International Journal of Computer Games Technology

Cyber Games and Interactive Entertainment

Guest Editors: Suiping Zhou, Zhongke Wu, and Ming-Quan Zhou

Cyber Games and Interactive Entertainment

International Journal of Computer Games Technology

Cyber Games and Interactive Entertainment

Guest Editors: Suiping Zhou, ZhongkeWu,
and Ming-Quan Zhou

Contents

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 713584, 2 pages
doi:10.1155/2009/713584

Editorial

Cyber Games and Interactive Entertainment

Suiping Zhou,[1] Zhongke Wu,[2] and Ming-Quan Zhou[2]

[1] *School of Computer Engineering, Nanyang Technological University, Singapore 639798*
[2] *College of Information Sience and Technology, Beijing Normal University, Beijing 100875, China*

Correspondence should be addressed to Suiping Zhou, asspzhou@ntu.edu.sg

Received 19 November 2009; Accepted 19 November 2009

Computer games and interactive entertainment have become a part of our life. The past decade has witnessed significant and fast advances in computer games technology. The boundary of 3D graphics in games has been pushed further with new and faster algorithms. Modeling of very large and complex 3D environments including buildings and terrains is a challenge. This involves efficient representations and data structures that help in the navigation and rendering of the environments in real time. Character behaviour modelling has recently gained the interest of researchers. Similar developments have been found in game physics to bring realistic behaviour to objects in a game environment. A wide range of character faces including talking heads have been deployed in games. These demand new methods to represent deformable faces. Game AI has grown rapidly with several new techniques in learning that have found applications in modern day games. Several new algorithms have also been developed recently for path planning and character behaviour in navigation. Rapid advances in technology and production skill are producing game engines that lead to the development of games content which are increasingly entertaining and impressive.

This special issue on Cyber Games and Interactive Entertainment focuses on the latest research and development work in games and interactive entertainment. This special issue presents some selected papers from the International Conference on Cyber Games 2008 (CG2008) held on 27–30 October 2008 at Beijing, China. Authors were invited to submit revised and extended version of their papers from the conference.

Jurgelionis et al. in "Platform for distributed 3D gaming," present a new cross-platform approach for distributed 3D gaming in wired/wireless local networks. The article introduces novel system architecture and protocols used to transfer the game graphics data across the network to end devices. Simultaneous execution of video games on a central server and a novel streaming approach of the 3D graphics output to multiple end devices enable the access of games on low-cost set-top boxes and handheld devices that natively lack the power of executing a game with high-quality graphical output. This is relevant for pervasive gaming in various environments like at home, hotels, or internet cafes; it is beneficial to run games also on mobile devices and modest performance CE devices avoiding the necessity of placing a noisy workstation in the living room or costly computers/consoles in each room of a hotel.

Researchers and developers in the field of computer games usually find that the difficulty to simulate the motion of actual 3D model trees lies in the fact that the tree model itself has very complicated structure, and many sophisticated factors need to be considered during the simulation. Though there are some works on simulating 3D tree and its motion, few of them are used in computer games due to the high demand for real-time in computer games. In the article on "Real time animation of trees based on BBSC in computer games," Ao et al. propose an approach of animating trees in computer games based on a novel tree model representation—Ball B-Spline Curves (BBSCs). By taking advantage of the good features of the BBSC-based model, physical simulation of the motion of leafless trees with wind blowing becomes easier and more efficient. The method can generate realistic 3D tree animation in real-time, which meets the high requirement for real time in computer games.

Mouse picking is the most commonly used intuitive operation to interact with 3D scenes in a variety of games as well as 3D graphics applications. High performance for such operation is necessary in order to provide users with

fast responses. The article on "Fast and reliable mouse picking using graphics hardware" by Zhao et al. proposes a fast and reliable mouse picking algorithm using graphics hardware for 3D triangular scenes. Their approach uses a multilayer rendering algorithm to perform the picking operation in linear time complexity. The objects-space-based ray-triangle intersection test is implemented in a highly parallelized geometry shader. After applying the hardware-supported occlusion queries, only a small number of objects (or subobjects) are rendered in subsequent layers, which accelerate the picking efficiency.

Hu et al. in their work on "A dense point-to-point alignment method for realistic 3D face morphing and animation" present a new point matching method to overcome the dense point-to-point alignment of scanned 3D faces. Instead of using the rigid spatial transformation in the traditional iterative closest point (ICP) algorithm, the authors adopt the thin plate spline (TPS) transformation to model the deformation of different 3D faces. Because TPS is a nonrigid transformation with good smooth property, it is suitable for formulating the complex variety of human facial morphology. A closest point searching algorithm is proposed to keep one-to-one mapping, and to get good efficiency the point matching method is accelerated by a KD-tree method. Having constructed the dense point-to-point correspondence of 3D faces, the authors create 3D face morphing and animation by key-frames interpolation and obtain realistic results.

In the article "Gamer's Facial Cloning for Online Interactive Games," Sattar et al. propose a solution to solve two bottlenecks in facial analysis and synthesis for an interactive system of human face cloning for nonexpert users of computer games. The problem arises during tactical maneuvers of the gamer, which makes single camera acquisition system unsuitable to analyze and track the face due to its large lateral movements. For an improved facial analysis system, the authors propose to acquire the facial images from multiple cameras and analyze them by multiobjective 2.5D active appearance model (MOAAM). To successfully clone or retarget the gamer facial expressions and gestures on to an avatar, the authors introduce a simple mathematical link between their appearances and present results to validate the efficiency, accuracy, and robustness of their approach.

Zhang and Prakash, in the article on "Face to face: anthropometry-based interactive face shape modeling using model priors," present a new anthropometrics-based method for generating realistic, controllable face models that can model faces specific to a population group or specific race. The method establishes an intuitive and efficient interface to facilitate procedures for interactive 3D face modeling and editing. It takes 3D face scans as examples in order to exploit the variations presented in the real faces of individuals. The system automatically learns a model prior from the datasets of example meshes of facial features using principal component analysis (PCA) and uses it to regulate the naturalness of synthesized faces. Solving the interpolation problem in a reduced subspace allows them to generate a natural face shape that satisfies the user-specified constraints. At runtime, the new face shape can be generated at an interactive rate.

Zhou et al. in the article "A new 3D model retrieval method with building blocks" propose a novel method of interactive 3D model retrieval with building blocks. First, by using a cube block as the base block in a 3D virtual space, the authors construct the query model with human-computer interaction method. Then through retrieving the polygon model of the database generated by the voxel model, the authors show how to get retrieval results in real time. As the numbers of 3D models available grow in many application fields, there is an increasing need for a search method to help people find them which are not effective where traditional search techniques are not always effective for 3D data.

In Multiplayer Ubiquitous Games (MUGs), players have to interact in the real world at both physical and virtual levels. Player profiles in MUGs offer an opportunity to provide personalized services to gamers. To provide an adaptable and personal content at any moment, anywhere, and in any context, Pellerin et al. in their article on "Player profile management on NFC smart card for multiplayer ubiquitous games," use player profiles in to provide personalized services to gamers. A Java API is used to integrate Smart Cards in the development of MUGs. This user centric approach brings new forms of gameplay, allowing the player to interact with the game or with other players anytime and anywhere. Smart Cards also help improve the security, ubiquity, and the user mobility in traditional MUGs.

Finally, we would like to thank all authors who have submitted their manuscripts to this special issue and the external reviewers for their invaluable contributions to the reviewing process. We would like to thank the Editor-in-Chief, Dr. Edmond Prakash, for giving us this great opportunity of organizing this special issue. We hope all researchers will enjoy and benefit from reading the articles in this IJCGT special issue on "Cyber games and interactive entertainment."

Suiping Zhou
Zhongke Wu
Ming-Quan Zhou

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 231863, 15 pages
doi:10.1155/2009/231863

Research Article

Platform for Distributed 3D Gaming

A. Jurgelionis,[1] P. Fechteler,[2] P. Eisert,[2] F. Bellotti,[1] H. David,[3] J. P. Laulajainen,[4] R. Carmichael,[5] V. Poulopoulos,[6,7] A. Laikari,[8] P. Perälä,[4] A. De Gloria,[1] and C. Bouras[6,7]

[1] *Department of Biophysical and Electronic Engineering, University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy*
[2] *Computer Vision & Graphics, Image Processing Department, Heinrich-Hertz-Institute Berlin, Fraunhofer-Institute for Telecommunications, 10587 Berlin, Germany*
[3] *R&D Department, Exent Technologies Ltd., 25 Bazel Street, P.O. Box 2645, Petach Tikva 49125, Israel*
[4] *Converging Networks Laboratory, VTT Technical Research Centre of Finland, 90571 Oulu, Finland*
[5] *Department of Psychology, Goldsmiths, University of London, New Cross, London SE14 6N, UK*
[6] *Research Unit 6, Research Academic Computer Technology Institute, N. Kazantzaki, Panepistimioupoli, 26504 Rion, Greece*
[7] *Computer Engineering and Informatics Department, University of Patras, 26500 Patras, Greece*
[8] *Software Architectures and Platforms Department, VTT Technical Research Centre of Finland, 02044 VTT, Espoo, Finland*

Correspondence should be addressed to A. Jurgelionis, jurge@elios.unige.it

Received 1 February 2009; Accepted 18 March 2009

Recommended by Suiping Zhou

Video games are typically executed on Windows platforms with DirectX API and require high performance CPUs and graphics hardware. For pervasive gaming in various environments like at home, hotels, or internet cafes, it is beneficial to run games also on mobile devices and modest performance CE devices avoiding the necessity of placing a noisy workstation in the living room or costly computers/consoles in each room of a hotel. This paper presents a new cross-platform approach for distributed 3D gaming in wired/wireless local networks. We introduce the novel system architecture and protocols used to transfer the game graphics data across the network to end devices. Simultaneous execution of video games on a central server and a novel streaming approach of the 3D graphics output to multiple end devices enable the access of games on low cost set top boxes and handheld devices that natively lack the power of executing a game with high-quality graphical output.

1. Introduction

Computer games constitute nowadays one of the most dynamic and fastest changing technological areas, both in terms of market evolution and technology development. Market interest is now revolving around capitalizing on the rapid increase of always-on broadband connectivity which is becoming ubiquitous. Broadband connection drives a new, digital "Future Home" as part of a communications revolution that will affect every aspect of consumers' lives, not least of which is the change it brings in terms of options for enjoying entertainment. Taking into account that movies and music provided by outside sources were at home long before the internet and broadband, the challenge is to invent new content consumption patterns of existing and new types of content and services [1].

At the same time, mobility and digital home entertainment appliances have generated the desire to play games not only in front of a home PC but also everywhere inside the house and also on the go. As a result of TV digitalization, set top boxes (STBs) have entered homes and, as a new trend, mini-laptops are gaining popularity. Several low-cost consumer electronics end devices (CE) are already available at home. Although these devices are capable of executing software, modern 3D computer games are too heavy for them.

Running an interactive content-rich multimedia applications (such as video games) requires the high performance hardware of a PC or a dedicated gaming device. Other devices such as set top boxes (STBs) or handheld devices lack the necessary hardware and adding such capabilities to these devices will cause their prices to become prohibitive [1].

A system which enables rendering of PC games on next-generation STB and personal digital assistant (PDA) devices without causing a significant increase in their price is a solution for future networked interactive media. This approach enables a pervasive accessibility of interactive media from devices that are running on different platforms (architecture and operating system), thus facilitating users to enjoy video games in various environments (home, hotel, internet café, elderly home) without the need to attach to a single device or operating system, for example, a Windows PC.

This paper describes the Games@Large (G@L) pervasive entertainment architecture which is built on the concept of distributed remote gaming [2] or Virtual Networked Gaming (VNG). It enables pervasive game access on devices (set top boxes and handheld devices) that typically do not possess a full set of technical requirements to run video games [1]. In general, the system executes games on a server PC, located at a central site or at home, captures the graphic commands, streams them to the end device, and renders the commands on the end device allowing the full game experience. For end devices that do not offer hardware accelerated graphics rendering, the game output is locally rendered at the server and streamed as video to the client. Since computer games are highly interactive, extremely low delay has to be achieved for both techniques. This interactivity also requires the game controllers' commands to be captured on the end device, streamed to the server, and injected into the game process [3]. The described functions are implemented by an application which is running on the client and a "return cannel" which is constructed between the clients and the server. The application on the client is responsible for recording any input command arriving from every existing input device while the return channel is utilized in order to send the commands from the clients to the server for execution. On the server side the commands are injected into the proper game window.

In order to ground our research and system developments, we have performed a thorough analysis of state of the art in gaming platforms available in today's market, presented in Section 2. The rest of the paper is organized as follows: Section 3 describes the Games@Large framework; Section 4 its components and operation fundamentals; Section 5 presents some experimental results on tests of the initial system and its components demonstrating multiple game execution on a PC and Quality of Service (QoS) optimized transmission of the games' graphics to the end devices via a wireless network; Section 6 presents the conclusions.

2. Gaming Platforms Analysis: State of the Art in Consoles, PC and Set Top Boxes

We have conducted an overview of state of the art in common gaming platforms such as consoles, PCs and set top boxes. One recent development in gaming market activities which has implications for new consumption patterns is technology based on distributed-cross-platform computing (or cloud computing); we introduce and overview this relatively new concept of Virtual Networked Gaming Platforms (VNGP) in Section 2.4.

2.1. Consoles. The home console system enables cheap hardware and guarantees product quality. Unlike the past, console functionality is being continuously upgraded post-release (e.g., web-browser and Wii channels on the Wii; high-definition video-on-demand downloading for Xbox 360; and PlayStation Home for PS3).

Xbox 360 (Microsoft). Microsoft were the first to release their next generation console, the Xbox 360, followed by the Xbox 360 Elite designed to store and display high definition video with a 120 GB hard drive. The Xbox 360 has perhaps the strongest list of titles of the three next gen consoles, including Halo 3 in 2007, though the style and content of each console's titles differ from the others and personal preferences play a role in which catalogue, and therefore which platform, appeals most to a certain gamer/user. In online functionality Microsoft is the most well established with its Xbox Live/Live Anywhere/Games for Windows-LIVE gaming services, Live Marketplace (used to distribute television and movies), and online Xbox Live Pipeline.

PlayStation 3/PS3 (Sony). As the most powerful games console ever made, it is the most future-proof in terms of where games development can go and its built-in Blu-Ray player. Expert reviews on the console have improved since its initial reception and commentators have remarked that the first PS3 games only use about 30–40% of the platform's capacity, so the gaming experience it offers should improve as developers use more of its capacity. PS3 functionality includes streaming movies or PS3 games to PSP over LAN. In Europe the PS3 is backwards compatible with most of the massive PS2 games catalogue (with all in US and Japan). Online Functionality/Support: The PS3 has a web browser based on NetFront; Home is a free community-based gaming service.

Wii (Nintendo). Nintendo's Wii features gesture recognition controllers allowing intuitive control and more physical play which must take much credit for the Wii's successful appeal to many consumers who had not been gamers before. The Wii also has a large back-catalogue of GameCube titles and developing games is cheaper and easier than for other platforms, suggesting a rapid proliferation of titles. Online functionality: Opera web browser software; a growing number of Wii Channels; the Message Board supports messaging with Wii users around the world via WiiConnect24, handles the Wii email messaging, and logs all play history, facilitating parental supervision; some titles now support multiplayer online play.

2.2. PCs. The PC is an open system which can be exploited by virtually any game manufacturer. It is also the broadest of gaming platforms—catering to casual games and casual gamers but also through to the top end of digital gaming in specialised gaming PCs. The PC has by far the

highest install base of all gaming platforms (discounting simple mobiles) with rising broadband connections and very well-developed online games services, such as Games for Windows-LIVE, PlayLinc, and many casual games sites (e.g., Verizon, DishGames, RealArcade, Buzztime). Though relatively expensive, it is bought and used for many things besides gaming but is often not equally accessible to all members of the household. This multifunctional nature of the PC is being eroded by nongaming functionality being added to consoles. Game Explorer is a new one-stop application within Vista designed to make game installation far simpler and also allows parents to enforce parental controls.

Input Devices. The PC and the games based on it use keyboard and mouse as the input device, which allows more complex games to be played but does not travel well into the living room where the large-screen TV, 10-foot viewing experience, comfy chairs, and social gaming are enjoyed by console gamers. There are some existing living room-friendly PC-game controllers though they have not been widely taken up. Microsoft's wireless game-pad is compatible with both the PC as well as the Xbox 360. A gamepad is, however, not suited for playing some game genres associated with the PC (notably MMOGs and real-time strategy/RTS) but viable alternative control devices do exist which could allow PC games of all genres to successfully migrate to the TV (e.g., Microsoft's qwerty keyboard add-on for the Xbox 360/PC wireless gamepad, trackball controllers such as the BodieLobus Paradox gamepad, or the EZ Commander Trackball PC Remote). They could also help the development of new games and peripherals and support web features on TV (such as Intel/Yahoo's planned Widget Channel).

2.3. Set Top Boxes. The Set Top Box is emerging as a platform for casual games and some service providers are offering games-on-demand services (e.g., the long-established Sky Gamestar). The fast growth of digital terrestrial television (DTT) in Europe also suggests the STB install base will rise steadily, potentially greatly increasing its role. With a potentially large mainstream audience, support from advertising revenues could be significant for STB gaming. Wi-Fi-enabled set top boxes (e.g., Archos TV+) are starting to emerge which combine a Wi-Fi Media Player with a high-capacity personal video recorder (PVR) for enjoying movies, music, photos, podcasts, web video, the full internet and more on widescreen TV.

Several companies are committed to enabling gaming services for the STB platform, including Zodiac Interactive, PixelPlay, Buzztime, TV Head, and PlayJam in the US, and Visiware, G-Cluster, and Visionik (part of NDS) in Europe. These companies provide their content and technology solutions to a few TV service providers currently deploying gaming services, including BSkyB, Orange, EchoStar, and Cablevision. Several Telco TV and DBS TV service providers in the US are actively exploring 3D STB gaming and their demos make many of today's cable STB games look antiquated (see Section 2.4 for details of NDS Xtreamplay technology).

User uptake of gaming platforms and choice of console depend on games catalogues and online services as well as hardware specifications and functionality. PC games are effectively tied to the desktop/laptop and console gaming is seen by many as expensive or for dedicated gamers only. The Wii has broadened the console user base but there remains a massive potential for mainstream gaming on TV given the right technology solution, content/services offerings and pricing. The open PC platform is supported by much programming expertise and is powerful and ubiquitous but PC games need to make the transition to the more comfortable and social TV-spaces with a wide range of low-cost, accessible, digitally distributed games-on-demand.

2.4. State of the Art in Virtual Networked Media Platforms. Of great relevance to Games@Large are developments in technology aimed at putting PC gaming onto TV screens. Service providers and web-based services are moving into the PC-gaming value chain and several commercial solutions for streaming games over the network exist already. These allow game play on smaller end-devices like low-cost PCs or set top boxes without requiring the games to be installed locally. Most of these systems are based on video streaming. A server executes the game, the graphical output is captured, and then transmitted as video to the client. For an interactive experience, such a system requires low end-to-end delay, high compression efficiency, and low encoding complexity. Therefore, many solutions have adapted standard video streaming and optimized for the particular graphical content. For example, t5 labs announced a solution for instant gaming on set top boxes via centralized PC based servers, which are hosted by cable TV or IPTV operators. In order to reduce the encoding complexity at the server which has to execute the game and the video encoder, particular motion prediction is conducted exploiting information about the graphical content. Reductions of 50–80 % in encoding complexity are reported. In contrast, StreamMyGame from Tenomichi Limited also offers a server solution which enables the user to stream his/her own PC games to another PC in the home, record the game play or broadcast the games for spectators. The streaming is based on MPEG-4 and typical bit-rates of 4 Mbit/s at XGA resolution are reported. Besides PCs, multiple different end devices are supported such as PlayStation 3, set top boxes and networked media devices. Similar to the other two approaches, G-Cluster's server client system also offers MPEG-based compression of game content and its streaming to end devices for remote gaming applications. Currently, this system has been employed by operators mainly for casual games. A system that offers high-definition (HD) resolution is the Xtremeplay technology of NDS. They enable the high resolution streaming of computer games to set top boxes, but adaptations of the game code to the Xtreamplay framework are required. High resolution streaming of game content is also provided by the Californian company Dyyno. However, their application is not interactive gaming over networks but the distribution of

game output to remote displays. Another somewhat different approach is AWOMO from Virgin Games. In contrast to the other approaches, they do not stream the game output but the game code. The game is downloaded and installed locally on a PC for execution. However, the technology offers a progressive game download, such that the user can start playing the game after only a small part of the data has been received. The remaining data is continuously fetched during game play. A similar approach is also used by the InstantAction system from GarageGames. Users can play 3D games in their web browser. InstantAction uses a small plug-in and an initial download of the game which are required to allow play.

Another streaming solution is offered by Orb (http://www.orbnetworks.com). Downloading Orb's free remote-access software, MyCasting 2.0, onto a PC (Windows only) transforms it into a 'broadcast device', the content of which can now be accessed from any web-enabled device (PC, mobile phone, etc.) with a streaming media player. MyCasting 2.0 now works with gaming consoles, enabling Xbox 360/Wii/PS3-owners to stream PC content onto the TV. Orb's software has enabled 17 million households (according to ABI Research) to bridge the PC-to-TV divide, at no cost, using what is essentially existing technology. However, streaming of video games is not supported.

Advances in wireless home entertainment networks and connectivity—which stream content between devices within the home—also present potentially important solutions for playing PC games on TV screens. For example, Airgo Networks' faster-than-wired True MIMO Media technology will allow streaming of rich high-definition television (HDTV) content to SimpleWare Home (STMicroelectronics) enabled devices within the home (at speeds faster than 10/100 Ethernet). Intel has collaborated with Verizon to launch a games-on-demand service that allows consumers to play PC games on their TV sets using Intel Viiv PCs. Also planned is a version of the online multiplayer service, PlayLinc, which will tie in with the service.

Although there is very little detailed technical information publicly available about the commercial systems, there have been many publications on streaming graphical content in the academic field. In [4], for example, a thin client has been presented, that uses high-performance H.264 video encoding for streaming the graphical content of an application to a weaker end device. In this work, the buffering scheme at the client has been optimized in order to achieve minimal delay necessary for interactive applications, but encoding is based on standard video encoding. In contrast, [5] exploits information from the graphics scene in order to directly compute the motion vectors and thus significantly reduces the computational complexity of the MPEG-4 encoding process. The work in [6] also uses MPEG-4 as codec but goes one step further by using more information from the graphics state. For example, different quantizer settings are used dependent on the z-buffer content. Thus, objects that are further away in the scene are encoded with lower quality than foreground objects closer to the camera. Both approaches, however, require an application that passes the necessary graphics information to the codec and does not work with existing game programs. If encoding complexity should be reduced even more, simple encoding techniques can be used. In [7], a nonstandard compliant codec is presented that allows the streaming of graphics content with very little encoding effort. Coding efficiency is, however, also much lower than when using highly sophisticated codecs like H.264.

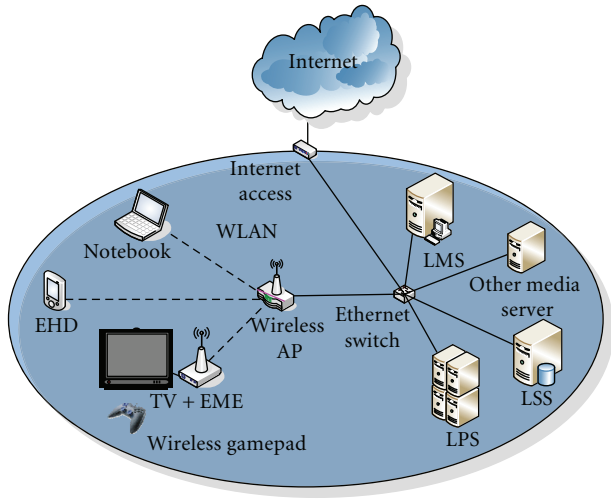


Figure 1: Games@Large framework.

The reviewed systems offer a variety of possibilities though all of them have limitations for interactive media such as video games, and especially existing game titles. For some of these formats games would need to be specially made or expensive hardware purchased, other formats provide moderate visual quality, unlike Games@Large's aim of being able to run all or most standard PC games including newly developed ones with high visual quality (in Sections 4 and 5 we will present some criteria for titles to be supported by Games@Large). Games@Large aims to offer benefits for wider stakeholders too (service providers, games developers/publishers, CE manufacturers, and advertisers) enabling business models which ensure that end users benefit not only from the technology solution but a wide choice of products and services at low cost.

3. Games@Large Framework

The Games@Large framework depicted in Figure 1 enables interactive media streaming from a PC-based machine to other CE, computer and mobile devices in homes and enterprise environments such as hotels, internet cafés and elderly homes.

The framework includes the following main components that are briefly introduced below and described in detail in Section 4.

Server Side. The Local Storage Server (LSS) is responsible for storage of games. The Local Processing Server (LPS) a Windows PC runs games from LSS and streams to clients. It is responsible for launching the game process after client-side

invocation, managing its performance, allocating computing resources, filing system and I/O activities, and capturing the game graphic commands or already rendered frame buffer for video encoding, as well as managing execution of multiple games. The LPS is further responsible for receiving the game controller commands from the end device and injecting them into the game process. The LPS is also responsible for streaming game audio to the client.

Graphic Streaming Protocol Stack. The Graphics Streaming Protocol is intended to become a standard protocol used for streaming 3D commands to an end device allowing lower performance devices such as STBs to present high performance 3D applications such as games without the need to actually execute the games on this device.

The video streaming scenario is intended for devices lacking hardware accelerated rendering capabilities. H.264 [8] is exploited for low-delay video encoding. Synchronisation and transmission is realised via UDP-based RTP/RTCP in a standard compliant way.

HE-AACv2 [9] is used for audio streaming. Again, synchronisation and transmission is realised via UDP-based RTP/RTCP in a standard compliant way.

Client Side devices. Notebook (NB); Enhanced Multimedia Extender (EME), which is a WinCE or Linux set top box; Enhanced Handheld Device (EHD)—a Linux-based handheld. The client module is responsible for receiving the 3D commands and rendering them on the end device using local rendering capabilities (OpenGL or DirectX). For the video streaming approach, H.264 decoding must be supported instead. The client is also responsible for capturing the controller (e.g., keyboard or gamepad) commands and transmitting them to the processing server [3].

4. Games@Large Framework Components

4.1. 3D Graphics Streaming. Today, interfaces between operating system level libraries, such as DirectX and OpenGL, and the underlying 3D graphics cards, occur in the operating system driver and kernel level and are transmitted over the computer bus. Simultaneous rendering of multiple games and encoding their output can overload a high-performance server. For that purpose DirectX, and/or OpenGL graphics commands, has to be captured at the server (LPS/PC) and streamed to the client (e.g., STB or a laptop) for remote rendering. This is similar to the 2D streaming of an X server in UNIX-based systems. Extensions for streaming 3D graphics also exist, for example, the OpenGL Stream Codec (GLS) that allows the local rendering of OpenGL commands. These systems usually work in an error-free TCP/IP scenario, with best effort transmission without any delay constraints.

The 3D streaming and remote rendering developed for Games@Large are achieved by multiple encoding and transmission layers shown in Figure 2. First of which is the interception and the very last one is the rendering on the client machine. All layers in between these two are independent of any specific graphics API. The latter implies that the postinterception 3D data streamed till the client rendering process is not specific to either DirectX or OpenGL, but rather utilises higher-level concepts common to all 3D graphics.

Since efficient direct translation from DirectX API commands to OpenGL commands is difficult, due to the significant differences between these APIs, a set of common generic concepts may be of assistance. In general, a 3D scene consists of multiple objects that are rendered separately. Before rendering an object, several parameters (states) must be set and these include lighting, textures, materials, the set of 3D vertices that make a scene, and further various standard 3D transforms (e.g., translate, scale, rotate).

Figure 2 depicts the detailed block diagram of the components involved in the 3D streaming. First, the 3D commands issued by the game executable to the graphic layer API used by the selected game (e.g., DirectX v9) need to be captured. The same technique used for capturing the DirectX v9 can also be used for capturing other versions of DirectX (and also the 2D version of DirectX-DirectDraw). This is implemented by providing to the game running on the LPS a pseudo-rendering environment that intercepts the DirectX calls. The proxy Dynamic Link Library (DLL) is loaded by the game on its start-up and runs in the game context. This library forms the server part of the pipeline which passes the 3D commands from the game executable to the client's rendering module.

In our implementation, we have implemented delegates objects for each of the 3D objects created by the game. Each such delegates object uses the 3D streaming pipeline for processing the command and its arguments. For many commands, a delegate's object can answer the game executable immediately without interaction with the client—this is done in many cases in order to avoid synchronized commands. For example, when the game needs to change a texture (or vertex buffer) on the graphic card, it first locks it, and then it changes the buffer and then unlocks the texture. Originally, those commands must be synchronized. But in our implementation, the delegate object for texture does not interact with the client when the game tries to lock the texture on the graphic card but postpone the call for the unlock call. When the game issues an unlock call, the delegate object checks what parts of the texture were changed and sends a single command to the client with the changes. The client implementation, which is aware of this logic, will first lock the corresponding texture on the client's graphic card, change the texture and unlock it. This is one example of commands virtualization that allows avoiding synchronous commands, and reducing the number of commands—typically such a set of commands is called hundreds of times per frame.

The Serialization Layer serializes various structures describing the graphics state to a buffer. Serializer's additional function is to fill the buffers until certain criteria is met (theoretically it can pass the buffer to compressor after each command which, of course, would not be efficient for networking). The compression layer's purpose is to use an

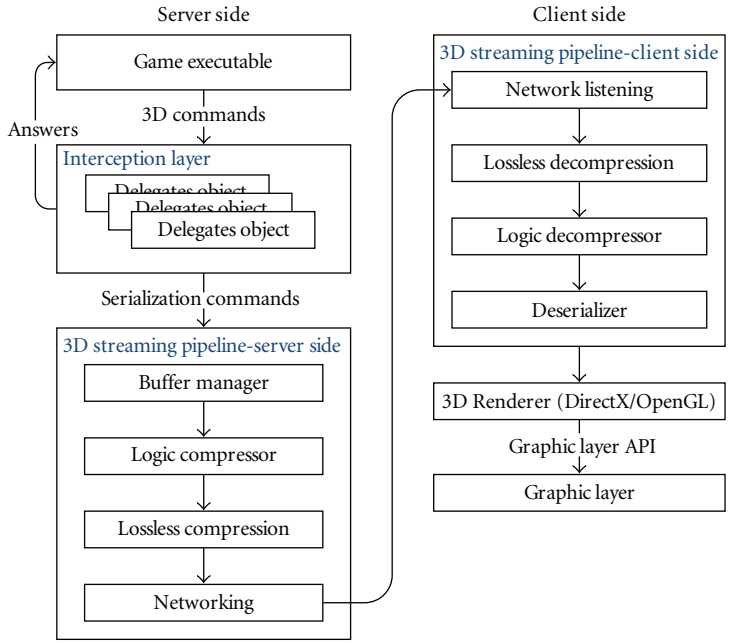


Figure 2: 3D Streaming—detailed block diagram.

efficient third-party compression library (e.g., zlib or LZO compression) to compress the 3D stream before sending it to the network.

The Network Layer is responsible for maintaining the connection with the client and for sending the buffers. After each sent buffer, an ACK (acknowledgement) is sent back by the client. The purpose of this ACK is to further synchronize server and client and to try to not overflow network buffers. The nature of the data requires that no buffer will be lost in transmission (which, in the current implementation, implies the use of TCP). A possibility to use or develop a transport protocol (e.g., UDP based) which could replace TCP is investigated.

On Microsoft Windows clients the renderer is using DirectX to render the commands, while in Linux clients the renderer is using OpenGL commands. There is a certain overhead in OpenGL rendering because some data (especially colour and vertex data) must be reorganised or rearranged in the processing stack before it can be given to OpenGL for rendering. This may result in increased demand of Central Processing Unit (CPU) processing and memory transfer between system memory and the Graphics Processing Unit (GPU) [3].

Although the graphic streaming approach is the preferable solution since it offers lower latency and enables execution of multiple games on one server, it cannot be used for some small handheld devices like PDAs or smart phones. These end-devices typically lack the hardware capability for accelerated rendering and cannot create the images locally for displaying them. Therefore, the alternative solution using video streaming techniques is described in the next section.

4.2. Video Encoding. The alternative approach to 3D Graphics Streaming in the Games@Large framework is Video Streaming. It is used mainly for end devices without a GPU, like handheld devices, typically having screens of lower resolution. Here the graphical output is rendered on the game server and the frame-buffer is captured and transmitted encoded as video stream. For video encoding, the H.264 video coding standard is used [8], which is the current state of the art in this field and provides the best compression efficiency. But in comparison to previous video coding standards, the computational complexity is significantly higher. However, by selecting appropriate encoding modes, the encoding complexity for the synthetic frames can be significantly reduced while preserving high image quality.

In order to keep the effort moderate for integrating new client end devices into the Games@Large framework, the video streaming subsystem has been developed in a fully standard-compliant way. Nevertheless, the server side encoding and streaming is adapted to the characteristics of the present end device. This means that end device properties such as display resolution or supported decoding profiles are selected appropriately on the server (e.g., optional H.264 encoding with CABAC [10], which typically increases the compression efficiency at the cost of increased computational load of decoding at the client). Similarly, the proportion of IDR frames in the resulting video stream, which are used to resolve the dependence on previous frames, can be set under consideration of the network properties.

The delay between image generation on the server side and presentation on the client side is crucial and has to be as small as possible in order to achieve interactive gaming. To reduce this delay a H.264 decoder for end devices has been developed which is implemented with a minimum of buffering. As soon as a video frame has been received it will be decoded and displayed. This is quite different to TV

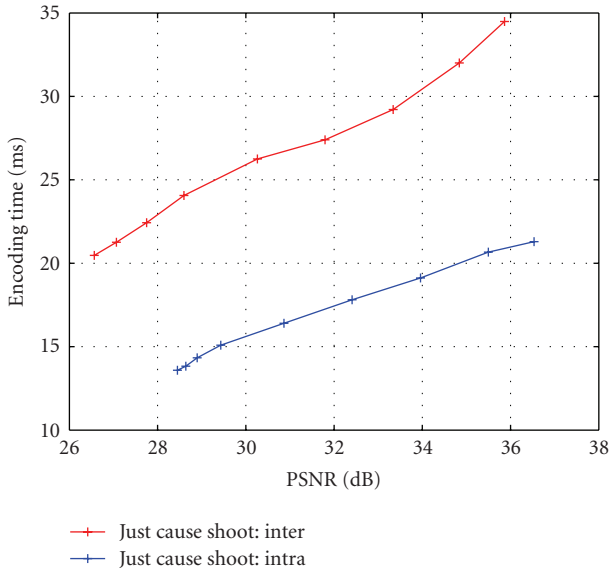


Figure 3: Comparison encoding timings for different quantizer settings.

streaming where large buffering is used to remove the effects of network jitters.

H.264 video encoding is computationally quite demanding. In Figure 3 the encoding times for a game scene are depicted for streams encoded with different quantizer settings which results in different qualities. It is clearly visible that for increased image quality the encoding time increases. Since the video encoding is executed in parallel to the actual game both are competing for the processor time. Aside from that, the desire to execute and stream several games simultaneously from a single game server increases the need for reduction in computational complexity in the video streaming system.

One method for reducing the complexity at the server is the removal of the scaling of the games output to the required resolution of the client device. For that purpose, the render commands of the game are intercepted and modified, so that the rendered images always fit the end device's resolution. Besides the reduction in complexity, an advantage of this technique is that the quality of the images achieved is much better, because the images are already rendered at the desired resolution without any scaling artefacts. An example is depicted in Figure 4.

Current research is focused on reducing the computational complexity of the H.264 encoder itself by incorporating enhancements based on the available rendering context information. The main idea is adapted from [11]. The motion prediction in video encoding, which is realized in common encoders as a computationally very demanding trial and error search, can be calculated directly by using the current z-buffer as well as projection parameters available in the games rendering context of OpenGL/DirectX. The encoding complexity can be reduced further by predicting the macroblock partitioning on the basis of discontinuities in the z-buffer. This is also usually realized in common encoders as a computationally demanding trial and error search. The key difference to [11] is that in [11] the authors assume to have full access to the rendering applications source code. In the Games@Large framework the output is generated from unmodified commercial games, which use quite sophisticated rendering techniques. The challenge here is to capture the appropriate information of the rendering context in order to correctly perform the motion prediction.

In order to transmit the encoded video stream in real-time, the RTP Packetization (Real Time Protocol [12]) is utilized. The structure of the H.264 payload for RTP is specified in [13]. Further details about real-time streaming and synchronization are discussed in Section 4.4.

4.3. Audio Encoding. Besides the visual appearance computer games also produce sounds. In order to deliver this audio data to the client in an efficient manner, an audio streaming sub-system has been developed. Since computer games typically produce their audio samples in a block-oriented manner, the current state-of-the-art audio encoder in this field has been integrated: the High Efficiency Advanced Audio Coding version 2 (HE AAC-v2) [9]. Our HE AAC-v2 implementation is configurable so that it can encode mono or stereo, 8 or 16 bits per sample and at several sample rates, for example, 22.05, 44.1, or 48 kHz. In order to stream the encoded audio data in real-time the RTP packetization (Real Time Protocol [12]) is utilized. The structure of HE AAC-v2 payload for RTP is specified in [14]. Further details about real-time streaming and synchronization are discussed in Section 4.4.

4.4. Synchronized Real Time Streaming. Since the performance of the system is highly dependent on the delay between content generation on the server side and its play back on the client, the video streaming as well as the audio streaming are based on the UDP-based RTP (Real Time Protocol [12]). Every RTP network packet contains a time stamp as well as a well defined structure of payload data.

In order to prevent errors of different timings among the video and audio channels and to overcome different kinds of network jitters, the RTP channels are explicitly synchronized. For this purpose the RTCP (Real Time Control Protocol [12]) has been integrated. The content-generating server periodically sends a so-called Sender Report RTCP Packet (SR) for each RTP channel. This SR contains a mapping from the timestamps used in the associated RTP channel to the global NTP (Network Time Protocol [15]). With this synchronization of each RTP channel to NTP time, all the RTP channels are synchronized implicitly with each other.

4.5. Client Feedback to the Game Server. The return channel on the server side is responsible for receiving the commands from each connected client and injecting them to the appropriate game; the one that the user is playing. The return

(a) (b)

Figure 4: Rendering in resolution adapted to particular end device.

channel is constructed by two communicating modules; the server side module and the client side module.

4.5.1. Server Side. The server side module that implements the return channel is part of the core of the system and more specifically the Local Processing Server. The return channel on the server side is responsible for receiving commands from each connected client and transforming them in such a form that they will be readable by the OS (Windows XP/Vista) and more specifically by the running instance of the game. The method utilizes a proxy of the DirectInput dynamic library and injects the commands directly to the DirectInput functions used by each game.

A crucial part of the server and client side return channel is the socket communication. The HawkNL [16] library is used for the communication between the server and the clients. This assures that the implementation of the socket is based on a system that is tested by a large community of users and that no major bugs exist on that part of the code. For faster communication between client and server we disable the Nagle Algorithm [17] of the TCP/IP communication protocol. Having done so, the delivery times of the packets are almost instantaneous as we omit any buffering delays.

