We define a function G^P as:

$$G^P(\mathbf{P}) = \sum M_i/(w_1*DT_1+w_2*DT_2+w_3*\sum D_i(1+\sin(\alpha_{i,1})+\sin(\alpha_{i,2}))$$

where $\sum M_i$ is the summation of the edge magnitudes, DT_1 is the distance between the path ends to the ends of the *feature segment*, and DT_2 is the average distance from the pixels on the path to the *feature segment*. D_i, $\alpha_{i,1}$ and $\alpha_{i,2}$ correspond to D_1, α_1 and α_2 in Figure 7-15), respectively. w_1, w_2, w_3 are the weights of each measurement (here we take the value 0.2, 0.2 and 0.8 respectively). Among all the paths found, the one $\mathbf{P}^*$ maximizing G^P is selected, i.e. $\mathbf{P}^*$= arg $\max_{\mathbf{P}}$ ($G^P(\mathbf{P})$). We show a few examples in Figure 7-16. This edge detection is not real-time and the calculation time depends on the size of the images.

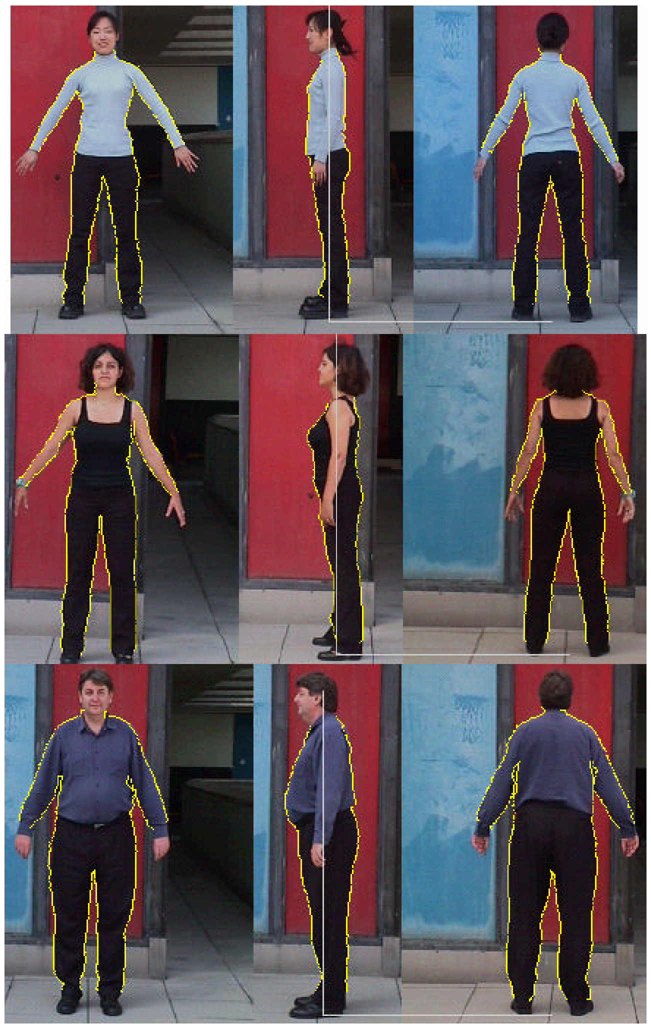


Figure 7-16: Images with super-imposed silhouette

7.4.2 Fine skin modification with silhouette information

After the skeleton adjustment and rough skin modification in the previous sections, the skin parts have been adjusted into proper orientation and rough size. Now we discuss how the image silhouettes are used to further modify the body so that the final body will produce the same silhouettes as those extracted from the images.

To build the 2D-3D association, we back project the 2D into 3D space. The silhouettes from the frontal or back view are mapped onto the *XY* plane and also the side view is mapped onto the *ZY* plane. For each view *v*, every slice S_i has two points, $\mathbf{V}^{v}_{i1}$ and $\mathbf{V}^{v}_{i2}$ on the occluding slice. These two points correspond to two pixels on the silhouette. Denote these two corresponding pixels as $\mathbf{P}^{v}_{j1}$ and $\mathbf{P}^{v}_{j2}$. Generally the number of pixels is much larger than the number of slices, and so we use the following formula to select the proper pixel pair:

$$P^{v}_{jk}=P^{v}_{1}+(MC_i-MC_1)/(MC_K-MC_1)*(P^{v}_{N}-P^{v}_{1})$$

where k=1,2, and $\mathbf{P}^{v}_{1}$ and $\mathbf{P}^{v}_{N}$ are the first and last pixel on the silhouette, $\mathbf{MC}_i$, i = 1, 2, ..., K is the mass center of slice i. All the parts except the arms have silhouettes from two views, i.e. frontal/back and side views. The arms only have silhouettes extracted from the frontal/back view. Now the problem is reduced to how to modify the model slice given 2 or 4 data points that are associated to points on the slice. We first apply a global translation to all the points on the slice so that the mass center of the slice coincides with the midpoint of the two back-projected pixels. Then we do a global scaling to the slice. The scaling factor is estimated as the ratio of the distance between the two associated pixels and that between the two points. The silhouette from frontal/back image is used to scale the slice on *XY* plane and that from side view is used to scale in *ZY* plane. In order to ensure the two points $\mathbf{V}^v_{\mathbf{i1}}$ and $\mathbf{V}^v_{\mathbf{i2}}$ that sit on the occluding slice to produce the pixels same as the two associated pixels $\mathbf{P}^v_{\mathbf{i1}}$ and $\mathbf{P}^v_{\mathbf{i2}}$, we apply a translation $\mathbf{T}_{\mathbf{i},m}$ to each point $\mathbf{V}_{i,m}$ of the slice as follows:

$$T_{i,m} = w*(P^v_{i1} - V^v_{i1})_{i1} + (1-w)*(P^v_{i2} - V^v_{i2})$$

where w=ArcL ($\mathbf{V}_{\mathbf{i,m}}, \mathbf{V}^v_{\mathbf{i2}}$)/ArcL ($\mathbf{V}^v_{\mathbf{i1}}, \mathbf{V}^v_{\mathbf{i2}}$), m=1,2,..., N_I, N_i is the number points on slice S_i, and ArcL(.) is a function computing the arc length of the slice curve. This makes sure the modified slice will generate the exactly same image pixel points under the same projection while keeping the curve smoothness.

Special care is needed for the shoulder-upper arms joining. The generic body has slices that are used to smoothly transit the shoulder to upper arm (see Figure 7-12 (d)). However there is actually no explicit corresponding information in the image. Fortunately, we can rely on the association of the upper arm with the torso to adjust these few slices. First of all, the orientation and the size in *XY* plane of these slices are adjusted by making use of the silhouette generated by the *middle shoulder point* and the estimated *armpit point*. Secondly, we have to enforce the upper arm to be seamed to the armhole, which is associated with the torso. In such a way, we can estimate a rough scaling in *YZ* plane since the torso has been modified by its side view silhouettes.

7.5 Texture mapping

We use only the frontal and back views for texture mapping since the two views (not the side view) are enough to cover the whole body except for the head. The head texture mapping is done with the frontal and the side views as shown in Section 7.3.1. This section is developed under collaboration with a colleague, Jin Gu, in MIRALab.

For the texture mapping, we have to give texture coordinates to points on skin envelope. Since there are two images used, we have to make a partition of the skin envelope polygons, either to the frontal view or to the back view by checking the cross product of the vertex normal with the viewing vector. If the point belongs to the frontal (back) view polygon (i.e. visible to frontal/back view point), we define the point as having frontal (back) view. If the vertex belongs to both the frontal view polygon and a back view polygon, we define the vertex as having frontal+back view. Since the vertex with frontal+back view is located on the boundary of the frontal and side views, we set two texture coordinates, one in the frontal view image and the other in the back view image. For the other vertices, it is straightforward to set the texture coordinate either in the frontal view image or in the back view image.

To get the texture coordinates, we use a projection onto the *XY* plane in the image space. Here we have to pay attention since there are two bodies either from the frontal and the side views or the back and side views. We follow the process such as:

1. Deform the body with the back and side views;
2. Project back/frontal+back viewpoints onto the back view image plane to get the texture coordinates;
3. Deform the body with the frontal and side views;
4. Project frontal/frontal+back viewpoints onto the frontal view image plane to get the texture coordinates.

Then the final individualized body has the proper texture mapping on both the frontal view and the back view.

However there are still some problems on the boundaries as shown in the left side images in Figure 7-17. There are mainly two reasons to cause this problem. First, due to the size of polygons on the virtual body, which is much bigger than the pixel size of images, the projected boundaries of the triangles on the frontal and back view boundary do not match to the detected boundaries based on pixel size. In addition due to the limited digitization resolution of the camera, the pixel colors on the boundary of foreground and background are usually the smeared combination of the boundary and foreground color. So it makes the noisy effect of the texture is magnified by the 3D triangles on the boundaries. Secondly although the two images used for texture mapping are usually taken under same illumination condition, they are not necessarily to have the same visual intensities or colors. So when they are mapped onto the 3D body, the difference in the color and intensity can be easily perceived.

In order to remove the first cause by digitization process, we modify the pixel colors within the neighborhood of each edge pixel. Since from the previous edge detection processing, we have already known which side of the boundary is background, we search along the perpendicular direction to the edge for a foreground pixel and take its color as the color for the edge neighborhood pixels. As the result shows, this simple processing removes the noisy effect of the digitization process.

Next, we need to smooth the frontal and back texture to remove the difference between the two images. From the feature points given in the previous processing, we can recognize semantically the various body parts, hence establish the part correspondences between the two images. We further find the pixel correspondence according to the boundary lengths. Within the neighborhood of the two pixels from frontal and back view images respectively, we use a linear blending. Let C^F and C^B are edge pixels in correspondence from the frontal and back view images respectively. Then for any pixel p^F and p^B in the linear neighborhood of length L_ε defined to be perpendicular to the local boundary at C^F and C^B respectively, we compute its color using the following blending function:

$$F^{bld}(p^F) = \alpha^F{}_1 * F(p^F) + (1.0 - \alpha^F{}_1) * B(p^B)$$

$$B^{bld}(p^B) = \alpha^B{}_1 * F(p^F) + (1.0 - \alpha^B{}_1) * B(p^B)$$

where $\alpha^F{}_1 = 0.5*(1.0+d(C^F,p^F)/L_\varepsilon)$, $\alpha^B{}_1 = 0.5*(1.0+d(C^B,p^B)/L_\varepsilon)$, F(.) and B(.) denote the original color of the original p^F and p^B while F^{bl} (.) and B^{bl} (.) denote the blended color for the pixels under consideration. Figure 7-17 shows the texture mapping results before and after texture blending.

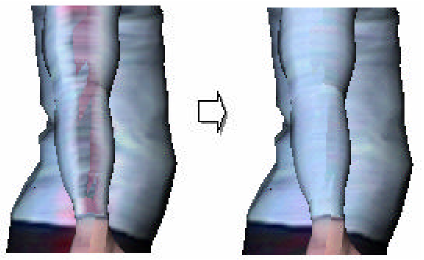


Figure 7-17: The effect of texture blending.

7.6 Results

7.6.1 Elbow skeleton correction

For some people, the elbow is bent and it creates the skeleton on the elbow region is visible coming out of the skin as shown in the left side of *Figure 7-18*, where the input photographs are shown in *Figure 7-25*. The problem comes from the way to construct skeleton. The skeleton is constructed using feature points as described in Section 7.3.2 and we construct the elbow skeleton joint using shoulder and wrist feature points as shown in *Figure 7-8*.

To correct the problem, we adjust the skeleton with silhouette information. So we move the elbow skeleton on the center (it can be positioned with proper bio-mechanical skeleton position with skin envelope) of the skin envelope of elbow which is a contour in our data structure. However since the skin parts have the corresponding skeleton joints, which transform the local coordinates of the skin part $\boldsymbol{i}$ to global coordinate by 4x4 matrix $\boldsymbol{M_i}$ (refer Section 7.2), we have to adjust the local coordinates of skins of upper_arm and lower_arm to have the same global position even though the position of elbow is changed. So we calculate the $\boldsymbol{M'_i}$ with updated position of elbow skeleton. Then to get the corrected coordinates $\boldsymbol{P'_i}$ of the coordinates $\boldsymbol{P_i}$ of points on the skin, we apply the following equations.

$$\boldsymbol{P'_i = P_i\, M_i\, (M'_i\,)^{-1}}$$

The resulting skeleton is shown in the right side in *Figure 7-18*.

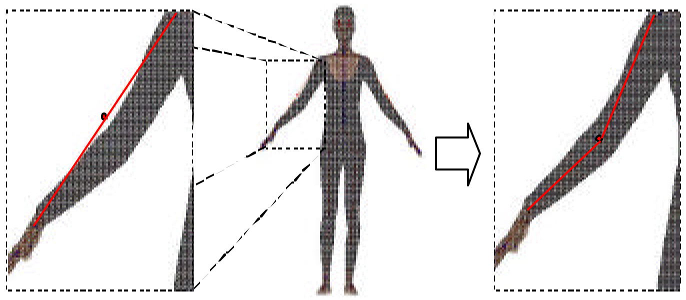
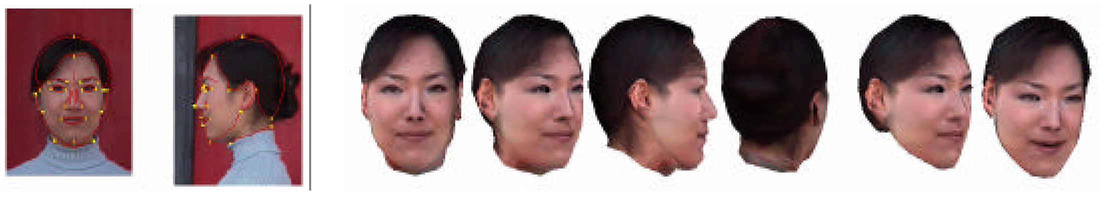


Figure 7-18: A problem on elbow for people with bent arms.

7.6.2 Connection between body and face

We processed face cloning and body cloning separately. Even though the size of the face is much smaller, we have to keep a detailed structure and high resolution for a face since we often zoom in the face to see facial animation and communication. So we use separated images for face cloning and body cloning and these two cloning methods use different texture mapping schemes. The body texture comes from the frontal and back view while the face texture comes from the frontal and side view due to the sphere like shape, which needs the side view for proper texture for ear parts. When we have a face and a body reconstructed separately, we have to connect them properly to make a smooth envelope for a perfectly smoothed body. We check the face size and location of the face on the frontal and side view body images using feature points *f_r_face*, *f_l_face* and *s_f_nose* in *Figure A-4*. Then simple translation and scaling locate the individualized face on the individualized body using features *face_L_HAIR6*, *face_R_HAIR6*, and *tip_NOSE* in *Figure A-1* and Figure A-3. We use an automatic sticking between them by the database built when we load the generic body as discussed in Section 7.2.1 where the nearest points on the neck and the head are found.



(a) (b)

Figure 7-19: (a) Two input photographs focused on the face. (b) Snapshots of a reconstructed head in several views and animation on the face.

Figure 7-19 (b) shows several views of the final reconstructed *head* out of two photographs in Figure 7-19 (a). When we connect this *head* with a body, we remove the *neck* since the *neck* is from the body due to the body skeleton animation for face rotation. The face animation is immediately possible as being inherited from the generic *head*. See the last face in Figure 7-19.(b) to see an expression on a face. The final bodies are shown in Figure 7-20 with the input images in Figure 7-16.

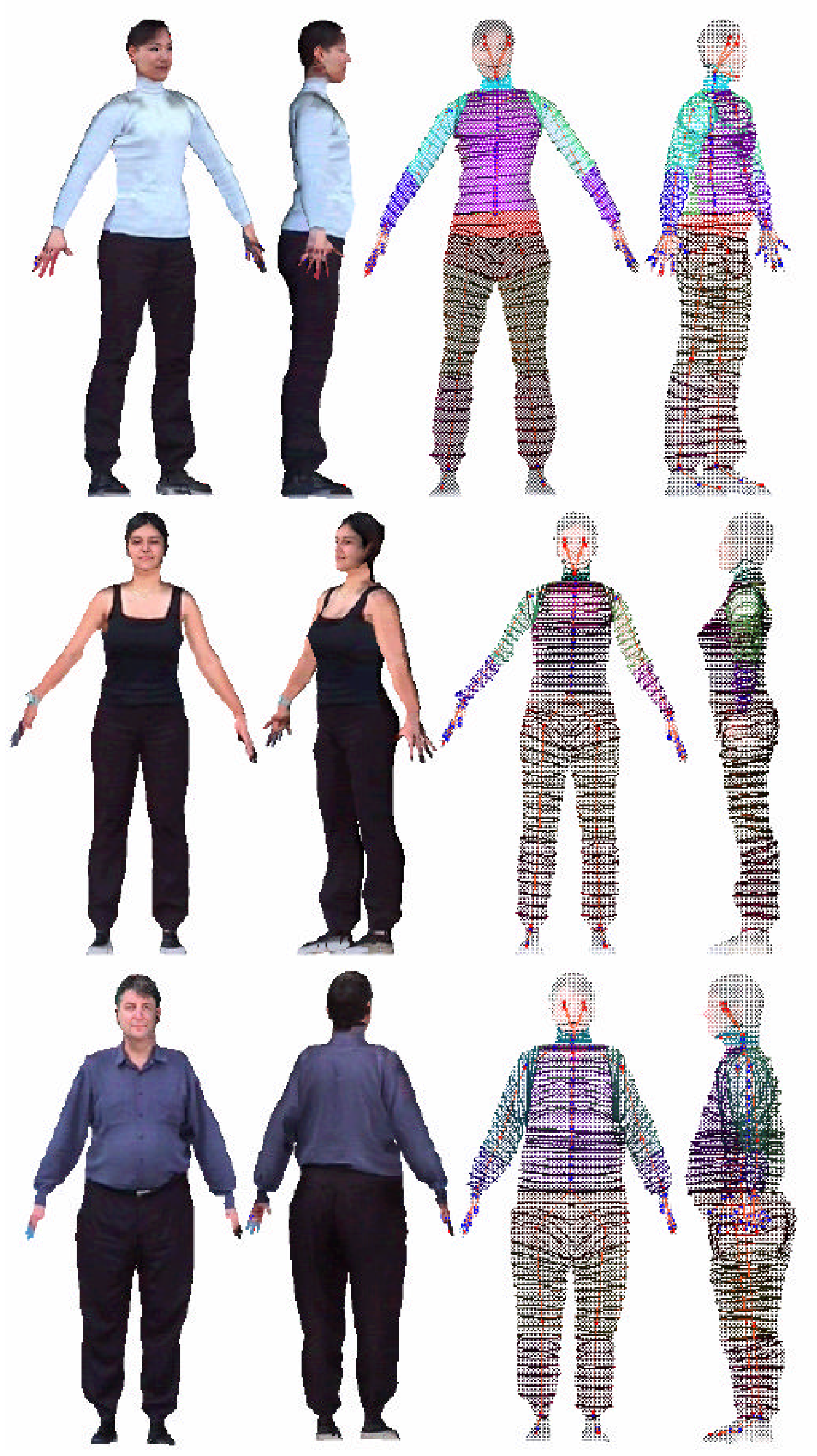


Figure 7-20: Final H-Anim 1.1 bodies with detailed individualized faces connected to the bodies. They are modified from one generic model.

