3D emotional rendering and animation models

Jing Huang

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0008

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « SIGNAL et IMAGES »

présentée et soutenue publiquement par

Jing HUANG

le 26 Févier 2013

Modèles de Rendu et Animation Émotionnelle en 3D

Directeur de thèse :
Catherine Pelachaud et **Tamy Boubekeur**

Jury

M. Ronan Boulic, Maître d'Enseignement et de Recherche (HDR), École Polytechnique Fédérale de Lausanne	Rapporteur
M. Venceslas Biri, Maître de Conférences (HDR), Université Marne-la-Vallée	Rapporteur
M. Christian Jacquemin, Professeur, Université Paris-Sud	Examinateur
Mme. Isabelle Bloch, Professeur, Télécom ParisTech	Examinateur
Mme. Catherine Pelachaud, Directeur de recherche, Télécom ParisTech	Directeur de thèse
M. Tamy Boubekeur, Maître de Conférences (HDR), Télécom ParisTech	Directeur de thèse

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

论文致谢 Acknowledgements

首先，我怀着无比崇敬的心情感谢巴黎高科电信工程师学校以及我的导师凯瑟琳·佩拉苏(Catherine Pelachaud)教授和塔米·布拜歌(Tamy Boubekeur)教授。在他们辛勤的指导和无微不至的关怀下，我圆满完成了3年博士研究生学业。他们广博精深的知识面以及严谨的治学态度带领我深入三维图像动画学领域。能够有幸成为他们的学生是我的骄傲。

同样的，我非常荣幸能够成为电脑图形学组和多媒体组两个实验室中的一员，两个实验室的同事都给予我极大的帮助。不仅仅在学习上，更加在生活上照顾关心我，让远离故乡的我，感受到家的温馨。

最后我要感谢我的家人。感谢我的爸爸黄建中，妈妈蒋莲萍给予了我极大的支持，让我可以实现理想来到法兰西，帮助我完成六年的留学生涯。感谢我的妻子胡彦彬一直在背后默默的付出。还有我即将出生的宝宝，给予我心灵的寄托。你们对我无私的爱和包容是我努力和坚持的动力。

My profuse thanks go to my supervisors Catherine Pelachaud and Tamy Boubekeur for their expertise and support. They spent 3 years on encouraging me, helping me and guiding me into a new multi-dimensional space. I'm so proud of being their Ph.D student during these 3 years.

I would like to thank all my colleagues also, in both CG lab and MM lab (Greta team) that support me for my thesis:

- Rendering part: Dr. Elmar Eisemann, Dr. Tobias Ritschel, Dr. Bert Buchholz, Dr. Matthias Hollnder, Beibei Wang,
- Geometry part: Dr. Julien Tierny, Dr. Jean-Marc Thiery, Dr. Guillaume Vialaneix, Dr. Noura Faraj, St é phane Calderon, Thierry Guillemot, Emilie Guy, Leila Schemali,
- Emotional Expression part: Dr. Magalie Ochs, Dr. Radoslaw Niewiadomski, Dr. Ken Prepin, Dr. Elisabetta Bevacqua, Dr. Etienne de Sevin, Dr. Sylwia Julia Hyniewska, Dr. Quoc Anh Le, Andre-Marie Pez, Pierre Philippe, Nesrine Fourati, Brian Ravenet, Florian Pecune, Mathieu Chollet, Nadine Glas, Yu Ding, Yafei Xing.

To My Family: my father Jianzhong Huang, my mother Lianping Jiang, my wife Yanbin Hu, my future baby, also Lucky and Alpha.

Resumé Français

L'animation et le rendu sont deux domaines de recherche importants dans l'informatique graphique. L'animation est l'étude des mouvements des objets virtuels et de leurs transformations dans le temps dans un monde virtuel. Il est possible de créer des mouvements crédibles en utilisant des simulations physiques et des modèles fondés sur des mécanismes contrôlables. Le rendu produit les images finales en convertissant des scènes 3D dans un espace 2D. Il existe de multiples techniques de post-traitement permettant d'examiner rapidement diverses combinaisons de techniques d'amélioration de rendu sans nécessiter de calculs supplémentaires. De nos jours, les techniques d'animation et de rendu sont de plus en plus répandues dans les domaines tels que les jeux vidéo et la production de films.

Le rendu est dédié à la génération d'images 2D à partir de modèles 3D. D'une part le rendu hors ligne peut utiliser un modèle physique précis pour produire des images photo-réalistes. D'autre part le rendu en temps réel tente de se rapprocher de ces mêmes résultats tout en réduisant le coût de calcul. L'Ombrage différé crée dans un premier temps le rendu géométrique avant de diffuser les informations relatives aux pixels via plusieurs mémoires tampons intermédiaires. Le processus d'ombrage peut être réduit à la partie visible de la caméra en utilisant des approches basées sur l'espace d'affichage. De nombreuses méthodes fondées sur cette approche ont été proposées pour améliorer les coûts de calculs. Dans cette thèse, nous explorons cette idée et nous développons de nouveaux algorithmes permettant d'obtenir une meilleure qualité visuelle et un coût de calcul réduit pour le rendu en temps réel.

L'occlusion ambiante (AO) est un moyen très répandu pour simuler l'éclairage indirect. Nous présentons une approche rapide et facile à mettre en œuvre pour l'approximation de l'occlusion ambiante de l'espace d'affichage. On calcule l'AO pour chaque pixel en intégrant les valeurs angulaires des échantillonneurs autour de la position du pixel qui pourrait bloquer l'éclairage ambiant. Nous appliquons une méthode séparable afin de réduire la complexité du calcul. Calculer d'abord l'occlusion sur une seule direction et ensuite la transporter dans une seconde passe qui

est l'évaluation de manière stochastique de l'ombrage finale en se basant sur des estimations AO s'avère extrêmement efficace. On peut obtenir des résultats visuellement prometteurs et proches d'une occlusion non séparable en combinant cette approche avec l'échantillonnage entrelacé et le flou de fonction de la géométrie.

La simulation des rides peut également être estimée sans changer l'information géométrique. Nous avons construit un modèle de rides en utilisant une technique graphique moderne qui effectue des calculs seulement dans l'espace d'affichage. Les animations faciales sont beaucoup plus réalistes avec la présence des rides. A partir de Facial Action Coding System (Système de Codage des Actions Faciales) (FACS) (Ekman, et al. 2002) , nous avons mis en place un système d'évaluation pour tester plusieurs facteurs qui peuvent influencer l'identification pour chaque unité d'action. L'objectif est d'identifier les expressions de visage à travers les animations faciales quand les agents virtuels communiquent leur comportement non-verbal. Plusieurs facteurs ont été prouvés, et les rides peuvent aider à reconnaître les unités d'action avec un taux plus élevé. Il s'est avéré que plusieurs facteurs, particulièrement les rides, peuvent améliorer le taux de reconnaissance des unités d'action.

L'animation peut être réalisée en utilisant des techniques différentes en fonction du but de l'application. D'une manière générale, les techniques d'animations peuvent être divisées en trois grandes catégories : les techniques de "morphing" (qui se définissent par les transformations qui s'appliquent directement sur les sommets du maillage), les techniques de cage (qui se définissent par les transformations qui se calculent en utilisant les cages correspondantes autour du maillage), et les techniques de squelette (les transformations qui sont calculées en utilisant une structure de squelette simplifiée). Etant donné que les os constituent l'unité fonctionnelle fondamentale qui supporte les vertèbres du corps, il est plus approprié de simuler l'animation des agents virtuels par l'utilisation de la structure de squelette. Cette version simplifiée du rendu de la peau ou "skinning" est facile à contrôler en utilisant des approches procédurale ou statistiques.

Dans les approches procédurales, la cinématique inverse (inverse kinematics : IK) peut être utilisée pour résoudre la posture hiérarchique du squelette. Nous présentons une méthode de cinématique inverse rapide et facile à mettre en IJuvre qui s'appuie sur un modèle masse-ressort et qui repose sur les interactions de forces entre les masses. Les interactions de forces entre les masses peuvent être vues comme un problème de minimisation de l'énergie, et les positions relatives peuvent être tracées en utilisant l'intégration "Verlet". Nous résolvons le problème d'IK dans l'espace de position au lieu de l'espace d'orientation. Les rotations des articulations sont donc calculées en utilisant une méthode "closed-form" dans un système de coordonnées où les axes locaux sont déjà prédéfinis. La combinaison de ces deux approches offre une très bonne qualité visuelle en haute performance de vitesse.

En se basant sur notre méthode d'IK, nous proposons un modèle de synthèse des gestes corporels expressifs intégrés dans notre plateforme d'agents conversa-

tionnels. Notre implémentation est fondée sur un modèle de corps basé sur une solution de cinématique hybride. Le mouvement relatif passif est utilisé pour corréler les différentes parties du corps dans un instant clé qu'on note "Key frame". Nous décrivons le schéma descriptif de notre modèle qui commence par une description symbolique du comportement corporel de l'agent jusqu'à la construction d'un ensemble de "key frame" et enfin la génération de l'animation de tout le corps enrichi par l'aspect expressif. Ce système offre plus de flexibilité pour configurer la cinématique expressive directe ou indirecte. Ceci peut être étendu à d'autres figures articulées.

De façon globale, cette thèse présente notre travail en 3D dans le rendu et l'animation. De nombreuses nouvelles approches ont été proposées pour améliorer la performance en termes de vitesse et de qualité des résultats. Toutes ces méthodes ont été intégrées dans notre système d'agent virtuel.

Summary

Animation and rendering are both important research domains in computer graphics. Animation is the study of motion of virtual objects. It observes the transformations of virtual objects through time in virtual worlds. Visual plausible motion can be performed by using physical simulations and controllable mechanism models. Rendering produces the final images by converting 3D scenes into 2D screen space. Diverse of post-processing techniques allows the user to quickly examine various combinations of enhancement techniques without excessive recomputations. After shading process, various extraordinary results can be achieved. Since computer generated imagery (CGI) is well developed in the past 20 years, both of animation and rendering are widely used in gaming, film production and different visualization applications.

On the one side, rendering is dedicated to generating images from 3D models. Offline rendering can use an accurate physical model to produce impressive photorealistic images. Real-time renderer tries to approximate the results, and reduce the computational cost. Deferred shading pipeline can render the geometry once and stream the pixel information into several intermediate buffer storages. The shading can be reduced to the visible portion of the camera by using screen space approaches. Many methods based on screen space computation have been proposed to improve time performance. In this thesis, we explore the screen space idea, develop some new simple algorithms for high visual quality, low cost online rendering.

Ambient occlusion (AO) is a widespread way to simulate indirect lighting. we present a fast easy-to-implement separable approximation to screen space ambient occlusion. We evaluate AO for each pixel by integrating angular values of samplers around the pixel position which potentially block the ambient lighting. We apply a separable fashion to reduce the complexity of the evaluation. Computing occlusion first along a single direction and then transporting this occlusion into a second pass that is stochastically evaluating the final shading based on the AO estimates proves extremely efficient. Combined with interleaved sampling and geometry-aware blur, visually convincing results close to a non-separable occlusion can be obtained at

much higher performance.

Wrinkle simulation can also be approximated without changing geometry information. We built a wrinkles model by using a modern graphics technique which performs computations only in screen space. With the help of wrinkles, the facial animation can be more realistic. In our Facial Action Coding System based facial animation system, we set up an evaluation system to test several factors that may influence the identification by each single action unit. The objective is to identify expressions through facial animations when studying virtual agent communications. Several factors have been proved, and wrinkles can help to recognize action units with a higher rate.

On the other side, animation can be achieved by using different techniques depending on the aim and purpose of an application. In a general sense, it can be divided into 3 types: the morphing method (transformations are applied directly on mesh's vertices), the cage method (transformations are computed by using corresponding cages around the mesh), the skeletal method (transformations are computed by using simplified bone structure). Since the bone is the fundamental functional unit which supports the vertebrate body, it is suitable to simulate virtual character by using skeletal structure. This simplified version of skinning body is easy to control by using data driven approaches or procedural methods.

In the procedural methods, inverse kinematics (IK) can be used to find the hierarchical posture solutions. We present a fast and easy-to-implement locally physics-based IK method. Our method builds upon a mass-spring model and relies on force interactions between masses. The force interactions between masses are formulated as an energy minimization problem, and the relative positions can be traced by using the Verlet integration. We solve the IK problem in the position space instead of the orientation space. Joint rotations are then computed using the closed-form method with predefined local axis coordinates. Combining these two approaches offers convincing visual quality results obtained with high time performance.

Base on our IK method, we propose our expressive body-gestures animation synthesis model for our Embodied Conversational Agent (ECA) technology. Our implementation builds upon a full body reach model using a hybrid kinematics solution. The relative passive motion is used to correlate the body parts in the key frame . We describe the full pipeline of our model that starts from a symbolic description of behaviours, to the construction of a set of key frames till the generation of the whole animation enhanced with expressive qualities. This system offers more flexibility to configure expressive Forward and Inverse Kinematics (FK and IK). It can be extended to other articulated figures.

Overall, this thesis presents our work in 3D rendering and animation. Several new approaches have been proposed to improve both the quality and the speed performance. All these methods have been used in our virtual agent system.

Contents

Resumé Long

Introduction

L'animation et le rendu sont deux domaines de recherche importants dans l'informatique graphique. L'animation est l'étude des mouvements des objets virtuels et de leurs transformations dans le temps dans un monde virtuel. Il est possible de créer des mouvements crédibles en utilisant des simulations physiques et des modèles fondés sur des mécanismes contrôlables. Le rendu produit les images finales en convertissant des scènes 3D dans un espace 2D. Il existe de multiples techniques de post-traitement permettant d'examiner rapidement diverses combinaisons de techniques d'amélioration de rendu sans nécessiter des calculs supplémentaires. De nos jours, les techniques d'animation et de rendu sont de plus en plus répandues dans les domaines tels que les jeux vidéo et la production de films.

Rendu

Le rendu est dédié à la génération d'images 2D à partir de modèles 3D, les grandes quantités de données de scène 3D pour construire leur représentation visuelle en conservant les propriétés de la scène, comme la géometrie, l'éclairage, les matériaux, l'environnement, et mème des personnalités de personnages virtuels. Dans le monde réel, toutes ces propriétés sont perçues à travers les interactions entre des objets et des lumières. Idéalement, la scène virtuelle peut être construite à l'aide de ces modèles physiques. L'éclairage direct représente l'interaction directe entre les sources de lumière et les objets virtuels. Il est facile de simuler des modèles simples mais le résultat de la simulation est loin d'être réaliste. L'éclairage indirect est un phénomène supplémentaire qui capture la lumière refléchie à la fois entre et à l'intérieur des surfaces de forme (par exemple, la diffusion de lumière sous la surface de la peau est un phénomène important pour le rendu réaliste des agents). Il améliore la qualité visuelle mais sa simulation est beaucoup plus complexe. Le rendu hors ligne peut utiliser les modèles physiques précis pour produire des images photo-réalistes mais il n'est pas possible de parvenir à des interactions temps réel avec de telles méthodes. Le rendu en temps réel tente de se rapprocher de ces même résultats tout en réduisant le coût de calcul. "Deferred Shading" crée dans

un premier temps le rendu géométrique avant de diffuser les informations relatives aux pixels via plusieurs mémoires tampons intermédiaires. Le processus d'ombrage peut être réduit à la partie visible de la caméra en utilisant des approches basées sur l'espace d'affichage [80] [115] [87]. De nombreuses méthodes fondées sur cette approche ont été proposées pour améliorer les coûts de calculs. Dans cette thèse, nous explorons cette idée et nous développons de nouveaux algorithmes permettant d'obtenir une meilleure qualité visuelle et un coût de calcul réduit pour le rendu en temps réel.

Animation

L'animation peut être réalisée en utilisant des techniques différentes en fonction du but de l'application. D'une manière générale, les techniques d'animation peuvent être divisées en trois grandes catégories : les techniques de "morphing" (qui se définissent par les transformations qui s'appliquent directement sur les sommets du maillage), les techniques de cage (qui se définissent par les transformations qui se calculent en utilisant les cages correspondantes autour du maillage), et les techniques de squelette (les transformations qui sont calculées en utilisant une structure de squelette simplifiée). Etant donné que les os constituent l'unité fonctionnelle fondamentale qui supporte le corps, il est plus approprié de simuler l'animation des agents virtuels par l'utilisation de la structure de squelette. Cette version simplifiée de l'animation de la peau ou "skinning" est facile à contrôler en utilisant des approches procédurale ou statistiques. Elle est aussi très efficace par rapport aux autres méthodes. Une autre méthode d'animation de muscles, similaire aux techniques de squelette, est basée sur le "Facial Action Coding System (FACS)" et est proposée pour l'animation faciale. Les Unités d'action(UA) [106] sont définies pour contrôler les contractions musculaires du visage humain. Ce sont des méthodes qu'on peut utiliser pour manipuler les mouvements des agents virtuels. En haut niveau de conception, la création d'animations compatibles avec une variété de corps multimodale est importante [3] [125] [132] [107] [108]. Les animations de personnages ont besoin d'être expressives et dans une certaine mesure naturelle, mais il est difficile de le réaliser à l'aide des outils existants. De plus, tous les systèmes d'interactions émotionnelles [118] [12] doivent être totalement en temps réel, et tous les calculs doivent être en ligne. Les expressions seront considérées comme des comportements dynamiques et ne seront plus statiques [101]. Ainsi, dans la deuxième partie de cette thèse, nous nous concentrons sur les générations d'animation en temps réel pour des personnages virtuels expressifs.

Contributions

L'occlusion ambiante (AO) est un moyen très répandu pour simuler l'éclairage indirect. Nous présentons une approche rapide et facile à mettre en œuvre pour

l'approximation de l'occlusion ambiante de l'espace d'affichage. On calcule l'AO pour chaque pixel en intégrant les valeurs angulaires des échantillons autour de la position du pixel qui pourraient bloquer l'éclairage ambiant. Nous appliquons une méthode de calcul pseudo-séparable afin de réduire la complexité du calcul. Nous calculons d'abord l'occlusion sur une seule direction et ensuite la transportons dans une seconde passe. L'ombrage final est évalué de manière stochastique en se basant sur les estimations locales de l'AO, ce qui s'avère être extrêmement efficace. Les résultats obtenus sont visuellement prometteurs et proches d'une occlusion ambiante non-séparable en combinant cette approche avec l'échantillonnage "interleaved" et le flou "geometry-aware".

La simulation des rides peut également être estimée d'une manière sans changer l'information géométrique. Nous avons construit un modèle de rides en utilisant une technique graphique moderne qui effectue des calculs seulement dans l'espace d'affichage. Les animations faciales sont beaucoup plus réalistes avec la présence des rides. Dans notre Système de Codage des Actions Faciales (FACS), nous avons mis en place un système d'évaluation pour tester plusieurs facteurs qui peuvent influencer l'identification pour chaque unité d'action. L'objectif est d'identifier les expressions de visage à travers les animations faciales quand les agents virtuels communiquent leur comportement non-verbal. Il s'est avéré que plusieurs facteurs, particulièrement les rides, peuvent améliorer le taux de reconnaissance des unités d'action.

Dans les approches procédurales, la cinématique inverse (inverse kinematics : IK) peut être utilisée pour résoudre la posture hiérarchique du squelette. Nous présentons une méthode de cinématique inverse rapide et facile à mettre en œuvre qui s'appuie sur un modèle masse-ressort et qui repose sur les interactions de forces entre les masses. Les interactions de forces entre les masses peuvent être vues comme un problème de minimisation d'énergie, et les positions relatives des masses peuvent être obtenues en intégrant les équations de la dynamique à l'aide d'un modèle "Verlet". Nous résolvons le problème d'IK dans l'espace de position au lieu de l'espace d'orientation. Les rotations des articulations peuvent être calculées en utilisant les méthodes de forme fermée ("closed-form solution") dans un système de coordonnées où les axes locaux sont déjà prédéfinis. La combinaison de ces deux approches offre une très bonne qualité visuelle et une haute performance.

En se basant sur notre méthode d'IK, nous proposons un modèle de synthèse des gestes corporels expressifs intégrés dans notre plateforme d'agents conversationnels. Notre implémentation est fondée sur un modèle de corps basé sur une solution de cinématique hybride. Le mouvement relatif passif est utilisé pour corréler les différentes parties du corps à un instant clé qu'on note "Key frame". Nous présentons le schéma descriptif de notre modèle qui commence par une description symbolique du comportement corporel de l'agent jusqu'à la construction d'un ensemble de "key frames" et finalement la génération de l'animation de tout le corps

est enrichi par l'aspect expressif. Ce système offre plus de flexibilité pour configurer la cinématique expressive directe ou indirecte. Ceci peut être étendu à d'autres figures articulées.

De façon globale, cette thèse présente notre travail en 3D dans le rendu et l'animation. De nombreuses nouvelles approches ont été proposées pour améliorer la performance en termes de vitesse et de qualité des résultats. Toutes ces méthodes ont été intégrées dans notre système d'agent virtuel.

Occlusion Ambiante Séparable

L'occlusion ambiante (AO) est une approximation efficace de l'éclairage indirect qui peut être calculée en temps réel pour des scènes dynamiques. Comme démontré par de nombreux travaux récents, elle améliore la perception des volumes, des concavités et des zones de contact des objets en 3D.

Formellement, l'AO d'un élément $\mathbf{p}$ de surface muni d'un vecteur normal $\mathbf{n}$ est définie comme l'intégrale sur l'hémisphère Ω de la visibilité :

$$AO(\mathbf{p},\mathbf{n}) = \frac{1}{\pi}\int_{\Omega} V(\mathbf{p},\omega)\mathbf{n}\cdot\omega\, d\omega,$$

avec $V(\mathbf{p},\omega)$ étant le terme de visibilité en $\mathbf{p}$ dans la direction ω. Souvent, le terme de visibilité V est couplé à une fonction d'atténuation qui assure que les objets distants sont ignorés.

L'intégrale AO peut être évaluée par l'intégration de Monte Carlo [72]. Bien que l'AO est moins chère à évaluer qu'un éclairage indirect complet, il est encore trop coûteux pour le calcul interactif. Par conséquent, des schémas d'approximation sont couramment utilisées pour des applications en temps réel, comme les jeux-vidéo.

Dans cette partie de la thèse, nous nous concentrons sur la méthode d'espace écran (SSAO) [92], qui utilise la mémoire tampon de profondeur pour approximer la scène 3D par une formulation dans un espace 2.5D. Nous présentons une approche rapide et facile à mettre en œuvre pour l'approximation de l'occlusion ambiante de l'espace d'affichage. On calcule l'AO pour chaque pixel en intégrant les valeurs angulaires des échantillonss autour de la position du pixel qui pourraient bloquer l'éclairage ambiant. Nous appliquons une méthode séparable afin de réduire la complexité du calcul. Nous calculons d'abord l'occlusion sur une seule direction et ensuite la transportons dans une seconde passe. L'ombrage final est évalué de manière stochastique en se basant sur les estimations locales de l'AO, ce qui s'avère être extrêmement efficace. Les résultats obtenus sont visuellement prometteurs et proches d'une occlusion ambiante non-séparable en combinant cette approche avec l'échantillonnage "interleaved" et le flou "geometry-aware".

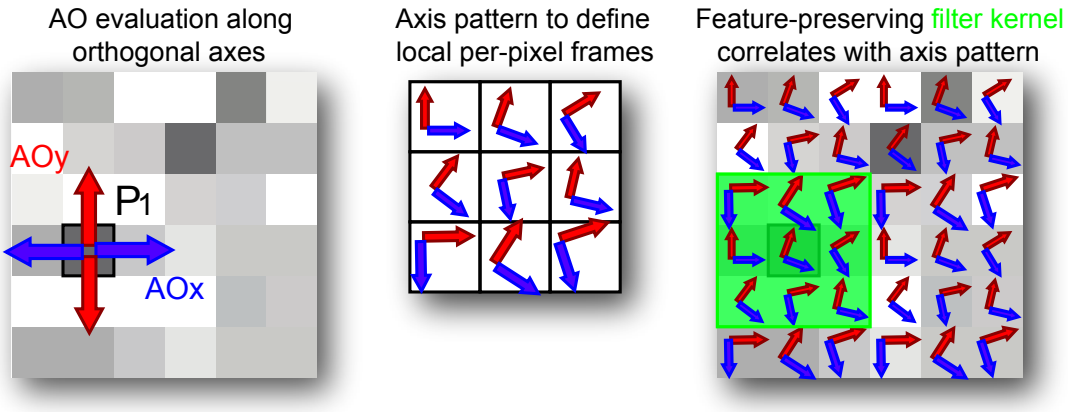

FIGURE 1 – **Principe :** Notre méthode séparable combine deux évaluations orthogonales en une dimension à l'écran (par exemple, axes x et y, à gauche). En changeant les axes de calcul par pixel, nous pouvons dériver une approximation stochastique valide de l'occlusion 2D en combinant le résultat des échantillons voisins.

Principe : Occlusion Ambiante Séparable

Le calcul de l'AO à un pixel $\{i,j\}$ à l'endroit du monde $\mathbf{p}_{i,j} \in \mathbb{R}^3$ avec un vector normal $\mathbf{n}_{i,j} \in S^2$ peut être compris comme une forme de filtrage local des buffers écran 2D (profondeur, les normales) :

$$AO(i,j) = \frac{1}{k^2} \sum_{x=i-k/2}^{i+k/2} \sum_{y=j-k/2}^{j+k/2} V(\mathbf{p}_{i,j}, \omega_{x,y}) \mathbf{n}_{i,j}.\omega_{x,y}, \tag{0.0.1}$$

où ω est le point au-dessus du pixel sur la sphère unité.

Comme démontré par Pham [110] pour le filtrage bilatéral d'images, les filtres locaux séparables peuvent être estimés d'une manière très efficace. Dans notre méthode, nous faisons l'approximation que le filtre peut être séparé sur deux dimensions x et y. Les deux filtres sont évalués en une dimension seulement avant d'être couplés.

Dans un premier passage, nous calculons l'occlusion ambiante $AO_{\mathrm{x}}(i,j)$ dans la direction X à chaque pixel en fixant y à 0. Dans une seconde passe, les pixels sont évalués par les pixels voisins de long de l'axe y $AO_{\mathrm{y}}(i,j)$, en fixant x à 0. La moyenne de ces deux évaluations partielles de AO fournit une forme séparable, mais ignore la plupart des dispositifs d'occlusion voisins, i.e. tous les éléments situés dans les directions diagonales (voir la Figure 1). Ces occlusions potentielles sont essentielles et doivent être prises en considération. Notre objectif est d'exploiter le calcul de la visibilité des points voisins pour obtenir une approximation de l'occlusion effective dans toutes les directions.

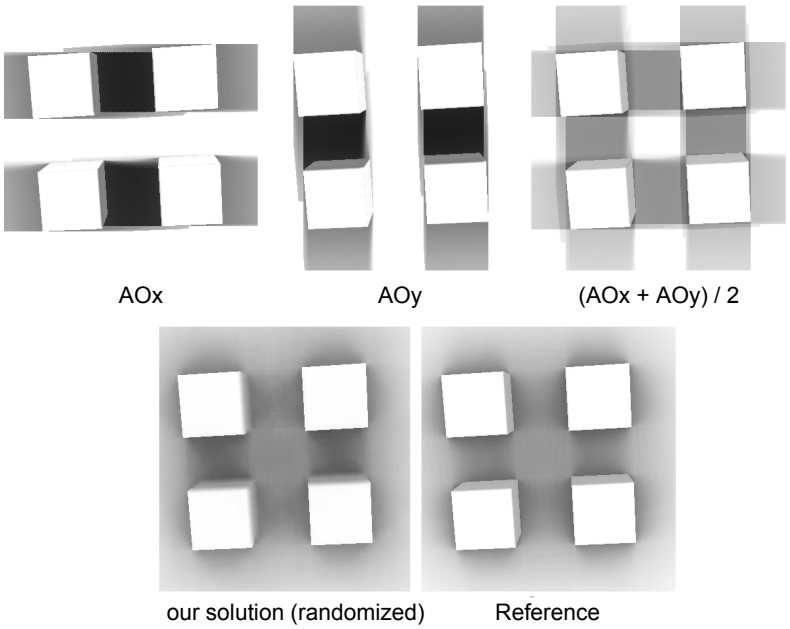

FIGURE 2 – Perturber de manière aléatoire la notion d'axes "horizontaux" et "verticaux" par pixel conduit à un bon résultat, tandis qu'une combinaison simple des deux axes conduit à des artefacts visibles.

Perturbation Même si un lissage fonction bilatéral contribue généralement à minimiser des artefacts de SSAO [10, 114], une construction s'appuyant sur une orientation globale conduit à des artefacts visibles même après la filtrage sur plusieurs échantillons adjacents (voir la Figure 2). Nous résolvons ce problème en remplaçant l'orientation globale $\{x, y\}$ par une locale et définie de manière aléatoire pour chaque pixel.

Afin de favoriser le processus subséquent de filtrage, nous utilisons l'échantillonnage entrelacé [61] pour créer des motifs de bruit. La taille du motif de bruit est choisie pour correspondre à la taille du support de filtrage, pour s'assurer que les artefacts soient supprimés car elle correspond implicitement à ces processus. S'appuyant sur les deux évaluations 1D uniquement, on transforme la complexité originale de SSAO de $O(k^2)$ en $O(k)$.

Implémentation et Résultats

Nous avons implémenté notre algorithme en OpenGL / GLSL sur une carte graphique NVIDIA GTX480. Pour démontrer l'indépendance de notre approche à d'autres types d'optimisation, nous avons utilisé trois techniques différentes de SSAO. Pour chaque technique, le résultat initial est comparé à sa version séparable. Les performances sont présentées dans le tableau 1 pour les images d'une résolu-

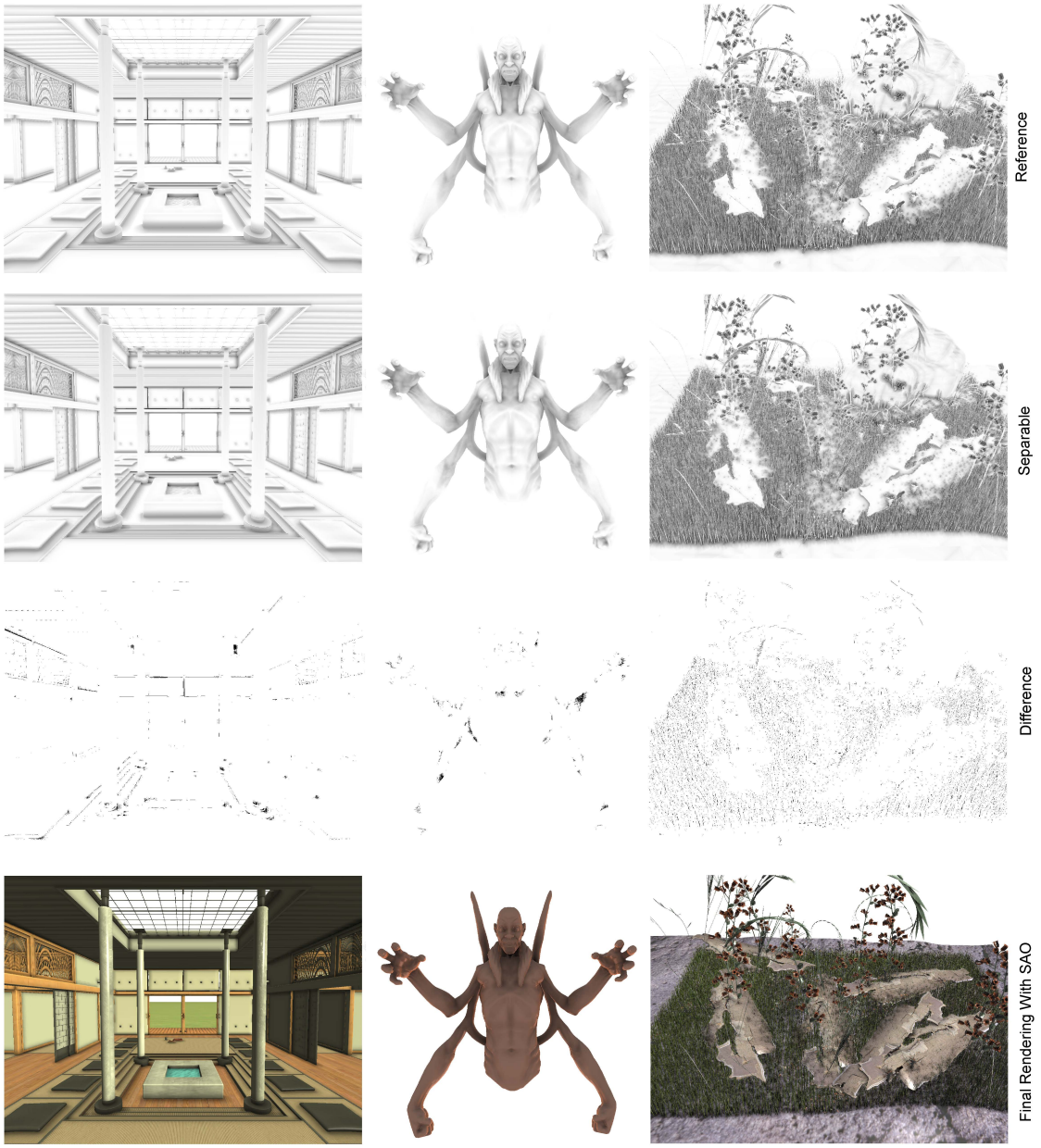

FIGURE 3 – Comparaison entre la référence (première rangée) et l'évaluation AO séparable (deuxième rangée) : pour les images de différence de perception (troisième rangée), les valeurs blanches se détachent sans différence alors que les valeurs noires indiquent des différences visibles.

Size	Samples / Separable	Crytek [92]	Vol. Obs. [79]	HBAO [10]
5	5 × 5 / no	3.2 ms	3.5 ms	3.6 ms
	5 × 2 / yes	3.4 ms	3.6 ms	3.5 ms
11	11 × 11 / no	13.9 ms	14.8 ms	15.0 ms
	11 × 2 / yes	5.8 ms	5.9 ms	6.0 ms
21	21 × 21 / no	49.7 ms	51.9 ms	51.9 ms
	21 × 2 / yes	9.9 ms	9.9 ms	10.2 ms

TABLE 1 – Comparaison de performance entre l'évaluation exacte et séparable pour les méthodes diverses avec des tailles de filtrage variées.

tion d'écran de 1024x768 sur les scènes de 50,000 triangles. Dans chaque cas, nous avons obtenu une qualité visuelle similaire en utilisant soit une évaluation exacte ou notre approximation séparable. Les durées montrent clairement l'accélération significative des méthodes différentes de SSAO, plus particulièrement lors de l'utilisation d'un grand rayon de prise en compte des objets dans le terme de visibilité. Nous avons également analysé l'erreur introduite par notre méthode et mesuré les différences de perception dans l'espace colorimétrique Lab [147] (voir la Figure 3, troisième rangée) entre les tampons de référence (voir la Figure 3, première rangée) et nos approximations (voir la Figure 3, deuxième rangée). On peut voir que les erreurs restent faibles dans la pratique et sont même cachées lorsque les autre effets d'illumination sont ajoutés pour le rendu final (voir la Figure 3, dernière rangée).

Discussion : Occlusion Ambiante Séparable

Nous avons introduit une nouvelle méthode pour approximer l'occlusion ambiante en espace écran en décomposant l'évaluation de l'AO dans un espace séparé. Notre technique produit des résultats de bonne qualité et réduit considérablement le temps de calcul de l'AO. Elle peut être facilement intégré dans les méthodes existantes telles que démontré par l'utilisation de trois techniques différentes précédemment introduites dans la littérature scientifique. En outre, elle hérite de toutes les propriétés du SSAO telles que l'adaptabilité de la vue et sa capacité à traiter les scènes dynamiques arbitraires. Nous croyons que les mêmes approximations séparables pourrait être utiles pour d'autres techniques de rendu d'espace d'écran ou générale.

Modèle de Rides et identification des Unités d'Action pour l'animation du visage

L'AO peut être utilisée pour améliorer la perception du contenu du modèle 3D, par exemple les visages humaines. On peut utiliser les techniques d'espace-écran pour accélérer le processus, mais ce sont des méthodes généraux de rendu. Dans cette partie, nous montrons qu'à la frontière du rendu et de l'animation de personnages virtuels, la formulation en espace écran peut aider à améliorer la qualité de l'animation faciale en modélisant les rides sur le visage.

Notre système d'animation faciale est construit à partir du système d'encodage d'Action du visage (FACS) [106]. Nous travaillons cependant sur les facteurs susceptibles d'influencer les unités d'action (UA) et ainsi de changer la perception de l'animation. Concernant les détails du visage, les rides peuvent être un facteur d'impact important, accroissant le réalisme lors des simulations d'expressions faciales. Dans ce travail, nous étudions donc les facteurs pouvant influencer l'identification des actions simples du visage (par exemple, froncer les sourcils ou sourir), à savoir l'intensité, la dynamique et l'application des rides. Les résultats de notre étude montrent que lors de l'évaluation de mouvements faciaux simples, ces derniers sont mieux identifiés s'ils sont dynamiques et avec une plus grande intensité. D'autre part, les expressions intenses de mouvements faciaux simples sont perçus comme moins naturels et moins réalistes. Lors de notre étude d'évaluation, nous avons constaté qu'avec l'aide des rides les utilisateurs peuvent facilement percevoir les mouvements du visage.

Modélisation des Rides

Il existe deux approches principales pour générer des rides : les méthodes basées sur la texture [30] ou la géométrie [74]. En raison du coût de calcul, nous avons choisi d'appliquer la technique basée sur la texture dans notre système. Cette méthode, comme la technique de Placage de relief, a été introduite par Blinn [14]. L'auteur propose de modifier le vecteur normal à la surface avant le calcul d'éclairage pour simuler visuellement une variation de la géométrie sans la réaliser de manière effective.

Pour finir notre étude d'identification des actions simples du visage, nous générons des rides dans l'espace 'écran. Pour ce faire, nous nous appuyons sur les travaux de Blinn en nous basant sur une méthode de "Placage de relief" implémentée en shader et executée en GPU. On simule l'effet des rides en effectuant le calcul du vecteur normal perturbé dans l'espace d'écran en utilisant le pixel shader en OpenGL. Ainsi, la complexité de la technique dépend du nombre de pixels, et non du nombre de sommets du modèle facial. Le vecteur normal perturbé ne dépend que du vecteur normal de la surface d'entrée et de la hauteur de l'espace écran.

AU Number	FACS Name	Wrinkle Number
1	Inner Brow Raiser	0
2	Outer Brow Raiser	1, 2
4	Brow Lowerer	3
6	Cheek Raiser	4, 5
10	Upper Lip Raiser	6, 7
12	Lip Corner Puller	8, 9
20	Lip stretcher	10, 11

TABLE 2 – Plusieurs UAs principales sont utilisées dans ce travail avec les rides correspondantes définies manuellement. Pour certaines UAs, les rides ont besoin d'être séparées selon leur côté (droit ou gauche).