4.5.2. Keyboard. The server side of the return channel receives the keyboard commands that originate from the client dedicated socket connection. The communication between the server and the client follows a specific protocol in order to (a) be successfully recognized by the server and (b) preserve the loss of keyboard input commands. An important aspect of the return channel infrastructure is the encryption of keyboard commands which is described in the following section.

For the case of a game that uses a DirectInput keyboard, we implement a proxy dll method. For this method, we create a modified dinput8.dll of our own, modifying only the function that is used for passing data to the virtual DirectInput keyboard device that is created when the game launches in order to read data from the original keyboard.

Encryption. The encryption procedure is needed only for the keyboard commands that the client transmits, since sensitive user data, such as credit card numbers or passwords, are only inserted using the keyboard. RSA encryption was selected as it fulfils the demands of our specific environment.

Start-Up Phase. When both the client and the server start, some local initializations take place. The client then launches a connection request to the server which is advertised to the network neighbourhood through the UPnP module. The server accepts the new client generating a unique RSA public-private key combination.

Transfer of Encrypted Keyboard Input. The idea that lies beneath the communication command channel architecture is depicted in Figure 5.

Each end device consists of many possible input devices for interacting with the server. When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, for example, the device discovery module which uses UPnP. The next step of the procedure is to capture the input coming from the controllers. This is achieved by recording the key codes coming from the input devices. Mice or keyboards are interrupt-driven while with joysticks or joy pads the polling method is used for reading. If the command that is to be transferred is originating from a keyboard device, the client uses the server's public key to encrypt the data after it has been suitably formatted adhering to a certain communication protocol. The encrypted message is transmitted to the server using the already existing socket connection.

Once the encrypted message has arrived at the server side, the server decrypts it using its private key, obtaining the initial keyboard commands that the client has captured.

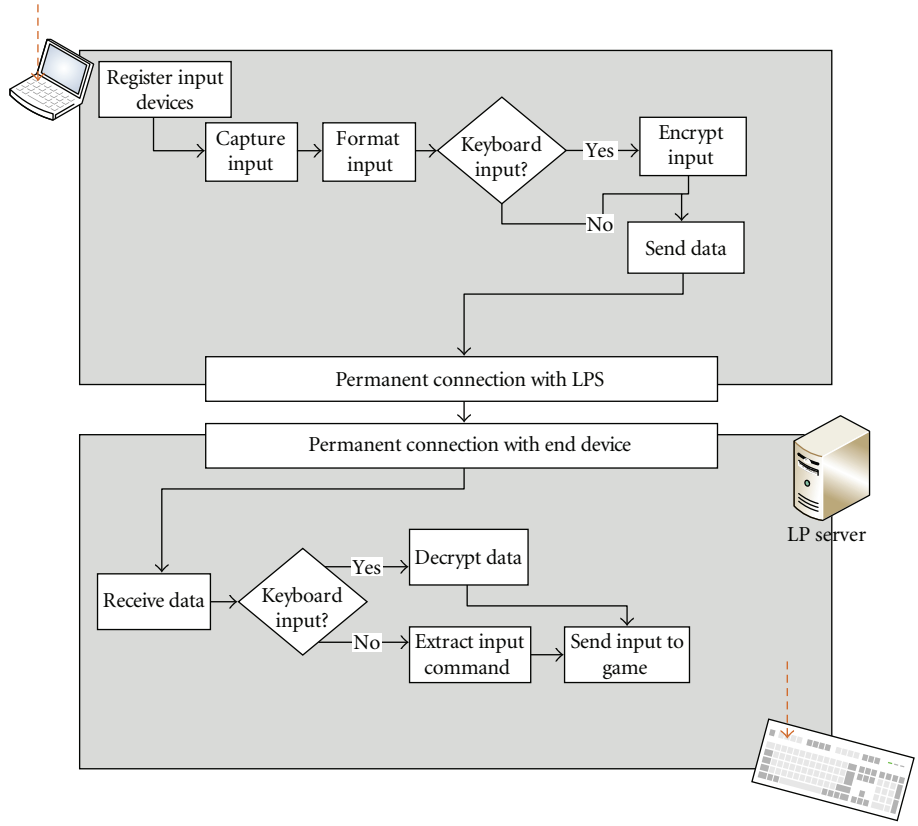


Figure 5: Encrypted command channel.

If the received message is not from a keyboard, the server bypasses the decryption stage, delivering the commands at the running game instance. The algorithm procedure of this step is described in the following sections.

4.5.3. Mouse. The server side of the return channel receives the mouse commands that originate from the client using the already open socket connection. The communication between the server and the client follows a specific protocol in order to be successfully recognized by the server and is exactly the same as the keyboard apart from the encryption part and the resolution part that follows.

An issue that arises when using the mouse input device is how the commands are executed correctly if the client has a different resolution to the server. This is because what is sent from the client to the server is the absolute mouse position. We realized that when a game is running on the client, the rightmost bottom position of the mouse equals the resolution of the game when running in 3D streaming, and it is equal to the screen resolution when running in Video streaming. On the server side, we observed that the matching of the resolutions should not be done with the resolution of the screen but again with the resolution of the game running on the server because every command is injected into the game window. The mouse positions have to be normalized on the client and the server side.

4.5.4. Joypad/Other. The server side of the return channel receives mouse and keyboard commands that originate from the client's Joypad/Other input via the already open socket connection. This means that any Joypad/Other input is firstly translated into suitable keyboard and mouse commands on the client side (using XML mapping files) and it is then transmitted to the server for execution at the game instance. The execution of these commands falls to the previously described cases.

4.6. Quality of Service Optimized Transmission. The Games@Large gaming architecture is based on streaming a game's 3D or video output to the client running on a separate device. This kind of distributed operation sets high requirements for the network in terms of bit rate and latency. A game stream with sufficient quality is targeted to require a bit rate of several megabits per second and the latencies have to be minimized to maximize the gaming quality. The same network which is used for gaming is also assumed to be available to other applications such as web surfing or file downloading. If the network did not have any kind of QoS support, these competing applications would have a negative effect on the gaming experience. Thus, the network has to implement QoS to satisfy the requirements of gaming regardless of other applications using the same network.

As presented in Figure 1, the network connection to the game client can be wireless. This is a further challenge for

providing QoS for the gaming application. Our platform is based on IEEE 802.11 standard family [18] wireless LAN (WLAN) technologies. Currently, the most used WLAN technology is IEEE 802.11g which could provide the bandwidth needed for four simultaneous game sessions in good conditions. The near future IEEE 802.11n will enhance the maximum bit rate, but still shares the same basic medium access (MAC) method which does not support QoS. Priority-based QoS can be supported in IEEE WLANs with the Wi-Fi Multimedia (WMM) extensions [19] specified by Wi-Fi Alliance. WMM is a subset of IEEE 802.11e standard [20] and divides the network traffic into four access categories which receive different priority for the channel access in competition situations. In this way applications with high QoS requirements can be supported with better service than others with less strict requirements. Our platform is based on IEEE 802.11 (either g or n) and WMM. As presented later in the results section, WMM can be used to enhance the gaming experience substantially compared to the case of basic WLAN MAC.

In addition to MAC layer QoS support, there is a need for QoS management solutions in a complete QoS solution. Our platform relies on UPnP QoS specification [21]. The specification defines services for policy management and network resource allocation. In practice, it acts as a middleware between the applications and the network devices performing the QoS provisioning.

The experimental results presented later in this paper prove that our standard-based solution enhances the game experience and gives superior performance compared to reference system without QoS support.

4.7. UPnP Device Discovery. To ensure easy system setup and operation as well as flexibility in dynamic home networks, various system components need to find each other automatically and be able to exchange information about their capabilities.

In the Games@Large system, we have selected to use the UPnP Forum [22] defined technologies for this functionality.

UPnP technology defines architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. The technologies leveraged in the UPnP architecture include common internet protocols such as IP, TCP, UDP, HTTP and XML [23].

The required functionality of device discovery is to allow a Games@Large client to find Games@Large servers in the network it is connected to. Device discovery is also available in servers to find other servers in the larger Games@Large network. For example, in a large system a Local Management Server (LMS) needs to find all LSSs in the network; in the home version, the logical servers are usually located in a single PC, but in an enterprise version, such as a hotel environment, there might be several physical server machines.

The device discovery component is also able to find information about services provided by the found devices. In the discovery phase the devices are also exchanging capability information, for example, an end device will inform the server of its capabilities, like screen resolution, connected input devices and so on. Servers can also advertise their capabilities to other servers and end devices.

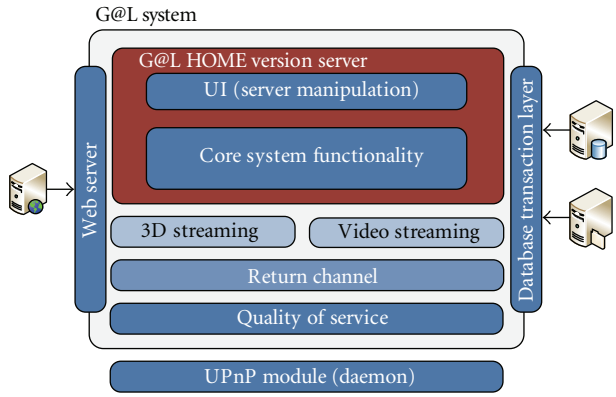


Figure 6: General server architecture.

4.8. System Integration. The local servers of Game@Large consist of three separate servers: LPS (Local Processing Server), LMS (Local Management Server), and LSS (Local Storage Server). In the (intended for the use in home environment) version, the main server of the system, is the Local Processing Server and at this stage it has (virtually) the core functionality which includes LPS, LMS, and LSS.

4.8.1. Local Processing Server. The "virtual" Local Processing Server is the core of the Games@Large System HOME version. It handles every communication with the clients while being responsible for every internal communication in parallel. The following Figure 6 represents the general server architecture.

At this stage of the implementation everything is manipulated within the server application. This web server is an Apache [24] server with support of PHP [25] and sqLITE [26] (as a PHP module) which is the database used in the HOME version of the system.

The LPS incorporates the implementations of 3D and Video Streaming, the Return Channel and the Quality of Service modules. In parallel it has a Web Server for serving the Web UI (user interface) to the clients and a Database Transaction Layer for the communication with the Database and the File System (game installations).

The basic procedure of the Processing Server is depicted in Figure 7.

When a client wants to connect to the system, it tries to locate the LPS that is running the G@L HOME system. The UPnP daemon that runs on the LPS "helps" each end device to locate the server's IP. The application that runs on each client launches a web browser with the given IP address and the LPS's Web Server starts interacting with the clients. The client is served with the corresponding web UI (different UI for each end device). The server is informed which UI has to be sent by a parameter that is passed together with the IP of the server in the web browser.

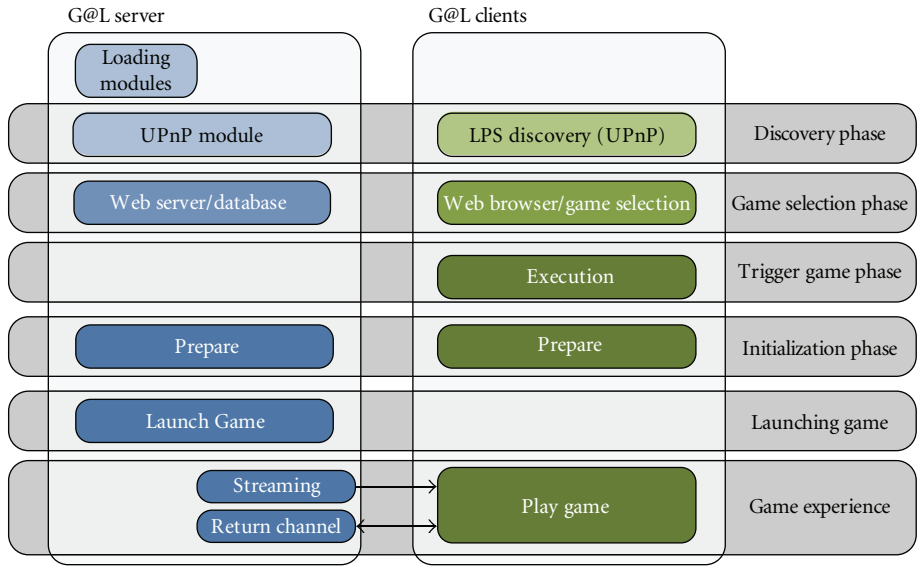


Figure 7: General flow of information.

After the log-in procedure of the end user, the game selection phase is launched. When the user selects a game to play the main client application is launched and the main communication procedures between the client and the server begin. The client is informing the LPS about its request to play a specific game. The LPS is processing the client's command and more specifically it starts the decision procedure.

During the decision procedure the server, with the help of the UPnP and QoS modules, observes the current system status and network utilization. If the game's Software, Hardware, and Network demands are met, then the game initialization procedure begins. The client is also informed that the launching of the game is imminent and thus it will be able to begin its initialization procedure. After the successful finishing of the initialization procedure, the game is launched with the 3D commands or video of the game streamed to the client. Additionally, the client is streaming the user's input commands to the server. The commands coming from the client are furthermore processed on the server side and they are delegated to the window of the game.

5. Experimental Results

In order to demonstrate multiple game execution and system performance analysis we designed a testbed [27] in which we could monitor the performance of network, devices and Games@Large system processes while running simultaneous game sessions. Figure 8 shows our testbed setup.

We performed our experiments with two client notebooks of which one was running Sprill (Casual game) and the second one Red Faction Demo (first person shooter game). The Games@Large server (Intel 2 GHz 2 CPUs, 2048 MB RAM, 256 MB dedicated video memory) running Windows XP was connected to a 100 Mbps switch which

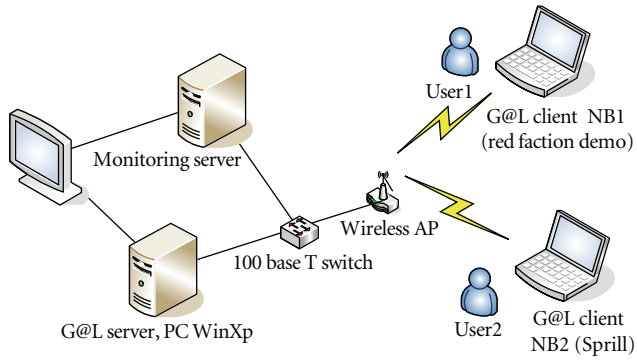


Figure 8: Games@Large testbed.

in turn connected to the WLAN Access Point (AP) via a wired Ethernet connection. The two client notebooks (NB1: Intel 2 GHz 2 CPUs, 1024 MB RAM, 384 MB shared video memory and NB2: AMD Athlon 2.1 GHz CPU, 1024 MB RAM, 256 MB dedicated video memory) were connected to the Wireless AP via the IEEE 802.11g wireless connection. For system performance monitoring we used an external Monitoring PC. All the PCs and NBs were SNMP/WMI enabled for performance monitoring purposes.

We used the PRTG Network Monitor [28] on the Monitoring PC to monitor network, device, and Games@Large processes with minimal influence on system's performance. Additionally we used FRAPS [29] to measure the games' frame rate.

The test scenario included a full system workflow which consisted of the following steps, shown also in Figure 7: G@L server discovery from the client device, web user interface access and game list browsing, selection of the game, and starting to play, described in detail in Section 4.8.1. Both the test participants were familiar with the 2 games used for tests.

Table 1: Frame rate per second for tested games run natively and on G@L system.

Mode	Game	Mean FPS	Std Dev
Run Natively (Server PC)	Red Faction	59.23	5.16
	Sprill	349.14	87.65
Run on G@L	Red Faction (NB1)	18.26	12.12
	Sprill (NB2)	125.27	108.59

We did not perform extensive user studies though we were recording user observations and perceptions about the gaming experience. Both participants commented that at the beginning there were some pauses in the game while it was loading, but after a while they disappeared. The gaming experience in the Mean Opinion Score (MOS) scale [30] was rated between 4 and 5. There were some differences with the original game play but participants were not frustrated and could enjoy the game play.

During the test sessions we were logging the frame rate of the games on the client devices, network, and G@L processes performance on the client and server. For analysis purposes, we ran both games natively on the server PC and measured their performance and frame rate in frames per second (FPS). Measurement results for native and G@L system game executions are presented in Tables 1 and 2.

Network usage during tests was measured on the server and both clients. Figure 9 shows the network usage bitrates for the G@L server simultaneously serving two clients. The mean sum bitrate for the clients is 6956.04 kbit/s for Red Faction game on NB1 and 8627 kbit/s for Sprill game on NB2, respectively.

The average bandwidth usage on the client (as well as on the server per single game) devices is correlated with the FPS. From mean bit rate of NB1 and NB2, and Table 1 we can see that when the frame rate is high, the network utilization is higher (Red Faction versus Sprill). The same correlation can be observed between the frame rate, CPU and memory utilization on the client and server. According to some proof of concept tests Windows based clients are running the games at higher frame rates than the Linux-based ones and those with the weak hardware capabilities.

The bandwidth that the game running on the LPS requires is directly proportional to the frame rate of the game. Clients that are capable of running games at high frame rates will spend a large portion of their time reading 3D data from the socket. After a frame has been read, an ACK is sent to the server so it can generate a fresh updated frame. When the frame-rate is above sufficient (20–25 FPS is enough for a good gaming experience), it can be artificially limited by the LPS to save the network resources. In such a way, it is possible for multiple (four good quality concurrent sessions per 1 AP/LPS) devices to be connected to the same LPS without overloading the network.

The server must have enough CPU power and memory to run the game natively. Additionally, the amount of video memory that a game requires when running natively, must be available in the system memory when running

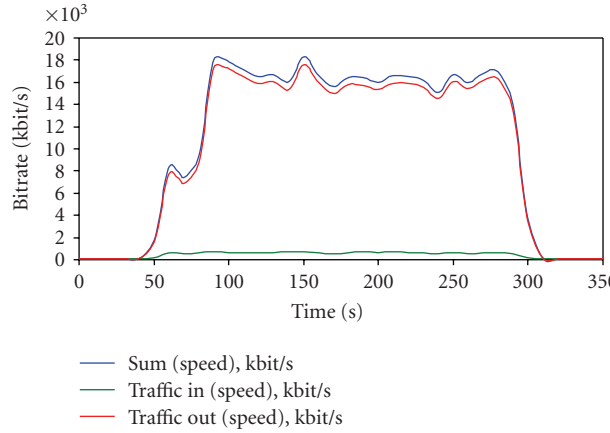


Figure 9: Bitrate versus Time for G@L Server.

in the G@L environment (that is because the graphic objects are emulated by the streaming module in the system memory). As for the CPU requirements, most games still do some graphics processing in software, so decoupling of the rendering from the game actually leads to a CPU gain on the server, see Table 2 (in spite the streaming and the compression). As long as the processing server has sufficient CPU and memory resources to run multiple games at once it can run them. Since the games' graphics are not rendered on the LPS (when 3D streaming), there is no competition between games for the GPU and neither for the full-screen mode.

The most important hardware requirement for the client device is the video adapter. It should have hardware acceleration capabilities to enable fast rendering of 3D scenes. As on the server, the graphic resources that the game stores in the video memory should be available in the system memory to enable manipulation prediction and cashing. So memory requirements for the client should be 200–300 MB available to the client application for fairly heavy games.

Besides the frame rate and technical requirements, such as hardware and network bandwidth, for a game to run playable on the end device in the Games@Large system it has to be compatible with the end device screen size and controller capabilities (e.g., some games cannot be played on small displays, other games cannot be controlled with the gamepad).

The above mentioned tests were performed over a Wi-Fi network without the QoS and with no other traffic (except the SNMP/WMI packets for system monitoring, but these crate a very small network load) present on the network than the one produced by the two game sessions of the client notebooks. Therefore latencies and other negative traffic effects did not assert during the tests, for example, measured mean round trip time (we sent an SNMP Ping of 30 Bytes from the server, every second 30 times) for both clients was <2 ms.

The solution for QoS optimized transmission described in Section 4.6 was evaluated by performing a series of tests in a laboratory environment. The experimental setup described

Table 2: G@L Server and Client process performance: Memory and CPU usage.

Mode	Game	Mean working set (Mbyte)/Std Dev	Mean CPU usage (%)/Std Dev
Run Natively (Server PC)	Red Faction	61.03/4.19	49.03%/1.75%
	Sprill	103.39/13.14	47.10%/7.68%
Run on G@L	Red Faction (NB1 process)	45.19/7.11	21.58%/7.03%
	Red Faction (Server process)	66.46/12.29	22.00%/8.11%
	Sprill (NB2 process)	112.15/30.36	67.69%/18.30%
	Sprill (Server process)	139.42/41.15	31.65%/10.67%

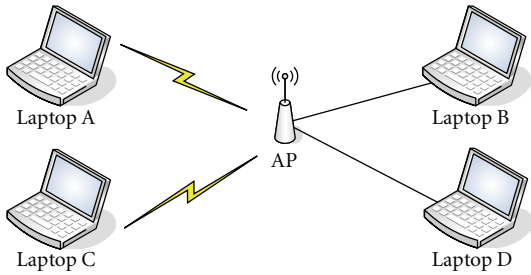


Figure 10: Test setup.

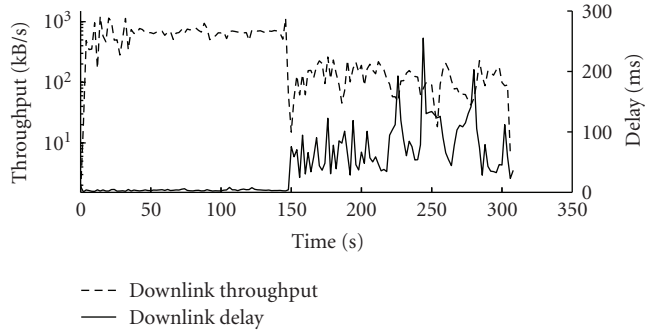


Figure 11: Downlink throughput and delay when both the game and the background traffic share the same priority.

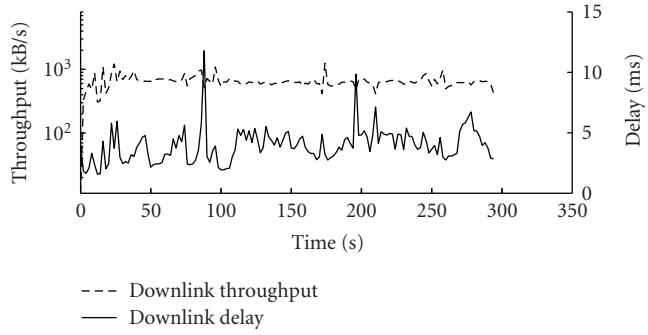


Figure 12: Downlink throughput and delay when the game has higher priority than the background traffic.

in Figure 10 includes four laptops and a WMM enabled WLAN access point (AP). Laptop A was used as a game client while laptop B was running the game server software. The game server was connected with a wired Ethernet connection to the WLAN and the client connection was wireless. In addition to game-related laptops, there were two additional laptops, C and D, which were used to generate background traffic to the network when testing the QoS capabilities of the solution. Similar to the game laptops, Laptop C was connected using a wireless connection and Laptop D with a wired connection. The laptops used were standard PC laptops equipped with IEEE 802.11g and WMM enabled wireless interfaces or 100 Mbps Ethernet interfaces. The AP was a normal WLAN AP with an addition of priority queuing in the AP kernel buffers in case of WMM queue overflow.

In each of the test cases, playing the game (Sprill, using 3D streaming) was begun in a wireless network without any additional traffic. From the middle until the end of the test, competing traffic stream was introduced from Laptop D to Laptop C. This stream was generated by an open source traffic generator iPerf. A single TCP session was used, thus simulating a file download between D and C using FTP or HTTP. The tests were performed with two QoS configurations. In the first one, all the traffic was sent using best effort priority, and in the second, the game traffic was using voice priority and the background best effort priority. Downlink and uplink throughput, delay, jitter, and packet losses for the gaming traffic were recorded using the QoSMeT tool [31] and the game client's realized frame rate was measured with Fraps [29].

The downlink performance is visualized in Figure 11 in terms of throughput and delay for the case with equal priority, and in Figure 12 for the case where gaming has higher priority. The effect of introducing the background traffic can be seen very clearly in Figure 11 while it is not visible in Figure 12.

The complete results are presented in Tables 3 and 4 for both cases respectively. In the case without prioritization, the game really suffers from the competing traffic in the WLAN. The downlink delay increases up to around 20 times as high as in uncongested conditions. This causes the realized frame rate at the client to decrease almost 90 percent which, together with the increased delay, practically destroys the game experience. When the game traffic is classified with a priority higher than the background traffic, the effect of competition is negligible. The downlink delay remains an acceptable level and the realized frame rate of the game at the client decreases only less than 10 percent.

Table 3: Average values when both the game and the background traffic share the same priority.

	Downlink throughput (kBps)	Uplink throughput (kBps)	Downlink delay (ms)	Uplink delay (ms)	Downlink jitter (ms)	Uplink jitter (ms)	Downlink packet loss (%)	Uplink packet loss (%)	Realized frame rate (fps)
Without competing traffic	651.3	26.6	3.4	1.7	1.0	0.8	0.020	0.009	82.8
With competing traffic	119.3	4.6	66.0	6.4	5.8	2.5	0.630	0.059	9.1
Ratio	0.18	0.17	19.60	3.67	5.73	3.24	31.16	6.65	0.11

Table 4: Average values when the game has higher priority than the background traffic.

	Downlink throughput (kBps)	Uplink throughput (kBps)	Downlink delay (ms)	Uplink delay (ms)	Downlink jitter (ms)	Uplink jitter (ms)	Downlink packet loss (%)	Uplink packet loss (%)	Realized frame rate (fps)
Without competing traffic	665.8	28.1	3.5	1.8	1.0	0.7	0	0.002	80.9
With competing traffic	624.4	25.7	4.2	2.2	1.2	0.8	0.003	0	73.5
Ratio	0.94	0.91	1.19	1.23	1.13	1.19	—	0	0.91

6. Conclusions

In this paper, we have presented a new distributed gaming platform for cross-platform video game delivery. An innovative architecture, transparent to legacy game code, allows distribution of a cross-platform gaming and entertainment on a variety of low-cost networked devices that are not able to run such games. This framework enables easy access to the game catalogue via the web based interface adapted for different end devices. A generalized protocol supports end devices with both OpenGL and DirectX API's. We have shown that it is feasible to use a single PC for multiple game executions and stream them with a high visual quality to concurrently connected clients via a wireless network using the QoS solution. The developed technology enables putting PC gaming onto TV screens which is a rapidly emerging trend in gaming market. Apart from that it also enables a pervasive video game access on handheld devices.

Future work is to support wider range of titles, we will need to implement the interception layer for all the graphic libraries used by the games which can supported by Game@Large. A possibility is investigated to use or develop a transport protocol (e.g., RTP), which could replace TCP for 3D streaming for the improvement of its performance over a wireless network. For video streaming current research is focused on reducing the computational complexity of the H.264 encoder itself by incorporating enhancements based on the available rendering context information using the motion prediction and by predicting the macroblock partitioning. In parallel, we will run extensive laboratory tests and field trials in the home environment in order to gather knowledge about users' perceptions and investigate the subjective expectations of gamers.

Acknowledgments

The work presented in this paper has been developed with the support of the European Integrated Project Games@Large (Contract IST-038453) which is partially funded by the European Commission.

References

[1] Y. Tzruya, A. Shani, F. Bellotti, and A. Jurgelionis, "Games@Large—a new platform for ubiquitous gaming and multimedia," in *Proceedings of the Broadband Europe Conference (BBEurope '06)*, Geneva, Switzerland, December 2006.

[2] S. Cacciaguerra and G. D'Angelo, "The playing session: enhanced playability for mobile gamers in massive metaverses," *International Journal of Computer Games Technology*, vol. 2008, Article ID 642314, 9 pages, 2008.

[3] I. Nave, H. David, A. Shani, A. Laikari, P. Eisert, and P. Fechteler, "Games@Large graphics streaming architecture," in *Proceedings of the 12th Annual IEEE International Symposium on Consumer Electronics (ISCE '08)*, pp. 1–4, Algarve, Portugal, April 2008.

[4] D. De Winter, P. Simoens, L. Deboosere, et al., "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications," in *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '06)*, Newport, RI, USA, May 2006.

[5] L. Cheng, A. Bhushan, R. Pajarola, and M. El Zarki, "Real-time 3d graphics streaming using mpeg-4," in *Proceedings of the IEEE/ACM Workshop on Broadband Wireless Services and Applications (BroadWise '04)*, pp. 1–16, San Jose, Calif, USA, July 2004.
[6] Y. Noimark and D. Cohen-Or, "Streaming scenes to MPEG-4 video-enabled devices," *IEEE Computer Graphics and Applications*, vol. 23, no. 1, pp. 58–64, 2003.
[7] S. Stegmaier, M. Magallón, and T. Ertl, "A generic solution for hardware accelerated remote visualization," in *Proceedings of the Symposium on Data Visualisation (VISSYM '02)*, pp. 87–94, Barcelona, Spain, May 2002.
[8] MPEG-4 AVC, "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.
[9] MPEG-4 HE-AAC, "ISO/IEC 14496-3:2005/Amd.2".
[10] P. Eisert and P. Fechteler, "Low delay streaming of computer graphics," in *Proceedings of the International Conference on Image Processing (ICIP '08)*, pp. 2704–2707, San Diego, Calif, USA, October 2008.
[11] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.
[12] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications".
[13] RFC 3984, "RTP Payload Format for H.264 Video".
[14] RFC 3640, "RTP Payload Format for Transport of MPEG-4 Elementary Streams".
[15] D. L. Mills, "Network time protocol version 4 reference and implementation guide," Tech. Rep. 06-6-1, Department of Electrical and Computer Engineering, University of Delaware, Newark, Del, USA, June 2006.
[16] Hawk Software, Hawk Network Library, http://www.hawksoft.com/hawknl.
[17] J. Nagle, "Congestion control in IP/TCP internetworks," RFC 896, January 1984.
[18] IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.
[19] Wi-Fi Alliance Technical Committee, QoS Task Group, WMM (including WMM power save) specification V1.1, 2004.
[20] IEEE 802.11e-2005, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements," 2005.
[21] UPnP QoS Architecture V2.0, http://www.upnp.org/specs/qos/UPnP-qos-Architecture-v2-20061016.pdf.
[22] UPnP Forum, http://www.upnp.org.
[23] UPnP device architecture, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.
[24] Apache Software Foundation, Apache HTTP Server, http://httpd.apache.org.
[25] PHP: HyperText Preprocessor, http://www.php.net.
[26] SQLite, http://www.sqlite.org.
[27] A. Jurgelionis, F. Bellotti, A. Possani, and A. De Gloria, "Designing enjoyable entertainment products," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '08)*, pp. 1–5, Florence, Italy, April 2008.
[28] PRTG Network Monitor, http://www.paessler.com.
[29] Fraps, "Real-time video capture benchmarking," http://www.fraps.com.
[30] Ch. Schaefer, Th. Enderes, H. Ritter, and M. Zitterbart, "Subjective quality assessment for multiplayer real-time games," in *Proceedings of the 1st Workshop on Network and System Support for Games*, pp. 74–78, Braunschweig, Germany, April 2002.
[31] J. Prokkola, M. Hanski, M. Jurvansuu, and M. Immonen, "Measuring WCDMA and HSDPA delay characteristics with QoSMeT," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 492–498, Glasgow, UK, June 2007.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 970617, 8 pages
doi:10.1155/2009/970617

Research Article

Real Time Animation of Trees Based on BBSC in Computer Games

Xuefeng Ao, Zhongke Wu, and Mingquan Zhou

College of Information Science and Technology, Beijing Normal University, Beijing 10087, China

Correspondence should be addressed to Xuefeng Ao, aoxuefeng@mail.bnu.edu.cn

Received 20 January 2009; Accepted 3 April 2009

Recommended by Suiping Zhou

That researchers in the field of computer games usually find it is difficult to simulate the motion of actual 3D model trees lies in the fact that the tree model itself has very complicated structure, and many sophisticated factors need to be considered during the simulation. Though there are some works on simulating 3D tree and its motion, few of them are used in computer games due to the high demand for real-time in computer games. In this paper, an approach of animating trees in computer games based on a novel tree model representation—Ball B-Spline Curves (BBSCs) are proposed. By taking advantage of the good features of the BBSC-based model, physical simulation of the motion of leafless trees with wind blowing becomes easier and more efficient. The method can generate realistic 3D tree animation in real-time, which meets the high requirement for real time in computer games.

1. Introduction

In current computer games, plants in scenes are usually consisted of simple plane pictures positioned in four orthogonal directions. 3D models of plants are seldom used in computer games. Recently, 3D plants come forth in some computer games which make users feel more realistic. For example, in [1], a palm tree model with approximately 400 Polys was created by Amped Labs LLC for the use in the Rise of Power game. There are also some top-level visualization corporations like Interactive Data Visualization, Inc. (IDV) providing functional system like SpeedTree for modeling 3D trees and simulating simple tree animation in computer games [2].

However, in the scope of our knowledge, we do not find any research publications on discussing 3D tree motion in computer games. In fact, many researchers have made contributions in tree modeling and its motion, but none is actually applied in computer games. In tree modeling, the main methods include the followings: L-system [3], image based tree modeling [4–6] and space colonization algorithm [7]. In the aspect of tree motion simulation, there are also many works. The earliest work can be retrieved was done by Wejchert and Haumann [8]. They used four simple fluid flow including uniform, sink, source and vortex to design and control the movements of the wind. And then the animation of the leaves going with the wind is simulated by computing the movement produced by wind force from normal and tangential direction in accordance with the traditional Newton theory. Mikio Shinya created a stochastic wind area and then simulated the tree swaging in the wind based on a modal analysis method [9]. Hiromi simulated tree motions like flying and breaking in tornado in the movie "twister" in which the tornado model was constructed with the turbulence theory [10]. In Feng's study [11], a single branch is divided into several little segments, and each of these segments can be viewed as a pole which cannot deform in the axis direction. Then the position of each point on a little segment after motion can be computed by applying the deformation equations of pole. In Alkagi's work [12], level of detail (LOD) technique was employed to reduce the computational complexity, and the animation of trees in real-time was implemented. During the computation of tree motion, a single branch is divided into seven parts of cone-shaped "links" that are interconnected by six "joints". And the bending of a branch is represented by the rotation of each of its joints. In [13], William Van Haevre realized tree motion at each arbitrary moment using a goal-based motion algorithm. As for recent works, Khalid Saleem animated tree branch breaking and flying effects in a 3D interactive visualization system for hurricanes and storm surge flooding [14]. Yubo Zhang introduced a data-driven approach that synthesizes tree animations from a set of precomputed motion data [15].

However, because of the high demand for real time in computer games, most of the above work cannot be applied directly in computer games. Some can be used to animating trees in computer games like Akagi's work as extra speeding technique was employed to reduce the computational complexity, and thus real-time animation can be generated [12].

There are two main factors hampering the application of 3D tree motions in computer games. For one thing, most of the tree representations are too complicated to implement real-time animation; for another, the simulation of tree animation is a sophisticated work because many physical computations like animation aerodynamics, material mechanics, and pole kinematics are involved.

In our paper, a novel tree model based on BBSC is introduced, and the method of simulating tree motions based on this model is proposed [16]. This model combined with this method is efficient for generating real-time tree motions in computer games. In the following sections, the paper is organized as follows. In Section 2, a novel tree model based on BBSC is described in detail; in Section 3, the model for physical simulation of wind is briefly introduced; in Section 4, the simulation of the tree animation is illustrated; in Section 5, the animation effect by our method is demonstrated, and the conclusion is given.

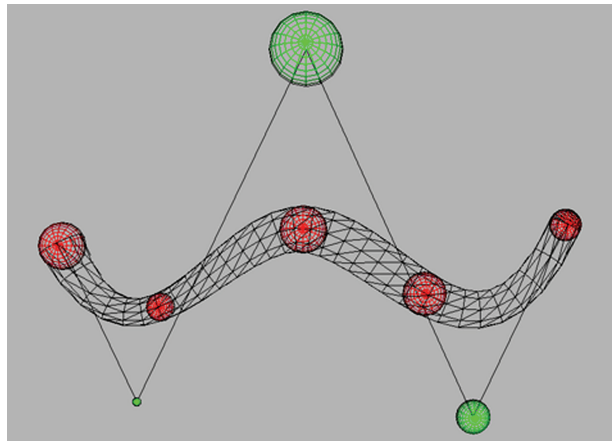
2. BBSC-Based Tree Modeling

Ball B-Spline Curve (BBSC) is a parametric solid representation of freeform tubular objects, which are skeleton-based parametric solid model. BBSC directly defines objects in B-Spline function form by using control sphere instead of control point in B-Spline curve. BBSC not only to describe every point inside 3D solid objects but also provides its center curve in B-Spline form directly. So the representation is more flexible for modeling, manipulation, and deformation.

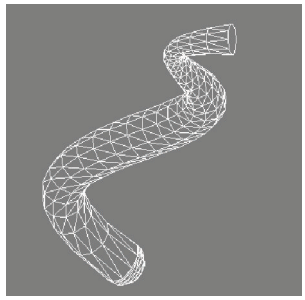
2.1. Ball B-Spline Curve (BBSC). Let $N_{i,p}(t)$ be the ith B-Spline basis of degree p with knot vector $[u_0, \ldots, u_m] = \{a, \ldots, a, u_{p+1}, \ldots, u_{m-p-1}, b, \ldots, b\}$. $\langle P_i; r_i \rangle$ is a ball centered at P_i with radius r_i.

The Ball B-Spline Curve (BBSC) is therefore defined as $\langle B \rangle(t) = \sum_{i=0}^{n} N_{i,p}(t) \langle P_i; r_i \rangle$, where P_i are control points, and r_i are control radii.

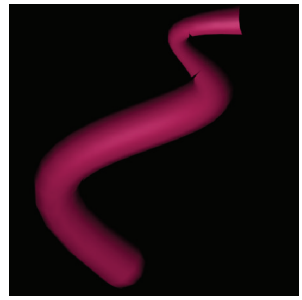
As $\langle B \rangle(t) = \sum_{i=0}^{n} N_{i,p}(t) \langle P_i; r_i \rangle = \langle \sum_{i=0}^{n} N_{i,p}(t) P_i; \sum_{i=0}^{n} N_{i,p}(t) r_i \rangle$, a Ball B-Spline curve can be regard as two parts: a 3D B-Spline curve, that is, the center curve (or skeleton): $c(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i$, and a B-Spline scalar function, that is, the radius function $r(t) = \sum_{i=0}^{n} N_{i,p}(t) r_i$. Therefore most properties and algorithms can be obtained by applying B-Spline curve and function to the two parts of BBSC, respectively. Owing to the perfect symmetry property of balls, the curve $c(t)$ constructed from the centers of balls is exactly the center curve of the 3D region represented by the BBSC. Different from BBSC, B-spline curve only represent a curve in 3D space. But BBSC inherits good properties from B-Spline curves. Most algorithms in B-Splines can be extended to BBSC. For example, we also have interpolation and approximation algorithm generating a BBSC. These algorithms are implemented by employing B-Spline curve's interpolation (approximation) algorithm to position data to get the center curve part of BBSC and B-Spline scalar function interpolation to these widths data to get radius function. Similarly we can modify the 3D shape by deforming BBSCs through modifying its control points and radii. Detailed description of the algorithm can be found in [16]. In Figure 1(a), a BBSC generated by interpolation is shown, in which the red balls are the data spheres, and the green spheres are the control spheres. In fact, the two end spheres are both data spheres and control spheres. The whole curve is tessellated so that later rendering and texture mapping processes can be implemented. In this figure we can easily see that, different from traditional B-Spline curve with 2D control points, the BBSC has the control spheres consisted of center points and radii. Figure 1(b) shows the BBSC in a different viewpoint, and Figure 1(c) is the rendering result of the transformed BBSC.