7.6.3 H-Anim default posture

As mention in Section 7.1, the H-Anim default posture is fixed to be able to animate. Our photographs as input have posture a bit different from the defined default posture for the reason of automatic edge detection. So after construction of the individualized virtual body, we correct the posture by modifying the skeleton joint angles for arms and legs.

7.6.3.1 Joint angles for skeleton

The skeleton joint coordinates in H-Anim are based on global coordinates, which means the location is indicated by (x, y, z) with origin (0, 0, 0) between two feet. This is a convenient criterion for modeling purposes, but it does not provide the direct way to animate skeleton joints where the only angles need to be updated without changing the skeleton segment length. There are three choices to represent coordinates of skeleton joints.

1) Global origin and rotation
2) Local origin on the parent joint and global rotation
3) Local origin with rotation, so called moving frame

The third method is the most proper way for animation purposes since, for instance, the arm can be bend only by updating 'flexion' value regardless the shoulder has translation or rotation. Since we do not need to deal with sophisticated animation to correct postures, we take the second method. So to correct the posture from legs and arms open posture to close posture, we change the representation of the coordination of skeleton joint from (x, y, z) originated on (0, 0, 0) to (ρ, ϕ, θ) originated on the parent's skeleton joint position by

$$x \leftarrow x - x_{parent} \qquad y \leftarrow y - y_{parent} \qquad z \leftarrow z - z_{parent}$$

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$x = \rho \sin\phi \cos\theta \qquad y = \rho \sin\phi \sin\theta \qquad z = \rho \cos\phi$$

Once we have the representation in (ρ, ϕ, θ) originated on the parent's skeleton joint position, it is possible to correct the angle of legs and arms. We set angles for *r_elbow*, *l_elbow*, *r_knee* and *l_knee* to be the default straight down position. Since this representation is not the moving frame, we have to correct all children joints with the difference between the original joint angles and updated joint angles. Figure 7-21 (a) and (b) show the corrected skeleton into H-Anim default posture.

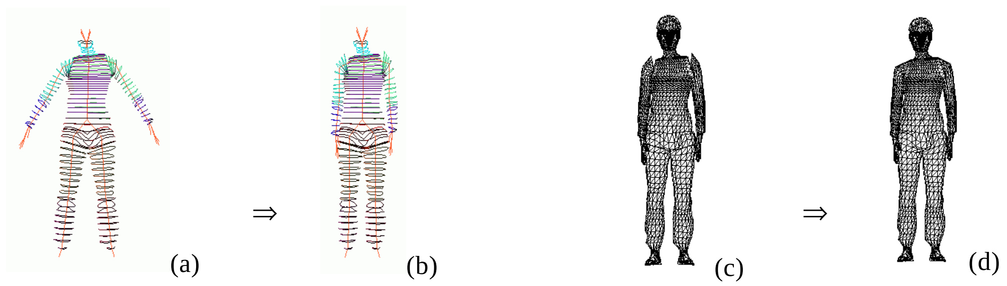


Figure 7-21: (a) to (b) shows correcting the joint angles for legs and arms. (c) to (d) shows correcting the skin envelope.

7.6.3.2 Skin adaptation

A modification of skeleton without applying skin deformation may create critical problems, especially for the shoulder part. See Figure 7-21 (b) and (c) for the separated shoulder envelope. Since we do not have data of the person in an H-Anim default posture, the simplest solution is the linear interpolation between mid-shoulder point on *torso* (See Figure 7-12(c)) and end-shoulder point on *upper_arm* (See Figure 7-12(d)). More sophisticated skin connection is also possible by using splines such as Bezier splines or

Hermite splines. The mid-shoulder point, end-shoulder point and the point in the second next contour on the *upper_arm* after end-shoulder point can be control points as shown in *Figure 7-22*. Then the shoulder part has a smoother surface.

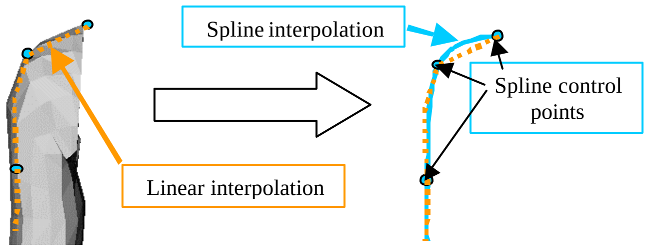


Figure 7-22: Spline connection for shoulder part.

For the *upper_leg* skin parts, simple connection with hip skin part is adaptable. Figure 7-21 (d) shows the final H-Anim humanoid in default posture, which has the smoothly connected skin part.

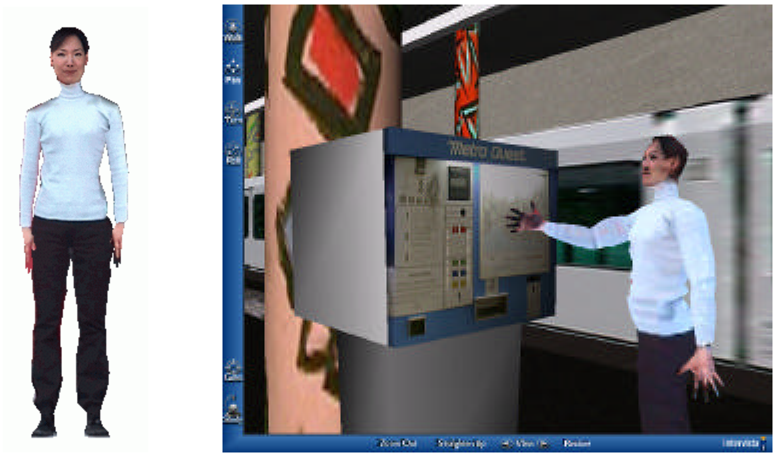


Figure 7-23: H-Anim humanoid in default posture and animation of a virtually cloned human in a virtual environment.

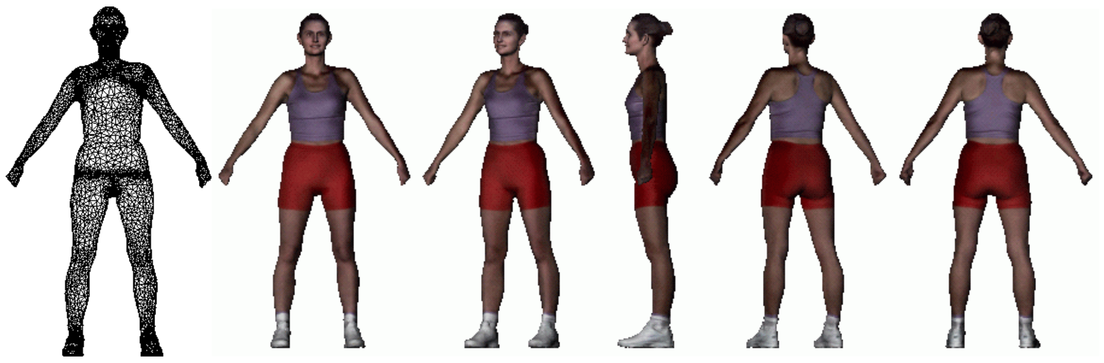
Since the generic body had H-Anim 1.1 structure, the individualized bodies keep the same structure, which means we can animate the bodies immediately. The final bodies are exported into VRML format and can be loaded in public web browsers. Figure 7-23 shows an animation example in a virtual environment. Since we are using seamless skin structure for a real-time animation, the skin and textures are smoothly connected during animation.

7.7 Validation of the cloning results

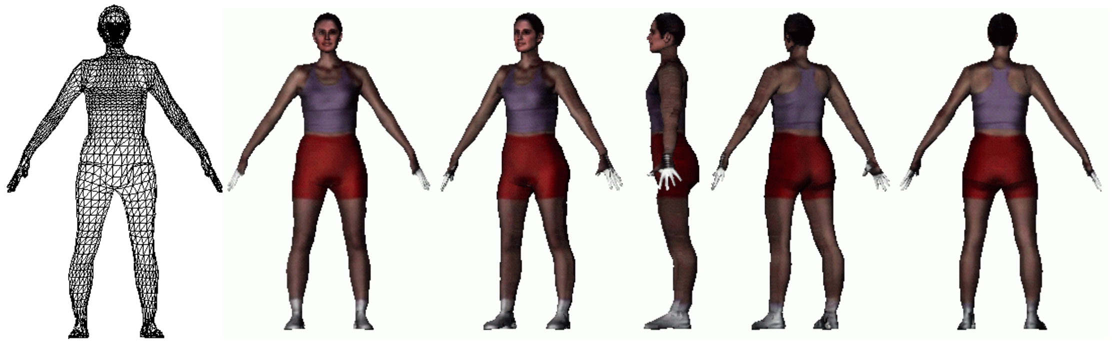
We use the same method used in Section 3.5 for the validation for the face cloning result. We use two kinds of validation with the body part of the same data collected from a freeware web site, *http://www*.cyberware.com/ wb-vrml/tammy/tammy.wrz, which was produced by Cyberware Digitizer for human body as described in Section 2.8. "tammy.wrl" is composed of 12 body parts, such as head, larm, lfoot, lhand, lleg, lshoulder, rarm, rfoot, rhand, rleg, rshoulder and torso.

7.7.1 Visual validation

We use snapshot to take three orthogonal photos of the body part after loading the laser-scanned model. We use the generic model in *Figure 7-6*. The first and third snapshots in *Figure 7-24* (a) are used as the input photographs (301x309) for the face cloning. The visual comparison of the original data and reconstructed data is shown in *Figure 7-24*.



(a) The wire-frame and snapshots of the laser-scanned model "Tammy"



(b) The wire-frame and snapshots of the reconstructed model

Figure 7-24: Comparison between two body snapshots in (a) and (b) taken in the same angle.

7.7.2 Error measurement with laser scanned data

Body part	***Avg. Error (cm)***
hip	1.18313
left_upper_arm	1.03196
front_torso	1.03044
right_upper_leg	0.927971
left_lower_leg	0.835839
left_upper_leg	0.795227
right_lower_leg	0.763058
right_upper_arm	0.748069
neck	0.702401
back_torso	0.701802
left_lower_arm	0.66277
right_lower_arm	0.445429
total	**0.844698**

Table 7-1: Error of each body part. The body part is ordered in descending order of average error.

We compared two virtual models using distance measurement in the same way introduced in 3.5.2. For every point P_i on each body part of the reconstructed "Tammy", we calculate the normal vector n_i. Then we get the intersection Q_i between the extension of n_i and the laser scanned "Tammy". Then the error E_i of P_i is the distance between P_i and Q_i.

The bounding box of reconstructed "Tammy" using cloning methodology *is 1.08421 X 1.76357 X 0.291584* where *x(-0.543076 , 0.541134), y(0.037182 , 1.80075)* and *z(-0.159267 , 0.132317)*, which means the height of the reconstructed model is about 1.76m. *Table 7-1* shows the average error, minimum error and maximum error for each body part. The unit is *meter (= 100 centimeter)*. The *hip* has the biggest average error (1.18313 *cm*) and we think the error comes from the curved character of the *hip* part which creates difficulty to adapt the exact shape just by two silhouettes from the frontal and side views. The *front_torso* has the biggest maximum error and it also has the same explanations. Overall, the average of error for all body parts is less than *1cm* which is acceptable when we consider the input data which are the only two views (the frontal and side views) used for shape reconstruction. Remember the back contributes only for the texture to make a final result. To make better results, the most important thing is to improve the generic model in a more natural shape, specially the region of highly curved surfaces since these parts are not adaptable individually by only two views.

7.7.3 Error measurement with tailor's data

The comparison between a living person and the virtual model is more difficult since the person is very deformable and she/he never stays in the same static shape. Nevertheless, we do error measurement with tailor's data and reconstructed model using cloning methodology. We take photos of "Marie-Jo" whom we check several measurements on her body. To measure her body, we consider the measurement used for cloth making. The measurement is not accurate since when we measure several times, it shows different number every time. *Figure 7-25* shows the input photographs used for the reconstruction while *Figure 7-26* shows the reconstructed body from the photographs.

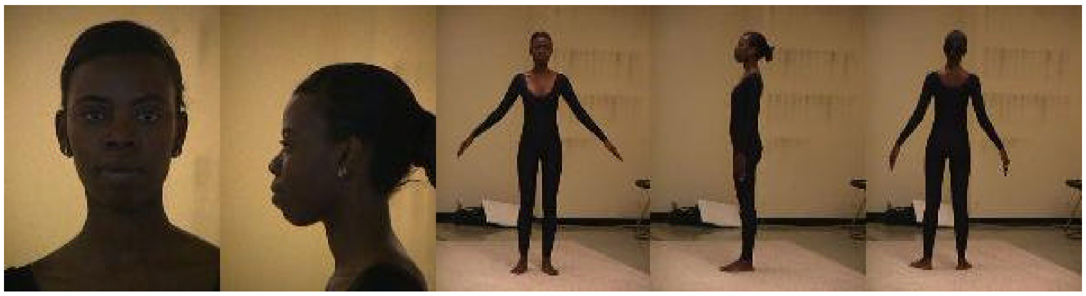


Figure 7-25: Five input photographs of "Marie-Jo"

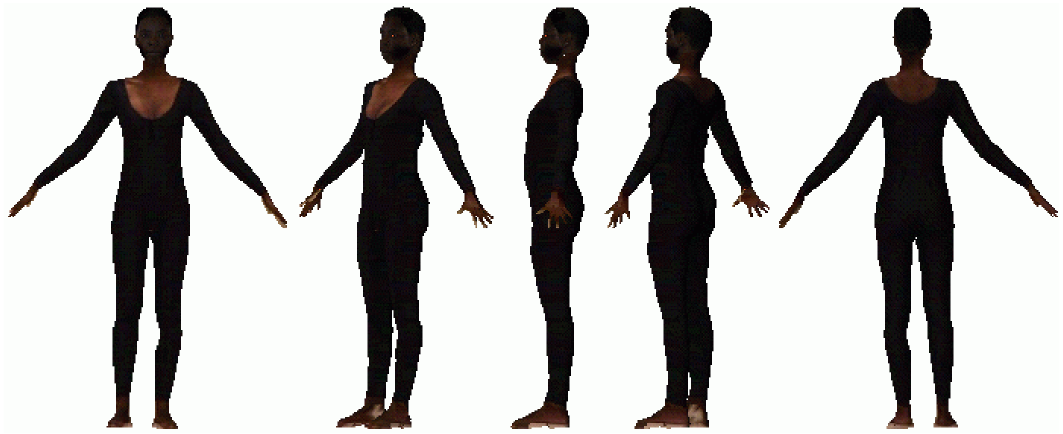


Figure 7-26: Reconstructed body of "Marie-Jo"

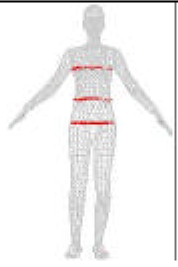
		*Tailor's (m)*	*model's (m)*	*Diff. (cm)*
	Height	1.74	1.75029	1.029
	Waist	0.68	0.727171	4.7171
	Hip	0.894	0.88432	-0.968
	Chest	0.835	0.819402	-1.5598

Table 7-2: Comparison of measurement on the actual body and reconstructed body.

To check waist, hip and chest, we cut the virtual body in horizontal planes. As a reference, the bounding box of the virtual model is *1.16365 X 1.75029 X 0.326703* where *x(-0.592613 , 0.571042), y(-0.0163918 , 1.7339)* and *z(-0.123011 , 0.203692)*. *Table 7-2* shows the comparison between the tailor's measure and the reconstructed body's measure. There are errors and as we can see the waist of the person becomes bigger while the chest becomes smaller. There can be several possible causes. One error comes from the estimation 3D just from two views of 2D. The input data we can get from photograph is very little differently from range data acquisition where all points on the view are used as input data. We use only silhouette data from two views to construct the 360-degree 3D. And the second comes from the tailor's measurement error. Another error possibly comes from the psychological reason where many women usually want to be measured with smaller waist and bigger chest and she breaths in while the tailor measures the waist and breast lines, which results that she deforms her body and it creates the error.