Nous n'avons ainsi pas besoin d'appliquer une transformation de l'espace tangeant à l'espace de l'environnement. Le vecteur normal perturbé est calculé en fonction des changements de composition du gradient local de la surface et des directions x et y pour chaque pixel dans l'espace d'écran. Nous définissons 12 groupes de rides dans les textures, qui correspondent aux UAs que nous cherchons à évaluer dans ce travail (voir la Figure 4 et le Tableau 4.1). Ces UAs sont souvent liées à l'affichage des 6 émotions de base. Lors de l'exécution, quand une unité d'action est activée sur le maillage du visage, le GPU reçoit sa valeur d'intensité et calcule les rides correspondantes. Le résultat final prend en compte toutes les rides actives (voir la Figure 5)).

Evaluation

Les évaluations perceptives de l'interaction affective utilisant les rides ont été réalisées par Courgeon et al. [30]. Ces derniers ont montré que l'utilisation de rides réalistes augmente l'expressivité des agents, améliorant la crédibilité aux yeux des utilisateurs. Ces études n'ont cependant pas montré l'impact d'un tel facteur sur la reconnaissance des catégories émotionnelles.

Dans notre travail, nous étudions les facteurs tels que l'intensité et la dynamique des rides, de même que leur influence sur l'identification des actions simples du visage. Nous avons proposé trois hypothèses à tester :

- H1. Les actions de haute intensité sont mieux identifiées, mais elles sont moins naturelle et moins réalistes,
- H2. Les actions affichées par une animation sont mieux identifiées que celles présentées par des images statiques,
- H3. Les actions du visage avec des rides sont mieux identifiées, ou, du moins, elles diminuent les confusions.

Les 6 actions faciales ont été étudiées dans un premier temps : 3 pour le haut

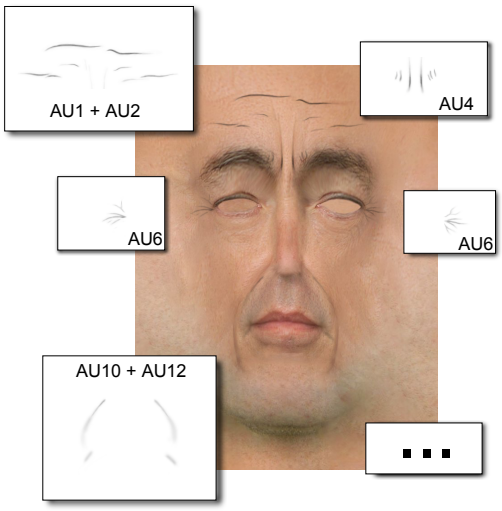

FIGURE 4 – Cartes de normales / Cartes de Placage de relief pour les rides par rapport aux unités d'action différentes.

	Haut	Bas
Animations	8	8
Images	8	8

TABLE 3 – Le nombre de stimuli dans chaque groupe pour chaque participant

du visage et 3 pour le bas du visage. Ces UAs sont : AU4 (froncement de sourcils), AU1+4 (sourcils obliques), AU1+2 (haussement de sourcils), AU6 (pommettes relevées), AU20 (lèvres étendues) et AU12 (coins des lèvres tirés vers le haut). Ces actions sont particulièrement importantes dans la communication des émotions par les agents virtuels. Les actions restantes ont été intentionnellement choisies afin de créer des confusions, par exemple les sourcils obliques et le haussement de sourcils. Deux agents différents ont été utilisés : l'un féminin et l'autre masculin. Chacune des 6 actions faciales a été affichée avec et sans rides et avec une intensité faible et élevée (voir les Figures 6 et 7). Ces actions ont en outre été présentées dans le cadre d'une animation (état dynamique) et par une image statique représentant l'intensité maximale de cette même action. Pour cette expérience, nous avons donc utilisé 128 stimuli différents, chaque animation ayant été validée par un expert certifié FACS.

Le test a été mis en ligne et organisé comme un ensemble de pages Web ; chacune de ces pages affiche un stimulus sous la forme d'une vidéo pouvant être visionnée

FIGURE 5 – Résultats de notre agent : **Gauche** : position neutre, **Milieu** : AU 1 + 2, **Droit** : AU 11 + 12.

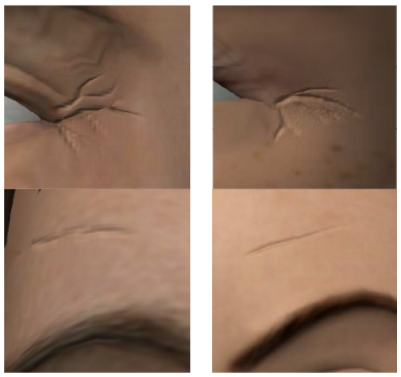

FIGURE 6 – Rides de deux agents (masculin à gauche et féminin à droite) correspondant aux AU6 (pommettes relevées) et AU1+2 (haussement des sourcils).

plusieurs fois. Les participants doivent répondre à toutes les questions relatives à une vidéo avant de passer à la prochaine animation, sans possibilité de revenir en arrière pour éventuellement modifier les précédentes réponses. Chaque participant peut évaluer un maximum de 32 stimuli (16 pour le haut du visage, 16 pour le bas avec 16 animations et 16 images) (voir le Tableau 3). Dans l'expérience, nous avons demandé aux participants d'identifier les stimuli en utilisant l'échelle de Likert à 5 points variant de "totalement pas d'accord" à "totalement d'accord". Dans les deux cas, les participants ont été invités à exprimer leur opinion sur le naturel et le réalisme du stimulus en utilisant deux échelles de Likert à 5 points. Pour la colonne EXP (voir le Tableau 4), nous n'utilisons que les valeurs des échelles de Likert correspondantes.

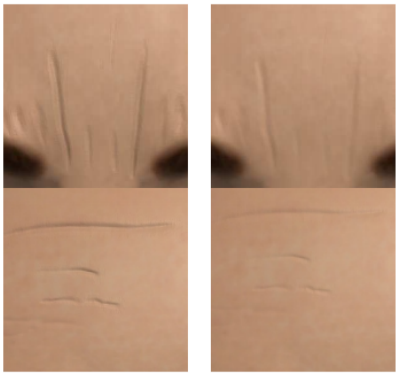

FIGURE 7 – Rides des mouvements de sourcils avec les intensités faibles et élevées.(AU4 et AU 1+2).

	intensité haute			**intensité basse**		
AU	**EXP**	**NAT**	**RE**	**EXP**	**NAT**	**RE**
1+2	**4.23**	3.40	3.32	**3.18**	3.46	3.33
4	**3.84**	3.75	3.69	**3.31**	3.90	3.82
1+4	**3.03**	3.28	3.23	**2.64**	3.33	3.35
6	**3.14**	**3.03**	**3.03**	**2.46**	**3.33**	**3.34**
20	**4.33**	**2.30**	**2.34**	**4.02**	**3.21**	**3.16**
12	**4.21**	**2.67**	**2.72**	**3.77**	**3.29**	**3.36**

TABLE 4 – L'effect d'intensité. EXP : expression ; NAT : naturel ; RE : réalisme.

Résultats

Au total, nous avons recueilli 2092 réponses provenant de 92 participants (âgés de 19 à 51 ans). L'échelle de Likert ne satisfait généralement pas la condition de distribution normale. Par conséquent, nous avons appliqué un test non paramétrique de Mann-Whistley à nos données. Le test de Mann-Whistley montre l'effet de l'intensité et de la dynamique sur le naturel et le réalisme. L'effet des rides a été observé séparément. Ensuite, afin de comparer les taux d'identification, nous avons considéré chaque UA séparément.

Cinq des six actions (voir le Tableau 4) ont été mieux identifiées sur les expressions les plus intenses. Dans le même temps 3 actions intenses (AU12, 20 et 6) ont été perçues moins réalistes alors que 2 d'entre elles étaient moins naturelles. Trois des six actions (AU4, AU6 et AU12) ont été mieux identifiées à partir des animations que des images. 4 actions ont été perçues plus naturelles et 2 plus réalistes lorsqu'elles étaient présentées sous forme d'animations (voir le Tableau 5). Enfin, une seule action, AU4 a été significativement mieux identifiée grâce aux rides.

	animation			image statique		
AU	**EXP**	**NAT**	**RE**	**EXP**	**NAT**	**RE**
1+2	3.82	**3.68**	**3.53**	3.55	**3.11**	**3.07**
4	**3.86**	3.79	3.74	**3.21**	3.87	3.77
1+4	**3.00**	**3.48**	3.41	**2.63**	**3.10**	3.15
6	**3.46**	3.24	3.24	**1.89**	3.10	3.10
20	4.20	**3.07**	**3.06**	4.15	**2.34**	**2.35**
12	**4.22**	**3.14**	**3.18**	**3.70**	**2.80**	**2.87**

TABLE 5 – L'effet de présentation.

Discussion

Cette partie de la thèse se focalise sur l'impact des rides, leur intensité et leur dynamique sur la perception des expressions faciales. La principale contribution de cette recherche est qu'au lieu d'utiliser des expressions stéréotypées (qui sont rares), nous analysons les actions simples du visage. Nous avons également montré que les expressions les plus intenses sont perçues comme étant moins naturelles et donc moins réalistes. Le rôle des rides est cependant ambivalent : Dans le cas des actions ambiguës, les rides peuvent ainsi augmenter la confusion contrairement à l'idée avancée dans l'hypothèse 3. Si la simulation des rides est utile pour la visualisation, elle ne peut pas vraiment aider à identifier l'expression. Notre travail confirme enfin que la dynamique d'une expression est l'indice le plus important à prendre en compte lors de l'identification de cette même expression.

Cinématique inverse basée sur le transfert des énergies

Pour animer des personnages virtuels, l'idée de base est d'utiliser des méthodes de postures clés. Ces méthodes peuvent être des méthodes utilisant des données de capture de mouvement ou des méthodes procédurales. Au regard du système Greta actuel, nous avons choisi un modèle procédural utilisant la cinématique.

La cinématique est une méthode générale permettant de manipuler les chiffres de façon interactive et de générer des postures clés. Cette méthode est largement utilisée dans l'animation par ordinateur et la robotique. Il existe une version de cette méthode qui s'appelle la cinématique inverse (IK). Elle consiste à trouver les rotations de chaque articulation en respectant les différents degrés de liberté de rotation prédéfinis et les contraintes de position. Elle est souvent utilisée pour animer des agents autonomes.

Nous proposons une solution combinant cette méthode, basée sur la physique

d'un système masse-ressort, avec un facteur heuristique. L'approche physique de notre méthode conduit à des solutions plus réalistes. Cette approche est rapide, et peut faire face à de multiples contraintes d'une manière efficace sans augmenter de trop les temps de calcul. Elle peut être utilisée de façon séquentielle pour la création d'animations de tout le corps des agents virtuels. Le transfert d'énergie peut être facilement combiné avec les paramètres d'expressivité dans des systèmes de personnages virtuels qui permettent de moduler la qualité expressive des mouvements. Dans le reste du chapitre, nous présentons d'abord les approches existantes, puis nous décrivons notre solution de cinématique inverse. Nous illustrons notre méthode avec plusieurs exemples.

Principe

Nous utilisons un système masse-ressort pour résoudre le problème de cinématique inverse (la contrainte étant en général pour le personnage animé d'atteindre une cible). Puis, un solveur général de contraintes angulaires intégré dans notre système permet de lever l'ambiguité sur les rotations des articulations du squelette. Si les informations de rotation du squelette ne sont pas nécessaires, comme dans le jeu du serpent [56], nous pouvons simplifier le processus en utilisant uniquement le système masse-ressort.

Masse-Ressort

Les systèmes masse-ressort sont définis comme des systèmes d'oscillateurs harmoniques de la mécanique Newtonienne classique. En infographie, ils sont souvent utilisés pour les système de particules. Les positions sont ajustées en minimisant l'énergie de force qui est conservée dans des ressorts. La loi de Hooke [78, 123] définit le système masse-ressort tel que :

$$\vec{F}(t) = -k\,\vec{l}\,(t) \tag{0.0.2}$$

où F est une force de rappel, proportionnelle au déplacement l dans un ressort ayant une rigidité positive k.

La déplacement de masses peut être calculée par la loi de Newton.

$$F(t) = m\frac{d^2}{dt^2}x(t) + \frac{\beta}{m}\frac{dx}{dt} = ma \tag{0.0.3}$$

où F est la force, β est le facteur d'amortissement et a est l'accélération de la masse. $x(t)$ est la position déplacée de la masse au temps t, $v = \frac{dx}{dt}$ la vitesse, et $a = \frac{d^2}{dt^2}x(t)$ l'accélération.

Pour résoudre la trajectoire des particules de notre cas, nous utilisons l'équation de l'énergie. La chaîne IK peut être considérée comme une chaîne de particules.

La formule de chemin peut être défini comme suit :

$$y(t_1) = f(t_1, y(t_0)) \tag{0.0.4}$$

La méthode de base pour simuler le mouvement des particules est basée sur le schéma d'intégration d'Euler $y_n = y_{n-1} + h \times f(t_{n-1}, y_{n-1})$. Il s'agit d'une approximation de la méthode de Newton [35, 29].

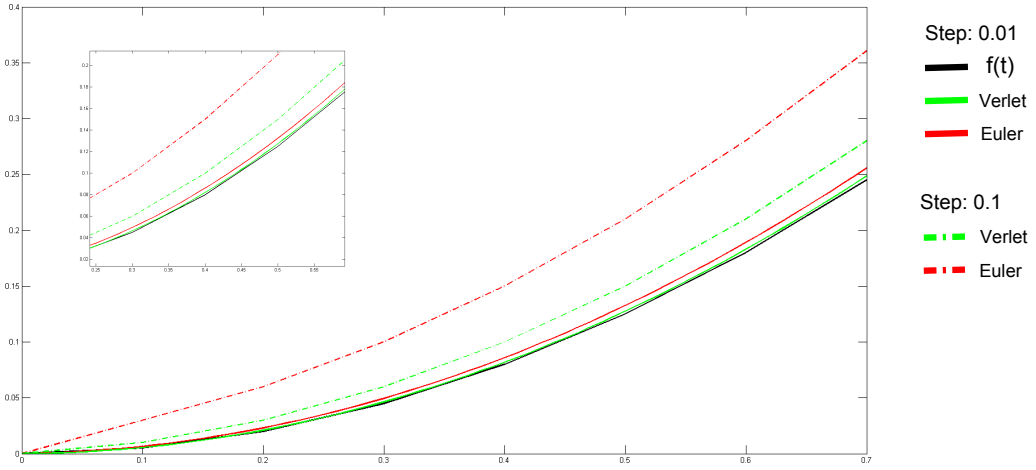

FIGURE 8 – Illustration de la différence entre les schémas de Verlet et d'Euler. Avec une valeur de pas discret plus grande, la méthode d'Euler induit des erreurs plus importantes et se révèle moins stable que la méthode de Verlet.

Dans notre cas, nous proposons d'utiliser l'intégration de Störmer-Verlet [134, 44] au lieu de celle d'Euler. Les comparaisons entre les méthodes d'Euler et de Verlet sont illustrées en figure 8. La méthode de Verlet est globalement plus stable et moins coûteuse que la méthode d'Euler.

Le schéma d'intégration de Störmer-Verlet est défini comme :

$$\frac{\Delta^2 \vec{x}_n}{\Delta t^2} = \frac{\frac{\vec{x}_{n+1} - \vec{x}_n}{\Delta t} - \frac{\vec{x}_n - \vec{x}_{n-1}}{\Delta t}}{\Delta t} = \frac{\vec{x}_{n+1} - 2\vec{x}_n + \vec{x}_{n-1}}{\Delta t^2} = \vec{a}_n \tag{0.0.5}$$

$$\vec{x}_{n+1} = 2\vec{x}_n - \vec{x}_{n-1} + \vec{a}_n \Delta t^2, \tag{0.0.6}$$

où $\vec{a}_n = Acceleration(\vec{x}_n)$.

Considérons une chaîne IK avec un ensemble de joints $J_i, i = 1, .., n$ et os B_i avec une longueur $\vec{d}_i$. Un ensemble de points de masse M_i et ressorts S_i est créé en faisant correspondre les articulations et les os. L'effecteur final est noté E et la cible

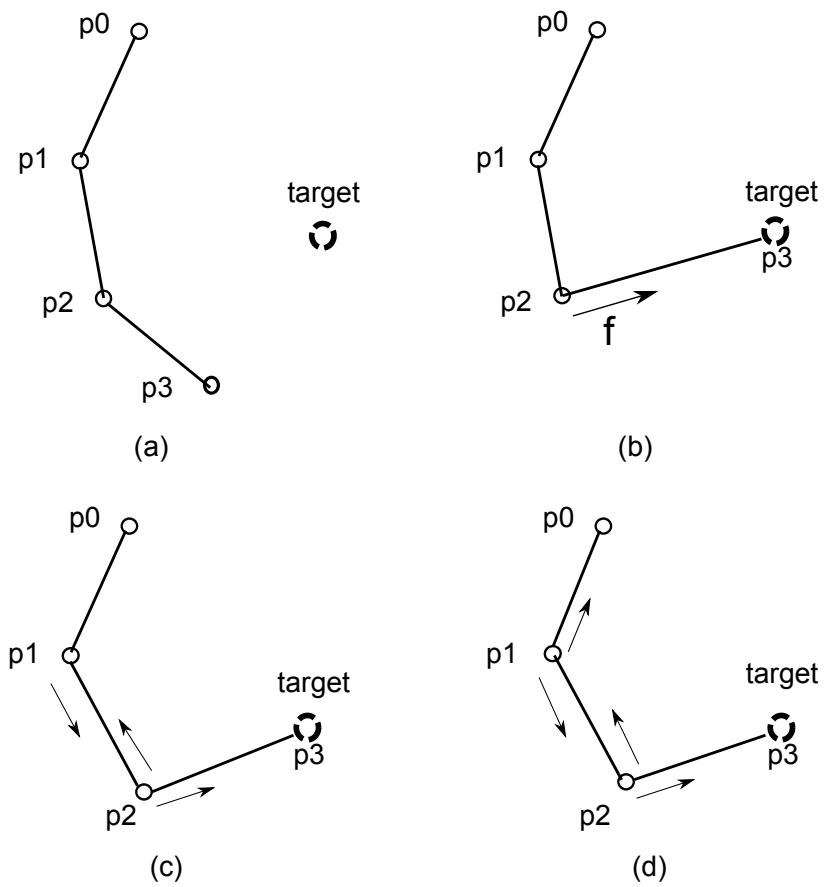

FIGURE 9 – Notre solution : (a) Initialiser configuration, (b) L'effecteur final est bougé vers la cible, (c) et (d) Le système de masse-ressort est simulé jusqu'à l'équilibre. p_i représente les articulations de la chaîne, et chaque joint a une série de positions x_n generées par notre solution à travers le temps.

T. Pour ce calcul, nous ignorons les informations de rotation, et réglons la valeur de la masse générale à 1, i. e. le système ne conserve que les informations de position des points de masse.

On commence chaque itération par l'ajout d'une force extérieure définie par l'utilisateur à l'effecteur final : $f = -k(T - E)$ (voir la Figure 9). Ensuite, nous calculons la force interne en utilisant le modèle masse-ressort. L'accélération conduit de manière itérative les points de masse aux nouvelles positions en utilisant la formule de Verlet (Eq.0.0.6). A chaque itération, nous vérifions la somme des forces internes de l'ensemble du système et nous assurons que $\|\sum \vec{F}_{sum}\| < \epsilon$. Le paramètre ϵ est le seuil déterminant la convergence du processus itératif. Enfin, l'effecteur est toujours convergent, i. e. se rapproche de la cible. On conserve le vecteur normalisé

de direction de chaque paire de points de masse (un point de masse et son parent) pour construire l'axe local pour chaque articulation. Le vecteur directionnel est également le vecteur directionnel d'articulation.

Après toutes les itérations, nous convertissons le système de masse dans un système commun.

Nous avons le vecteur directionnel de chaque paire joints (le joint et son parent). Nous utilisons ensuite la méthode de forme fermée [51] pour calculer la rotation de chaque articulation locale dans un système local où les axes sont définis à l'aide du vecteur de l'os et de son vecteur "up" de hauteur qui est perpendiculaire au vecteur de l'os. Ensuite on met à jour le système du squelette en entier. Le vecteur "up" est toujours perpendiculaire au vecteur directionnel de l'os, et il est automatiquement mis à jour lorsque l'on effectue une rotation sur le vecteur de l'os. Ceci est important pour la résolution des contraintes générales.

Contraints

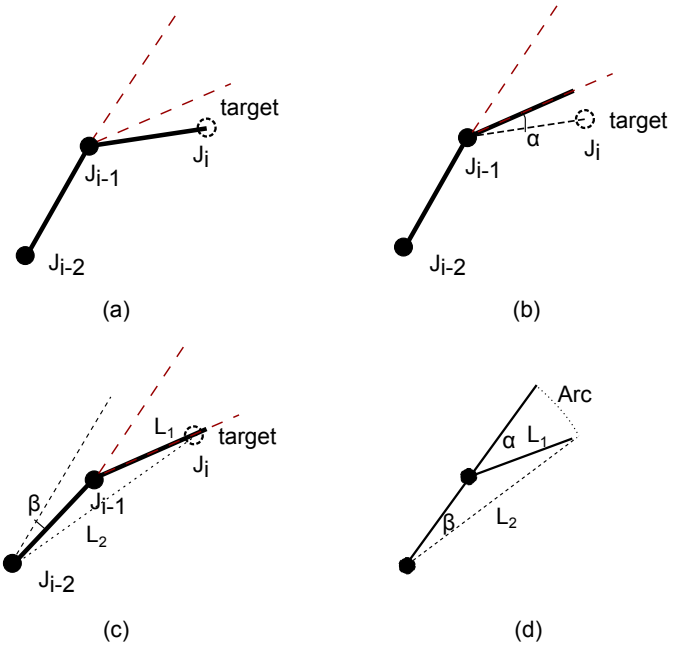

FIGURE 10 – Exemple de constraintes : **(a)** L'effecteur final J_i atteint la cible, mais la rotation de J_{i-1} dépasse les contraintes, représentées par la ligne pointillée. **(b)** Rotation de J_{i-1} jusqu'à la limite la plus proche avec l'angle α. **(c)** Rotation de J_{i-2} pour déplacer l'effecteur final vers la cible avec un angle β. **(d)** Approximation de β.

L'idée principale de notre méthode consiste à corriger l'orientation locale, de propager les valeurs excédantes au parent et, de manière itérative, converger vers un

résultat compatible avec les contraintes du système. Nous appliquons notre solveur de contraintes en vérifiant les valeurs de rotation de toutes les articulations (voir figure 10). L'ordre de vérification des contraintes est de l'effecteur final jusqu'à l'articulation définie comme la racine du squelette. La rotation doit être décomposée selon 3 axes.

Si la rotation dans l'espace local est supérieure à l'angle limite maximal ou inférieur à l'angle limite minimal, cela signifie que la valeur de rotation dépasse à la valeur de contrainte. Nous appelons cet événement un *événement contrainte*. Dans ce cas, nous coupons la valeur de dépassement de J_{i-1} et effectuons une seconde rotation β pour J_{i-2} de manière à repositionner l'effecteur final près de la cible. On peut facilement calculer β grâce à l'approximation suivante : $Arc = \alpha L_1 \approx \beta(L_2)$. Ensuite, nous vérifions la rotation locale de J_{i-2}.

Nous procédons de façon ascendante, en poussant les valeurs excédantes d'un joint à son parent au cas où la contrainte n'est pas satisfaite jusqu'à ce que l'articulation de base soit atteinte. Si la racine déclenche également l'événement contrainte, nous ne coupons que la valeur excédante et redémarrons la boucle complète jusqu'à ce qu'une solution soit trouvée où le nombre maximum de boucles soit atteint. Ce processus itératif permet de résoudre les contraintes générales et de garder l'effecteur final le plus proche possible de la cible.

Résultats

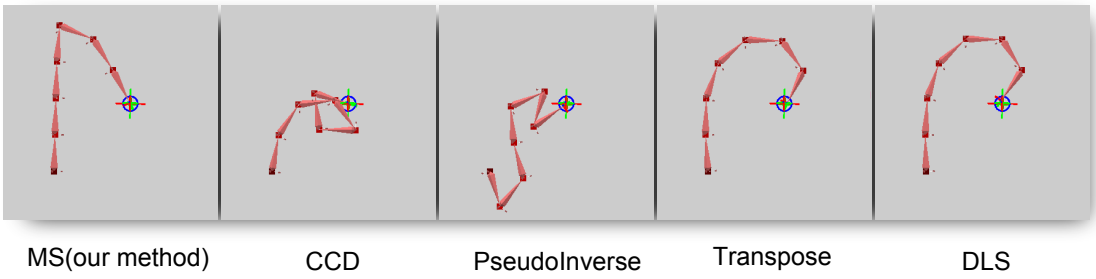

FIGURE 11 – Résultats en utilisant différentes solutions de cinématique inverse.

Les résultats sont illustrés dans la figure 11. Par rapport à d'autres méthodes, notre méthode donne de bons résultats avec moins de changements au niveau des articulations par rapport aux états initiaux. Plus particulièrement, si nous comparons notre méthode avec CCD, notre méthode ne montre pas les artefacts "rolling" présentés dans la solution CCD (voir la Figure 12)

Nous avons également comparé les performances de ces méthodes avec notre méthode (nous mesurons tout un processus incluant les calculs de position et les calculs d'orientation), comme indiqué dans le tableau 6. Notre méthode est aussi efficace que CCD, mais beaucoup plus rapide que les autres méthodes.

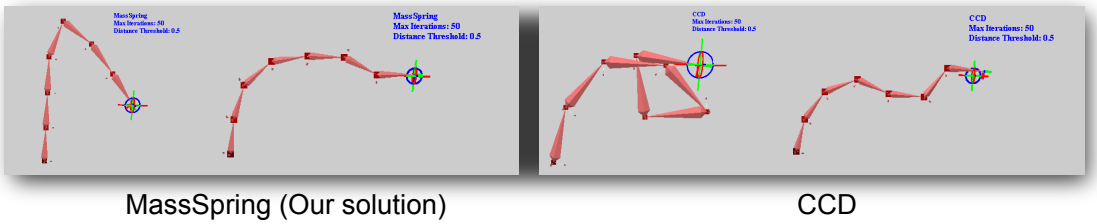

FIGURE 12 – Une comparaison entre notre solution et la méthode de CCD : CCD peut se retourner sur lui-même.

Méthode	Jointures	Itérations	Temps (ms)
notre Méthode	6	26.9	0.132
sur Position	10	40.1	0.169
CCD	6	33.2	0.131
[137]	10	39.7	0.148
PseudoInverse	6	51.8	1.095
[116] [140]	10	55.9	1.310
Transpose	6	6.8	0.124
[144]	10	17.1	0.235
DLS	6	55.6	1.237
[25]	10	68.7	1.450

TABLE 6 – Comparaison de performances entre notre méthode et d'autres méthodes sans contraintes pour une chaine de 6 et de 10 jointures. Les résultats sont obtenus sur un échantillon de 15 tests en moyenne.

Discussion

Nous avons présenté un nouvel algorithme basé sur la physique qui utilise un système masse-ressort. Il fournit des résultats de haute qualité visuelle avec des temps de calcul qui sont comparables et parfois meilleurs que les méthodes existantes. Mais notre méthode souffre de l'oscillation qui dérive de l'utilisation du système masse-ressort. Même si de petites oscillations n'ont pas d'influence sur la posture finale après quelques itérations, nous avons besoin de configurer les paramètres du système masse-ressort pour réduire les oscillations. Nous avons fusionné ce modèle avec un modèle de paramètres d'expressivité pour pouvoir animer des personnages humains.

Pipeline d'animation expressive procedurale

Les Agents Conversationnels sont des agents humains virtuels qui peuvent communiquer par la voix, les expressions faciales, les gestes, les mouvements émotionnels du corps, etc. Il est nécessaire que ces agents puissent afficher une grande variété de comportements. La génération d'animations efficaces et réalistes de créatures virtuelles est toujours un problème difficile dans le domaine de l'animation par ordinateur. Dans la plupart des systèmes existants, l'animation des différentes parties du corps se font indépendamment les unes des autres. Par exemple, un geste du bras et un mouvement du torse sont calculés séparément, puis combinés. Un tel calcul donne lieu à une animation dénaturée et raide.

Notre travail se concentre sur la réalisation de l'animation pour les agents conversationnels virtuels. Notre modèle d'animation prend en entrée une séquence de comportements multimodaux à générer. Elle s'appuie sur une solution cinématique hybride pour générer la posture du corps entier. De plus, notre solution génère deux types de mouvements. D'une part, elle calcule tous les mouvements indiqués sur chaque modalité (par exemple, le bras, le torse ou la tête du mouvement). D'autre part, elle considère également les mouvements liés à d'autres modalités. Par exemple, elle va générer un mouvement de l'épaule et du torse résultant d'un mouvement de bras donné (par exemple, lorsque le bras doit atteindre un point éloigné dans l'espace). Considérer à la fois les mouvements liés et interdépendants peut mener à une animation plus réaliste et naturelle. Notre modèle intègre également un module expressif avec des variations qualitatives des mouvements du corps. Notre algorithme est efficace : il peut générer des animations réalistes du corps entier en temps réel.

Notre pipeline

Nous avons mis en place un système d'agents conversationnels qui respecte le standard Saiba [68] illustré à la figure 13. Notre système prend en entrée un fichier décrit en FML (Functional Markup Language), le langage de balises standard qui définit les intentions et les états émotionnels. Le planificateur de comportement traduit les balises FML dans des séquences de la norme BML (Behavior Markup Language), langage de balises, qui définit les séquences de comportement [68]. Les séquences de signaux de BML possédant des références temporelles sont instanciées dans le Réalisateur de comportement. En nous basant sur ces définitions de comportements haut-niveau, nous utilisons notre pipeline d'animation pour construire des séquences d'animation bas-niveau.

Notre pipeline d'animation commence par la réception des signaux symboliques dans le planificateur de mouvement. Puisque tous les signaux sont reçus en streaming, nos calculs d'animation doivent être réalisés à la volée. Chaque signal (main, torse, la tête, etc) comporte des phases, des paramètres d'expressivité, une trajec-

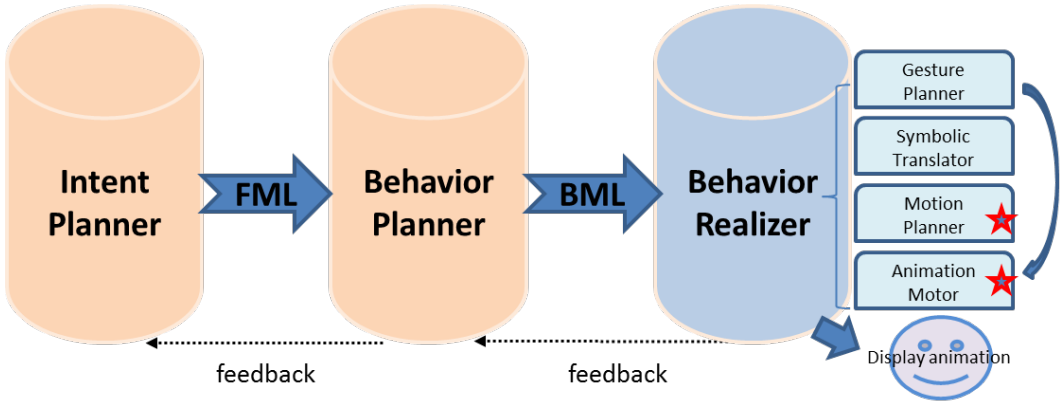

FIGURE 13 – Le pipeline standard SAIBA définit les modules, fonctions et protocoles dŠune architecture dŠACAs.

toire et la description de la forme et du mouvement. Les signaux reçus sont envoyés dans notre pipeline pour générer une séquence d'animation, comme illustré dans la figure 14. Des descriptions directes sont utilisées pour définir la cinématique directe (FK), ces données pouvant être extraites avec des méthodes de capture de mouvement. Le processus de ciblage recalcule la trajectoire des gestes en utilisant les paramètres dŠexpressivité gestuelle. Ensuite, nous réunissons les séquences d'animation individuelles pour chaque partie du corps (tête, torse, les gestes, etc) en un unique ensemble couvrant tout le corps. Nous utilisons la cinématique inverse pour définir les états initiaux du squelette de l'agent. Nous définissons plusieurs postures par défaut, qui sont également utilisées pour les phases de détente définies dans les phases de geste. Notre méthode de cinématique inverse termine le calcul des images clés pour la posture du corps. Ensuite, il effectue un reciblage avec les paramètres d'expressivité. La dernière étape de notre pipeline est la génération des images d'animation à partir des images clés et enfin la conversion de ces images d'animation en BAP (format d'animation du corps de la norme MPEG-4) pour animer notre agent conversationnel virtuel.

Expressivité

Nous avons défini un ensemble de paramètres d'expressivité pour moduler la qualité des mouvements du corps. Nous regroupons ces paramètres en 3 séries : 'Volume de geste' contrôle la variation spatiale des gestes ; 'Variation séquentielle' contrôle les variations temporelles des gestes ; 'Variation de puissance' nous permet de mieux contrôler la dynamique de ces mouvements.

Volume de Geste L'idée du volume de geste est d'utiliser certains paramètres pour contrôler la posture spatiale de posture. Le paramètre spatial est utilisé pour

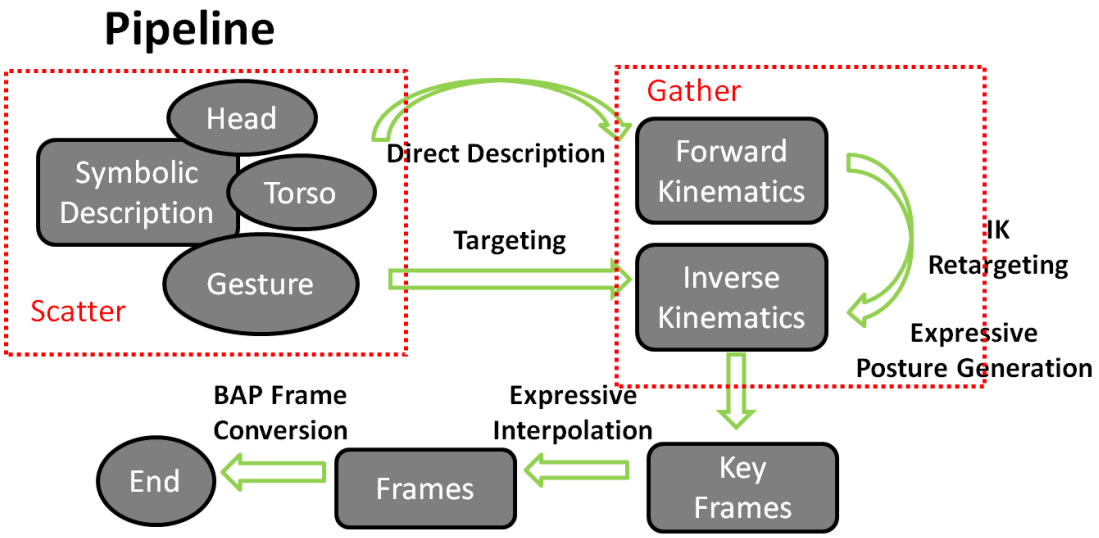

FIGURE 14 – La pipeline d'animation prend en entrée des séquences de gestes symboliques. Il calcule toute l'animation avec des paramètres dŠexpressivité.

contrôler la variation des secteurs de McNeill [88] (voir la Figure 15). Nous définissons aussi le paramètre "Ouverture" qui change la forme du geste calculé à l'étape de cinématique inverse. Sa valeur varie entre 0 et 1. Une petite valeur resserre le corps, et rapproche le coude près du centre du corps, et une grande valeur augmente l'espace entre les bras et le torse, comme illustré dans la Figure 16 à gauche. Le paramètre "Ouverture" influe sur l'orientation de l'angle de pivotement du coude [131]. Une valeur plus grande du paramètre "Ouverture" applique également une plus grande rotation sur le torse qui libère indirectement la tension des bras, comme dans la figure 16 à droite.

Variation séquentielle Trois paramètres, "Fluidité", "Puissance" et "Tension", sont utilisés pour moduler la trajectoire de geste ainsi que sa dynamique. "Fluidité" désigne le degré de continuité d'un mouvement. Pour le simuler, nous utilisons l'idée proposée par [28] [47] de paramétrer les splines de Kochanek Bartels (TCB splines) [64]. Nous définissons "Puissance" comme une force qui modifie implicitement la vitesse avec une accélération. "Tension" [112] décrit la quantité d'énergie qui doit être dépensée pour maintenir certains positions, des positions ayant besoin de plus ou moins dŠénergie, et indiquant donc que plus d'effort doit être exercé pour maintenir la position dŠorigine. Ces deux derniers paramètres sont simulés en faisant varier le paramètre de "Biais" et la tension de paramètre des splines TCB. Nous avons également simulé des accélérations en utilisant diverses fonctions de "easing in-out" pour modifier l'horodatage des images clés (interpolation pour la trajectoire) et les cadres-locals (interpolation pour une rotation d'articulation).

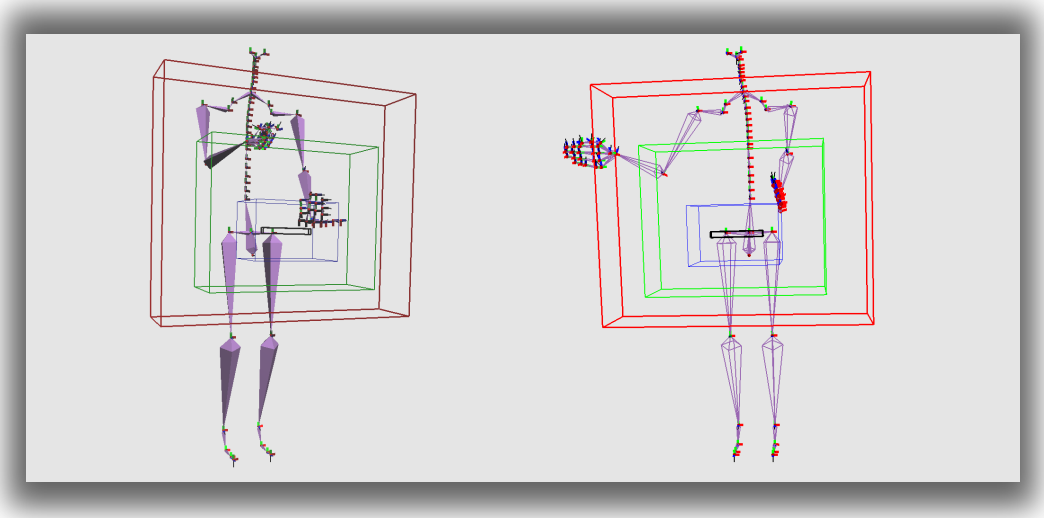

FIGURE 15 – Les secteurs de McNeill sont définis en face de notre personnage virtuel et définissent l'espace de gestes.