(a) The data spheres (red) and the control spheres (green)



(b) The BBSC after transformation



(c) The rendered BBSC

Figure 1: A BBSC created by interpolation.

The BBSC presented above has many features which make it very suitable to construct 3D trees in games.

(a) Solid mathematical fundamentals.

(b) Precise evaluation.

(c) Flexibility of manipulations and deformations.

(d) More compact dataset than discrete or linear representations when defining a freeform 3D object.

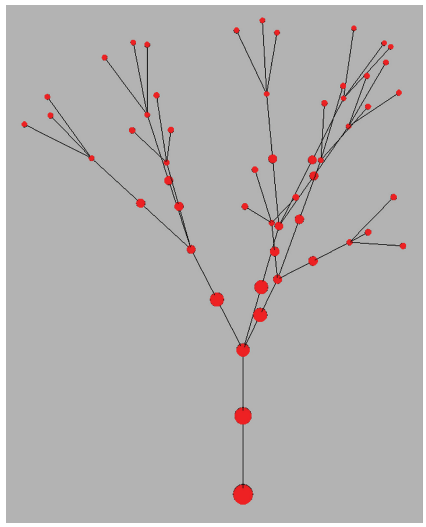


Figure 2: Geometrical model represented by BBSCs.

(a) Mesh mode (b) Rendering result

Figure 3: Tessellated BBSC-based tree.

Therefore, we can generate real-time animation of the BBSC-based trees easily.

2.2. Geometric Representation with BBSC. BBSC is a parametric representation of 3D freeform solid objects [16]. Its evaluation is precise and efficient, and it is flexible for manipulations, deformations, and morphing. These properties provide the potential to build flexible botanical tree model. Figure 2 shows the geometric relationship between these data spheres of a BBSC-based tree. The whole tree is consisted of several BBSCs which are created by interpolation. The red spheres are the data spheres used to be interpolated. Each sphere is consisted of its center point and radius. Each sphere is represented by its center point and radius. Thus a tree is described by these center points and radii of these data spheres. And in Figure 3, the resulting tree constructed from BBSCs is shown. Figure 3(a) is the tessellation result of the BBSC-based tree model, and Figure 3(b) is the rendering result. After tessellation, texture mapping technique can be applied. Therefore, various kinds of trees can be generated through texture mapping techniques in games.

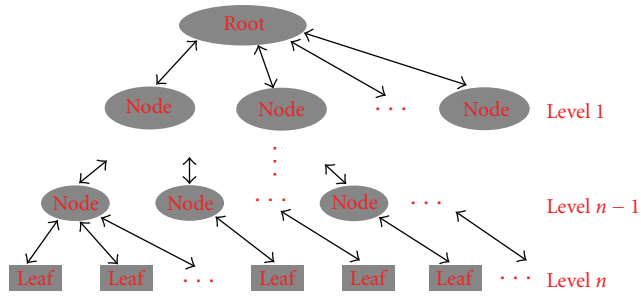


Figure 4: Topological structure of representing a tree.

2.3. Topological Representation. A graph-based data structure (tree data structure) is built to represent the complex hierarchical structures of trees shown in Figure 4. In each node of the tree data structure, the topological information of its parent and children and its geometric representation based on BBSC are stored. The construction of the topological model aims to generate real-feeling animation of the whole tree. The hierarchical structure will be made use of to compute the movements of the branches from low level to top level.

3. Model for Physical Simulation of Wind

In the wind model, we adopt Feng Jinhui's method for physical simulation of wind [11, 17]. Here, a summary of the method is given.

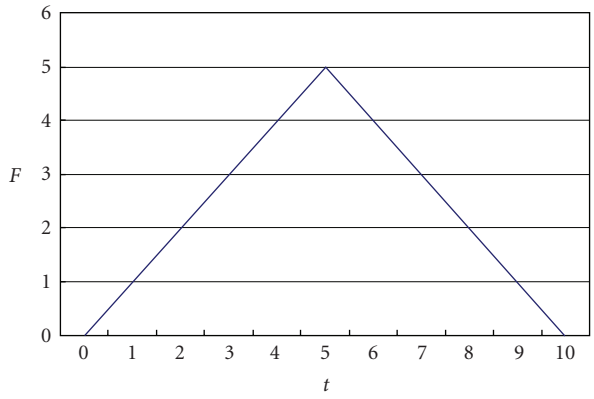
3.1. Wind Force Size. In our method, the users are allowed to choose the wind force model and set the wind direction according to their requirements. Two kinds of wind force model are provided, and arbitrary wind direction in x-z panel can be set.

3.1.1. Gust of Wind. The gust of wind increases gradually from zero to the highest point and then decreases gradually to zero again. The model can be represented in the following equation:

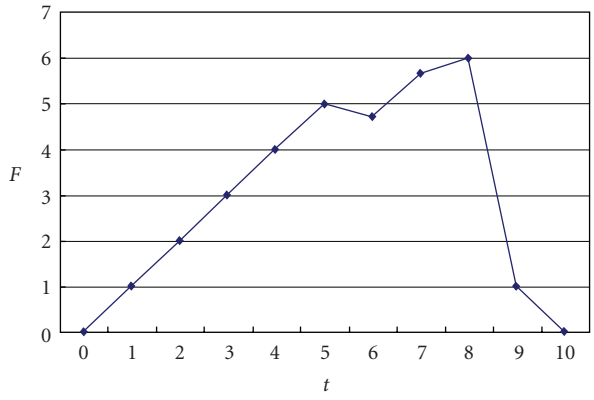
$$F_1 = \begin{cases} at + b, & 0 \le t \le t_c, \\ \dfrac{c - d(t_c - t)}{t_c}, & t_c \le t \le \text{max\_time}. \end{cases} \quad (1)$$

3.1.2. Stable Wind. The stable wind increases gradually from zero to certain grade, and for some seconds retains at this grade, then finally decreases gradually to zero. The model can be represented in the following equation:

$$F_2 = \begin{cases} at + b, & 0 \le t \le t_c, \\ c + d\sin(t), & t_c \le t \le \text{mid\_time}, \\ \dfrac{e - f(t_c - t)}{t_c}, & \text{mid\_time} \le t \le \text{max\_time}. \end{cases} \quad (2)$$



(a) The gust of wind model



(b) The stable wind model

Figure 5: The relationship between the wind force and the time.

In the above two equations, t_c is the time constant, and different wind models can be easily obtained by modifying the model parameters.

Figure 5 shows the wind force changes with the time. Figure 5(a) gives an example of the gust of wind changing with the time, and Figure 5(b) is the example of the stable wind changing with the time.

3.2. Wind Force Direction. Users are allowed to set arbitrary wind direction in x-z panel by inputting the angle between the wind direction and the x-axis positive direction. 360 wind directions along with the counter-clockwise can be obtained by increasing the angle from zero degree to 360 degree.

The later computation of the motion of the branches is based on the above wind model.

4. Animation of BBSC-Based Trees

In the introduction part, we have noticed that when computing the motion of a certain branch, those researchers generally segment a certain tree branch into several segments and then view those little segments as poles. Therefore, the deformation method of a pole can be applied to the little segment very easily to generate relatively natural-looking tree animation.

As described in Section 2, our tree model based on BBSC is a proper and efficient model in computer games. In fact, this model shows more value when computing tree motion. Now that the branches are generated from several data spheres within, we can just use the position of the data spheres to segment the current branch. And considering a branch is an actually BBSC created by interpolating several data spheres, we need only compute the position of the data spheres after motion rather than every point within the segment. The new curve obtained by interpolating the new data spheres after motion is hence regarded to be the new representation of the branch after motion. With this model, the deformation of tree branches can be computed by defining the relationship between wind forces and data spheres repositioning. And the new position of the data spheres can be computed by simulating the bending of a pole.

4.1. Dynamics Model for Branches. As shown in Figure 6(a), a pole with one end A fixed bends under the distribution force q. Then the displacement and the rotation angle can be computed for each position x. According to pole kinematics theory, the deformation of a pole can be represented by two parameters: the deflection and the rotation angle. The deflection can be just viewed as the displacement of the current position. However, in order to lessen the computation complexity, we consider only the rotation angle in our model. Furthermore, rather than each point in the current segment, we should only consider the rotation angle of the end B under force q, which can be obtained from the following equation:

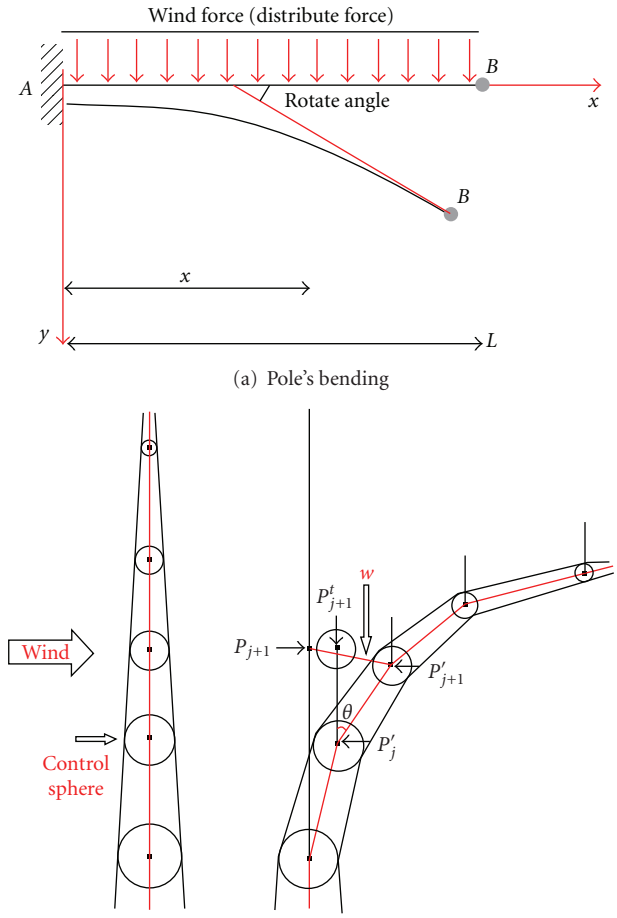
$$\theta_B = \frac{qL^3}{6EI_Z}. \tag{3}$$

In the above formula, q is the wind force, L is the length of the pole, and I_Z is the Bending Section Modulus. E is the Young's Modulus, which is used to measure the elastic characteristic of certain materials and is decided only by the physical feature of the material. We indeed have omitted many complex computation processes which are indispensable in the field of Mechanics of Materials; however it is fully accepted in computer games for realistic.

And the BBSC-based tree's bending by simulating a pole's bending is shown in Figure 6(b), in which P_{j+1} is the original coordinates of the current moving point, P'_{j+1} is the coordinates of the points after moving, P^t_{j+1} is the position of P_{j+1} after horizontal translation, and P'_j is the coordinates of the former points after moving within the current branch. The rotation angle under the wind from x-axis direction is θ. By applying (3), we get

$$\theta = \frac{F_{\text{Wind}} \cdot \text{dist}\left(P'_j, P^t_{j+1}\right)}{6EI_Z}. \tag{4}$$

In the above equation, F_{Wind} is the wind force, and $\text{dist}(P'_j, P^t_{j+1})$ is the distance between P'_j and P^t_{j+1} which represents the length of the current segment.



(a) Pole's bending

(b) BBSC model's bending

Figure 6: BBSC model's bending by simulating pole's bending.

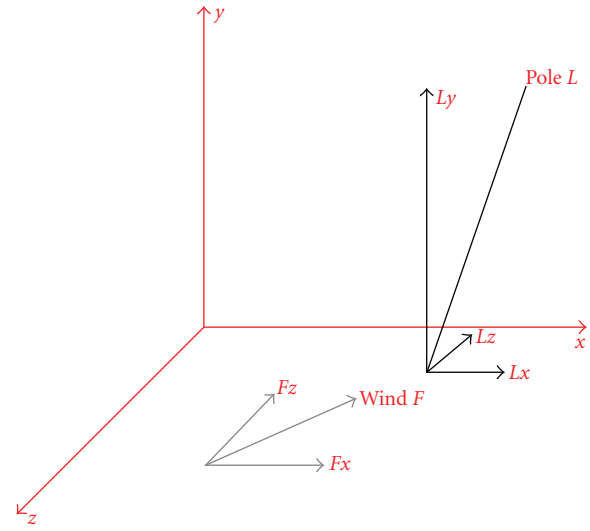


Figure 7: Wind decomposition and pole decomposition.

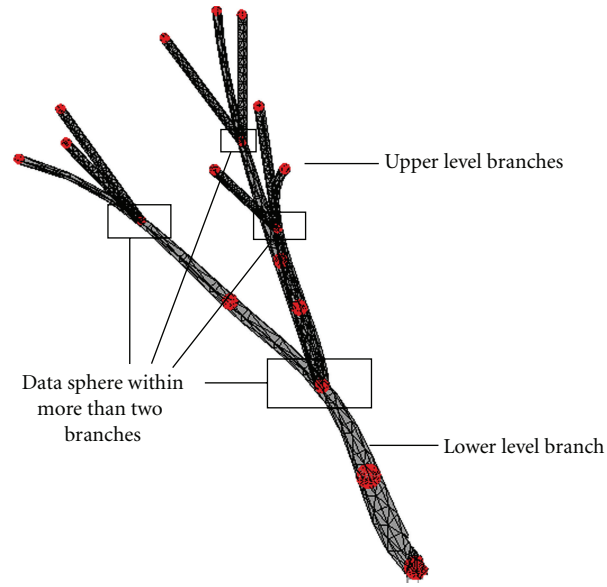


Figure 8: Data spheres contained by more than two branches.

4.2. Solution of Motion. In our model, the branches have been already divided into several segments, and each segment between two data spheres can be viewed as a pole. Considering the high request for real-time in computer games, we just apply simple deformation method of a pole to each segment. Therefore only bending is taken into account. The rotation along the cross section and the deformation along the axis direction are both neglected. The bending of a branch can then be described as three Eula angles which represent the rotation angles of the data spheres around the x-axis, y-axis, and z-axis, respectively.

Suppose the coordinate vectors of the data point before and after deformation are (x, y, z) and (x', y', z'), respectively. Then the two vectors has the relationship as in the following equation:

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = [R] \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}. \tag{5}$$

In the above equation, R is the rotation matrix, which can be described as in (6). The rotation sequence is as follows: firstly, rotates around z-axis by angle θ_z; then rotates around the rotated y-axis by angle θ_y; finally rotates around the rotated x-axis by angle θ_x:

$$[R] = \begin{bmatrix} c\theta_y c\theta_z & c\theta_y s\theta_z & -s\theta_y \\ s\theta_x s\theta_y c\theta_z - c\theta_x s\theta_z & s\theta_x s\theta_y s\theta_z + c\theta_x c\theta_z & c\theta_y s\theta_x \\ c\theta_x s\theta_y c\theta_z + s\theta_x s\theta_z & c\theta_x s\theta_y s\theta_z - s\theta_x c\theta_z & c\theta_y c\theta_x \end{bmatrix}. \tag{6}$$

In the above equation, c and s are the cosine and sine values of the related angles. And the position of any data points after deformation can be computed with this rotation matrix as long as all the angles have been figured out.

The three angles can be obtained by applying the pole deformation theory under the situation of decomposing the wind vector and the pole segment vector, respectively. The wind force vector can be decompounded into two vectors

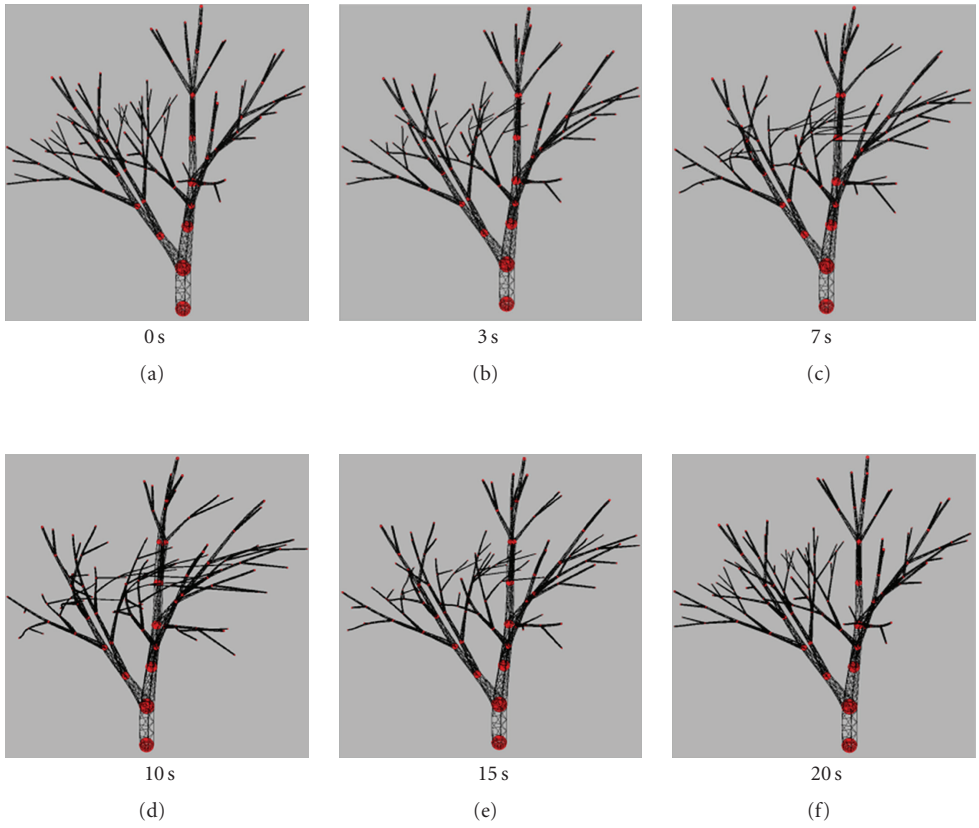


Figure 9: Six states extracted from the 20 seconds animation of a tree.

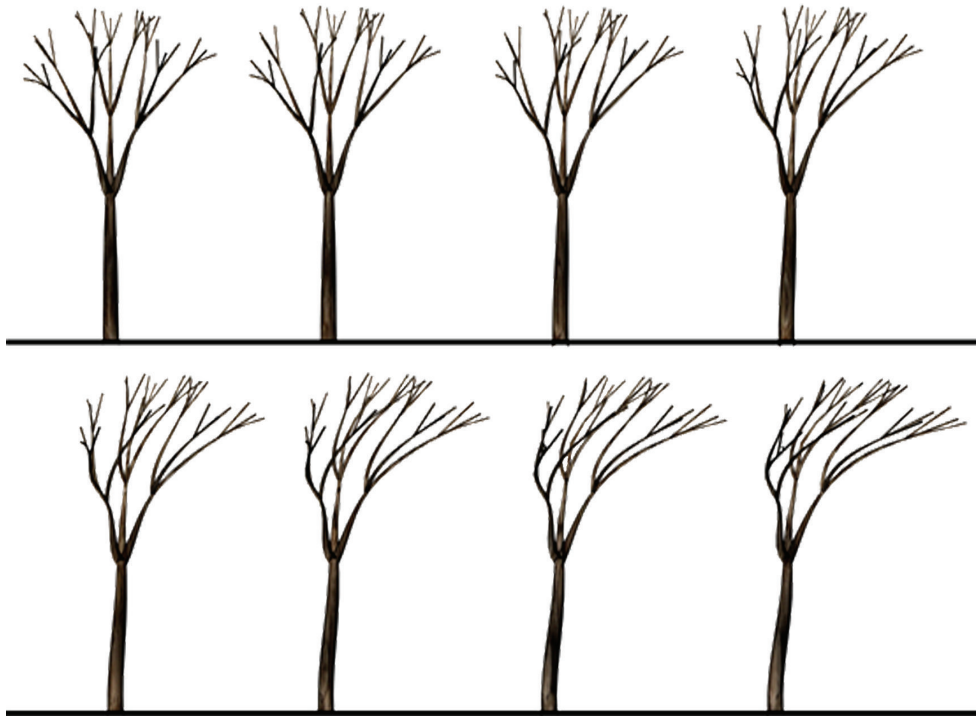


Figure 10: Eight states extracted from the 40 seconds animation of a tree.

which are along x-axis and z-axis as the wind force always lies in the x-z panel. And the vector between the start point and the end point of a segment can be decompounded into three vectors which are along x-axis, y-axis, and z-axis, respectively. The decomposition process is illustrated as in Figure 7.

Then there are four situations about the wind force acting on the segment vector.

(a) The wind force along the x-axis acting on the segment vector along the y-axis leads the segment to rotate around the z-axis by angle α_z.

(b) The wind force along the x-axis acting on the segment vector along the z-axis leads the segment to rotate around the y-axis by angle α_{y1}.

(c) The wind force along the z-axis acting on the segment vector along the x-axis leads the segment to rotate around the y-axis by angle α_{y2}.

(d) The wind force along the z-axis acting on the segment vector along the y-axis leads the segment to rotate around the x-axis by angle α_x.

In each above situation, the related angle can be computed through (4).

And finally, the rotation angle of the data points can be computed as follows:

$$\begin{aligned} \theta_x &= \alpha_x, \\ \theta_y &= \alpha_{y1} + \alpha_{y2}, \\ \theta_z &= \alpha_z. \end{aligned} \tag{7}$$

The rotation matrtix for the current data sphere can be obtained through (6). Then, the position of the data sphere after motion can be figured out by multiplying the rotation matrix to the original vector of the data sphere as (5).

4.3. Movements of the Whole Branch. In fact, the solution of motion described as above just aims to the little segment between two data spheres. And the motion of the whole branch is obtained by computing the motion of its data spheres from bottom to top one by one and then interpolating the new data spheres after motion. For a backbone branch, the initial data sphere is the root; otherwise the initial data sphere is also within another branch. The root data sphere is obviously not moving. But the motion of the data spheres contained by more than two branches should be computed carefully. If the current branch is an upper level one, then the position of the initial data sphere could just employ the position obtained in lower level branch. For example, in Figure 8, the data sphere bounded by the rectangle is contained by three branches. Then for the upper level branches, the motion of the initial data sphere doesn't need to be computed anymore as it can be obtained by employing the motion which has been computed in the lower level branch directly.

4.4. Animation of the Whole Tree. As the tree has been constructed in accordance with the hierarchical structure as in Figure 4, the motion of the whole tree can be obtained by computing the motion of the branches from root branch to the leave branch hierarchically. The depth-first traverse algorithm is employed to solve the computing sequence problem.

5. Results and Conclusions

5.1. Results. Giving related parameters and certain time t, the movements of each data sphere, furthermore each branch, and finally the whole tree under the wind force in the current moment can be gotten. And by giving a period of time, we can get the continuous animation of the tree under different wind model. In Figure 9, the states of one tree in the moment of 0 seconds, 3 seconds, 7 seconds, 10 seconds, 15 seconds, and 20 seconds during 20 seconds animation under the gust of wind from x-axis direction are shown. In Figure 10, eight states of one tree during the 40 seconds animation under the gust of wind from x-axis direction are demonstrated.

5.2. Conclusions. In this paper, an approach of generating real-time animation of trees based on a novel model—BBSC—is proposed, which can be applied in computer games. BBSC is a good representation for 3D trees and plants in particular for computer games as its solid mathematical representation and more compact dataset. Moreover, animation of trees can be generated easily as the data spheres have divided the branches into little segments automatically thus the motion of the tree can be obtained by computing the motion of each of these data spheres. By interpolating these data spheres after moving, the motion of the branch, and finally the motion of the whole tree can be implemented. The experimental results show that this method is proper and efficient for simulating tree animation in computer games.

Acknowledgment

The project is sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

References

[1] M. Mitman, "Free Palm Tree," GarageGames, http://www.garagegames.com/community.

[2] "SpeedTree," Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/SpeedTree.

[3] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer, New York, NY, USA, 1990.

[4] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," *ACM Transactions on Graphics*, vol. 26, no. 3, article 87, pp. 1–7, 2007.

[5] C.-H. Teng, Y.-S. Chen, and W.-H. Hsu, "Constructing a 3D trunk model from two images," *Graphical Models*, vol. 69, no. 1, pp. 33–56, 2007.

[6] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Transactions on Graphics*, vol. 26, no. 3, article 88, 2007.

[7] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proceedings of the*

Eurographics Workshop on Natural Phenomena (EGWNP '07), pp. 63–70, Prague, Czech Republic, September 2007.

[8] J. Wejchert and D. Haumann, "Animation aerodynamics," in *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*, pp. 19–22, New York, NY, USA, July 1991.

[9] M. Shinya and A. Fournire, "Stochastic motion—motion under the influence of wind," in *Proceedings of the Computer Graphics Forum (Eurographics '92)*, pp. 119–128, Cambridge, UK, September 1992.

[10] H. Ono, "Practical experience in the physical animation and destruction of trees," in *Proceedings of the 8th Eurographics Workshop on Computer Animation and Simulation*, D. Thalmann and M. van de Panne, Eds., pp. 149–159, Springer, Budapest, Hungary, September 1997.

[11] J. H. Feng, Y. Y. Chen, T. Yan, and E. H. Wu, "Going with wind—physically based animation of trees," *Chinese Journal of Computers*, vol. 21, no. 9, pp. 669–773, 1998.

[12] Y. Akagi and K. Kitajima, "Computer animation of swaying trees based on physical simulation," *Computers & Graphics*, vol. 30, no. 4, pp. 529–539, 2006.

[13] W. V. Haevre, F. D. Fiore, and F. V. Reeth, "Physically-based driven tree animations," in *Proceedings of the Eurographics Workshop on Natural Phenomena (EGWNP '06)*, pp. 1–8, Vienna, Austria, September 2006.

[14] K. Saleem, S.-C. Chen, and K. Zhang, "Animating tree branch breaking and flying effects for a 3D interactive visualization system for hurricanes and storm surge flooding," in *Proceedings of the 9th IEEE International Symposium on Multimedia Workshops (ISMW '07)*, pp. 335–340, Taichung, Taiwan, December 2007.

[15] L. Zhang, Y. Zhang, Z. Jiang, L. Li, W. Chen, and Q. Peng, "Precomputing data-driven tree animation," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 371–382, 2007.

[16] H. S. Seah and Z. K. Wu, "Ball B-Spline based geometric models in distributed virtual environments," in *Proceedings of the Workshop towards Semantic Virtual Environments (SVE '05)*, pp. 1–8, Villars, Switzerland, March 2005.

[17] J. H. Feng, *Going with Wind—Physically-Based Animation*, Institute of Software, Chinese Academy of Science, Beijing, China, 1999.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 730894, 7 pages
doi:10.1155/2009/730894

Research Article

Fast and Reliable Mouse Picking Using Graphics Hardware

Hanli Zhao,[1] Xiaogang Jin,[1] Jianbing Shen,[2] and Shufang Lu[1]

[1] *State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China*
[2] *School of Computer Science & Technology, Beijing Institute of Technology, Beijing 10008, China*

Correspondence should be addressed to Xiaogang Jin, jin@cad.zju.edu.cn

Received 15 December 2008; Accepted 4 March 2009

Recommended by Zhongke Wu

Mouse picking is the most commonly used intuitive operation to interact with 3D scenes in a variety of 3D graphics applications. High performance for such operation is necessary in order to provide users with fast responses. This paper proposes a fast and reliable mouse picking algorithm using graphics hardware for 3D triangular scenes. Our approach uses a multi-layer rendering algorithm to perform the picking operation in linear time complexity. The objectspace based ray-triangle intersection test is implemented in a highly parallelized geometry shader. After applying the hardware-supported occlusion queries, only a small number of objects (or sub-objects) are rendered in subsequent layers, which accelerates the picking efficiency. Experimental results demonstrate the high performance of our novel approach. Due to its simplicity, our algorithm can be easily integrated into existing real-time rendering systems.

1. Introduction

Mouse picking, as the most intuitive way to interact with 3D scenes, is ubiquitous in many interactive 3D graphics applications, such as mesh editing, geometry painting and 3D games. In many Massive Multi-player Role Playing Games (MMRPGs), for instance, thousands of players compete against each other, and the picking operation is frequently applied. Such applications require picking to be performed as fast as possible in order to respond to players with a minimum time delay. In recent years, programmable graphics hardware is getting more and more powerful. How to make full use of the co-processors in the picking operation becomes important.

The WYSIWYG method, which takes advantage of graphics hardware to rerender scene objects into an auxiliary frame buffer, was first proposed by Robin Forrest in the mid-1980s and used in 3D painting by Hanrahan and Haeberli [1]. In their method, each polygon is assigned a unique color value which is used as an identifier. Given the cursor position on the screen and the id buffer, the picked position on the surface can be found by retrieving data from the frame buffer. However, this approach has weaknesses for complex scenes in that all objects in the view frustum must be rerendered. This may take a long time for complex scenes and therefore lower the picking performance. By integrating the WYSIWYG method and hardware bilinear interpolation [2], Lander presented a method to calculate the exact intersection information, that is, the barycentric coordinate in the intersected triangle. By setting additional color values with $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ (normalized with floating-point precisions) to the three triangle vertices respectively, he calculated the barycentric coordinate by interpolation after the rasterization stage. However, the computed barycentric coordinate is in the projected screen-space but not in the object-space, which may restrict its application.

In this paper, we propose a simple, fast and reliable picking algorithm (FRMP) using graphics hardware for 3D triangular scenes. By combining the multi-layer culling approach of Govindaraju et al. [3] with a GPU-based implementation of Möller and Trumbore's ray-intersection test [4], the picking can be performed in linear time complexity. Our approach has the following features.

(1) It is fast—our approach is 2 to 14 times as fast as the traditional GPU-based picking one.

(2) It is reliable—our approach performs the operation in object-space, and the exact intersection information can be computed.

(3) It is parallel—the ray-triangle intersection detection is implemented as a geometry shader.

(4) It is simple—our novel approach operates directly on triangular meshes and can be easily integrated into existing real-time rendering systems.

The rest of the paper is organized as follows. Section 2 reviews some related work. Section 3 describes our new algorithm, whereas experimental results and discussions are presented in Section 4. We conclude the paper and suggest future work in Section 5.

2. Related Work

Intersection detection is widely used in computer graphics. The mouse picking operation can be performed by an ordinary ray-object intersection test and accelerated by lots of schemes for high efficiency.

The methods for interference detection are typically based on bounding volume data structures and hierarchical spatial decomposition techniques. They are K-d trees [5], sphere trees [6, 7], AABB trees [8, 9], K-DOPs trees [10], and OBB trees [11]. The objects (triangles) are organized in clusters promoting faster intersection detection. The spatial hierarchies are often built in the preprocessing stage and should be updated from frame-to-frame when the scene changes, which is not appropriate in most cases for mouse picking.

Hardware occlusion queries are also used in collision detection for large environments to efficiently compute all the contacts at high frame rates by Govindaraju et al. [3, 12, 13]. These GPU-based algorithms use a linear time multi-pass rendering algorithm to compute the potentially colliding set. They even achieve interactive frame rates for deformable models and breaking objects. In their method, the objects (triangles) list can be traversed from the beginning up to the end and thus no spatial organization (KD and other trees) are required. The WYSIWYG method for mouse picking, which was first proposed by Robin Forrest in the mid-1980s and used in 3D paint by Hanrahan and Haeberli [1] and further studied by Lander [2], Akenine-Möller and Haines [14], belongs to this class. Its efficiency is high in many cases. However, it has limitations as discussed in the introduction section. CPU methods for picking objects were introduced by [15] in the Direct3D platform and by [16] in the OpenGL platform. However, their efficiency decreases dramatically as the number of input primitives increases. Motivated by the multi-layer culling approach of Govindaraju et al., we do not construct a time consuming hierarchy. Instead, we use a multi-layer rendering algorithm to perform a linear time picking operation. In this paper, we perform the exact object-space-based ray-triangle intersection test [4] in a geometry shader by taking advantage of its geometric processing capability. The overall approach makes no assumptions about the object's motion and can be directly applied to all triangulated models.

Some acceleration techniques for real-time rendering need to be applied in our method. Triangle strips and view frustum culling were introduced by [17, 18], respectively. It is possible to triangulate the bounding boxes of objects as strips and to cull away objects that are positioned out of the view frustum. Hardware occlusion queries for visibility culling were studied by [19–21]. GPU-based visibility culling is also important in our algorithm.

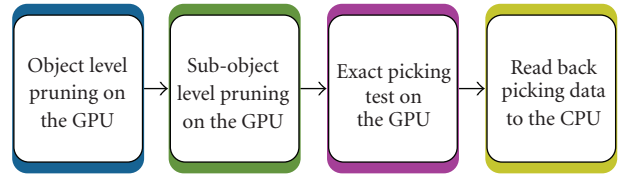


Figure 1: Algorithm workflow.

3. Hardware Accelerated Picking

Our mouse picking operation takes the screen coordinate of the cursor and the scene to be rendered as input, and outputs the intersection information, such as object id, triangle id, and even the barycentric coordinate of the intersection point. In this section, we first present an overview of our algorithm and then we discuss it in detail.

3.1. Algorithm Overview. Our FRMP method exploits the new features of the 4th generation of PC-class programmable graphics processing units [22]. Figure 1 illustrates the algorithm workflow. The overall algorithm is outlined as follows.

(1) Once the user clicks on the screen, compute the picking ray origin and direction in the view coordinate system.

(2) Set the render target with one-pixel size.

(3) Set the render states *DepthClipEnable* and *DepthEnable* to *FALSE*.

(4) After the view frustum culling, render the bounding boxes of the visible objects. We issue a boolean occlusion query for each object during this rendering pass.

(5) Render the bounding boxes of all sub-objects whose corresponding occlusion query returns *TRUE*. Again we issue a boolean occlusion query for each sub-object during this rendering pass.

(6) Reset the states *DepthClipEnable* and *DepthEnable* to *TRUE*.

(7) Render the actual triangles whose corresponding occlusion query returns *TRUE*. Now we only issue one occlusion query for all triangles.

(8) If the occlusion query returns *TRUE*, trivially read back the picking information from the one-pixel-sized render target data; otherwise, no object is picked.

The novel multi-layer rendering pass on programmable graphics shaders is outlined below:

(1) Transform the per-vertex position to the view coordinate system in the vertex shader.

(2) Perform the object-space-based ray-triangle intersection test in the geometry shader, output a point with picking information if the triangle is intersected. The x- and y-components of the intersection point are set to 0, and the z-component is assigned as the depth value of the point. Then the point is passed to the rasterization stage.

(3) Output the picking information directly in the pixel shader.

3.2. New Features in the Shader Model 4.0 Pipeline. The *Shader Model* 4.0 fully supports 32-bit floating-point data format, which meets the appropriate precision requirement for general purpose GPU computing (GPGPU). The occlusion query can return the number of pixels that pass the z-testing, or just a boolean value indicating whether or not any pixel passes the z-testing. In our case, we only need the boolean result that whether some objects are rendered or none are rendered.

The *Geometry Shader*, which is first introduced into the shader model 4.0 pipeline, takes the vertices of a single primitive (point, line segment, or triangle) as input and generates the vertices of zero or more primitives. The input and output primitive types need not match but they are fixed for the shader program. We use a triangle as the input primitive, as the ray-triangle intersection detection needs to be implemented here. We get a point as output. If the intersection test is passed, a point primitive with intersection information is returned. If the test is failed, no point is output.

3.3. Intersection Test in the Geometry Shader. In this section, we present the ray-intersection test introduced by Möller and Trumbore [4]. We implement the algorithm in a geometry shader by taking advantage of its geometric processing capability.

A ray, $\mathbf{r}(t)$, is defined by an origin point, $\mathbf{o}$, and a normalized direction vector, $\mathbf{d}$. Its mathematical formula is shown in (1):

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}. \tag{1}$$

Here the scalar, t, is a variable that is used to generate different points on the ray, where t-values of greater than zero are said to lie in front of the ray origin and so are a part of the ray and negative t-values lie behind it. Also, since the ray direction is normalized, a t-value generates a point on the ray that is t distance units away from the ray origin.

When the user clicks the mouse, the screen coordinates of the cursor are transformed through the projection matrix into a view-space ray that goes from the eye-point through the point clicked on the screen and into the screen.A point, $\mathbf{t}(u, v)$, on a triangle is given by the explicit formula (2).

$$\mathbf{t}(u, v) = (1 - u - v)\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2, \tag{2}$$

where (u, v) is the barycentric coordinate, which satisfies $u \geq 0$, $v \geq 0$ and $u + v \leq 1$. The point of intersection between the picking ray, $\mathbf{r}(t)$, and the triangle, $\mathbf{t}(u, v)$, satisfies the equation $\mathbf{r}(t) = \mathbf{t}(u, v)$, which yields:

$$\mathbf{o} + t\mathbf{d} = (1 - u - v)\mathbf{v}_0 + u\mathbf{v}_1 + v\mathbf{v}_2. \tag{3}$$

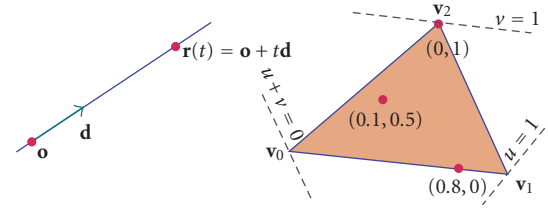


FIGURE 2: (Left) a simple ray and its parameters. (Right) barycentric coordinate for a triangle, along with some example point values.

An illustration of a ray and the barycentric coordinate for a triangle are shown in Figure 2. Denoting $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$, $\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$, and $\mathbf{s} = \mathbf{o} - \mathbf{v}_0$, the solution to (3) can be easily obtained by using Cramer's rule [23]:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\det(-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2)} \begin{pmatrix} \det(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{s}, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{e}_1, \mathbf{s}) \end{pmatrix}. \tag{4}$$

As a result, the intersection information is obtained by solving (4). As this process is independent of the triangles, we can parallelize it in graphics hardware. This equation is adapted with optimizations since the *determinant* of a matrix is an intrinsic function in the High Level Shading Language (HLSL). The intersection test is conducted in the view space and if it is passed, we output a point primitive. The x- and y-components of its position coordinate are 0 because the render target used in our algorithm is only one-pixel in size. The z-component is the depth value which is obtained by transforming the distance value into the projection space. The GPU will automatically add a primitive id as the triangle identifier in the *Input Assembler Stage*. In addition, the barycentric coordinate value (u, v) and the object id are also obtained from the picking information. The pseudo-code in the geometry shader is presented in Algorithm 1.

3.4. Multi-Layer Visibility Queries. We use a multi-layer rendering algorithm to perform linear time intersection tests, taking advantage of the 4th generation of PC-class programmable graphics processing units. The overall approach makes no assumption about the object's motion and is directly applicable to all triangulated models.