7.8 Conclusion

In this chapter, we introduce a model-based approach to photo-realistic animatable virtual human cloning from several photographs. The method takes as input two photos of the face of the subject and three photos of her/his body. An H-Anim 1.1 compliant generic body is taken to serve as our reference model for a body. Unlike other existing systems, which require special environment to obtain input data, we seek the solution through a friendly user interface for an accurate localization of feature points, which are used both for automatic edge extraction and for modifying the generic body into individualized ones. A simple but effective heuristics-based boundary extraction algorithm is proposed to automatically extract the body silhouette from the images. Then to avoid the possible overlapping for skin parts, we introduce a two-step modification, first rough matching just with feature points information and then fine matching with detected edge information. The body texture mapping is processed using two images. Moreover, we show how we connect the individualized head (refer Chapter 3) to the individualized body to form a complete animatable human model. As a result, we are able to animate the cloned human models in virtual environments. It is the first photograph cloning methods applied to the continuously seamless mesh body modeling. This method shows better cloning of the whole body in terms of both reconstruction and animation. The results are used for real-time body animation with motion tracking and also used for 3D clothes extraction [VW 00], which combined the 3D body surface information with the 2D pixel information on texture images for clustering separated clothes.

Chapter 8 System Implementation

This chapter devotes to explain how the systems are implemented to guide the others produce the same system if they wish.

8.1 Face cloning program

The basis of the implementation are as follows. See *Figure 8-1* for the system architecture.

- Graphical interface – *Rapidapp* (SGI)/Visual c++ (PC)

  Rapidapp (SGI) is an interactive tool for creating applications by generating c++ code. It provides predefined interface components to facilitate the use of other graphics libraries such as *OpenInventor.* The c++ classed generated by *Rapidapp* provide a user interface that combines widgets and components based on *IRIS IM toolkits*. This developing environment offers an easy and rapid way to construct a flexible and user-friendly for the system. It is composed of several menu buttons and three *OpenInventor* viewers. Each button in the menu bars is connected to a callback function.

- Graphical library – *OpenInventor*™

  The 3D model visualization and interactive graphical operations of the system are based on *OpenInventor*, an object-oriented 3D toolkit built on *OpenGL* and *X Window* and used for interactive 3D graphics. Different rendering modes, various interactive manipulations to 3D models, and parameter adjustment of the camera and lights can be efficiently implemented with *OpenInventor.*

- Core code – c++

  It is used to support algorithms and control modeling in the system. These include main three models such as 2D-feature modeler, 3D-shape modeler and 2D-texture modeler. All these modules are presented as c++ classes.

- Linking to programs in MIRALab – DFFD and face animation program

  The DFFD routine is connected to the 3D-shape modeler to support several times manipulation and face animation library is connected to show the animation of the given face.

- Linking to free software – JPEG library

  JPEG library from the public domain is connected to support to read and to create images in JPEG format.

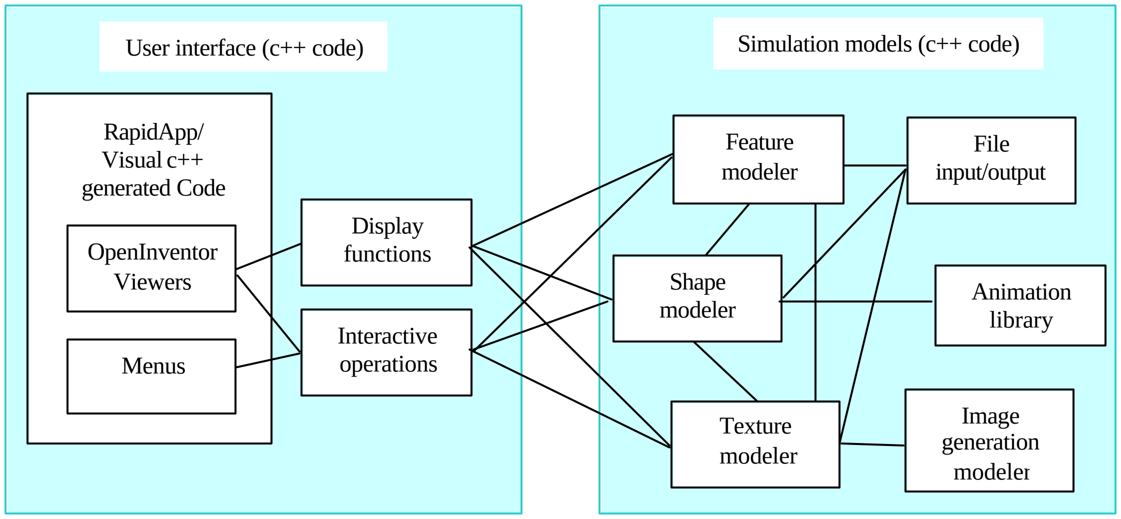


Figure 8-1: System architecture for face cloning system from photographs/range data

8.1.1 User-friendly Graphical Interface

There are three managers and each of them is responsible for one *OpenInventor* Viewer. So we explain managers for 2D-feature manager, 3D-shape manager and 2D-texture manager by explaining the functions related to the viewers. *Figure 8-2* shows the looking of the GI.

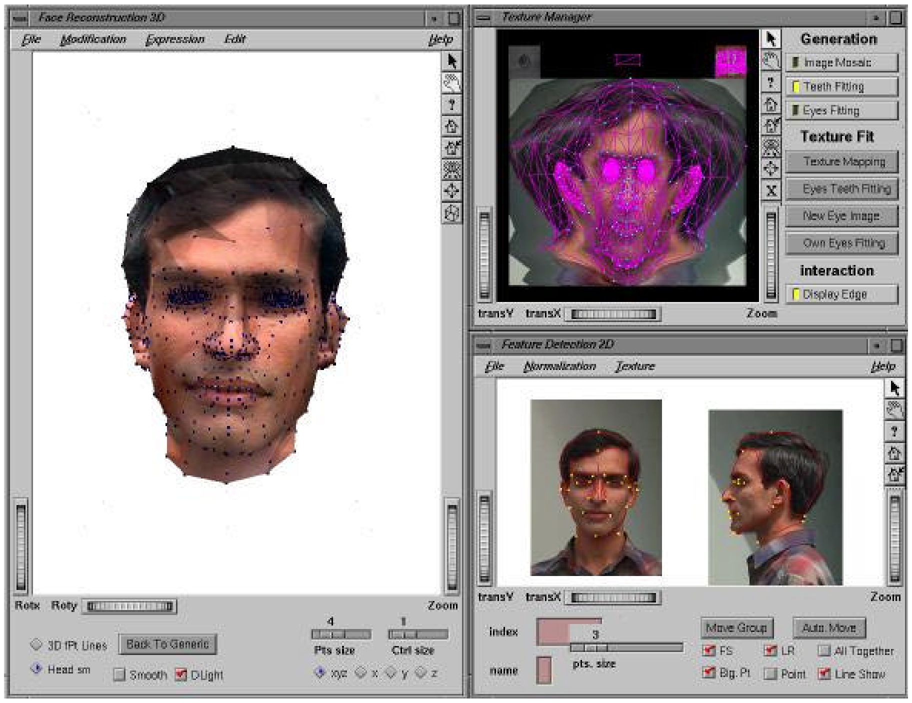


Figure 8-2: A user-friendly graphics interface with three managers. It is easy for any user to get a personalized animatable face in few minutes.

8.1.1.1 OpenInventor viewer

OpenInventor viewer is employed as a 3D graphical viewer in the system. Different modes of rendering (texture, color and wire frame) can be chosen to display the surface by specifying *OpenInventor drawStyle*.

The 3D object can be transformed. Triangles, vertices or a mesh of points can be picked on the surface. Beside the skin surface can also be modified interactively using the *PickFilterCallBack*. The camera can be changed easily in the scene by a variety of positions and orientations, while the light can be set up with different types, direction, color and intensity.

8.1.1.2 Function description for three managers

The detailed descriptions of the functions and how to create individualized faces are explained in the Chapter 3. The overall flowchart how to manipulate the system to get the result is also given in *Figure 3-1*. The system starts by loading two photographs. Then the 2D-feature manager supports the normalization and feature localization. We use 3D-shape manager to obtain the individualized shape and then the 2D-texture manager creates the texture images to give the skin color on the 3D surface.

To introduce more independent functions of the system, we use graphical outface. Since the graphical interface menu buttons initiate the functions, we describe the functions by explaining the callback functions called by the button on the graphical interface. While *Figure B-5* shows the several buttons and *OpenInventor* viewer corresponding to 2D-feature manager, *Figure B-6* and *Figure B-7* show the functions related to the cascade buttons. The *File CascadeButton* menus are mainly for file Input/Output and the *Normalization CascadeButton* are mainly for the normalization parameters. *Texture CascadeButton* does only to open the 2D-texture manager.

See *Figure B-8*, *Figure B-9* and *Figure B-10* to see the descriptions of functions for the 3D-shape manager. *Edit CascadeButton* only changes the background color in the OpenInventor viewer, which is a public function, provided by *OpenInventor.*

See *Figure B-12* to see the descriptions of functions for the 2D-texture manager.

8.1.2 Communication

As we discuss in Chapter 6, there are some difficulties in obtaining satisfactory results using MPEG-4 FDPs in fully automatic way, especially for texture fitting between the hair and the face region and nose-wings due to the lack of feature points. We provide a user-friendly interface to accelerate the results, whose diagram is shown in *Figure 8-3*. Normally the 2D-feature detection manager is processed in a semi-interactive way and then the 3D-modification manager and 2D-texture manager follow in a fully automatic way. However, we introduce a visual feedback method to correct some minor errors, which also allows the possibility to make a diverse shape beside the shape from images. For instance, to make the nose and eyes bigger, after getting texture fitting, we can go back to 2D feature manager to move the feature points for eyes and a nose to a bit different position and go to the 3D shape manager to change the shape. Then it has nice texture, but also different shape from the given image.

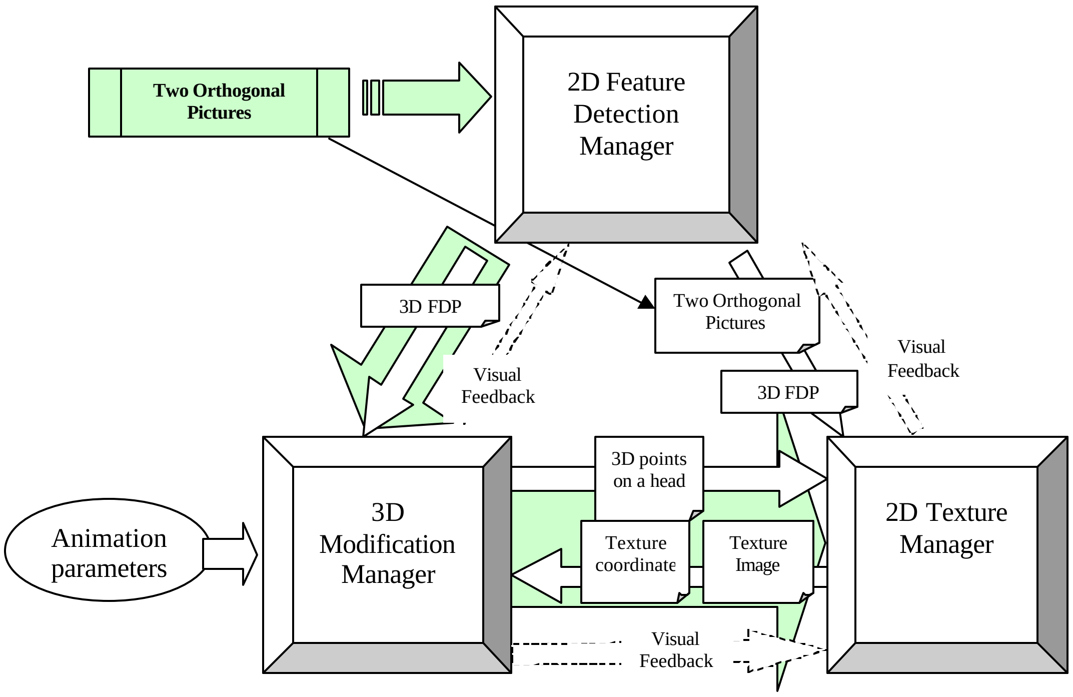


Figure 8-3: Communication in Graphical interfaces for a feature detection manager in 2D, a modification manager in 3D and a texture manager.

8.1.3 Input and output

Each viewer (manager) controls own input/output. It is implemented in order to allow the maximum flexibility to read and write information from/to the model data files. The system defines several specific formats and utilizes some In-house file formats and public file formats for images.

- General input for the face cloning system
    1. A file of the generic head shape data in *SM* format –

       generic.sm6
    2. A file of the generic animation region data in *RG* format –

       generic.rg 7
    3. Images of eye and teeth –

       eye.jpg or eye.rgb, teeth.rgb or teeth.jpg
    4. Expression parameters if we want to see the expression inside cloning system
- Individual input for the face cloning system from photographs. Note that they do not need to be perfect orthogonal, but need to be almost-orthogonal
    1. A photograph of the frontal view –

       personName_FV.jpg
    2. A photograph of the side view –

6 *sm* : In-house surface information file format for face and cloth used in MIRALab-University of Geneva.

7 *rg*: In-house regional information file format for face used in MIRALab-University of Geneva.

personName_SV.jpg

- Individual input for the face cloning system from range data
  1. A shape and texture coordinates in VRML1.0 format – personName.dataIV[8]
  2. A photograph of the corresponding to the shape.
- Individual output for the face cloning system
  1. A file containing norm parameter for the translation and scaling of the two photographs – personName.nrm[9]
  2. A file containing 2D feature information – personName.coord[10]
  3. A file with all data for head shape in OpenInventor fomat – personName_head.iv
  4. An image for texture mapping either in JPEG or in RGB format – personName.jpg or personName.rgb[11]
  5. A file for the shape in .sm format
  6. A file for the texture coordinates in .tx2[12] format
  7. FDPs and FAPs if FDPs are used for face cloning system.

8.2 Body cloning program

The implementation for the body-cloning program is basically the same as the face cloning system, so that we do not repeat the same explanation. If more detailed description is needed, refer Section 8.1.

The basis of the implementation are as follows:

- Graphical interface – Rapidapp (SGI)
- Graphical library – OpenInventor™
- Core code – c++

[8] *dataIV* : OpenInventor (VRML1.0) format with some specific definition of naming.

[9] *nrm* : A file format deigned specially for face cloning program.

[10] *coord* : A file format deigned specially for face cloning program.

[11] *rgb* : A image format specially for SGI.

[12] *tx2*: In-house texture coordinate information file format for face used in MIRALab-University of Geneva.

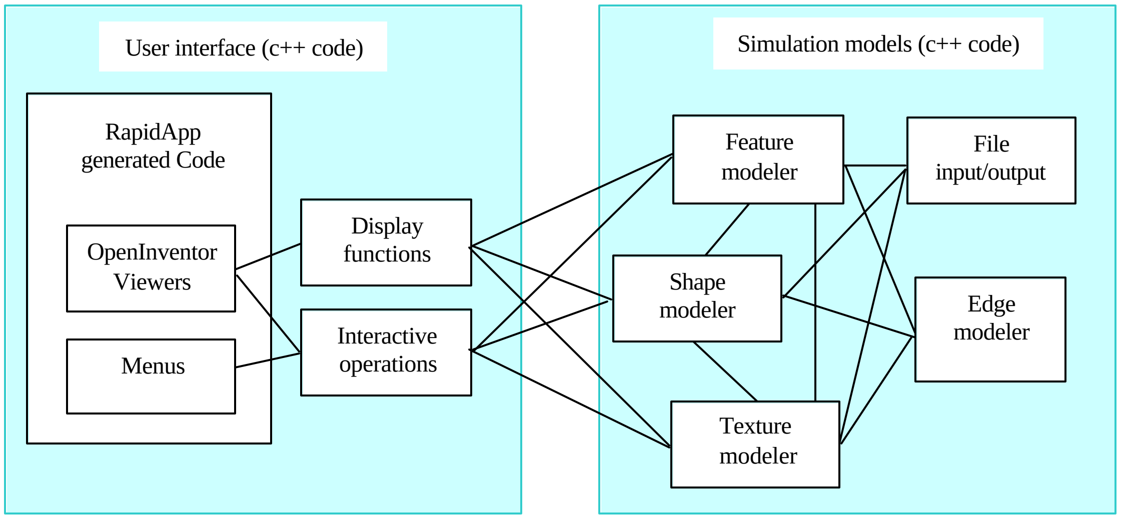
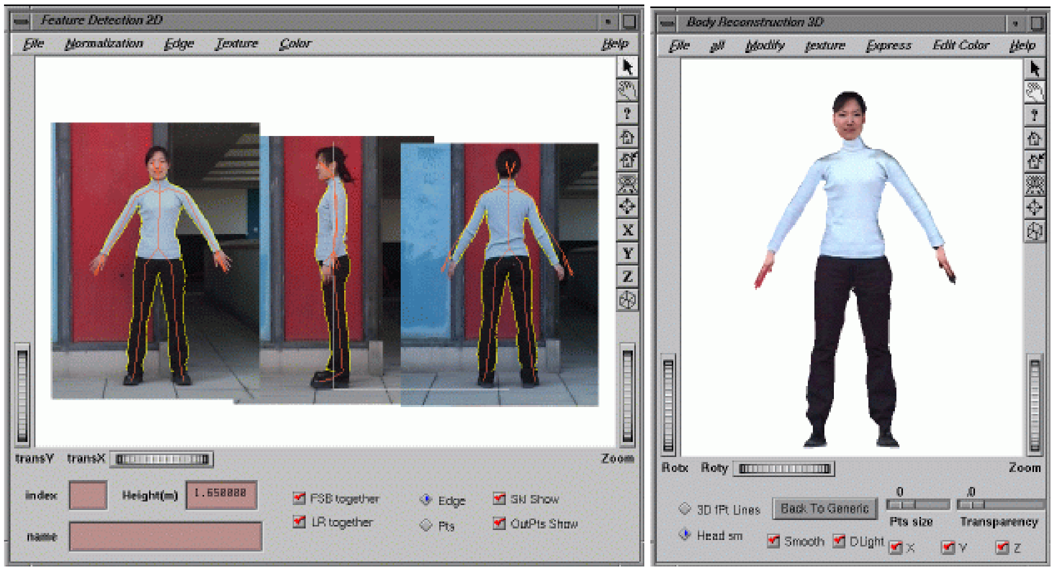


Figure 8-4: System architecture for body cloning system from photographs

8.2.1 User-friendly Graphical Interface

Figure 8-5 shows a user-friendly interface with two main windows for four managers. The 2D viewer is managed by the Feature manager and Edge manager while the 3D viewer is managed by shape manager and texture manager.