Variation de puissance Dans notre système, le paramètre de puissance n'affecte pas seulement la vitesse du geste, mais aussi les postures des images clés. Cela signifie quŠun geste avec une plus grande valeur de puissance influe plus sur le corps. Par exemple, un mouvement du bras fait avec une puissance élevée affecte également les épaules et le torse. Cette propagation de mouvements entre les parties du corps (les épaules et le torse) est possible avec notre solution hybride de cinématique inverse.

Résultats de notre agent virtuel

Le pipeline proposé a été intégré dans notre système d'agent virtuel. Le système de squelette est basé sur le modèle de MPEG-4 H-ANIM 1.1. On peut noter que les mouvements du torse et des épaules peuvent être générés automatiquement avec le processus de propagation de mouvements. De tels mouvements peuvent se produire implicitement sans qu'aucun mouvement du torse ne soit défini par des balises BML. Les paramètres d'expressivité peuvent modifier un geste donné et faire varier les mouvements de nos personnages virtuels(voir la Figure 17).

Discussion

Dans ce travail, nous avons présenté un pipeline d'animation expressive pour le corps complet utilisant une méthode procédurale. Nous avons proposé une approximation hybride pour le calcul de posture basée sur l'interdépendance des différentes

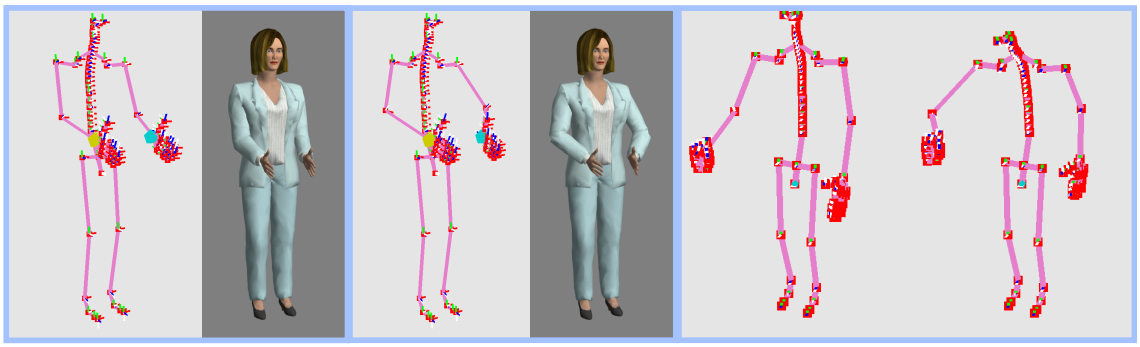

FIGURE 16 – Exemples avec différentes valeurs différentes de "Ouverture" : **(Gauche)** : "Ouverture" influence la forme de geste. **(Droit)** : "Ouverture" influence la position de torse.

parties du corps. Notre méthode expressive offre des résultats visuels de haute qualité en temps réel. Ce système offre une plus grande flexibilité pour configurer des cinématiques expressives. Il peut être étendu à d'autres types de figures articulées.

Conclusion de Thèse

Dans l'ensemble, nous avons proposé plusieurs nouvelles techniques et des extensions de méthodes pour le rendu et l'animation : (i) Nous avons exploré la technique espace-écran pour simuler les effets d'éclairage, qui peut améliorer à la fois la vitesse et la qualité visuelle de rendu en temps réel. La technique d'espace-écran peut être appliquée pour les rendus généraux comme lŠOcclusion Ambiente, et aussi les rendus spécifiques comme la simulation de rides pour le visage human. Nous montrons que notre rendu améliore la qualité visuelle, et la perception d'animations émotionelles. (ii) Nous avons proposé un pipeline d'animation expressive en utilisant une solution de cinématique inverse efficace, dans lequel nous avons introduit des mouvements relatifs. Des paramètres d'expressivité sont utilisés pour modifier à la fois les postures d'images clé et la séquence d'animation. Notre solution améliore la qualité des mouvements du corps d'agents virtuels tout en atteignant des performances rapides. Nous tenons à poursuivre notre recherche et à contribuer à la qualité des algorithmes efficaces pour l'infographie.

```
<gesture category="test" id="test">
    <phase type="STROKE-START">
        <hand distanceFixed="true" fingersOrientationRatio="1.0"
            handShapeFixed="true" horizontalFixed="true"
            palmOrientationRatio="0.0" side="Right" verticalFixed="true">
            <verticalLocation>YUpperC</verticalLocation>
            <horizontalLocation>XC</horizontalLocation>
            <locationDistance>ZNear</locationDistance>
            <handShape>form_fist</handShape>
            <palmOrientation>INWARD</palmOrientation>
            <fingersOrientation>AWAY</fingersOrientation>
            <palmOrientationSupplementary>DOWN</palmOrientationSupplementary>
            <fingersOrientationSupplementary>AWAY</fingersOrientationSupplementary>
        </hand>
    </phase>
    <phase type="STROKE-END">
        <hand distanceFixed="true" fingersOrientationRatio="1.0"
            handShapeFixed="true" horizontalFixed="true"
            palmOrientationRatio="0.0" side="Right" verticalFixed="true">
            <verticalLocation>YCC</verticalLocation>
            <horizontalLocation>XP</horizontalLocation>
            <locationDistance>ZFar</locationDistance>
            <handShape>form_3apart</handShape>
            <palmOrientation>INWARD</palmOrientation>
            <fingersOrientation>AWAY</fingersOrientation>
            <palmOrientationSupplementary>DOWN</palmOrientationSupplementary>
            <fingersOrientationSupplementary>AWAY</fingersOrientationSupplementary>
        </hand>
    </phase>
</gesture>
```

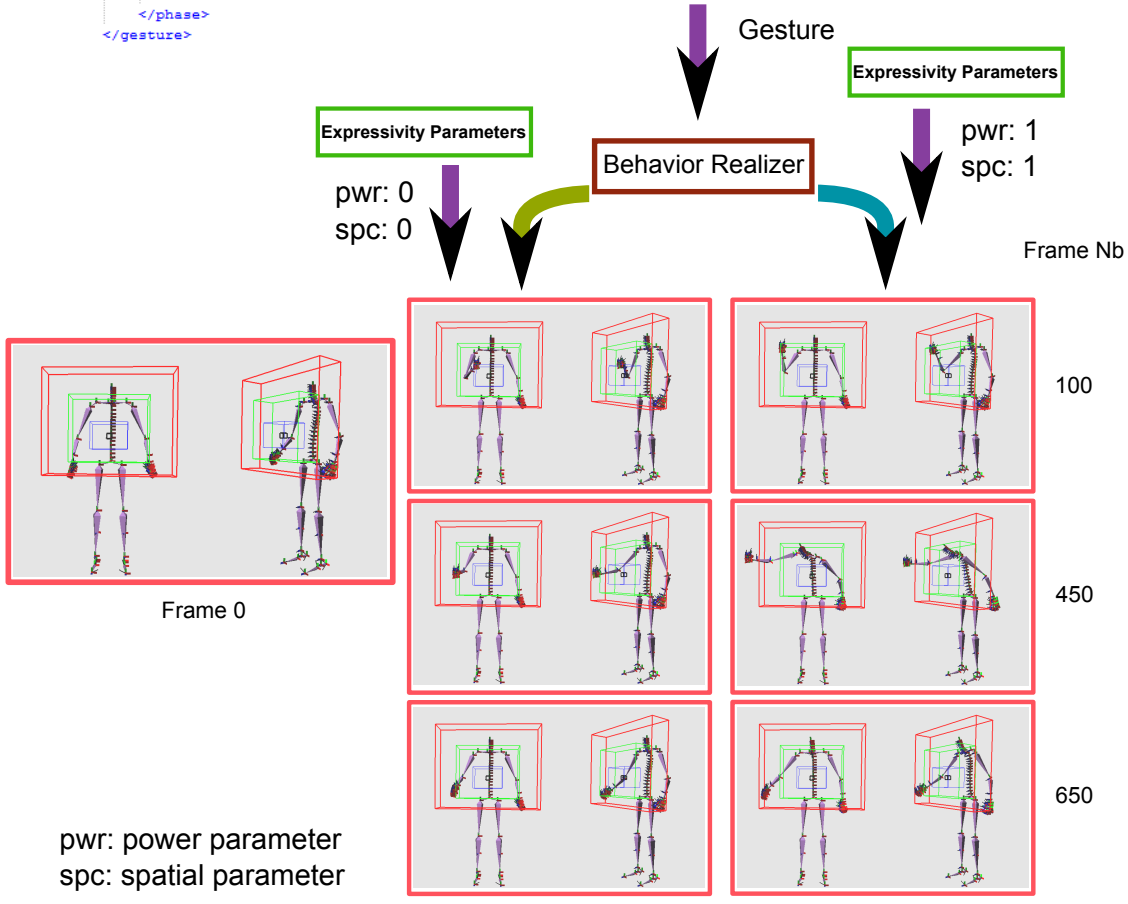

FIGURE 17 – Deux mouvements différents sont créés par un même geste avec différents paramètres d'expressivité.

Chapter 1

General Overview

1.1 Motivation

Animation and rendering are two main research areas in the computer graphics field. Animation is the study of motion of virtual objects. Rendering converts the virtual 3D scene into 2D image. Many techniques have been proposed to improve their quality and performance, depending on their aim and context. On the one hand, for offline purposes, such as film production, the quality of final results is the most important factor to be evaluated. On the other hand, the low cost approximation is first considered for real-time applications like computer games. In this thesis, we explore real-time techniques, both in rendering and animation. The aim is to achieve a good visual quality with high speed performance for 3D visualization system. We also make use of our 3D animation and rendering methods in this thesis project, to introduce new expressive factors into our models.

This PhD topic takes place within the CECIL project, an ANR project which started in September 2009 and lasted 3 years. The aim of the project CECIL is to contribute to the area of interactive systems by placing emotion at the heart of the interaction between the system and the human users. To meet this general objective, the project has defined clearly (and disambiguate the definition of) a set of emotions, and has incorporated them into the reasoning processes of an embodied conversational agent (ECA). The whole process is built into an ECA system, called «**Greta**».

«Greta»

Our «**Greta**» system is based on a mature applicable ECA model. In particular, the system has been endowed with the capabilities to express its emotions by means of different modalities including facial expressions, gestures and language. Mainly, the system takes as input a text to be said by the agent. The text has been enriched with information on how the text ought to be said (i.e. with which communicative acts it should be said). The behavioral engine computes the synchronized verbal and nonverbal behaviours of the agent. Finally, the expressive behaviours of the agent can be observed through different terminals. In other words, the whole ECA system has to determine which expressions to show in a particular context and be able to display them.

The first agent of our ECA system is called «**Greta**». She is a real-time three dimensional embodied conversational agent with a 3D model of a woman compliant with MPEG-4 animation standard. She is able to communicate using a rich palette of verbal and nonverbal behaviours. Greta can talk and simultaneously show facial expressions, gestures, gaze, and head movements. Two standard XML languages FML and BML allow the user to define her communicative intentions and behaviours based on standard SAIBA architecture [68]. Greta can be used with different external TTS software solutions. Currently she can speak various languages: English, Italian, French, German, Swedish and Polish. She is used in various European projects: IL-HAIRE, VERVE, REVERIE, TARDIS, SSP-Net and national French projects: Feder Anipev, ANR IMMEMO, ANR CECIL, GV-LEx, etc.

Figure 1.1: The first agent named **Greta** in our ECA system

The goal of my work is to improve our embodied conversational agent (ECA) sys-

tem which is capable of displaying expressive emotional behaviours. To this aim, we need to build qualitative models for the existing multimodal expression of emotions system. Our models can deal with any of multimodal signals such as facial expressions, gaze direction (eye and head direction) and gestures. The models must also consider behavioral expressivity. Emotions are expressed through the whole body and not solely through facial expressions. The model of emotional behaviours should be extended to the whole body and it should be dynamic. Gestures, postures and gaze direction are signs of emotions. The multimodal behaviours are synchronized with each others to produce a coherent message. Moreover, the expression of emotion does not correspond to a static expression. Its behaviour is dynamic, and evolves through time. This condition constrains our system to be run totally in real-time. For this purpose, we focus on a real-time graphics pipeline with both animation and rendering. For a general introduction to computer graphics, we refer the reader to [38].

1.2 Outline

In rendering, the first step is to determine a way that takes large amount of 3D scene data and builds their visual representation by conserving the scene properties like geometry, lights, materials, environment, and even personalities for virtual characters. In the real world, all these properties are perceived through interactions between objects and lighting. Ideally, the virtual scene can be built by using such physical models. Direct lighting represents the direct interaction between light sources and virtual objects. It is easy to simulate with simple models. But the simulation result is far from realistic. Indirect lighting is a supplementary phenomena which catches the light bounces both between surfaces and inside shape (e. g. , skin subsurface scattering is an important phenomenon for realistic agent rendering). It improves the visual quality, but its realization is much more complex. For offline rendering, photorealistic renderers can achieve impressive results that represent an accurate physical model, but it is not possible to achieve interactions with such methods. Real-time simulation requires faster approximation models. The trade-off between visual quality and speed is essential here.

We start our work by establishing the state of the art in rendering in the field of screen space rendering techniques [80] [115] [87]. Beyond the use of ambient

occlusion [80] and one-bounce global illumination [115], then a particular focus will be set on the development of a new screen space formulation of subsurface scattering [57] [89], adapting the notion of Gaussian diffusion in texture space [34] to a more general case. A first approach, based on anisotropic depth image diffusion will be explored, using recent advances in fast bilateral filter approximation [109] [27] [2] to propose scalable realistic materials featuring significant subsurface scattering effects. This part of this thesis focuses on how screen space and object/texture space methods can be balanced [114] to define a proper efficient screen space model.

Animation is another component of the graphics pipeline. Rigid object transformations can not generate plausible motion representations. Given a surface mesh to be animated dynamically, the updating process on vertices is neither fast nor easy to control. In the film industry, CG artists often use morph targets technique to generate facial animations since these methods are robust and offer better quality as well as a high degree of fidelity for expressions. However, they are too slow for high quality motion and require intensive artist interaction. On the other side, for real-time computation motives, such as gaming or conversational virtual agents, the flexibility and dynamic control need to be explored. Bones or muscles systems have therefore been proposed. For example, the Facial Action Coding System (FACS) is used as parameters schema in different facial animation system where Action Units (AUs) [106] are defined to control muscle contractions on the human face. Rigid bones combined with skinning techniques can offer more efficient control of the character's virtual body. The most common way to manipulate such a skeletal system is to employ a data-driven or procedurally controlled system. In the procedural domain, Inverse Kinematics (IK) is widely used. Current IK methods can achieve this work in real-time, but the convergence is still not efficient enough or cannot generate satisfying results easily. For virtual character gesture simulations, the expressive quality of gesture motion is more required than just solving one chain.

On a higher level, creating a variety of multimodal body performed animations is challenging [3] [125] [132] [107] [108] . Animations need to be more human communicative and expressive. But it is hard to create natural-looking animation sequences for virtual characters. Several existing frameworks can generate animation separately for each module defined by body parts. They solve the motion

locally without interactions. Some other works have also achieved to create expressive motions. They parameterize postures and motion sequences by considering different factors. It is often the case that a trade-off between dynamic interaction and (perceived) realism must be found. Moreover, all the systems for emotional interactions [118] [12] need to be totally real-time. Since all computations need to be online, expressions will be viewed as dynamic behaviours and no more static ones [101]. Thus, in the second part of this PhD thesis, we focus on real-time animation for expressive characters.

In the end, the combination of both dynamic human body animation and realistic rendering will be proposed to provide scalable expressive avatars within our projects and the GRETA system.

1.3 Contributions

In this thesis, we present several efficient solutions for real-time rendering and real-time animation in the context of ECA. The 3D data can be generated and updated by our expressive animation pipeline using procedural methods. Our renderer takes the updated 3D data as input. The final 3D models can be explicitly converted into a 2D image sequence. The pixels of each image are shaded with satisfying visual quality approximations in screen space. While populating the Greta systems with these features, we make the following technical contributions:

Separable Approximation of Ambient Occlusion In this part of the thesis, we present a fast easy-to-implement separable approximation to screen space ambient occlusion (see chapter 3) (AO). We evaluate AO for each pixel by integrating angular values of samplers around the pixel position which potentially block the ambient lighting. We use a separable mechanism to reduce the complexity of the evaluation. Computing occlusion first along a single direction and then transporting this occlusion into a second pass that is stochastically evaluating the final shading based on the AO estimates proves extremely efficient. Combined with interleaved sampling and geometry-aware blur, visually convincing results close to a non-separable occlusion can be obtained at much higher performance.

Identification of Single Facial Actions with the help of Screen Space Animated Wrinkles In our «**Greta**» system, we use Facial Action Coding System (FACS) for facial animation. Different Action Units (AUs) are defined and combined to represent movements on the human face. To identify expressions through facial animations is important when studying virtual agent communications. We set up an evaluation system to test several factors that may influence the identification by each single action unit. We also built a wrinkles model by using a modern graphics technique which performs computations only in screen space (see chapter 4). With the help of wrinkles, the facial animation can be more realistic, and AUs can be recognized with a higher rate.

Mass-Spring Based Inverse Kinematics Inverse Kinematics (IK) is a common tool for controlling skeleton systems. In this part of the thesis, we present a fast and easy-to-implement locally physics-based Inverse Kinematics(IK) method. Our method builds upon a mass-spring model and relies on force interactions between masses. We solve the IK problem in the position space instead of the orientation space (see chapter 7). Joint rotations are computed using the closed-form method with predefined local axis coordinates. Combining these two approaches offers convincing visual quality results obtained with high time performance. Moreover, compared to other methods, it is even more competitive when using multiple positional constraints. Our IK solver is suitable for multiple constraints application and constrained 3D humanoid models.

Expressive Kinematics Procedural Animation Pipeline In this part of thesis, we propose our expressive body-gestures animation synthesis model for our Embodied Conversational Agent (ECA) technology. Our implementation builds upon a full body reach model using a hybrid kinematics solution. We describe the full pipeline of our model that starts from a symbolic description of behaviours, to the construction of a set of key frames till the generation of the whole animation enhanced with expressive qualities. Our approach offers convincing visual quality results obtained with high real-time performance.

All these contributions have been interpreted and experimented into the rendering and animation components of Greta system. It will be released in an upcoming version of the system.

1.4 Thesis Organization

The presentation of these contributions are divided into Rendering (Part I) and Animation (Part II). Each part starts by introducing the backgroud in the specific domain, followed by the presentation of our contributions. We close the thesis by a conclusion and our perspectives how future research could exploit it (Part III).

Part I

Rendering

Chapter 2

Rendering Background

Rendering is a sub-field of computer graphics dedicated to generating images from 3D models. It differs from image processing as it focuses on synthesizing color information. Therefore, rendering an image requires modeling the camera parameters, the shapes, appearances and the lighting in the environment. The main processing steps are solving for visibility, evaluating reflectance and applying post-processing to the image.

3D models typically come as a collection of primitives, called a mesh (triangles for triangle mesh). Each mesh can represent the shape of a specific object. A mesh is composed of vertex information, normal vector information, connectivity information.

Two methods are widely used to generate an image of a 3D scene:

– Ray-Tracing [4] [141]: The basic idea is to shoot rays from a camera, and find the closest object which blocks the ray's path. The pixel's color is the value of the object at the intersection position.

– Rasterization [16] [146]: This method is based on the projection from 3D to 2D. It defines 3D objects discretization to fill the corresponding pixel with a particular reflectance response which depends on visibility, lighting and material properties. This process is highly optimized and programmable on modern GPU architecture. Hidden surfaces need to be tested to avoid drawing invisible primitives. Sutherland et al. [126] developped the early Scan line HSR (Hidden Surface Removal) algorithm to create renderings of solid objects. Until now, HSR techniques such as back-face detection, depth sorting, ray casting and the Z-Buffer are still widely used in modern computer graphics.

In this thesis, we present several real-time rendering algorithms, each of which uses the GPU rasterization pipeline.

The color at a given point, as seen from a given view (e. g. , pixel color) can be formulated by the rendering equation [59]:

$$L_o(\mathbf{p},\omega_o) = L_e(\mathbf{p},\omega_o) + \int_\Omega f_r(\mathbf{p},\omega_i,\omega_o) L_i(\mathbf{p},\omega_i)(-\omega_i \cdot \mathbf{n}) d\omega_i \tag{2.0.1}$$

where p is a point on a surface, Ω is a hemisphere surrounding p. L_e is the emitted radiance from the object. L_o is the existant radiance from p in a direction ω_o. L_i is the incident irradiance from the angle ω_i. f_r is a bi-directional reflection distribution function (BRDF) which describes the material properties of p.

The BRDF model describes how light bounces off the surface. A lighting model can often be precised separately with direct lighting and indirect lighting. Moreover, programmers like to use 3 distinct light terms: ambient, diffuse, specular.

2.1 Reflection Model for Shading

Several smooth shading models have been proposed for meshes in the 1970s (see Figure 2.1). In Gouraud shading [41], the lighting is calculated per vertex and then interpolated. Phong [113] interpolated the normal vectors linearly and used the interpolated normal at a pixel to determine its color.

As the shading model is often written as $I = Ambient + Diffuse + Specular$, the essential shading method is to compute each component and sum them up to compute the final shading color.

2.1.1 Ambient Lighting

The ambient term in the shading model represents the simplified version of low frequency indirect lighting. For each surface point in the environment, the ambient light comes from all directions. Upon rendering, the most simple way of ambient light computation is to brighten equally all objects in the scene with the specified fixed color. In this case, the ambient light is independent of the explicitly defined light sources in the scene. To trade speed for visual quality, we often use AO (ambient occlusion) [54] to approximate the global illumination for the ambient part.

Figure 2.1: Quality of different shading methods.

2.1.2 Lambertian Reflectance for Diffuse Lighting

Lambertian reflectance [71] is used for modeling the diffuse term (see Figure 2.2). It describes the phenomena that when a purely diffuse surface is viewed from any angle, it has the same apparent radiance. The reflection intensity from a "Lambertian" surface is directly proportional to the cosine of the angle θ between the light's direction vector from the surface point and the surface normal vector.

$$I_D = \mathbf{L} \cdot \mathbf{N} C I_L \tag{2.1.1}$$

where I_D is the intensity of the diffuse reflection, C is the color and I_L is the intensity of the incoming light. $\mathbf{L}$ and $\mathbf{N}$ are the light direction and surface normal vector.

2.1.3 Phong and Blinn-Phong Shading Models

The two methods depictured in Figure 2.2 are used to simulate the specular lighting. Phong shading [113] is formulated as:

$$R = 2(L \cdot N)N - L \tag{2.1.2}$$

$$I = (R \cdot V)^{\alpha} \tag{2.1.3}$$

where R is the reflection direction of light direction L, N is the normal vector, V is the vector from the surface to the camera center, α is the specular shininess, I is

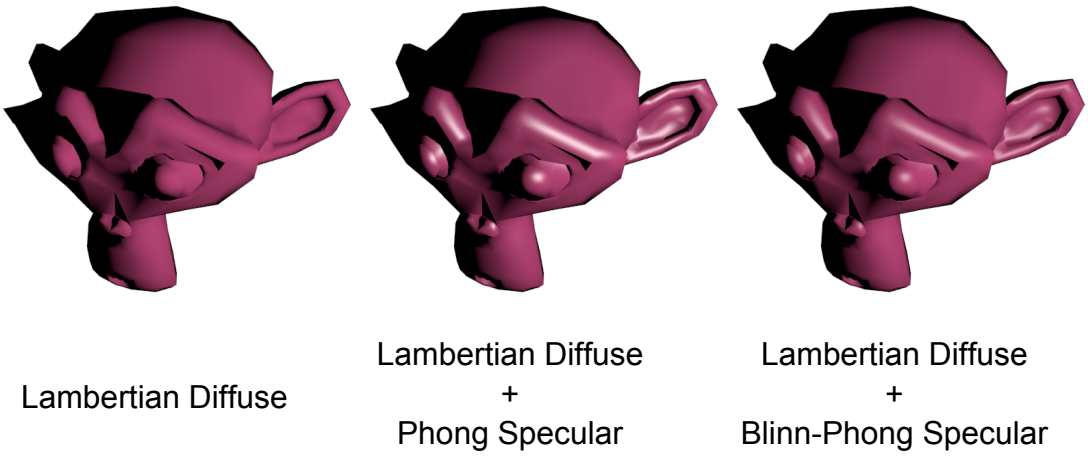

Figure 2.2: Three basic models for defining diffuse and specular lighting.

the final specular intensity.

Blinn-Phong Shading [13] is defined as:

$$H = \frac{L+V}{|L+V|} \tag{2.1.4}$$

$$I = (N \cdot H)^{\alpha} \tag{2.1.5}$$

where H is half angle vector between the light direction L and the view direction V.

Compared to the Phong model, Blinn-Phong shading offers a more accurate approximation of empirically determined BRDFs for many types of surfaces. Although more accurate models (i. e. physically plausible) have been proposed, we mainly use these simple models in this thesis, as they can be evaluated very quickly.

2.2 Screen Space and Deferred Shading Pipeline

In modern rendering strategy, the shading procedure can be reduced to shade only the visible fragements. We use a deferred shading pipeline to lay out all our rendering techniques in screen space. Multiple render targets (MRT) are used to generate intermediate buffers (see Figure 2.3), assembled in a so called Geometric Buffers (called G-buffer) for all necessary per-pixel attributes such as normal, world

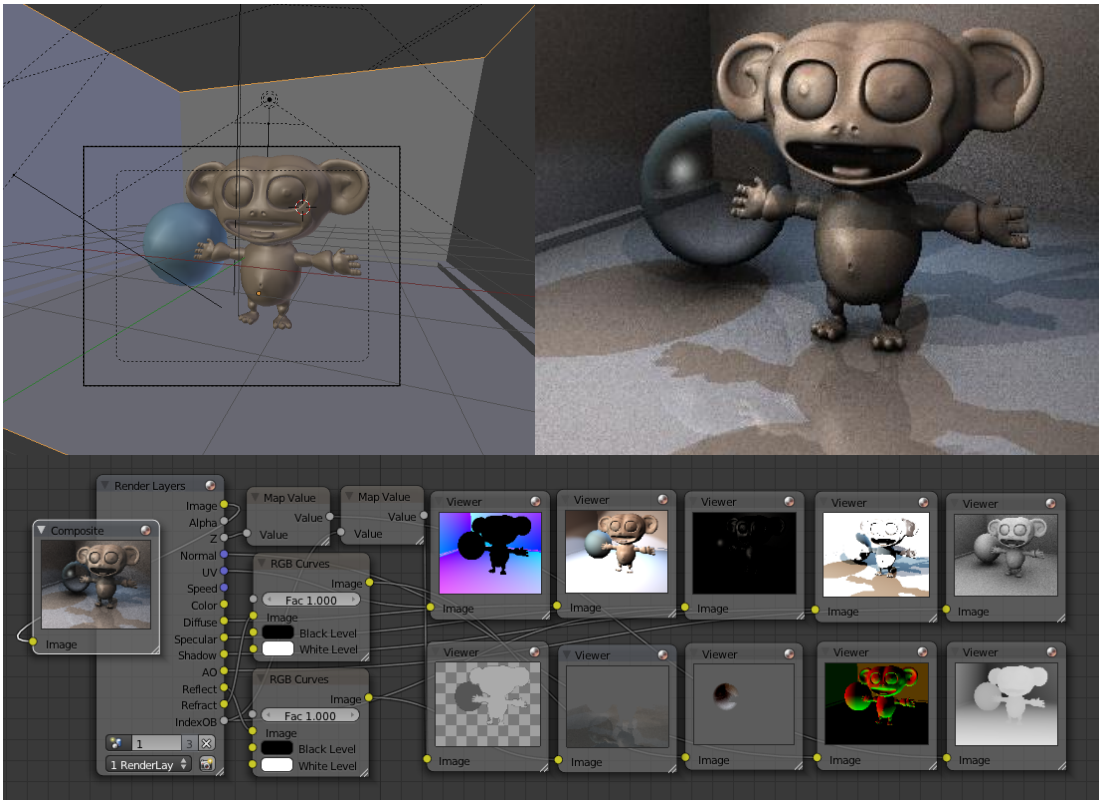

Figure 2.3: Intermediate buffers containing different rendering information in «Blender».

position, albedo and reflectance parameters. Instead of immediately shading scene objects into the final color buffer in the scene rasterization pass, shading is applied in a second pass, where all visible attributes in the frustum are made random accessible, therefore enabling approximations of non local effects. As all the information is saved in the G-buffer textures, costly shading computation only depends on the number of pixels.

This method was first introduced by Takafumi Saito et al. [119] to enhance the comprehensibility and visual quality of 3D images. By using a G-buffers as an intermediate representation, the process of artificial enhancement is abstracted from geometric processes (projection and hidden surface removal) and physical processes (shading and texture mapping), and performed as a post-process.

Our renderer is based on a deferred shading pipeline as described in Figure 2.4.

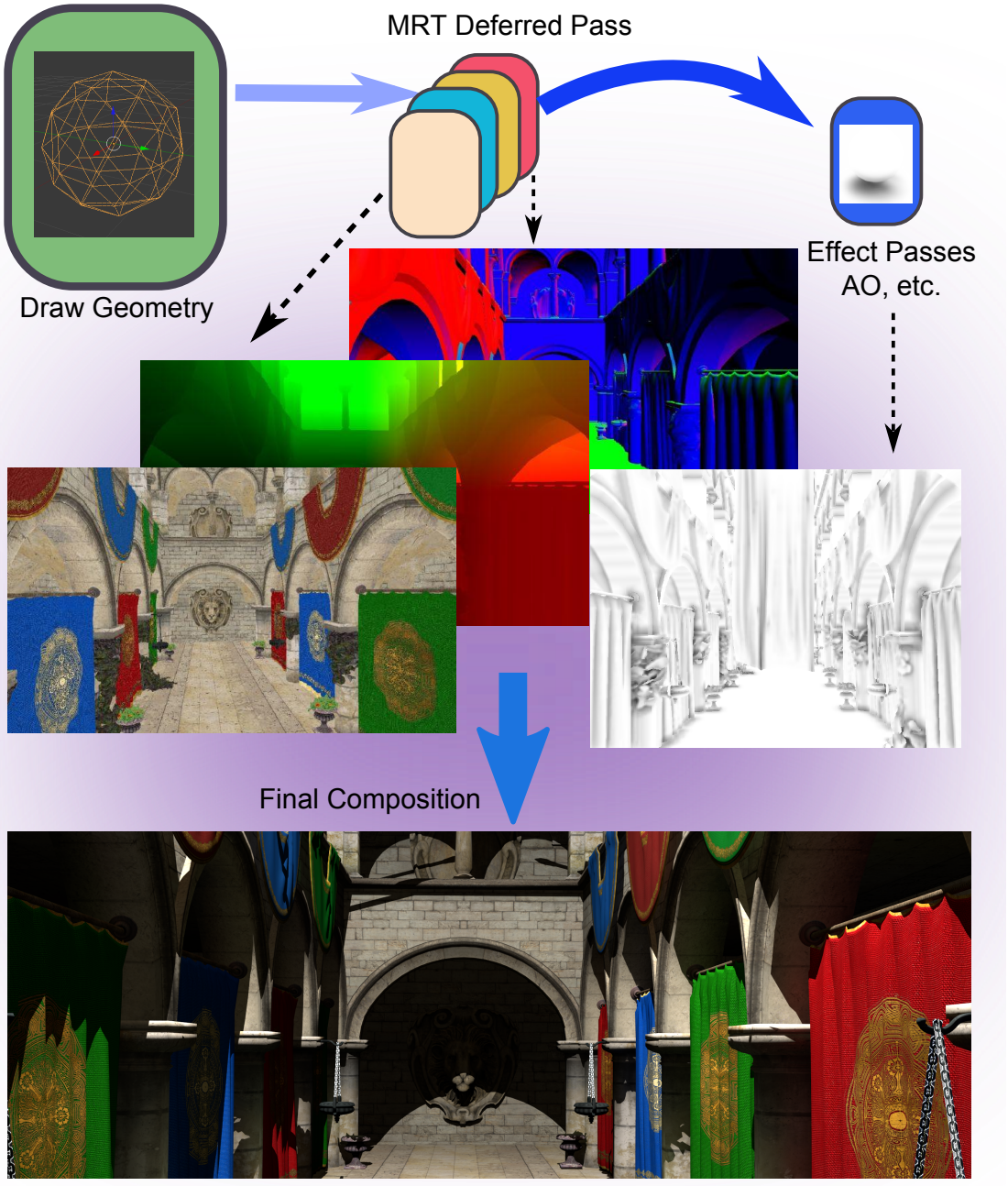

Figure 2.4: Deferred shading pipeline of our «Etoile» renderer.

Chapter 3

Screen Space Ambient Occlusion

The first part of this thesis describes an inexpensive method to simulate global lighting effects. The objective is to enhance the perception of 3D objects in virtual space. Ambient occlusion (AO) is a good method for this purpose, and acts as an economic substitution to the poor quality "constant" ambient term. In particular, we show that the visual quality and speed can be improved to reach even higher performances.

3.1 Introduction

Ambient occlusion (AO) is an effective approximation of indirect lighting that can be computed in real-time for dynamic scenes. As demonstrated in many recent algorithms, it improves the perception of volumes, concavities and contact areas of 3D objects.

AO is a real positive value, defined at every point of a surface as the proportion of occluders present in the vicinity of the point. This value is then used to replace the standard constant ambient term of popular direct shading models. Formally, the AO at a surface element $\mathbf{p}$ with normal $\mathbf{n}$ is defined as the integral over the surface-aligned hemisphere Ω:

$$AO(\mathbf{p},\mathbf{n}) = \frac{1}{\pi}\int_{\Omega} V(\mathbf{p},\omega)\mathbf{n}\cdot\omega\, d\omega,$$

with $V(\mathbf{p},\omega)$ being the *visibility term* from $\mathbf{p}$ in direction ω. Often V is coupled to a falloff function that ensures that distant occluders are ignored.

Figure 3.1: SSAO example in the game «StarCraft 2» 2010 – Blizzard

The AO integral can be evaluated using Monte Carlo integration [72]. Although AO is less expensive than a complete indirect lighting evaluation, it remains too expensive for interactive computation. Therefore, approximations are commonly used for real time applications. In this part of the thesis, we focus on screen space AO (SSAO) [92] which uses the depth buffer as an approximation of the scene, simplifying the formulation to a 2.5D filter.

In this context, fast computations for the visibility function and for the sampling pattern are two extensively studied optimizations. Observing that ambient occlusion is a form of local filter in screen space, we suggest a stochastically inspired separable computation of the 2D visibility that results in images that are similar to the non-separable version, but significantly more efficient.

3.2 Previous Works

Langer et al. [73] [22] were the first to analyze the perception of geometry lit on a cloudy day, which was later introduced as AO to the rendering community by Landis [72]. Zhukov et al. [1] introduced the ambient light illumination model to simulate the indirect diffuse illumination. As well, several approaches [91, 81, 21] successfully used the principle to depict small surface variations. The strength of the resulting shape cues lead to a high interest in AO and its efficient computation. A considerable number of methods were developed along two main branches: *geometry-based* AO and *screen space* AO. Their names do not refer to where AO is computed (vertex or pixel), but to the information considered as occluders – full 3D

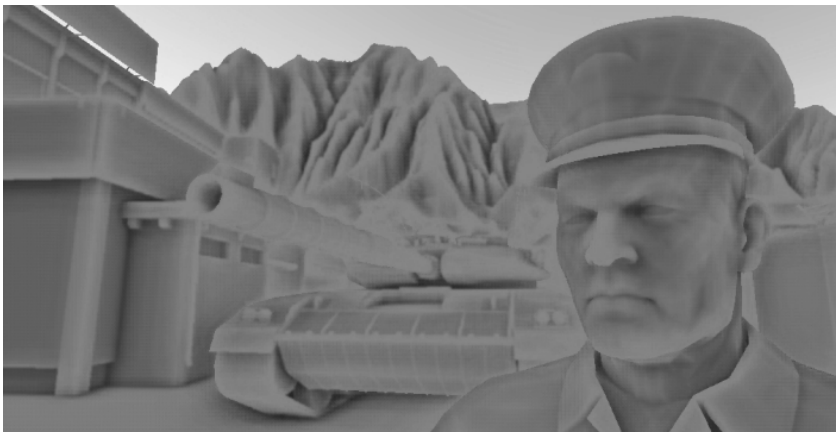

Figure 3.2: Screen-Space Ambient Occlusion in a complete ambient lighting situation from «Finding Next Gen – CryEngine 2».

geometry or view-dependent 2.5D Z-buffers samples.

Geometry-based AO Bunnell [24] presents an AO technique to treat vertices dynamically as receivers and emitters. In a preprocess, a distance-based vertex hierarchy is created that is traversed during runtime. Each vertex gathers shadow values from all emitters in an efficient manner, but the method's performance depends on the geometric complexity.

Kontkanen and Laine [65] precompute AO functions in a cubemap surrounding each object. At runtime, they evaluate these AO functions to produce cast shadows between objects. Their system achieves real-time performance, but requires a long precomputation and large amounts of memory, and is also limited to scenes featuring rigid motions only. Instead of cubemaps, one can use full 3D textures [83].

Reinbothe et al. [114] address fully dynamic scenes by combining object- and image-space computations. They sample occlusion against an on-the-fly voxelization of the scene before applying a feature-sensitive filtering on the AO.

McGuire [84] renders occlusion volumes around polygon primitives that each initiate a fragment shader to compute the occlusion contribution of the entity. The quality is high, but performance is geometry and AO falloff dependent because it depends on the bounding-box sizes.

Screen space AO (SSAO) The idea of SSAO – as introduced in the game Crysis [92] – is that the depth buffer values around every pixel, as seen from the

camera, can serve as a scene approximation in the vicinity of the pixel's world location. AO is evaluated by randomly sampling 3D points around the current pixel. These are projected into the depth buffer and compared to the stored value using a binary (or linear) depth comparison to approximate a volumetric obscurance.

Another such method was presented earlier by Loos and Sloan [79] who replaced point sampling by line or area sampling to better approximate the volumetric occupation in a sphere around the current pixel. Based on an observed relation to the cosine falloff, Ruiz et al. [117] decided to integrate the free space of a tangent sphere shifted in the normal direction above the current pixel.

Shanmugam et al. [120] present two methods for high- and low-frequency occlusion (for nearby and distant surfaces respectively). They define a sphere for each surface pixel in screen space and sample from a Normal-Depth buffer around the current pixel location. These samples are used to define occluders as spherical caps that are projected in direction on the current pixel's unit hemisphere.

Horizon-based ambient occlusion (HBAO) [10] uses the surface's angle of elevation to approximate AO. They compute this angle by summing up the tangent and the horizon angle in view space for a predefined set of screen directions and choose the most representative one.

Finally, Ritschel et al. [115] extend SSAO by considering directional occlusion to simulate bounced light and propose depth-peeling to better handle occluders.

SSAO is usually faster than geometric solutions, but its performance depends on the 2D domain used to evaluate occlusion around a point. E.g., for a disc, a linear growth of the radius implies a quadratic increase in the number of samples. Large regions can be costly, independently of the applied visibility method.