First of all, we set a 1×1 sized texture as a render target after the view frustum culling. Instead of rendering the actual triangles, we then render the bounding boxes of the visible objects. We issue a boolean occlusion query for each object during this rendering pass. As we know, the render state *DepthClipEnable* controls whether to clip primitives whose depth values are not in the range of $[0, 1]$ or not; the render state *DepthEnable* determines whether to perform the depth testing or not. After the view frustum

```
(1)  float 3 × 3 edge; float4 Picker
(2)  edge[0] = input[1] − input[0]
(3)  edge[1] = input[2] − input[0]
(4)  Picker.w = det(float3 × 3(−Ray, edge[0], edge[1]))
(5)  if Picker.w == 0 then
(6)      return
(7)  end if
(8)  if Picker.w < 0 then
(9)      Picker.w = −Picker.w
(10)     edge[2] = input[0] − float3(0, 0, 0)
(11) else
(12)     edge[2] = float3(0, 0, 0) − input[0]
(13)     Picker.x = det(float3 × 3(−Ray, edge[2], edge[1]))
(14) end if
(15) if Picker.x < 0 || Picker.x > Picker.w
(16)     return
(17) end if
(18) Picker.y = det(float3 × 3(−Ray, edge[0], edge[2]))
(19) if Picker.y < 0 || Picker.x + Picker.y > Picker.w
(20)     return
(21) end if
(22) // get the distance in the view-space
(23) Picker.z = det(float3 × 3(edge[2], edge[0], edge[1]))
(24) Picker.z = Picker.z / Picker.w ∗ PickingRay.z
(25) PICKING_GS_OUTPUT output
(26) output.Pos = float4(0, 0, Picker.z, 1)
(27) // transform the distance value into projection space
(28) output.Pos.zw = mul(output.Pos.zw,
         float2 × 2(mxProj[2].yz, mxProj[3].yz))
(29) output.Info = float4(Picker.xy/Picker.w, FacetID,
         ObjectID)
(30) outStream.Append(output)
```

Algorithm 1: Object-based intersection test.

culling, there are some objects intersected with the near-plane or the far-plane of view frustum. The depth values of some vertices may not be in the range of $[0, 1]$. In order to collect all the possible intersected objects for the next layer, we set *DepthClipEnable* and *DepthEnable* to *FALSE*. If any occlusion query is passed, the corresponding object may intersected with the picking ray and thus its actual triangles will be rendered; otherwise, it is pruned. Since a large number of objects are not intersected during this step, we can greatly reduce the rendering time compared with the WYSIWYG method, which requires us to render all the objects.

Second, we render the bounding boxes of all sub-objects whose corresponding occlusion query returns *TRUE*. Again we issue a boolean occlusion query for each sub-object during this rendering pass. Since some systems need to handle large models, which may not fit entirely into the GPU memory, we group adjacent local triangles to form a sub-object and prune the potential regions considerably as suggested in [3].

Next, the actual triangles of the unpruned sub-objects are rendered. We only issue one occlusion query for all the triangles during this step. We would like to get the exact intersection result after this step. Triangles outside the view frustum are discarded, and only the closest triangle is needed. Thus the render states *DepthClipEnable* and *DepthEnable* are reset to *TRUE*.

Lastly, if the occlusion query passes, the triangle with the minimal distance from the eye-point is picked and its intersection information can be retrieved from the 1×1 sized render target texture. This causes an additional delay while reading back data from the graphics memory to the system memory. In the WYSIWYG method, we need to lock the window-sized texture to get the picking information but this is slow when the window size is large. Actually our novel algorithm only needs to store the information in the smallest sized texture. If the occlusion query fails, we need not read the data from the render target because we know that nothing has been picked. In the WYSIWYG method, however, one cannot know if anything has been picked until one reads the corresponding data from the texture.

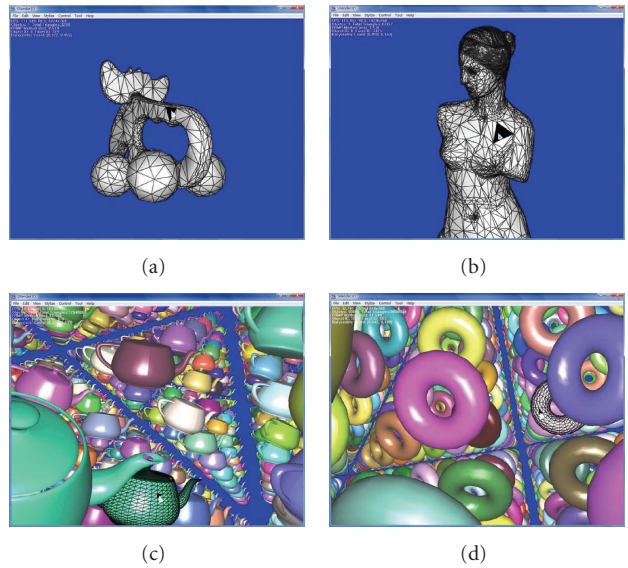


Figure 3: The four test scenes: the toy elk (upper left), Venus (upper right) the teapots (lower left) and the tori (lower right). Note that the picked objects are shown in wireframe and the picked triangles are shown in black, whereas other objects are shaded normally.

4. Experimental Results and Discussion

Our algorithm takes the screen coordinates of the cursor and the scene to be rendered as the input, and outputs intersection information, such as object id, triangle id, and even the barycentric coordinate of the intersection point. Now our algorithm can be used with platforms which support Direct3D 10 APIs. We have incorporated our FRMP method into a Direct3D 10-based scene graph library and tested it on four scenes in order to evaluate its efficiency for different scene types. All tests were conducted on a PC with a 1.83 GHz Intel Core 2 Duo 6320 CPU, 2 GB main memory, an NVIDIA Geforce 8800 GTS GPU, 320 MB graphics memory, and Windows Vista 64bit Operating System.

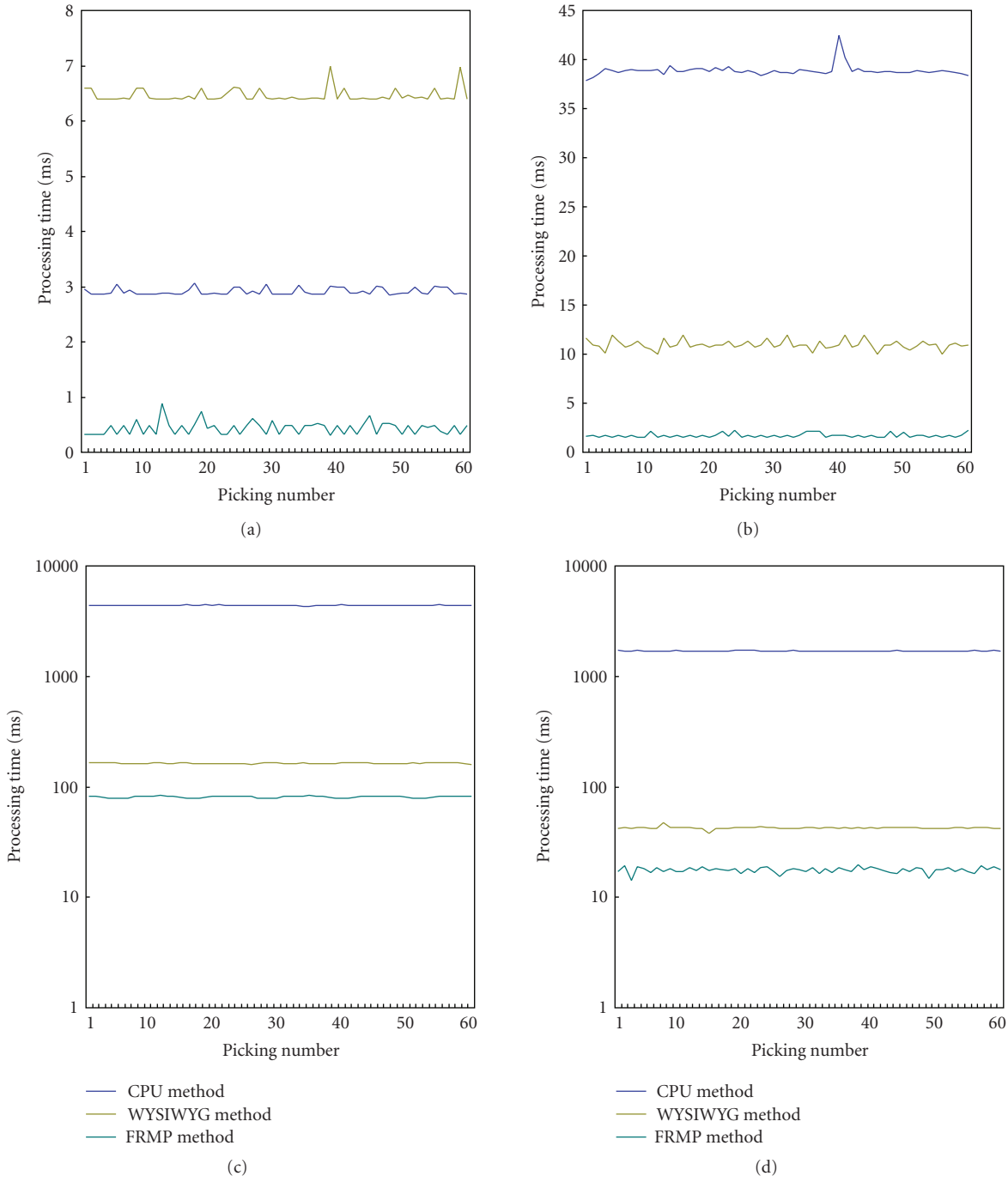


Figure 4: Processing time comparisons for the toy elk (upper left), the Venus (upper right), the teapots (lower left) and the tori (lower right). Note that the lower two scenes use a logarithmic scale to capture the high variations in processing times. Note that if no object is picked, the processing times of our method will even be faster because we picked one object on purpose to perform these tests.

4.1. The Test Scenes. The four test scenes comprise of an arrangement of a toy elk model (3290 polygons), a Venus model (43 357 polygons), 2000 randomly rotated teapots (12.64 M polygons) and 10 000 randomly rotated tori (8 M polygons), all are in resolution of 1024×768 pixels. The test scenes are depicted in Figure 3.

The toy elk scene only has 3290 triangles, while the Venus scene consists of large number of triangles. Both are simple cases to handle for the picking operation as only one object is used and is not occlusion culled. These two scenes were tested in order to evaluate the efficiencies in simple cases. Such cases may occur in mesh editing or geometry painting applications.

The teapots scene with 12.64 M triangles and the tori scene with 8 M triangles are complex cases and are designed to rotate randomly from frame-to-frame. They can offer

Table 1: Statistics for the four test scenes. The processing times are in (miliseconds).

Model name	Triangles per model	Modelnumber	Method	Longest time(miliseconds)	Shortest time(miliseconds)	Average time(milisecond)	Speedup
Toy elk	3290	1	CPU	3.064	2.852	2.910	1.000
			WYSIWYG	6.993	6.390	6.464	0.450
			FRMP	0.891	0.314	0.441	6.599
Venus	43 357	1	CPU	42.497	37.824	38.859	1.000
			WYSIWYG	11.974	10.010	10.948	3.549
			FRMP	2.249	1.521	1.702	22.831
Teapot	6320	2000	CPU	4500.598	4320.855	4387.013	1.000
			WYSIWYG	165.392	160.398	163.254	26.872
			FRMP	83.334	78.293	80.959	54.188
Torus	800	10 000	CPU	1720.887	1696.976	1706.358	1.000
			WYSIWYG	47.918	38.016	42.411	40.234
			FRMP	19.636	14.120	17.651	96.672

good occlusions as most of their objects are occluded in most instances.

4.2. Comparison of the Results. For each test scene, we report the processing times of our fast and reliable mouse picking (FRMP) algorithm in comparison to the CPU implementation of our algorithm, and to the traditional GPU method (WYSIWYG) (see Figure 4). Note that in our tests we have picked an object. Had we not done so, our algorithm would have performed even better than the competition. This is because when no bounding box intersects with the picking ray, our approach will not render the actual triangles and return *FALSE* directly.

As we can see from a number of scene statistics shown in Table 1, our method can produce a speedup of more than two as compared to the traditional WYSIWYG method. In the toy elk scene, our method was 2469 miliseconds faster than the CPU method, while the WYSIWYG method was 3554 miliseconds slower than the CPU method. That is because the whole window-sized texture data needs to be read back to the main memory to check the intersection even for small models. In the Venus scene, as the triangle number is increased, our method and the WYSIWYG method produce a speedup of 22.831 and 3.549, respectively. Even in the teapot scene and in the torus scene, our method maintained a good speedup over the WYSIWYG method. If a very large model cannot be loaded into the video memory in its entirety, then our GPU-based algorithm seems to be slower than the CPU-based approach. Fortunately such occurrences are rare in many real-time applications.

5. Conclusions and Future Work

We have presented a novel algorithm for intersection tests between a picking ray and multiple objects in an arbitrarily complex 3D environment using some new features of graphics hardware. The algorithm in this paper is fast, more reliable, parallelizable, and simple. Our algorithm is applicable to all triangulated models, making no assumptions about the input primitives and can compute the exact intersection information in object-space. Furthermore, our FRMP picking operation can achieve high efficiency as compared with traditional methods. Due to its simplicity, our algorithm can be easily integrated into existing real-time rendering applications. Our FRMP picking approach is of relevance to interactive graphics applications.The presented approach still leaves some room for improvement and for extensions. For instance, alternative acceleration techniques for real-time rendering may be applied to our FRMP method. Moreover, additional hardware features will be useful with the progress of the graphics hardware. In the future, we would like to extend and to apply our technique to the generic collision detection field.

Acknowledgments

The authors would like to thank the Cybergames '08 conference and special issue reviewers for their dedicated help in improving the paper. Many thanks also to Xiaoyan Luo, Charlie C. L. Wang, and Feifei Wei for their help and their valuable advice. The models used for the test in our paper can be downloaded from http://shapes.aim-at-shape.net/. This work was supported by the National Natural Science Foundation of China (Grant nos. 60533080 and 60833007) and the Key Technology R&D Program (Grant no. 2007BAH11B03).

References

[1] P. Hanrahan and P. Haeberli, "Direct WYSIWYG painting and texturing on 3D shapes," *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 215–223, 1990.

[2] J. Lander, "Haunted trees for halloween," *Game Developer Magazine*, vol. 7, no. 11, pp. 17–21, 2000.

[3] N. K. Govindaraju, S. Redon, M. C. Lin, and D. Manocha, "CULLIDE: interactive collision detection between complex models in large environments using graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware (HWWS '03)*, pp. 25–32, Eurographics Association, San Diego, Calif, USA, July 2003.

[4] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," in *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, Los Angeles, Calif, USA, July-August 2005.

[5] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, New York, NY, USA, 1985.

[6] I. J. Palmer and R. L. Grimsdale, "Collision detection for animation using sphere-trees," *Computer Graphics Forum*, vol. 14, no. 2, pp. 105–116, 1995.

[7] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Transactions on Graphics*, vol. 15, no. 3, pp. 179–210, 1996.

[8] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *Journal of Graphics Tools*, vol. 2, no. 4, pp. 1–13, 1997.

[9] T. Larsson and T. Akenine-Möller, "Collision detection for continuously deforming bodies," in *Proceedings of the Annual Conference of the European Association for Computer Graphics (EUROGRAPHICS '01)*, pp. 325–333, Manchester, UK, September 2001.

[10] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.

[11] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: a hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 171–180, ACM, New Orleans, La, USA, August 1996.

[12] N. K. Govindaraju, M. C. Lin, and D. Manocha, "Quick-CULLIDE: fast inter- and intra-object collision culling using graphics hardware," in *Proceedings of IEEE Virtual Reality Conference (VR '05)*, pp. 59–66, IEEE Computer Society, Bonn, Germany, March 2005.

[13] N. K. Govindaraju, M. C. Lin, and D. Manocha, "Fast and reliable collision culling using graphics hardware," in *Proceedings of the 11th ACM Symposium on Virtual Reality Software and Technology (VRST '04)*, pp. 2–9, ACM, Hong Kong, November 2004.

[14] T. Akenine-Möller and E. Haines, *Real-Time Rendering*, AK Peters, Natick, Mass, USA, 2nd edition, 2002.

[15] Microsoft Corporation, *DirectX Software Development Kit*, Microsoft Corporation, Redmond, Wass, USA, 2007.

[16] D. Shreiner, Ed., *OpenGL® 1.4 Reference Manual*, Addison Wesley Longman, Redwood City, Calif, USA, 4th edition, 2004.

[17] F. Evans, S. Skiena, and A. Varshney, "Optimizing triangle strips for fast rendering," in *Proceedings of the 7th IEEE Visualization Conference*, pp. 319–326, IEEE Computer Society Press, San Francisco, Calif, USA, October-November 1996.

[18] U. Assarsson and T. Möller, "Optimized view frustum culling algorithms for bounding boxes," *Journal of Graphics Tools*, vol. 5, no. 1, pp. 9–22, 2000.

[19] H. Zhao, X. Jin, and J. Shen, "Simple and fast terrain rendering using graphics hardware," in *Advances in Artificial Reality and Tele-Existence*, vol. 4282 of *Lecture Notes in Computer Science*, pp. 715–723, Springer, Berlin, Germany, 2006.

[20] J. Bittner and V. Havran, "Exploiting temporal and spatial coherence in hierarchical visibility algorithms," in *Proceedings of the 17th Spring Conference on Computer Graphics (SCCG '01)*, p. 156, IEEE Computer Society, Budmerice, Slovakia, April 2001.

[21] J. Bittner, M. Wimmer, H. Piringer, and W. Purgathofer, "Coherent hierarchical culling: hardware occlusion queries made useful," *Computer Graphics Forum*, vol. 23, no. 3, pp. 615–624, 2004.

[22] D. Blythe, "The direct3D 10 system," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 724–734, 2006.

[23] E. W. Weisstein, "Cramer's Rule," http://mathworld.wolfram.com/CramersRule.html.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 609350, 9 pages
doi:10.1155/2009/609350

Research Article

A Dense Point-to-Point Alignment Method for Realistic 3D Face Morphing and Animation

Yongli Hu, Mingquan Zhou, and Zhongke Wu

College of Information Science and Technology, Beijing Normal University, Beijing 100875, China

Correspondence should be addressed to Yongli Hu, hu_yongli00@sina.com

Received 29 January 2009; Accepted 13 March 2009

Recommended by Suiping Zhou

We present a new point matching method to overcome the dense point-to-point alignment of scanned 3D faces. Instead of using the rigid spatial transformation in the traditional iterative closest point (ICP) algorithm, we adopt the thin plate spline (TPS) transformation to model the deformation of different 3D faces. Because TPS is a non-rigid transformation with good smooth property, it is suitable for formulating the complex variety of human facial morphology. A closest point searching algorithm is proposed to keep one-to-one mapping, and to get good efficiency the point matching method is accelerated by a KD-tree method. Having constructed the dense point-to-point correspondence of 3D faces, we create 3D face morphing and animation by key-frames interpolation and obtain realistic results. Comparing with ICP algorithm and the optical flow method, the presented point matching method can achieve good matching accuracy and stability. The experiment results have shown that our method is efficient for dense point objects registration.

1. Introduction

Constructing alignment of 3D objects is a crucial element of data representations in computer vision and graphics. Generally the dense alignment is a point-to-point mapping from one surface onto another surface, where each point gets the correspondent point according to its inherent property, such as the points of nose tip on different 3D faces are correspondent points according to the feature of human face. However, the practices and applications of dense point correspondence have been increasing over the last years. The straightforward application of the dense alignment is to compute objects morphing and animation. More important, if the point correspondence of a class of objects has been established, it is achievable to construct a representation for these objects. The most typical and simple model is the linear combination model described in [1], where a 3D face morphable model was constructed on the aligned 3D faces, and given a facial image the 3D face can be reconstructed by a model matching procedure. The other applications, involving objects recognition based on 2D/3D images, shape retrieval, and 3D surface reconstruction in computer vision, are all relied on dense surface correspondence.

For dense 3D objects, as the complexity of model structure and the hugeness of data, it is a challenging problem to get good correspondence result, especially to high-resolution scanned 3D faces. In fact, the correspondence of different 3D faces is not a well-defined problem. When two faces are compared, only some distinct feature points, such as the tip of nose, the corner of mouth, and the center of eyes, have the clearly correspondent points, while it is difficult to define the correspondence for the points on the smooth regions, such as the cheeks and the forehead. However, even matching the distinct feature points may be a difficult problem because it involves many of the basic problems of computer vision and feature detection. To conquer the correspondence problem of dense 3D faces, we present a closest point matching method based on the thin plate spline (TPS) transformation. In this method, the source 3D face is firstly transformed onto the destination 3D face by TPS transformation, which is constructed from the interpolation on the feature points hand-placed on the source and target 3D face. Then using a revised closest point matching algorithm, the point-to-point alignment between 3D faces is obtained. We create 3D face morphing and animation from the interpolation between

the aligned 3D faces. The realistic deformation results and the experiments comparing with the related methods show that our correspondence algorithm may be an appropriate approach.

The remainder of the paper is structured as follows. In Section 2 we review some related work. In Section 3 the TPS transformation of 3D faces is described in detail. Then the point-to-point alignment is established in Section 4. In Section 5, 3D face morphing and animation are implemented, and experimental results are given. Finally this work is concluded.

2. Related Work

In the past decades, there are many methods and algorithms that are presented to solve surface alignment and dense point correspondence for different applications. All these researches fasten on two element problems about the point matching: the spatial transformation and feature correspondence searching. The former one is to find a suitable transformation for the aligning objects. These spatial transformations can be classified into rigid transformation and nonrigid transformation. The rigid transformation is generally used in the alignment of an object and itself, such as the different viewpoint scenes or the overlapped parts of the object. The nonrigid transformation, including affine transformation, spline function, and radial-based function, now is the dominant method used in the cases existing nonrigid deformation. The latter issue of point alignment generally concerns how to determine the right correspondence by the inherent features of the objects, which commonly have the forms in geometry properties, like points, lines, curves, and surfaces, or the abstract measurements, such as moment, entropy, and mutual information. There are several surveys [2–6] that have given comprehensive reviews about this subject. The following are some typical work related to our method.

One of the most popular point matching methods is the iterative closest point (ICP) algorithm proposed by Besl and McKay [7]. It iteratively searches for closest points in two surface patches and optimizes the rigid transformation to minimize the average distance of these closest points. The original ICP algorithm demands adequate prealignment and does not usually guarantee the one-to-one correspondence, as a result various improved ICP methods were proposed. Rusinkiewicz and Levoy provided good surveys over these ICP variants [8]. Although these improvements have enhanced the convergence of ICP and achieved high registration accuracy, the rigid transformation constrains its application. In many nonrigid deformation cases, ICP is not suitable, such as 3D faces.

Blanz and Vetter made dense correspondence between 3D facial scans [1, 9], taking advantage of the fact that the radial coordinate from Cyberware scans can be expressed as a height map image with the intensity representing the radius in cylinder coordinate system. They used optical flow technique to establish correspondence between texture images and height maps images, and the correspondence was refined by a bootstrapping method if large amount of the prototypic scans obtained. A 3D face representation named morphable model was constructed from the set of aligned 3D faces. Recently, they proposed a new dense 3D correspondence method [10] based on their 3D faces database. In this method, a facial feature learning strategy and automatic properties extraction algorithm were used for alignment optimization. Although their alignment has convincing results, it demands large quantities of 3D facial scans, and some 3D information will be lost when the alignment is perceived from 2D images optical flow computation.

Similarly, the notable TPS-RPM method of Chui and Rangarajan [11] attempted to incorporate TPS into the framework of ICP for point matching. A binary correspondence matrix was used in this method to record the matching relation of all points and eliminate outliers. In point matching procedure, a soft-assign and deterministic annealing optimization was implemented to compute point correspondence iteratively. Although their experiments show good results on some sparse 2D/3D point sets, the method can easily get trapped in bad local minima if the objects are not approximately aligned initially [12]. And this method is not suitable for the alignment of 3D faces with large quantity of dense points because of the limitation of the dimension of the correspondence matrix and the impracticalness of applying TPS on the whole dense point sets.

The interpolation idea in [13] is very close to our method. To synthesis facial expression from photographs, a general 3D facial model was fitted to the individual faces based on radial basis functions using 13 feature points [13]. But the general 3D facial model created by Alias—Wavefront tools—is a relative sparse model comparing with the dense 3D faces. In addition, the fitting procedure and its refinement are different from the closest point matching algorithm here.

There are other researches associated with surface or dense point correspondence, but the applications are various. The medical image registration may be the dominant domain, others applications include 3D objects reconstruction, representation, and recognition. To get good correspondence results, many approaches require large training data. But we focus on the dense point correspondence of 3D faces and its application on 3D face morphing and animation which require only two objects.

3. 3D Face Deformation Based on Thin Plate Spline

To get more accurate point matching result, the prototypic objects are generally transformed into a reference before alignment. There are rigid transformation, affine transformation, and nonaffine deformation. As the 3D faces have complex shape feature, it is difficult to find a rigid or affine transformation with good deformation results. The nonaffine transformation is considered as the proper mapping method. For the scanned 3D faces with high dimensional dense points, the data is too large to do a global transformation for all points. The alternative solution is to use subsampling sparse point sets. Here we use an

Figure 1: The landmarks placed on the 3D faces for TPS transformation using an interactive tool.

interactive tool to pick out 25 landmarks on the aligning 3D faces. Figure 1 shows the landmarks on the 3D faces. These landmarks are the main feature points that refer to the morphological properties of human face, and will be used as the controlling points to constraint the TPS deformation between 3D faces in our method.

It is frequent in spline theory to generate a smoothly interpolated mapping between two sets of landmark points. We adopt TPS to model the deformation of 3D faces. TPS was introduced by Harder and Desmarais [14], and Bookstein [15] firstly used TPS for medical image registration. TPS is a class of nonrigid spline mapping functions with desirable properties, such as globally smooth, and easily computable, and the most important is that TPS transformation can be separated into affine and nonaffine components. So TPS has been widely used in 2D image or 3D data registration for variety applications. The following gives the implementation of TPS transformation for 3D faces in detail.

The TPS transformation can be regard as a mapping from space R^3 to R^3, so we denote TPS as $f : R^3 \mapsto R^3$. For the convenience of explication, we use F_1, F_2 that denote the source 3D face and destination 3D face for aligning. F_1, F_2 can be looked as two point sets hat have the following expression:

$$\begin{aligned} F_1 &= \{P_{1i}P_{1i} = (x_{1i}, y_{1i}, z_{1i}),\ i = 1, \cdots, N_1\}, \\ F_2 &= \left\{P_{2j} \mid P_{2j} = \left(x_{2j}, y_{2j}, z_{2j}\right),\ j = 1, \cdots, N_2\right\}, \end{aligned} \tag{1}$$

where N_1 and N_2 are the points number of F_1 and F_2 such that $N_1 \leq N_2$. The landmark points sets of F_1 and F_2 are denoted as

$$\begin{aligned} M_1 &= \left\{L_{1j} \mid L_{1j} = \left(x^*_{1j}, y^*_{1j}, z^*_{1j}\right),\ j = 1, \cdots, M\right\} \\ M_2 &= \left\{L_{2j} \mid L_{2j} = \left(x^*_{2j}, y^*_{2j}, z^*_{2j}\right),\ j = 1, \cdots, M\right\}, \end{aligned} \tag{2}$$

where M is the count of landmarks (here $M = 25$). These landmarks are the controlling points for TPS transformation, that is, TPS satisfies the following interpolation conditions at the landmark points:

$$f\left(L_{1j}\right) = L_{2j}, \quad j = 1, \ldots, M. \tag{3}$$

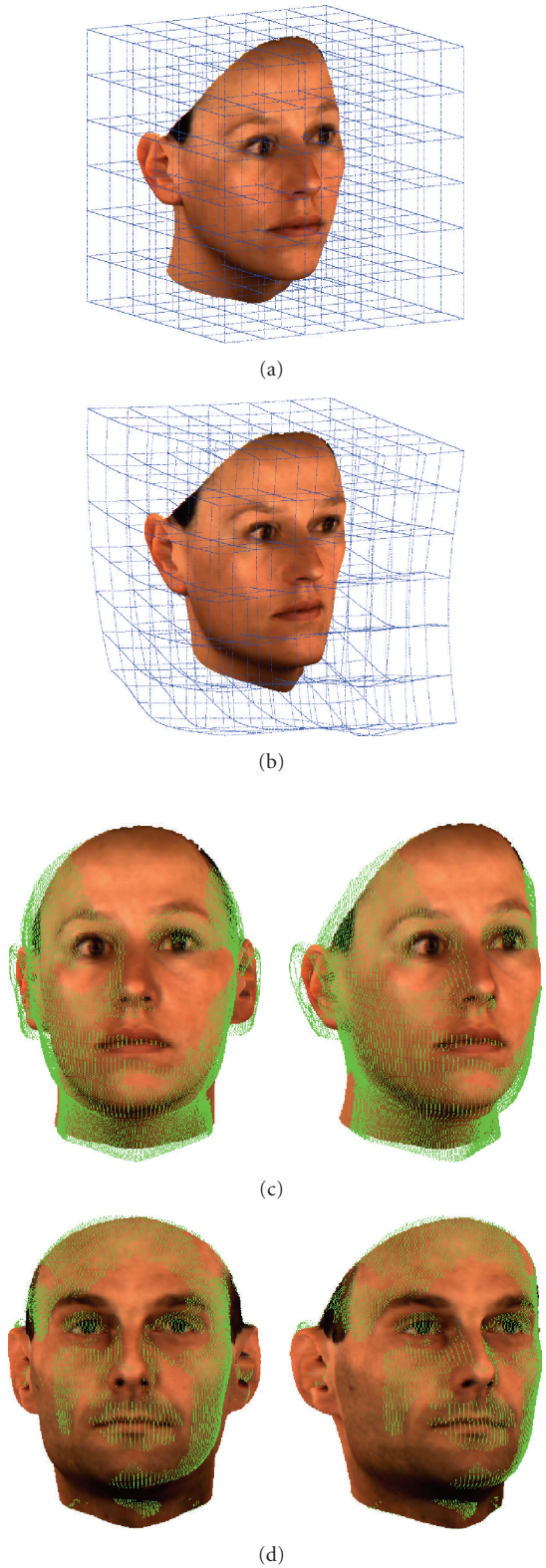


Figure 2: The TPS deformation of the source 3D face. The top one is the source 3D face with a standard partitioned cube. The second is the source 3D face deformed by TPS, and the distorted cube shows the spatial deformation of TPS. The third two are the images of the source 3D face comparing with the deformed source 3D face which displays as the sparse mesh. The bottom two are the images of the destination 3D face comparing with the deformed source 3D face.

At the same time, TPS is restricted by the blend smooth constraint, formed by the minimization of the following blending energy function, the sum of squares of all second-order partial derivatives:

$$E(f) = \iiint_{R^3} \left[\left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + \left(\frac{\partial^2 f}{\partial z^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial xy}\right)^2 + 2\left(\frac{\partial^2 f}{\partial xz}\right)^2 + 2\left(\frac{\partial^2 f}{\partial yz}\right)^2 \right] dx\, dy\, dz. \tag{4}$$

It is proved that TPS can be decomposed by affine component and nonaffine component [15]. This fact is generally represented as the following formula:

$$f(P) = Pd + Kw, \tag{5}$$

where P is the point on the source 3D face F_1 and has the homogeneous coordinates $(1, x, y, z)$. d is a 4×4 affine transformation matrix. K named TPS kernel is an $1 \times M$ vector with the form $K = (K_1(P), \ldots, K_M(P))$ such that $K_j(P) = \|P - L_{1j}\|$, $j = 1, \ldots, M$. w is an $M \times 4$ warping coefficient matrix representing the nonaffine deformation.

To get TPS transformation, the matrices d and w must be determined. There are two solutions to this problem, the interpolating and noninterpolating methods. If TPS needs not be interpolated, that is, formula (3) is not strictly satisfied, the following energy function can be minimized to find the optimal answer:

$$E'(\lambda, w, d) = \frac{1}{M} \sum_{j=1}^{M} \left\| L_{2j} - f\left(L_{1j}\right) \right\| + \lambda \cdot E(f), \tag{6}$$

where λ is the weight to control the smooth component, and for a fixed λ there will be a unique minimum for the energy function.

In the interpolating case, formula (3) is satisfied, putting (5) into (3), and confining w to nonaffine transformation, that is, $M_1'^T w = 0$, it leads a direct solution for d and w formed by the following matrix relation:

$$\begin{bmatrix} w \\ d \end{bmatrix} = \begin{bmatrix} K' & M_1' \\ M_1'^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} M_2' \\ 0 \end{bmatrix}, \tag{7}$$

where M_1' and M_2' are $M \times 4$ matrix whose rows are the homogeneous coordinates of the landmark points belonging to M_1 and M_2, respectively. K' is an $M \times M$ symmetry matrix which represents the spatial relation between the landmark points of the source 3D face and hasthe element k_{ij} with the following formation:

$$k_{ij} = \left\| L_i - L_j \right\|, \quad i = 1, \ldots, M, \ j = 1, \ldots, M. \tag{8}$$

In our work, the landmarks placed on the source and target 3D faces are looked as the correspondent points with the same facial feature, hence the condition in (3) will be satisfied, and the interpolating method is adopted here to solve the TPS transformation. From (7) the matrices d and w will be determined, and the source 3D face F_1 will be deformed by TPS transformation, we denote the deformed 3D face of F_1 as F_1'. Figure 2 shows the TPS deformation of the source 3D face and the deformed 3D face is compared with the source 3D face and the destination 3D face. It is proved that the deformed source 3D face is closer to the destination 3D face than the source 3D face, so it leads a more accurate points alignment. In the next section, the point-to-point correspondence between F_1' and F_2 will be done by a closest point matching process.

4. Dense Point Alignment by Closest Point Matching

Although the rigid transformation of ICP algorithm is not used in our method, we adopt the similar closest point matching schemes like ICP. That is, for each point on the deformed source 3D face F_1', the closest point will be found on the destination 3D face F_2. Before the closest point matching, the closest point criterion must be defined. ICP algorithm generally uses the distance between points or the distance between point and point set to define the closest point, and the distance refers to Euclidean distance. Here we define the closet point in the sense of the distance from a point to a point set. To the point P_{1i}' on F_1', the correspondent point P_{2j} on F_2 is determined by the following minimum requirement:

$$P_{2j}' = \min_{j=1,\ldots,N_2} \mathrm{DIS}\left(P_{1i}', P_{2j}\right), \tag{9}$$

where DIS(,) is a function defined to compute the distance between two points. As the deformation among 3D faces is a type of nonrigid transformation, the Euclidean distance used to determine the closest points in rigid transformation is not the proper method in nonrigid situation. Considering the modality of human face, the curvature is an important property interrelated to the local surface feature. Here the distance is defined as a weighted combination of Euclidean distance and the difference of the mean curvature of the points. The distance $\mathrm{DIS}(P_1, P_2)$ of points P_1, P_2 has the following formation:

$$\mathrm{DIS}(P_1, P_2) = \delta \cdot \|P_1 - P_2\| + (1 - \delta) \cdot |(MC(P_1) - MC(P_2))|, \tag{10}$$

where δ is the weight to balance the Euclidean distance and the curvature difference such that $0 \le \delta \le 1$. In the following experiments we set $\delta = 0.5$. $MC(\cdot)$ is the function to compute the mean curvature of the points on 3D faces.

Having determined the closest point matching criterion, for each point on F_1', the closest point searching must be executed on the target 3D face F_2. As the huge data of the source and target 3D faces, the whole closest points searching is a very time consuming procedure with computation $O(N_1 \times N_2)$. To get high point matching efficiency, we adopt the K dimensional binary search tree (KD-tree) technique in the point matching method. The KD-tree algorithm was introduced by Bentley [16] and has been widely utilized in the nearest neighbor searching [17]. It is a binary search

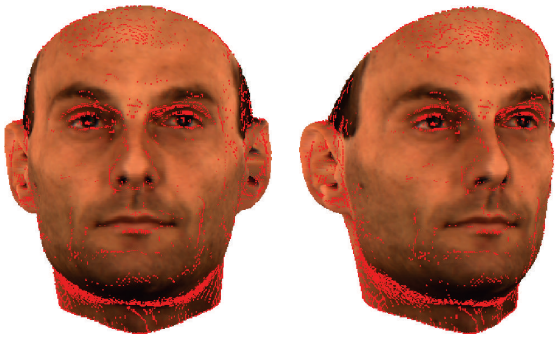


Figure 3: The collision points on the destination 3D face.

tree in which each node represents a partition of the k dimensional space. The root node represents the entire space, and the leaf nodes represent subspaces containing mutually exclusive small subsets of the relevant points. The space partitioning is carried out in a recursive binary fashion. The average performance of the KD-tree searching has complexity of $O(N_1 \times \log N_2)$.

The other obstacle has to be settled for the closest point matching is that the current method does not preserve one-to-one mapping. In fact, some points on the deformed 3D face F_1' may be mapped onto the same point on the destination 3D face F_2. We denote these points on F_2 as collision points which have more than one correspondent points on F_1'. Generally the collision points are produced by the points of outliers or the points with local complex geometry feature. Considering the high resolution of 3D faces and the distribution of these collusion points, the latter one is concerned with the main problem. The distribution of these collision points on the destination 3D face F_2 is shown in Figure 3. To eliminate these collision points, a revised point matching algorithm is proposed. The main idea of the method is to construct a distance list for every collision point, and only the point with minimum distance is regarded as the truly correspondent point. The following is the outline of the one-to-one point matching algorithm.

(1) Create KD-tree for the destination 3D face F_2.

(2) For each point on the deformed source 3D face F_1', search its closest point on F_2.

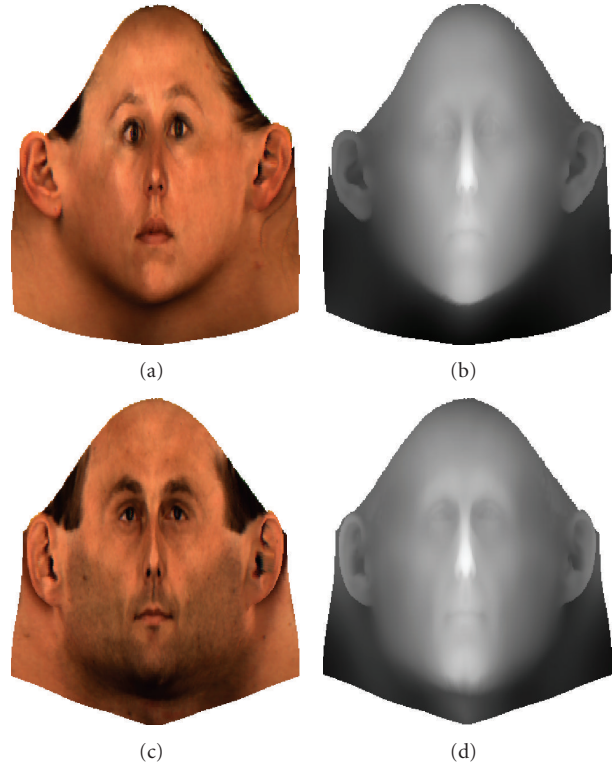
(3) Detect the collision points on F_2, if not exist, go to 6.