Height(m) 1.650000

Figure 8-5: A user-friendly graphics interface with two managers.

8.2.1.1 Function description for four managers

The detailed descriptions of the functions and how to create individualized bodies are explained in the Chapter 7. The overall flowchart how to manipulate the system to get the result is also given in *Figure 7-1*. The system starts by loading three photographs. Then the feature manager supports the normalization and feature localization. After we call the edge manager to detect the edges on images. Then we apply shape manager in two steps to obtain the individualized shape and then the texture manger is performed to create new texture images.

Since the graphical interface menu buttons initiate the functions, we describe the functions by explaining the callback functions called by the button on the graphical interface. *Figure B-13* shows the 2D viewer. There are three main *Cascade Button*s, where *Figure B-14* shows the *file CascadeButton* for file input/output and *Figure B-15* shows the *Normalization CascadeButton* are mainly for the height and normalization parameters. The *Edge CascadeButton* simply calls the function to detect edges with given features shown in the OpenInventor viewer. *Figure B-16* shows the 3D viewer. There are four main *Cascade Button*s. The *file CascadeButton* shown in *Figure B-17* is responsible mainly for output of the result in various formats. *Figure B-18* is the *modification CascadeButton* that is corresponding to the shape manager. *Figure B-19* is for the *texture CascadeButton* for the texture coordinate calculation and texture image generation with blending function managed by the texture manager. Since there are several steps to get the finalized body, there is a menu to be more easy manipulation. The *all CascadeButton* shown in *Figure B-16* is a shortcut to escape to press many buttons. The callback function is an integration of many functions to perform the back view modification and texture mapping, then frontal view modification and texture mapping to get the final body.

8.2.2 communication

The body-cloning program is less critical for features compared to the face-cloning program. Only one critical feature for the body-cloning program with the GI is the automatic edge detection. How easy the automatic edge detection depends on how homogeneous the background is. We have chosen to use the arbitrary background with help of feature points to detect edges. Sometimes the edges between legs or arms are not very obvious and it is not very simple to decide where the features are. Then the graphical interface is useful to give some visual effect to adjust features to get edges for our preference.

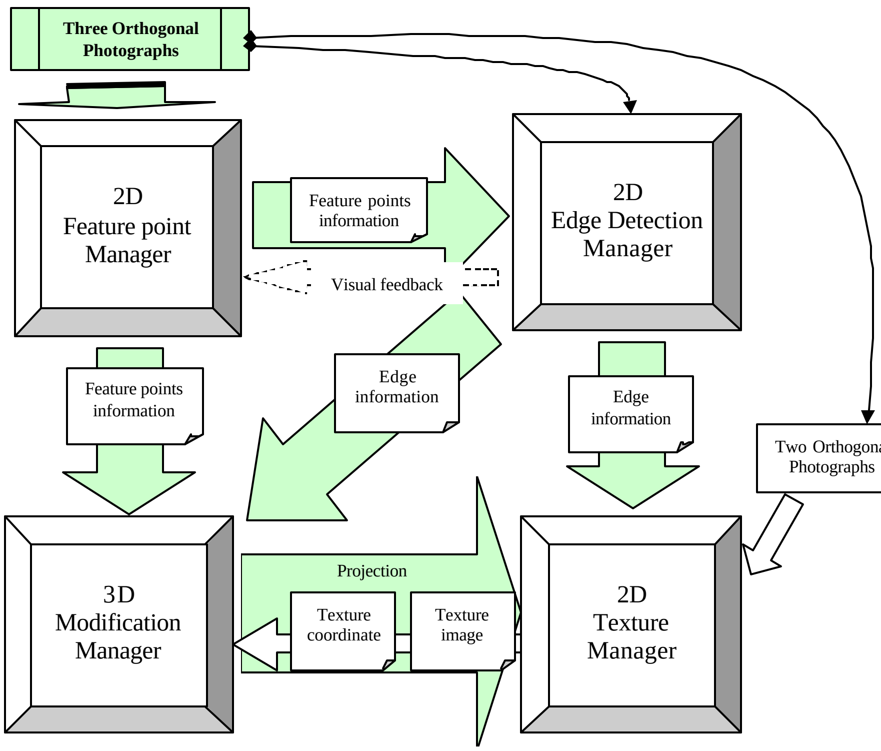


Figure 8-6: Communication in Graphical interfaces for feature and edge detection managers in 2D, a modification manager in 3D and a texture manager in 2D.

8.2.3 Input and output

Each viewer controls own input/output. The system defines several specific formats and utilizes H-Anim 1.1 format and public file formats for images.

- General input for the body cloning system
    1. Files of the generic head and body shape data
    2. A file containing the hierarchy of the skeleton
- Individual input for the body cloning system from photographs
    1. A photograph of the frontal view –

       personName_FBD.jpg
    2. A photograph of the side view –

       personName_SBD.jpg
    3. A photograph of the back view –

       personName_BBD.jpg
    4. The height of the person
    5. A file of the head of the person in OpenInventor format which is an output of the face cloning program –

       personName_head.iv
    6. An texture image of the head of the person in JPEG format which is an output of the face cloning program –

       personName.jpg
- Individual output for the body cloning system
    1. A file containing norm parameter for the three photographs –

       personName.hnrm13
    2. A file containing 2D feature information –

       personName.ptImg 14
    3. A file with all data for body and head shape in OpenInventor fomat –

       personName_body.iv
    4. Files with all data for body and head shape in H-Anim1.1 fomat (VRML2.0) –

       personName.wrl15,

       personName_head.wrl,

       personName_left_hand.wrl, personName_right_hand.wrl,

       personName_left_foot.wrl, personName_right_foot.wrl,
    5. Three photograph for texture mapping in JPEG format –

       personName_FB.jpg, personName_BB.jpg, personName.jpg

---

13 A file format deigned specially for body cloning program.

14 A file format deigned specially for body cloning program.

15 A file in H-Anim 1.1 format

Chapter 9 Conclusion

Modeling is...

The process of creating useful “maps” of human movement.

- Version 1.0 by an unknown
- Version 2.0 by WonSook Lee

We dream and we play. The virtual-twin has the freedom to play our dreams. The virtual-twin gives us the way to immerse in a dream world.

Our aim is to realize our true shape in a virtual world without making any limitation to a specialized group of people. This thesis focused on a methodology to be used for everyone in an easy way. The easy input is now very critical for any approach, which aims to be used widely in this newly aged time with booming Internet and the accelerating Internet population. The Internet is the meeting point between Moore’s Law[16] and Metcalfe’s Law[17], which enables anybody in this world to access virtual world. Sooner or later, the individualized, personalized and more natural human looking environment will be realized in the computer world as we have progressed more and more user-friendly human-machine interface in the last two decades. We selected the photographs as our primary input for individualized face and body modeling and we successfully produced realistic looking virtual-twin devoted to the real-time face and body animation. Using our Feature-based approach on animatable virtual human cloning, everyone can access the virtual world in his or her shape.

First our contribution is summarized in Section 9.1 and then the list in Section 9.2 shows how well our cloning system has realized the efficiency and robustness in several public demonstrations where the system was completely exposed in front of people for the trials. Finally the discussion of the problems to be solved and further research plans in the final section 9.3 ends the thesis.

9.1 Contribution

We have described in detail the methodology and algorithms implemented in developing an integrated software system for computer-aided animatable human modeling. In our research, we have developed a face cloning system with two kinds of input data such as photographs and range data and body cloning system from photograph data. And as applications, we have done dynamic 3D-morphing system and MPEG-4 face cloning experiments.

The main contributions of our research work are the following.

- Easy input like photographs is the first priority to build the system and even the images sent by email are successfully processed to make the virtual-twin and animation.
- A complete integration of whole face and body parts from five photographs is implemented.
- The integrated methodology to deal with different input data for the face cloning is proposed.

[16] It states that the power of the silicon chip would double almost annually (the current definition is approximately every 18 months) with a proportionate decrease in cost.

[17] It says that the “value” of a network is the square of the number of its nodes.

- Several problems in steps in the conventional face cloning systems from photographs are solved such as efficient and robust semi-automatic feature detection methods, 3D-deformation approaches rather than in 2D, more robust 3D deformation using DFFD, fully automatic generation of seamless texture and etc.
- Continuous mesh for generic body is used for individualization and successfully implemented resulting real-time animation without texture problems.
- International standard H-Anim 1.1 for the body-cloning program is considered and the result is successfully produced in the standard format.
- We do an experiment to adapt our face feature points to the FDPs in the International standard MPEG-4 and discussed the good points and problems.
- User-friendly interface allows any user can use the system as proved in several public demonstrations.

The final and most important contribution is we make people feel that virtual world is there beside them. We would like to call them as **active** and **immortal** Narcisse[18].

9.2 Validation through public demonstrations

There are hundreds faces reconstructed using the cloning system from photographs. They are clones of MIRALab staff members, visitors, photographs sent by emails, photographs collected from the public domain on the net and visitors in several public demonstrations. The first public demonstration took place in Orbit'98, Basel, Switzerland where about 70 people are cloned during 5 days demonstration on a PC platform, thanks to Hyewon Seo who has ported the program from SGI to PC and many other people in MIRALab, Univ. Geneva, who perform the demonstration. In these events, several kinds of looking of people (Asian/Caucasian/Africa, female/male, young/aged) were cloned and animated in a virtual world.

1. ORBIT'98 in Basel (CH) - platform PC

   The virtual-twins of visitors are 3D-morphed with other visitors immediately.
2. CEBIT'99 in Hannover (DE) - platform SGI

   The virtual-twins of visitors are 3D-morphed with other visitors immediately.
3. SMAU'99 in Milano (IT) - platform SGI

   The virtual-twins of visitors are 3D-morphed with other visitors immediately.
4. TELECOM'99 in Geneva (CH) - platform PC

   The virtual-twins of visitors are integrated in the virtual scenery and animated immediately there.
5. Uni-Mail Fete'99 in Geneva (CH) - platform SGI

   The virtual-twins of visitors are 3D-morphed with other visitors immediately.
6. L'Oreal booth in Millennium Dome 2000 in London (GB) (one year demonstration) - platform PC

   The virtual-twin faces of visitors are processed through an aging system developed by a colleague, Laurence Boissieux, in MIRALab.
7. Computer-expo 2000 at Lausanne – platform PC

[18] Narcisse: A very beautiful young man who was seduced by his own image reflected on the fountain water and died from the passion he had for himself. A flower grew, where he died, took his name.

9.3 Problems to be solved and further research

The feature-based approach to reconstruct the human shape with animation structure shows the results with photo-realistic human with very easy and general inputs such as photographs. It shows this approach is very efficient for the general usage where the visual effect is very important. Nevertheless there are still problems to be solved to adapt several needs and further research is needed. We provide several ideas about problems and solutions for the further research in this field.

1. Limitation on the precision with photographs input

   However as we discussed in the validation sections such as Section 3.5 and Section 7.7, the feature-based reconstruction from limited angles of photographs can not make as accurate and precise shapes as laser scanners can provide. In this case, the face cloning from range data mentioned in Section Chapter 3 is useful to give animation structure on the accurate shape data. However in this case, some source need to provide the range data. Nevertheless it is useful when we need accuracy with animation or feature information on the geometrical shape such as in the medical field. If we like to extend the feature-based approach to the medical fields such as surgery simulation, more precise and biomechanical-based features need to be defined and the deformation of the shape with control points displacement must be based on the real movement of muscles and related biomechanical parameters.

2. Hairstyle

   In addition, we do not deal with the various hairstyles, which give a lot of characteristics to recognize people. These limitations of hairstyle causes some people to think the reconstructed virtual faces do not give the same impression as the photographs do. The research on the various hairstyle reconstruction from photographs is a must for the true representative of the person.

3. Multi-resolution for the level-of-detail (LOD)

   This thesis deals with only one kind of the generic model with a fixed resolution. However when the crowd simulation is performed with high number of virtual humans, LOD is a must and the virtual human modeling section is responsible to provide several levels. There are two possible solutions. The first is to make several levels of generic models for faces and bodies and generate virtual humans with several levels. We can provide the generic models from the loading module at the beginning and apply the cloning system several times to produce. The second is that we provide a LOD algorithm to reduce the resolution of the resulting virtual human. In this case, the generic model must have the highest resolution.

4. Body with clothes texture/ body with clothes on?

   The resulting body from the body cloning system has only one layer of skin to deform. The clothes are treated like skin when we do animation and deformation. This is efficient for real-time body animation. However to separate clothes from body envelope is a must if we like to have more sophisticated animation. A colleague, Takao Furakawa, in MIRALab successfully done an experiment for the automatic clothes construction from the cloned body from photographs [VW 00]. However this experiment made partition between visible skin region and clothes region in 3D. The cloth animation on a moving body needs to have an underlined body shape under the clothes. Since our reconstruction for the body has only one layer for both the skin and the clothes, further research to find the underlying body shape from outlying clothes is needed unless we find some specially designed clothes animation to deform clothes without underlying skin layer. We suggest two possible solutions. One is we take photographs of a person twice, such as first with very tight clothes showing all body outlines and second with clothes. Then an adjustment between two bodies from two kinds of input photographs is sufficient to support the separated body and clothes animation. The other solution is we find the underlying body shape from outlying clothes using the general information for the body shape. Natural shape of the generic model can help to find the body outlines under the clothes. If the generic body has quite natural shape (several generic body shapes depending on the ethnic groups must be also very helpful), the feature-based modification mentioned in Section 7.3 can be useful to find the body lines while the edge-based modification finds the clothes lines. In the body cloning system, we limit the clothes style by a shape similar to the body shape such as trousers. This limitation can be solved by providing clothes database and find the shape with a corresponding generic clothes. In some cases such as the human subject puts on a skirt, features need to be found more carefully.

Appendix A Feature Names

The following *Figure A-1*, *Figure A-2* and *Figure A-3* show face features while *Figure A-4* body features. The brighter and bigger points on *Figure A-1*, *Figure A-2* and *Figure A-3* are key features that guide other features mentioned in Section 3.1.3.

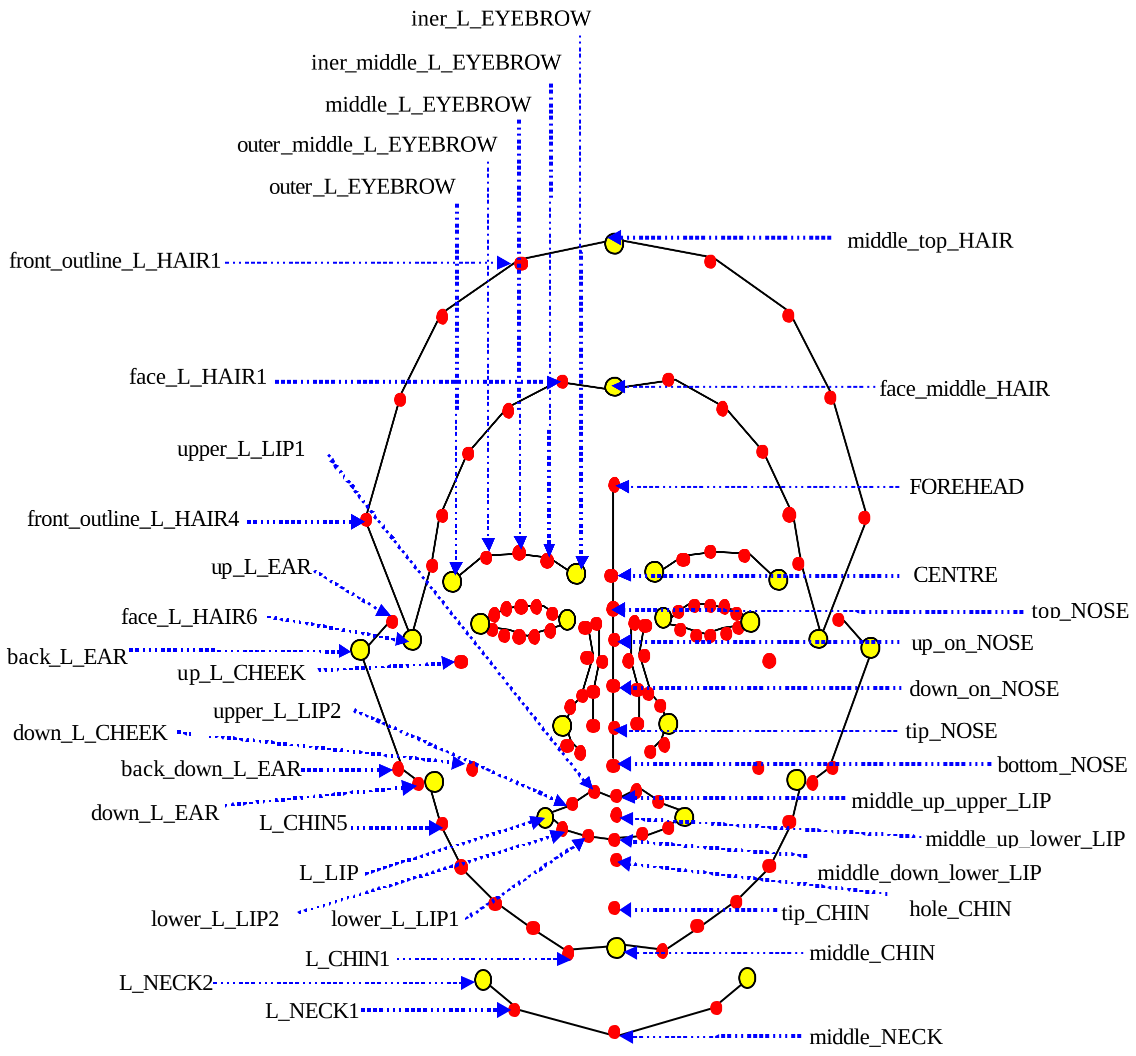


Figure A-1: Feature names on the left side and the middle part of the frontal view of the person.