3.3 SSAO Generation

The ambient occlusion is usually defined as the integral of the visibility function, often restricted to a user-defined distance over the hemisphere. Intuitively, it measures how much light might potentially be blocked by nearby geometry. The ambient occlusion is independent of the light sources explicitly defined in the scene. An efficient and basic method of approximating SSAO is to sample the depth buffer around every pixel in the camera direction, and to compare the depth value in a

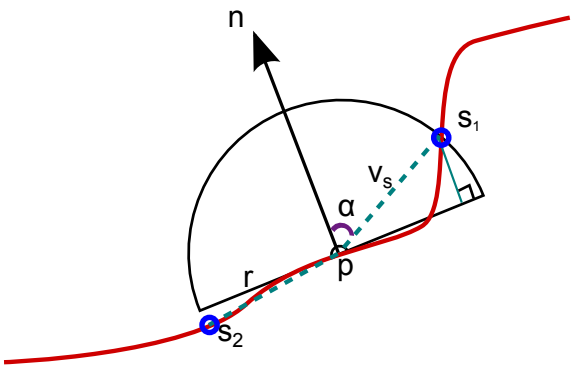

Figure 3.3: Integrating the angular value of each pixel gives a better quality of SSAO. $s_{1,2...}$ are the samples. We compute the angular value α which approximates the occlusion cone for each sample.

binary way like Crytek SSAO does [92].

$$V_{p,s_i} = depth(s_i) > depth(p)\,?\,1 : 0 \tag{3.3.1}$$

where $depth()$ is the depth function for fetching the depth value of a pixel, s_i is the sample pixel. This method is fast, but the quality is low (see Figure 3.2). We propose to substitute the binary comparison with an angular computation between current pixel and neighbor samplers (see Figure 3.3).

$$V_{p,s_i} = k \times (1 - max(\cos\alpha, 0)) \times Fr(d, \epsilon) \tag{3.3.2}$$

Let's consider the current pixel p with normal vector n. The $\cos\alpha$ of a sample pixel s_i around p can be computed by the dot product of the normal vector n with the directional vector v_s, where v_s is equal to the subtraction of the position of s_i and the position of current pixel p. k is an intensity factor, Fr is a distance weighted function with user defined threshold ϵ.

For the distance weighted function Fr, we can choose to use different kernels, such as the Wendland kernel [139], or Gaussian kernel [32]. We use the Wendland kernel, defined as:

$$w(d, \epsilon) = \begin{cases} (1 - \frac{d}{\epsilon})^4(\frac{4d}{\epsilon} + 1) & if\ 0 \le d \le \epsilon \\ 0 & if\ d > \epsilon \end{cases} \tag{3.3.3}$$

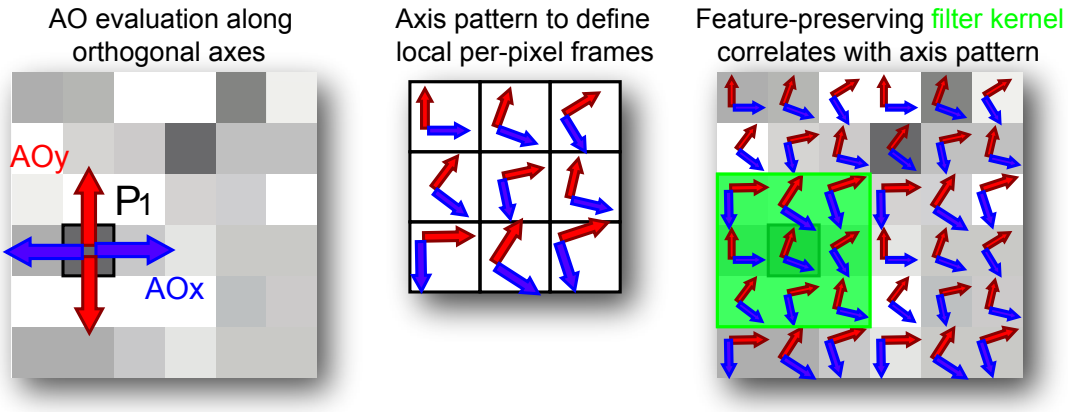

Figure 3.4: **Principle:** Our separable approximation combines two 1D evaluations in orthogonal screen directions (e.g. x, y axis, left). By changing coordinate frames per pixel, we can derive a stochastically valid approximation of the 2D occlusion by combining the result of neighboring samples.

where d is the current distance, ϵ is the maximum or threshold distance.

3.4 Separable Approach

3.4.1 Separable Approach Inspiration

Several methods have inspired us to reduce the complexity of the above-described approach by using a **Separable Approach**. Gaussian filtering [149] is the main example. It can be evaluated in a separable fashion on pictures for instance. 2D computations can be replaced safely by two 1D computations, reducing the complexity from n^2 to n.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} = G(x)\,G(y) \tag{3.4.1}$$

3.4.2 Separable Approximation of SSAO

Consider computing AO at a pixel $\{i,j\}$ at world location $\mathbf{p}_{i,j} \in \mathbb{R}^3$ with normal $\mathbf{n}_{i,j} \in S^2$. We observe that, when formulated in screen space, AO can be understood

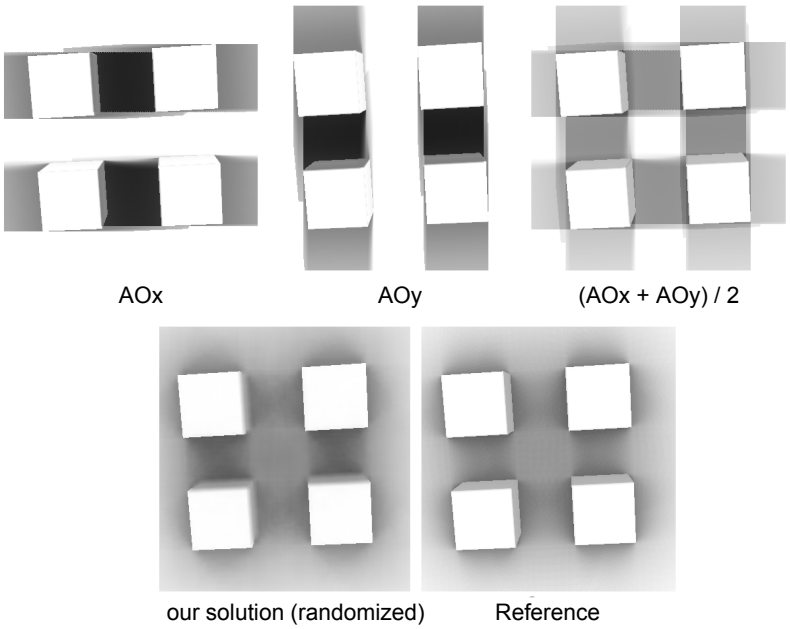

Figure 3.5: Randomizing the notion of "horizontal" and "vertical" axis on a per-pixel basis leads to a good result when compared to the reference AO, while a simple combination of both axes leads to visible artifacts.

as a form of local filtering of the 2D screen buffers (depth, normals):

$$AO(i,j) = \frac{1}{k^2} \sum_{x=i-k/2}^{i+k/2} \sum_{y=j-k/2}^{j+k/2} V(\mathbf{p}_{i,j}, \omega_{x,y}) \mathbf{n}_{i,j}.\omega_{x,y}, \tag{3.4.2}$$

where ω is the point above the pixel on the unit sphere.

As shown by Pham [110] for bilateral image filtering, even when not formally separable, local filters can be efficiently approximated in a separable fashion. In our method, we approximate AO by an x-term and a y-term. Both are evaluated in 1D only (i.e. sampling a half circle of directions).

In a first pass, we compute the occlusion $AO_{\mathrm{x}}(i,j)$ in direction X at every pixel by fixing y in the sum to 0. In a second pass, pixels are occluded by neighbor pixels along the y-axis $AO_{\mathrm{y}}(i,j)$, this time fixing x to 0. Averaging these two partial AO evaluations provides a basic form of separable approximation, but ignores most of the neighboring occluders, i.e. all elements located in diagonal directions (see Figure 3.4). Such potential occlusions are crucial to take into consideration. Our goal is to exploit the visibility computation from neighboring points to derive an

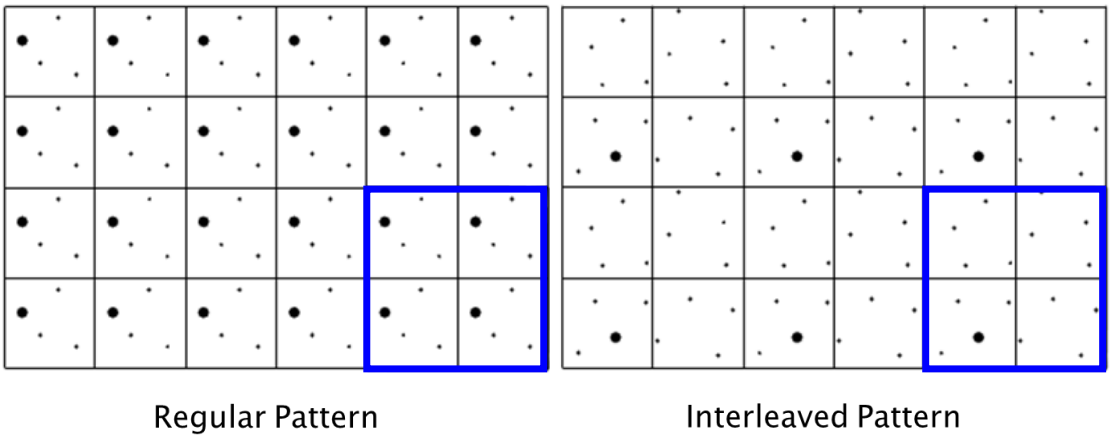

Figure 3.6: showing the difference between the regular sampling pattern and the interleaved sampling pattern [61].

approximation of the actual occlusion in all directions.

Randomization Although a feature-preserving smoothing usually helps in softening SSAO artifacts [10, 114], the decision to rely on a global frame leads to visible artifacts even after filtering over several adjacent samples (see Figure 3.5). We solve this problem by replacing the global $\{x, y\}$ frame by a local, randomized one: local axes are built by choosing a random direction for each pixel.

In order to favor the subsequent filtering process, we use interleaved sampling [61] (see Figure 3.6) to create noise patterns. The size of the noise pattern is chosen to match the filter support size to ensure that artifacts are mostly removed (see Figure 3.7) because it implicitly correlates both processes. Relying on two 1D evaluations only, turns the original $O(k^2)$ SSAO complexity into $O(k)$. The big advantage of our separable approach is that any SSAO occlusion estimator can be used, combined with any occlusion point sampling, as long as it can be restricted to a single direction in screen space. It is fully compatible with the screen space pipeline in any modern rendering engines.

Noise Pattern Different noise patterns have been used to build the interleaved sampling [61] pattern in our SSAO application [111]. The different distribution patterns (see Figure 3.8) can be classified into several groups:

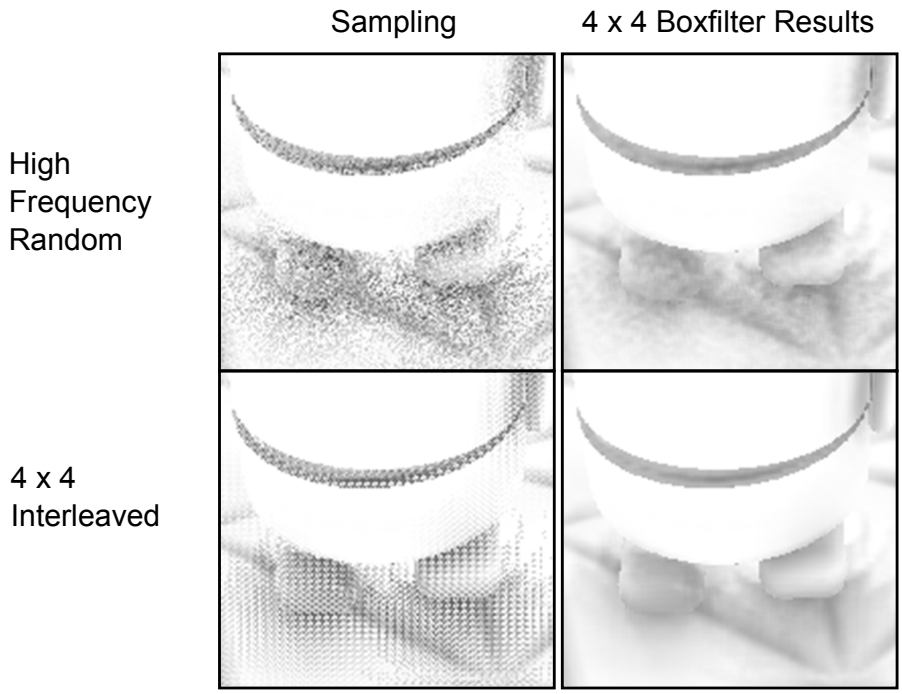

Figure 3.7: The image shows the advantage of interleaved sampling for SSAO in the "Toasters" scene, using the same size for boxfilter kernel can easily remove the noise pattern which was created by interleaved sampling. With the totally random sampling, it is difficult to control.

– Random
– Stratified sampling: Regular and Jittered
– Low-discrepancy sampling: Halton sequence and Hammersley sequence

The distribution pattern is a key component for ensuring the quality of the final AO buffer. We use the Hammersley sequence in the end which seems to be the best choice in practice.

3.5 Implementation and Results

We have implemented our algorithm in OpenGL/GLSL on an NVIDIA GTX480 graphics card. To demonstrate the independence of our approach to other kinds of optimization, we used three different SSAO techniques. For each technique, the original result is compared to its separable version. Performances are reported in Table 3.1 for a 1024x768 screen resolution on scenes of 50k triangles. In each

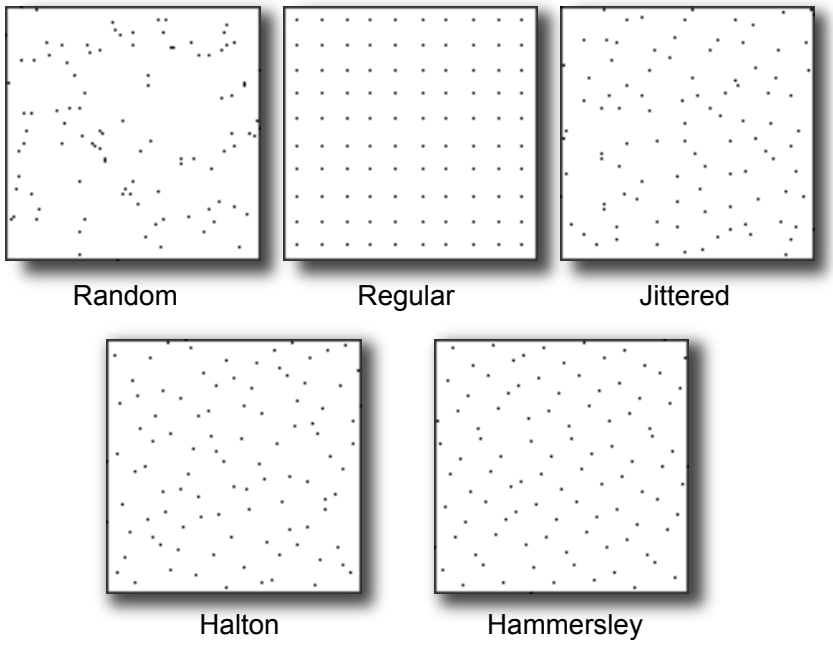

Figure 3.8: Five distribution patterns have been used in our application from left to right: Random, Regular, Jittered, Halton, Hammersley.

case, we obtained a similar visual quality using either an exact evaluation or our separable approximation. The timings clearly show the significant acceleration for various SSAO methods, most particularly when using a large AO support radius. We also analyzed the error introduced by our method and measured perceptual differences in the Lab color space [147] (see Figure 3.10, third row) between the reference AO buffers (see Figure 3.10, first row) and our approximated ones (see Figure 3.10, second row). We can see that they remain low in practice and are even hidden when other illumination effects are added to the final rendering (see Figure 3.10, last row). Choosing only a single direction instead of two axes of a local frame proved insufficient. The corresponding artifacts did not justify the speedup (see Figure 3.9).

3.6 Discussion

We have introduced a new approximation method for screen space ambient occlusion by decomposing the AO evaluation into a separable transport and integrat-

Size	Samples / Separable	Crytek [92]	Vol. Obs. [79]	HBAO [10]
5	5 × 5 / no	3.2 ms	3.5 ms	3.6 ms
	5 × 2 / yes	3.4 ms	3.6 ms	3.5 ms
11	11 × 11 / no	13.9 ms	14.8 ms	15.0 ms
	11 × 2 / yes	5.8 ms	5.9 ms	6.0 ms
21	21 × 21 / no	49.7 ms	51.9 ms	51.9 ms
	21 × 2 / yes	9.9 ms	9.9 ms	10.2 ms

Table 3.1: Performance comparison between exact and separable evaluation for various methods and filter sizes.

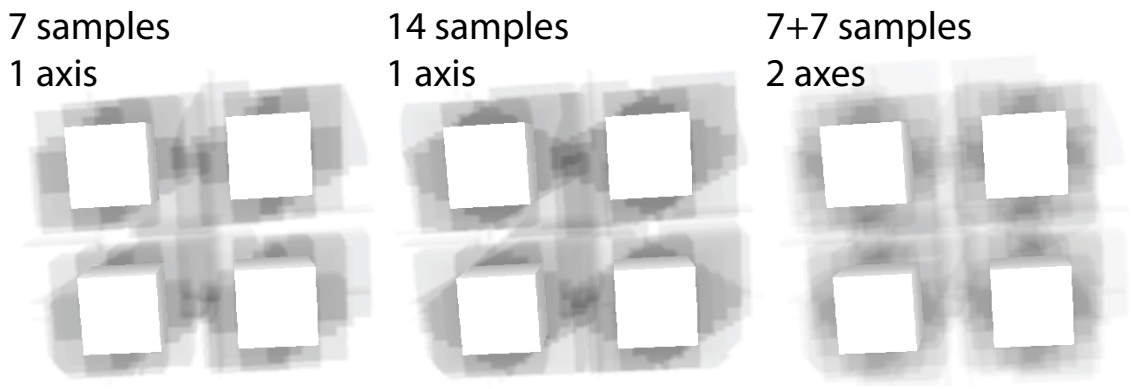

Figure 3.9: One axis does not provide enough information (left), even when using the same number of samples which leads to the same cost (middle), the result is inferior to a two-axis separation (right).

ing AO values by using the angular approximation. Our approximation shows good quality and reduces significantly the AO computation time. It can be easily integrated into existing AO methods as demonstrated by the use of three different techniques. Furthermore, it inherits all SSAO properties such as the natural view-dependent adaptivity and its ability to deal with arbitrary dynamic scenes. We believe that similar separable approximations could be useful for other screen space or general rendering techniques and we plan to investigate this possibility in the future. In the following chapters, we study complementary screen space effects.

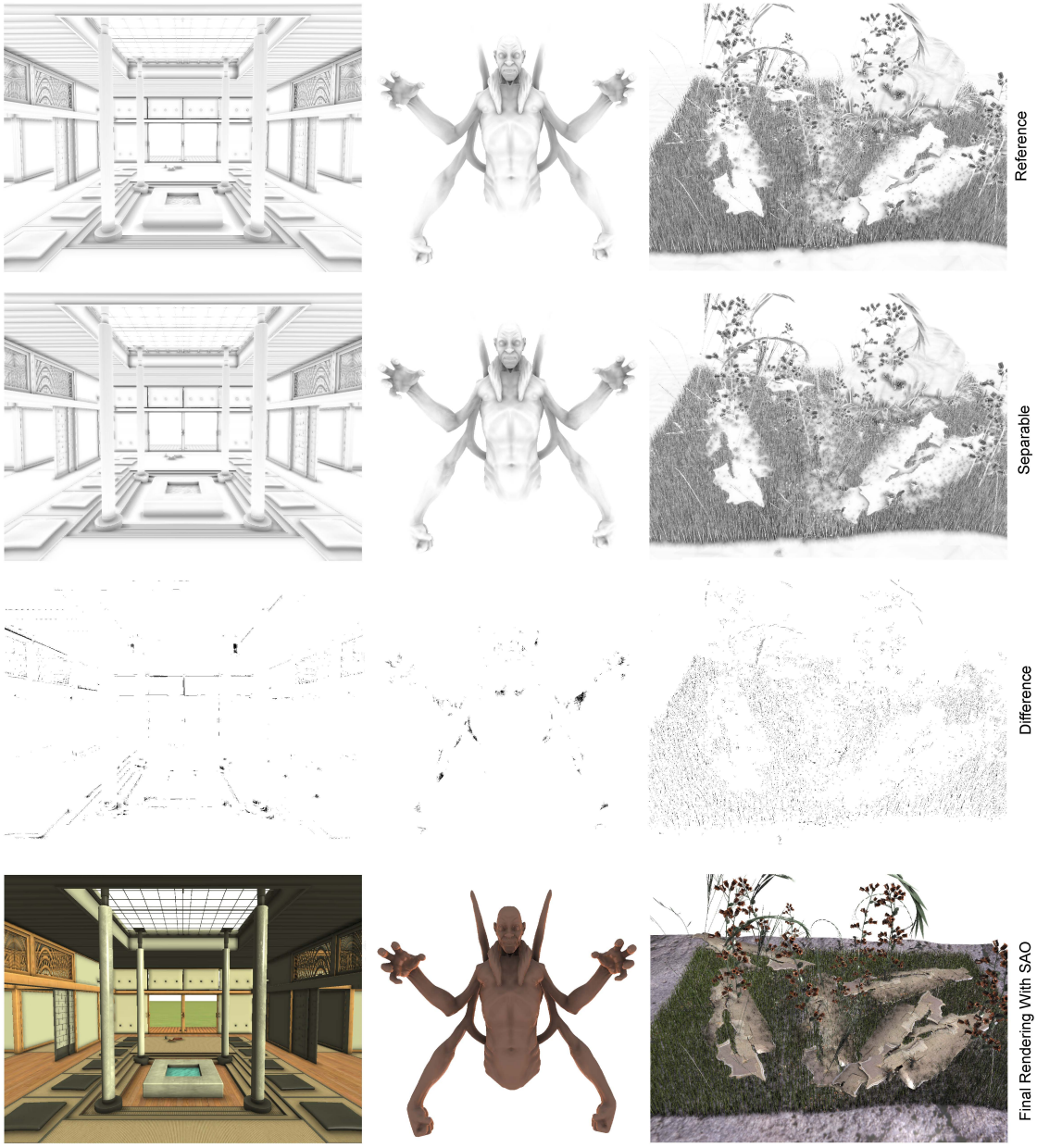

Figure 3.10: Comparison between reference (first row) and separable AO evaluation (second row): for the perceptual difference images (third row), white values stand for no difference while black values indicate visible shifts.

Chapter 4

Screen Space Animated Wrinkles and Identification of Single Facial Actions

While addressing appearance in the previous chapter, we now show that at the boundary between rendering and animation, screen space method can help improving facial animation quality. Our facial animation system is built upon the **Facial Action Coding System (FACS)**. We look for some factors that may influence the Action Units (AUs) and change the perception of animation. As a part of facial details, expressive facial wrinkles can be an important impact factor that increases the realistic impressions of facial expression simulations.

Facial expression is a rather complex system. It results from muscle contraction. Wrinkles often appear on the face and tend to occur as the results of some facial expressions. They are definitely part of the expression. FACS describes, for each AU, not only the changes due to muscle contraction but also wrinkle appearance. As facial actions can be extremely subtle, we aim to measure how wrinkles can enhance the recognition of facial action at a perceptual level in this work.

4.1 Introduction

For the wrinkle simulation of our virtual character, I worked with my colleagues Radoslaw Niewiadomski and Sylwia Hyniewska. In this work, we study factors that may influence the identification of single facial actions (e.g., frown or cheek raising), namely their intensity, dynamics and the application of wrinkles. Results of our

evaluation study show that single facial actions are better identified when they are dynamic and with higher intensity. On the other hand, intense expressions of single facial actions are perceived less natural and less realistic. From our evaluation study, we found that with the help of wrinkles, users can easily perceive facial movements.

4.2 Previous Works

Several studies (e.g. [30] [60] [104] [9]) have been done on the perception of synthesized facial expressions of virtual agents (VAs). They mainly focus on the stereotypical full-blown expressions of emotions. However recent research, e.g. [62] shows that expressions of emotion are rather a sequence of facial actions than a full-blown single-shot display.

The latter ones occur rarely in the real-life interactions. Single facial actions are ambiguous as they can be the components of different facial expressions. For example the action lips-corners-up (i.e. AU 12) occurs in the expression of joy as well as of embarrassment [62]. They are also more subtle (consequently more difficult to perceive). While the perception of a full-blown stereotypical expression can be done from a subset of its visible features (e.g. frown in anger, see [100]) it is not the case for single facial actions which are unitary features. Thus, to understand the perception of synthetic emotional expressions and to generate more realistic ones, it is more appropriate to focus on single facial actions rather than on stereotypical expressions.

For our perceptive test we have chosen some frequently appearing facial actions corresponding to certain Action Units (AUs [106]). We have tested 3 factors that may influence their perception: dynamic displays (versus static images), their (muscular) intensity, and the role of wrinkles. We have measured their identification rate, their realism and their naturalness.

The previous studies focus on the identification of synthetic expressions of basic emotions. Kätsyri and Sams [60] showed that synthetic dynamic expressions were identified better than static ones only for expressions whose static displays were not distinctive. In a similar study of Noel et al. [104] the effect of the dynamics was, however, not observed. In Bartneck and Reichenbach's work [9] the higher intensity expressions were recognized better. Finally, in Courgeon et al. [30] the application

AU Number	FACS Name	Wrinkle Number
1	Inner Brow Raiser	0
2	Outer Brow Raiser	1, 2
4	Brow Lowerer	3
6	Cheek Raiser	4, 5
10	Upper Lip Raiser	6, 7
12	Lip Corner Puller	8, 9
20	Lip stretcher	10, 11

Table 4.1: Several main AUs are used in this work with manually defined corresponding wrinkles. For some AUs, the wrinkles need to separate into left and right sides.

of wrinkles increases the agent's expressivity but does not improve the recognition. All these studies used the stereotypical expressions of the basic emotions. However, these stereotypical expressions can be identified easily even without wrinkles or only from images. This may not be the case for more "unitary" actions. These actions may be more easily confounded with each other. This drove our decision to study single facial actions rather than full-blown expressions.

4.3 Our Implementation with Wrinkles Model

4.3.1 Action Units for Facial Animation

The **Facial Action Coding System (FACS)** [106] by Paul Ekman, Wallace V. Friesen is a method for measuring human facial behaviour. Movements of each individual group of facial muscle is encoded by FACS from slightly different instant changes in the facial appearance. It is standardized from the physical expression of emotions views. It is useful for psychologists, animation artists and programmers. There are 30 Action Units (AUs) that have been defined for the facial actions of individual muscles or groups of muscles in the 2002 version. Main Action Units of facial movements that we used in our work are reported in Table 4.1.

Since the facial animation framework with AUs has already been integrated into our «**Greta**» system, we only need to connect the AUs with the wrinkles implementations.

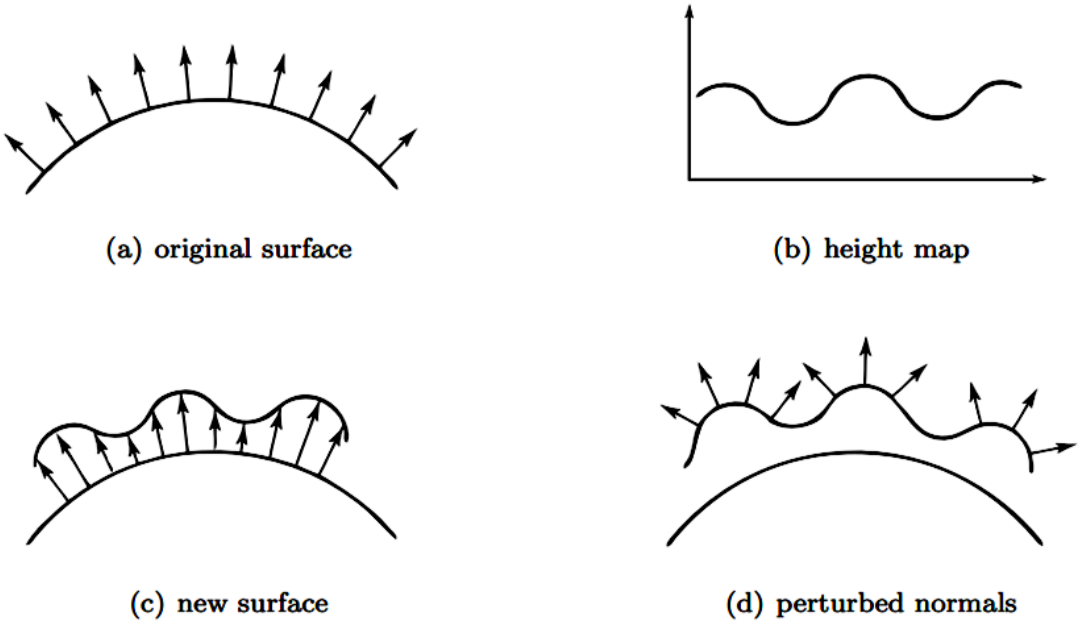

Figure 4.1: Blinn [14] describes their bump mapping method that bends the normal vector by estimating new the surface.

4.3.2 Wrinkles Implementation

There are two main approaches to generate wrinkles: texture [30] and geometry based methods [74]. Due to the computational cost, we have chosen to apply the texture based technique in our system. This method, such as the bump mapping technique, was first introduced by Blinn [14]. The author proposed to modify the surface normal vector before the lighting computation, thus the perturbed normal can give the visually satisfying result without changing the surface geometry (see Figure 4.1).

The computation of Blinn's method can be formulated as:

$$\tau = \sigma + \beta \cdot \vec{n} \tag{4.3.1}$$

where τ is the new surface, σ is the original surface, β is the height field value, $\vec{n}$ is the original normal vector.

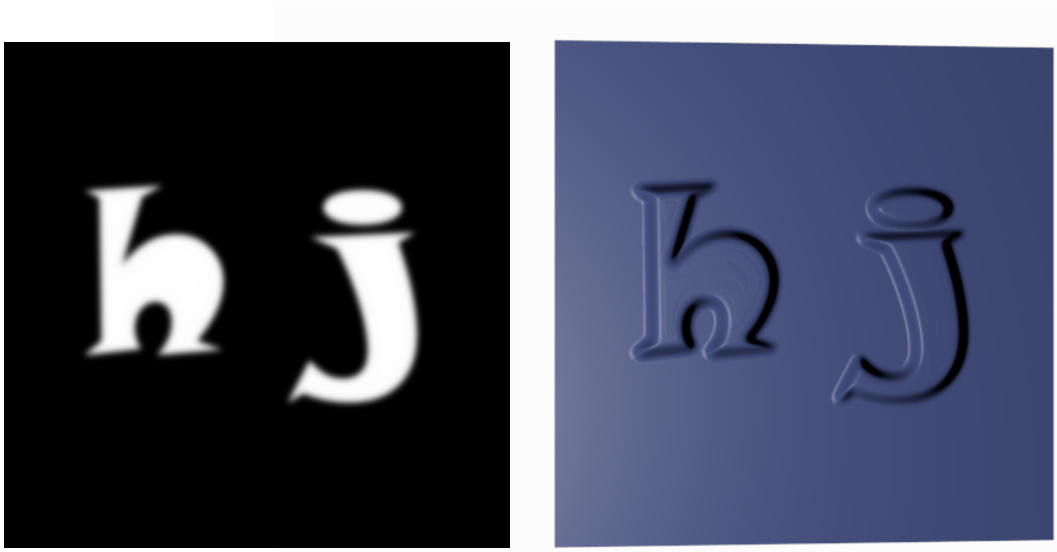

Figure 4.2: Bump mapping applied to a blue plane. Visually satisfying results can be achieved without geometric displacement.

$$\tau_u = \sigma_u + \beta_u \cdot \vec{n} + \beta \cdot \vec{n}_u \tag{4.3.2}$$

$$\tau_v = \sigma_v + \beta_v \cdot \vec{n} + \beta \cdot \vec{n}_v \tag{4.3.3}$$

where τ_u and τ_v are the derivatives in the u,v directions, which are also the curvatures of the current surface in the u,v directions.

The computation can be approximated if β is small, the last term of both equations 4.3.2 and 4.3.3 can be ignored.

$$\tau_u \approx \sigma_u + \beta_u \cdot \vec{n} \tag{4.3.4}$$

$$\tau_v \approx \sigma_v + \beta_v \cdot \vec{n} \tag{4.3.5}$$

$$\vec{n}' = \tau_u \times \tau_v \approx (\sigma_u + \beta_u \cdot \vec{n}) \times (\sigma_v + \beta_v \cdot \vec{n}) \tag{4.3.6}$$

where $\vec{n}'$ is the perturbed normal. The value of the perturbed normal vector equals the cross product of curvatures in u,v directions on the surface 4.3.6.

The result of using the bump mapping technique can be seen in Figure 4.2.

For the purpose of the study (the identification of single facial actions) with

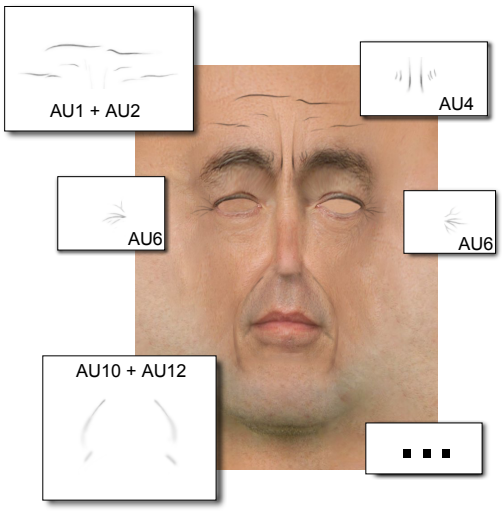

Figure 4.3: Normal Maps/Bump maps for wrinkles corresponding to different action units.

wrinkles, we generate wrinkles using Screen space bump wrinkle approach. It extends Blinn's idea [14] and is based on a GPU bump mapping approach [90]. We simulate the wrinkle effect by performing the computation of the perturbed normal vector only in screen space with fragment Shader in OpenGL. So the complexity depends on the number of pixels only, not on the number of vertices of the facial model. The perturbed normal vector depends only on the input surface normal and on the height value in screen space. We do not need to apply a transformation from tangent space to world space. The perturbed normal vector is computed by composing the changes of local surface gradient of x and y directions in screen space for each pixel. We define 12 groups of wrinkles in textures, which correspond to the AUs that we seek to evaluate in this work (see Figure 4.3 and Table 4.1). These AUs are often linked to the display of the 6 basic emotions and will be discussed in the next section. At runtime, when an action unit is about to be activated on the face mesh, the GPU receives its intensity value and computes the corresponding wrinkles. The final result is the composition of all active wrinkles (see Figure 4.4).

Figure 4.4: Results of our agent: **Left**: neutral pos, **Middle**: AU 1 + 2, **Right**: AU 11 + 12.

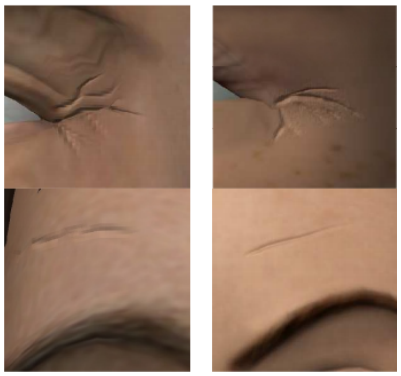

Figure 4.5: Wrinkles of two agents (a male on the left, and a female on the right) corresponding to AU6 (cheek raise) and AU1+2 (raised eyebrows).

4.4 Evaluation

The perceptive evaluations of affective interaction using wrinkles have been done by Courgeon et al. [30]. They proved that realistic wrinkles increase agents' expressivity and the user's preference, but they didn't show the impact of such a factor on the recognition of emotional categories.

In our work, we investigate factors such as intensity and dynamics, with the help of wrinkles simulation, and how they influence the identification of single facial actions. We proposed three hypotheses to test:

- H1. High intensity facial actions are better identified but they are less natural and less realistic,
- H2. Facial actions displayed by an animation are better identified than the ones presented by static images,

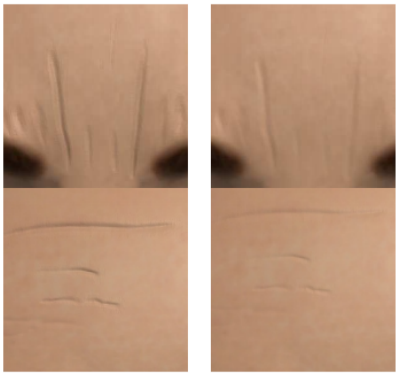

Figure 4.6: Wrinkles resulting of eyebrow movements with high and low intensity (AU4 and AU 1+2).

– H3. Facial actions with wrinkles are better identified, or, at least, they decrease confusions.

The following 6 facial actions were investigated first: 3 for the upper face and 3 for the lower face. They are: AU4 (see as frown movement), AU1+4 (oblique eyebrows), AU1+2 (raised eyebrows), AU6 (cheeks raise), AU20 (lip straightly extended) and AU12 (lip corners pulled up). These actions are particularly important in communicating emotions by VAs. Some may occur in several expressions (e.g. AU4: frown), others have a particular role (e.g. AU6: cheeks raise in the spontaneous expression of joy). The remaining actions were intentionally chosen to create confusions, e.g. oblique and raised eyebrows. Two different agents were used: one female and the other male. All 6 facial actions were displayed with and without wrinkles and with low and high intensity (see Figures 4.5 and 4.6). Finally, they were shown by: image in the expression at its maximum (static condition) and in animation (dynamic condition). This gives 96 stimuli (2 agents × 6 actions × 2 intensities × 2 intensities of wrinkle × 2 showing modes: static or animation). In the experiment we used 128 different stimuli. The remaining 32 stimuli presented other actions for confusion purpose (such as wide opening of eyes (AU5) or no action at all) and were not considered in the results. All the animations were validated by a certificated FACS expert.

The test was placed on the web and organized as a set of web pages. Each web page displayed one stimulus at a time. Participants could see the animations only once and they had to answer all the questions before proceeding to the next

	Upper	Lower
Animations	8	8
Images	8	8

Table 4.2: The number stimuli in each group for each participant

	high intensity			**low intensity**		
AU	**EXP**	**NAT**	**RE**	**EXP**	**NAT**	**RE**
1+2	**4.23**	3.40	3.32	**3.18**	3.46	3.33
4	**3.84**	3.75	3.69	**3.31**	3.90	3.82
1+4	**3.03**	3.28	3.23	**2.64**	3.33	3.35
6	**3.14**	**3.03**	**3.03**	**2.46**	**3.33**	**3.34**
20	**4.33**	**2.30**	**2.34**	**4.02**	**3.21**	**3.16**
12	**4.21**	**2.67**	**2.72**	**3.77**	**3.29**	**3.36**

Table 4.3: The effect of intensity (significant values in bold). EXP: expression; NAT: naturalness; RE: realism.

animation. Each participant could evaluate a maximum of 32 stimuli (totally 16 upper, 16 lower-face with 16 animations and 16 images) (see Table 4.2). They were told that they could stop the experiment whenever they wanted. This explains the different number of answers for each stimulus. In the experiment we asked participants to identify stimuli using the 5 point Likert scale from "totally disagree" to "totally agree". For the upper facial actions there are 3 separate scales labelled "raised eyebrow in half circle" (AU1+2), "frowning" (AU4) and "raised eyebrow in oblique" (AU1+4). For the lower face expressions there are 3 other scales: "cheek raise" (AU6), "lip extension - corner up" (AU12), "lip extension - straight" (AU20). In both cases participants were asked to express their opinion on the naturalness and realism using two separate 5 point Likert scales. For the EXP column (see Table 4.3), we only use the answer from the corresponding expression 5 point Likert scales.