(4) For each collision point P_{2j}, find the correspondent points on F_1' reversely, denote the point with minimum distance as P_{1i}', and record the correspondent pair points (P_{1i}', P_{2j}).

(5) Remove the point P_{1i}' from F_1', delete the node P_{2j} from the KD-tree, then go to (2)

(6) Record the remained correspondent pairs of points without collision.

By the revised closest point matching algorithm, the correspondent point searching procedure maintains one-to-one mapping, though more computation is required.



(a) (b) (c) (d)

Figure 4: The texture and height mapping images for optical flow computation. The top two are the texture and height mapping images of the source 3D face. The bottom two are the texture and height mapping images of the destination 3D face.

5. Experimental Results of 3D Face Morphing and Animation

If the point-to-point correspondence of 3D faces is established, the direct application of the alignment is to create 3D face morphing and animation, which have wide applications in computer game, virtual reality, and animating actor in entertainment movies.

The scanned 3D faces we used come from MPI Face database [18] and BJUT-3D Face Database [19]. As the 3D facial scans have high resolution, which generally have more than 70 000 vertices and 140 000 triangles with texture information, the realistic animation results will be achieved if accurate point correspondence is obtained. Here we use the simple key-frames interpolation method to produce the face morphing and animation between the source and destination faces. The points on the key-frames 3D face are computed by linear interpolation between the correspondent points. The texture and the geometry normal of the correspondent points are interpolated at the same time.

The experiment of face morphing is implemented on two 3D faces selected from MPI face database, one face is female and the other is male. As the difference of the two faces is adequate to express variety of the human face modality, the nonrigid transformation is demanded to do with the deformation. The face animation is created on the same person's 3D faces with different expressions selected from

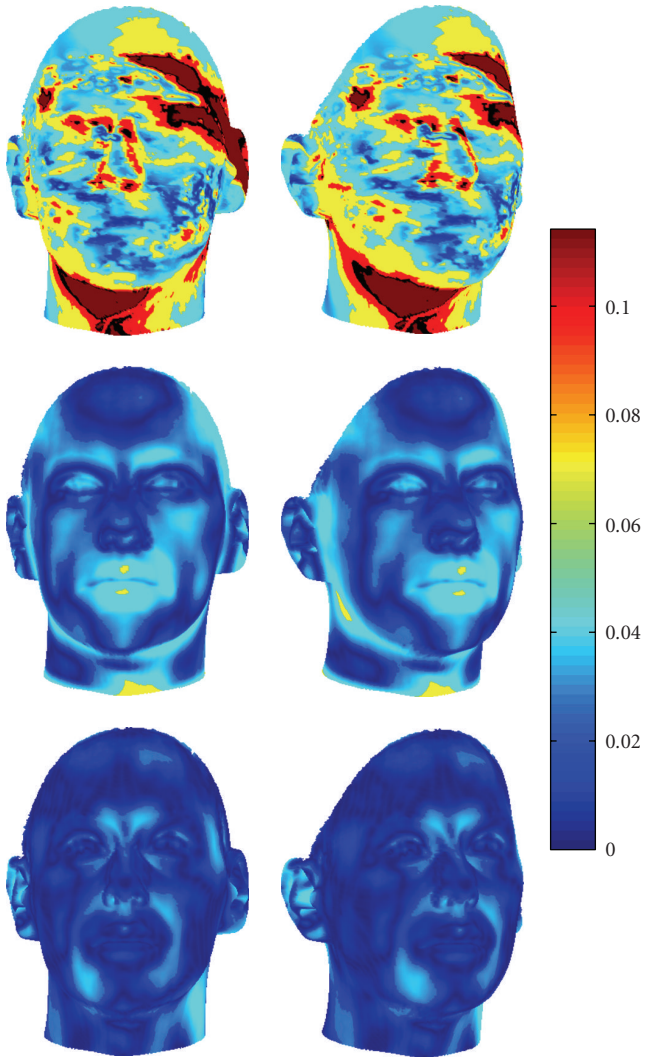


Figure 5: The distances of the correspondent points are visualized as colors on the source 3D face. The color of each point represents the distance from the point to its correspondent point with the color-mapping on the right. The top two are the results of the optical flow method. The middle two are the results of the ICP algorithm with rigid transformation. The bottom two are the results of the TPS method.

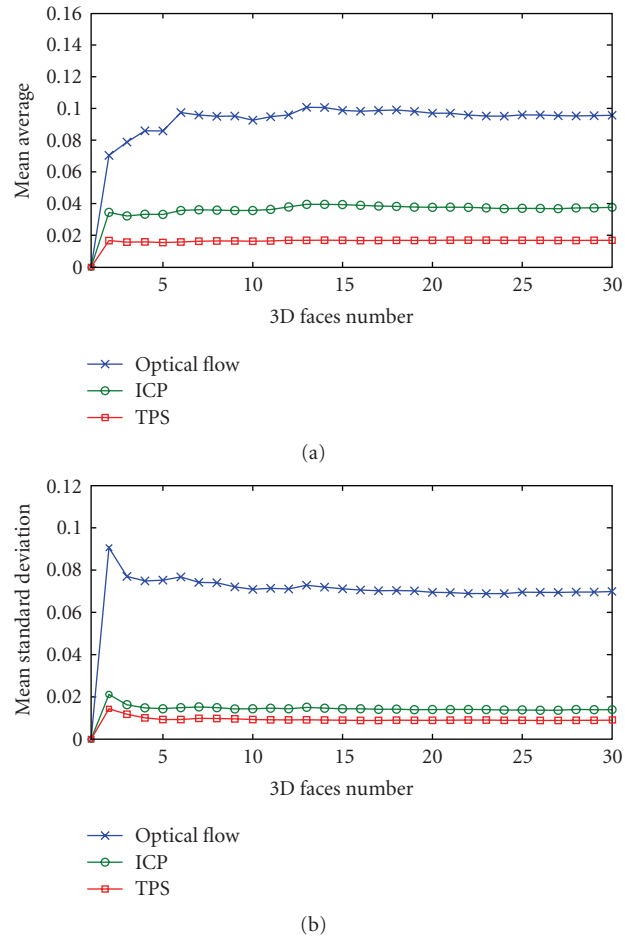


Figure 6: The trend of the mean average and standard deviation of the distances between the correspondent points of the 3D faces in the aligning set with 3D faces number increasing. The top is the mean average of the distances. The bottom is the mean standard deviation of the distances.

BJUT-3D Face Database. The sequence of key-frames of the face morphing and face animation is shown in Figure 7. On the whole, the vision reality of the morphing and animation is satisfied, though the local areas with relative complex shape feature and the areas with missing points as the scanning reason are not looking good, such as the areas of mandible and ears.

To compare our TPS method with the original ICP algorithm [7] and the optical flow method [9], the MPI source 3D face is aligned to the target 3D face using these three methods, respectively. To compute the point correspondence by the optical flow method, the source and target 3D faces are spread into texture and height mapping images (shown in Figure 4) by cylinder coordinate transformation. Then the facial texture and height mapping images are aligned by an optical flow algorithm, here we adopt the optical flow algorithm proposed by Horn and Schunck [20]. Finally the point correspondence of 3D faces is obtained from the alignment of 2D images by the reversed cylinder coordinate transformation. In ICP and TPS methods, the source 3D face is transformed by rigid transformation and TPS deformation, respectively. Then using the proposed closest point searching method, the two transformed faces are aligned with the destination 3D face. To evaluate the alignment results of these three methods, the average and standard deviations of the distances between the correspondent points on the source and destination 3D face are computed respectively.

The results of these three methods are shown in Table 1. It is denoted that all the vertices of the 3D faces are standardized into $[0, 1]$ interval before the experiment. The distances of correspondent points of these three methods are also visualized on the source 3D face (shown in Figure 5). The average and standard deviations of the distances and

Figure 7: The sequence of 3D face morphing and animation. The left column and the right column are the source 3D faces and the destination 3D faces. The top two rows are the middle frames morphing a female 3D face to a male 3D face selected from MPI 3D face database. The middle two rows are the animation sequence of a person from the neutral state to an aspiratory action state. The bottom two rows are the animation of a person from the neutral expression to smile expression. The 3D faces in the bottom four rows are selected from BJUT-3D Face Database.

its visualization in Figure 5 reveal that the TPS method has the best point matching accuracy, while the optical flow method performs poorly in dense points alignment, and the ICP is in-between of the former two methods. The optical flow is generally used in perception of the movement of objects in video sequence [21]. When the difference between the facial images is too large to satisfy the continualness requirement of adjoining frame images, the optical flow

Table 1: The average and standard deviation of the distances between the correspondent points on the source 3D face and destination 3D face.

	Optical flow	ICP	TPS
Average of the distances	0.05683	0.01673	0.00804
Standard deviation of the distances	0.03840	0.01069	0.00637

computation will fail with obvious error. It is the main reason for referring to the poor results of the optical flow method. In fact, the nonrigid transformation is more suitable for 3D faces deformation than rigid transformation, so that the TPS method has the better results than ICP algorithm.

To examine the stability of the TPS method, we selected 30 3D faces from BJUT-3D Face Database as an aligning set. The dense point alignment is implemented on the aligning set using the above three methods. The experiment is done with the 3D faces number of the aligning set increasing, that is, the 3D faces are added into the aligning set gradually. At first, the aligning set composes of two 3D faces, then 3D faces are added into one by one, until all 30 face are added. At the same time, the mean average and standard deviations of the correspondent points distances of the 3D faces in the aligning set are computed. Figure 6 shows the change of the mean average and standard deviations with the increasing of 3D faces number respected to the optical flow method, ICP algorithm and TPS method. The experimental results show that the mean average distance and its standard deviations of these three methods are all converging toward a stable value, and TPS method has better stability and correspondence accuracy than the ICP algorithm and the optical flow method.

6. Conclusion

In this paper, we describe a new dense point-to-point alignment method and apply it on scanned 3D faces. In the method, TPS is adopted to model the deformation of 3D faces, and a closest point matching algorithm is proposed to search the correspondent points and simultaneously guarantees the alignment one-to-one mapping. To reduce the closest points searching time and get good point matching accuracy, a KD-tree technique and a user-defined distance function which considers the points local curvature are integrated with the point matching algorithm. The dense point alignment is used in 3D faces morphing and animation by key-frames interpolation and gets satisfied realistic visual results. Contrasting with ICP algorithm and the optical flow method, the error analysis on the selected pair of MPI 3D faces and the experiment on 30 BJUT 3D faces prove that our method is efficient for dense point correspondence. Furthermore, the method does not require large facial database and can easily extend to other dense objects.

In our work, the landmarks of 3D faces are picked up by an interactive tool, though the manual marking procedure is simple, and taking little time, it limits the method apply in many areas, such as realtime application and the large quantity of objects situation. So the future work firstly focus on the fully automatic point matching algorithm. The intuitively thought is to find the suitable automatic feature detection method, but it is another challenging problem in pattern recognition and computer vision. The additional points to be improved of this work include refining the aligning accuracy by exploring proper representation of the local geometry feature, constructing the whole head model with hair to get more natural looking, and making practical applications.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 60736008 and no. 60872127) and the Postdoctoral Science Foundation of China (Grant no. 20080430316). The 3D facial scans were provided by the Max-Planck Institute for Biological Cybernetics in Tuebingen, Germany and the Multimedia and Intelligent Software Technology Beijing Municipal Key Laboratory of Beijing University of Technology in Beijing, China.

References

[1] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pp. 187–194, Los Angeles, Calif, USA, August 1999.

[2] L. G. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.

[3] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998.

[4] M. A. Audette, F. P. Ferrie, and T. M. Peters, "An algorithmic overview of surface registration techniques for medical imaging," *Medical Image Analysis*, vol. 4, no. 3, pp. 201–217, 2000.

[5] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[6] R. Wan and M. Li, "An overview of medical image registration," in *Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '03)*, p. 385, Xi'an, China, September 2003.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling*, pp. 145–152, Quebec, Canada, May-June 2001.

[9] T. Vetter and V. Blanz, "Estimating coloured 3D face models from single images: an example based approach," in *Proceedings of the 5th European Conference on Computer Vision (ECCV '98)*, vol. 2, pp. 499–513, Freiburg, Germany, June 1998.

[10] F. Steinke, B. Schölkopf, and V. Blanz, "Learning dense 3D correspondence," in *Advances in Neural Information Processing Systems 19*, pp. 1313–1320, MIT Press, Cambridge, Mass, USA, 2007.

[11] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[12] V. Jain and H. Zhang, "Robust 3D shape correspondence in the spectral domain," in *Proceedings of IEEE International Conference on Shape Modeling and Applications (SMI '06)*, pp. 118–129, Matsushima, Japan, June 2006.

[13] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pp. 75–84, Orlando, Fla, USA, July 1998.

[14] R. L. Harder and R. N. Desmarais, "Interpolation using surface splines," *Journal of Aircraft*, vol. 9, no. 2, pp. 189–191, 1972.

[15] F. L. Bookstein, "Principal warps: thin-plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1992.

[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[17] M. Greenspan and M. Yurick, "Approximate K-D tree search for efficient ICP," in *Proceedings of the 4th International Conference on 3-D Digital Imaging and Modeling (3DIM '03)*, pp. 442–448, Banff, Canada, October 2003.

[18] N. F. Troje and H. H. Bülthoff, "Face recognition under varying poses: the role of texture and shape," *Vision Research*, vol. 36, no. 12, pp. 1761–1771, 1996.

[19] Y. Hu, B. Yin, Y. Sun, and S. Cheng, "3D face animation based on morphable model," *Journal of Information and Computational Science*, vol. 2, no. 1, pp. 35–39, 2005.

[20] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[21] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 726496, 16 pages
doi:10.1155/2009/726496

Research Article

Gamer's Facial Cloning for Online Interactive Games

Abdul Sattar,[1] Nicolas Stoiber,[2] Renaud Seguier,[1] and Gaspard Breton[2]

[1] *SUPELEC/IETR, SCEE, Avenue de la Boulaie, 35576 Cesson-Sevigne, France*
[2] *Orange Labs, RD/TECH, 4 rue du Clos Courtel, 35510 Cesson-Sevigne, France*

Correspondence should be addressed to Abdul Sattar, abdul.sattar@supelec.fr

Received 30 January 2009; Revised 3 June 2009; Accepted 11 July 2009

Recommended by Zhongke Wu

Virtual illustration of a human face is essential to enhance the mutual interaction in a cyber community. In this paper we propose a solution to solve two bottlenecks in facial analysis and synthesis for an interactive system of human face cloning for non-expert users of computer games. Tactical maneuvers of the gamer make single camera acquisition system unsuitable to analyze and track the face due to its large lateral movements. For an improved facial analysis system, we propose to acquire the facial images from multiple cameras and analyze them by multiobjective 2.5D Active Appearance Model (MOAAM). Facial morphological dissimilarities between a human face and an avatar make the facial synthesis quite complex. To successfully clone or retarget the gamer facial expressions and gestures on to an avatar, we introduce a simple mathematical link between their appearances. Results obtained validate the efficiency, accuracy and robustness achieved.

1. Introduction

Over the last decade computer games have became more and more an interactive entertainment. Virtual representation of a character has gained the interest of both gamers and researchers. Gamers do not want to sit and play the games, instead they need to get involved in the game to an extent to visualize opponent's face and interact with him virtually. The use of virtual representation of a human face in game consoles or creating avatars has been tremendously increasing. In addition, a growing number of websites now host virtual characters technologies to deliver their contents in a more natural and friendly manner. Gestures and features (e.g., eyes, nose, mouth and eyebrows) of a human face are actually the reflection of a person's inner emotional state and personality. They are also believed to play an important role in social interactions, as they give clues to a gamer's state of mind and therefore help the communication partner to sense the tone of a speech, or the meaning of a particular behavior. For these reasons, they can be identified as an essential nonverbal communication channel in game consoles.

To track, analyze and synthesize gamer's face efficiently and to ensure the interaction of a gamer, system needs to overcome two bottlenecks in facial analysis and synthesis. Facial analysis deals with the face alignment, pose, features, gestures and emotions extractions. Excitements caused by the tactical moves of a game, compel the gamer to move around in various directions. These maneuvers produce large lateral movements of a face, which makes it difficult for a facial analysis system to track and analyze the face. For a facial synthesis system, cloning or retargeting the features, emotions and orientation of a human face on to an avatar is again one of the challenging tasks. Cloning or retargeting is difficult due to the facial morphological differences between a real face and an avatar. Furthermore, large and complex face deformations due to the expressions made by a nonrigid human face makes the online system computationally complex to clone or replicate it on to an avatar.

We propose a robust and efficient gamer's online cloning interactive system as shown in Figure 1. Our system is composed of two cameras installed on the extreme edges of the screen to acquire real-time images of the gamer. Gamer's face is analyzed and his pose and expressions are synthesized by the system to clone or retarget his features in the form of an avatar so that the gamers can interact with each other virtually. In the following paragraphs we briefly explain solutions by the facial analysis and synthesis systems, embedded in our proposed interactive system.

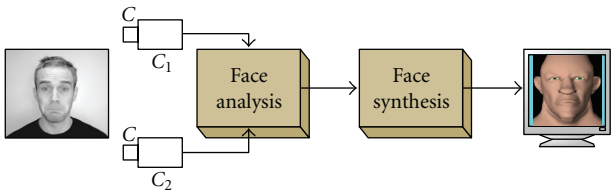


Figure 1: Global system.

1.1. Face Analysis. Human faces are nonrigid objects. The flexibility of a face is well tackled with the appearance-based or deformable model methods [1], which are remarkably efficient for features extraction and alignment of frontal-view faces. As we will see in Section 2, researchers worked out the bottlenecks of face analysis by emphasizing on the model generation and their search methodologies. However we emphasize on increasing the amount of data to be processed with the help of multiple cameras as shown in Figure 1. In single-view system face alignment cannot be accomplished when a face occludes itself during its lateral motion, such as in a profile view only half of the face is visible. To overcome this dilemma we exploit data from another camera and associate it with the one unable to analyze at the first place. In multicamera system, optimization of more than one error is to be performed between a model and query images from each camera. Searching for an optimum solution of a single task employing two or more distinct errors requires multiobjective optimization (MOO). Many MOO techniques exist but to analyze the face we propose optimization of MOAAM by Pareto-based NSGA-II [2] due to its exploitation and exploration ability, nondominating strategy and population based approach which provide the mutual interaction of the results by multiple cameras. In this paper, we use our previous work of [3] and improved our system by obtaining new results based on a new synthetic face database.

1.2. Face Synthesis. In facial synthesis system the purpose is to retarget or clone gamer's face orientation and its features on the synthetic model so that the gamers can interact with each other virtually. Cloning and retargeting is difficult, because avatar does not have the same morphology as the gamer. Our contribution in this system is the introduction of a simple mathematical relation between their appearances called ATM (Appearance Transformation Matrix). To calculate it we make use of two databases explained in Section 5.1. The first database is a large collection of human facial expressions (H-database) and the second database is an optimal database of synthetic facial expressions (A-database) constructed for the avatar based on the analysis of the H-database. Our second contribution is to provide an interactive system for the gamer to build his own database and calculate gamer's specific ATM. The generation of the gamer's database is based on our face analysis system of MOAAM and is obtained by requesting the gamer to imitate few specific and relevant facial expressions displayed on the screen.

Whole system works in two phases. First of all, user's oriented face is analysed by MOAAM, which gives its appearance and pose parameters. These appearance parameters are pose-free and belongs to the frontal face of the user. Therefore they are transformed by ATM in the synthetic face's parameter space and synthetic face is synthesized accordingly. After that pose parameters obtained previously by MOAAM analysis are used to adjust the orientation of the avatar being displayed on the screen.

Remaining of the paper is organized as follow. Section 2 presents the previous and related work in both the domains of facial analysis and synthesis. Section 3 presents the preliminary concepts of our system. Section 4 describes the work done in face analysis. Section 5 explains the system to synthesize a face. Detailed description of our proposed interactive system is elaborated in Section 6, while Section 7 concludes the paper.

2. Previous and Related Work

In this section we have divided the previous and related work for both facial analysis and synthesis into two subsections. However, our first contribution in the facial analysis domain is explained in detail in Section 4. And our second contribution in the facial synthesis domain is explained in Section 5.

2.1. Face Analysis

2.1.1. Multiple 2DAAM. Active Appearance Model (AAM) is one of the well-known deformable method [1] efficient in feature extraction and alignment of a face. References [4, 5] performed pose prediction by using 3 AAM models, one dedicated to the frontal view and two for the profile views. References [6, 7] implemented Active Shape Model (ASM) for the face alignment, by using 5 poses of each face to create a model. Reference [8] also used 3 DAMs (Direct Appearance Models) for face alignment. Reference [9] used another appearance based architecture employing 5 view-specific template detectors to track large range head yaw by a monocular camera. The Radial Basis Function Network interpolates the response vectors obtained from normalized correlation from the input image and 5 template detectors.

Use of more than one model of AAM has some disadvantages: (i) Storage of shapes and textures of the images of all the models requires an enormous amount of storage memory. (ii) Extensive processing of computing 3 AAM in parallel to determine the model required for query images, eventually makes the system sluggish. Moreover classical AAM search methodology requires precomputed regression matrices, which become a burden on time and memory as the amount of training images increases.

Coupled View AAM is used in [10] to estimate the pose. In the training phase they include 2D shapes and 2D textures of both frontal and profile views of each subject. Appearance parameters of their CV-AAM have the capability to estimate the pose. Appearance parameters of their model can tune both the shape and the profile angle of a face. For the profile angle estimation they have used several appearance

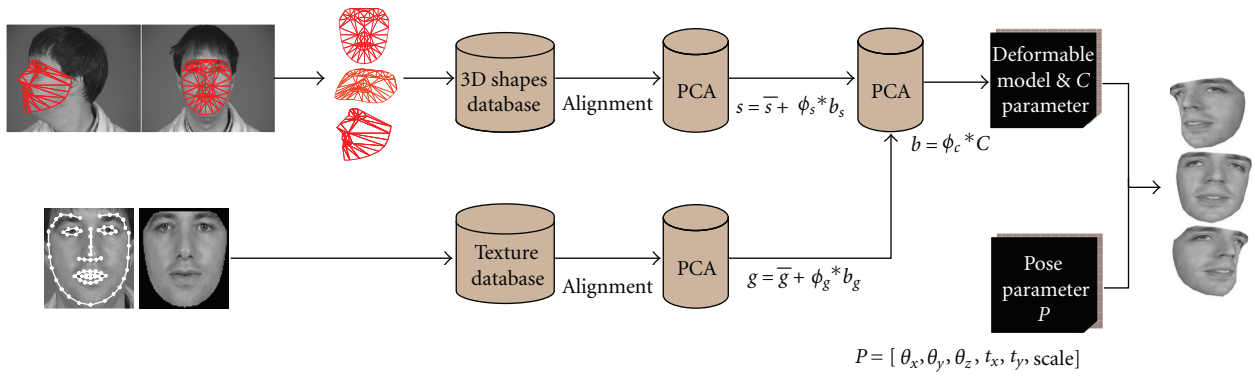


Figure 2: AAM modeling.

parameters which can be replaced by one pose parameter in a 3D AAM. Thus, increase in the number of parameters decreases the rapidness of the system.

2.1.2. 3DAAM. Face can also be aligned by 3D deformable model methods in which a set of images are annotated in 3D to model a face. Reference [11] used 3D face model Candide along with simple gradient descent method as a search algorithm for face tracking. References [12] used 2D+3D AAM along with a fitting algorithm, called inverse compositional image alignment algorithm, which is again an extension of a gradient descent method. Reference [13] applied 3D AAM for face tracking in a video sequence using same IC-LK (Inverse Compositional Lucas-Kanade) algorithm. The optimization by gradient descent lack the properties of exploration and diversity, hence cannot be used in MOO. In our previous work of [14] we have used genetic algorithm instead of gradient descent for the optimization in 2.5D AAM.

2.1.3. Multiview Fitting by 2D or 3DAAM. Pose angles can be estimated by fitting the above 2D or 3D deformable models on multiple images acquired by two, three or multiple cameras. Reference [15] proposed a robust algorithm of fitting a 2D+3D AAM to multiple images acquired at the same instance. Their fitting methodology, instead of decomposing into three independent optimizations from three cameras, adds all the errors. Moreover they used gradient descent (ICLK: Inverse Compositional Lukas Kanade) algorithm as a fitting method, which eventually requires to precompute Jacobians and Hessian matrix. Reference [16] proposed another algorithm of face tracking by Stereo Active Appearance Model (STAAM) fitting, which is an extension of the above fitting of 2D+3D AAM to multiple images. Lack of exploration capability of the method makes ICLK very sensitive to initialization.

In [17] the advantages of adaptive appearance model based method with a 3D data-based tracker using sparse stereo data is combined. Reference [18] proposed a model-based stereo head tracking algorithm and is able to track six degrees of freedom of head motions. Their face model contains 300 triangles compare to our 113 triangles usually used in classical AAM and ICLK based AAM and so forth. Moreover their initialization process requires user intervention. Reference [19] performed 2D head tracking for each subject from multiple cameras and obtained 3D head coordinates by triangulation. Lack of ground truth error calculations creates uncertainty in the accuracy of their system. Furthermore slight calibration error massively deteriorates the triangulation.

Our proposition of face alignment is based on two cameras using 2.5D AAM optimized by Pareto-based multiobjective genetic optimization of NSGA-II. It not only eliminates the steps of precomputation but also provides both exploration and exploitation capability in the search by NSGA-II. Hence it is not sensitive to initialization.

2.2. Face Synthesis. By facial cloning, we refer to the action of transferring the animation from a source (typically a human face) to a target (another human face or a synthetic one). The cloning (or *retargeting*) can be either direct or indirect. In direct retargeting, the purpose is to transfer the motion itself of a few selected interest markers (and optionally a texture) from one face to another [20]. The marker trajectories usually undergo a transformation that compensates for the morphological differences between the source and the target face [21–24]. This morphological adaptation is not always satisfactory, especially if the source and the target faces are very different. An interesting way to get around this difficulty is to turn to indirect retargeting. In indirect retargeting, the motion data is not transferred as such, but is first converted by a specific model to a better representation space, or parameter space, more suited for the motion transfer [25–27]. In the next paragraph we will go over some of the most common representations used for indirect retargeting.

In order for a facial parameterization to be suited for retargeting applications, it must be adapted to the extraction of parameters from motion capture data, and offer an accurate description of facial deformations. Early parameterization schemes like direct parameterizations [28] or pseudomuscle systems [29–31] usually have the advantage of being simple to conceptualize and computationally efficient, but the obtained parameter sets are generally not optimal. In particular, when not operated carefully, they

can generate inconsistent facial configurations. Besides, it is not straightforward to extract the values of the parameters from raw facial motion data (video or 3D motion capture). Muscle physics systems attempt to simulate more rigorously the mechanical behavior of the human face, and thus tend to improve the degree of realism of facial deformations [32]. Yet, as for direct parameterization, the manipulation of the muscle network is not particularly intuitive, and the extraction of muscular contractions from video or motion capture data remains an open problem [33]. A popular facial parameterization which directly originates from observation is the Facial Action Coding System (FACS) [34]. This scheme was originally meant to describe facial expressions in a standardized way in terms of combination of basic facial Action Units (AU). Its coherence and good practical performances made it an interesting tool on which to build performance based animation systems. The MPEG-4 standard later extended this concept for facial animation compression purposes, introducing the Facial Animation Parameters (FAP) [35]. The FACS and MPEG-4 FAP have been used to capture and retarget static and dynamic facial expressions between human and synthetic faces [36, 37]. The disadvantage of methods based on multiple separate action units, is that the natural correlation between multiple facial action occurring in each facial expression is ignored. Thus the animation resulting from these approaches tend to be somewhat nonhuman or robotic.

More recently studies have aimed at obtaining more natural parameterization by performing a statistical modeling of the facial motion. This consist in gathering a collection of relevant examples (database) and to statistically detect particular variation modes, which encompass the specificity of the source or the target. The facial parameters correspond to the contribution of these modes. When two faces have corresponding models, Animations can be easily transferred by mapping the model parameters from one face to the other. Many studies have pointed that motion data consisting of only the positions of a few markers cannot efficiently capture the subtleties of human facial expressions, and have proposed to also capture the textural information [38]. Active Appearance Models (AAM) are frequently used for that propose, since they encompass the motion of well chosen geometric points as well as the pixel intensity changes occurring on the faces, which account for finer deformation of the skin [1]. References [39, 40] obtain impressive results of facial expressions transfer between multiple human faces based on an AAM parameterization. For this type of retargeting scheme to be successful however, the appearance models of the source and the target must characterize the same scope of expressions. In particular their databases must correspond. Constructing a database of expressions for a synthetic face which matches the scope of the source human database is not trivial. Reference [41] transfer facial expressions from the AAM parameters of a human face to an avatar based on a blendshape database. The database of the avatar consists of key expressions selected from the human database, however too few expressions are used for the virtual face to allow for a detailed expression retargeting. Reference [42] later improved this approach by preprocessing the human database in order to automatically isolate individual facial actions. Each of the facial actions can then be reproduced on the avatar to construct a blendshape database. For a reasonable number of facial expressions, this approach ensures the compatibility between the source and target database, without requiring the construction of many avatar facial examples. Yet, for a more complete scope of facial movements, the number of individual facial actions can become large, and thus the number of facial configurations for the avatar database as well. Moreover, by decomposing the expressions into individual units, the correlation between these units when performing an expression is lost in the parameterization. Reference [27] performs a linear retargeting of monocular human appearance parameters to muscle-based animation parameters. The transfer function is based on the matching of a human database of key expressions with a database of corresponding animation parameters for the synthetic face. Yet, the choice of the database key expressions is subjective in that case. Moreover the synthetic face is animated with muscle contraction parameters which can sometimes lead to incoherent interpolation, and prevents the system from being used with other types of animation methods.

We propose a new method to efficiently transfer facial expressions from a human face to a synthetic face, based on pose-free active appearance model parameters delivered by our multiple camera system. The method analyzes a human expression database, and automatically determines which key expressions have to be constructed in the avatar database for the expression retargeting to be efficient.

3. Preliminary Concepts

3.1. 2.5D AAM Modeling. 2.5D AAM of [3, 14] is constructed by (i) 2D landmarks of the frontal view (width and height of a face model) and x coordinates of landmarks in profile view (depth of a face model) combined to make 3D shape model and (ii) 2D texture of only frontal view mapped on its 3D shape. In the training phase of 2.5D AAM, 68 points are marked manually as shown in Figure 2.

All the landmarks obtained previously are resized and aligned in three dimensions using Procrustes analysis ([43, 44]). The mean of these 3D landmarks is calculated which is called mean shape. Principal Component Analysis (PCA) is performed on these shapes to obtain shape parameters with 95% of the variation stored in them:

$$s_i = \bar{s} + \phi_s * b_s, \tag{1}$$

where s_i is the synthesized shape, $\bar{s}$ is the mean shape, ϕ_s are the eigenvectors obtained during PCA and b_s are the shape parameters.

The 3D mean shape obtained in the previous step is used to extract and warp (based on the Delaunay triangulation) the frontal views of all the face images. Only two dimensions of the mean shape are used to get 2D frontal view textures. That is why we call our model as 2.5D AAM, since it is composed of landmarks represented in 3D domain but only 2D texture is warped on this shape to adapt 2.5D model.

Figure 3: Snapshots of rotating 2.5D AAM.

Mean of these textures is calculated. Followed by, another PCA to acquire texture parameters with 95% of the variation stored in these parameters:

$$g_i = \overline{g} + \phi_g * b_g, \tag{2}$$

where g_i is the synthesized texture, $\overline{g}$ is the mean texture, ϕ_g are the eigenvectors obtained during PCA and b_g are the texture parameters.

Both of the above parameters are combined by concatenation of b_s and b_g. And a final PCA is performed to obtain the appearance parameters:

$$b = \left[b_s b_g\right]^T, \qquad b = \phi_C * C, \tag{3}$$

where ϕ_C are the eigenvectors obtained by retaining 95% of the variation and C is the matrix of the appearance parameters, which are used to obtain shape and texture of each face of the database.

2.5D model can be translated as well as rotated with the help of pose vector P:

$$P = \left[\theta_x, \theta_y, \theta_z, t_x, t_y, \text{Scale}\right]^T, \tag{4}$$

where θ_x corresponds to the face rotating around the x axis (pitch: shaking head up and down), θ_y to the face rotating around the y axis (yaw: profile views) and θ_z to the face rotating around the z axis (roll). t_x, t_y are the offset values from the supposed origin and Scale is a scalar value for the magnification of the model in all the dimensions. Figure 3 shows the model rotating by changing θ_y, making left and right semi profile views.

In segmentation this deformed, rotated and translated shape model obtained by varying C and P parameters, is placed on the query image I to warp the face to mean frontal shape. After this shape normalization we apply photometric texture normalization to overcome illumination variations. The objective is to minimize pixel error

$$e = \sqrt{\sum_x [I(C,P) - M(C)]^2}, \tag{5}$$

where $I(C,P)$ is the segmented image and $M(C)$ is the model obtained by C parameters. To choose good parameters we need an optimization method. In our proposition, both of these pose P and appearance parameters C are optimized by genetic optimization of NSGA-II.

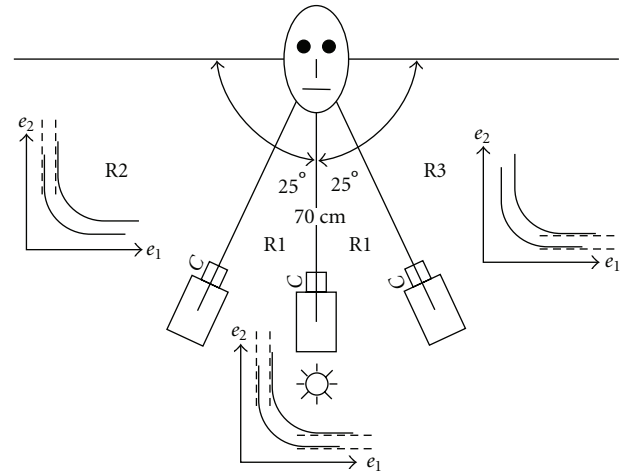


Figure 4: MultiView System.

3.2. Multiple Camera System. In single-view system face alignment cannot be accomplished when a face occludes itself during its lateral motion. Such as in a profile view only half of the face is visible. To overcome this dilemma we exploit data from another camera and associate it with the one unable to analyze at the first place. This association helps the search methodology to reduce the possibility of divergence. Moreover better outcomes of one camera can escort the other. In multiview systems, higher the amount of processing data higher is the robustness ability of a system however efficiency deteriorates due to high consumption of processing time and memory. In other words a trade-off is required between robustness and efficiency.

A database of facial images capable of self assessing is desired to validate our application. The community lacks such a database which involves lateral motion of a face captured by more than one camera. In order to implement our application we developed a multiview scenario. The purpose of constructing this multiview system is to emulate the scenario of integrating two off the shelf webcams placed on the extreme edges of the display screen facing towards the user as shown in Figure 4.

AAM rendered on the facial images of both webcams are blended together to represent a face model seen by a virtual camera placed in between. The results of this virtual webcam are compared by a third camera actually placed at the center. In other words it is a comparison between a multicamera system by MOAAM with single-camera system by SOAAM (Single-Objective AAM). These three cameras are placed 25 degrees apart on a boundary of a circle with a radius of 70 cm as shown in Figure 4. Center of this circle serves as a principal point for each camera. Seven individuals from a research team are invited for screen shots with the intention of obtaining 1218 images with lateral motion. Each individual rotates his face gradually from frontal view to left and right profile views. At each instance three images from each webcam are acquired simultaneously to obtain temporally synchronized images.

Figure 5: Test database images: Same pose from 3 webcams.

3.2.1. Illumination. It remains steady through out the sequence. It is accomplished by a white ambient light placed behind the central camera as shown in the Figure 4. The light we used comes with the stand and a built-in umbrella holder to give extra flexibility. By adjusting the umbrella's position we have rejected the bright spot on the face. It works well for taking facial images with webcams.

3.2.2. Camera Calibration. It is performed by a publicly available toolbox [45]. A simple planar checkerboard is placed in front of the cameras and sequence of images are taken to calculate calibration parameters. With the help of the toolbox, four corners of the checker board are extracted and calibration is performed with respect to the grid of the checkerboard. The toolbox calculates intrinsic parameters (focal length, principal point, distortion and skew) and extrinsic parameters (rotation vector and translation vector) for each camera. With the help of these parameters, all the facial images of these cameras are calibrated.

Figure 5 shows some images of test database acquired from three webcams. A similar scenario is emulated in the software MAYA for a video of synthetic faces. The synthetic face database does not contain camera calibration error hence it is helpful to analyze results free of calibration errors. Figure 6 show some examples of test database of synthetic faces (Synthetic face in the first row was obtained from www.ballistic.com, while remaining face models were made in a software named as "Facial Studio". All of them were imported in MAYA for rendering the synthetic facial images). Some of the facial images of M2VTS [46] (learning database) are also shown in Figure 7.

4. Face Analysis

The main objective of our application is to clone a real human face in the form of an avatar. For such an application face analysis plays an important role for face synthesis. The more efficient the analysis is, facial synthesis is likely to be more accurate. To obtain an efficient and robust face analysis system we acquire a human face with two cameras and analyze it by an appearance based morphable model of 2.5D AAM.

4.1. MOAAM. In single-view system, single error between model and query image is optimized. However in multiview system, the optimization of more than one error is to be performed between a model and query images from each camera. AAM fitting on multiviews is shown in Figure 8. In multiview AAM, the model is rendered on both the images from each camera with the same C parameters.

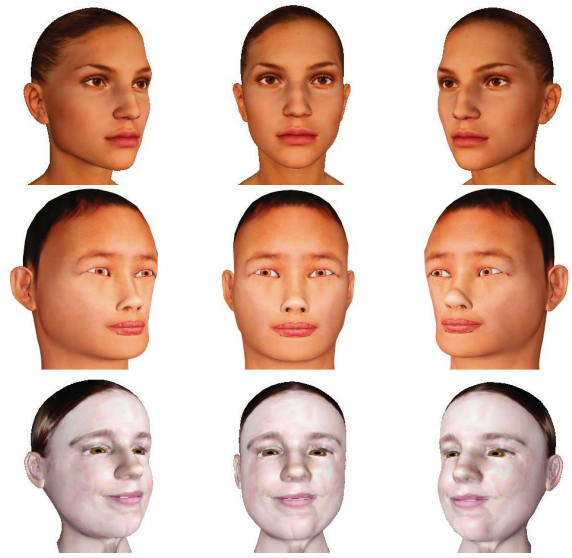


Figure 6: Test database synthetic images.

Figure 7: Learning database images.

The P parameters also remain the same except a yaw angle offset (θ_{offset}) is introduced between the models rendering on two images. After segmentation, pixel errors between both the images and models are calculated. The objective is to minimize pixel error of (5) obtained from each of the two cameras

$$e_1 = \sqrt{\sum_x [I_1(C, P_1) - M(C)]^2},$$
$$e_2 = \sqrt{\sum_x [I_2(C, P_2) - M(C)]^2}, \tag{6}$$

where P_1 and P_2 are linked by an offset of yaw angle. In order to optimize both errors we propose Pareto-based NSGA-II MOO.