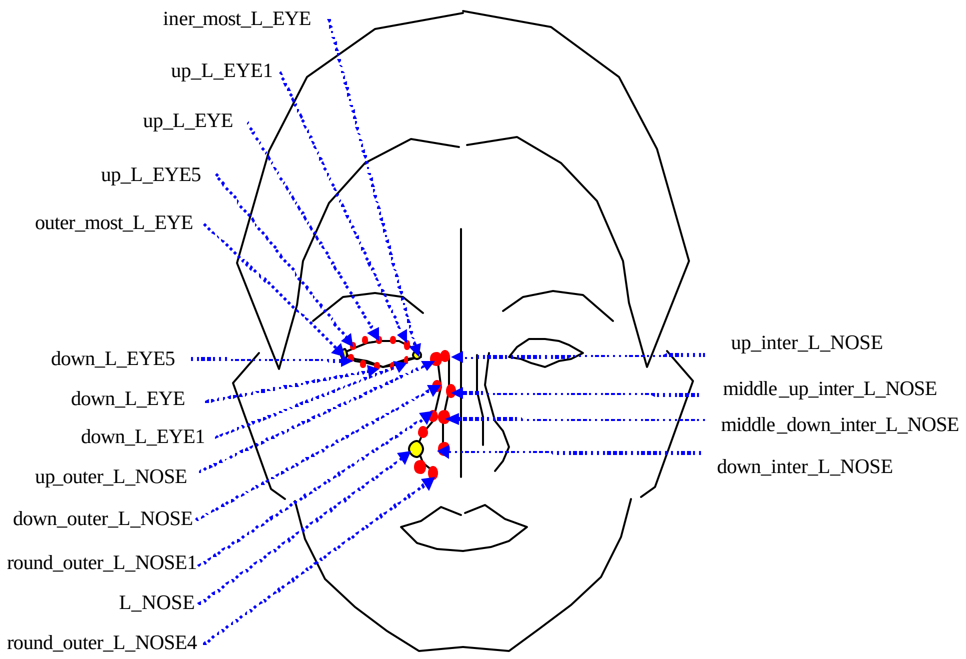


Figure A-2: Feature names on the left side the eye and the nose of the frontal view of the person.

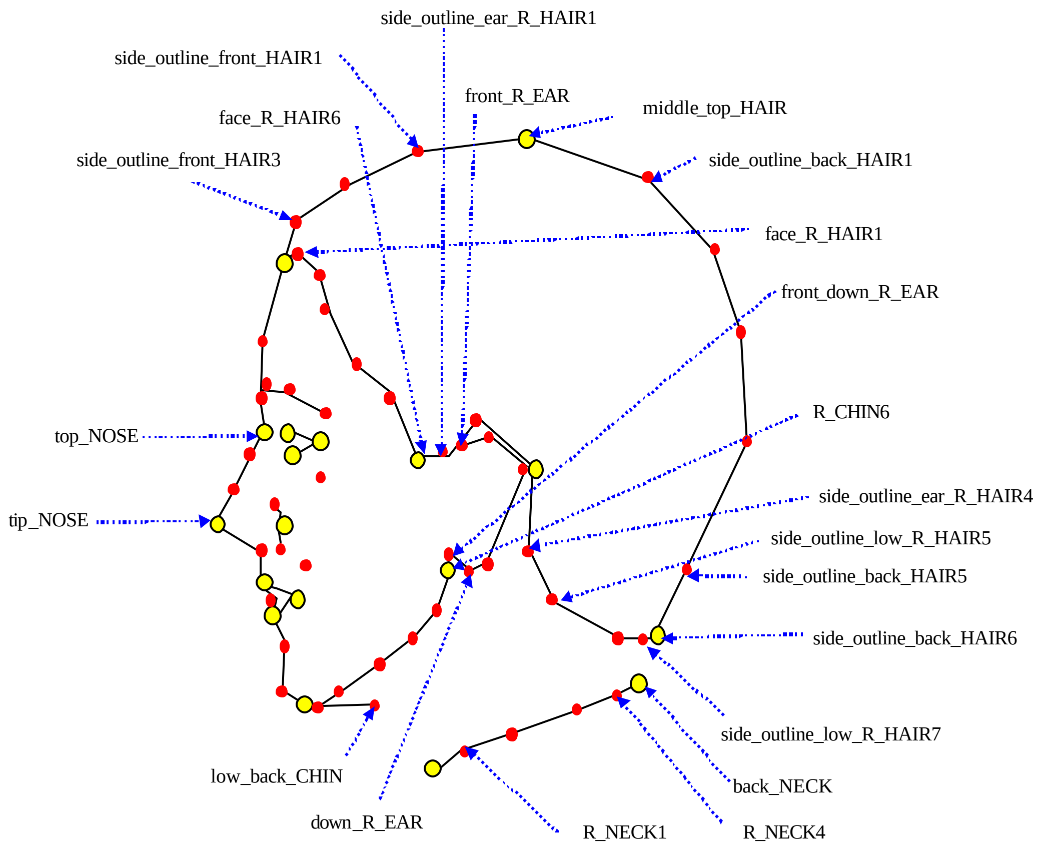


Figure A-3: Feature names on the right side of the side view of the person.

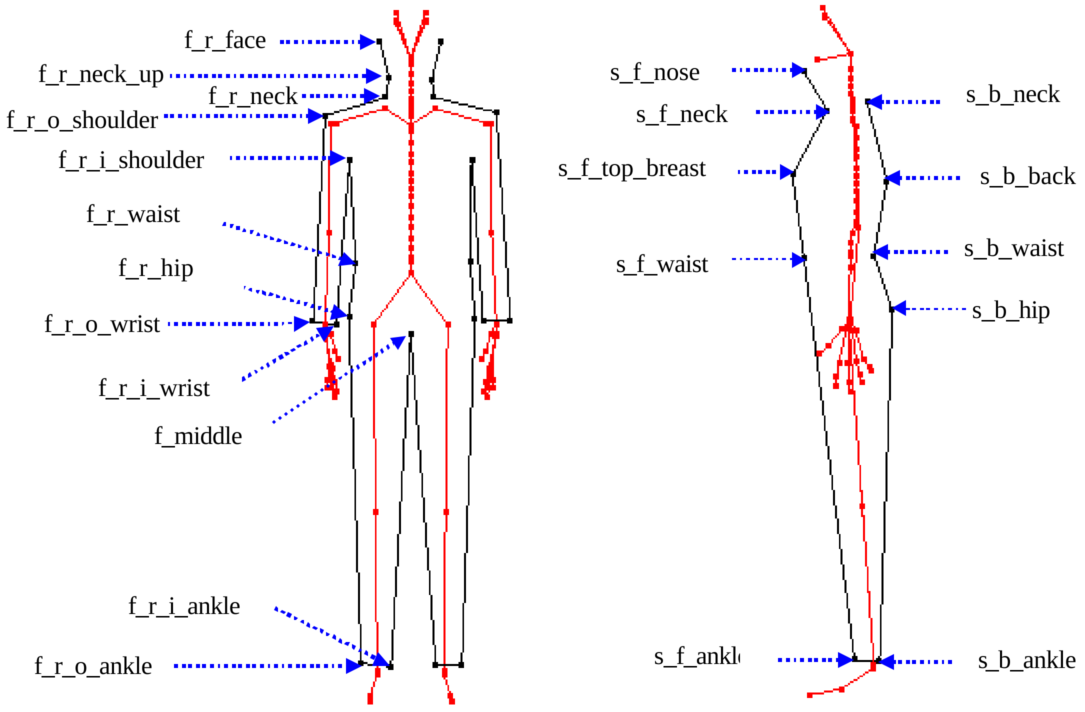


Figure A-4: Feature names on the body. The left side has the name as f_l_* dual to the corresponding the right side.

Appendix B Description for GI

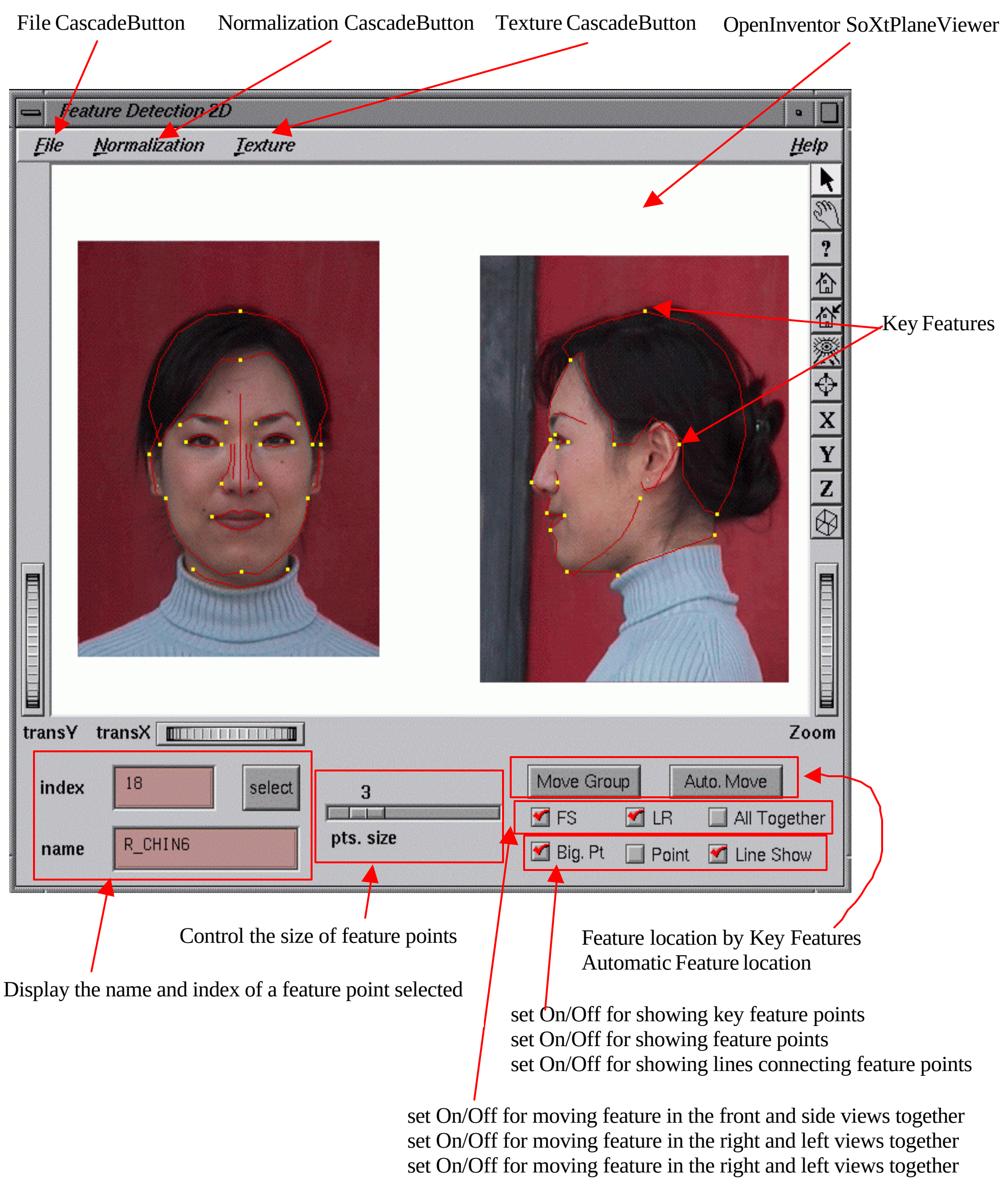

File CascadeButton
Normalization CascadeButton
Texture CascadeButton
OpenInventor SoXtPlaneViewer
Feature Detection 2D
File
Normalization
Texture
Help
Key Features
transY
transX
Zoom
index
18
select
name
R_CHIN6
3
pts. size
Move Group
Auto. Move
FS
LR
All Together
Big. Pt
Point
Line Show
Control the size of feature points
Display the name and index of a feature point selected
Feature location by Key Features
Automatic Feature location
set On/Off for showing key feature points
set On/Off for showing feature points
set On/Off for showing lines connecting feature points
set On/Off for moving feature in the front and side views together
set On/Off for moving feature in the right and left views together
set On/Off for moving feature in the right and left views together

Figure B-5: Face cloning: 2D Feature manager

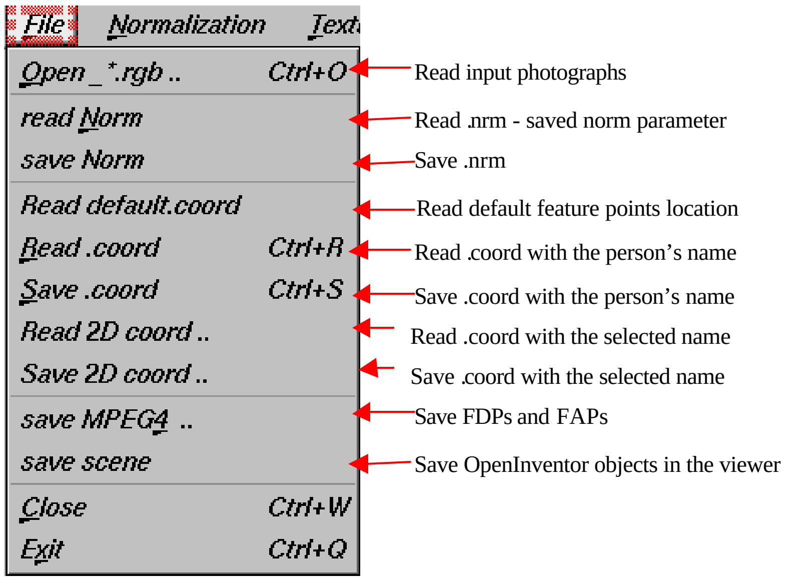

File
Normalization
Text
Open _*.rgb .. Ctrl+O
read Norm
save Norm
Read default.coord
Read .coord Ctrl+R
Save .coord Ctrl+S
Read 2D coord ..
Save 2D coord ..
save MPEG4 ..
save scene
Close Ctrl+W
Exit Ctrl+Q
Read input photographs
Read .nrm - saved norm parameter
Save .nrm
Read default feature points location
Read .coord with the person's name
Save .coord with the person's name
Read .coord with the selected name
Save .coord with the selected name
Save FDPs and FAPs
Save OpenInventor objects in the viewer

Figure B-6: Face cloning: Callback functions on File CascadeButton in 2D Feature manager

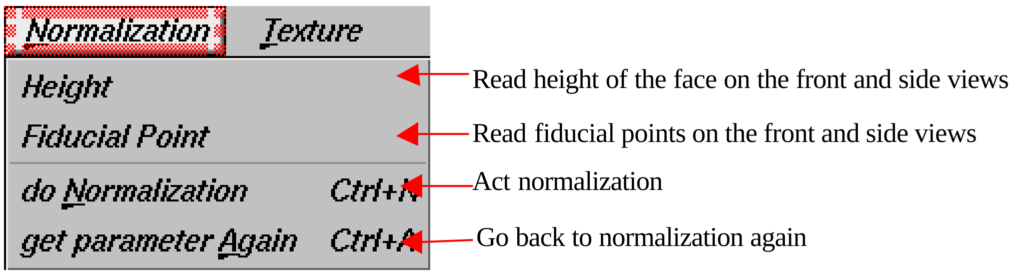

Normalization
Texture
Height
Fiducial Point
do Normalization Ctrl+N
get parameter Again Ctrl+A
Read height of the face on the front and side views
Read fiducial points on the front and side views
Act normalization
Go back to normalization again

Figure B-7: Face cloning: Callback functions on Normalization CascadeButtonin in 2D Feature manager