4.5 Results

We collected 2092 answers in total from 92 participants (age 19-51). The Likert scale usually does not satisfy the condition of the normal distribution. Therefore, we applied an additional non parametric Mann-Whistley test to the data. The global

	animation			static image		
AU	**EXP**	**NAT**	**RE**	**EXP**	**NAT**	**RE**
1+2	3.82	**3.68**	**3.53**	3.55	**3.11**	**3.07**
4	**3.86**	3.79	3.74	**3.21**	3.87	3.77
1+4	**3.00**	**3.48**	3.41	**2.63**	**3.10**	3.15
6	**3.46**	3.24	3.24	**1.89**	3.10	3.10
20	4.20	**3.07**	**3.06**	4.15	**2.34**	**2.35**
12	**4.22**	**3.14**	**3.18**	**3.70**	**2.80**	**2.87**

Table 4.4: The effect of presentation (significant values in bold).

Wrinkles:	**4**	**1+4**	**1+2**	**20**	**12**	**6**
with	0.73	0.29	0.57	0.56	0.74	0.47
without	0.5	0.47	0.68	0.52	0.77	0.45

Table 4.5: "First choice" percentage.

Mann-Whistley test shows the effect of intensity and dynamics on naturalness and realism. At the same time the effect of wrinkles was not observed. Next, in order to compare the identification rates we considered each facial action separately. Five out of six actions (see Table 4.3) were better identified on the more intense expressions than on the low intense ones (M-W test, $p<0.05$). Similar tendency was observed for AU1+4 (M-W test, $p=0.065$). At the same time 3 intense actions (AU12, 20 and 6) were perceived less realistic while 2 of them were less natural (M-W test, $p<0.05$). Similar tendency was observed for the third action, AU6, (M-W test, $p=0.065$). Three out of six actions (AU4, AU6 and AU12) were better identified from the animations than from the images (M-W test, $p<0.05$). Similar tendency was observed for AU1+4 (M-W test, $p=0.088$). 4 actions were perceived more natural and 2 more realistic when presented as animations (see Table 4.4).

Finally, only one action, AU4, (M-W, test $p<0.05$) was significantly better identified with wrinkles. To check if the confusion rate was influenced by the wrinkles or not, we evaluated the number of "first choices" (see Table 4.5): For each facial action we calculated the percentage of answers in which the corresponding label received a higher score than the remaining facial labels.

For concluding our evaluation, six facial actions were considered in our evaluation. In detail, **AU 4**: The frown is the only action that is better identified with wrinkles, from the videos and with higher intensity. **Oblique (AU1+2) and raised**

eyebrows (AU1+4): We chose these two actions as they can be easily confounded. They use the same unitary action, namely **AU1 (inner raise eyebrow)**. Indeed, both received the smallest values in the "first choice" test. It seems that the wrinkles did not help in distinguishing between these two actions. **AU6** was recognized much better in the intense and dynamic conditions. Indeed, this action is particularly difficult to be perceived from static images. Nevertheless, similarly to other lower face actions the intense AU6 was perceived less natural. **AU12 and AU20**: The last two actions can be confused (especially with low intensity). Indeed, they were better recognized in the intense and dynamic conditions rather than in the image condition. On the other hand, wrinkles did not improve the identification of these two actions.

4.6 Discussion

This part of the thesis studied the role of the intensity, the dynamics and the wrinkles in the perception of the synthesized facial displays. In this experiment, we tested 3 hypotheses concerning these three factors. The important contribution of this research is that instead of using stereotypical expressions (that rarely occur), we analyze single facial actions, i.e. unitary components of more "real-life"-like expressions. Secondly, more intense expressions are perceived less natural and less realistic. Last but not least the role of wrinkles is ambivalent. The effect of wrinkles was limited (Hypothesis 3). In the case of ambiguous actions, wrinkles may even increase the confusion. The wrinkle simulation is good for visualization, but cannot really help to identify the expression. Our work also confirms that the dynamics of an expression is the most important cue allowing identification.

Chapter 5

Adaptive Screen Space Subsurface Scattering

We complete our rendering module with several important visual effects: subsurface scattering (SSS), depth of field (DOF) and shadow mapping.

5.1 Screen Space Subsurface Scattering

When the light penetrates the surface of translucent objects, refraction and reflection effects occur under the surface. Light is scattered when it passes in the media of different material. It will exit the object at different positions on the surface. Implementing such a subsurface scattering (SSS) mechanism is important in 3D computer graphics for the simulation of translucent materials such as marble, milk and skin. In particular, this phenomenon is critical when rendering human skin which has several different composition layers, and therefore instrumental for ECA rendering.

The SubSurface reflection equation is given as

$$L^e(x_o,\omega_o) = \int_S \int_{\Omega+(x_i)} L^i(x_i,\omega_i) S(x_i,\omega_i;x_o,\omega_o) \cdot (\omega_i \cdot N_i) d\omega_i dx_i, \tag{5.1.1}$$

where $L^e(x_o,\omega_o)$ and $L^i(x_i,\omega_i)$ are the exiting and incident radiance. S is the object's surface, $\Omega+(x_i)$ is the hemisphere for the position x_i in the normal direction N_i, and $S(x_i,\omega_i;x_o,\omega_o)$ is the BSSRDF function.

BSSRDF is an eight-dimensional function representing the energy of incoming light with direction ω_i at position x_i and the outgoing energy with direction ω_o at position x_o on the surface. The complexity of the BSSRDF depends on the geometry of the surface and the characteristics of the material. According to the complexity, SSS simulations can be divided into two groups: single scattering and multiple scattering.

Hanrahan and Krueger [46] developed a model for single subsurface scattering in layered surfaces in terms of one-dimensional linear transport theory. They use a general Monte Carlo method combined with a ray tracer to describe the light transmission. This model is particularly appropriate for common layered materials appearing in nature, and is based on the bidirectional reflectance distribution function (BRDF).

Donner and Jensen [36] presented a multi-dipole diffusion approximation method for simulating multiple scattering in thin and multi-layered translucent materials, they use the Kubelka-Munk theory in frequency space and combine results of multiple layers of translucent materials. This method gives realistic results close to the physiological appearance, but requires lots of predefined parameters. Also, for each dipole diffusion, the parameters cannot be used for the multi-dipole model directly which makes this method too complex and difficult to use. Besides that, the computation is expensive, thus, it is not suitable for real-time rendering.

In the real-time rendering domain (see Figure 5.1), d'Eon et al. [33] have used a set of Gaussian filters to approximate the subsurface scattering term. They discovered that the reflectance and diffusion transmission profiles which were defined by the multi-pole model [36], can be approximated by a weighted sum of Gaussians in texture space, and the deviation error term should be minimized between these two models. Let $R(r)$ be a radial diffusion profile, then the error term can be defined as:

$$error = \int_0^\infty r(R(r) - \sum_{i=1}^{k} w_i G(v_i, r))^2 dr \tag{5.1.2}$$

Where both weights w_i and the variances v_i for k Gaussians are allowed to vary, and the Gaussian of the variance v is:

$$G(v, r) := \frac{1}{2\pi v} e^{-r^2/2v} \tag{5.1.3}$$

Compared to the multi-pole model [36], using the approximation of Gaussians,

Figure 5.1: **Above**: d'Eon et al. [33] texture space gaussian convolution method, **Below**: Jimenez and Gutierrez [58] screen space subsurface scattering results.

the long series of convolutions computation in the frequency domain can be approximated to a good acceptable version for real-time rendering.

We follow the idea of Jimenez and Gutierrez [58] who approximated SSS in screen space. We take into account all the previous algorithms of convolving a diffusion profile, we choose to generate several blurred maps using the Wendland function [139] (see Eq. 5.1.4).

$$W(t,h) = \begin{cases} (1-\frac{t}{h})^4(\frac{4t}{h}+1), & if\ 0<=t<=h \\ 0, & if\ t>h \end{cases} \tag{5.1.4}$$

where t is the current distance, h is the maximum or threshold distance.

Our convolution kernel is a Wendland based trilateral convolution filter (see Eq. 5.1.5) with the depth buffer as the second parameter and also add and the normal buffer to consider surface details. We take the advantage of multi-render target (MRT) to store diffuse, specular, ambient, normal vector, position multiple

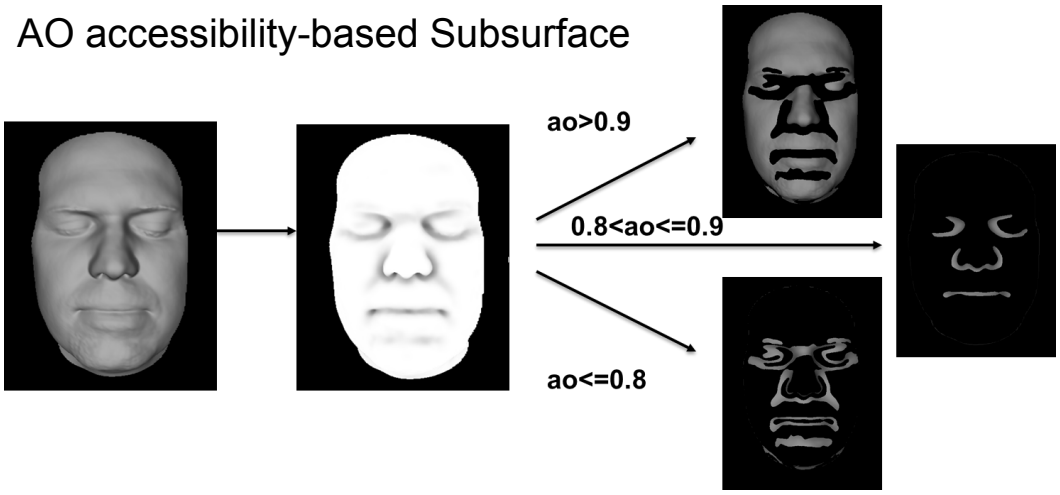

Figure 5.2: AO decides the levels of detail of SSS computation.

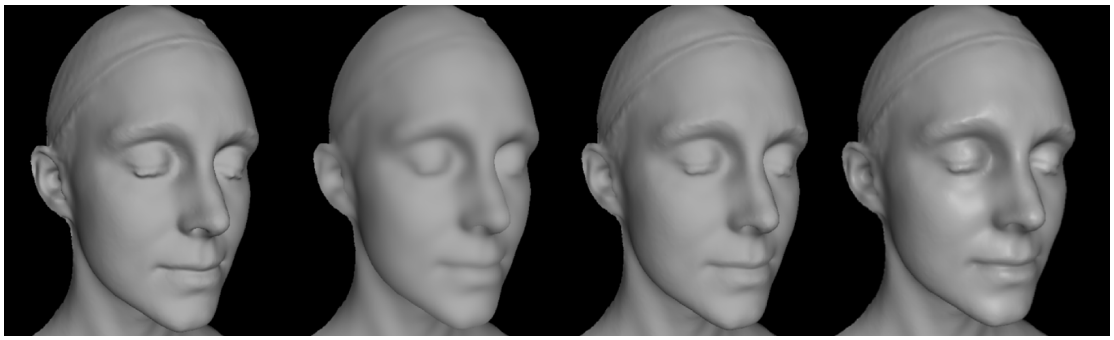

Figure 5.3: **Left**: diffuse component only, **Middle**: single resolution scattering result, **Right**: our multi-dimentional adaptive scattering result.

render targets and ambient occlusion components in the different frame buffers. We apply our convolutions on the diffuse profile in the SSS pass.

$$TCF(p) = \sum W(Max_{x,y}, distance_{(x,y)}(p,q)) \times W(Max_z, distance_z(p,q)) \times (max(dot(n_p, n_q), 0)) \times diffuseColor \quad (5.1.5)$$

where $Max_{x,y}$ and Max_z are the parameter that define the scattering region, p is the current pixel, q is the neighbor pixel, n_p and n_q are the normal vectors of p, q.

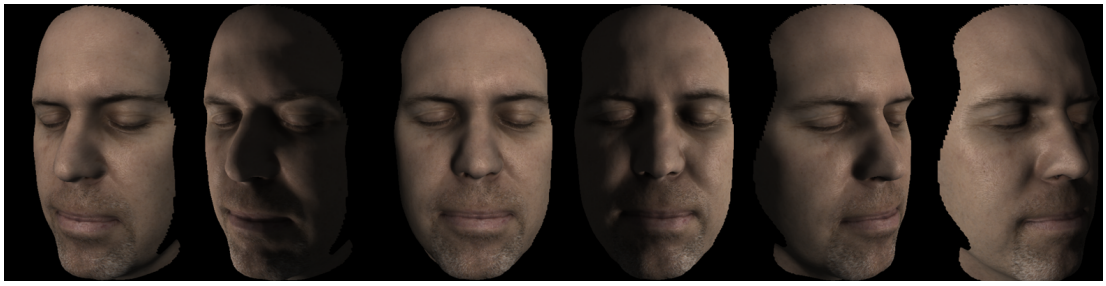

Figure 5.4: The final results with our implementation. Model is from [82].

5.2 Accessibility-based Screen Space Subsurface Scattering

For achieving a high speed in our algorithm, we use low resolution AO, and we use the id component as the matte mask, that we apply in our fragment shader only on the matte profile. We also discovered that highly curved surface will produce more unavoidable subsurface scattering effect. On the other hand, flat surface will have less difference in the diffuse profile after the convolution. We used the AO component as the controller of the accessibility of our convolution kernel, only the part in the shadow will evaluate full series of convolutions, and for the other parts, we apply a single sample filter or avoid the convolution at all. The process can be seen in Figure 5.2. Further results are depictured in Figure 5.3 and Figure 5.4.

5.3 Discussion

Accessibility-based screen space subsurface scattering combines AO and subsurface scattering using a screen space technique to guarantee real-time rendering. We use AO for self shadowing but also to control the quantity and quality of our subsurface scattering. Our AO based accessibility technique is designed to achieve high quality results close to the complete screen space subsurface scattering, but with less calculations. In GPU programming, dynamic branching is costly in terms of performance. This led us to find out the balance between the cost of using dynamic branching and the cost of our pixel computation. In this part of this thesis, we explored more screen space rendering techniques. They are not physically correct, but provide a good approximation (for virtual environments) in terms of visual

quality. In particular, they are efficient and suitable for real time applications. In the following chapters, we describe the second sub-topic of this thesis dealing with "character animation".

Part II

Animation

Chapter 6

Animation Background

Our objective is to make the virtual world much more life-like. With the purpose of creating a realistic scene, the second step is to animate the virtual scene. How to make the 3D objects move in the scene, especially to control the virtual character becomes our research topic.

In general, "Animation" refers to a recorded sequence of images which can illustrate figures in motion going through time. An animation video is often generated by a camera-like recording tool which can then be displayed by a projector with a viewing screen, or other viewing devices. The image sequence needs to be shown in a fast succession relative to the standard frame rates, usually 24, 25 or 30 frames per second. All these images can be made by either hand drawing or computer generated imagery (CGI), using 3D object computation, or even combining different techniques.

Since the computer technologies were well developed in the 20th century, animation was revolutionized by CGI. Pixar produced their first feature film "Toy Story" which is completely computer generated in 1995. And then different animation studios developed their own ways of generating realistic creatures. Now, CGI techniques are widely used both in cartoons and in films. But for having a good visual quality in films, the production cycle is long both in the computation process and in the design process by a large group of artists.

In other fields, such as computer games, modern CGI technologies are often required. The gaming perception (such as facial movements, body expressions and emotions) needs to become more realistic, not only in rendering, but also in simulations. How to generate a good fluent motion with realistic creatures becomes

the main challenge. Also in the virtual reality domain, such as the communication between virtual people, or virtual people and humans, they all have a large variety of personalities. Thus, both the artificial intelligence decisions and the visualization computations need the efficiency. For real-time applications, all these processes should be running online.

The main task of our animation project is to mimic the virtual character's movements and generate animation sequences for our Embodied Conversational Agent "Greta". We need to control our virtual characters dynamically with low degrees of freedom, and convert high level symbolic data into low level motion structured data. The converted data can be used by a renderer to display the final output images.

6.1 Animation System for 3D Models

In graphical animation system, virtual creatures have their own motions. How to manipulate the 3D models and simulate their habitual actions is the main problem of animation system. Typically, for a human figure, it needs to mimic motions like a human being by using a high level definition. But in the low level, the real-time system must be able to deform 3D meshes in an accurate and efficient way. Several methods (see Figure 6.1) exist that can achieve such a process:

1. The morphing method (see Figure 6.1 on the left) can be directly applied on the vertices of a mesh for each frame, the deformation depends on the positions of all vertices. This method can give a good visual quality of the animation, but it needs quite a lot of work by artists, and all the frames need to be pre-defined. It can not generate a full dynamical animation in real-time, and it requires a large memory bandwidth.

2. The skeleton method (see Figure 6.1 in the middle) manipulates the skeleton of a mesh, and the surface deformation is applied by using weighted skinning methods with skeleton information. Skeleton method performs real-time deformations on the skeleton, the skinning technique transforms the final skeleton deformation to the mesh surface. Comparing with Morphing method, it needs less memory as it only saves skeleton information for each frame. But the bones' weights need to be predefined by artists. And this method works only for rigid body deformations. It is difficult to generate a realistic animation for soft objects such as clothes.

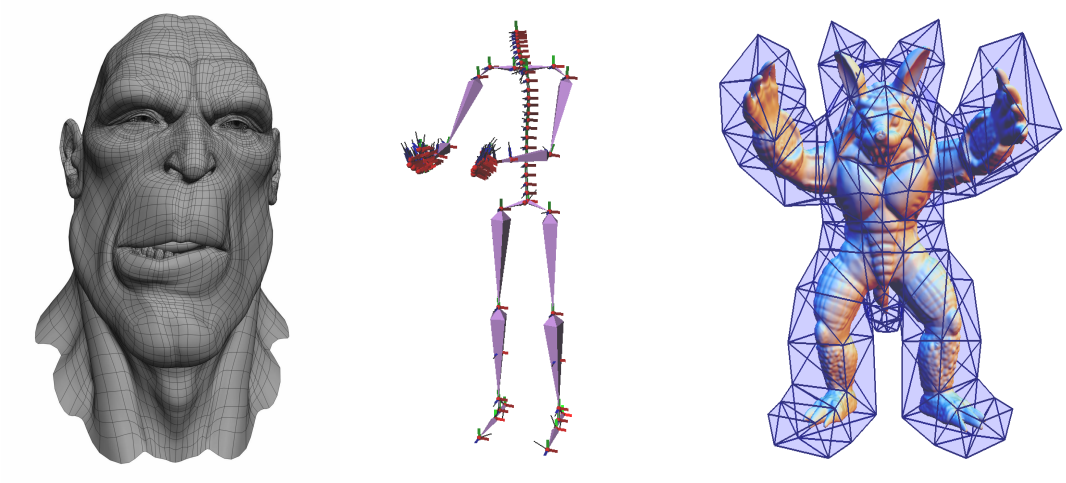

Figure 6.1: Three different types of animation methods: vertex based method on the **Left** (from FACE ROBOT), skeleton based method in the **Middle** (generated by «Etoile») and cage based method on the **Right** (from Jean-Marc Thiery [130]).

3. The cage deformation (see Figure 6.1 on the right) constructs a cage around the current mesh parts, then the mesh deformation is built upon the deformation of the cage. The cage deformation is less efficient than the skeleton method, but it offers more flexibilities and a better quality on soft objects by using more control points on the cage.

Since a bone is the fundamental functional unit which supports the vertebrate body, it is suitable to simulate virtual character by using skeletal structure.

6.2 Animation Field

6.2.1 Procedural Methods

The procedural methods are designed to use geometrical solution to compute the animation postures and animation sequences on the fly. Kinematics is one of such methods for manipulating interactively articulated figures and generating key postures. It is widely used in computer animation, robotics, bionics simulation studies, mechanical engineering, etc. Mainly, it can be separated into Forward Kinematics (FK) and Inverse Kinematics (IK). FK uses the pre-defined joint rotations to build the current skeleton posture. IK uses the current target and different geometrical

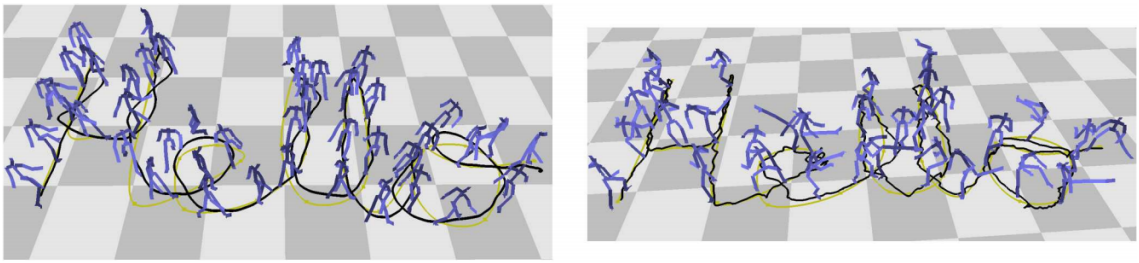

Figure 6.2: The Motion graph results for locomotion are from [67].

constraints to calculate the combination of the possible joint rotations to define the current posture. IK is the general problem in the reaching model. Jacobian methods are proposed to use linear approximation to make the end-effector close to the target. Tang et al. [128] proposed to use Shake model based particles system to solve unconstrained equations, the geometrical constraints can be solved by introducing Lagrange multipliers. More details about the IK solution will be talked in the chapter 7.

6.2.2 Motion Concatenation

Procedural generated animations are somehow lacking naturalness. In particular, human character gesture simulations often result in rather robotic and ragdoll like animations. More lifelike animated characters can make the immersive virtual environment more realistic and more believable. Meanwhile, the high quality and high performance animation synthesis is still a big open challenge. One common solution is to use **Motion Capture** data. A motion graph is an animation **concatenation** system which needs to do fast data searching, re-ordering and re-sequencing existing motion clips to build new animation sequences.

Lee et al. [76] built a framework for representing, searching and producing 3D animation in real-time. They can choose 3 interfaces for generating animations: choice based, sketch based, and a performance-driven vision based interface.

Kovar et al. [67] introduced the motion graph system to model the transitions between different motion clips in Siggpraph 2002 6.2. Later, they extended their graph system by improving their identification of logically similar motions to build transition points from large data sets.

Arikan et al. [5] used Support Vector Machine (SVM) to annotate similar motions in a database. The user annotation could improve similar motion identifica-

tion with varying vocabularies. It can generate smooth motions and perform the specified actions at given times.

Similarity functions are useful for identifying the location of good transition points. Wang et al. [136] evaluated the cost function described by Lee et al. [76]. They also proposed optimal weights set for the cost function using a constrained least squares technique. They showed better results compared to the original version. But their optimal weights are only suitable for the specified motions which cannot be applied to generic animation.

Heck et al. [48] proposed a parametric method that can be applied to the motion graph. Motion clips can be generated accurately by blending together examples in each motion space. They use different sampling methods to refine good transition candidates in continuous parametrized motion spaces. The parametric blending can generate seamless streams of motion transitions in and between motion spaces. This method can be easily used for interactive character control.

In the game industry, people build graph structures manually – so-called «move trees» [93]. They represent transitions that connect several clips. A new appropriate motion clip could be generated frame by frame by using a state machine which relies on the requests from users. Although the building process of move trees is labor intensive, it is well suited for simple but highly interactive animations.

6.2.3 Motion Control

6.2.3.1 Dynamic Control

A proportional-integral-derivative controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems. In animation field, a simplified PD controller has been introduced to generate smooth motions by adjusting the joint's torques on the fly. It only considers the proportional term and the derivative term. The proportional term P produces an output value that is proportional to the current error value. It is the main driving energy for generating motions. The derivative term D represents the rate of change over time. Derivative control can reduce the magnitude of the overshoot and improve the stability of combined controller-process. Parameters in the control systems can be trained by using machine learning.

Shapiro and Faloutsos [121] provide an interface for controlling humanoid char-

Figure 6.3: The stylized results are from [20]. Five motion sequences synthesized from the same choreography, but in different styles (one per row).

acters under physical simulation. Variety of different semantic levels is achieved. Their system provides high level commands and also low level key frame selection. Their interactive controls can drive reactive controls under physical simulation for an intelligent, controllable and interactive character.

SIMBICON system [148] proposed a simple control strategy of biped locomotion for the physics-based simulation. It can generate a large variety of gaits and styles in real-time, including walking in all directions (forwards, backwards, sideways, turning), running, skipping, and hopping. Combining with kinematic solver and dynamic parameters, the system can produce smooth motions with respecting some spatial constraints, e. g. pushes in all directions.

Muico et al. [94] introduced algorithms that construct composite controllers that track multiple trajectories in parallel instead of sequentially switching from one control to the other. Their controllers are built into graph system. Animation can be generated in real time.

6.2.3.2 Stylized Machine Learning

Style Machines [20] achieve the stylistic motion synthesis by learning motion patterns from a highly varied set of motion capture sequences, as a distinct choreography can perform motions with a distinct style illustrated in Figure 6.3. They use stylistic Hidden Markov model (SHMM) to train different stylized models. A new style motion can be generated by interpolating the SHMM sub-space parameters. This approach treats animations as a pure data modeling task. There is no kinematics nor physics modeling. All information are from motion data.

Chapter 7

An Efficient Energy Transfer Inverse Kinematics Solution

To animate virtual characters, sequential character postures need to be generated. We can use motion capture data based approaches or procedural methods. Following the protocol of the existing «Greta» system, we choose to take a procedural fashion, using Kinematics.

7.1 Introduction

Kinematics is a general method for manipulating interactively articulated figures and generating key postures. It is widely used in computer animation, robotics, bionics simulation studies, mechanical engineering, etc. In computer graphics, articulated skeleton models are used to control virtual vertebral living creatures, such as human beings or animals, which frequently appear in films and video games.

One category of Kinematics is Inverse Kinematics (IK) method. Such a process provides the joint rotation solution for individual degrees of freedom via predefined rotation and position constraints. It is often used to animate autonomous agents as neither predefined movements nor key frame precomputations can be done offline.

We propose a low computational cost, physically-based IK solver which combines a mass-spring system with a traditional heuristics factor. The physical manner drives more realistic solutions to the IK problem. Our method [52] is fast, and can deal with multiple constraints in an efficient way without increasing too much

computations. It can be used in a sequential IK manner for creating full body animations of virtual agents. The energy transfer can be easily combined with the expressivity parameters in virtual character systems which allows us to modulate the expressive quality of the movements. In the remainder of the chapter, we first present existing approaches, then we describe our IK solution. We illustrate our method with several examples.

7.2 Previous Works

In robotics and animation, the production of realistic and plausible animation in a fast and efficient way is still a challenging task. The key frame method is a single manner to generate animation sequences. IK methods can be used to create key frame postures. Different reaching models have been proposed by using different IK solutions. They can be clustered into 3 main methods: analytical, numerical and hybrid methods.

7.2.1 Analytical Methods

Analytical methods or closed form solutions, e. g. as proposed by Wu et al. [145] and Gan et al. [39], specify the figure with constraints and solve the IK problem by studying possible relative posture properties. These methods are fast and accurate, but they can be used only for specific constrained models (e. g. humanoid models).

7.2.2 Numerical Methods

Numerical methods achieve satisfactory solutions for more general cases, but require more computational iterations. Zhao and Badler [151] introduced the idea of local minimization of a set of non-linear equations and proposed their Cartesian space constraints solution.

Tang et al. [128] proposed to use the Verlet integration [134] to solve unconstrained equations, then based on a set of Lagrange multipliers, they applied some constraint forces to correct their motions using Shake model in molecular simulations.

Most recent methods [140, 116] [144] [135] [150] [25] are using the Jacobian matrix to get a linear approximation. This family of methods models the end effector's

movement relative to the changes of the whole system which includes translation and rotation transformations between joints.

The Jacobian matrix is a matrix of the first order partial derivatives of vector functions. In an IK system, it keeps the first order linear behaviour of the whole system. The linear equation is given by $\Delta p = J(\theta) \times \Delta\theta$, where Δp is the end effectors' movement, $J(\theta)$ is the Jacobian matrix that represents the end effectors' movements relative to the changes of local rotation, $\Delta\theta$ is the local rotation. We need to compute the rotation angle by inverting the function: $\Delta\theta = J(\theta)^{-1} \times \Delta p$. So the remaining problem is to solve the inverse of the Jacobian matrix, which is usually done by taking the pseudo-inverse of J. Several methods have been proposed to compute or approximate the Jacobian inverse, such as Pseudo-Inverse Jacobian [140, 116], Jacobian Transpose [144], Damped Least Squares (DLS) [135], Singular Value Decomposition Jacobian (SVD-DLS) [150], Selectively Damped Least Squares (DLS) [25].

The second family of numerical IK solvers is based on Newton's method or its approximations, such as those suggested in [35, 29]. The solution is to minimize the error functions. Newton's method is based on a Taylor series expansion to compute the highly complex Hessian matrix. It has a high computational footprint for iterations, but it offers smooth motion results. When compared to the Jacobian inverse methods, it does not have the singularity problem.

The third family is based on heuristics. One of the most popular methods is Cyclic Coordinate Descent (CCD) [137]. The main idea is to align one joint with the end effector and the target at a time, and to bring the end effector closer to the target iteratively. Several extensions have been made, e. g. by Welman [138] and Canutescu [26]. As a heuristic method, thus without any matrix computation, CCD is extremely efficient. But it suffers from unrealistic looking results and is difficult to handle when multiple end effectors are used.

Another heuristic method was presented by Muller [96], called Triangulation Inverse Kinematics. It is based on the cosine rule to compute each joint's rotation by starting at the root joint and traversing up to the end effector. It is guaranteed to find a solution without constraints. This method requires less computation than CCD does. However, it can only be used for single chain problems. It cannot be applied for complex constraint models. The corresponding constraint version does not provide good results due to independent computations for each joint.

Several methods also propose to solve the IK problem in the position space instead of the orientation space. Aristidou et al. [6] propose a method based on forward and backward iterative movement. They start by moving the end effector forward to the target and then adjusting each joint's position backwards one at a time. Instead of using rotational adjustments, they minimize the distance between the joint and the target by finding a joint's new position along a line. Further, they propose to solve the multiple end effectors problem and constraints problem. Our method follows a similar idea where joint rotation converges by adjusting each joint's position using mass-spring model.

7.2.3 Hybrid Methods

Hybrid methods, such as the Morphology-independent representation of motions [70] and Sequential Inverse Kinematics (SIK) [17, 133], combine analytical and numerical approaches to reconstruct 3D human body movement. Here, the IK problem is solved sequentially by using different analytical iterative algorithms for various parts of the body in a specified order.

Alternative methods exist as well, such as Sequential Monte Carlo [31] which uses Importance sampling combined with forward kinematics to solve the IK problem. Such a method avoids the computation of the inverse matrix, using the hidden Markov model [23] to define a possible hidden state, and combines a filtering framework to reformulate the IK chain.

In physics-based computer animation research, many works have addressed the control of articulated figures using robotics-inspired proportional-derivative (PD) controllers which can handle different types of motor tasks, such as walking, running, swimming, boxing [153] [19].

There exist also some other methods using particle system, such as that proposed by Hercker et al. [49] and Zordan et al. [152]. They are designed for some complex models with highly varying skeleton morphologies. Our method is also based on energy interactions, but is designed as a lightweight expressive solver which is easy to implement and can generate expressive postures with good visual qualities.

IK methods are useful for defining gestures without precomputing key frames. In the next section, we will describe our method which is efficient, has a good visual quality and does not suffer from the singularity problem.

Skeletal Animation System

Skeleton animation is a general tool for performing body animations. Our method can be applied to a skeleton which is then used for skinning to achieve animation of virtual creatures. Our skeleton is composed by connected joints. Each joint has three degrees of freedom (DOFs) for rotations, and only the root joint has three additional DOFs for controlling the translation of the whole skeletal system. For computations, we define an up vector as a local axis for each joint. In the next section, we demonstrate how our method can be applied efficiently to such skeletons.

7.3 Our Mass-Spring Based Inverse Kinematics

As shown in previous section, different methods have been proposed to solve IK problem. They can achieve either a good visual quality or a good speed performance. It is hard to achieve both together. Since our model is based on physics simulation, it offers a good visual form, and the performance can also be more efficient than these previous methods.

An Efficient Energy Transfer Inverse Kinematics Solution We build a mass-spring system to deal with target reaching tasks. Then, a general angular constraint solver has been integrated in our system for solving the rotations. If the skeleton rotation information is not needed such as in the snake game [56], we can simplify the process by only performing the reaching task. The whole process can be separated into 3 steps (The code can also be found in Appendix 11.2):

1. apply mass-spring model to solve joints' positions (function: moveMassSpringSystem).
2. compute local orientation of joints (function: calculateLocalRotation).
3. solve the orientation constraints (function: checkJointsDOFsRestrictions).

7.3.1 Mass-Spring Systems (MSS)

Mass-Spring Systems are defined as harmonic oscillator systems in classical mechanics. In computer graphics, they are often used to model particle system. Positions are adjusted by minimizing the force energy which is conserved in springs. Hooke's Law [78, 123] defines the mass spring system as:

$$\vec{F}(t) = -k\ \vec{l}\ (t) \tag{7.3.1}$$

where F is a restoring force, proportional to the displacement l in a spring with a positive spring constant k.

The composition of forces received for a mass is defined as:

$$\vec{F_{sum}} = \sum_{i=0}^{n} \vec{F}_i\ (t) \tag{7.3.2}$$

We need to find the new position $x(t)$ for each mass to achieve a stable state in the system.

The formula needs to be extended to calculate the new displacement of a mass by using Newton's second law of motion:

$$F(t) = m\frac{d^2}{dt^2}x(t) + \frac{\beta}{m}\frac{dx}{dt} = ma \tag{7.3.3}$$

where F is the force, β is the damping factor and a is the acceleration of the mass. $x(t)$ is the displaced position of the mass at time t, velocity $v = \frac{dx}{dt}$, and acceleration $a = \frac{d^2}{dt^2}x(t)$.

To solve the particle path in our case, we use the force energy equation. This is suitable since the IK chain can be regarded as one particle chain. The path formula can be defined as:

$$y(t_1) = f(t_1, y(t_0)) \tag{7.3.4}$$

The basic method for solving the particle motion is based on Euler's equation $y_n = y_{n-1} + h \times f(t_{n-1}, y_{n-1})$. It is an approximation of Newton's method [35, 29].

In our case, we propose to use the Störmer-Verlet integration [134, 44] instead of Euler's equation. Comparisons between Euler and Verlet methods are illustrated in Figure 7.1. The Verlet method is more stable and has no significant additional cost over the Euler method.

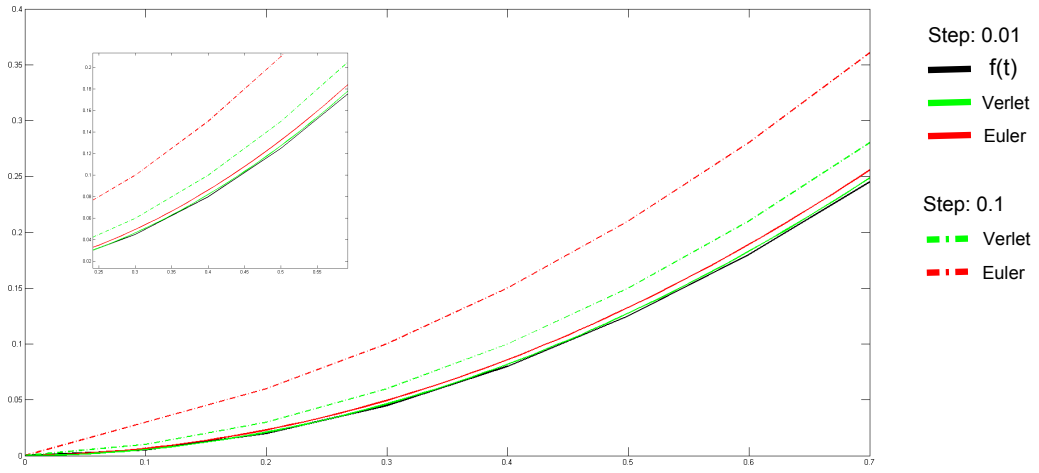

Figure 7.1: illustrates the difference between Verlet and Euler methods. The original function is: $f(t) = v \times t + 0.5 \times a \times t^2$, where v is the speed and a is the acceleration. With a larger step value, the Euler method induces larger error and proves less stable than the Verlet method.

The Störmer-Verlet scheme is defined as

$$\frac{\Delta^2 \vec{x}_n}{\Delta t^2} = \frac{\frac{\vec{x}_{n+1} - \vec{x}_n}{\Delta t} - \frac{\vec{x}_n - \vec{x}_{n-1}}{\Delta t}}{\Delta t} = \frac{\vec{x}_{n+1} - 2\vec{x}_n + \vec{x}_{n-1}}{\Delta t^2} = \vec{a}_n \tag{7.3.5}$$

$$\vec{x}_{n+1} = 2\vec{x}_n - \vec{x}_{n-1} + \vec{a}_n \Delta t^2, \tag{7.3.6}$$

where $\vec{a}_n = Acceleration(\vec{x}_n)$.

Let us consider an IK chain with a set of joints $J_i, i = 1,..,n$ and bones B_i with length $\vec{d}_i$. A set of mass points M_i and springs S_i is created by matching the joints and bones. The end effector is given as E and the target as T. For this computation, we ignore the rotation information, and set the general mass value equal to 1, i. e. the system only conserves the positional information of mass points. Two implementions can be used to solve the chain.