4.1.1. NSGA-II. Genetic Algorithm is a well-known search technique. We have used its multiobjective version of Non-dominated Sorting Genetic Algorithm (NSGA-II) proposed by [2] to optimize the appearance C and pose parameters P. The target is to find out the best possible values of these parameters giving minimum pixel errors between the model and the query images of both cameras. In this optimization technique each parameter is considered as a gene. All the genes of C and P are concatenated to form a chromosome. A population of particular number of chromosomes is randomly created. Pixel errors (fitness) between query images and the model (represented by each chromosome) are calculated. Tournament selection is applied to select

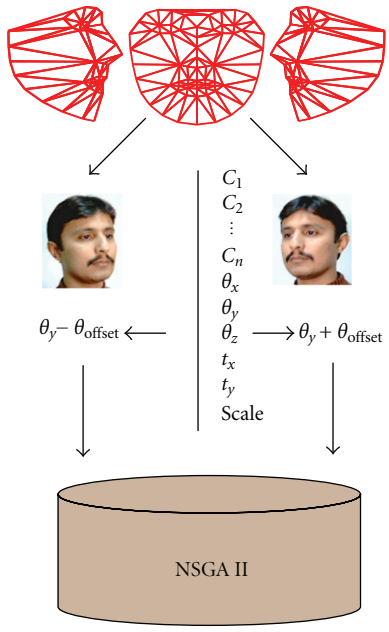


Figure 8: Fitting of MOAAM.

parents from the population to undergo reproduction. Two point crossover and Gaussian mutation is implemented to reproduce the next generation of chromosomes. Selection and reproduction is based upon nondominating sort. The objective is to minimize both of these pixel errors, hence nondominating scenario is to be implemented by Pareto optimization.

4.1.2. Pareto Fronts. The fitting of AAM to image data is performed by minimization of the error function. In MOO several error functions are to be minimized, hence mutual relation of these errors point towards the appropriate MOO method. Dominating errors can be dealt with non Pareto-based MOO, but in this scenario both cameras serves the same purpose of acquiring images of a face. Hence non-dominating scenario is to be implemented with the desired Pareto optimum solution. The basic idea is to find the set of solutions in the population that are Pareto nondominated by the rest of the population as shown in Figure 9(a). These solutions are assigned the highest rank and are removed from further assignment of the ranks. Similarly, the remaining population undergoes the same process of ranking until the population is suitably ranked in the form of Pareto fronts as shown in the Figure 9(b). In this process some kind of diversity is required in the solutions to avoid convergence to a single point on the front. This diversity can be achieved by the exploration quality of Genetic Algorithm.

4.1.3. Switching of MOO to SOO. Processing data from two cameras is meaningful as long as they are relevant. With respect to a camera if a face is oriented such a way that it occludes itself there is no need of processing data from this camera. Eventually in order to avoid wastage of processing we divide field of views of both cameras in three regions R1, R2 and R3 as shown in Figure 4. To determine the region of the face orientation Pareto-based NSGA-II is applied to evolve populations until small number of generations. After each generation evolution, the histogram of genes of the entire population representing the yaw of a face is observed. This histogram follows one of the three curves of Figure 10. Histogram curve-1 corresponds to region-1, where the information from both the cameras are meaningful and data from any one of them cannot be neglected. Whereas histogram curve-2 and curve-3 corresponds to region-2 and region-3 respectively, where the information from one of the camera is sufficient enough to localize the facial features and other camera can be discarded. After few generations, current population decides whether to stay in MOO or to switch to single objective optimization (SOO). Mathematically, let us suppose Pop is a set of population given as

$$Pop = \begin{vmatrix} X_{11} & X_{12} & \cdots & X_{1k} & \cdots & X_{1M} \\ X_{21} & X_{22} & \cdots & X_{2k} & \cdots & X_{2M} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{N1} & X_{N2} & \cdots & X_{Nk} & \cdots & X_{NM} \end{vmatrix}, \quad (7)$$

where N is the number of chromosomes X and M is the number of genes of each chromosome. Now we observe the k_{th} gene of each chromosome which represents yaw angle of the model. In order to calculate the histogram of chromosomes, we assign 1 to ζ such as

$$\zeta_i = \begin{cases} 1 & -\theta_{\text{th}} \le X_{ik} \le \theta_{\text{th}} \\ 0 & X_{ik} \le -\theta_{\text{th}} \text{ or } X_{ik} \ge \theta_{\text{th}} \end{cases} \quad 1 \le i \le N, \quad (8)$$

where θ_{th} is the threshold angle equals to the half of the angle between two cameras. ϵ is the ratio of number of chromosomes representing the face position in region-1 to the total number of chromosomes:

$$\epsilon = \frac{\sum_{i=1}^{N} \zeta_i}{N} = \begin{cases} < 0.50, & \text{Single camera mode,} \\ \ge 0.50, & \text{Multiview mode.} \end{cases} \quad (9)$$

The value of ϵ decides whether to stay in MOO and utilize both cameras or to switch to single camera mode.

4.1.4. MOAAM Fitting. For MOAAM (also called MVAAM: Multiview AAM) fitting we refer readers to our previous work of [3], which illustrates stepwise detailed description of MOAAM fitting on a query image. It includes steps of initialization, reproduction, segmentation, fitness calculations, nondominating sort, replacement and switching of MOO to SOO. In our previous work we have highlighted the effects of slight errors caused by the camera calibration and the ground truth points for a real face database.

Camera calibration problem arises when we compare MOAAM results to SOAAM. As we have already mentioned in Section 3.2 that models obtained from two cameras placed

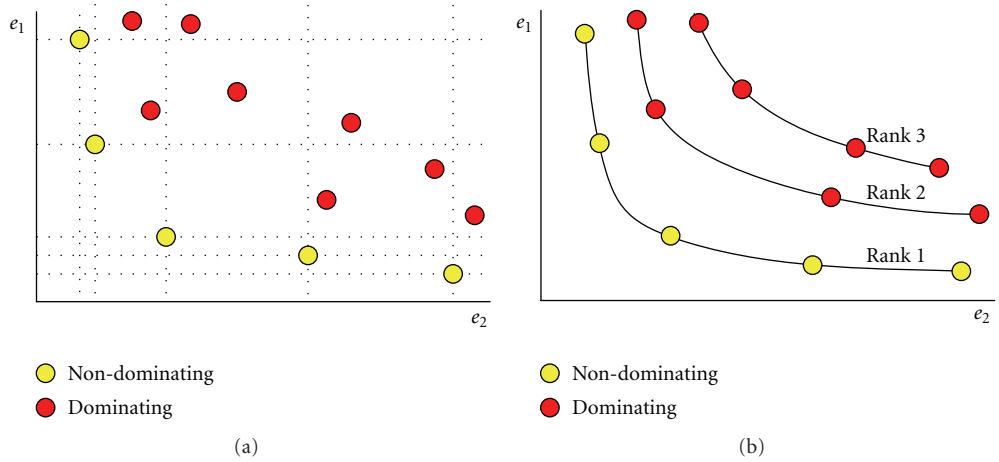


Figure 9: Pareto Fronts.

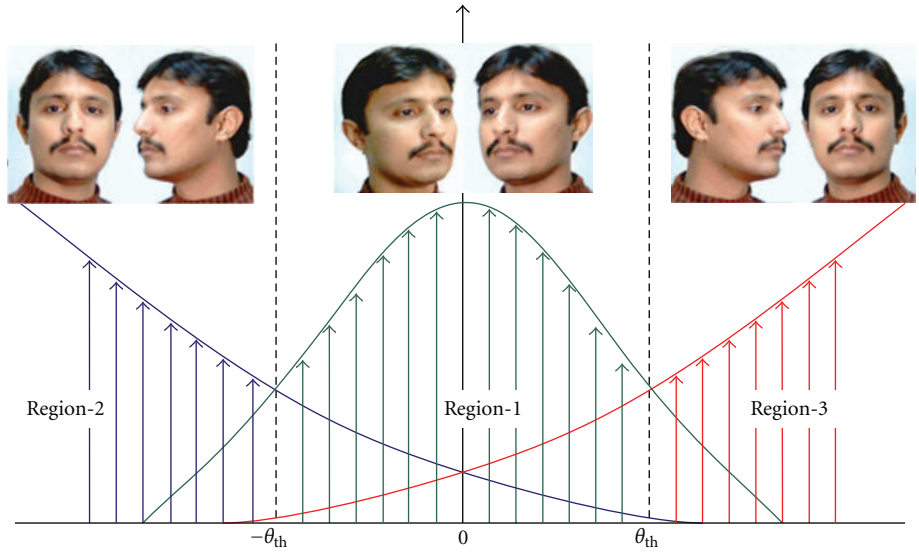


Figure 10: Histogram of chromosomes versus head orientation.

at the extreme edges of the display are blended together to compare it with the one obtained from the central camera. This comparison is highly prone to the calibration error of all the three cameras. Whereas the results from a single camera (SOAAM) do not experience any calibration problem. In this paper we have manage to overcome this dilemma by building a synthetic face database of several individuals. The scenario shown in Figure 4 is emulated, in the software named as MAYA, by placing different synthetic characters in between two virtual cameras each calibrated and located 50° apart. A third camera is placed in-between these two cameras for the comparison of results of a single camera and double camera. These cameras have all the characteristics of an actual camera along with the capability to fix intrinsic and extrinsic parameters to obtain 100% calibration.

Ground truth points are the exact localization of the face orientation and features (nose, eyes and mouth). In real face database there is a possibility of slight errors in the ground truth points since they are marked manually on each facial feature of each image. However in synthetic facial images this problems is solved by obtaining these locations automatically through scripts written in MAYA. With all these modifications we have verified our proposition of MOAAM and have updated our results.

4.2. Experimental Results. We performed simulations using 64×64 pixels AAM by annotating 37 subjects of publicly available databases of M2VTS [46]. However for testing database we have used both real face database and synthetic face database. Both these databases contains 2418 facial images, of 7 real and 10 synthetic faces, from each camera. Among 2418, 806 images are considered to be taken from central camera to validate our results. In testing phase face alignment is performed on all the views from left profile

Figure 11: (a) and (b) Comparison of SOAAM and MOAAM (operating in R2 or R3). (c) and (d) Comparison of SOAAM and MOAAM (operating in R1).

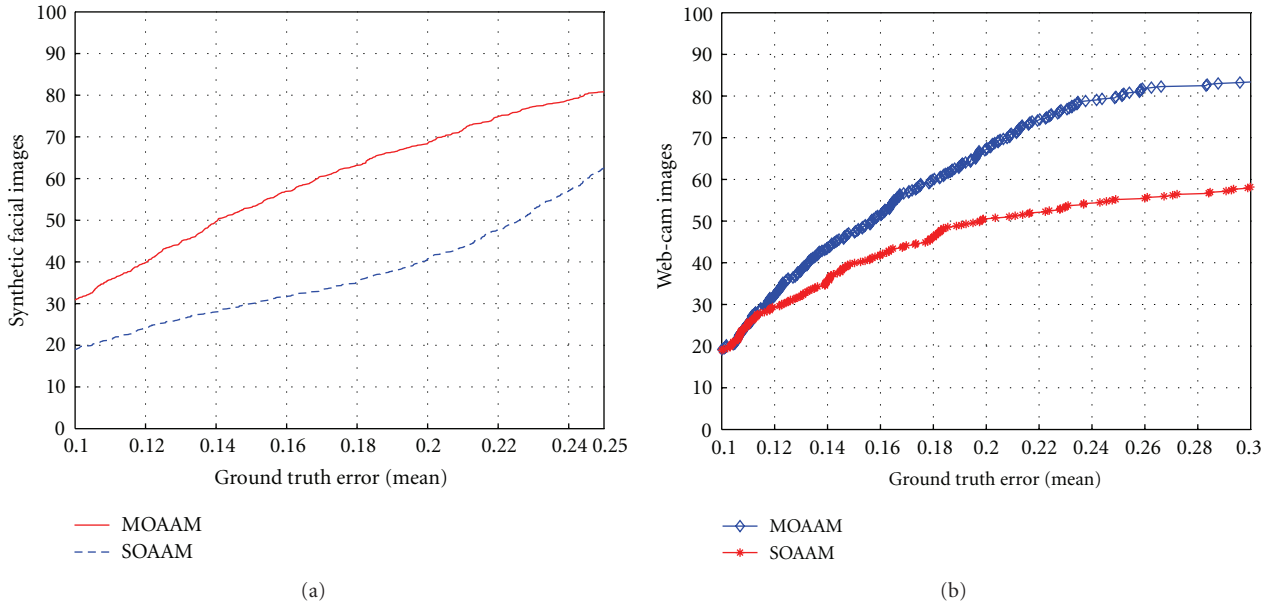


Figure 12: (a) Comparison of GTE_{mean} for MOAAM and SOAAM (Synthetic face images). (b) Comparison of GTE_{mean} for MOAAM and SOAAM (Web-cam images).

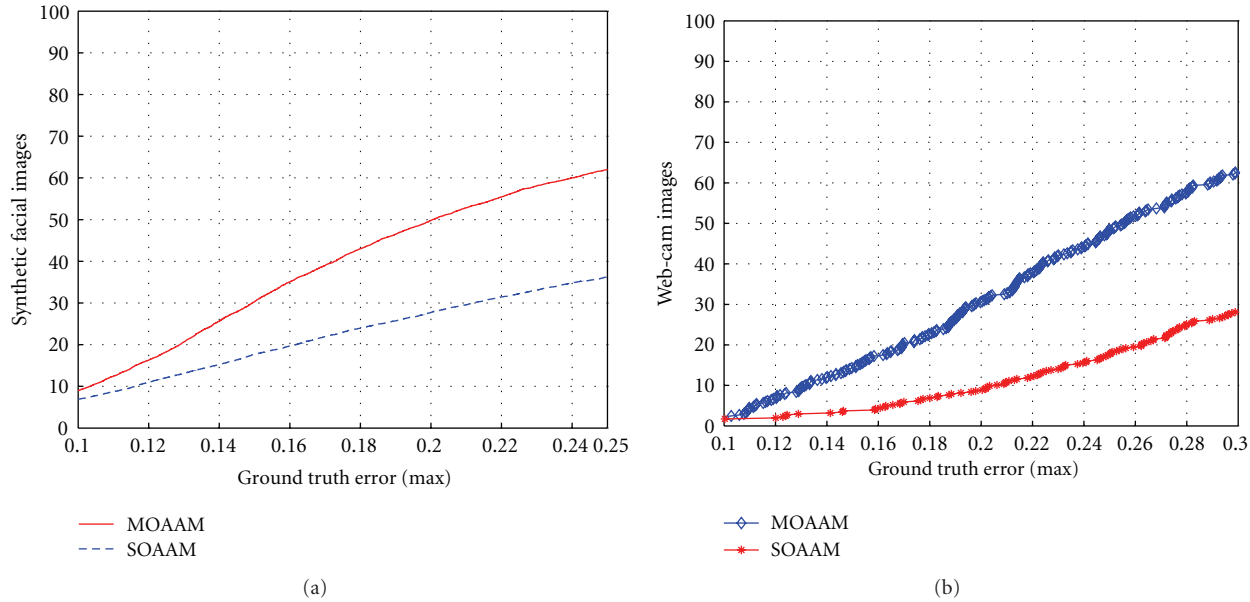


FIGURE 13: (a) Comparison of GTE_{max} for MOAAM and SOAAM (Synthetic face images). (b) Comparison of GTE_{max} for MOAAM and SOAAM (Web-cam images).

to right profile. Two sets of experiments are performed: SOAAM and MOAAM.

4.2.1. Single-Objective AAM. In SOAAM, AAM is rendered on the image sequence from the central camera, which is placed to highlight the benefit of MOAAM. As far as optimization is concerned, SOAAM is optimized by classical GA optimization. Same selection and reproduction criteria of NSGA-II are implemented in GA, in order to give a good comparison.

4.2.2. Multiobjective AAM. In MOAAM, same AAM is rendered on the face image sequence from the other two cameras, which are actually the part of our multiview system. Localization of face on these two images from each camera is performed by Pareto-based MOO of NSGA-II.

Best chromosomes obtained at the end of MOAAM and SOAAM contain best appearance and pose parameters for a given face. Features like eyes, nose and mouth can be extracted from these shapes as shown in Figure 11. First three rows correspond to synthetic faces while remaining rows represent real human faces. It can be seen from the images that as the face moves laterally the feature localization gets far better in two cameras (MOAAM) than in single central camera (SOAAM).

Figure 12(a) shows percentage of aligned synthetic images versus mean ground truth error (GTE_{mean}) of facial features (eyes, nose and mouth). GTE_{mean} is actually the mean error obtained by comparing MOAAM analyzed locations and manually marked locations of all the facial features of a facial image. The error is normalized by D_{eye} which corresponds to the distance between eyes, that is, an error of 1 corresponds to a mean error equal to the distance between the eyes. To eliminate the vagueness of ground truth markings we consider results starting from 0.1 of D_{eye}, which means any two algorithms having a GTE_{mean} less than 0.1 is considered to be equally accurate. While for the maximum threshold results less than 0.25 of D_{eye} is considered to be well converged results. Figure 12(a) depicts that our system of MOAAM fitting by NSGA-II is a lot better than SOAAM fitting. In MOAAM 69% of the images are aligned with a GTE_{mean} less than 0.2 of D_{eye}. Whereas SOAAM aligned 41% of the total images. Similarly Figure 12(b) shows the results of experiments on real faces (previous work); MOAAM 68% and SOAAM 50%.

Figures 13(a) and 13(b) illustrate the comparison of both algorithms with respect to normalized maximum ground truth error (GTE_{max}) for both synthetic and real facial images databases respectively. GTE_{max} represents worst localization of a facial feature (eyes, nose or mouth) normalized by D_{eye}. Figure 13(a) depicts that MOAAM aligned 50% and SOAAM aligned 28% of synthetic facial images with GTE_{max} less than 0.2 of D_{eye}. Whereas Figure 13(b) shows MOAAM aligned 30% and SOAAM aligned 10% of real faces.

As far as time consumption is concerned, it is obvious that at the worst MOAAM required twice of the processing time compared to SOAAM but at the same time accuracy, robustness and increased field of view (FOV) is achieved. Moreover our technique of finding the region of face and discarding the data from the camera by NSGA-II reduces this twice factor. SOAAM required 1600 warps whereas MOAAM instead of 3200 warps required 2700 warps. Each warp equals 90% of the time consumed by an iteration, that is, 0.03 milliseconds in Pentium-IV 3.2 GHz. Therefore each facial image requires 90 milliseconds for the analysis without any prior knowledge of the pose, however in tracking mode we can reduce this time by employing pose parameters of previous frames, which eventually reduces the number of

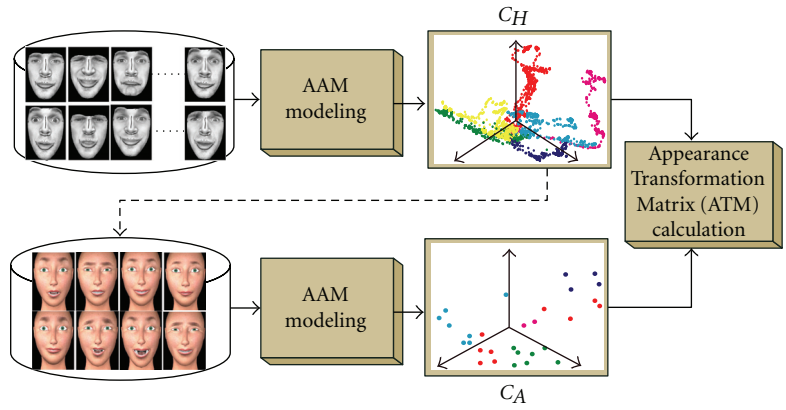


Figure 14: Overview of the face synthesis system. (Colors represents different types of expressions and are shown for the clarity of the display only).

warps (iterations). Moreover facial analysis by MOAAM can be made as a generic or a person specific MOAAM. In generic MOAAM the query face is totally unknown and to analyze it we need a vast learning database, whereas in person specific MOAAM model is generated from facial images of the same individual who would be analyzed by the system. Eventually person specific MOAAM is more time efficient and robust compared to generic MOAAM.

5. Face Synthesis

The goal of our application is to clone the gamer's facial expression to an avatar. The cloning consists of transferring the facial expressions from a source (typically a human face) to a target (another human face or a synthetic one). The avatar facial deformations then originates from real human movements (performance-based facial animation), which usually look more natural than manually-designed facial animation. Moreover, since the expressions of the gamer are captured and transferred in real-time, the facial animation of the avatar acts as a real gaming experience, and significantly improves the interactivity of the game compared to prerecorded animation sequences.

5.1. System Description. In this section, we present a general description of a system that provides an efficient parameterization of an avatars face for the production of emotional facial expressions, relying on captured human facial data. Here we make use of two databases of our previous work of [47]. An illustration of the system and its applications is displayed on Figure 14.

5.1.1. H-Database. The entry point of the system is a database of approximately 4000 facial images of emotional expressions (H-database). These images have been acquired on an actor performing facial expressions without rigid head motion. The database was constructed to contain an important quantity of dynamic natural expressions, both extreme and subtle, categorical and mixed. A crucial aspect of the analysis is that the captured expressions do not carry any emotional label. The facial images will allow us to model the deformation of the face according to a scheme used in Section 3.1. The AAM procedure delivers a reduced set of parameters which represent the principal variation patterns detected on the face. Every facial expression can be projected onto this parameter space referred to as the appearance space (Figure 14 presents symbolic 3D representations of this space, although it may contain 15 to 20 dimensions). Note that this process is invertible: it is always possible to project a point of the appearance space back to a facial configuration, and thus synthesize the corresponding facial expression as a facial image.

5.1.2. A-Database. A reduced parameter space similar to the one described above can be constructed for the synthetic face, provided that a database of facial expressions for the virtual character is available (A-database). In this section we show how to identify a reduced set of facial configurations from the human database so that a coherent appearance space is constructed for the avatar (typically 25 to 30 expressions). The purpose of this avatar database creation scheme is that the appearance spaces of the human and the synthetic face have the same semantical meaning, and model the same information. It is then easy to construct a mathematical link between them (the ATM as illustrated on Figure 14).

The appearance space for the synthetic face is built through statistical modeling, similarly to the human appearance space (Section 5.1.1). For real faces, thousand of database samples can be produced with a video camera and a feature-tracking algorithm, whereas the elements of an equivalent synthetic database are manually-designed facial configurations, which are not easy to obtain. It is thus desirable to keep the number of required samples small.

Our idea for building the A-database, is to use the human database, and extract the expressions that have an important impact on the formation on the appearance space. Indeed, a lot of samples from the human database bring redundant information to the modeling process, and are therefore not essential in the A-database. Following this logic, we are able to reduce the set of necessary expression to a reasonable size.

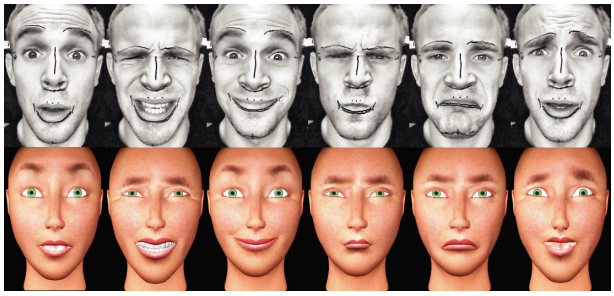


Figure 15: The first elements of the human expression located on the convex hull of the point cloud formed by all database elements (top). Avatar's expressions corresponding to each human expression (bottom).

Practically, We select the extreme elements of the database, meaning the elements presenting the maximal variations with respect to a neutral facial expression. In terms of parameter space, these elements are located on the convex hull of the point cloud formed by all database elements and are detected using [48]. These samples are responsible for shaping the meaningful variance of the database and thus encompass the major part of its richness. By manually reproducing these selected expressions on the face of the virtual character, we can build its very own appearance model according to the method presented in Section 3.1. Our studies have shown that 25–30 expressions are enough to train an efficient appearance model.

For the human database, we used more than 4000 elements. Using the convex hull procedure we have been able to identify 25–30 representatives for the reduced database (see Figure 15), with a small reconstruction error. Such a reduced database can be constructed for any synthetic character, and any human face based on the same extracted elements (see construction of the gamer's database in Section 6.2). Having to design several facial configurations manually on a synthetic character is a limitation of the method, yet it also can be seen as an advantage: our system does not rely on any particular facial control method (muscle systems, blendshapes, etc). Any scheme able to provide good facial configurations can be used. Our system can therefore easily be integrated in already-established workflows.

The database construction method creates a specific connection between the two databases, and thus the two appearance spaces. In the next sections, we will see how we benefit from it to animate the avatar based on the human motion data.

5.1.3. Appearance Transformation Matrix (ATM). The ideas developed in the previous section have lead to the construction of analogous appearance spaces for the human face and the synthetic face. Both spaces are connected, since the construction of the avatar appearance space is based on elements replicated from the human database. It follows that we have a correspondences between points in the human appearance space and points in the avatar space. We propose to use this sparse correspondence to construct an analytical

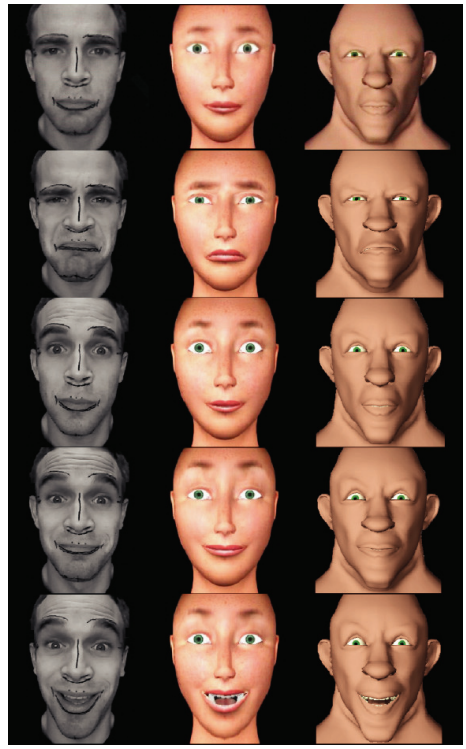


Figure 16: Examples of cloning of facial expressions. The expressions captured on the human face (left) are successfully transferred to the faces of avatars (middle and right). First row shows neutral faces.

link between both spaces. This link will then be used to transform human appearance parameters C_H into avatar appearance parameters C_A, and thus clone a human facial expression on the synthetic face.

It can be noted that the modeling scheme of AAM we use is linear equations (1), (2) and (3). Linear variations and combinations are thus preserved by the modeling steps, and we wish to maintain this linear chain in the retargeting process. Therefore, as in other approaches like [27], we applied a simple linear mapping on the parameters of the appearance spaces:

$$\begin{vmatrix} C_{H(11)} & C_{H(12)} & \cdots & C_{H(1k)} \\ C_{H(21)} & C_{H(22)} & \cdots & C_{H(2k)} \\ \vdots & \vdots & \ddots & \vdots \\ C_{H(m1)} & C_{H(m2)} & \cdots & C_{H(mk)} \end{vmatrix} = A_0 * \begin{vmatrix} C_{A(11)} & C_{A(12)} & \cdots & C_{A(1k)} \\ C_{A(21)} & C_{A(22)} & \cdots & C_{A(2k)} \\ \vdots & \vdots & \ddots & \vdots \\ C_{A(n1)} & C_{A(n2)} & \cdots & C_{A(nk)} \end{vmatrix} \tag{10}$$

where m and n are the number of appearance parameters of human and synthetic appearance space respectively, while k is the number of expression stored in the database. Hence if

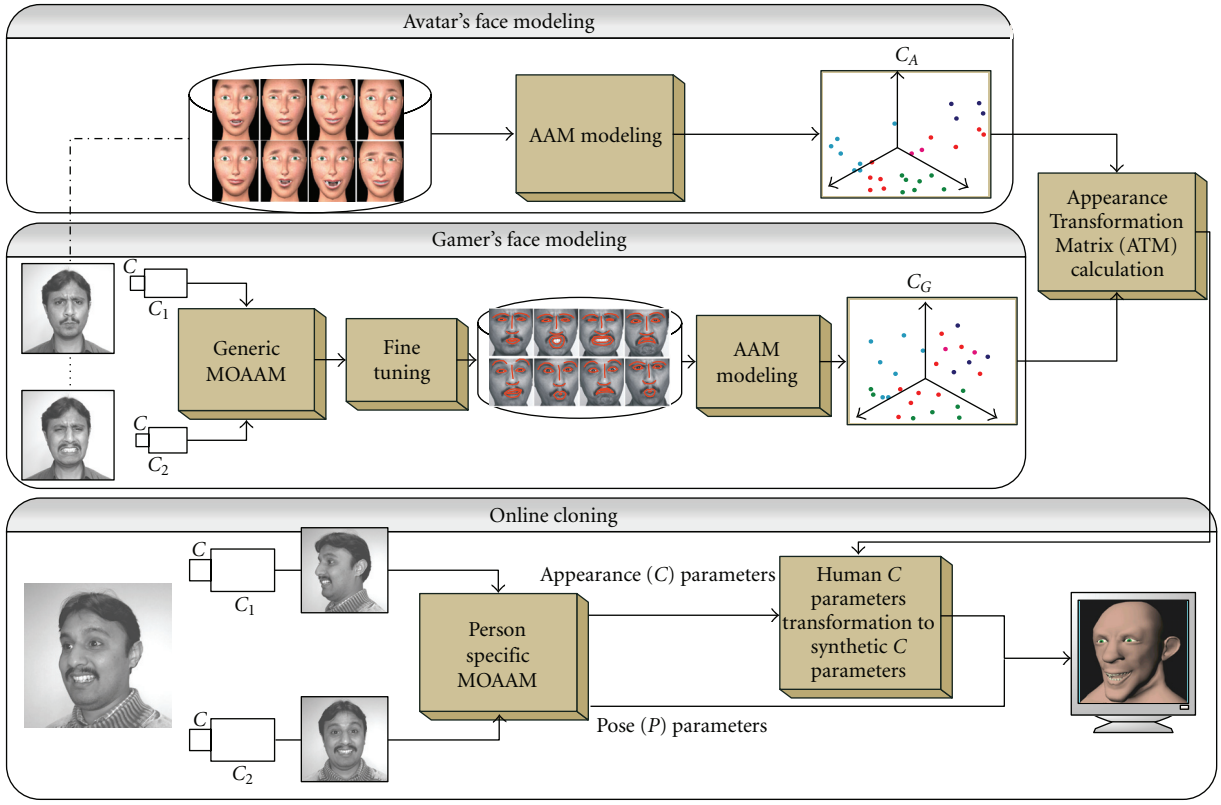


Figure 17: Block diagram of the interactive system.

Table 1: Experimental details. (Avatar1 and Avatar2 are shown in the middle and right columns of Figure 16 resp.).

	C-Parameters	Expression (k)
Human	$m = 24$	28
Avatar1	$n = 20$	28
Avatar2	$n = 18$	28

C_H is a $m \times k$ matrix and C_A is a $n \times k$ matrix, A_0 will be of $m \times n$.

The matrix A_0 is obtained through linear regression on the set of corresponding points. Depending on the dimensionality of the appearance spaces (usually 15 to 20), it can be profitable to turn to Principal Component Regression [49] to cope with a possible underdetermination of the regression problem. Retargeting results are illustrated by a few snapshots on Figure 16. Experimental details used are given in Table 1. Complete sequences of expression retargeting can also be found on the accompanying video.

6. Interactive System

Our proposition is a complete human machine interactive system for a game console. Figure 17 is a detailed description of our system. This time it is viewed from perspective of stages of the global system. System is composed of three stages.

6.1. Avatar's Face Modeling. In this section, we make use of procedure of Section 5.1.2 to obtain a database of simple and realistic facial expressions of an avatar called A-database. The visual aspect of the synthetic character is chosen by the user. Different classes of synthetic faces are available representing different ages, races, gender, physique and features and so forth. Once the class of the avatar is chosen, the required facial expressions, already stored in the system, are generated for this face (from the expressions identified in Section 5.1.2). Note that the system's user has the possibility to edit the suggested facial expression to personalize the look of its avatar by manually clicking and moving the vertices. Ultimately the A-database contains the expressions, on the user-chosen character, which are necessary to form the A-Database.

We can build the its appearance model according to the method presented in Section 3.1. This procedure delivers a reduced set of parameters which represent the principal variation patterns observed on the synthetic face (C_A). Manual marking of the landmark on the synthetic face is not needed as the synthetic face is already generated by the system and it contains the location of each vertex.

6.2. Gamer's Face Modeling. The procedure of training is very simple and unproblematic. The essence of this phase is to make the system learn the facial deformations of the gamer's face so it can replicate the localization of features, emotions and gestures on the synthetic face. The construction of the

Table 2: Processing time for Gamer's face modeling. (offline).

Processing block	Time
Generic MOAAM	90 milliseconds/frame
Fine tuning	30–40 sec/image
AAM modeling	5.68 sec (28 expressions)
Computation of ATM	24 milliseconds

Table 3: Processing time for online cloning.

Processing block	Time (msec/frame)
Person specific MOAAM	34
Transformation by ATM	0.015
Rendering (800×600) (2740 vertices)	30

Gamer's database is similar to the one of the avatar. The gamer has to mimic the expressions that have an important impact on the formation of the appearance space (identified in Section 5.1.2). In practice, the required facial expressions are displayed serially for the user to imitate. Facial images are captured by generic MOAAM, as explained in Section 4 to automatically localize the facial features. Since user is unknown to the system therefore generic MOAAM containing an AAM model based on M2VTS facial images database is used. Feature localized by MOAAM is displayed on the screen for the user to fine tune the location of each feature. Finally all the facial images of the gamer are generated, each corresponding to synthetic facial expression of the A-Database. By reproducing these selected facial expressions of the gamer, we can build its very own appearance model along with its reduced appearance parameters C_G according to the method presented in Section 3.1. With C_G and C_A (obtained in previous section) we can calculate ATM mathematically (see Section 5.1.3). This ATM is gamer dependent and can be used for cloning only for particular gamer who was involved in generating it in the first place. Time cost for this phase is tabulated in Table 2.

$$C_A = A_0 * C_G. \tag{11}$$

6.3. Online Cloning. From the previous two sections we obtained an ATM capable of transforming the appearance parameters from the gamer's appearance space to the avatar's appearance space. In online cloning, this transformation involves only a matrix multiplication of real-time gamer's appearance parameters C_G with A_0 to obtain avatar's appearance parameters C_A. This analytically simple framework enables real-time performances. The virtual illustration of a gamer is cloned in the form of an avatar synthesized by C_A and ultimately display on the screen as shown on Figure 17.

The appearance parameters of a gamer are acquired in real-time by our facial analysis system of multiple cameras. Tactical moves of the game causes the gamer to move a lot in different direction. Yet the retargeting scheme of Section 5.1 has been designed for stable heads. Employing multiple cameras resolved this problem. Two cameras placed at the extreme edges of the screen acquire real-time image of the gamer and at the same time his facial features and pose are analyzed by person specific MOAAM. Person specific MOAAM model is generated from the gamer database of the previous section and it contain all the *pose-free* facial variations of the gamer.

User's oriented face is analysed by MOAAM, to give its appearance and pose parameters. These appearance parameters are pose-free and belongs to the frontal face of the user. These parameters are transformed by ATM in the synthetic face's parameter space and synthetic face is synthesized by them. After that pose parameters obtained previously by MOAAM analysis are used to adjust the orientation of the avatar being displayed on the screen. As shown in the cloning section of the Figure 17, appearance parameters undergoes transformation while pose parameter are directly reproduced on the avatar face to clone both the gamer's expressions and gestures. Time cost of each block, for a Pentium-IV 3.2 GHz platform, is tabulated in Table 3. The linearity of the AAM scheme allows the reproduction of both extreme and intermediate facial expressions and movements, with low computing requirements.

7. Conclusions

In this paper we proposed a solution to solve two bottlenecks of facial analysis and synthesis in an interactive system of human face cloning for nonexpert users of computer games. Facial emotions and pose of gamers cloned to bring their realistic behavior to virtual characters. Bottlenecks of analyzing the human face and synthesizing it in the form of an avatar are dealt with.

Large lateral movements of a gamer makes it impossible to analyze and track his face with single camera. To overcome this dilemma we exploit data from another camera and associate it with the one unable to analyze at the first place. Earlier the cost of a webcam and slow processor demotivated the possibility of managing excessive amount of data from multiple cameras. Currently with wide availability of inexpensive webcams the multiview system is as practical as single-view. To analyze the acquired multiview facial images we proposed multiobjective 2.5D AAM (MOAAM) optimized by Pareto-based NSGA-II. We have presented new results (Section 4.2) because of the problem of calibration and ground truth points in our previous work. Our approach of MOAAM is accurate, robust and capable of extracting the pose, features and gestures even with large lateral movements of a face.

As far as facial synthesis is concerned, cloning the human facial movements onto an avatar is not trivial due to their facial morphological differences. We proposed a new technique of calculating the mathematical semantic correspondence between the appearance parameters of the human and avatar (ATM matrix). We calculated this ATM for the gamer to be able to clone his emotions on the avatar in real-time. The interactive system we have presented is complete and easy to use. We have shown the results of facial

features and pose extraction and how we synthesize these facial details on an avatar by calculating the ATM with the gamer's help.

Although gamer's and avatar's database construction and its training is a long and tedious job. But it is supposed to be done once every time a new gamer is introduced. On the other hand our system is capable of performing online cloning of each frame in 64.015 milliseconds (i.e., 15 frames per second), as being nearly a real-time system. For the moment, this approach is limited to be used in an interactive system for the gamers, but it would be interesting to extend it for larger events, like conferences and meetings, with multiple cameras installed on different corners of the room and displayed on video projectors. Moreover it can be used efficiently in communication where the channel bandwidth is limited, since only the small amount of appearance and pose parameters are transmitted from the human face to the avatar for face synthesis.

Acknowledgments

Part of this work is financially supported by Higher Education Commission Pakistan. The authors would also like to thank the anonymous reviewers.

References

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *Proceedings of the European Conference on Computer Vision (ECCV '98)*, vol. 2, pp. 484–498, Freiburg, Germany, June 1998.

[2] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-II," in *Parallel Problem Solving from Nature VI Conference*, pp. 849–858, 2000.

[3] A. Sattar and R. Seguier, "MVAAM (multi-view active appearance model) optimized by multi-objective genetic algorithm," in *Proceedings of the 8th IEEE International Conference on Automatic Face and Gesture Recognition (FG '08)*, 2008.

[4] T. F. Cootes and C. J. Taylor, "Statistical models of appearance for computer vision," Tech. Rep., Imaging Science and Biomedical Engineering, University of Manchester, 2004.

[5] S. Ting, B. C. Lovell, and C. Shaokang, "Face recognition robust to head pose from one sample image," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 1, pp. 515–518, 2006.

[6] X. Shenjun and A. Haizhou, "Face alignment under various poses and expressions," in *Proceedings of the 1st International Conference on Affective Computing and Intelligent Interaction (ACII '05)*, vol. 3784, p. 3784, Beijing, China, 2005.

[7] L. Yanghua, L. Yang, T. Linmi, and X. Guangyou, "Multiview face alignment guided by several facial feature points," in *Proceedings of the 3rd International Conference on Image and Graphics (ICIG '04)*, pp. 238–241, 2004.