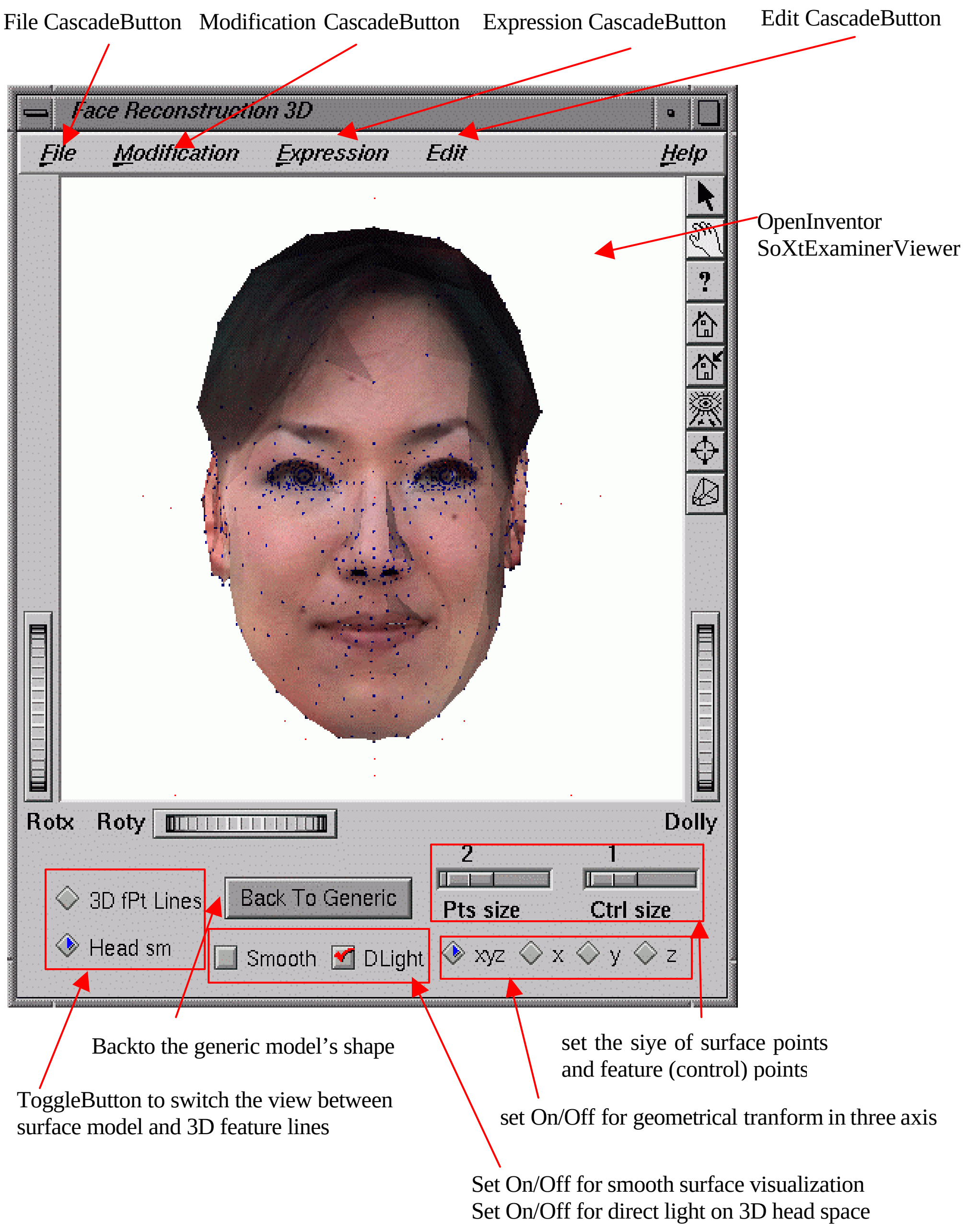

File CascadeButton
Modification CascadeButton
Expression CascadeButton
Edit CascadeButton
Face Reconstruction 3D
File
Modification
Expression
Edit
Help
OpenInventor
SoXtExaminerViewer
Rotx
Roty
Dolly
2
1
Pts size
Ctrl size
3D fPt Lines
Head sm
Back To Generic
Smooth
DLight
xyz
x
y
z
Backto the generic model's shape
ToggleButton to switch the view between surface model and 3D feature lines
set the siye of surface points and feature (control) points
set On/Off for geometrical tranform in three axis
Set On/Off for smooth surface visualization
Set On/Off for direct light on 3D head space

Figure B-8: Face cloning: 3D shape manager

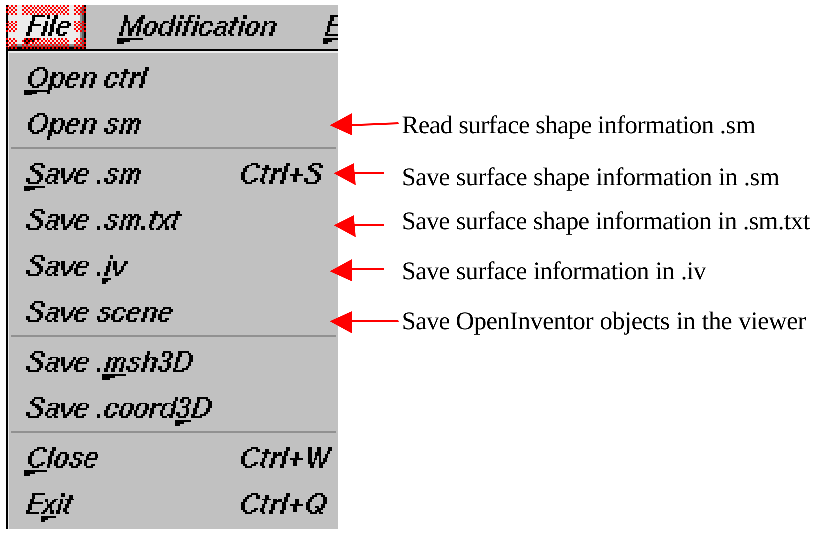

File
Modification
Open ctrl
Open sm
Save .sm Ctrl+S
Save .sm.txt
Save .iv
Save scene
Save .msh3D
Save .coord3D
Close Ctrl+W
Exit Ctrl+Q
Read surface shape information .sm
Save surface shape information in .sm
Save surface shape information in .sm.txt
Save surface information in .iv
Save OpenInventor objects in the viewer

Figure B-9: Face cloning: Callback functions on File CascadeButton in 3D-shape manager

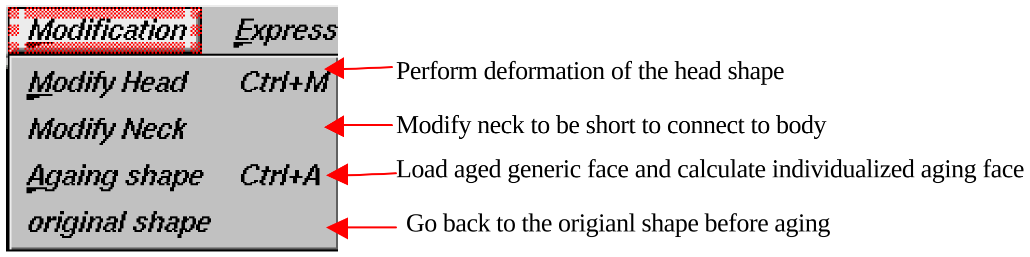

Modification
Express
Modify Head Ctrl+M
Modify Neck
Againg shape Ctrl+A
original shape
Perform deformation of the head shape
Modify neck to be short to connect to body
Load aged generic face and calculate individualized aging face
Go back to the origianl shape before aging

Figure B-10: Face cloning: Callback functions on Modification CascadeButton in 3D-shape manager

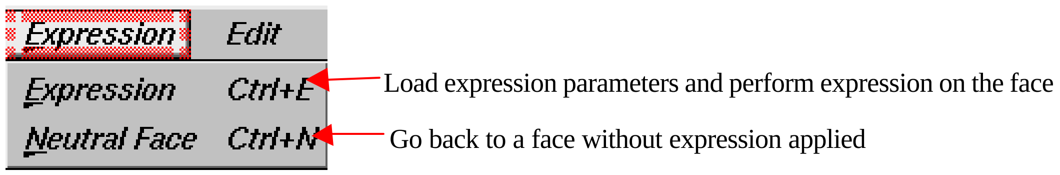

Expression
Edit
Expression Ctrl+E
Neutral Face Ctrl+N
Load expression parameters and perform expression on the face
Go back to a face without expression applied

Figure B-11: Face cloning: Callback functions on Expression CascadeButton in 3D-shape manager

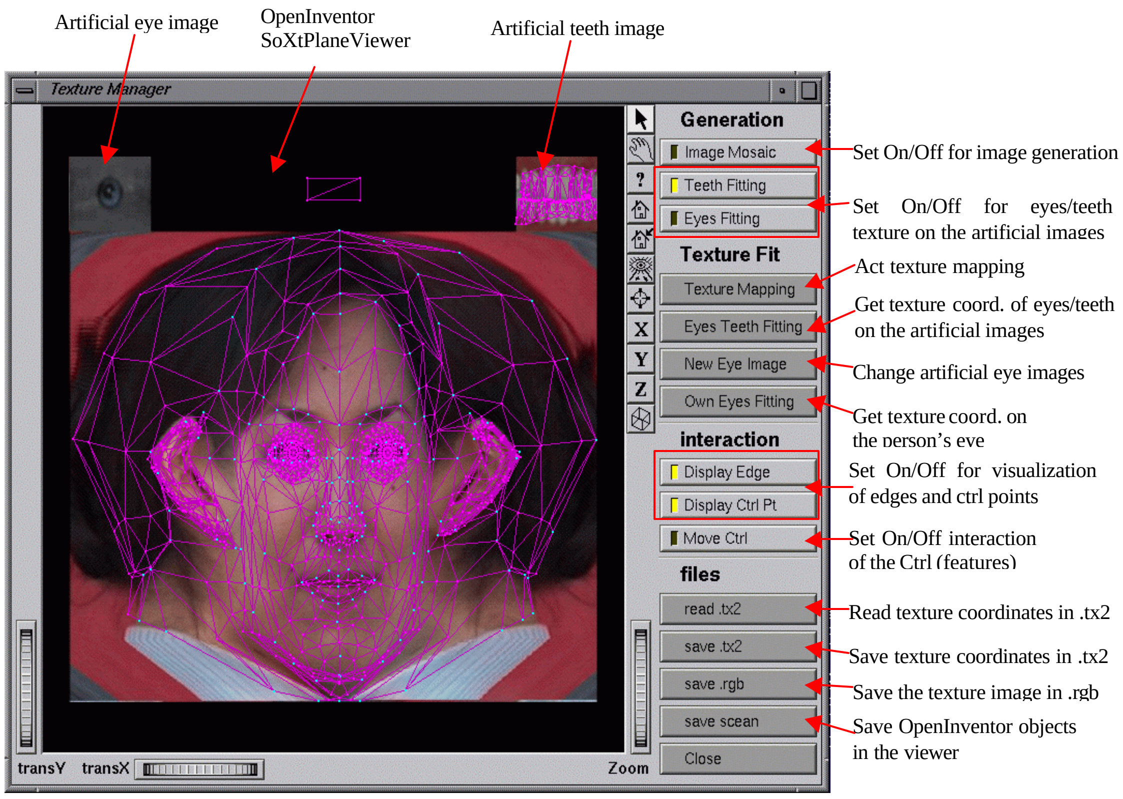

Artificial eye image
OpenInventor SoXtPlaneViewer
Artificial teeth image
Texture Manager
Generation
Image Mosaic
Teeth Fitting
Eyes Fitting
Texture Fit
Texture Mapping
Eyes Teeth Fitting
New Eye Image
Own Eyes Fitting
interaction
Display Edge
Display Ctrl Pt
Move Ctrl
files
read .tx2
save .tx2
save .rgb
save scean
Close
transY
transX
Zoom
Set On/Off for image generation
Set On/Off for eyes/teeth texture on the artificial images
Act texture mapping
Get texture coord. of eyes/teeth on the artificial images
Change artificial eye images
Get texture coord. on the person's eye
Set On/Off for visualization of edges and ctrl points
Set On/Off interaction of the Ctrl (features)
Read texture coordinates in .tx2
Save texture coordinates in .tx2
Save the texture image in .rgb
Save OpenInventor objects in the viewer

Figure B-12: Face cloning: 2D texture manager

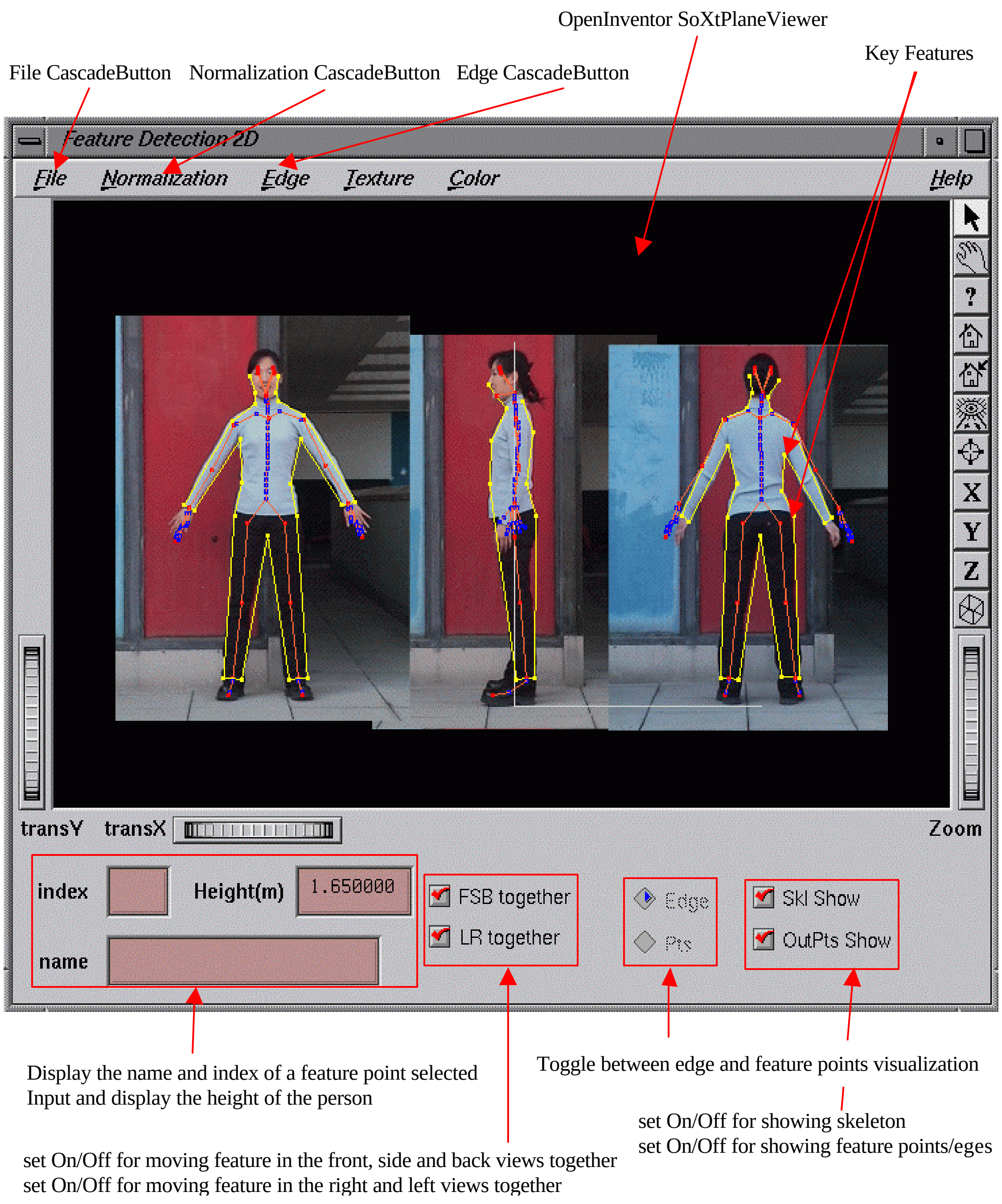

OpenInventor SoXtPlaneViewer
Key Features
File CascadeButton
Normalization CascadeButton
Edge CascadeButton
Feature Detection 2D
File
Normalization
Edge
Texture
Color
Help
transY
transX
Zoom
index
Height(m)
1.650000
name
FSB together
LR together
Edge
Pts
Skl Show
OutPts Show
Display the name and index of a feature point selected
Input and display the height of the person
Toggle between edge and feature points visualization
set On/Off for showing skeleton
set On/Off for showing feature points/eges
set On/Off for moving feature in the front, side and back views together
set On/Off for moving feature in the right and left views together

Figure B-13: Body cloning: 2D viewer

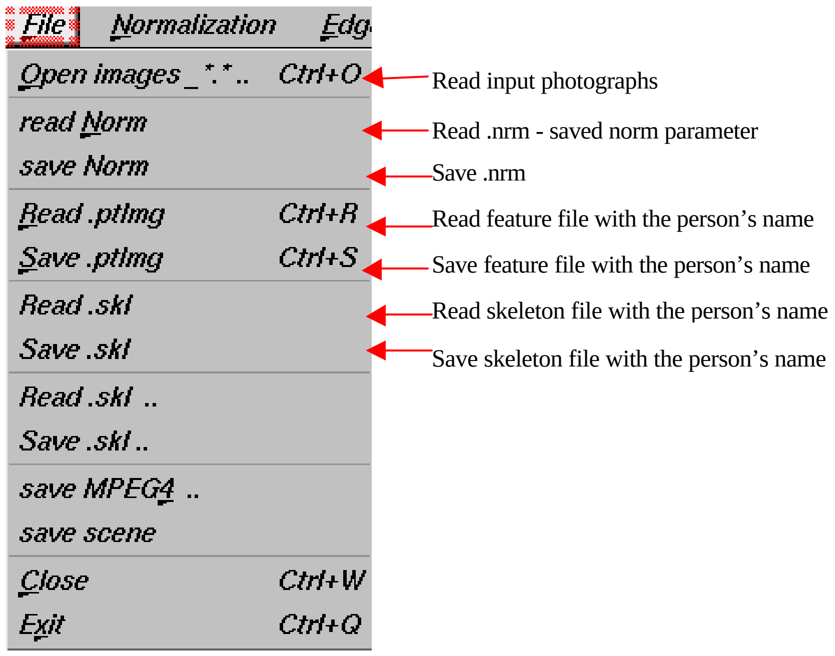


Figure B-14: Body cloning: Callback functions on File CascadeButton on 2D viewer

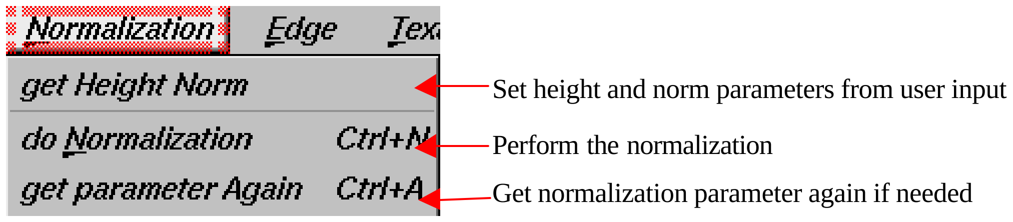

Normalization
Edge
Tex
get Height Norm
do Normalization
Ctrl+N
get parameter Again
Ctrl+A
Set height and norm parameters from user input
Perform the normalization
Get normalization parameter again if needed