Method 1: Force based method We start each iteration by adding a user defined external force to the end effector: $f = -k(T - E)$. Then, we compute the in-

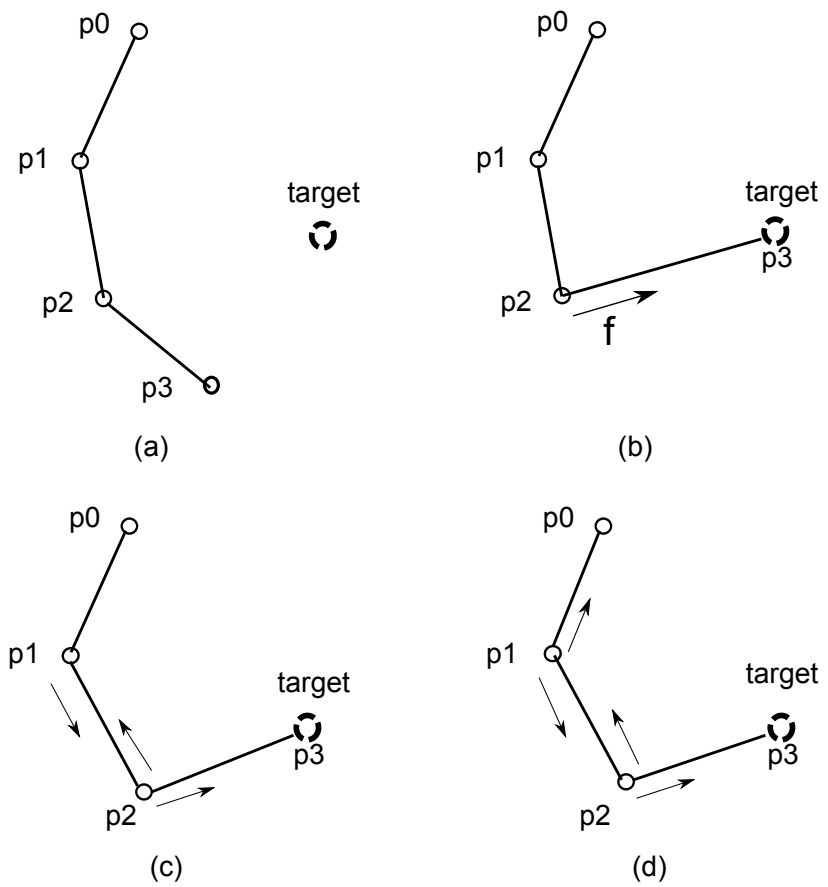

Figure 7.2: illustrates the procedure of our solver: (a) Initial configuration, (b) End-effector is moved to target position, (c) and (d) mass-spring system is simulated until equilibrium. p_i represents the joints on the chain, and each joint has a serial position x_n generated by our solver through time.

ternal force using the mass spring model. The acceleration can iteratively drive the mass points to the new positions by using Verlet's formula (Eq.7.3.6). During each iteration, we check the sum of internal forces of the entire system and make sure that $energy = \|\sum \vec{F}_{sum}\| < \epsilon$. We defined the ϵ as a threshold which influences the iterations. Finally, the end effector will always converge, i. e. get closer to the target. We keep the normalized directional vector of each pair of mass points (one mass point and its parent mass point) to build the local axis for each joint. The directional vector is also the joint directional vector.

Algorithm 1: ChainAlgorithm for solving positions in Mass-Spring System

Input: Masses Position p_i for $i = 1,...,n$ corresponds to Joints, Spring length s_j corresponds to the length of each bone for $j = 1,...,m$ bones, target position t for the end effector (p_n is the end effector's position)
Output: the new Masses Position p_i for $i = 1,...,n$ corresponds to Joints
Method:

1: // find the new target position by clamp function, if the target is not reachable
2: $t = clamp(t, p_i)$
3: // compute force on mass based on spring system
4: $energy = f_n = k(t - p_n)$
5: **while** $energy > \epsilon$ **do**
6: **for** $i = 0,...,n$ **do**
7: $p_i = Verlet(f_n, p_i)$
8: **end for**
9: **for** $j = 0,...,m$ **do**
10: $f_{j+1} = f_{j+1} + k(p_{j+1} - p_j - s_j)$
11: $f_j = f_j - k(p_{j+1} - p_j - s_j)$
12: **end for**
13: $energy = \sum \|f_n\|$
14: **end while**

Method 2: Position based method This is an optimized version of method 1. It is illustrated in Figure 7.2. We fix the end effector p_3 to the target position, using spring energy to correct the positions of mass points.

To achieve faster convergence, we use the clamp function [25] which retargets the chain to avoid too much oscillations. When the target is too far away, this function generates the new target that is in the chain's reaching space and makes the chain point to the original target.

The Clamp function for retargeting is defined as:

$$T_{new} = \begin{cases} T & if \quad d < D_{max} \\ J_0 + (T - J_0)\frac{d}{D_{max}} & if \quad not \end{cases} \tag{7.3.7}$$

where T_{new} is the new target, T is the original target, J_0 is the base joint, D_{max} is the sum of all bones' lengths, $d = distance(T, J_0)$.

After all iterations, we convert the mass system into a joint system. We have the directional vector of each joints pair (a joint and its parent joint). We then use the closed-form method [51] to compute the local rotation of each joint in a local axes

system defined by using its bone vector and its up vector which is perpendicular to the bone vector, and we update the whole skeleton system. The up vector is always perpendicular to the bone directional vector, and it is automatically updated when we perform a rotation on its bone vector. This is important when we need to solve general constraints.

Algorithm 2: Multi-Targeting Algorithm for solving positions in Mass-Spring System

Input: k sub-chains with k targets positions t_o for $o = 1,...,k$ and k end effectors E_o for $o = 1,...,k$
Output: the new Masses Position p_i for $i = 1,...,n$ corresponds to Joints
Method:

```
1: energy = ∑k(t_o − E_o)
2: while energy > ϵ do
3:    // compute for each chain IK
4:    for o = 1,...,k do
5:       chainAlgorithm(o)
6:    end for
7:    energy = updateSkeletonWithWeightMethod()
8: end while
```

7.3.2 Multi-Targeting

Models can be defined by one or more kinematic chains. For example, a human body can be defined by several chain constraints by multiple targets. Even a complex chain model can have multiple positional constraints, i. e. multiple end effectors. Therefore, it is useful to adapt the IK solver to support multiple targeting tasks. As mass-spring systems are used to cope with large data sets of particles, our mass-spring system based IK method can easily be extended to be suitable for multi-targeting tasks. The mass-spring system can also be applied in a parallel fashion [40]. The algorithm is as follows: All the end effectors are set to their target positions, then we start the iterations, as illustrated in Figure 7.3. When using multiple end effectors, there is at least one joint that is connected to multiple joints. We call such a joint a "sub base joint". In [6], the authors propose to calculate the centroid of the virtual sub base joint of each sub chain when doing the forward stage. The centroid of all the virtual sub base joints corresponds to the new position

of the sub base joint. In contrast, with our method, we do not need such a centroid supplementary stage; it is directly obtained. The last step of our algorithm returns to the skeleton structure. In skeletal models, one joint can only have one rotational value. We use either the priority or weight method [8] to convert from rotations to the skeleton structure. The first uses the preferential sub-chain rotation, while the latter the average by weights between sub-chain rotations. This process is only used for updating the rotation values from mass-spring system into skeleton system.

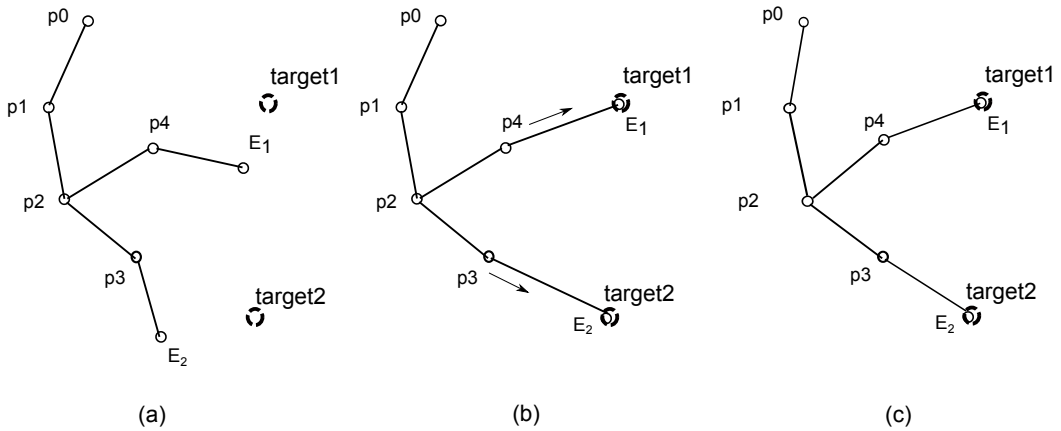

Figure 7.3: illustrates the multiple end effectors procedure: (a) Initial configuration, (b) End effectors are moved to targets positions, (c) mass-spring system is simulated until equilibrium.

7.3.3 General Joint Constraints

In most analytical skeleton models, joints can be modeled as hinges, pivots, ellipsoids, etc. Several anatomical and biomechanical methods exist that formalize specific motion intervals for articulated characters, with the main purpose of having multiple parameters to describe the hierarchical structure within a motion space.

We chose to use a ball and socket model for joints. The ball and socket joint is defined by 3 DOFs, i. e. with 3 euler angles, different rotation types are combined together. The euler angles can be converted from a quaternion [69].

Several methods have been presented for solving constraint problems. Zhao and Badler [151] introduced their Cartesian space constraints solution. Wilhelms and Van Gelder [142] define spherical joint limits with reach cones, that have been predefined by the user. The final joint value is obtained by projecting the joint segment

onto the reach cone to ensure that they are all within certain limits. Blow et al. [15] present a similar idea using a reach window, but they aim to find the closest point on the polygon instead of the direct intersection point. Korein et al. [66] decompose the ball and socket joint rotation into arbitrary orientation components, and directly control the rotation of the joint. Baerlocher and Boulic [7] also propose to define a two components constraint to stabilize the Jacobian method. Since our method has a similar position adjustment process as [6], our IK solver is compatible with their constraint solver for solving positional constraints.

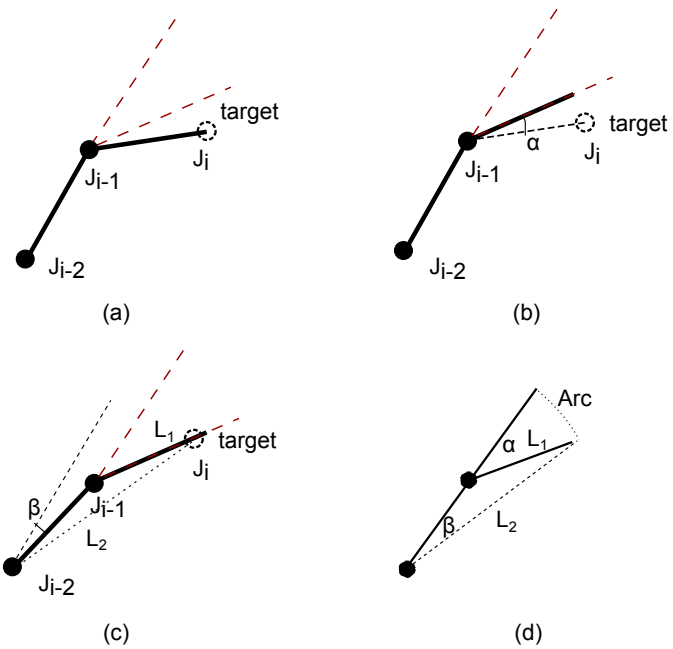

Figure 7.4: Example of mass-spring joint system constraints: **(a)** The end effector J_i reaches the target, but the rotation of J_{i-1} exceeds the constraints, shown by the dotted line. **(b)** Rotation of J_{i-1} to the nearest limit boundary with angle α. **(c)** Rotation of J_{i-2} to move the end effector closer to the target with angle β. **(d)** Approximation of β.

Our skeleton system is based on the conversion from global space position into local space rotation. We propose another angular constraint (boxed intervals) solver based on a parametrization of rotations in local Euler space with X, Y, Z axes. Nevertheless, all the rotations are computed and applied with quaternions before the constraint stage. We need to do the conversion between quaternion and euler angles [69]. Each DOF constraint is defined within range values. The minimum and

maximum values are used to limit the rotation of one joint in each DOF, resulting in six boundaries. The main idea of our method is to correct the local orientation, propagate exceeding values to the parent and iteratively converge to a good result within limits.

Algorithm 3: Algorithm for solving orientation constraints in X, Y, Z

Input: Joint local orientations q_i for $i = 0,...,n$ corresponding to Joints, q_i can be decomposed into $q_i.x$, $q_i.y$, $q_i.z$ 3 dimentions, their Joints' positions p_i, their 3d orientation constraints DOF_i, Target position t for the end effector,
Output: the new Joint local orientation q_i for $i = 0,...,n$
Metode:

1: **for** $i = n,...,0$ **do**
2: $q'_i = q_i$
3: **if** $q_i.x > DOF_i.x$ **then**
4: $q'_i.x = DOF_i.x$
5: **end if**
6: **if** $q_i.y > DOF_i.y$ **then**
7: $q'_i.y = DOF_i.y$
8: **end if**
9: **if** $q_i.z > DOF_i.z$ **then**
10: $q'_i.z = DOF_i.z$
11: **end if**
12: **if** $q_i \neq q'_i$ **then**
13: $q = \frac{q'_i}{q_i} \times \left\| \frac{t-p_i}{t-p_{i-1}} \right\|$
14: $q_{i-1} = q \times q_{i-1}$
15: **end if**
16: **end for**

We apply our constraint solver by checking all the joints' rotation values after the mass spring process. The order is from the end effector to the root joint. The rotation needs to be decomposed along 3 axes. As illustrated in Figure 7.4, we check the local rotation of J_{i-1}. If the rotation in local space is larger than the max limit angle or smaller than the min limit angle, it means the rotation value is exceeding the constraint value. We call this event a *constraint event*. In that case, we cut off the exceeded value of J_{i-1} and perform a second rotation β for J_{i-2} to reposition the end effector. We can easily compute β by the following approximation: $Arc = \alpha L_1 \approx \beta(L_2)$.

Afterwards, we check the local rotation of J_{i-2}. We proceed in a bottom-up fash-

ion, by pushing exceeding values of a joint to its parent in case the constraint is not satisfied until the base joint is reached. If the root also triggers the constraint event, we only cut off the exceeded value and restart the full loop until a solution is found or the max loops number is reached. This iterative process allows us to solve general constraints and to keep the end effector as close as possible to the target.

7.4 Results

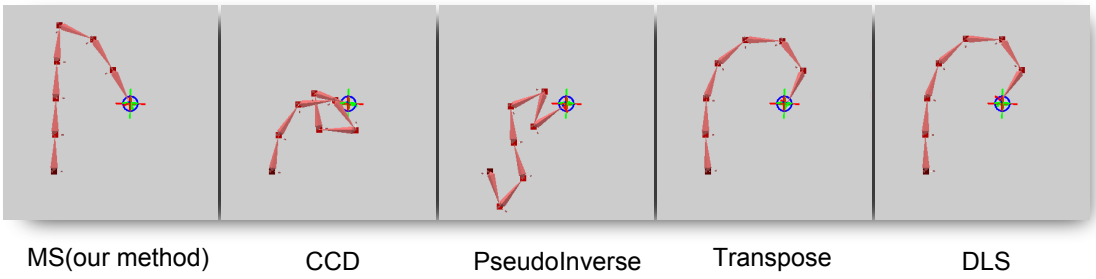

Figure 7.5: Results of using different IK solvers without constraints.

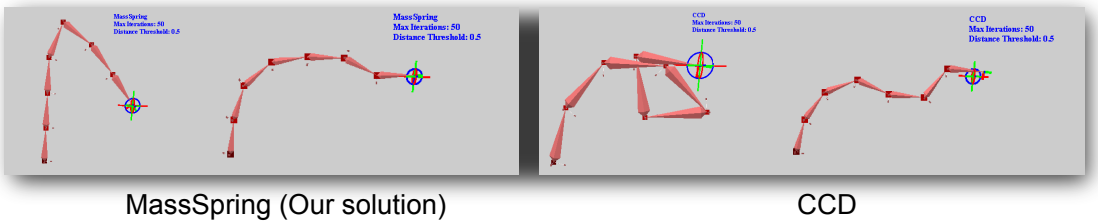

Figure 7.6: One comparison result between our method and CCD method: the CCD solution results in a chain that rolls around itself.

Our method has been implemented in Java and C++. We have implemented several other existing methods (CCD, PseudoInverse, Tranpose, DLS, etc) in C++ for comparison and experimental purposes. For each IK solver we have implemented, we measure the duration of the whole process including the computation of positions and rotations for the whole chain until the end effector reaches the target.

The results are illustrated in Figure 7.5. When compared to other methods, our approach gives good results with less change on the joints from initial states. More particularly, if we compare our method with CCD, our method does not show

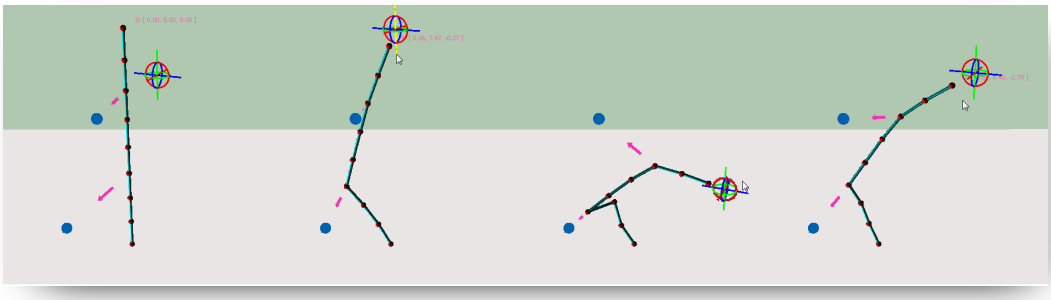

Figure 7.7: Our method supports multiple targeting constraints. The 4 images illustrate examples of a chain with two positional constraints and one target (using the manipulator). Our method can find the best solution for getting closer to both the target and the positional constraints. User-defined weights can be assigned to tell how both can be complied.

the rolling artifacts present in the CCD solution (see Figure 7.6), as mentionned in [6]. Our method supports multiple positional constraints as shown in Figure 7.7. It solves this case efficiently, while other IK methods suffer from artifacts in this particular situation.

Method	Joints	Iterations	Time (ms)
Our Method 1	6	49.6	0.195
Force-based	10	60.5	0.253
Our Method 2	6	26.9	0.132
Position-based	10	40.1	0.169
CCD	6	33.2	0.131
[137]	10	39.7	0.148
PseudoInverse	6	51.8	1.095
[116] [140]	10	55.9	1.310
Transpose	6	6.8	0.124
[144]	10	17.1	0.235
DLS	6	55.6	1.237
[25]	10	68.7	1.450

Table 7.1: Performance comparison between our method and other methods without constraints for a chain of 6 and of 10 joints. The results are averaged over 15 runs.

We also compared the performances of these methods with ours (we measure a whole updating process for both the position computations and the orientation computations), as shown in Table 7.1. Our method is as time-efficient as CCD but much

faster than the other methods. When an animation needs to be computed, we transform the position information into angular values. This allows us to interpolate these angular values for each frame. When such a transformation is not required as [56], our method can be even faster. We split the IK reaching task into two parts: reaching a target and updating the rotation while checking the constraints.

Limitation Our IK system suffers from the oscillation that derives from using Mass Spring system. Even though small oscillations will not influence the final posture after iterations, we do need to setup appropriate parameters for the mass spring system to reduce the oscillations.

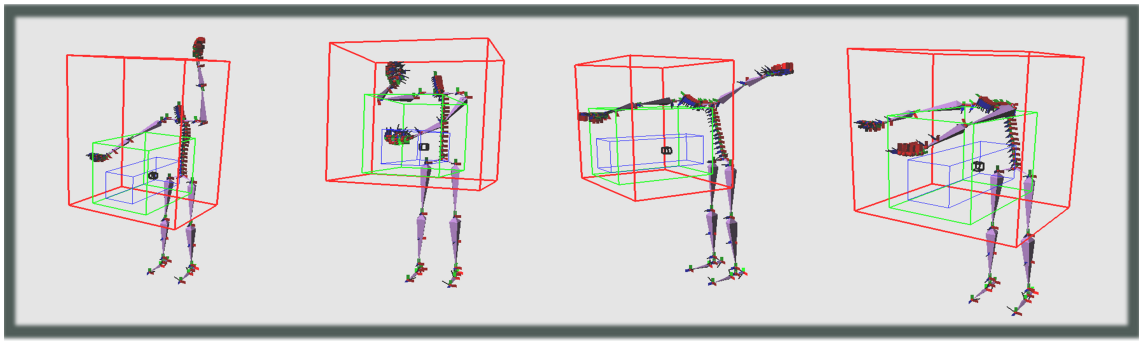

Figure 7.8: Some postures generated by «Greta» gesture editor using our IK solution.

Virtual Agent We have integrated our IK solver into our virtual character system (see Figure 7.8). The skeleton is decomposed into several chains (eg. the arm chain is defined from clavicle to wrist, the torso chain is connected by sacrum, lumbar, thorasic). So far we concentrate on upper body, which is mainly composed of 3 chains (left arm, right arm and torso chain), other chains can also be added according to user needs. Different constraints can be applied to these IK chains (see Figure 7.9). The animation of the virtual character is obtained through 2 steps: compute the body posture to reach the defined constraints and apply the movement expressivity parameters. These are later used to specify qualitatively the movement of the agent (such as the speed and acceleration of a movement, its tension and amplitude). We have merged our IK solver with our model of expressivity parameters [53] (see chapter 8). In particular, we use the arm chain's force energy and the value of the expressivity parameters to influence shoulders and torso movements.

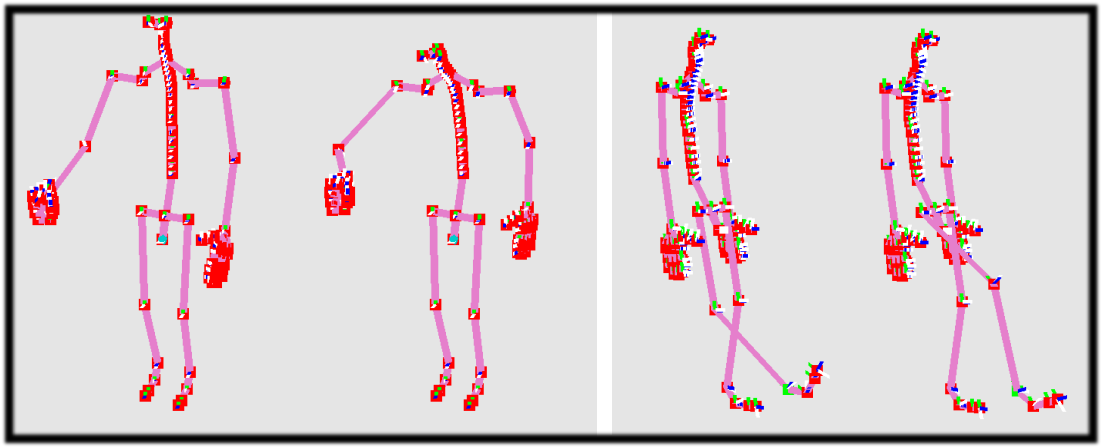

Figure 7.9: The IK chains of body parts are constrained: **(Left)**: The arm swivel angle (elbow) is constrained by the expressivity parameters [53]. **(Right)**: Our constraint corrects the leg's rotation.

7.5 Conclusion

We have presented a novel algorithm to approximate physically-based methods for solving inverse kinematics by using mass-spring systems. Our IK method can be applied to any chain type, with multiple constraints. It provides high quality visual results with computation times that are comparable and, at times, even lower than existing IK methods. Moreover, using mass spring system provides more control parameters, such as mass quality, spring stiffness and damping factor to tune the movement. Our model offers more flexibility to configure IK chains of any complexity. We have merged our model with a model of expressivity parameters to animate human characters that will be presented in the next chapter. Our method is also fast enough and stable enough to be considered as a good option in computer games.

Chapter 8

An Expressive Procedural Animation Pipeline

In the previous chapter, we detailed how to animate a human character using its skeleton, by controlling a very small number of articulations, while using a novel Inverse Kinematics method to optimize the global skeleton's structure. In this chapter, we show how to combine these standard deformation pipelines with our novel Expressive animation solution, that improves state-of-the-art's methods. We show the benefits of this technique in the context of human-machine interaction systems, where realism is a key to the comfort of the interaction between a human and an embodied conversational agent.

8.1 Introduction

Embodied Conversational Agents are virtual human agents that can communicate through voice, facial expressions, emotional gestures, body movements etc. They use their verbal and nonverbal behaviours to convey their intentions and emotional states. It is necessary for the ECAs to display a large variety of behaviours. ECAs are interactive entities; all their animations have to be rendered online.

Generating efficient and realistic animations of virtual creatures has always been a challenging problem in the computer animation field. Kinematics is a general method for manipulating interactively articulated figures and generating postures. In computer graphics, articulated skeleton models are used to control vir-

tual vertebral living creatures, such as human beings or animals, which appear frequently in films and video games. Inverse Kinematics (IK) (see chapter 7) is a method for computing joint rotation values of individual degree of freedom via predefined rotation and position constraints. In most of existing systems, animation of body parts is done independently of each others. For example, an arm gesture and torso movement are computed separately and then combined. Such a computation gives rise to unnatural animation and stiff-looking creatures.

Our work [53] focuses on the realization of animation for virtual conversational agents. Our animation model takes as input a sequence of multimodal behaviours to generate. It relies on a hybrid kinematics solution for generating full body posture. Moreover, our solution generates two types of motion. On the one hand it computes all movements specified over each modality (eg arm, torso or head movement). On the other hand, it also considers movements arising from other movements. For example, it will generate a torso and shoulder movement resulting from a given arm movement (eg, when the arm has to reach a distant point in space). Considering both independent and dependent movements can give rise to a more realistic and natural animation. Our model also embeds an expressive module with qualitative variations of body movements. Our algorithm is efficient; it can generate realistic whole body animations in real-time.

In the remainder of this chapter, we first present existing approaches, then turn attention to our approach. Afterwards, we describe our animation pipeline with the expressivity model. We illustrate our explanation with examples.

8.2 Previous Work

In this section we will present different works regarding expressivity and animation computations that are related to our work. Several approaches have been proposed to model expressive behaviours. The EMOTE system [28] introduced low level parametrization to generate expressive gesture. The parameters are abstracted from Laban principles (1980). The Greta system [102] [47], defines a low level parametrization derived from psychology literatures. In the realization of animation, both of these works decomposed character skeleton into small parts (the head, the torso, the arms, etc) and solved the system by different controllers acting locally. Such an approach does not allow modeling motion propagations, ie how mo-

tion over one modality may affect another one. In our work, we choose to deal with whole body motion with a global view.

Michael Neff et al. [97] [98] [99] presented their aesthetic motion generation system. Their model starts from a high level expressive language that is translated into precise semantic units that can be simulated by physical or kinematics methods. The translation mechanism encloses a selection and a refinement step for choosing the gesture movements.

SmartBody system [129] proposed a controlling system that employs arbitrary animation algorithms. The system can schedule different task controllers, and also allows realizing modular animation control and propagating motions over the body parts. As SmartBody, we achieve a whole body management from a lower level. But our approach is based on the hierarchical dependency of the agent's body structure.

The EMBR system [50] offers an animation pipeline with their motion factory, scheduler and pose blender modules. The animation to be realized is described with the EMBR script. It can deal with different formats of animation (IK, motion data, etc). Our system follows a similar animation pipeline. However our pipeline solves the conflict for body parts. Indeed, our system is based on IK techniques. IK is used to do retargeting, and it makes the final decision for key postures. Several inverse kinematics [8] [18] [49] methods are also proposed to achieve reaching tasks. Such methods need to calculate the whole body posture in realtime for a given trajectory of the wrist position in space.

Tan [127] talks about the importance of using the postural expressions to express action tendencies. Meanwhile, behavioral studies on posture have also been made [63]. Although their studies are based on static postures, the authors noticed that expressive postures are rather important. They also note that generating automatically variations of expressive postures is useful for simulating human-like animation.

8.3 Overview of our pipeline

We have implemented a framework of embodied conversational agents that respects the SAIBA [68] framework illustrated in Figure 8.1. Our framework takes as input a file described with FML the standard Functional Markup Language which defines the intentions and emotional states in a high level manner. The

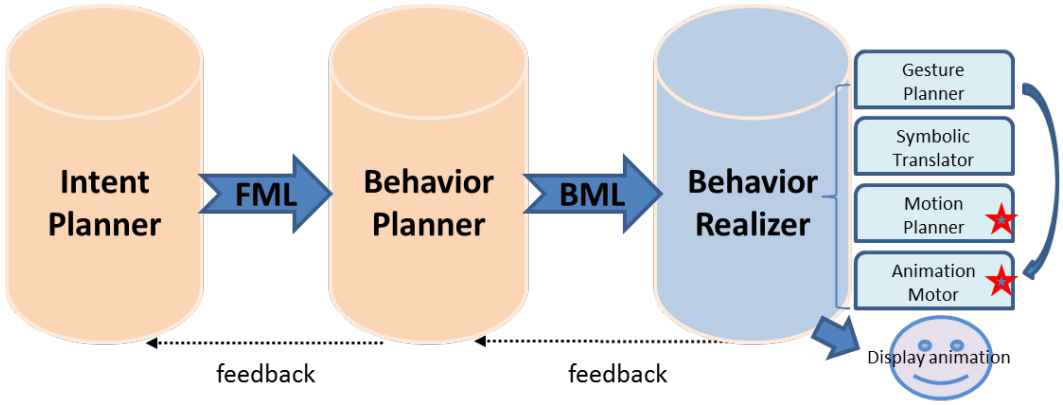

Figure 8.1: The standard SAIBA framework defines the modularity, functionality and the protocols for ECAs. In the Behaviour Realizer module, the parts with a star correspond to our work in the animation pipeline.

Behaviour planner translates the FML tags into sequences of standard BML, Behaviour Markup Language, entries [68]. Sequences of time-marked BML-like signals are instantiated within the Behaviour Realizer. Based on these high level definitions, we use our animation pipeline to build low level animation sequences.

Our animation pipeline starts by receiving BML-like symbolic signals time stamped in the motion planner. All signals are received by streaming, and hence our animation computations need to be achieved on the fly. Each signal (hand, torso, head, etc) includes gesture phases, expressivity parameters, gesture trajectory and the description of shape and motion. As shown in Figure 8.2, signals can be scattered (step 1) from different modalities, ie torso movements, head movements, hand gesture movements (two groups: left and right sides). After receiving scattered signals, we apply our pipeline to generate our animation sequence illustrated in Figure 8.2.

In the remainder of this chapter we detail each component of our pipeline. Expressivity parameters are presented in section 8.7.

8.4 Targeting process

The targeting process describes the hand gesture trajectory [47]. This is often referred to the "Path driven" approach that is used to generate new key frames for trajectory path. Path form, such as line, circle, can be defined by mathematical

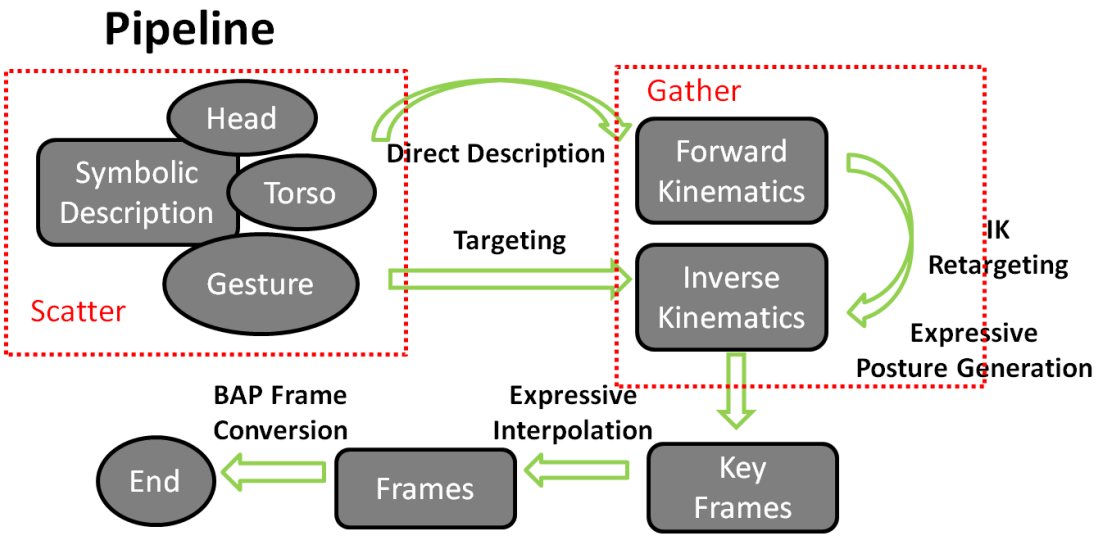

Figure 8.2: The animation pipeline takes as input sequences of symbolic gestures. It computes the whole expressive animation.

functions. For building these path forms without the dynamic branching, we chose to use an approximation of a sinus function: $f(t) = R * sin(T * \pi * t + Shift)$, where R is the amplitude that defines the radius of local circle, T is the temporal variation of frequency, $Shift$ controls the path direction. The final path position P is defined as $P = P_{center} + P(f(t_x), f(t_y), f(t_z))$. By varying the 3 parameters, we can construct different gesture paths. For example, for linear path, T value can be just set to zero. Some other possible gesture paths are saved as sequences of key points corresponding to 3D positions in files.

8.5 Gathering process

The purpose of the gathering process is to generate the full body key frames that corresponds to body postures. We chose to compute the full body posture as a whole and not as a concatenation of body parts positions as it allows capturing dependent movements across body parts. For example, reaching hand or gaze targets may affect torso movements. When looking on the far left, head, shoulder and torso are all turned to the left.

We sort out sequences of all body parts into one list of key frames ordered by time markers. Information of each key frame is filled with the information from

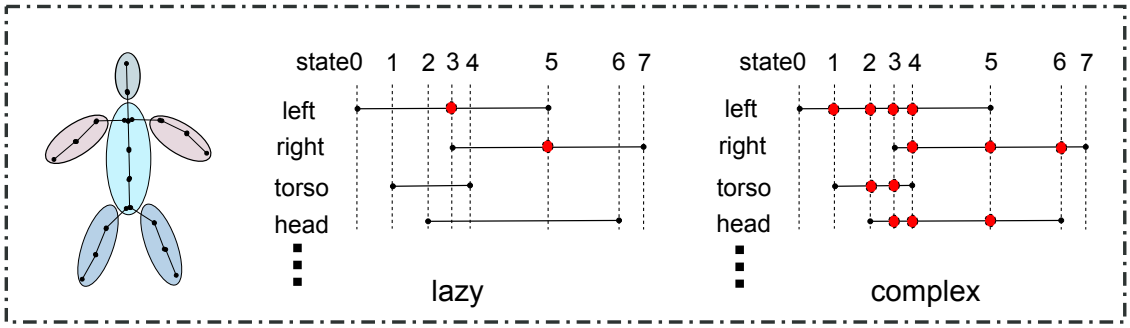

Figure 8.3: **(Left)**: shows the character skeleton decomposition, **(Middle)**: shows the lazy approach. **(Right)**: shows the complex approach.

the body part sequences. To complete the key frame specification, we can use either the "lazy" approach or the "complex" approach that are illustrated in Figure 8.3 and 8.4. The "complex" approach considers that all the movements are equally important, and we need to fill all the body parts for each key frame by interpolating between the previous element and following element of its sequence (linear interpolation); On the other hand, the "lazy" approach privileges some movements. e.g., if a key information has only torso movement, then the torso movement is dominant, and we can only apply torso movement. If a hand gesture is missing, but for one given sequence animation, it is very important, then we must fill it. For the lazy approach, we only fill in existing parts for each key frame and interpolate the important missing movements. The importance is defined by priority parameters. The lazy approach is flexible and has less computation. Both of these two methods can be used to generate the insert values for key frames (see Figure 8.4) The resulting key frame list is used in the posture generation stage for creating posture key frames.

8.6 Posture generation

To compute a body posture, we can apply forward kinematics, inverse kinematics or a mix of both techniques. We use different body parts information as inputs which describe the postures, and outputs will be the final key frame postures (generated by forward and inverse kinematics). The forward kinematics uses the description defined in the gesture phases to realize the initial states of the key frames.

When needed, we retarget the gestures by using inverse kinematics. A hybrid

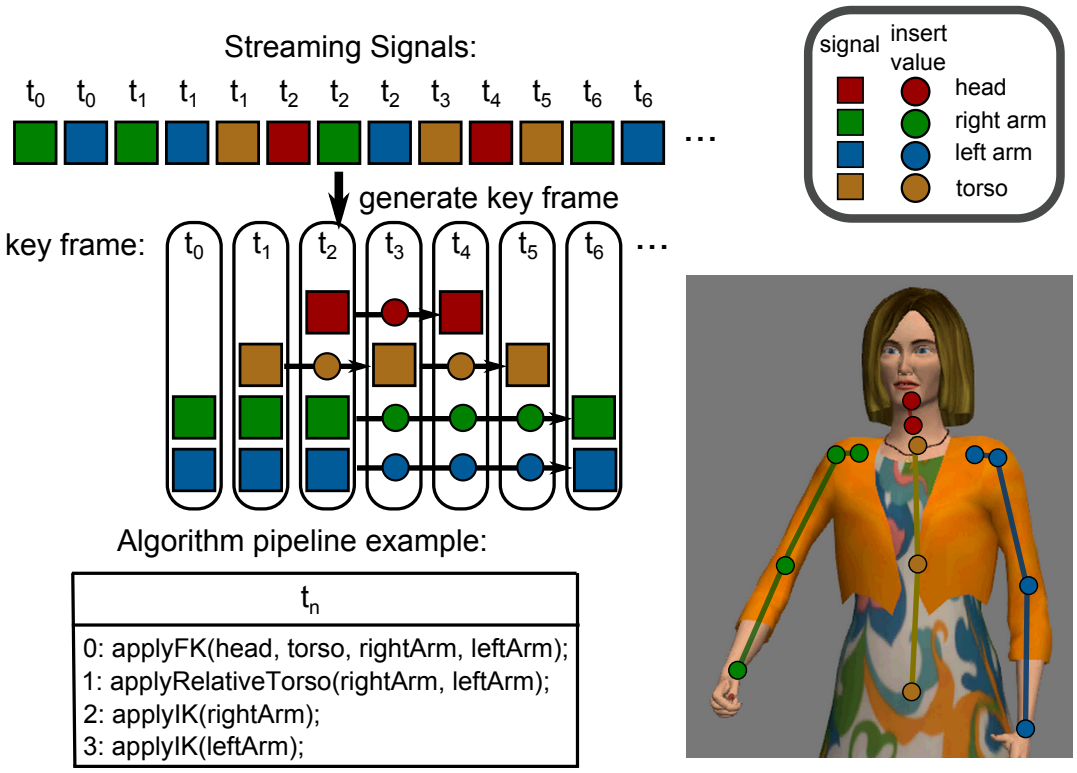

Figure 8.4: **(Top)**: signals from body parts are streamed into our system to generate key frames, **(Bottom)**: our pipeline combines both FK and IK to generate final postures. Insert Value can be filled by using "complex"(by interpolation of key values) or "lazy"(by previous key value) methods.

inverse kinematics solution is used to solve dependency between body movements. For head and torso movements, we use simple analytical methods. For shoulders and hand gestures, we use our constrained mass spring IK solver (see section 7.3). However, we have to make sure that all targets are in the reaching space of the arms combined with the torso. If the target is too far away, the movement is transformed as hands pointing in the direction of the target. We do not consider foot step forward to solve this situation.