[8] S. Yan and C. QianSheng, "Multi-view face alignment using direct appearance models," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition (FGR '02)*, pp. 238–241, 2002.

[9] M. Romero and A. F. Bobick, "Tracking head yaw by interpolation of template responses," in *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '04)*, vol. 5, p. 83, 2004.

[10] T. F. Cootes, G. V. Wheeler, K. N. Walker, and C. J. Taylors, "Coupled-view active appearance models," in *Proceedings of British Machine Vision Conference*, vol. 1, pp. 52–61, 2000.

[11] F. Dornaika and J. Ahlberg, "Fast and reliable active appearance model search for 3d face tracking," in *Proceedings of Mirage INRIA Rocquencourt*, Paris, France, 2003.

[12] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-time combined 2D+3D active appearance models," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 2, pp. 535–542, 2004.

[13] J. Sung and D. Kim, "Extension of AAM with 3D shape model for facial shape tracking," in *Proceedings of the International Conference on Image Processing (ICIP '04)*, vol. 5, pp. 3363–3366, 2004.

[14] A. Sattar, Y. Aidarous, and R. Seguier, "GAGM-AAM: a genetic optimization with gaussian mixtures for active appearance models," in *Proceedings of the 15th International Conference on Image Processing (ICIP '08)*, pp. 3220–3223, 2008.

[15] C. Hu, J. Xiao, I. Matthews, S. Baker, J. F. Cohn, and T. Kanade, "Fitting a single active appearance model simultaneously to multiple images," in *Proceedings of the British Machine Vision Conference*, 2004.

[16] D. Kim and J. Sung, "A real-time face tracking using the stereo active appearance model," in *Proceedings of the International Conference on Image Processing (ICIP '06)*, pp. 2833–2836, 2006.

[17] F. Dornaika and A. Sappa, "Improving appearance-based 3d face tracking using sparse stereo data," in *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP '06)*, 2006.

[18] R. Yang and Z. Y. Zhang, "Model-based head pose tracking with stereovision," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition (FGR '02)*, p. 255, 2002.

[19] J. Tu, T. Huang, Y. Xiong, T. Rose, and F. K. H. Quek, "Calibrating head pose estimation in videos for meeting room event analysis," in *Proceedings of the International Conference on Image Processing (ICIP '06)*, vol. 5, pp. 3193–3196, 2006.

[20] J. Chai, J. Xiao, and J. Hodgins, "Vision-based control of 3d facial animation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '023*, pp. 193–206, 2003.

[21] L. Williams, "Performance-driven facial animation," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '90)*, pp. 235–242, Dallas, Tex, USA, September 1990.

[22] F. Pighin, R. Szeliski, and D. H. Salesin, "Resynthesizing facial animation through 3D model-based tracking," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 143–150, 1999.

[23] J.-Y Noh and U. Neumann, "Expression cloning," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '81*, pp. 21–28, 2001.

[24] Z. Liu, Y. Shan, and Z. Zhang, "Expressive expression mapping with ratio images," in *Proceedings of the 19th Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 271–276, 2001.

[25] C. Kouadio, P. Poulin, and P. Lachapelle, "Real-time facial animation based upon a bank of 3d facial expressions," in *Proceedings of the Computer Animation*, pp. 128–136, 1998.

[26] E. S. Chuang and C. Bregler, "Performance driven facial animation using blendshape interpolation," Tech. Rep. CS-TR-2002-02, Stanford University, 2002.

[27] Y. Zhang, M. Luo, and S. Xu, "An efficient markerless method for resynthesizing facial animation on an anatomy-based model," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '07)*, pp. 971–974, 2007.

[28] F. I. Parke, "Parameterized models for facial animation," *IEEE Computer Graphics and Applications*, vol. 2, no. 9, pp. 61–68, 1982.

[29] M.-L. Viaud and H. Yahia, "Facial animation with wrinkles," in *Eurographics Worshop on Animation and Simulation*, 1992.

[30] N. Magnenat-Thalmann, E. Primeau, and D. Thalmann, "Abstract muscle action procedures for human face animation," *The Visual Computer*, vol. 3, no. 5, pp. 290–297, 1988.

[31] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann, "Simulation of facial muscle actions based on rational free form deformations," *Computer Graphics Forum*, vol. 11, no. 3, pp. 59–69, 1992.

[32] K. Waters, "A muscle model for animation threedimensional facial expression," in *Proceedings of the 3rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, vol. 21, pp. 17–24, 1987.

[33] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic determination of facial muscle activations from sparse motion capture marker data," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 417–425, 2005.

[34] P. Ekman and W. V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*, Consulting Psychologists Press, 1978.

[35] A. Eleftheriadis, C. Herpel, G. Rajan, and L. Ward, "Mpeg-4 systems, text for iso/iec fcd 14496-1 systems," in *MPEG-4 SNHC*, 1998.

[36] M. Byun and N. I. Badler, "Facemote: qualitative parametric modifiers for facial animations," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, vol. , pp. 65–71, 2002.

[37] C. Curio, M. Breidt, M. Kleiner, Q. C. Vuong, M. A. Giese, and H. H. Bülthoff, "Semantic 3D motion retargeting for facial animation," in *Proceedings of the Symposium on Applied Perception in Graphics and Visualization (APGV '06)*, pp. 77–84, 2006.

[38] E. S. Chuang and C. Bregler, "Mood swings: expressive speech animation," *ACM Transactions on Graphics*, vol. 24, no. 2, pp. 331–347, 2005.

[39] I. Macedo, E. V. Brazil, and L. Velho, "Expression transfer between photographs through multilinear AAM's," in *Proceedings of the Brazilian Symposium of Computer Graphic and Image Processing (SIBGRAPI '06)*, pp. 239–246, 2006.

[40] B.-J. Theobald, I. A. Matthews, J. F. Cohn, and S. M. Boker, "Real-time expression cloning using appearance models," in *Proceedings of the 9th International Conference on Multimodal Interfaces (ICMI '07)*, pp. 134–139, 2007.

[41] L. Zalewski and S. Gong, "2D statistical models of facial expressions for realistic 3D avatar animation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 2, pp. 217–222, San Diego, Calif, USA, 2005.

[42] D. Cosker, S. Roy, P. L. Rosin, and D. Marshall, "Re-mapping animation parameters between multiple types of facial model," *Lecture Notes in Computer Science*, vol. 4418, pp. 365–376, 2007.

[43] M. B. Stegmann, *Active appearance models: theory, extensions and cases*, mastersthesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2000.

[44] C. Goodall, "Procrustes methods in the statistical analysis of shape," *Journal Royal Statistical Society*, vol. 53, pp. 285–339, 1991.

[45] "Camera calibration toolbox," http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[46] S. Pigeon, "M2VTS: multi modal verification for teleservices and security applications," 1996.

[47] N. Stoiber, R. Seguier, and G. Breton, "Automatic design of a control interface for a synthetic face," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '09)*, 2009.

[48] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469–483, 1996.

[49] J. O. Rawlings, *Applied Regression Analysis*, Wadsworth and Brooks Cole, 1988.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 573924, 15 pages
doi:10.1155/2009/573924

Research Article

Face to Face: Anthropometry-Based Interactive Face Shape Modeling Using Model Priors

Yu Zhang[1] and Edmond C. Prakash[2]

[1] *Institute of High Performance Computing, 1 Fusionopolis Way, 16-16 Connexis, Singapore 138632*
[2] *Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, UK*

Correspondence should be addressed to Yu Zhang, zhangyu_luo@hotmail.com

Received 1 February 2009; Accepted 19 February 2009

Recommended by Suiping Zhou

This paper presents a new anthropometrics-based method for generating realistic, controllable face models. Our method establishes an intuitive and efficient interface to facilitate procedures for interactive 3D face modeling and editing. It takes 3D face scans as examples in order to exploit the variations presented in the real faces of individuals. The system automatically learns a model prior from the data-sets of example meshes of facial features using principal component analysis (PCA) and uses it to regulate the naturalness of synthesized faces. For each facial feature, we compute a set of anthropometric measurements to parameterize the example meshes into a measurement space. Using PCA coefficients as a compact shape representation, we formulate the face modeling problem in a scattered data interpolation framework which takes the user-specified anthropometric parameters as input. Solving the interpolation problem in a reduced subspace allows us to generate a natural face shape that satisfies the user-specified constraints. At runtime, the new face shape can be generated at an interactive rate. We demonstrate the utility of our method by presenting several applications, including analysis of facial features of subjects in different race groups, facial feature transfer, and adapting face models to a particular population group.

1. Introduction

One of the most challenging tasks in graphics modeling is to build an interactive system that allows users to model varied, realistic geometric models of human faces quickly and easily. Applications of such a system range from entertainment to communications: virtual human faces need to be generated for movies, computer games, advertisements, or other virtual environments, and facial avatars are needed for video teleconference and other instant communication programs. Some authoring tools for character modeling and animation are available (e.g., Maya [1], Poser [2], DazStudio [3], PeoplePutty [4]). In these systems, deformation settings are specified manually over the range of possible deformation for hundreds of vertices in order to achieve desired results. An infinite number of deformations exist for a given face mesh that can result in different shapes ranging from the realistic facial geometries to implausible appearances. Consequently, interactive modeling is often a tedious and complex process requiring substantial technical as well as artistic skill. This problem is compounded by the fact that the slightest deviation from real facial appearance can be immediately perceived as wrong by the most casual viewer. While the exiting systems have exquisite control rigs to provide detailed control, these controls are based on general modeling techniques such as point morphing or free-form deformations, and therefore lack intuition and accessibility for novices. Users often face a considerable learning curve to understand and use such control rigs.

To address the lack of intuition in current modeling systems, we aim to leverage the anthropometrical measurements as control rigs for 3D face modeling. Traditionally, anthropometry—the study of human body measurement—characterizes the human face using linear distance measures between anatomical landmarks or circumferences at predefined locations [5]. The anthropometrical parameters provide a familiar interface while still providing a high level of control to users. While this is a compact description, they do not uniquely specify the shape of the human face. Furthermore, particularly for computer face modeling,

the sparse anthropometric measurements taken at a small number of landmarks on the face do not capture the detailed shape variations needed for realism. The desire is to map such sparse data into a fully reconstructed 3D surface model. Our goal is a system that uses model priors learned from prerecorded facial shape data to create natural facial shapes that match anthropometric constraints specified by the user. The system can be used to generate a complete surface mesh given only a succinct specification of the desired shape, and it can be used by expert and novice alike to create synthetic 3D faces for myriad uses.

1.1. Background and Previous Work. A large body of literature on modeling and animating faces has been published in the last three decades. A good overview can be found in the textbook [6] and in the survey [7]. In this work, we focus on modeling static face geometry. In this context, several approaches have been proposed. They can be roughly classified into the creative approach and the reconstructive approach.

The creative approach is to facilitate manual specification of the new face model by a user. Parametric face models [8–11] and many commercial modelers fall into this approach. The desire is to create an encapsulated model that can generate a wide range of faces based on a small set of input parameters. They provide full control over the result, including the ability to produce cartoon effects and the high efficiency of geometric manipulation. However, manual parameter tuning without geometric constraints from real human faces for generating realistic faces is difficult and time-consuming. Moreover, the choice of the parameter set depends on the face mesh topology and therefore the manual association of a group of vertices to a specific parameter is required.

The reconstructive approach is to extract face geometry from the measurement of a living subject. The reconstructive approach is to extract face geometry from the measurement of a living subject. In this category, the image-based technique [12–18] utilizes an existing 3D face model and information from few pictures (or video streams) for the reconstruction of face geometry. Although this kind of technique can provide reconstructed face models easily, its drawbacks are the inaccurate geometry reconstruction and inability to generate new faces that have no image counterparts. Another limiting factor of this technique lies in that it gives very little control to the user.

With a significant increase in the quality and availability of 3D capture methods, a common approach towards creating face models uses laser range scanners to acquire both the face geometry and texture simultaneously [19–22]. Although the acquired face data is highly accurate, unfortunately, substantial effort is needed to process the noisy and incomplete data into a model suitable for modeling or animation. In addition, the result of this effort is a model corresponding to a single individual; and each new face must be found on a subject. The desired face may not even physically exist. Furthermore, the user does not have any control over the captured model to edit it in a way that produces a novel face.

Besides these approaches, DeCarlo et al. [23] construct a range of face models with realistic proportions using a variationally constrained optimization technique. However, without the use of the model priors, their method cannot generate natural models unless the user accurately specifies a very detailed set of constraints. Also, this approach requires minutes of computation for the optimization process to generate a face model. Blanz and Vetter [24] present a process for estimating the shape of a face from a single photograph. This is extended by Blanz et al. [25], who present a set of controls for intuitive manipulation of facial attributes. In contrast to our work, they manually assign attribute values to characterize the face shape, and devise attribute controls using linear regression. Vlasic et al. [26] use multilinear face models to study and synthesize variations in faces along several axes, such as identity and expression. An interface for gradient-based face space navigation has been proposed in [27]. Principal components that are not intuitive to users are used as navigation axes in face space, and facial features cannot be controlled individually. The authors focus on a comparison of different user interfaces.

Several commercial systems for generating composite facial images are available [28–30]. Although they are effective to use, a 2D face composite still lacks some of the advantages of a 3D model, such as the complete freedom of viewpoint and the ability to be combined with other 3D graphics. Additionally, to our knowledge, no commercial 2D composite system available today supports automatic completion of unspecified facial regions according to statistical properties. FaceGen 3 [31] is the only existing system that we have found to be similar to ours in functionality. However, there is not much information available about how this function is achieved. As far as we know, it is built on [24] and the face mesh is not divided into different independent regions for localized deformation. In consequence, editing operations on individual facial features tend to affect the whole face.

1.2. Our Approach. In this paper, we present a new method for interactively generating facial models from user-specified anthropometric parameters while matching the statistical properties of a database of scanned models. Figure 1 shows a block diagram of the system architecture. We use a three-step model fitting approach for the 3D registration problem. By bringing scanned models into full correspondence with each other, the shape variation is represented by using principal component analysis (PCA), which induces a low-dimensional subspace of facial feature shapes. We explore the space of probable facial feature shapes using high-level control parameters. We parameterize the example models using the face anthropometric measurements, and predefine the interpolation functions for the parameterized example models. At runtime, the interpolation functions are evaluated to efficiently generate the appropriate feature shapes by taking the anthropometric parameters as input. Apart from an initial tuning of feature point positions, our method works fully automatically. We evaluate the performance of our method with cross-validation tests. We also compare our method against optimization in the PCA

subspace for generating facial feature shapes from constraints of the ground truth data.

In addition, the anthropometric-based face synthesis method, combined with our database of statistics for a large number of subjects, opens ground for a variety of applications. Chief among these is analysis of facial features of different races. Second, the user can transfer facial feature(s) from one individual to another. This allows a plausible new face to be quickly generated by composing different features from multiple faces in the database. Third, the user can adapt the face model to a particular population group by synthesizing characteristic facial features from extracted statistics. Finally, our method allows for compression of data, enabling us to share statistics with the research community for further study of faces.

Unlike a previous approach [23], we utilize the prior knowledge of the face shape in relation to the given measurements to regulate the naturalness of modeled faces. Moreover, our method efficiently generates a new face with the desired shape within a second. Our method also differs significantly from the approach presented in [24, 25] in several respects. First, they manually assign the attribute values to the face shape and devise attribute controls for single control using linear regression. We automatically compute the anthropometric measurements for face shape and relate several attribute controls simultaneously by learning a mapping between the anthropometric measurement space and the feature shape space through scattered data interpolation. Second, they use a 3D variant of a gradient-based optical flow algorithm to derive the point-to-point correspondence between scanned models. This approach does not work well for faces of different races or in different illumination given the inherent problem of using static textures. We present a robust method of determining correspondences that does not depend on the texture information. Third, their method tends to control the global face and requires additional constraints to restrict the effect of editing operations to a local region. In contrast, our method guarantees local control thanks to its feature-based nature.

The main contributions of our work are as follows.

(i) A general, controllable, and practical system for face modeling and editing. Our method estimates high-level control models in order to infer a particular face from intuitive input controls. As correlations between control parameters and the face shape are estimated by exploiting the real faces of individuals, our method regulates the naturalness of synthesized faces. Unspecified parts of the synthesized facial features are automatically completed according to statistical properties.

(ii) We propose a new algorithm which uses intuitive attribute parameters of facial features to navigate face space. Our system provides sets of comprehensive anthropometric parameters to easily control face shape characteristics, taking into account the physical structure of real faces.

(iii) A robust, automatic model fitting approach for establishing correspondences between scanned models.

(iv) The automatic runtime synthesis is efficient in time complexity and performs fast.

The remainder of this paper is organized as follows: Section 2 presents the face data we use. Section 3 elaborates on the model fitting technique. Section 4 describes the construction of local shape spaces. The face anthropometric parameters used in our work are illustrated in Section 5. Section 6 and Section 7 describe our techniques of feature-based shape synthesis and subregion blending, respectively. After presenting and explaining the results in Section 8, we present a variety of applications of our approach in Section 9. Section 10 gives concluding remarks and our future work.

2. Scanned Data and Preprocessing

We use the USF face database [32] that consists of Cyberware face scans of 186 subjects with a mixture of gender, race, and age. The age of the subjects ranges from 17 to 68 years, and there are 126 male and 60 female subjects. Most of the subjects are Caucasians (129), with African-Americans making up the second largest group (37), and Asians the smallest group (20). All faces are without makeup and accessories. The laser scans provide face structure data which contains approximately 180 k surface points and a 360×524 reflectance (RGB) image for texture-mapping (see Figures 2(a) and 2(b)). We also use a generic head model which consists of 1.092 vertices and 2.274 triangles. Prescribed colors are added to each triangle to form a smooth-shaded surface (see Figure 2(c)).

Let each 3D face scan in the database be S_i $(i = 1, \ldots, M)$. Since the number of vertices in S_i varies, we resample all faces in the database so that they have the same number of vertices all in mutual correspondence. Feature points are identified semi-automatically to guide the resampling. Figure 3 depicts the process. As illustrated in Figure 3(a), a 2D *feature mask* consisting of polylines groups a set of 86 feature points that correspond to the feature point sets of MPEG-4 Facial Definition Parameters (FDPs) [33]. The feature mask is superimposed onto the front-view face image obtained by orthographic projection of a textured 3D face scan into an image plane. The facial features in this image are identified by using the Active Shape Models (ASMs) [34] and the feature mask is fitted to the features automatically. The 2D feature mask can be manipulated interactively. A little user interaction is needed to tune the feature point positions due to the slight inaccuracy of the automatic facial feature detection. But this is restricted to slight corrections of wayward feature points. The 3D positions of the feature points on the scanned surface are then recovered by back-projection to the 3D space. In this way, we efficiently define a set of feature points on a scanned model S_i as $U_i = \{\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,n}\}$, where $n = 86$. Our generic model G is already tagged with the corresponding set of feature points $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ by default.

3. Model Fitting

3.1. Global Warping. The problem of deriving full correspondence for all models S_i can be stated as: resample the

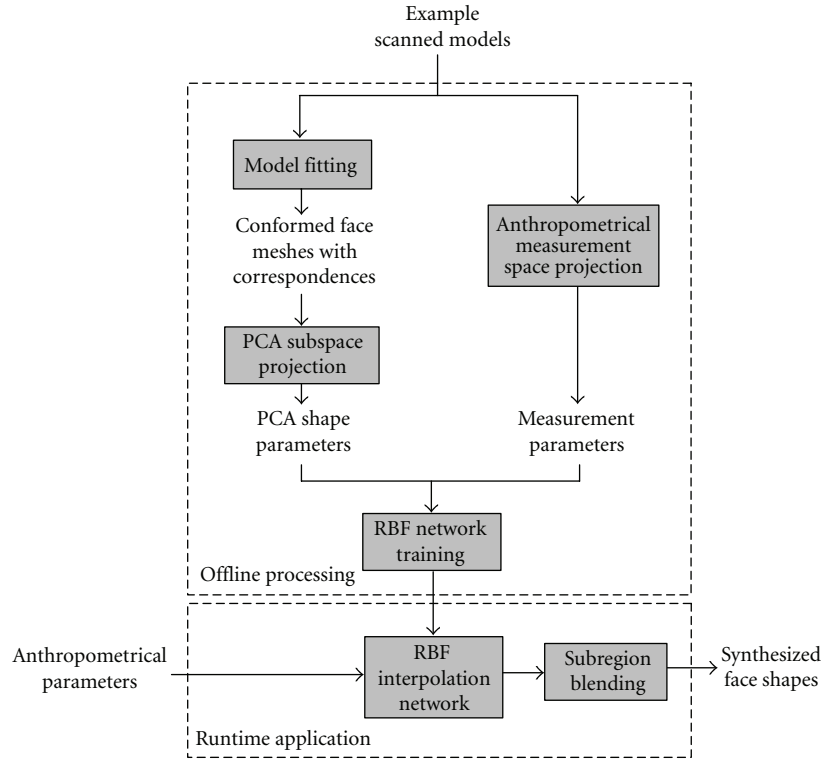
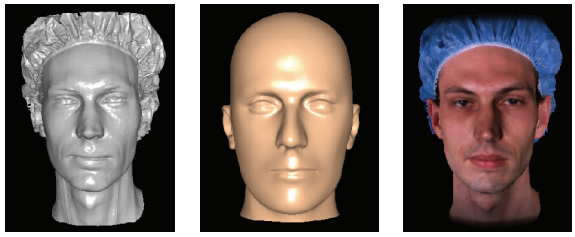


Figure 1: Overview of the interactive face shape synthesis system.



(a) (b) (c)

Figure 2: Face data: (a) scanned face geometry; (b) texture-mapped face scan; (c) generic model.

surface for all S_i using G under the constraint that $\mathbf{v}_j$ is mapped to $\mathbf{u}_{i,j}$. The displacement vector $\mathbf{d}_{i,j} = \mathbf{u}_{i,j} - \mathbf{v}_j$ is known for each feature point $\mathbf{v}_j$ on the generic model and $\mathbf{u}_{i,j}$ on the scanned surface. These displacements are utilized to construct the interpolating function that returns the displacement for each generic mesh vertex:

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{w}_j \phi_j\left(\left\|\mathbf{x} - \mathbf{v}_j\right\|\right) + \mathbf{M}\mathbf{x} + \mathbf{t}, \tag{1}$$

where $\mathbf{x} \in \mathcal{R}^3$ is a vertex on the generic model, $\|\cdot\|$ denotes the Euclidean norm and ϕ is a radial basis function. $\mathbf{w}_j$, $\mathbf{M}$ and $\mathbf{t}$ are the unknown parameters. Among them, $\mathbf{w}_j \in \mathcal{R}^3$ are the interpolation weights, $\mathbf{M} \in \mathcal{R}^{3\times 3}$ represents rotation and scaling transformations, and $\mathbf{t} \in \mathcal{R}^3$ represents translation transformation.

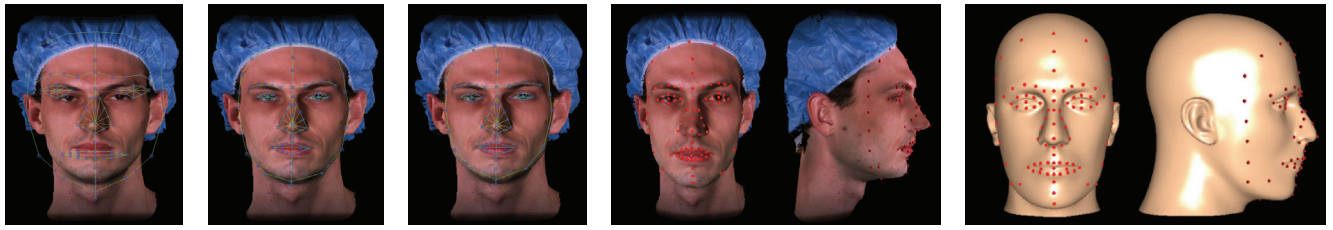
Different functions for $\phi(r)$ are available [35]. We had better results with the multi-quadric function $\phi(r) = \sqrt{r^2 + \rho^2}$, where ρ is the locality parameter used to control how the basis function is influenced by neighboring feature points. ρ is determined as the Euclidean distance to the nearest other feature point. To determine the weights $\mathbf{w}_j$ and the affine transformation parameters $\mathbf{M}$ and $\mathbf{t}$, we solve the following equations:

$$\mathbf{d}_{i,j} = \mathbf{f}\left(\mathbf{v}_j\right)|_{j=1}^{n}, \qquad \sum_{j=1}^{n} \mathbf{w}_j = 0, \qquad \sum_{j=1}^{n} \mathbf{w}_j^T \mathbf{v}_j = 0. \tag{2}$$

This system of linear equations is solved using the LU decomposition to obtain the unknown parameters. Using the predefined interpolation function as given in (1), we calculate the displacement vectors of all vertices to deform the generic model.

3.2. Local Deformation. The warping with a small set of correspondences does not produce a perfect surface match. We further improve the shape using a local deformation which fits the globally warped generic mesh $\widetilde{G}$ to the scanned model S_i by iteratively minimizing the distance from the vertices of $\widetilde{G}$ to the surface of S_i. To optimize the positions of vertices of $\widetilde{G}$, the local deformation process minimizes an energy function:

$$E\left(\widetilde{G}\right) = E_{\text{ext}}\left(\widetilde{G}, S_i\right) + E_{\text{int}}\left(\widetilde{G}\right) \tag{3}$$



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

Figure 3: Semi-automatic feature point identification: (a) initial outline of the feature mask; (b) after automatic facial feature detection; (c) after interactive tuning; (d) and (e) 3D feature points identified on the scanned model and the generic model.

(a) (b) (c)

Figure 4: Model fitting: (a) deformed generic mesh after model fitting; (b) scanned model; (e) texture mapping of the deformed generic mesh.

where E_{ext} stands for the external energy and E_{int} the internal energy.

The external energy term E_{ext} attracts the vertices of $\widetilde{G}$ to their closest compatible points on S_i. It is defined as

$$E_{\text{ext}}\left(\widetilde{G}, S_i\right) = \sum_{j=1}^{N_G} \zeta_j \left\|\mathbf{x}_j - \mathbf{s}_j\right\|^2, \tag{4}$$

where N_G is the number of vertices on the generic mesh, $\mathbf{x}_j$ is the jth mesh vertex, and $\mathbf{s}_j$ is the closest compatible point of $\mathbf{x}_j$ on S_i. The weights ζ_j measure the compatibility of the points on $\widetilde{G}$ and S_i. As $\widetilde{G}$ closely approximates S_i in the global warping procedure, we consider a vertex on $\widetilde{G}$ and a point on S_i to be highly compatible if the surface normals at each point have similar directions. Hence, we define ζ_j as:

$$\zeta_j = \begin{cases} \mathbf{n}\left(\mathbf{x}_j\right) \cdot \mathbf{n}\left(\mathbf{s}_j\right) & \text{if } \mathbf{n}\left(\mathbf{x}_j\right) \cdot \mathbf{n}\left(\mathbf{s}_j\right) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathbf{n}(\mathbf{x}_j)$ and $\mathbf{n}(\mathbf{s}_j)$ are the surface normals at $\mathbf{x}_j$ and $\mathbf{s}_j$, respectively. In this way, dissimilar local surface patches are less likely to be matched, for example, front-facing surfaces will not be matched to back-facing surfaces. To accelerate the minimum-distance calculation, we precompute a hierarchical bounding box structure for S_i so that the closest triangles are checked first.

The transformations applied to the vertices within a region of the surface may differ from each other considerably, resulting in a non-smoothly deformed surface. To enforce local smoothness of the mesh, the internal energy term E_{int} is introduced as follows:

$$E_{\text{int}}\left(\widetilde{G}\right) = \sum_{j=1}^{N_G} \sum_{k \in \Omega_j} \left\|\left(\mathbf{x}_j - \mathbf{x}_k\right) - \left(\widetilde{\mathbf{x}}_j - \widetilde{\mathbf{x}}_k\right)\right\|^2, \tag{6}$$

where Ω_j is the set grouping all neighboring vertices $\mathbf{x}_k$ that are linked by edges to $\mathbf{x}_j$, and $\widetilde{\mathbf{x}}_j$ and $\widetilde{\mathbf{x}}_k$ are the original positions of $\mathbf{x}_j$ and $\mathbf{x}_k$ before local deformation. Including this energy term constrains the deformation of the generic mesh and keeps the optimization from converging to a solution far from the initial configuration.

Minimizing $E(\widetilde{G})$ is a nonlinear least-square problem and optimization is performed using L-BFGS-B, which is a quasi-Newtonian solver [36]. The optimization stops when the difference between E at the previous and current iterations drops below a user-specified threshold. After the local deformation, each mesh vertex takes texture coordinates associated with its closest scanned data point for texture mapping. Finally, we reconstruct surface details in a hierarchical manner by taking advantage of the quaternary subdivision scheme and normal mesh representation [37]. Figure 4 shows the results of model fitting. Hence, a spatial correspondence is established by the generated normal meshes.

4. Forming Feature Shape Spaces

We perceive the face as a set of features. In this work, the global face shape is also regarded as a feature. Since all face scans are in correspondence through mapping onto the generic model, it is sufficient to define the feature regions on the generic model. We manually partition the generic model into four regions: eyes, nose, mouth and chin. This segmentation is transferred to all normal meshes to generate individualized feature shapes with correspondences (see Figure 5). In order to isolate the shape variation from the position variation, we normalize all scanned models with respect to the rotation and translation of the face before the model fitting process.

We form a shape space for each facial feature using PCA. Given the set $\Gamma = \{F\}$ of features, let $\{F_i\}_{i=1,\ldots,M}$ be a set of example meshes of a feature F, each mesh being associated to one of the M scanned models in the database. These meshes are represented as vectors that contain the x, y, z

Figure 5: Four facial features decomposed from the level 2 normal mesh.

coordinates of N vertices $F_i = (x_1^i, y_1^i, z_1^i, \ldots, x_N^i, y_N^i, z_N^i) \in \mathcal{R}^{3N}$. The average over M example meshes is given by $\psi_0 = (1/M)\sum_{i=1}^{M} F_i$. Each example mesh differs from the average by the vector $dF_i = F_i - \psi_0$. We arrange the deviation vectors into a matrix $\mathbf{C} = [dF_1, dF_2, \ldots, dF_M] \in \mathcal{R}^{3N\times M}$. PCA of the matrix $\mathbf{C}$ yields a set of M non-correlated eigenvectors ψ_i and their corresponding eigenvalues λ_i. The eigenevectors are sorted according to the decreasing order of their eigenvalues. Every example model can be regenerated using (7).

$$F_i(\boldsymbol{\alpha}) = \psi_0 + \sum_{j=1}^{K} \alpha_{ij}\psi_j, \tag{7}$$

where $0 < K < M$ and $\alpha_{ij} = (F_i - \psi_0)\cdot\psi_j$ are the coordinates of the example mesh in terms of the reduced eigenvector basis. We choose the K such that $\sum_{i=1}^{K}\lambda_i \geq \tau\sum_{i=1}^{M}\lambda_i$, where τ defines the proportion of the total shape variation (98% in our experiments). In this model each eigenvector is a coordinate axis. We call these axes eigenmeshes.

5. Anthropometric Parameters

Face anthropometry provides a set of meaningful measurements or shape parameters that allow the most complete control over the shape of the face. Farkas [5] describes a widely used set of measurements to characterize the human face. The measurements are taken between the landmark points defined in terms of visually-identifiable or palpable features on the subject face using carefully specified procedures and measuring instruments. Such measurements use a total of 47 landmark points for describing the face. As described in Section 2, each example in our face scan database is equipped with 86 landmarks. Following the conventions laid out in [5], we have chosen a subset of 38 landmarks for anthropometric measurements (see Figure 6).

Farkas [5] describes a total of 132 measurements on the face and head. Instead of supporting all 132 measurements, we are only concerned with those related to five facial features (including global face outline). In this paper, 68 anthropometric measurements are chosen as shape control parameters. As an example, Table 1 lists the nasal measurements used in our work. The example models are placed in the standard posture for anthropometric measurements. In particular, the axial distances correspond to the x, y, and z axes of the world coordinate system. Such a systematic collection of anthropometric measurements is taken through all example models in the database to determine their locations in a multi-dimensional measurement space.

6. Feature Shape Synthesis

From the previous stage we obtain a set of examples of each facial feature with measured shape characteristics, each of them consisting of the same set of dimensions, where every dimension is an anthropometric measurement. The example measurements are normalized. Generally, we assume that an example model F_i of feature F has m dimensions, where each dimension is represented by a value in the interval (0,1]. A value of 1 corresponds to the maximum measurement value of the dimension. The measurements of F_i can then be represented by the vector

$$\mathbf{q}_i = [q_{i1}, \ldots, q_{im}], \quad \forall j \in [1, m] : q_{ij} \in (0, 1]. \tag{8}$$

This is equivalent to projecting each example model F_i into a *measurement space* spanned by the m selected anthropometric measurements. The location of F_i in this space is $\mathbf{q}_i$.

With the input shape control thus parameterized, our goal is to generate a new deformation of the facial feature by computing the corresponding eigenmesh coordinates with control through the measurement parameters. Given an arbitrary input measurement vector $\mathbf{q}$ in the measurement space, such controlled deformation should interpolate the example models. To do this we interpolate the eigenmesh coordinates of the example models and obtain smooth range over the measurement space. Our feature shape synthesis problem is thus transformed to a scattered data interpolation problem. Again, the RBFs are employed. Given the input anthropometric control parameters, a novel output model with the desired shapes of facial features is obtained in runtime by blending the example models. Figure 7 illustrates this process. Our scheme first evaluates the predefined RBFs at the input measurement vector and then computes the eigenmesh coordinates by blending those of the example models with respect to the produced RBF values and precomputed weight values. Finally, the output model with the desired feature shape is generated by evaluating the shape reconstruction model (7) at those eigenmesh coordinates. Note that there exist as many RBF-based interpolation functions as the number of eigenmeshes.

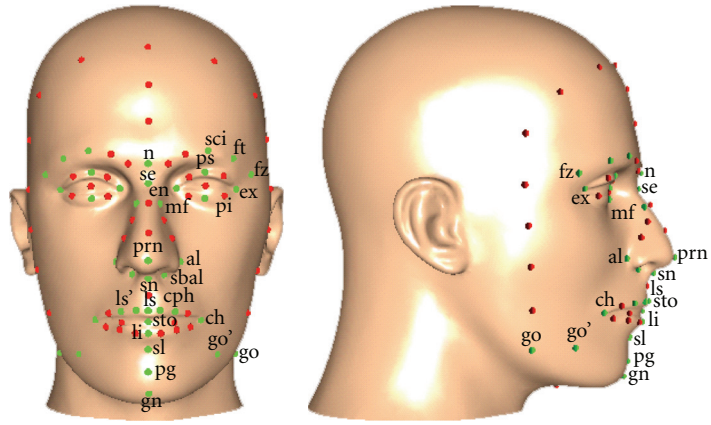
The interpolation is multi-dimensional. Consider a $\mathcal{R}^{\mathrm{m}} \rightarrow \mathcal{R}$ mapping, the interpolated eigenmesh coordinates $a_j(\cdot) \in \mathcal{R}$, $1 \leq j \leq K$ at an input measurement vector $\mathbf{q} \in \mathcal{R}^{\mathrm{m}}$ are computed as

$$a_j(\mathbf{q}) = \sum_{i=1}^{M} \gamma_{ij} R_i(\mathbf{q}) \quad \text{for } 1 \leq j \leq K, \tag{9}$$

where $\gamma_{ij} \in \mathcal{R}$ are the radial coefficients and M is the number of example models. Let $\mathbf{q}_i$ $(1 \leq i \leq M)$ be the measurement vector of an example model. The radial basis function $R_i(\mathbf{q})$ is a multi-quadric function of the Euclidean distance between $\mathbf{q}$ and $\mathbf{q}_i$ in the measurement space:

$$R_i(\mathbf{q}) = \sqrt{\|\mathbf{q} - \mathbf{q}_i\|^2 + \rho_i^2} \quad \text{for } 1 \leq i \leq M, \tag{10}$$

where ρ_i is the locality parameter used to control the behavior of the basis function and determined as the



(a) (b)

Figure 6: Head geometry with anthropometric landmarks (green dots). The landmark names are taken from [5].

Table 1: Anthropometric measurements of the nose.

Landmarks	Measurement Name	Landmarks	Measurement Name
mf-mf	Nasal root width	n-pm	Nasal bridge length
al-al	Nose width	aI-pm	Ala surface length
sbal-sbal	Alar base width	al-sn	Alar point-subnasale length
sbal-sn	Nostril floor width	n-pm	Inclination of the nasal bridge
sn-pm	Nasal tip protrusion	sn-prn	Inclination of the columella
en-se	Nasal root depth	aI-pm	Inclination of the alar-slope line
en-se	Nasal root slope	n-se-pm	Nasofrontal angle
aI-pm	Ala length	al-pm-al	Ala-slope angle
al-mf	Nasal bridge angle	se-pm-sn	Nasal tip angle
n-sn	Nose height	pm-sn-ls	Nasolabial angle

Euclidean distance between $\mathbf{q}$ and the closest other example measurement vector.

The jth eigenmesh coordinate of the ith example model, a_{ij}, corresponds to the measurement vector of the ith example model, $\mathbf{q}_i$. Equation (9) should be satisfied for $\mathbf{q}_i$ and a_{ij} $(1 \le i \le M)$:

$$a_{ij} = \sum_{i=1}^{M} \gamma_{ij} R_i(\mathbf{q}_i) \quad \text{for } 1 \le j \le K. \tag{11}$$

The radial coefficients γ_{ij} are obtained by solving this linear system using an LU decomposition. We can then generate the eigenmesh coordinates, hence the shape, corresponding to the input measurement vector $\mathbf{q}$ according to (9).

7. Subregion Shape Blending

After the shape interpolation procedure, the surrounding facial areas should be blended with the deformed internal facial features to generate a seamlessly smooth face mesh. The position of a vertex $\mathbf{x}_i$ in the feature region F after deformation is $\mathbf{x}'_i$. Let $\mathcal{V}$ denote the set of vertices of the head mesh. For smooth blending, positions of the subset $\overline{\mathcal{V}_F} = \mathcal{V} \setminus \mathcal{V}_F$ of vertices of $\mathcal{V}$ that are not inside the feature region should be updated with deformation of the facial features. For each vertex $\mathbf{x}_j \in \overline{\mathcal{V}_F}$, the vertex in each feature region that exerts influence on it, $\mathbf{x}^F_{k_i}$, is the one of minimal distance to it. It is desirable to use geodesic distance on the surface, rather than Euclidean distance to measure the relative positions of two mesh vertices. We adopt an approximation of the geodesic distance based on a cylindrical projection which is preferable for regions corresponding to a volumetric surface (e.g., the head). The idea is that distance between two vertices on the projected mesh in the 2D image plane is a fair approximation of geodesic distance. Thus, $\mathbf{x}^F_{k_i}$ is obtained as:

$$\left\|\mathbf{x}_j - \mathbf{x}^F_{k_i}\right\|_G \approx \min_{\{i \mid i \in \mathcal{V}_F\}} \left\|\mathbf{x}^*_j - \mathbf{x}^*_i\right\|, \tag{12}$$

where $\mathbf{x}^*_i$ and $\mathbf{x}^*_j$ are the positions of vertices on the projected mesh, and $\|\cdot\|_G$ denotes the geodesic distance. Note that the distance is measured offline in the original undeformed generic mesh. For each non-feature vertex $\mathbf{x}_j$, its position is updated in shape blending as:

$$\mathbf{x}'_j = \mathbf{x}_j + \sum_{F \in \Gamma} \exp\left(-\frac{1}{\alpha}\left\|\mathbf{x}_j - \mathbf{x}^F_{ki}\right\|_G\right)\left\|\mathbf{x}'^F_{ki} - \mathbf{x}^F_{ki}\right\|, \tag{13}$$

where Γ is the set of facial features and α controls the size of the region influenced by the blending. We set α to 1/10 of the diagonal length of the bounding box of the head model.