Figure B-15: Body cloning: Callback functions on Normalization CascadeButton on 2D viewer

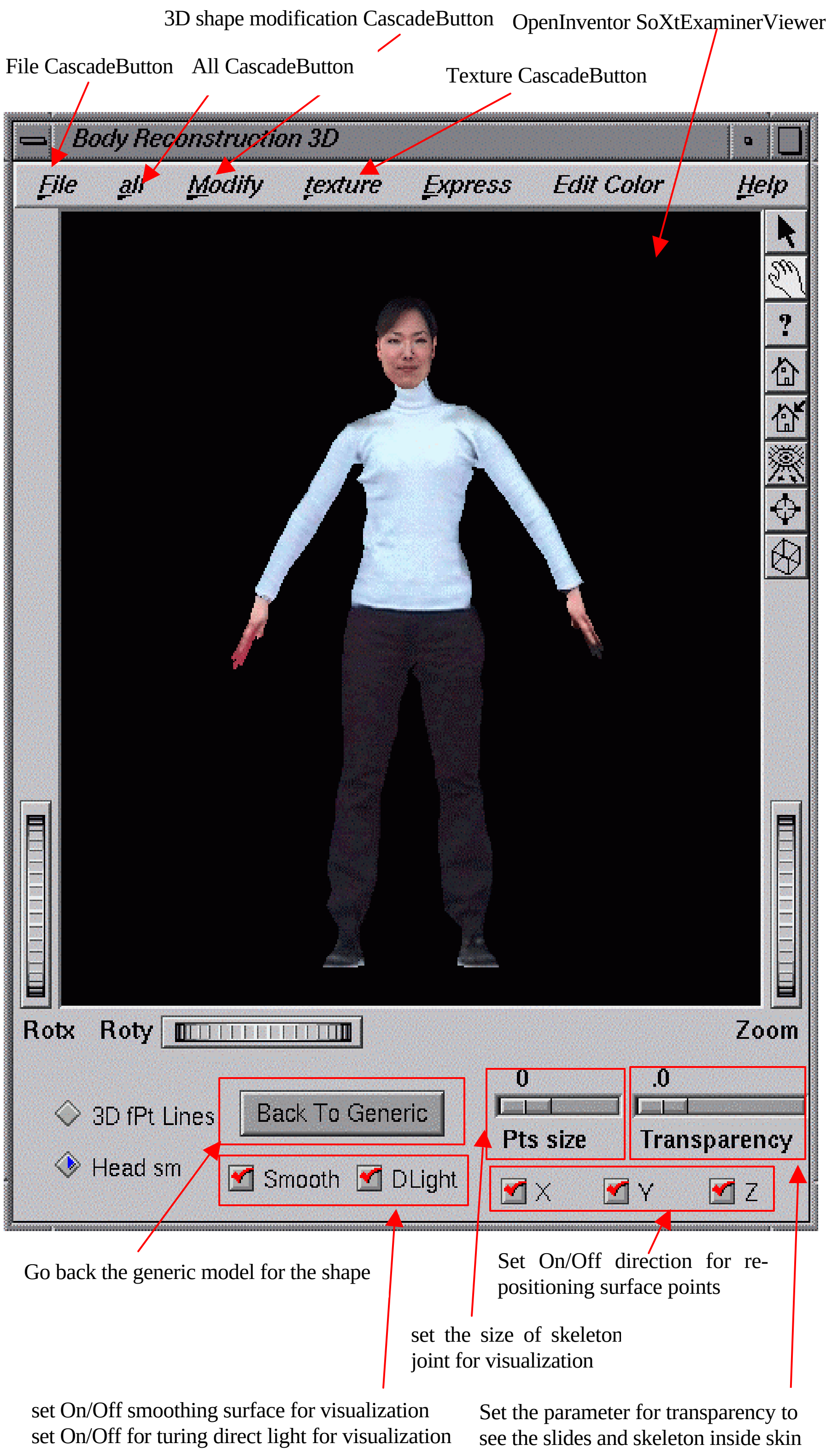
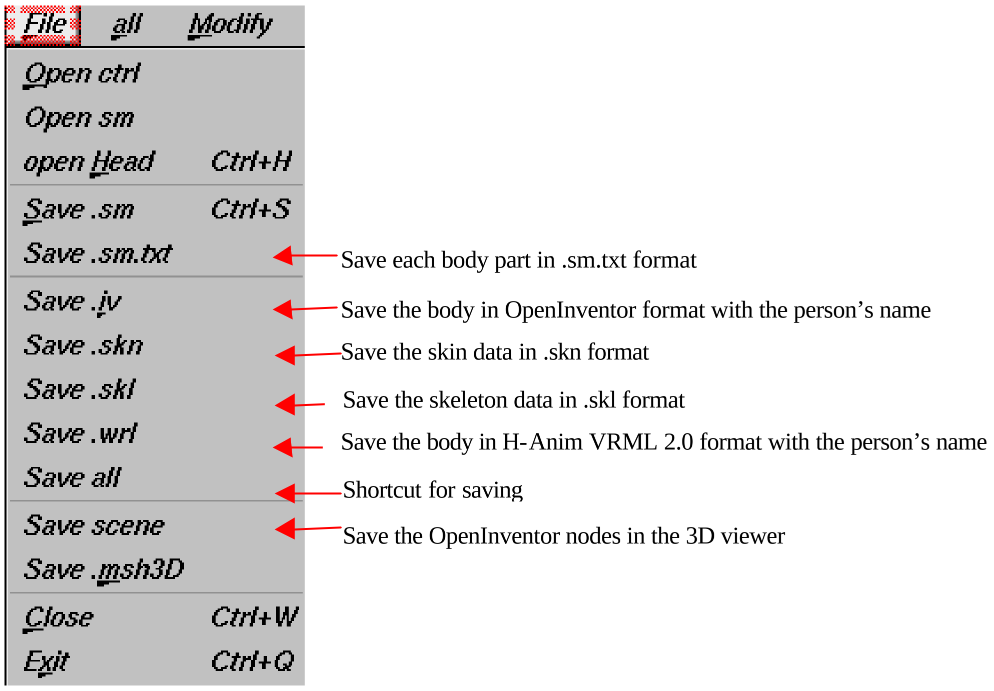
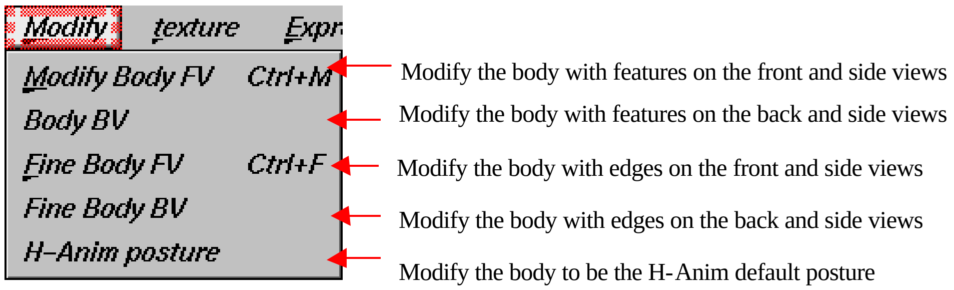

3D shape modification CascadeButton
OpenInventor SoXtExaminerViewer
File CascadeButton
All CascadeButton
Texture CascadeButton
Body Reconstruction 3D
File
all
Modify
texture
Express
Edit Color
Help
Rotx
Roty
Zoom
3D fPt Lines
Head sm
Back To Generic
Smooth
DLight
0
Pts size
.0
Transparency
X
Y
Z
Go back the generic model for the shape
Set On/Off direction for re-positioning surface points
set the size of skeleton joint for visualization
set On/Off smoothing surface for visualization
set On/Off for turing direct light for visualization
Set the parameter for transparency to see the slides and skeleton inside skin

Figure B-16: Body cloning: 3D viewer



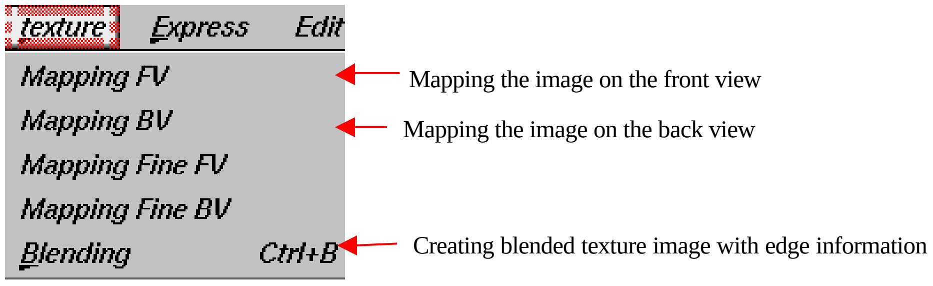
Save each body part in .sm.txt format
Save the body in OpenInventor format with the person’s name
Save the skin data in .skn format
Save the skeleton data in .skl format
Save the body in H-Anim VRML 2.0 format with the person’s name
Shortcut for saving
Save the OpenInventor nodes in the 3D viewer

Figure B-17: Body cloning: Callback functions on File CascadeButton on 3D viewer



Modify the body with features on the front and side views
Modify the body with features on the back and side views
Modify the body with edges on the front and side views
Modify the body with edges on the back and side views
Modify the body to be the H-Anim default posture

Figure B-18: Body cloning: Callback functions on Modification CascadeButton on 3D viewer



Mapping the image on the front view
Mapping the image on the back view
Creating blended texture image with edge information

Figure B-19: Body cloning: Callback functions on Texture CascadeButton on 3D viewer

Appendix C Sibson Coordinates

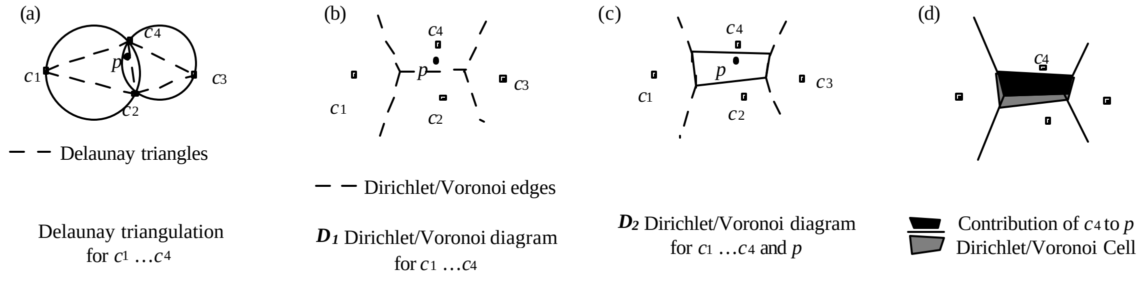


Figure C-20: Sibson coordinates. For simplification, the example will be described in 2D. The extension to 3D is straightforward.

Here we shortly describe Sibson coordinates, and how they are applied in data interpolation. Given a set of points, any location inside the convex hull of the set can be expressed as a linear combination of its Delaunay neighbors in the set. The coefficients of the linear relation are non-null, and their sum is equal to one. They are known as the Sibson coordinates from Sibson's work in [Sibson 80].

We consider the Dirichlet diagram D_1 (see Figure C-20 (b)) associated to a given set of points and the Dirichlet diagram D_2 (see Figure C-20 (c)) associated to the same set plus a arbitrary point P inside the convex hull of the set. Some Dirichlet tiles of D_1 overlaps with ***p***'s Dirichlet tile. Let c_i be a point of the set whose Dirichlet tile T_1 in D_1 overlaps with ***p***'s tile T_2 in D_2 (see Figure C-20 (d)). ***p***'s Sibson coordinate u_i relative to c_i is given by the ratio of the area of the intersection of T_1 and T_2 by the total area of T_2.

Given a set of control points ***C***, and a point ***p***, ***p*** can be expressed as a function of a subset of ***C*** that influences ***p***, $C_n = \{c_i\}$ with $i = 0, \ldots, n$, using Sibson coordinates, so that

$$p = \sum_{i=0}^{n} ui\, c_i$$

where $\sum_{i=0}^{n} ui = 1$ and $u_i > 1$ for any i in $[0,n]$.

A control point is defined at the same position as the surface point, so that any displacement applied to the control point will be completely applied to the surface point. This relation can be expressed by assigning a Sibson coordinate of 1.0 to the surface point relative to its associated control point.

Delaunay circles Figure C-20 (a) are used to determine which control points influence the point. For example, it is easy calculation to know that ***p*** belongs to two Delaunay circles $O_1\{c_1, c_2, c_4\}$ and $O_2\{c_2, c_3, c_4\}$.

Bibliography

[Addleman 97] Addleman S., "Whole-Body 3D Scanner and Scan Data Report", in Three Dimensional Image Capture, SPIE, pp. 2-5, 1997.

[Akimoto 93] Akimoto T., Suenaga Y. and Richard S. W., "Automatic Creation of 3D Facial Models", IEEE Computer Graphics & Applications, September, IEEE Computer Society Press, 1993.

[Arad 94] Arad N., Dyn N, Reisfeld D, Yeshurun Y., "Image warping by radial basis functions: applications to facial expressions", Graphical Models and Image Processing, Academic Press, Vol. 56, No. 2, pp. 161-172, Mar 1994.

[Arai 96] Arai K., Kurihara T., Anjyo K., "Bilinear interpolation for facial expression and metamorphosis in real-time animation", Visual Computer, Springer, Vol. 12, No. 3, 1996.

[Aurenhammer 91] Aurenhammer F., "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure", ACM Computing Survey, 23, 3, September 1991.

[Babski 99] Babski C., Thalmann D., "A Seamless Shape For HANIM Compliant Bodies", Proc. VRML 99, ACM Press, pp. 21-28, 1999.

[Babski 00] Babski C., Thalmann D., "Realtime animation and Motion Capture in Web Human Director(WHD), Proc. Web3D & VRML2000 Symposium, 2000.

[Baker 94] Baker E. and Seltzer M., "Evolving Line Drawings", In the Proceedings of Graphics Interface, Morgan Kaufmann Publishers, pp. 91-100, 1994.

[Ballard 96] Ballard D. H. and Brown C. M., Computer Vision, pp. 131-147, Prentice-Hall InC., 1982.

[Barrus 96] Barrus J. W., Waters R. C., Anderson D. B., "Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments", Proceedings of IEEE VRAIS, IEEE Computer Society Press, 1996.

[Beier 92] Beier T., Neely S., "Feature-based image metamorphosis", In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 35-42, 1992.

[Benoît 92] Benoît C., Lallouache, T., Mohamadi, T., & Abry, C., "A set of French visemes for visual speech synthesis", G Bailly & C. Benoît, (Eds.), Talking machines: Theories, Models, and Designs, Amsterdam: North Holland, Elsevier, 485-504, 1992.

[Beylot 96] Beylot P., Gingins P., Kalra P., Magnenat-Thalmann N., Maurel W., Thalmann D., and Fasel F., "3D Interactive Topological Modeling using Visible Human Dataset", Computer Graphics Forum, Blackwell publisher, 15(3): 33-44, 1996.

[Blake 94] Blake A., Isard M., "3D position, attitude and shape input using video tracking of hands and lips", Computer Graphics (Proc. SIGGRAPH'94), ACM Press, 28:185-192, July 1994.

[Blanz 99] Blanz V. and Vetter T. "A Morphable Model for the Synthesis of 3D Faces", Computer Graphics (Proc. SIGGRAPH'99), ACM Press, pp. 187-194, 1999.

[Burt 83] Burt P. J. and Andelson E. H., "A Multiresolution Spline With Application to Image Mosaics", ACM Transactions on Graphics, ACM Press, 2(4): 217-236, Oct., 1983.

[Carlsson 93] Carlsson C., Hagsand O., "DIVE - a Multi-User Virtual Reality System", Proceedings of IEEE VRAIS '93, Seattle, Washington, IEEE Computer Society Press, 1993.

[Cohen 93] Cohen M.M. and Massaro D.W., "Modeling Coarticulation in Synthetic Visual Speech", Models and Techniques in Computer Animation, N. M. Thalmann & D. Thalmann (Eds.), Tokyo: Springer-Verlag, pp. 139-156, 1993.

[Coleman 82] Coleman EN, Jain R, Obtaining 3-Dimensional Shape of Textured and Specular Surfaces Using Four-Source photometry, IEEE Computer Graphics and Image Processing, IEEE Computer Society Press,18:309-328, 1982.

[Daanen 97] Daanen H., Talory S.E., Brunsman M.A. and Nurre J.H., “Absolute accuracy of the Cyberware WB4 whole body scanner”, in Three-Dimensional Image Capture, SPIE, pp. 6-12, 1997.

[DeCarlo 96] DeCarlo D. and Metaxas D., “The Integration of Optical Flow and Deformable Models with Applications to Human Face Shape and Motion Estimation”, Proc. CVPR’96, IEEE Computer Society Press, pp. 231-238, 1996.

[DeCarlo 98] DeCarlo D., Metaxas D. and Stone M., “An Anthropometric Face Model using Variational Techniques”, In Computer Graphics (Proc. SIGGRAPH '98), ACM Press, pp. 67-74, 1998.

[DeRose 88] DeRose T.D., “Composing Bézier Simplexes”, ACM Transactions on Graphics, ACM Press, 7(3), pp. 198-221, 1988.