More concretely, we start by computing the potential target of torso. We check the targets of both hands $T1, T2$. We look if the hands movements need torso movements. The arm length l_{arm}, the vertebrae (**vt1**, **vl5** in MPEG4 H-ANIM) positions P_{vt1}, P_{vl5} are already defined in the skeleton system. The horizontal pointer of torso is the direction of the center of hands reaching targets. The amount of vertical lean

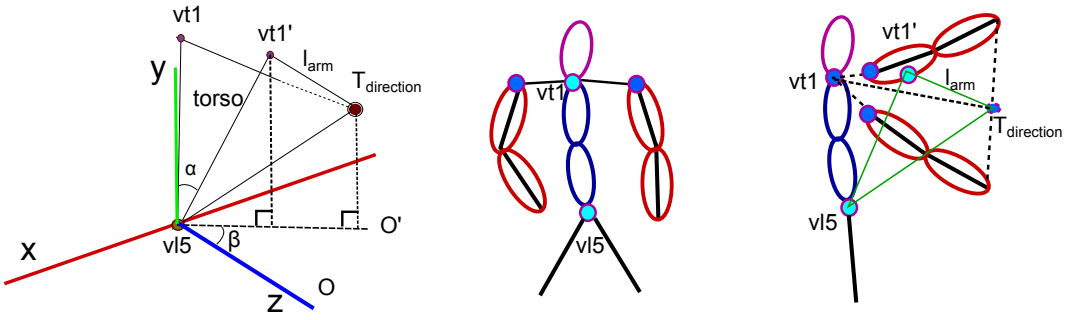

Figure 8.5: shows the trigonometry approximations for computing relative torso rotation. In some situations where the target is out of the reach of the arm, the torso needs to be animated accordingly. $T_{direction}$ is the center of hand gesture positions. l_{arm} is the length of arm. α is the rotation of vertical lean. β is the horizontal turning rotation.

is computed by using trigonometry approximations (see Figure 8.5). Then we compute the hand gestures with this new torso posture. We apply our MassSpring IK solver (see section 7.3) on the arm chain, which can apply light weight shoulder movements by defining arm chain from the sternoclavicular till the wrist. It allows us to model passive shoulder movement (see Figure 8.6). The whole Design-Flow of generating a posture (IK) can be explicitly defined as follow:

1. compute Torso movement based on the potential target defined by the hand gestures (Figure 8.5).
2. compute hand gesture for both arms using our MassSpring IK chain solver.

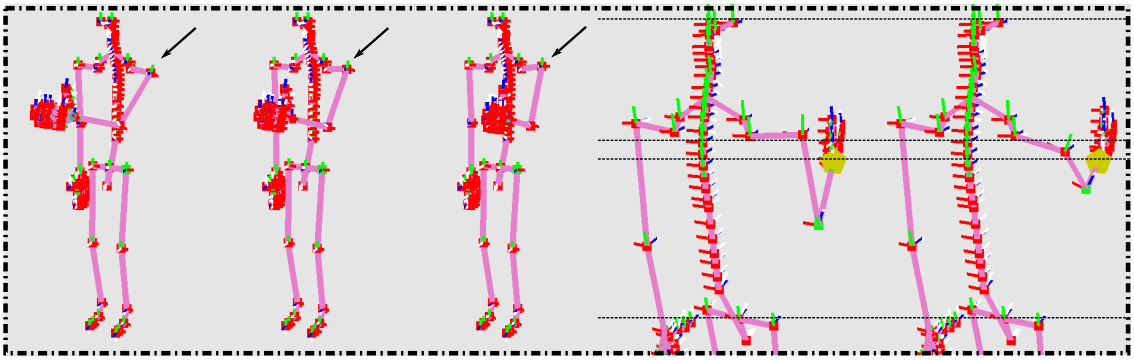

Figure 8.6: shows the passive shoulder movements. The variation results can be generated by applying different force energy values on IK chain with the same target: larger values imply more movements on the shoulder.

After this IK stage, we generate the animation sequence of the key frames using simple joint rotation information. We use the quaternion based spherical linear interpolation (slerp) to generate all frames, and convert them into "BAP" frames (Body Animation Parameter (MPEG-4) frames).

8.7 Expressivity

Symbolic Representation In communicative gesture system, it is hard to use continue space values to represent all gesture spatial information. Based on previous research [88] [47], our system also choose to model the gesture space using symbolic representations in 3 dimensions by McNeill's sectors [88]:

1. X: XEP, XP, XC, XCC, XOppC
2. Y: YUpperEP, YUpperP, YUpperC, YCC, YLowerC, YLowerP, YLowerEP
3. Z: ZNear, ZMiddle, ZFar

These symbolic values are useful from a psychological point of view. All the translation from symbolic to special real values for each agent are made by user experimentations from communicative study experts in previous research [88] [47].

Expressivity Parameters We define the expressivity as the manner of executing the gestures. We have modulated the quality of body movements by a set of expressivity parameters. We group these parameters into 3 sets: Gesture Volume controls the spatial variation of gestures; Sequential Variation controls the time based variation; Power Variation allows us to further control the dynamism of these movements.

Gesture Volume The idea of Gesture Volume is to use certain parameters to control the spatial form of posture shapes. The spatial parameter is used to control the variation of McNeill's sectors [88] (see Figure 8.7). We have two levels of control over these sectors. The first one is based on the spatial parameter which will scale the sectors using the same method as [47]. The second level adjustment depends on the torso position. The sector centers would be influenced by a scale factor and a rotation factor which are given by the torso. The scale factor $s = (P_{vt1'} - P_{vl5'})/(P_{vt1} - P_{vl5})$ is the ratio between the new length and the old length of the torso. The rotation

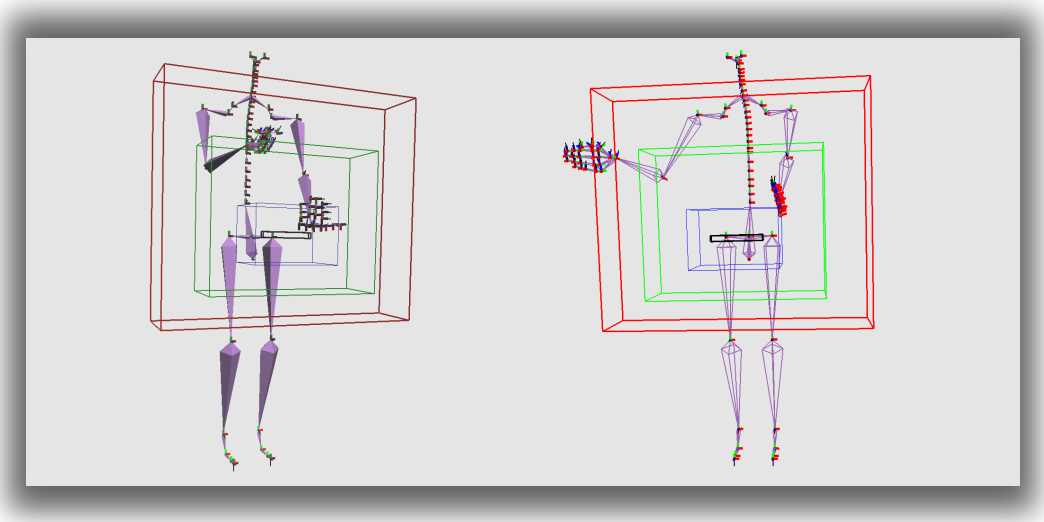

Figure 8.7: The McNeill's sectors are defined in front of our virtual character that define the gesture space.

factor is the rotation value applied on **vl5** after the forward kinematics stage, which is used to change the sector box's orientation.

We also have the openness parameter that changes the gesture form computed in the IK stage. Its value ranges between 0 and 1. A small value tightens the body; it makes the elbow closer to the center; and a big value increases the space between the arm and the torso as illustrated in Figure 8.8 on the left. The openness parameter affects the orientation of the elbow swivel angle [131]. A larger openness value also applies a bigger rotation on the torso that indirectly releases the tension of the arms as shown in Figure 8.8 on the right.

Sequential Variation Three parameters, "Fluidity", "Bias" and "Tension", are used to modulate the gesture path as well as its timeline. Our signals are already time-stamped in the scatter module (see Figure 8.2). "Fluidity" refers to the degree of continuity of a movement. To simulate it, we use similar idea as proposed by [28] [47] to parametrize the Kochanek Bartels splines (TCB splines) [64] (see Figure 8.9). We define "Bias" as a force that generate an overshoot for certain acceleration. "Tension" [112] describes the amount of energy that has to be expended for some positions but not others, and hence more effort needs to be exerted for the gesture to keep its original position. These two last parameters are simulated by

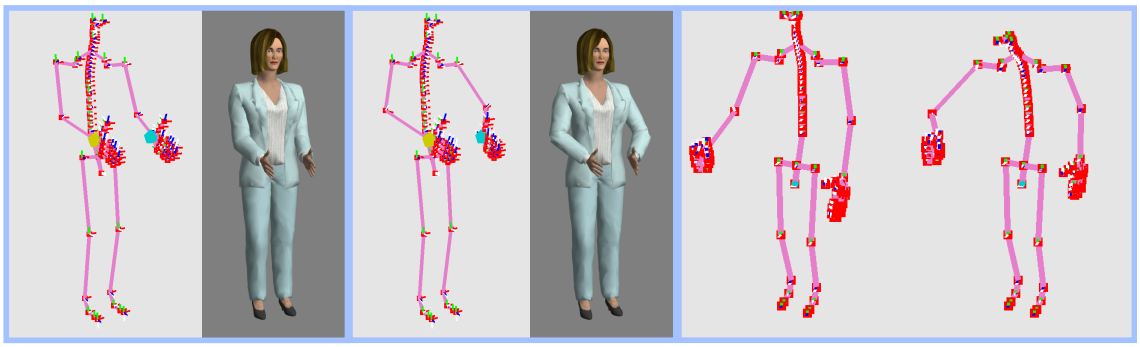

Figure 8.8: Examples of different values of the openness parameter: **(Left)**: openness influences the gesture form. **(Right)**: openness influences the torso position.

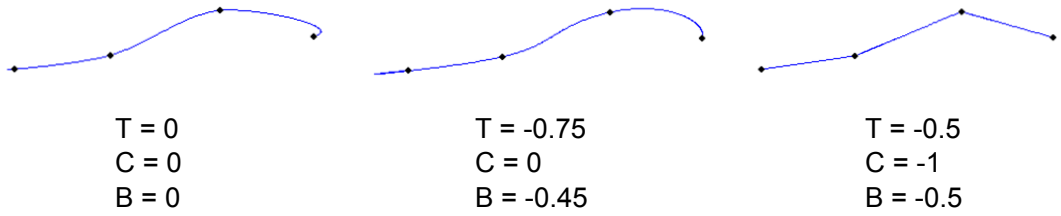

Figure 8.9: The trajectory changes when we apply different parameters: "Tension", "Continuity", "Bias".

varying the bias parameter and the tension parameter of TCB splines.

We also simulate accelerations with "Power" parameter by using various easing in-out functions (see Figure 8.10 and 8.12) to change time stamps for key frames (interpolation for target path) and frames (interpolation for joint rotation).

The formulas are defined as (see Figure 8.10):

- **Linear**: $f(x) = x$
- **OutQuart**: $g(x) = -((x-1)^4 - 1)$
- **OutBack**: $h(x) = (x-1)^2 * ((1.70158+1) * (x+1) + 1.70158) + 1$

We use Linear function for normal gestures with a "Power" equals 0, and OutQuart function is used for having acceleration effect. OutBack function can be used to simulate the overshoot in rotation space. We also use easingOutBounce function (see Figure 8.12) to simulate big tension with rebounce effects on elbow and shoulder rotations in rotation space.

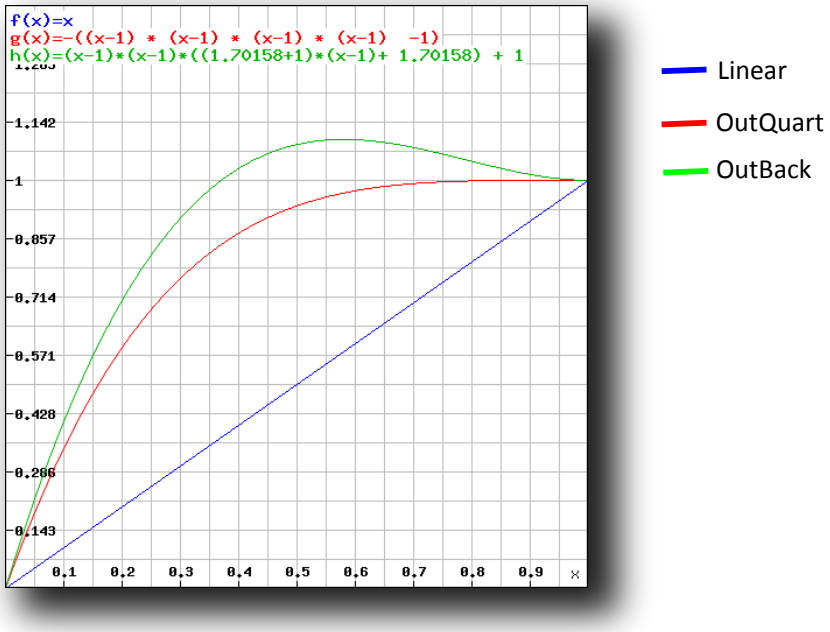

Figure 8.10: Three different easing in-out functions with implementation formulas.

Additions: Power Variation In our system, the power parameter does not only affect gesture speed, but also key frame postures. It means that a larger value of power influences more body parts. For example, a movement of the arm done with a high power will affect also shoulder and torso movements. This propagation of movements between body parts is possible as we use a full body IK framework. Torso can be affected by hand gestures as we define target energies. It is a similar idea as the constraints priority [8]. Our hybrid solver builds upon an interactive method which is controlled by the "Power" parameter. As "Power" increases, it can influence the whole body gesture, both shoulders and torso. For the shoulder part, as we mentioned in the section 8.6, passive motions can be generated by Mass Spring IK chain. For the torso part, we have an expressive torso controller based on hand gesture key frame states. Our torso controller is defined using a proportional manner to the "Power" parameter:

$$\Delta\tau = p \times (\theta_{t1} - \theta_{t0}) \tag{8.7.1}$$

where $\Delta\tau$ is the variation of current torso rotation, p is the "Power" parameter, θ_{t1} is original rotation of current key frame, θ_{t0} is original rotation of last key frame.

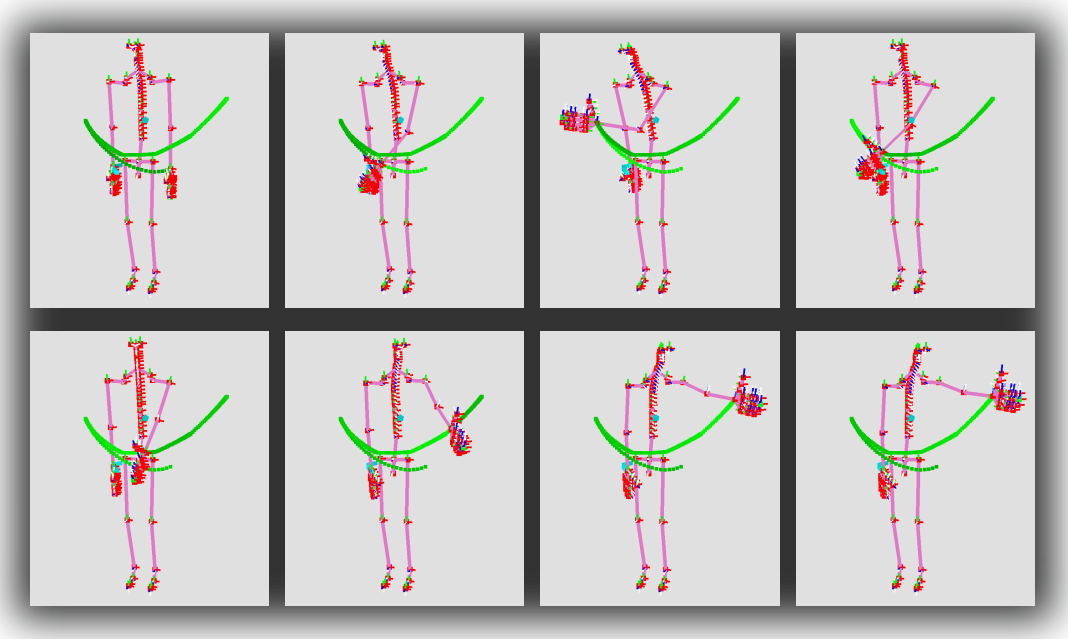

Figure 8.11: One sequence animation using IK postures of our skeleton

8.8 Results of our Virtual Agent

The described pipeline has been integrated into our virtual agent framework. The skeleton system is based on MPEG-4 H-ANIM 1.1 model. Our virtual agent system takes as input an FML file. It instantiates this input file into a sequence of BML tags, which is the input of our animation pipeline.

Our solution performs in two passes: (i) forward kinematics specified by the BML tags; (ii) inverse kinematics that further influences the body depending on the energy descriptions and the expressivity parameters. We can note that torso and shoulder movements can be automatically generated due to movement propagations. Such motion can happen implicitly without any torso movements defined by BML tags, as illustrated in Figure 8.11. The expressivity parameters can modify a given gesture and perform various movements for our virtual characters. As you see in Figure 8.13, with a same gesture defined, large spatial parameter generates

further reaching target for IK that creates larger gesture space, and larger power parameter offers a large expressive torso movement (relative motions). We also build our gesture editing tool (see Figure 8.14) which is used to generate different gestures for our gesture library.

8.9 Conclusion and Discussion

In this work, we have presented a full body expressive animation pipeline using procedural key framing method. We have proposed a hybrid approximation for posture computation based on the dependency of body parts. Our pipeline is compatible with existing target reaching models. Our expressive posture method provides high quality visual results in real-time. This system offers more flexibility to configure expressive FK and IK. It can be extended to other articulated figures.

In this work, we didn't provide the perceptual study. We prefer first analyze motion data for emotional expressive parameters [11], and then build animations with these reasonable parameters. The perceptual study will be done after this process. Even though our strategy improves the input animations, the modified animation still lacks human-like behaviour for now. Improving further the results would require more advanced strategies allowing the embodied agent to act with respect to the context or the on-going conversation. Our pipeline could be extended to map gestures that were recorded using performance capture data, allowing to reproduce faithful and realistic human gestures.

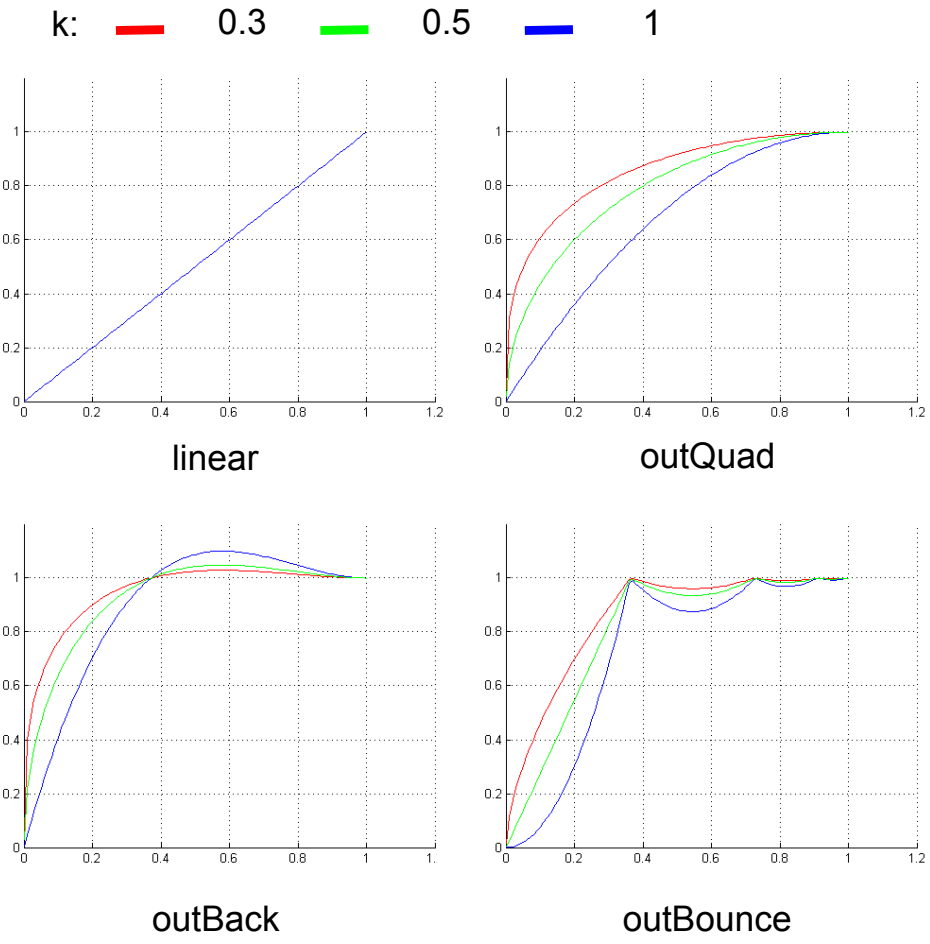

Figure 8.12: 4 examples of easing functions being used to change the animation time line, with different values of steepness factor k.

```
<gesture category="test" id="test">
    <phase type="STROKE-START">
        <hand distanceFixed="true" fingersOrientationRatio="1.0"
            handShapeFixed="true" horizontalFixed="true"
            palmOrientationRatio="0.0" side="Right" verticalFixed="true">
            <verticalLocation>YUpperC</verticalLocation>
            <horizontalLocation>XC</horizontalLocation>
            <locationDistance>ZNear</locationDistance>
            <handShape>form_fist</handShape>
            <palmOrientation>INWARD</palmOrientation>
            <fingersOrientation>AWAY</fingersOrientation>
            <palmOrientationSupplementary>DOWN</palmOrientationSupplementary>
            <fingersOrientationSupplementary>AWAY</fingersOrientationSupplementary>
        </hand>
    </phase>
    <phase type="STROKE-END">
        <hand distanceFixed="true" fingersOrientationRatio="1.0"
            handShapeFixed="true" horizontalFixed="true"
            palmOrientationRatio="0.0" side="Right" verticalFixed="true">
            <verticalLocation>YCC</verticalLocation>
            <horizontalLocation>XP</horizontalLocation>
            <locationDistance>ZFar</locationDistance>
            <handShape>form_3apart</handShape>
            <palmOrientation>INWARD</palmOrientation>
            <fingersOrientation>AWAY</fingersOrientation>
            <palmOrientationSupplementary>DOWN</palmOrientationSupplementary>
            <fingersOrientationSupplementary>AWAY</fingersOrientationSupplementary>
        </hand>
    </phase>
</gesture>
```

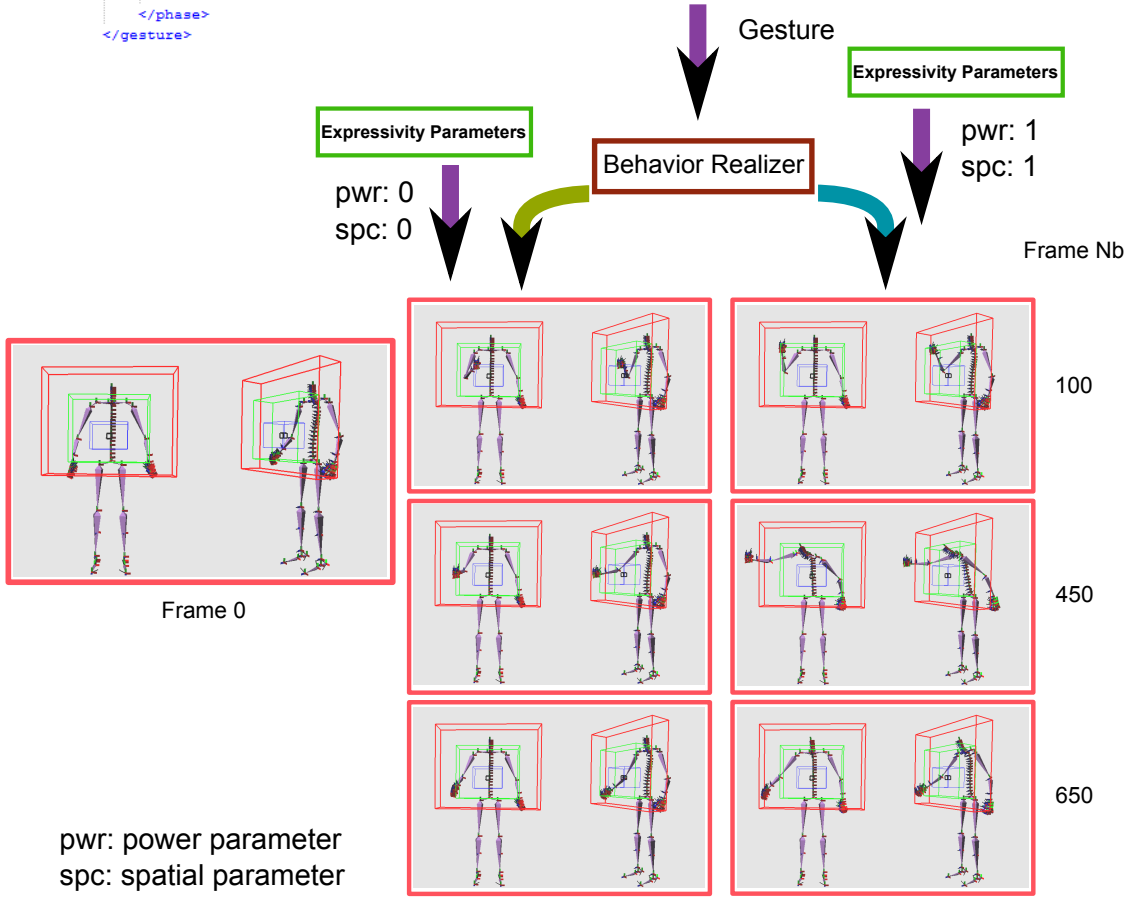

Figure 8.13: A given gesture being modified by the expressive parameters to generate two different movements.

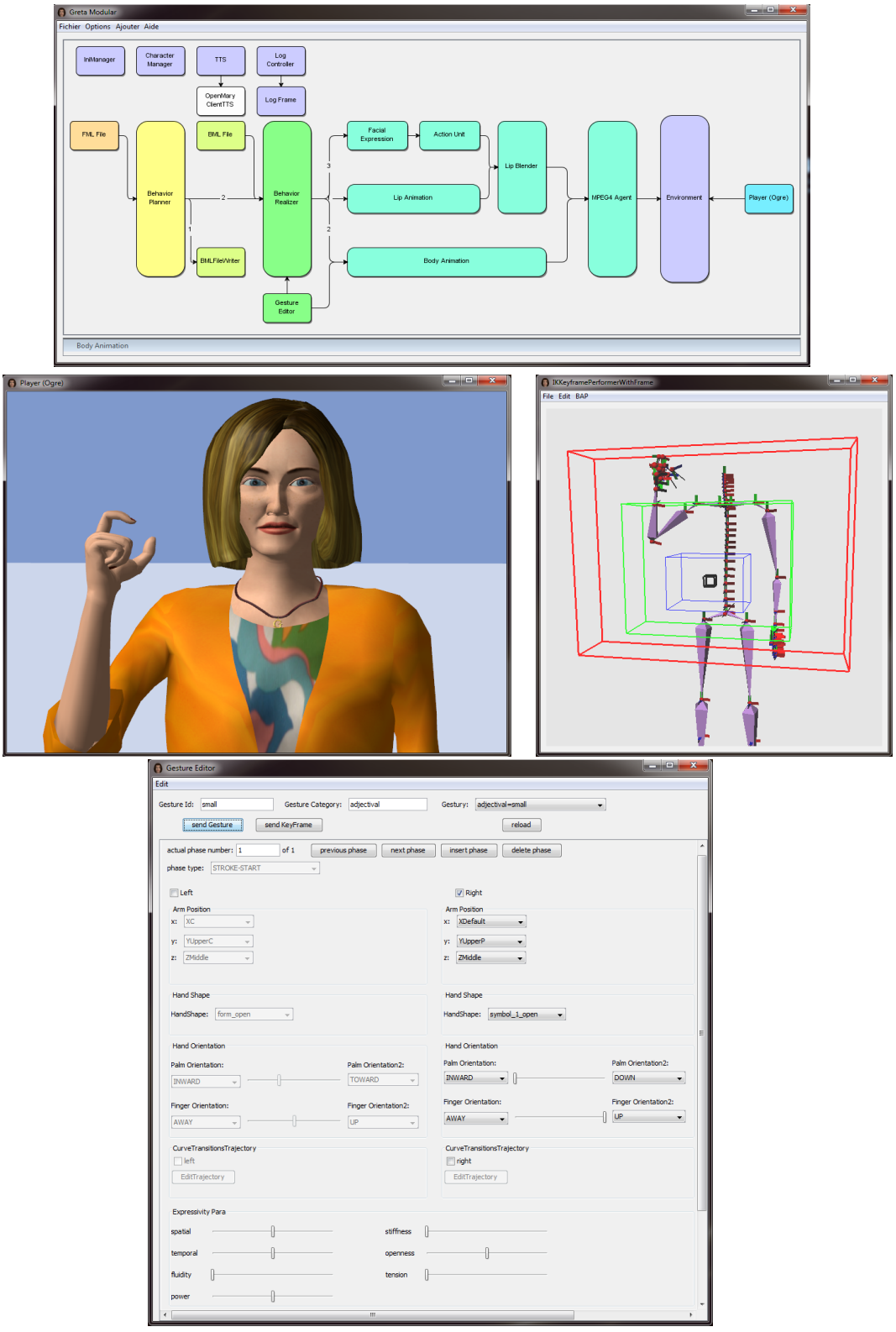

Figure 8.14: Some modules in «**Greta**» system: **Above**: greta modular being used to manage the whole system, **Middle**: greta players being used to visualize the virtual world, **Below**: greta gesture editor being used to generate gestures.

Part III

Conclusion and Perspectives

Chapter 9

Conclusion

In this thesis, we explored new efficient methods for both rendering and animation. These new approaches can be used to animate 3D characters and to render them with high performances and plausible expressive qualities. The efficient IK solution is used to generate procedural animation, it can be also extended for other type of motions and activities. Our rendering methods can alternatively be chosen to compose the final images by using pure pixel level computations.

In most cases, the animation and rendering algorithms are independent of the graphics pipeline. The link between these two parts is the data structure. On the one hand, the animation process prepares and parameterizes the geometric information. On the other hand, the rendering process revisits the geometry and appearance data, and reuses them to build 2D pixel data, resulting in realistic lighting at interactive speed.

In our project "Mass Spring Inverse Kinematics", we explored the use of physics and analytical solutions to generate the chain postures. To find suitable positions for all joints in a chain, we map joints into the mass spring system by defining the equilibrium length equal to the length of bone. The spring energy transfer mechanism leads to a stable state of the whole system. The stable state could be a solution for the original chain. Therefore, we solve the chain in the position space instead of the traditional rotation space. The convergence is fast when using Verlet integration. Secondly, if the rotation information is needed, we apply analytical solutions such as Horn's close form method to compute the spatial local rotations. Our solution can easily be combined with different constraints. The simple boxed intervals constraint performs the angular correction when computing rotations and doing

conversion between Euler angles and quaternions. Multiple targeting constraints can be applied to different masses by adding supplementary force energies without increasing the computational cost too much. Thus, the process is more efficient. We demonstrate that our method is comparable to existing methods. This method is later used to generate key frame for our procedural animation system.

Our procedural animation pipeline is an important part in our conversational virtual agent system «**Greta**». We use the IK method to generate full body postures. Expressive character animation for virtual agents is different from simple IK posture generation. Since character animation needs to specify different modules corresponding to body parts. Many existing systems compute movements locally for each body part, and blend the information together in the end. In this case, it is hard to have a full body performance with interactive motions between body parts. Therefore, we build each key frame by synchronizing all information of the body. Then, the full body IK solution is combined with parameters regarding expressivity to generate various body postures. Relative shoulder movements and torso movements are modulated by expressive factors. We also use different easing equations to simulate expressive gestures, such as power, tension, etc. Since procedural animation methods can create motions efficiently, they are appropriate for real-time usages, but they still lack lifelike behaviour. For our future research, we will combine procedural approaches with motion data based methods to improve the quality of our animations.

Regarding expressive rendering, we improved the Screen space ambient occlusion method for the indirect lighting simulation in real-time applications by using a separable approach. A simple method from Crytek takes samples around the current pixel and performs a depth comparison between each sample and this pixel. The AO value is the average of comparisons. This method is fast but only provides a low visual quality approximation to global illumination. Other methods such as Ray marching can offer a higher quality but they are less efficient. Our method computes the AO value by integrating the cone angles of the occlude around the current pixel. This guarantees high quality of the AO term comparable to the screen space ray marching method. We also take advantage of local randomization with interleaved sampling which is rather efficient. A separable feature-preserving smoothing kernel that matches the size of the randomization kernel is used to soften artifacts, and, at the same time, to conserve the geometric appearance. We show

that our method offers good quality with higher performance. It inherits all SSAO properties such as the natural view-dependent adaptivity and its ability to deal with arbitrary dynamic scenes. Our approach is fully compatible with existing AO methods and can be integrated easily into modern rendering pipelines.

Original wrinkles are the natural deformations on human faces. By applying transformations on the normal vectors in wrinkle regions, we do not need to change the surface of the 3D model. A simple and low cost wrinkle simulation is performed on our virtual character by using a texture-based bump mapping technique. We use Blinn's idea and profit from the advantages of parallel computing to compute the perturbed normal in a fragment shader. Each wrinkle is linked with specified AUs. The facial AUs trigger the corresponding wrinkles. Our facial wrinkle model is completely dynamic and suitable for facial animations. With the help of our wrinkle model, we evaluate the factors that may influence the identification of each single facial AU. We show that a high intensity and a dynamic image sequence can improve the identification of facial AU. Even though the contribution of our wrinkle model for such a process cannot be proved, the wrinkles do increase the perception of reality and naturalness of our virtual characters' facial animations.

We also implemented some other screen space rendering techniques, such as subsurface scattering for human faces, real-time colored transparent soft shadow mapping, a gather approach for the depth field, etc. All these methods are implemented in our renderer «**Edison-Etoile**». Presently, each algorithm has its own computation cost. A simple combination of them is not acceptable for real-time tasks. We are still trying an efficient combination to manage all effects together to achieve both good quality and high efficiency.

Overall, we introduced several new techniques and extensions of methods for both rendering and animation field: (i) we explored screen space technique for simulating lighting effects which can improve both speed and visual quality for real-time renderer. (ii) we proposed an expressive key frame posture animation pipeline using an efficient IK solution. We introduced relative motions for having a full body performance. Expressivity parameters are used to modify both the key frame posture and the animation sequence. Our solution improves the quality of virtual agent body movements with high speed performance. We would like to continue our research and contribute to high quality and efficient algorithms for computer graphics.

Chapter 10

Perspectives

Motion Capture and Large Graph Model Searching

Procedural generated animations (see chapter 8) are still lacking naturalness. In particular, human character gesture simulations often result in rather robotic and ragdoll like animations. More lifelike animated characters can make the immersive virtual environment more realistic and more believable. Meanwhile, the high quality and high performance animation synthesis is still a big open challenge. One common solution is to use **Motion Capture** data.

In this work, we use existing motion capture data to drive the animation system based on a motion graph (MG) [67] [48]. It could be applied onto gesture space to generate virtual character's conversational full body animations as well.

A motion graph is an animation **concatenation** system which needs to do fast data searching, re-ordering and re-sequencing existing motion clips to build new animation sequences. The two main objectives of a motion graph are:

- **Completeness**: How to create and design a data set of motions with various behaviours,
- **Efficiency**: How to achieve correct and fast search results from a data set,
- **Flexibility**: How to give users an easy way to control and use the data set.

Graph Searching Random walks on the motion graph are not interesting for most cases. So searching suitable motion frames that satisfy some objectives is an

essential process. The basic solution is to define an error function such as:

$$f(w) = f([e_1, ...e_n]) = \sum_{i=1}^{n} g([e_1, ...e_n], E_i) \tag{10.0.1}$$

where $f(w)$ is the error function that measures the total path error on the graph, $g(w,E)$ is a scalar function that counts the error generated by each graph walk step compared to the expected value E. In this case, the motion generation becomes a process of solving an error function minimization. The efficiency of the search process is important when motion sequences need to be generated online. Interesting options are speed improvements regarding the error function, the simplification of the search algorithm or the increase of the capacity during the search process. We would like to explore parallel computing ideas for modern animation systems with large data support. High performance searching for large data structures will be a big challenge.

Further Step: Expressive Gesture Space Modeling Motion Graphs have been applied to the locomotion for a long time. Performing the MG in gesture space is a new topic in the virtual character domain. We have developed a procedural animation pipeline for virtual agents, but without using motion data, the animation lacks naturalness. Several other methods have also been proposed to use motion capture data, such as Inverse Blending [55], or a style based data learning method [20] [42]. These can generate high quality postures and stylize their motion sequence in between key frames. All these methods can be used for virtual character expressive gesture animations. What we want to do is to combine both procedural methods and motion capture data methods. The motion sequence will be generated by motion data which can be more realistic and can have better transitions, the procedural method is used to correct the targeting process. Multiple dimensional blending technique can be applied on multiple closest graph paths and procedural reaching model on animation frame level. The gain will be both the naturalness and the flexibility. Style factors can also be set up by using data mining methods. Emotional factor based on real data analysis can offer better supports on the generation of expressive gestures. This domain can be far better explored in the future.

Part IV

Appendix

Chapter 11

Examples and Pseudo Code

All the demos and examples can be found in my website:
http://perso.telecom-paristech.fr/~jhuang/
jing.huang@telecom-paristech.fr

11.1 Separable Approximation of SSAO

Listing 11.1: Shader code of SAO.

```
uniform sampler2D normalIDMap; //normal + objectID
uniform sampler2D mvPosMap; //position in modelview space
uniform mat4 In_ProjectionMatrix;
uniform float radius;
uniform float bias;
uniform float darkness;
uniform int stepCount;
uniform sampler2D SamplesTexture;
uniform int SamplesNumber; //SamplesNumber
uniform int PatternSize;  //PatternSize
mat4 othogonalMarix;
out vec4 FragColor[1];

float marchOrthgonalRay(vec4 currentPosition,
   vec4 currentNormalVector, vec2 direction,
   int stepCount, float radius, float bias,
   sampler2D mvPositionMap,
   mat4 In_ProjectionMatrix)
{
   float aoh = marchRay(currentPosition, currentNormalVector,
direction, stepCount, radius, bias, mvPositionMap,
```

```
In_ProjectionMatrix);
   float aov = marchRay(currentPosition, currentNormalVector,
direction * othogonalMarix, stepCount, radius, bias,
mvPositionMap, In_ProjectionMatrix);
   return (aoh + aov) * 0.5;
}

void main (void) {
    vec4 n = texture (normalIDMap, gl_TexCoord[0].st);
    if (n.w < 0.5) discard;
    vec4 p = texture (mvPosMap, gl_TexCoord[0].st);
    int patternIndex = (int(gl_FragCoord.x) % int(PatternSize))
    + int(PatternSize) * (int (gl_FragCoord.y)
    % int (PatternSize)); //interleaved sampling
    vec2 dir = getSample(patternIndex, SamplesTexture).xy;
    float ao = marchOrthgonalRay(p, n, dir, stepCount,
radius, bias, mvPosMap, In_ProjectionMatrix);
    FragColor[0] = vec4 (1 - ao * darkness);
}
```

11.2 Mass-Spring based Inverse Kinematics

Listing 11.2: code of Mass-Spring IK

```
void computerIK(vector<Joint> joints, Target target, int max_Iter, float lengthTreshold,
 bool enableConstraint)
{
    Target target_new = checkForRetargetting(joints, target);
    int size = joints.size() - 1;
    Vec3f endEffector = joints[size]->getWorldPosition();
    int itor = 0;
    float length = Vec3f(endEffector - target_new.getPosition()).length();
    while (itor < max_Iter && length > lengthTreshold)
    {   //using mass-spring to solve positions
        moveMassSpringSystem(joints, target_new.getPosition());
        for(unsigned int i = 1; i < joints.size(); i++ )
        {
            calculateLocalRotation(joints[i]); //update orientations
        }
        if(enableConstraints)
        {
            checkJointsDOFsRestrictions(joints); //check fo orientation constraints
        }
        endEffector = joints[size]->getWorldPosition();
        length = Vec3f(endEffector - target_new.getPosition()).length();
    }
}
```

Chapter 12

Other Rendering Implementations

12.1 Other Effects in Screen Space

Depth of Field is a physics phenomena of cameras due to the fact that lenses can only focus on one distance at a time. On each side of the focus distance, the sharpness of the image will be decreased. A point that should project to a point in the image will become an area on the screen. This area is called circle of confusion (CoC). In object space, the CoC changes linearly regarding the distance to the lens, first shrinking when it approaches the front focal plane, and then reaching zero (perfect focus) within the focus distance, and then increasing again linearly with distance from that distance interval (see Figure 12.2).

Several methods [43] [77] have already been proposed to generate high quality focus and defocus effects. They achieved extremely high-quality images by using an accumulation frame buffer, but it is hard to provide high speed performance. In real-time applications, the gather method is a good choice for achieving high frame rates. We implemented the simple idea that follows the focus-defocus CoC rule:

$$R(p_i) = \frac{|z(p_i) - z_f|}{z(p_i)} \times R_{CoC} \tag{12.1.1}$$

where R is the radius of the region for the gathering process, $z(p_i)$ is the depth value, z_f is the focus plane which could also be an interval, R_{CoC} is the radius of CoC area. We show our result in Figure 12.2.

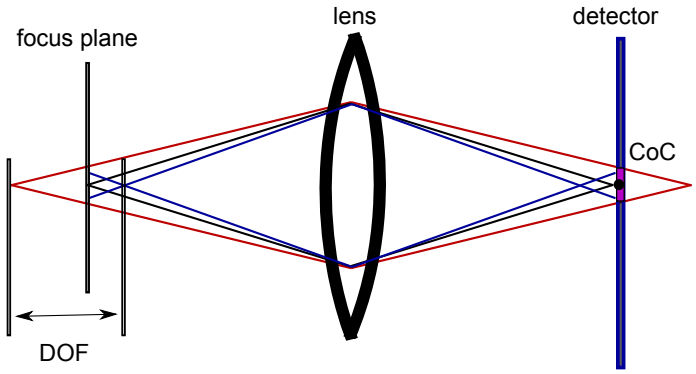

Figure 12.1: CoC region changes due to the depth of field effect.

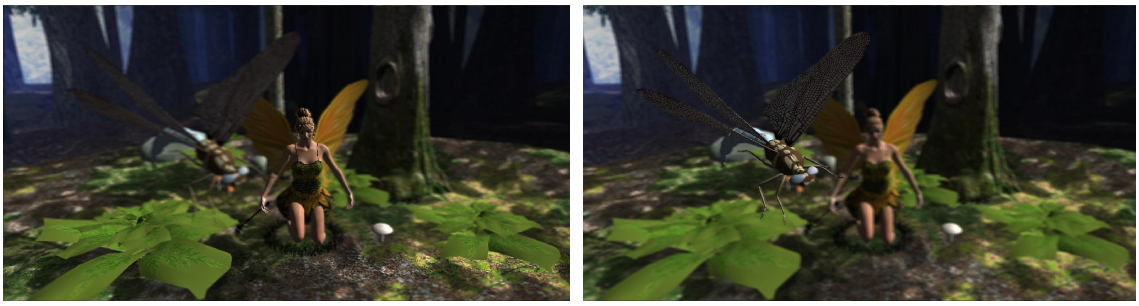

Figure 12.2: Our results of depth of field implementation: **Left**: focus on the fairy; **Right**: focus on the dragonfly. The mesh «Fairy Forest» is from «The Utah 3D Animation Repository».

Shadow Mapping concept was introduced by Lance Williams [143] in 1978. It is used to generate high performance shadow effects in 3D graphics. In our renderer, we choose a soft shadowing method called percentage-closer soft shadows [37]. This method computes the casting shadow by checking a set of neighbouring pixels with bi-linear filtering, doing depth comparisons for each of them, and then applying a box filtering for the neighbor set. To simulate penumbras, we utilize the light's size and the blocker/receiver distance from the light source:

$$w_{Penumbra} = (d_{Receiver} - d_{Blocker}) \cdot w_{Light} / d_{Blocker} \tag{12.1.2}$$

The result can be seen in Figure 12.3.

We also interpret colored transparent shadow mapping [86]. We apply screen-door masks [95] on the transparent objects both during the shadow pass and the

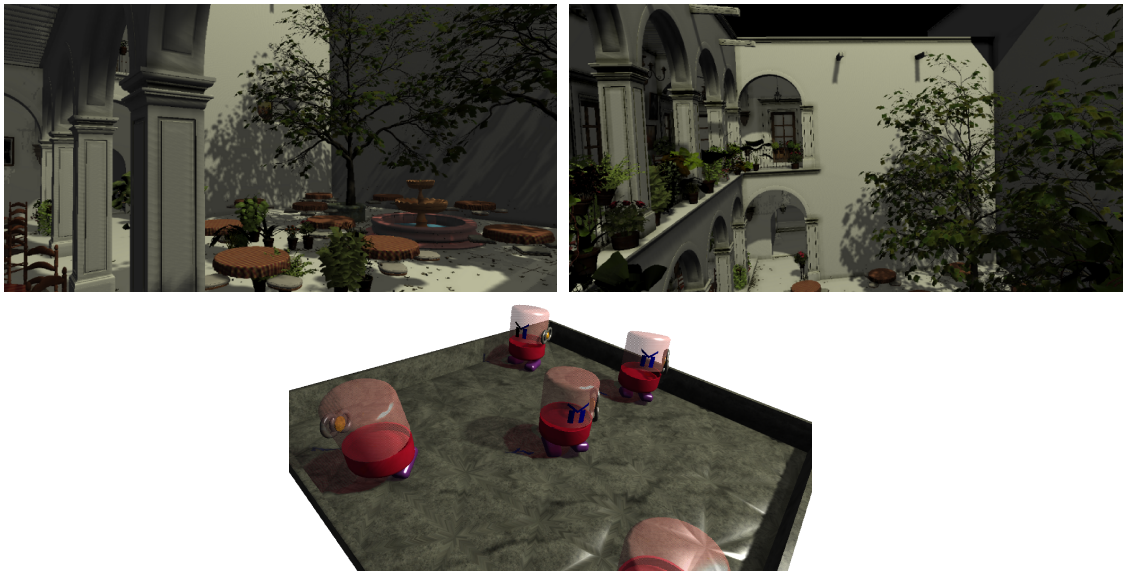

Figure 12.3: Above: Shadows casted results in San Miguel Scene from [85]. Below: The colored transparent Shadow Map result in the Toasters Scene.

(main) scene rendering pass. This process is used to gather the colors from different layers in the light transfer path, and therefore can achieve the transparency simulation. The result can be seen in Figure 12.3.

Chapter 13

Animation and Deformation Computations

Skeletal System

We explain more in this part about the skeletal animation which is used in our animation system. The character animation can be seen as a rigid body deformation. Using skeletal systems to animate virtual characters is also very efficient. Thus, the skeleton method is widely used in real-time character animation engine.

A character in a skeletal system is represented in two parts: a surface representation which is used to draw the character (called skin or mesh) and a hierarchical set of interconnected bones (called the skeleton or rig) used to animate (pose and key frame) the mesh, the components in between the connected bones are the joints. The skeleton is a hierarchy that saves a set of joints and their dependencies. A basic joint has one optional parent and several optional children. It has a three dimensional local transformation including its position and orientation. Rotation transformations are often used for all joints. For the translation information, only the root joint uses them to manipulate the global skeleton position.

The simple computation of the hierarchy can be defined as:

$$Rg_{J_{n+1}} = Rg_{J_n} * Rl_{J_{n+1}} \tag{13.0.1}$$

$$Pg_{J_{n+1}} = Rg_{J_{n+1}} * Pl_{J_{n+1}} + Pg_{J_n} \tag{13.0.2}$$

where R is a rotation matrix, Rg_{J_n} is a global rotation of joint n, Rl_{J_n} is local

rotation in the local coordinate system of joint n, Pg_{J_n} is global position of joint n, and Pl_{J_n} is local position in the local coordinate system of joint n.

Euler angles and quaternions

A rotation in the 3 dimensional Euclidean space is usually described as its axis of rotation $u = (u_x, u_y, u_z)(||u|| = 1)$ and its angle of the rotation θ. It can be represented by a 3×3 orthogonal matrix R ($R^t \cdot R = R \cdot R^t = I_3$), and the rotation p' of a vector $p \in \mathbb{R}^3$ is obtained by applying it to the rotation matrix: $p' = R \cdot p$.

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{13.0.3}$$

It can be represented using Euler angles [122] [124], by decomposing it as the product of 3 rotation matrices R_x, R_y, R_z, that have their axis of rotation that is aligned onto the Euclidean basis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{13.0.4}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{13.0.5}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13.0.6}$$

The original matrix R can be recovered as the product of this decomposition:

$$R = R_z(\psi)R_y(\theta)R_x(\phi) =$$

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad (13.0.7)$$

This decomposition corresponds to a counterclockwise/right-handed rotation with Euler angles ϕ, θ, ψ, with x-y-z convention.

A useful representation of rotations in $\mathbb{R}^3$ is the quaternions [45]. A quaternion is writen as the form: $w+xi+yj+zk$, where w is **real** part also called **scalar** part, (x,y,z) is the **imaginary** part or **vector** part.

Two quaternions $\mathbf{q} = w+xi+yj+zk$ and $\mathbf{q}' = w'+x'i+y'j+z'k$ can be multiplied by using the following rules: (a) $i*i = j*j = k*k = i*j*k = -1$, (b) $i*j = k$,$j*i = -k$, $j*k = i$, $k*j = -i$, $k*i = j$, $i*k = -j$.

It is easy to see that the multiplication of quaternions is not a commutative operation. The conjugate quaternion $\mathbf{q}^* = w-xi-yj-zk$ is the inverse of $\mathbf{q} = w+xi+yj+zk$ for the multiplication.

The space of quaternions can be identified to the four dimensional Euclidean space $\mathbb{R}^4$, and the set of **unit quaternions** can be identified to the set of rotations in $\mathbb{R}^3$:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \mathbf{q}_{\mathbf{u},\theta} = \begin{bmatrix} \sin(\frac{\theta}{2})*u_x \div (\sqrt{u_x^2+u_y^2+u_z^2}) \\ \sin(\frac{\theta}{2})*u_y \div (\sqrt{u_x^2+u_y^2+u_z^2}) \\ \sin(\frac{\theta}{2})*u_z \div (\sqrt{u_x^2+u_y^2+u_z^2}) \\ \cos(\frac{\theta}{2}) \end{bmatrix} \quad (13.0.8)$$

Given a three dimensional vector (a,b,c), it can be shown that the three dimensional vector (a',b',c') given by $(a',b',c',0) = \mathbf{q}_{\mathbf{u},\theta} * (a,b,c,0) * \mathbf{q}_{\mathbf{u},\theta}{}^*$ is indeed the rotation of (a,b,c) around the axis u for an angle of θ.

The Euler angles system is easier to understand, but the interpolation between two vectors u and v performed by interpolating linearly their Euler angles is depen-

dent to the convention that is chosen ($xyz, zyx, ...$), and yields paths on the Gauss sphere that do not correspond to geodesics. In contrast, quaternions offer this last feature, and they can perform linear transformation on rotations with constant velocity. The interpolation between two quaternions $\mathbf{q_0}$ and $\mathbf{q_1}$ is called **great arc interpolation** or **spherical linear interpolation** (Slerp):

$$\text{Slerp}(\mathbf{q_0}, \mathbf{q_1}, t) = \frac{\sin[(1-t)\Omega]}{\sin\Omega}\mathbf{q_0} + \frac{\sin[t\Omega]}{\sin\Omega}\mathbf{q_1} \tag{13.0.9}$$

where t is the interpolation parameter (between 0 and 1), $\mathbf{q_0}$ and $\mathbf{q_1}$ represent two rotations, Ω is the half angle between them.

Conversions between quaternions and Euler angles can be obtained by:

$$\mathbf{q} = R_z(\psi)R_y(\theta)R_x(\phi) \tag{13.0.10}$$

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \end{bmatrix} \tag{13.0.11}$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\frac{2(wx+yz)}{1-2(x^2+y^2)} \\ \arcsin(2(wy-zx)) \\ \arctan\frac{2(wz+xy)}{1-2(y^2+z^2)} \end{bmatrix} \tag{13.0.12}$$

All these formulas are built into our animation library and are used to update, constrain the rotations of joints in skeletal animations and also to interpolate key-frame-rotations for final animation generation.

Chapter 14

Publications

- Jing Huang, Tamy Boubekeur, Tobias Ritschel, Matthias Holländer, and Elmar Eisemann. Separable Approximation of Ambient Occlusion. In *Proceedings of Eurographics 2012 short*, pages 29–32, 2011
- Jing Huang and Catherine Pelachaud. Expressive body animation pipeline for virtual agent. In Yukiko Nakano, Michael Neff, Ana Paiva, and Marilyn Walker, editors, *Intelligent Virtual Agents, 12th International Conference on Intelligent Virtual Agents*, volume 7502 of *Lecture Notes in Computer Science*, pages 355–362. Springer Berlin Heidelberg, 2012
- Jing Huang and Catherine Pelachaud. An efficient energy transfer inverse kinematics solution. In *Proceedings of Motion In Game 2012*, volume 7660, pages 278–289, Berlin, Heidelberg, 2012. LNCS
- Radoslaw Niewiadomski, Jing Huang, and Catherine Pelachaud. Effect of facial cues on identification. In *The 25th Annual Conference on Computer Animation and Social Agents (CASA 2012)*, page 4, Singapore, 2012
- Quoc Anh Le, Jing Huang, and Catherine Pelachaud. A common gesture and speech production framework for virtual and physical agents. In *Workshop of the 14th ACM International Conference on Multimodal Interaction*, 2012
- Magalie Ochs, Elisabetta Bevacqua, Ken Prepin, Quoc Anh Le, Yu Ding, Jing Huang, Radoslaw Niewiadomski, and Catherine Pelachaud. La compréhension machine à travers l'expression non-verbale. In *Intercompréhension - de l'intraspécifique à l'interspécifique*, 2011

Bibliography

[1] An Ambient Light Illumination Model. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, Eurographics, pages 45–56. Springer-Verlag Wien New York, 1998. 18

[2] Andrew Adams, Natasha Gelfand, Jennifer Dolson, and Marc Levoy. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph.*, 28(3): 21:1–21:12, July 2009. ISSN 0730-0301. 4

[3] Irene Albrecht, Marc Schröder, Jörg Haber, and Hans-Peter Seidel. Mixed feelings: expression of non-basic emotions in a muscle-based talking head. *Virtual Real.*, 8(4):201–212, August 2005. ISSN 1359-4338. xii, 4

[4] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM. 11

[5] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 402–408, New York, NY, USA, 2003. ACM. 52

[6] Andreas Aristidou and Joan Lasenby. FABRIK: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011. ISSN 1524-0703. 58, 64, 66, 69

[7] Paolo Baerlocher and Ronan Boulic. Parametrization and range of motion of the ball-and-socket joint, 2000. 66

[8] Paolo Baerlocher and Ronan Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20:402–417, 2004. 65, 75, 84

[9] Christoph Bartneck and Juliane Reichenbach. Subtle emotional expressions of synthetic characters. *Int. J. Hum.-Comput. Stud.*, 62(2):179–192, February 2005. ISSN 1071-5819. 30

[10] Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks*, pages 22:1–22:1, 2008. xvi, xviii, 20, 24, 27

[11] Daniel Bernhardt and Peter Robinson. Detecting affect from non-stylised body motions. In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 59–70, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-74888-5. 86

[12] E Bevacqua, K Prepin, R Niewiadomski, E de Sevin, and C Pelachaud. Greta: Towards an interactive conversational virtual companion. *Artificial Companions in Society: Perspectives on the Present and Future*, page 315, 2009. xii, 5

[13] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977. ISSN 0097-8930. 14

[14] James F. Blinn. Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph.*, 12(3):286–292, August 1978. ISSN 0097-8930. xix, 32, 34

[15] Jonathan Blow. Inverse kinematics with quaternion joint limits. *Game Developer*, 2002. 66

[16] W. Jack Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Commun. ACM*, 13(9):527–536, September 1970. ISSN 0001-0782. 11

[17] R Boulic, J Varona, L Unzueta, M Peinado, Angel Suescun, and F Perales. Evaluation of on-line analytic and numeric inverse kinematics approaches driven by partial vision input. *Virtual Reality*, 10(1):48–61, 2006. 58

[18] Ronan Boulic and Daniel Thalmann. Combined direct and inverse kinematic control for articulated figure motion editing, 1992. 75

[19] A. Bounard, S. Gibet, and M. M. Wanderley. Hybrid inverse motion control for virtual characters interacting with sound synthesis: Application to percussion motion. *Vis. Comput.*, 28(4):357–370, April 2012. ISSN 0178-2789. 58

[20] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 183–192, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 54, 98

[21] Stefan Bruckner and Eduard Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Trans. on Vis. and Comp. Graph.*, 13:1344–1351, November 2007. ISSN 1077-2626. 18

[22] Heinrich H. Bulthoff and Michael S. Langer. Perception of shape from shading on a cloudy day, 1999. 18

[23] Host Bunke and Terry Caelli, editors. *Hidden Markov models: applications in computer vision*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002. 58

[24] M. Bunnell. Dynamic ambient occlusion and indirect lighting. *GPU Gems*, 2: 223–233, 2005. 19

[25] Samuel R Buss and Jin-su Kim. Selectively damped least squares for inverse kinematics. *Methods*, 10(3):1–13, 2004. xxx, 56, 57, 63, 69

[26] Adrian A Canutescu and Roland L Dunbrack. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Science*, 6(5):963–972, 2003. 57

[27] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. 4

[28] Diane Chi, Monica Costa, Liwei Zhao, and Norman Badler. The EMOTE model for effort and shape. In *In Proceedings of SIGGRAPH 2000*, SIGGRAPH '00, pages 173–182, New York, NY, USA, 2000. xxxiii, 74, 82

[29] Kwan Wu Chin, Brian R Von Konsky, and Andrew Marriott. Closed-form and generalized inverse kinematics solutions for the analysis of human motion. *IEEE Engineering in Medicine and Biology Society*, 19, 1997. xxvi, 57, 60

[30] Matthieu Courgeon, Stéphanie Buisine, and Jean-Claude Martin. Impact of expressive wrinkles on perception of a virtual character's facial expressions of emotions. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 201–214, Berlin, Heidelberg, 2009. Springer-Verlag. xix, xx, 30, 32, 35

[31] Nicolas Courty and Élise Arnaud. Inverse kinematics using sequential Monte Carlo methods. In *5th International Conference on Articulated Motion and Deformable Object, AMDO 2008, July, 2008*, Lecture Notes in Computer Science, 2008. 58

[32] E. R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. 21

[33] Eugene d'Eon, David Luebke, and Eric Enderton. Efficient rendering of human skin. In *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, pages 147–158, June 2007. 42, 43

[34] Eugene d'Eon, David Luebke, and Eric Enderton. A system for efficient rendering of human skin. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. 4

[35] D.F.Shanno and P.C.Kettle. Optimal conditioning of quasi-newton methods. *Mathematics of computation*, 24(111), 1970. xxvi, 57, 60

[36] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph.*, 24(3):1032–1039, 2005. ISSN 0730-0301. 42

[37] Randima Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. 104

[38] James D. Foley, Richard L. Phillips, John F. Hughes, Andries van Dam, and Steven K. Feiner. *Introduction to Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994. ISBN 0201609215. 3

[39] John Q. Gan, Eimei Oyama, Eric M. Rosales, and Huosheng Hu. A complete analytical solution to the inverse kinematics of the pioneer 2 robotic arm. *Robotica*, 23:123–129, January 2005. ISSN 0263-5747. 56

[40] Joachim Georgii and Rüdiger Westermann. Mass-spring systems on the gpu. *Simulation Modelling Practice and Theory*, 13:693–702, 2005. 64

[41] H. Gouraud. Continuous shading of curved surfaces. *IEEE Trans. Comput.*, 20(6):623–629, June 1971. ISSN 0018-9340. 12

[42] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, August 2004. ISSN 0730-0301. 98

[43] Paul Haeberli and Kurt Akeley. The accumulation buffer: hardware support for high-quality rendering. *SIGGRAPH Comput. Graph.*, 24(4):309–318, September 1990. ISSN 0097-8930. 103

[44] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the stormer/verlet method. *Acta Numerica*, 12:399–450, 2003. xxvi, 60

[45] William Rowan Hamilton. On quaternions, or on a new system of imaginaries in algebra. *Philosophical Magazine*, 25(3):489–495, 1844. 109

[46] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1993. ACM. 42

[47] Björn Hartmann, Maurizio Mancini, and Catherine Pelachaud. Implementing expressive gesture synthesis for embodied conversational agents. *Proceedings of the 6th international conference on Gesture in Human-Computer Interaction and Simulation*, pages 188–199, 2006. xxxiii, 74, 76, 81, 82

[48] Rachel Heck and Michael Gleicher. Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 129–136, New York, NY, USA, 2007. ACM. 53, 97

[49] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph.*, 27(3):27:1–27:11, August 2008. ISSN 0730-0301. 58, 75

[50] Alexis Heloir and Michael Kipp. EMBR - a realtime animation engine for interactive embodied agents. In *IVA*, pages 393–404, 2009. 75

[51] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripourt. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, 1987. xxviii, 63

[52] Jing Huang and Catherine Pelachaud. An efficient energy transfer inverse kinematics solution. In *Proceedings of Motion In Game 2012*, volume 7660, pages 278–289, Berlin, Heidelberg, 2012. LNCS. 55

[53] Jing Huang and Catherine Pelachaud. Expressive body animation pipeline for virtual agent. In Yukiko Nakano, Michael Neff, Ana Paiva, and Marilyn Walker, editors, *Intelligent Virtual Agents, 12th International Conference on*

Intelligent Virtual Agents, volume 7502 of *Lecture Notes in Computer Science*, pages 355–362. Springer Berlin Heidelberg, 2012. 70, 71, 74

[54] Jing Huang, Tamy Boubekeur, Tobias Ritschel, Matthias Holländer, and Elmar Eisemann. Separable Approximation of Ambient Occlusion. In *Proceedings of Eurographics 2012 short*, pages 29–32, 2011. 12

[55] Yazhou Huang and Marcelo Kallmann. Motion parameterization with inverse blending. In *Proceedings of the Third International Conference on Motion In Games*, Berlin, 2010. Springer. 98

[56] Lander Jeff. *Making Kine more flexible*, volume 5. Darwin 3D, 1998. xxv, 59, 70

[57] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 511–518, New York, NY, USA, 2001. ACM. 4

[58] Jorge Jimenez, Veronica Sundstedt, and Diego Gutierrez. Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept.*, 6(4):1–15, 2009. ISSN 1544-3558. 43

[59] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20 (4):143–150, August 1986. ISSN 0097-8930. 12

[60] Jari Kätsyri and Mikko Sams. The effect of dynamics on identifying basic emotions from synthetic and natural faces. *Int. J. Hum.-Comput. Stud.*, 66 (4):233–242, April 2008. ISSN 1071-5819. 30

[61] Alexander Keller and Wolfgang Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276, London, UK, UK, 2001. Springer-Verlag. xvi, 24

[62] Dacher Keltner. Signs of appeasement: Evidence for the distinct displays of embarrassment, amusement, and shame, 1995. 30

[63] Andrea Kleinsmith and Nadia Bianchi-Berthouze. Recognizing affective dimensions from body posture. In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 48–58, Berlin, Heidelberg, 2007. Springer-Verlag. 75

[64] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH*, January 1984. ISSN 0097-8930. xxxiii, 82

[65] Janne Kontkanen and Samuli Laine. Ambient occlusion fields. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 41–48, New York, NY, USA, 2005. ACM. 19

[66] James Urey Korein. *A geometric investigation of reach*. MIT Press, Cambridge, MA, USA, 1985. 66

[67] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, July 2002. ISSN 0730-0301. 52, 97

[68] Brigitte Krenn, Stacy Marsella, Andrew N. Marshall, Hannes Pirker, Kristinn R. Thĺőrisson, and Hannes Vilhjĺćlmsson. Towards a common framework for multimodal generation in ecas: The behavior markup language. In *In Proceedings of the 6th International Conference on Intelligent Virtual Agents, Marina*, pages 21–23, 2006. xxxi, 2, 75, 76

[69] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 1999. 65, 66

[70] R Kulpa, F Multon, and B Arnaldi. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum*, 24(3):343–351, 2005. 58

[71] Jean-Henri Lambert. Jean-henri lambert, photométrie ou de la mesure et de la gradation de la lumière, des couleurs et de l'ombre (1760). trad. du latin par j. boye, j. couty, m. saillard, introd. et notes de m. saillard (paris : L'harmattan, 1997). *Revue d'histoire des sciences*, 54(2):267–268, 2001. 13

[72] H Landis. Production-ready global illumination, 2002. xiv, 18

[73] M. S. Langer and S. W. Zucker. Shape-from-shading on a cloudy day. *J. Opt. Soc. Am. A*, 11(2):467–478, Feb 1994. 18

[74] C. Larboulette and M.-P. Cani. Real-time dynamic wrinkles. In *Computer Graphics International, 2004. Proceedings*, pages 522 –525, june 2004. xix, 32

[75] Quoc Anh Le, Jing Huang, and Catherine Pelachaud. A common gesture and speech production framework for virtual and physical agents. In *Workshop of the 14th ACM International Conference on Multimodal Interaction*, 2012.

[76] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.*, 21(3):491–500, July 2002. ISSN 0730-0301. 52, 53

[77] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. Depth-of-field rendering with multiview synthesis. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia)*, 28(5), 2009. 103

[78] Walter Lewin. Hook's law, simple harmonic oscillator. mit course: Classical mechanics, lecture, 1999. xxv, 59

[79] Bradford James Loos and Peter-Pike Sloan. Volumetric obscurance. In *Proc. ACM I3D '10*, pages 151–156, 2010. xviii, 20, 27

[80] Miguel Sainz Louis Bavoil. Nvidia tech report: Screen-space ambient occlusion. Technical report, NVIDIA, 2008. xii, 3, 4

[81] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. In *ACM SIGGRAPH*, pages 1206–1213, 2006. 18

[82] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, EGSR'07, pages 183–194, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 45

[83] Mattias Malmer, Fredrik Malmer, Ulf Assarsson, and Nicolas Holzschuch. Fast precomputed ambient occlusion for proximity shadows. *Journal of Graphics Tools*, 12(2):59–71, 2007. 19

[84] M. McGuire. Ambient occlusion volumes. In *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 47–56, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. 19

[85] Morgan McGuire. Computer graphics archive, August 2011. `http://graphics.cs.williams.edu/data`. 105

[86] Morgan McGuire and Eric Enderton. Colored stochastic shadow maps. *ACM Symposium on Interactive 3D Graphics and Games*, February 2011. 104

[87] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 77–89, New York, NY, USA, 2009. ACM. xii, 3

[88] McNeill. *Hand and Mind: WHAT GESTURES REVEAL ABOUT THOUGHT*. The University of Chicago press, Chicago, 1992. xxxiii, 81

[89] Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and Hans-Peter Seidel. Efficient rendering of local subsurface scattering. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 51–, Washington, DC, USA, 2003. IEEE Computer Society. 4

[90] Morten S. Mikkelsen. Bump mapping unparametrized surfaces on the gpu. *journal of graphics, gpu, and game tools*, 15(1):49–61, 2010. 34

[91] Gavin Miller. Efficient algorithms for local and global accessibility shading. In *SIGGRAPH*, pages 319–326, 1994. 18

[92] Martin Mittring. Finding next gen: CryEngine 2. In *ACM SIGGRAPH '07 Courses*, pages 97–121, 2007. xiv, xviii, 18, 19, 21, 27

[93] Mark Mizuguchi, John Buchanan, and Tom Calvert. Data driven motion transitions for interactive games. In *Eurographics 2001 ShortPapers*, Eurographics 2001, 2001. 53

[94] Uldarico Muico, Jovan Popović, and Zoran Popović. Composite control of physically simulated characters. *ACM Trans. Graph.*, 30(3):16:1–16:11, May 2011. ISSN 0730-0301. 54

[95] Jurriaan D. Mulder, Frans C. A. Groen, and Jarke J. van Wijk. Pixel masks for screen-door transparency. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 351–358, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. 104

[96] Muller-Cajar and R.Mukundan. Triangualation - a new algorithm for inverse kinematics. *Proceedings of Image and Vision Computing 2007*, pages 181–186, 2007. 57

[97] Michael Neff and Eugene Fiume. Modeling tension and relaxation for computer animation. In *Proceedings of Symposium on Computer Animation 2002*, SCA '02, pages 81–88, New York, NY, USA, 2002. ACM. 75

[98] Michael Neff and Eugene Fiume. Artistically based computer generation of expressive motion. In *In Proceedings of the Adaptation in Artificial and Biological Systems*, pages 29–39, 2004. 75

[99] Michael Neff and Eugene Fiume. AER: aesthetic exploration and refinement for expressive character animation. In *Proceedings of Symposium on Computer Animation 2005*, SCA '05, pages 161–170, New York, NY, USA, 2005. ACM. 75

[100] R. Niewiadomski, S.J. Hyniewska, and C. Pelachaud. Constraint-based model for synthesis of multimodal sequential expressions of emotions. *Affective Computing, IEEE Transactions on*, 2(3):134 –146, july-sept. 2011. 30

[101] Radoslaw Niewiadomski, Sylwia Hyniewska, and Catherine Pelachaud. Modeling emotional expressions as sequences of behaviors. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 316–322, Berlin, Heidelberg, 2009. Springer-Verlag. xii, 5

[102] Radoslaw Niewiadomski, Elisabetta Bevacqua, Quoc Anh Le, and C. Pelachaud. Cross-media agent platform. In *Web-3D*, Web3D '11, pages 11–19, 2011. 74

[103] Radoslaw Niewiadomski, Jing Huang, and Catherine Pelachaud. Effect of facial cues on identification. In *The 25th Annual Conference on Computer Animation and Social Agents (CASA 2012)*, page 4, Singapore, 2012.

[104] S. Noel, S. Dumoulin, T. Whalen, and J. Stewart. Recognizing emotions on static and animated avatar faces. In *Haptic Audio Visual Environments and their Applications, 2006. HAVE 2006. IEEE International Workshop on*, pages 99 –104, 2006. 30

[105] Magalie Ochs, Elisabetta Bevacqua, Ken Prepin, Quoc Anh Le, Yu Ding, Jing Huang, Radoslaw Niewiadomski, and Catherine Pelachaud. La compréhension machine à travers l'expression non-verbale. In *Intercompréhension - de l'intraspécifique à l'interspécifique*, 2011.

[106] W. Friesen P. Ekman. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, 1978. xii, xix, 4, 30, 31

[107] Marco Paleari and Christine Lisetti. Psychologically Grounded Avatars Expressions. In *29th Annual Conference on Artificial Intelligence*, June 2006. xii, 4

[108] Xueni Pan, Marco Gillies, Tevfik Metin Sezgin, and Celine Loscos. Expressing complex mental states through facial expressions. In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 745–746, Berlin, Heidelberg, 2007. Springer-Verlag. xii, 4

[109] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. A gentle introduction to bilateral filtering and its applications. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 1:1–1:50, New York, NY, USA, 2008. ACM. 4

[110] Tuan Q. Pham and Lucas J. Vliet. Separable bilateral filtering for fast video preprocessing. In *In IEEE Internat. Conf. on Multimedia and Expo, CD1ÍC4*, pages 1–4. IEEE, 2005. xv, 23

[111] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010. 24

[112] Alistair D.N. Edwards Philip A. Harling. Hand tension as a gesture segmentation cue. In *In Proceedings of the Progress in Gestural Interaction*, pages 75–88. MIT mimeo, 1997. xxxiii, 82

[113] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18 (6):311–317, June 1975. ISSN 0001-0782. 12, 13

[114] C. Reinbothe, T. Boubekeur, and M. Alexa. Hybrid ambient occlusion. *EUROGRAPHICS '09 Areas Papers*, 2009. xvi, 4, 19, 24

[115] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proc. ACM I3D '09*, pages 75–82, 2009. xii, 3, 4, 20

[116] Rodney G. Roberts and Anthony A. Maciejewski. Singularities, stable surfaces, and the repeatable behavior of kinematically redundant manipulators. *International Journal of Robotics Research*, 13:70–81, 1994. xxx, 56, 57, 69

[117] Marc Ruiz, Lázló Szirmay-Kalos, Tamás Umenhoffer, Imma Boada, Miquel Feixas, and Mateu Sbert. Volumetric ambient occlusion for volumetric models. *Vis. Comput.*, 26:687–695, June 2010. 20

[118] Zsofia Ruttkay, Han Noot Paul ten Hagen, Han Noot, and Paul Hagen. Emotion disc and emotion squares: tools to explore the facial expression space. *Computer Graphics Forum*, 22:49–53, 2003. xii, 5

[119] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, September 1990. ISSN 0097-8930. 15

[120] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 73–80, New York, NY, USA, 2007. ACM. 20

[121] Ari Shapiro and Petros Faloutsos. Interactive and reactive dynamic control. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. 53

[122] Ken Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, July 1985. ISSN 0097-8930. 108

[123] J. C. Simo and T. J. R Hughes. *Computational Inelasticity*. Springer, 1998. xxv, 59

[124] Gregory G. Slabaugh. Computing euler angles from a rotation matrix, 1986. 108

[125] Nicolas Stoiber, Renaud Seguier, and Gaspard Breton. Automatic design of a control interface for a synthetic face. In *Proceedings of the 14th international conference on Intelligent user interfaces*, IUI '09, pages 207–216, New York, NY, USA, 2009. ACM. xii, 4

[126] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, March 1974. ISSN 0360-0300. 11

[127] Ning Tan, Céline Clavel, Matthieu Courgeon, and Jean-Claude Martin. Postural expressions of action tendencies. In *Proceedings of the 2nd international workshop on Social signal processing*, New York, NY, USA, 2010. ACM. 75

[128] Wen Tang, Marc Cavazza, Dale Mountain, and Rae A. Earnshaw. Real-time inverse kinematics through constrained dynamics. In *Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, CAPTECH '98, pages 159–170, London, UK, UK, 1998. Springer-Verlag. 52, 56

[129] Marcus Thiebaux, Stacy Marsella, Andrew N. Marshall, and Marcelo Kallmann. Smartbody: behavior realization for embodied conversational agents. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*, AAMAS '08, pages 151–158, 2008. 75

[130] Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. Cager: Cage-based reverse engineering of animated 3d shapes. *Computer Graphics Forum*, 2012. 51

[131] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models Image Process.*, 2000. xxxiii, 82

[132] N. Tsapatsoulis, A. Raouzaiou, S. Kollias, R. Cowie, and E. Douglas-Cowie. *Emotion Recognition and Synthesis based on MPEG-4 FAPs*. in MPEG-4 Facial Animation, Igor Pandzic, R. Forchheimer (eds), John Wiley and Sons, UK, 2002., 2002. xii, 4

[133] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. Full-body performance animation with sequential inverse kinematics. *Graph. Models*, 70:87–104, September 2008. ISSN 1524-0703. 58

[134] Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, Jul 1967. xxvi, 56, 60

[135] C W Wampler, II. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Trans. Syst. Man Cybern.*, 16: 93–101, January 1986. ISSN 0018-9472. 56, 57

[136] Jing Wang and Bobby Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 232–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 53

[137] L C T Wang and C C Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7:489–499, 1991. xxx, 57, 69

[138] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. *Science*, C(April):86, 1993. 57

[139] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1): 389–396, December 1995. 21, 43

[140] Daniel E Whitney. Resolved motion rate control of manipulators and human prostheses. *Science*, MM(2):47–53, 1969. xxx, 56, 57, 69

[141] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980. ISSN 0001-0782. 11

[142] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *journal of graphics, gpu, and game tools*, 6(2):27–41, 2001. 65

[143] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, August 1978. ISSN 0097-8930. 104

[144] W.A Wolovich and H Elliott. *A computational technique for inverse kinematics*, volume 23, pages 1359–1363. IEEE, December 1984. xxx, 56, 57, 69

[145] Xiaomao Wu, Lizhuang Ma, Zhihua Chen, and Yan Gao. A 12-dof analytic inverse kinematics solver for human motion control. *Journal of Information Computational Science*, 1(1):137–141, 2004. 56

[146] Chris Wylie, Gordon Romney, David Evans, and Alan Erdahl. Half-tone perspective drawings by computer. In *Proceedings of the November 14-16, 1967, fall joint computer conference*, AFIPS '67 (Fall), pages 49–58, New York, NY, USA, 1967. ACM. 11

[147] Hector Yee. A perceptual metric for production testing. *Journal of Graphics Tools*, 9 (4), 2004. xviii, 26

[148] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: simple biped locomotion control. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. 54

[149] Ian T. Young and Lucas J. van Vliet. Recursive implementation of the gaussian filter, June 1995. ISSN 0165-1684. 22

[150] Jing Yuan. Local svd inverse of robot jacobians. *Robotica*, 19(1):79–86, January 2001. ISSN 0263-5747. 56, 57

[151] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.*, 13:313–336, October 1994. ISSN 0730-0301. 56, 65

[152] Victor Brian Zordan and Nicholas C. Van Der Horst. Mapping optical motion capture data to skeletal motion using a physical model. *ACM SIGGRAPH/Eurographics symposium on Computer animation 2003*, pages 245–250, 2003. 58

[153] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701, July 2005. ISSN 0730-0301. 58