[DeRose 98] DeRose T., Kass M., Truong Tien, “Subdivision Surfaces in Character Animation”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 85-94, 1998.

[Doenges 97] Doenges P., Lavagetto F., Ostermann J., Pandzic I.S., Petajan E., “MPEG-4: Audio/video and synthetic graphics/audio for mixed media”, Signal Processing: Image Communication, Elsevier Sceince, 9 (4) (1997) pp. 433-463, 1997.

[Ekman 78] Ekman P., Friesen W.V., “Facial Action Coding System, Investigator’s Guide Part 2”, Sonsulting Psychologists Press Inc, 1978.

[Escher 97] Escher M., Magnenat-Thalmann N., “Automatic 3D Cloning and Real-Time Animation of a Human Face” Akimoto, Proc. Computer Animation, IEEE Computer Society Press, pp. 58-66, 1997.

[Escher 98] Escher M., Panzic I., Magnenat-Thalmann N., “Facial Deformation from MPEG-4”, Proc. Computer Animation'98, IEEE Computer Society Press, 1998.

[Farin 90] Farin G., “Surface Over Dirichlet Tessellations”, Computer Aided Geometric Design, Elsevier Science, 7, pp. 281-292, North-Holland, 1990.

[Fua 96] Fua P. and Leclerc Y.G., “Taking Advantage of Image-Based and Geometry-Based Constraints to Recover 3-D Surfaces”, Computer Vision and Image Understanding, Academic Press, 64(1): 111-127, Jul., 1996.

[Fua 00] Fua P., “Regularized Bundle-Adjustment to Model Heads from Image Sequences without Calibration Data”, International Journal of Computer Vision, Kluwer Publisher, In Press.

[Gao 98] Gao P., Sederberg TW, “A work minimization approach to image morphing”, Visual Computer, International Journal of Computer Graphics, Springer, Vol. 14, No. 8/9, pp. 390-400, 1998.

[Gu 98] Gu J., Chang T., Gopalsamy S., and Shen H., “A 3D Reconstruction System for Human Body Modeling”, In Modelling and Motion Capture Techniques for Virtual Environments (Proc. CAPTECH'98), (Springer LNAI LNCS Press), pp. 229-241, 1998.

[Guenter 98] Guenter B., Grimm C., Wood D., Malvar H., Pighin F., “Making Faces”, Computer Graphics (Proc. SIGGRAPH'98), ACM Press, pp. 55-66, 1998.

[Hilton 99] Hilton A., Beresford D., Gentils T., Smith R. and Sun W., “Virtual People: Capturing human models to populate virtual worlds”. In Computer Animation (Proc. Computer Animation'99), IEEE Computer Society Press, pp. 174-185, 1999.

[Ip 96] Ip H. H.S., Yin L., “Constructing a 3D individual head model from two orthogonal views”, The Visual Computer, Springer, 12:254-266, 1996.

[Kakadiaris 95] Kakadiaris I. A. and Metaxas D., “3D Human Body Acquisition from Multiple views”, in Proc. of the Fifth ICCV, IEEE Computer Society Press, pp. 618-623, 1995.

[Kakadiaris 96] Kakadiaris I. A. and Metaxas D., “Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection”, In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society Press, pp. 81-87, San Francisco, CA, June 18-20 1996.

[Kalra 91] Kalra P, Mangili A, Magnenat-Thalmann N, Thalmann D, “SMILE: A Multilayered Facial Animation System”, Proc IFIP WG 5.10, Tokyo, Japan (Ed Kunii TL), Springer, pp. 189-198, 1991.

[Kalra 92] Kalra P., Mangili A., Magnenat-Thalmann N., Thalmann D., Simulation of Muscle Actions using Rational Free Form Deformations, Proc Eurographics '92, Computer Graphics Forum, Blackwell publisher, Vol. 2, No. 3, pp. 59-69, 1992.

[Kass 88] Kass M., Witkin A., and Terzopoulos D., “Snakes: Active Contour Models”, International Journal of Computer Vision, Kluwer Publisher, pp. 321-331, 1988.

[Kent 92] Kent J., Carlson W., Parent R., “Shape Transformation for Polygon Objects”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 47-54, 1992.

[Koch 96] Koch R. M., Gross M. H., Carls F. R., von Büren D. F., Fankhauser G., Parish Y. I. H, “Simulating Facial Surgery Using Finite Element Models”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 421-428, 1996.

[Koenen 97] Koenen R., Pereira F., Chiariglione L., “MPEG-4: Context and Objectives”, Image Communication Journal, Special Issue on MPEG-4, Elsevier Sceince, Vol. 9, No. 4, May 1997.

[Kouadio 98] Kouadio C., Poulin P., Lachapelle P., “Real-Time Facial Animation based upon a Bank of 3D Facial Expression”', Proc. CA98 (International Conference on Computer Animation), IEEE Computer Society Press, pp128 - 136, 1998.

[Kurihara 91] Kurihara T. and Arai K., “A Transformation Method for Modeling and Animation of the Human Face from photographs”, Proc. Computer Animation'91, Springer-Verlag, Tokyo, pp. 45-58, 1991.

[Lai 95] Lai K.F. and Chin R.T., “Deformable contours Modeling and extraction”, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press, vol. 17, no.11, pp. 1084-1090, 1995.

[LeBlanc 91] LeBlanc A., Kalra, P., Magnenat-Thalmann, N. and Thalmann, D. “Sculpting with the 'Ball & Mouse' Metaphor”, Proc. Graphics Interface'91, Calgary, Canada, Morgan Kaufmann Publishers, pp. 152-9, 1991.

[LeeS 96] Lee S.-Y., Chaw K-Y, Hahn J., Shin S.-Y., “Image morphing using deformation techniques”, Journal Visualization Computer Animation, John Wiley & Sons Ltd., 7:3-23, 1996.

[LeeS 95] Lee S.-Y., Chwa K.-Y., Shin S.-Y., Wolberg G., Image metamorphosis using Snakes and Free-Form deformations, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 439-448, 1995.

[LeeY 95] Lee Y., Terzopoulos D., and Waters K., “Realistic Modeling for Facial Animation”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 55-62, 1995.

[Lerios 95] Lerios A., Chase D. Garfinkle, Marc Levoy, “Feature-Based Volume Metamorphosis”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 449-456, 1995.

[Litwinowicz 94] Litwinowicz P., Williams L., “Animating images with drawings”, In Computer Graphics (Proc. SIGGRAPH'94), ACM Press, pp. 409-412, 1994.

[Macedonia 94] Macedonia M.R., Zyda M.J., Pratt D.R., Barham P.T., Zestwitz, “NPSNET: A Network Software Architecture for Large-Scale Virtual Environments”, Presence: Teleoperators and Virtual Environments, MIT Press, Vol. 3, No. 4, 1994.

[Magnenat-Thalmann 87] Magnenat-Thalmann N., Thalmann D., “The direction of Synthetic Actors in the film Rendez-vous à Montrèal”, IEEE Computer Graphics and Applications, IEEE Computer Society Press, 7(12): 9-19, 1987.

[Magnenat-Thalmann 89] Magnenat-Thalmann N., Angelis M.De, Hong T., Thalmann D., “Design Transformation and Animation of Human Faces”, Visual Computer, Springer, Vol. 5, No. 1-2, pp. 32-39, 1989.

[Magnenat-Thalmann 95] Magnenat-Thalmann N., Pandzic I. S., Moussaly J.-C., Huang Z., Shen J., “The Making of Xian Terra-cotta Soldiers”, Computer Graphics: Developments in Virtual Environments, (Academic Press) (ISBN 0-12-227741-4), 1995.

[Magnenat-Thalmann 90] Magnenat-Thalmann N., Thalmann D., Synthetic actors in computer-generated 3D films, Springer-Verlag, Tokyo. 1990.

[Magnenat-Thalmann 98] Magnenat-Thalmann N., Kalra P., Escher M., “Face to Virtual Face”, Proc. of the IEEE, IEEE Computer Society Press, Vol.86, No. 5, May, 1998.

[Moccozet 97] Moccozet L., Magnenat-Thalmann N., “Dirichlet Free-Form Deformations and their Application to Hand Simulation”, Proc. Computer Animation'97, IEEE Computer Society Press, pp. 93-102, 1997.

[Nagel 98] Nagel B., Wingbermühle J., Weik S., Liedtke C.-E., “Automated Modelling of Real Human Faces for 3D Animation”, Proceedings ICPR, Brisbane, Australia, 1998.

[Pandzic 94] Pandzic I., Kalra P., Magnenat-Thalmann N., Thalmann D., “Real-time facial interaction”, In Displays, Butterworth-Heinemann, 15 (3): 157-163, 1994.

[Pandzic 96] Pandzic I., Capin T., Magnenat-Thalmann N., Thalmann D., 'Towards Natural Communication in Networked Collaborative Virtual Environments', In Proceedings of FIVE '96, Pisa, Italy, 1996.

[Parke 82] Parke F. “Computer generated animation of faces”, In ACM National Conference, ACM Press, pp. 451-457, 1972

[Pearce 86] Pearce A., Wyvill B., Wyvill G., Hill D., “Speech and Expression: A Computer Solution to Face Animation”, Proc Graphics Interface '86, Morgan Kaufmann Publishers, pp. 136-140, 1986.

[Pighin 98] Pighin F., Hecker J., Lischinski D., Szeliski R., Salesin D. H., “Synthesizing Realistic Facial Expressions from photographs”, In Computer Graphics (Proc. SIGGRAPH'98), ACM Press, pp. 75-84, 1998.

[Plänkers 99] Plänkers R., Fua P., D'Apuzzo N., “Automated Body Modeling from Video Sequences” in Proc. IEEE International Workshop on Modelling People (mPeople), IEEE Computer Society Press, Corfu, Greece, September, 1999

[Proesmans 97] Proesmans M., Van Gool L. “Reading between the lines - a method for extracting dynamic 3D with texture”, In Proceedings of VRST’97, ACM Press, pp. 95-102, 1997.

[Saji 92] Saji H., Hioki H., Shinagawa Y., Yoshida K. and Kunii T., “Extracting of 3D Shapes from the moving human face using lighting Switch photometry”, Creating and Animating the virtual world, Springer-Verlag, Tokyo, pp. 69-86, 1992.

[Sannier 97] Sannier G., Magnenat-Thalmann N., “A User-Friendly Texture-Fitting Methodology for Virtual Humans”, Computer Graphics International'97, IEEE Computer Society Press, 1997.

[Sannier 99] Sannier G., Balcisoy S., Magnenat-Thalmann N., Thalmann D., “VHD: A System for Directing Real-Time Virtual Actors”, The visual Computer, Springer, 1999, Vol.15, No 7/8, 1999, pp. 320-329.

[Sederberg 86] Sederberg T. W., Parry S. R., “Free-Form Deformation of Solid Geometric Models”, Computer Graphics (Proc. SIGGRAPH'86), ACM Press, pp. 151-160, 1986.

[Sibson 80] Sibson R., “A Vector Identity for the Dirichlet Tessellation”, Math. Proc. Cambridge Philos. Soc., 87, pp. 151-155, 1980.

[SNHC1 98] SNHC, “INFORMATION TECHNOLOGY-GENERIC CODING OF AUDIO-VISUAL OBJECTS Part 2: Visual”, ISO/IEC 14496-2, Final Draft of International Standard, Version of: 13, November 1998, ISO/IEC JTC1/SC29/WG11 N2502a, Atlantic City, October 1998.

[SNHC2 98] SNHC, “Text for ISO/IEC FDIS 14496-1 Systems (2nd draft)”, ISO/IEC JTC1/SC29/WG11 N2501, November 1998.

[Terzopoulos 90] Terzopoulos D., Waters K., “Physically-Based Facial Modeling and Animation”, Journal of Visualization and Computer Animation, John Wiley & Sons Ltd., Vol. 1, pp. 73-80, 1990.

[Terzopoulos 93] Terzopoulos D., Waters K., “Analysis and synthesis of facial image sequences using physical and anatomical models”, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press, 15(6): 569-579, 1993.

[Viaud 92] Viaud M., Yahia H., “Facial Animation with Wrinkles”, In Proc. 3rd Workshop on Animation, Eurographics'92, Cambridge, Springer Verlag, 1992.

[Weik 98] Weik S., Wingbermühle J., Niem W., “Automatic Creation of Flexible Antropomorphic Models for 3D Videoconferencing”, Proceedings of Computer Graphics International CGI, IEEE Computer Society Press, June 22-26, Hanover, Germany, 1998

[Williams 90] Williams L., “Performance-Driven Facial Animation”, In Computer Graphics (Proc. SIGGRAPH), ACM Press, pp. 235-242, 1990.

[Wolberg 90] Wolberg G., “Digital image warping”, IEEE Computer Society Press, Los Anamitos, Calif, 1990.

[Wolberg 98] Wolberg G., “Image morphing: a survey”, Visual Computer, International Journal of Computer Graphics, Springer, Vol. 14, No. 8/9, pp. 360-372, 1998.

[Wu 94] Wu Y., Magnenat-Thalmann N. and Thalmann D., “A Plastic-Visco-Elastic Model for Wrinkles In Facial Animation And Skin Aging”, Proc. Pacific Conference '94, pp. 201-213, World Scientific, 1994.

[Wu 97] Wu Y., Kalra P. and Magnenat-Thalmann N., “Physically-based Wrinkle Simulation & Skin Rendering”, In Proc. Eurographics Workshop on Animation and Simulation, Springer-Verlag, pp. 69-79, 1997.

[Zheng 94] Zheng J.Y., “Acquiring 3-d models from sequences of contours”, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press,16(2): 163-178, February 1994.

[http:cyberscan] http://www.viewpoint/com/freestuff/cyberscan

[http:cyberwareBody] http://www.cyberware.com/wb-vrml/

[http:festival] http://www.cstr.ed.ac.uk/projects/festival

[http:H-Anim] http://www.H-Anim.org

[http:mpeg-4] http://www.cselt.it/leonardo/paper/mpeg-4/mpeg-4.htm

[http:turing] http://www.turing.gla.ac.uk/turing/copyrigh.htm

Related publications by the author

[MMM 97] Lee W., Kalra P., Magnenat-Thalmann N., "Model Based Face Reconstruction for Animation", Proc. MMM'97, World Scientific Press, Singapore, pp.323-338, 1997.

[VW 98] Lee W., Lee E., Magnenat-Thalmann N., "Real Face Communication in a Virtual World", Proc. Virtual Worlds 98, Springer LNAI Press, Paris, pp.1-13, 1998.

[3IA 98] Lee W., Magnenat-Thalmann N., "From Real Faces To Virtual Faces: Problems and Solutions", Proc. 3IA'98 (International Conference on Computer Graphics and Artificial Intelligence), Limoges (FRANCE), Dimitri PLEMENOS, pp.5-19, 1998.

[CAPTECH 98] Lee W., Magnenat-Thalmann N., "Head Modeling from Pictures and Morphing in 3D with Image Metamorphosis based on triangulation", Proc. Captech98 (Modelling and Motion Capture Techniques for Virtual Environments), Springer LNAI LNCS Press, Geneva, pp.254-267, 1998.

[pECCV 98] Lee W., Magnenat-Thalmann N., "Fast Head Modeling for Animation", Post-ECCV workshop on Facial Image Analysis and Recognition Technology, University of Freiburg, Germany, June 6, 1998.

[VR 99] Lee W., Wu Y., Magnenat-Thalmann N., "Cloning and Aging in a VR Family", Proc. IEEE VR'99 (Virtual Reality), IEEE Computer Society Press, Houston, Texas, March 13-17, 1999.

[CA 99] Lee W., Escher M., Sannier G., Magnenat-Thalmann N., "MPEG-4 Compatible Faces from orthogonal photos", Proc. CA99(International Conference on Computer Animation), IEEE Computer Society Press, Geneva, Switzerland. May 26-29, pp.186-194, 1999.

[WADS 99] Lee W., Beylot P., Sankoff D., Magnenat-Thalmann N., "Generating 3D Virtual Populations from Pictures of a Few Individuals", In Proc. WADS99(1999 Workshop on Algorithms And Data Structures), Springer LNCS proceedings, Vancouver, Britich Columbia, Canada, August 12-14, pp. 134-144, 1999.

[mPEOPLE 99] Lee W., Magnenat-Thalmann N., "Generating a Population of Animated faces from Pictures" In Proc. IEEE International Workshop on Modelling People (ICCV'99 Workshop mPeople), IEEE Computer Society Press, Corfu Ho liday Palace, Kerkyra (Corfu), Greece, September 20, 1999

[i3 99] Lee W., Goto T., Magnenat-Thalmann N., "Cloning, Morphing, then Tracking Real Emotions", In Proc. i3 annual conference, Sienna, The Human Communication Research Centre, Italy, October 20-22, pp. 111– 115, 1999

[PRESENCE 00] Pandzic I., Babski C., Capin T., Lee W., Magnenat-Thalmann N., Soraia Raupp Musse, Laurent Moccozet, Heywon Seo and Daniel Thalmann, "Simulating Virtual Humans in Networked Virtual Environments", sibmitted to Presence, MIT Press, 2000.

[EG 00] Lee W., Gu J., Magnenat-Thalmann N., "Generating Animatable 3D Virtual Humans from Photographs", accepted in Eurographics 2000, Volume 19, Number 3, Computer Graphics Forum, Blackwell publisher, 2000.

[VW 00] Furakawa T., Gu J., Lee W., Magnenat-Thalmann N., "3D Clothes Modeling from Photo Cloned Human Body", accepted in Virtual Worlds 2000, Springer, 2000.

[IVC 00] Lee W., Magnenat-Thalmann N., "Fast Head Modeling for Animation", Journal Image and Vision Computing, Volume 18, Number 4, pp.355-364, Elsevier Sceince, 1 March, 2000.