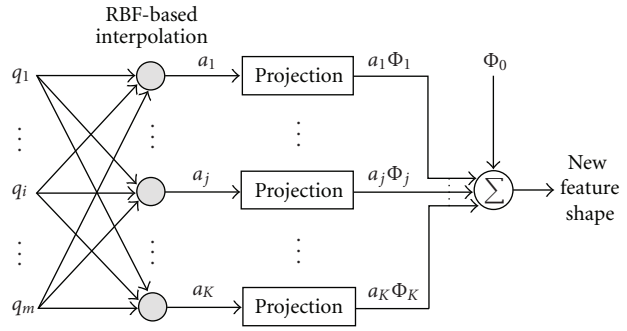


Figure 7: Generating a new facial feature shape by blending example models through interpolation of their eigenmesh coordinates.

(a) (b)

Figure 8: Synthesis of the nose shape: (a) Without shape blending, the obvious geometric discontinuities around the boundary of the nose region impair realism of the synthesis to a large extent. (b) Using our approach, the geometries of the feature region and surrounding areas are smoothly blended around their boundary.

Figure 8(b) shows the effect of our shape blending scheme employed in synthesizing the nose shape.

8. Results

Our method has been implemented in an interactive system with C++/OpenGL, where the user can select facial features to work on interactively. A GUI snapshot is shown in Figure 9. Our system starts with a mean model which is computed as the average of 186 meshes of the RBF-warped models and textured with the mean cylindrical full-head texture image [38]. Our system also allows the user to select the desired feature(s) from a database of pre-constructed typical features, which are shown in the small icons on the upper-left of the GUI. Upon selecting a feature from the database, the feature will be imported seamlessly into the displayed head model and can be further edited if needed. The slider positions for each of the available feature in the database are stored by the system so that their configuration can be restored whenever the feature is chosen. Such a feature importing mode enables coarse-to-fine modification of features, making the face synthesis process less tedious. We invited several student users who naturally lack the graphics professional's modeling background to create face models using our system. The laymen appreciated the intuitiveness and continuous variability of the control sliders. Table 2 shows the details of the datasets.

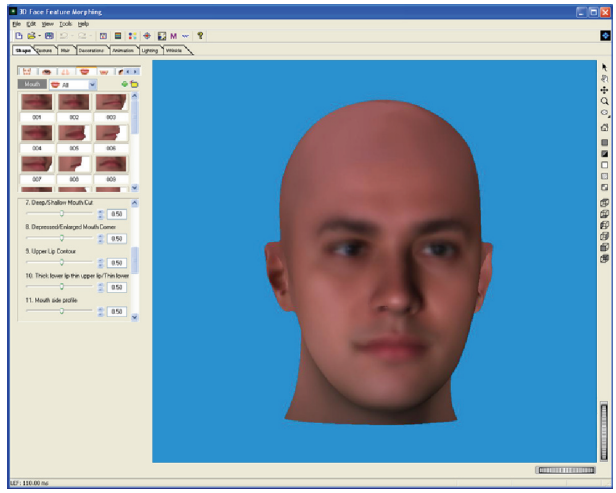


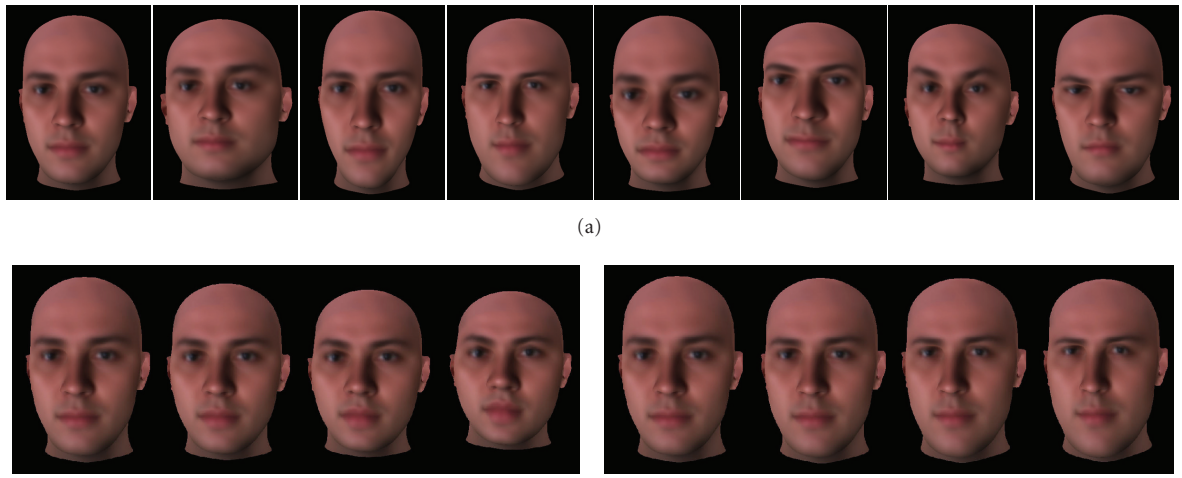
Figure 9: GUI of our system.

Table 2: Details of the data used in our system. M is the number of examples, N is the number of mesh vertices (the number of original dimensions equals $3N$), K is the number of reduced dimensions of the PCA space, and m is the number of anthropometric control parameters.

	Full head	Eyes	Nose	Mouth	Chin
M	186	186	186	186	186
N	16192	2914	1782	2105	643
K	34	23	26	20	18
m	16	13	20	12	7

Figure 10 illustrates a number of distinct facial shapes synthesized to satisfy user-specified local shape constraints. Clear differences are found in the width of the nose alar wings, the straightness of the nose bridge, the inclination of the nose tip, the roundness of eyes, the distance between eyebrows and eyes, the thickness of mouth lips, the shape of the lip line, the sharpness of the chin, and so forth. A morphing can be generated by varying the shape parameters continuously, as shown in Figures 10(b) and 10(c). In addition to starting with the mean model, the user may also select the desired head model of a specific person from the example database for further editing. Figure 11 illustrates face editing results on the models of two individuals for various user-intended characteristics.

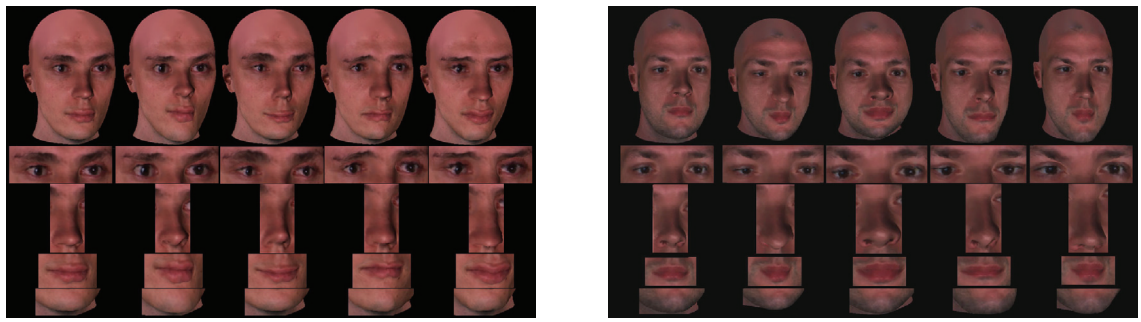
In order to quantify the performance, we arbitrarily selected ten examples in the database for the cross validation. Each example has been excluded from the example database in training the face synthesis system and its shape measurements were used as a test input to the system. The output model was then compared against the original model. Figure 12 shows a visual comparison of the result. We assess the reconstruction by measuring the maximum, mean, and root mean square (RMS) errors from the feature regions of the output model to those of the input model. The 3D errors are computed by the Euclidean distance between each vertex of the ground truth and synthesized model. Table 3 shows the average errors measured for the ten reconstructed models.



(a)

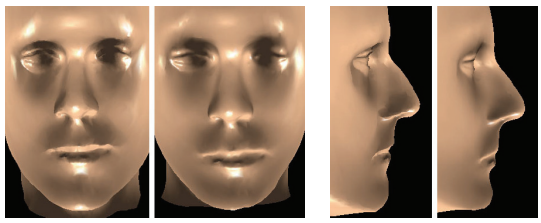
(b) (c)

Figure 10: (a) New faces synthesized from the average model (leftmost) with global and local shape variations. (b) and (c) Face shape morphing (left to right in each example).



(a) (b)

Figure 11: Feature-based face editing on the models of two individuals. In each example, the original model is shown in the top-left.



(a) (b)

Figure 12: Comparison of an original model (left in each view) and synthesized model (right in each view) in cross validation.

The errors are given using both absolute measures (/mm) and as a percentage of the diameter of the output head model bounding box.

We compare our method against the approach of optimization in the PCA space (Opt-PCA). Opt-PCA performs optimization to estimate weights of the eigen-model (7). It starts from the mean model on which the anthropometric landmarks are in their source positions. The corresponding target positions of these landmarks are the landmark positions on the example model. We then optimize the mesh shape in the subspaces of facial features using the downhill simplex algorithm such that the sum of distances between the source and target positions of all landmarks is minimized. Table 4 shows the comparison between our method and Opt-PCA. Opt-PCA produces a large error since the number of landmarks is small and it is not sufficient to fully determine weights of the eigen-model. Opt-PCA is also slow since there are many PCA weights to be optimized iteratively.

Our system runs on a 2.8 GHz PC with 1 GB of RAM. Table 5 shows the time cost of different procedures. At runtime, our scheme spends less than one second in generating a new face shape upon receiving the input parameters. This includes the time for the evaluation of RBF-based interpolation functions and for shape blending around the feature region boundaries.

9. Applications

Apart from creating plausible 3D face models from users' descriptions, our feature-based face reconstruction approach is useful for a range of other applications. The statistics of facial features allow analysis of their shapes, for instance,

Table 3: Cross validation results of our 3D face synthesis system.

	Eyes	Nose	Mouth	Chin
Average max.	3.85 (0.91%)	2.55 (0.84%)	2.86 (0.94%)	4.46 (1.06%)
Average mean	2.57 (0.57%)	1.62 (0.38%)	2.04 (0.49%)	2.25 (0.53%)
Average RMS	3.62 (0.86%)	2.23 (0.53%)	2.84 (0.67%)	3.14 (0.74%)

Table 4: Comparison of our method with the optimization approach. Each value is an average of ten trials with different example models.

	Opt_PCA				Our method			
	Eyes	Nose	Mouth	Chin	Eyes	Nose	Mouth	Chin
Mean error (mm)	2.83	3.27	3.84	6.65	2.57	1.62	2.04	2.25
Time (s)	34.8	21.5	23.5	5.3	0.4	0.5	0.4	0.3

Table 5: Time consumed for different processes of system implementation. For some processes (in italic), the time spent per example is shown. Notation: time consumed in interactive operation (T_I), time consumed in automatic computation (T_A).

Process	T_I	T_A
Offline processing		
Feature point identification	3–5 minutes	6 seconds
Global warping	N/A	2 seconds
Local deformation	N/A	4 minutes
Multi-resolution model generation	N/A	5 seconds
Computing eigenmeshes by PCA	N/A	2 hours
Computing eigenmesh coordinates	N/A	0.5 seconds
Computing anthropometric measurements	N/A	0.2 seconds
LU decomposition	N/A	2 minutes
Runtime		
Feature shape synthesis	N/A	0.6 seconds

to discern differences between groups of faces. They also allow synthesis of new faces for applications such as facial feature transfer between different faces and adaptation of the model to local populations. Moreover, our approach allows for compression of 3D face data, facilitating us to share statistics with other researchers to allow the synthesis and further study of high-resolution faces.

9.1. Analyzing the Shape of Facial Features. As the first application, we consider analysis of the shape of facial features. This is useful for classification of face scans. We wish to gain insight into how facial features change with personal characteristics by comparing statistics between groups of faces. We calculate the mean and standard deviation statistics of anthropometric measurements for each facial feature of different groups. The morphometric differences between groups are visualized by comparing the statistics of each facial feature in a diagram. We follow this approach to study the effects of race and gender.

Race. To investigate how the shape of facial features changes with race, we compare three groups of 18–30 year-old Caucasian (72 subjects), Mongolian (18 subjects), and Negroid (26 subjects) which are divided almost equally between the genders. The group statistics are shown in Figure 13, colored with blue, green, and red, respectively. It shows that the Caucasian nose is narrow, the Mongolian nose is medial, and the Negroid nose is wide. The statistics indicate a relatively protruding, narrow nose in Caucasian. The Mongolian nose is less protruding and wider, and the Negroid nose has the smallest protrusion. The nasal root depth and nasofrontal angle are the largest for the Caucasian, exhibiting significant differences compared with the smaller Negroid and smallest Mongolian values. This suggests the high nasal root in Caucasian and relatively flat nasal root in Negroid and Mongolian. Significant differences among the three races are also found in inclination of the columella and nasal tip angle, indicating the hooked nose in Caucasian and the snub nose in Mongolian and Negroid.

For the eyes, the main characteristics of the Caucasian group are the largest eye fissure height, the smallest intercanthal width and eye fissure inclination angle. These suggest that the Caucasian eyes typically have larger openings with horizontally aligned inner and external eye corners. The Mongolian group has the largest intercanthal width, and the greatest inclination in the shortest eye fissure and the smallest eye fissure height, which indicate the relatively small eye openings separated in a large horizontal distance with positions of the inner eye corners lower than those of the external ones. Blacks have the largest eye fissure length and binocular with, which denote the relatively wide eyes in this group.

As shown in Figure 13(c), many measurements of the mouth of Negroid (e.g., mouth width, upper and lower lip height, upper and lower vermilion height) are the largest among the three races. They are significantly different from those in Caucasian or Mongolian group. Mongolian has the relatively narrow mouth and thin lips. In Caucasian the skin portion of the upper and lower lips and their vermilion height are the smallest. However, the proportions of the upper and lower lip heights in the three races reveal the similarity.

From statistics illustrated in Figure 13(d), the Negroid chin has the characteristics of a long vertical profile dimension and small width. The smallest value of inclination of the chin from the vertical and the largest mentocervical angle also indicates a less protruding chin for Negroid. In

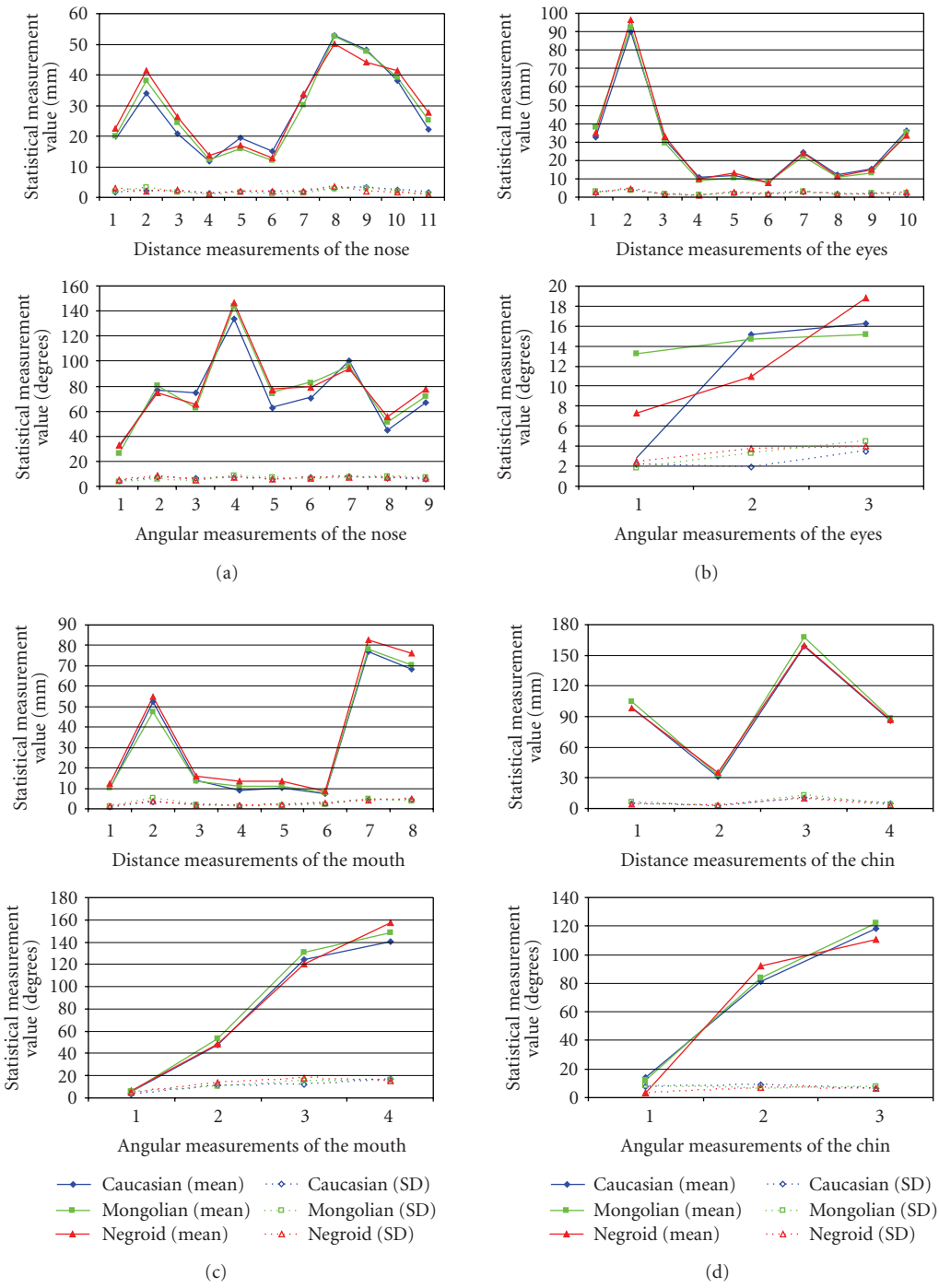


FIGURE 13: Comparison of statistics of facial feature measurements between races (blue, green and red for groups of Caucasian, Mongolian and Negroid, resp.). Each facial feature: statistics of the distance measurements (top) and statistics of the angular measurements (bottom).

Mongolian, the chin is the widest among the three races. The smallest chin height is found in Caucasian. Also, the chin of Caucasian is slightly wider than that of Negroid, but markedly narrower than that of Mongolian.

Gender. To study the effect of gender, we compare in Figure 14 18–30-year-old Caucasian females (35 subjects, in red) to Caucasian males of the same age group (37 subjects, in blue). The change of the shape of facial features from females to males is different in character from that of the change between varying racial groups. The larger values of most distance measurements of the nose indicate that males have wide alar wings and wide, long nose bridge. The value of the nasal root depth is also indicative of high upper nose bridge of the male subjects. In females, the nose bridge and alar are narrower; the nose tip is sharper and

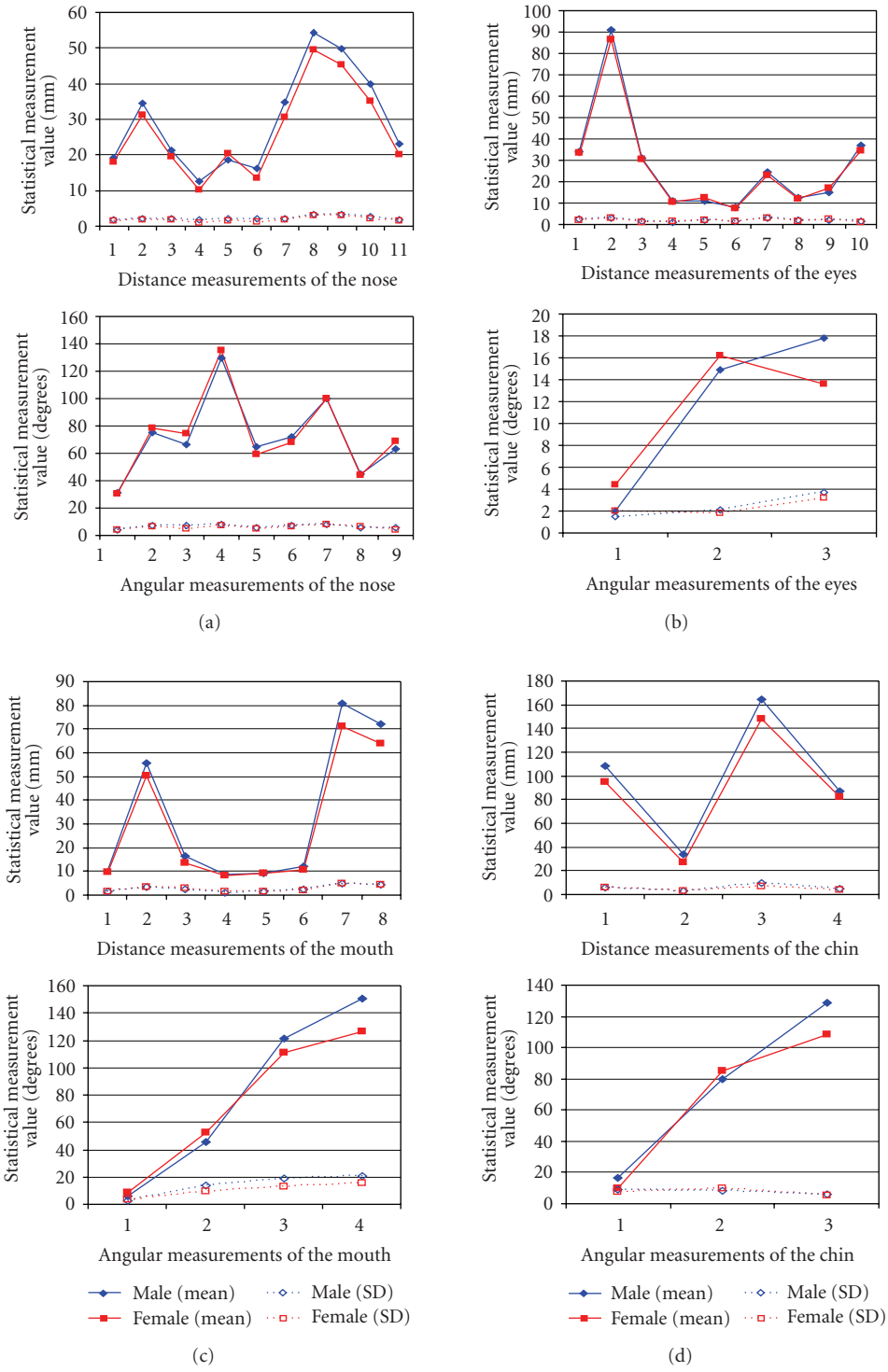


Figure 14: Comparison of statistics of facial feature measurements between genders (females in red and males in blue). Each facial feature: statistics of the distance measurements (top) and statistics of the angular measurements (bottom).

more protruding. In addition, the vertical profile around the junction of the nose bridge and the anterior surface of the forehead in females is flatter, which is suggested by the larger nasofrontal angle. The inclination of the nose bridge and columella reveals the similarity in two genders.

Regarding anthropometric measurements of the eyes, males have the larger intercanthal width and binocular width, which imply that their eyes are more separated with regard to the sagittal plane (vertical plane cutting through the center of the face). The width of the eye fissure of males

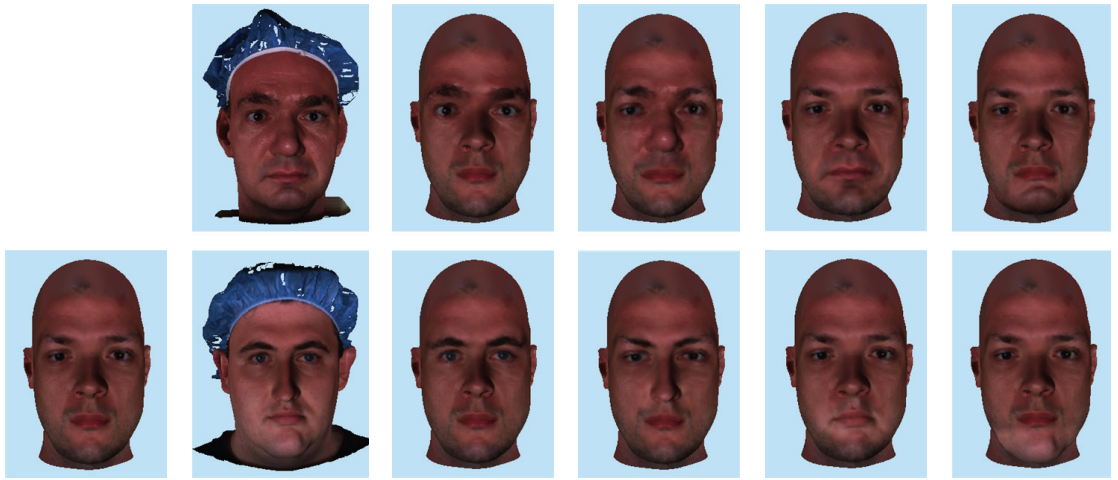


Figure 15: Transfer of facial features. We start with a source model (a) and synthesize facial features of the eyes (c), nose (d), mouth (e) and chin (f) on it by coercing the shape parameters to match those of two example faces (b).

is slightly larger than that of females, whereas the heights of the eye fissure of two genders are similar. Males also have the large height of the lower eyelid. In females, the height of the upper eyelid and distance between eyebrows and eyes are larger. Another characteristic of females is the large inclination of the eye fissure.

Most distance measurements of the mouth in the male group are larger in both genders, as shown in Figure 14(c). This suggests that males have a much wider mouth with the large skin portion of the upper and lower lips. However, the vermilion heights of the upper and lower lips in two groups reveal the similar thickness of the lips in two genders. The differences exhibited in the angular measurements are indicative of more protruding lips and convex lip line of the female subjects.

The diagram in Figure 14(d) shows that the chin of males is characterized by large size in three dimensions (width, height and depth) due to the large underlying mandible. The greater inclination angle of the chin and smaller mentocervical angle also indicate a relatively protruding chin in males compared to that of females.

9.2. Facial Feature Transfer. In the applications of creating virtual characters for entertainment production, sometimes it is desirable to adjust the face so that it has certain facial features similar to those of a particular person. Therefore, it is useful to be able to transfer desired facial feature(s) between different human subjects. One might wish, given a database of example faces, to select a face or multiple faces to which to adjust facial features.

Our high-level facial feature control framework allows the transfer of desired facial features from example faces to a source model in a straightforward manner. We can alter the feature of the source model with a feature-adjustment step which coerces the anthropometric measurement vector to match that of the target feature of an example face. The new shape of the selected feature is reconstructed on the source model and can be further edited if needed.

Figure 15(a) shows the source model which is the approximation of an example 3D scan using the deformed generic mesh. Figures 15(c) to 15(f) show the results of matching the shape measurements of the features of this model to those of two example faces shown in Figure 15(b). The synthesis keeps global shape of the source model, while transferring features of the target subject to the source subject. With decomposition of the face into local features, typical features of different target faces can be transferred in conjunction with each other to the same source model. Figure 16 shows a composite face built from facial features of four individuals.

9.3. Face Adaptation to Local Populations. Adapting the model to local populations falls neatly into our framework. The problem of automatically generating a population is reduced to the problem of generating the desired number of plausible sets of control parameters. It is convenient to generate each parameter value independently as if sampled from the Gaussian normal distribution with its mean and variance. The generated control parameter values both respect a given population distribution, and—thanks to the use of interpolation in the local feature shape spaces—produce a believable face. The examples of this process are shown in Figure 17.

9.4. Face Data Compression and Dissemination. For the face synthesis based on a large example data set, the ability to organize examples into database, compress, and efficiently transmit them is a critical issue. The example face meshes used for this paper are restricted from being transmitted in their full resolution because of their dense-data nature. In our method, we take advantage of the fact that the objects under our consideration are of the same class and that they lie in correspondence to compress data very efficiently. Instead of storing instances of geometry data for every example, we adopt a compact representation obtained by extracting the statistics with PCA, which are several orders of magnitude smaller than the original 3D scans. This accounts

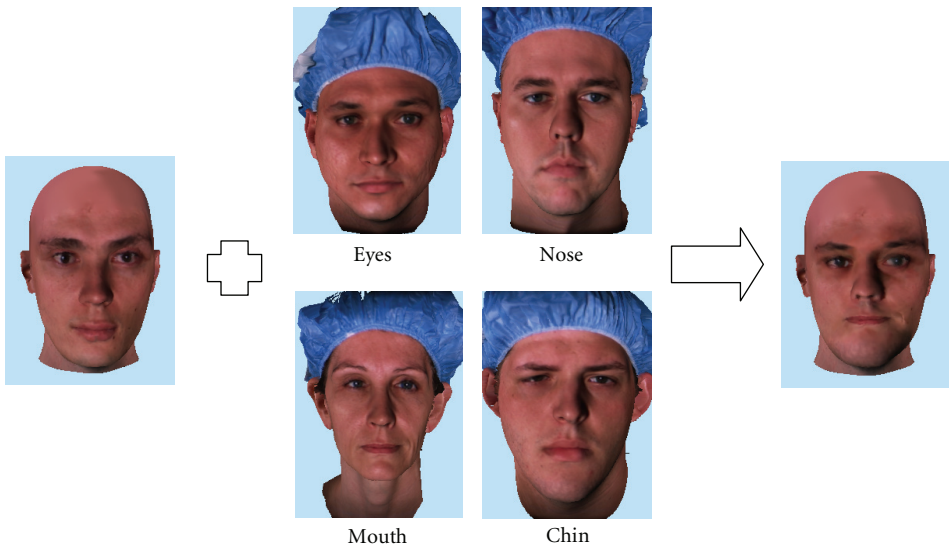


FIGURE 16: Facial features of four example faces (b) in our database are transferred to the source model (a) to generate a novel composite face (c).

FIGURE 17: Adapting the face to population groups: (a) average face; (b), (c) and (d) synthesized faces with the ethnicity of Caucasian, Mongolian and Negroid, respectively; (e) and (f) synthesized male and female faces, respectively.

for the space gain from M times the dimensionality of high-resolution 3D scans (hundreds of thousands), to K ($K \leq M$) times the dimensionality of an eigenmesh (several thousands), with M and K being the number of examples and eigenmeshes respectively. For all faces, we also make available the statistics of facial feature measurements within different population groups. These statistics along with the eigenmeshes should make it possible for other researchers to investigate new applications beyond the ones described in this paper.

10. Conclusion and Future Work

We have presented an automatic runtime system for generating varied, realistic face models. The system automatically learns a statistical model from example meshes of facial features and enforces it as a prior to generate/edit the face model. We parameterize the feature shape examples using a set of anthropometric measurements, projecting them into the measurement spaces. Solving the scattered data interpolation problem in a reduced subspace yields a natural face shape that achieves the goals specified by the user. With an intuitive slider interface, our system appeals to both beginning and professional users, and greatly reduces the time for creating natural face models compared to existing 3D mesh editing software. With the anthropometrics-based face synthesis, we explore a variety of applications, including analysis of facial features in subjects with different races, transfer of facial features between individuals, and adjusting the apparent race and gender of faces.

The quality of the generated model depends on the model priors. Therefore, an appropriate database with large number and variety of the faces must be available. We would like to extend our current database to incorporate more 3D face examples of Mongolian and Negroid races as well as to increase the diversity of age. We also plan to increase the number of facial features to choose from. To improve the system interface, we would like to integrate the "dragging" interaction mode which allows for directly choosing one or more feature points of a facial feature and then dragging

them to the desired positions to generate a new facial shape. This involves updating multiple anthropometric parameters in one step and results in large scale changes.

References

[1] "Autodesk Maya," http://www.autodesk.com/maya.

[2] "Poser 7," http://graphics.smithmicro.com/go/poser.

[3] "DazStudio," http://www.daz3d.com.

[4] "PeoplePutty," http://www.haptek.com.

[5] L. G. Farkas, *Anthropometry of the Head and Face*, Raven Press, New York, NY, USA, 1994.

[6] F. I. Parke and K. Waters, *Computer Facial Animation*, AK Peters, Wellesley, Mass, USA, 1996.

[7] J. Y. Noh and U. Neumann, "A survey of facial modeling and animation techniques," USC Technical Report 99-705, Univeristy of Southern Californina, Los Angeles, Calif, USA, 1999.

[8] S. DiPaola, "Extending the range of facial types," *Journal of Visualization and Computer Animation*, vol. 2, no. 4, pp. 129–131, 1991.

[9] N. Magnenat-Thalmann, H. T. Minh, M. de Angelis, and D. Thalmann, "Design, transformation and animation of human faces," *The Visual Computer*, vol. 5, no. 1-2, pp. 32–39, 1989.

[10] F. I. Parke, "Parameterized models for facial animation," *IEEE Computer Graphics and Applications*, vol. 2, no. 9, pp. 61–68, 1982.

[11] M. Patel and P. Willis, "Faces: the facial animation, construction and editing system," in *Proceedings of the European Computer Graphics Conference and Exhibition (Eurographics '91)*, pp. 33–45, Vienna, Austria, September 1991.

[12] T. Akimoto, Y. Suenaga, and R. S. Wallace, "Automatic creation of 3D facial models," *IEEE Computer Graphics and Application*, vol. 13, no. 5, pp. 16–22, 1993.

[13] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pp. 55–65, Orlando, Fla, USA, July 1998.

[14] C. J. Kuo, R.-S. Huang, and T.-G. Lin, "3-D facial model estimation from single front-view facial image," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 3, pp. 183–192, 2002.

[15] W.-S. Lee and N. Magnenat-Thalmann, "Fast head modeling for animation," *Image and Vision Computing*, vol. 18, no. 4, pp. 355–364, 2000.

[16] Z. Liu, Z. Zhang, C. Jacobs, and M. Cohen, "Rapid modeling of animated faces from video," *Journal of Visualization and Computer Animation*, vol. 12, no. 4, pp. 227–240, 2001.

[17] I. K. Park, H. Zhang, V. Vezhnevets, and H. K. Choh, "Image-based photorealistic 3d face modeling," in *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition (FGR '04)*, pp. 49–54, Seoul, Korea, May 2004.

[18] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pp. 75–84, Orlando, Fla, USA, July 1998.

[19] R. Enciso, J. Li, D. Fidaleo, T.-Y. Kim, J.-Y. Noh, and U. Neumann, "Synthesis of 3d faces," in *Proceedings of the 1st USF International Workshop on Digital and Computational Video (DCV '99)*, pp. 8–15, Tampa, Fla, USA, December 1999.

[20] K. Kähler, J. Haber, H. Yamauchi, and H.-P. Seidel, "Head shop: generating animated head models with anatomical structure," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 55–63, San Antonio, Tex, USA, July 2002.

[21] K. Kähler, J. Haber, and H.-P. Seidel, "Geometry-based muscle modeling for facial animation," in *Proceedings of Graphics Interface*, pp. 37–46, Ottawa, Canada, June 2001.

[22] Y. Lee, D. Terzopoulos, and K. Waters, "Realistic modeling for facial animation," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pp. 55–62, Los Angeles, Calif, USA, August 1995.

[23] D. DeCarlo, D. Metaxas, and M. Stone, "An anthropometric face model using variational techniques," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pp. 67–74, Orlando, Fla, USA, July 1998.

[24] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pp. 187–194, Los Angeles, Calif, USA, August 1999.

[25] V. Blanz, I. Albrecht, J. Haber, and H.-P. Seidel, "Creating face models from vague mental images," *Computer Graphics Forum*, vol. 25, no. 3, pp. 645–654, 2006.

[26] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," in *Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, pp. 426–433, Los Angeles, Calif, USA, July-August 2005.

[27] T.-P. G. Chen and S. Fels, "Exploring gradient-based face navigation interfaces," in *Proceedings of Graphics Interface*, pp. 65–72, London, Canada, May 2004.

[28] "PROfit^{TM} from ABM United Kingdom Ltd.," http://www.abm-uk.com.

[29] "E-FIT^{TM} from Aspley Ltd.," http://www.efit.co.uk.

[30] "Identi-Kit.NET^{TM} from Smith & Wesson®," http://www.identikit.net.

[31] "FaceGen Modeller 3.0 from Singular Inversions Inc.," http://www.FaceGen.com.

[32] "USF DARPA HumanID 3D Face Database," Courtesy of Prof. Sudeep Sarkar, University of South Florida, Tampa, Fla, USA.

[33] ISO/IEC, "Overview of the MPEG-4 standard," http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm.

[34] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[35] J. C. Carr, R. K. Beatson, J. B. Cherrie, et al., "Reconstruction and representation of 3D objects with radial basis functions," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 67–76, Los Angeles, Calif, USA, August 2001.

[36] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.

[37] I. Guskov, K. Vidimče, W. Sweldens, and P. Schroöder, "Normal meshes," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 95–102, New Orleans, La, USA, July 2000.

[38] Y. Zhang, "An efficient texture generation technique for human head cloning and morphing," in *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP '06)*, pp. 267–278, Setúbal, Portugal, February 2006.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 572030, 6 pages
doi:10.1155/2009/572030

Research Article

A New 3D Model Retrieval Method with Building Blocks

Mingquan Zhou,[1] Qingsong Huo,[2] Guohua Geng,[2] and Xiaojing Liu[3]

[1] *College of Information Science and Technology, Beijing Normal University, Beijing 100875, China*
[2] *College of Information Science and Technology, Northwest University, Xi'an 710127, China*
[3] *Department of Computer Technology and Application, Qinghai University, Xining 810016, China*

Correspondence should be addressed to Qingsong Huo, huoqingsong@126.com

Received 31 January 2009; Accepted 19 February 2009

Recommended by Suiping Zhou

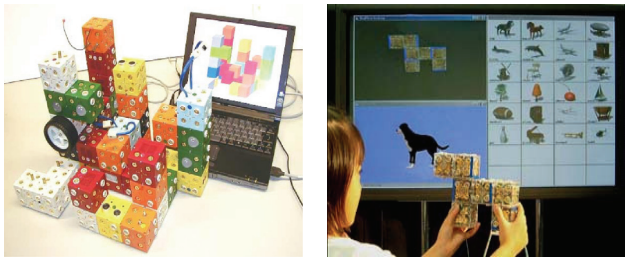
As the numbers of 3D models available grow in many application fields, there is an increasing need for a search method to help people find them. Unfortunately, traditional search techniques are not always effective for 3D data. In this paper, we describe a novel method of interactive 3D model retrieval with building blocks. First, by using a cube block as the baseblock in a 3D virtual space, we may construct the query model with human-computer interaction method. Then through retrieving the polygon model of the database generated by the voxel model, we may get retrieval results in real time. Experiments are conducted to evaluate the performance of the proposed method.

1. Introduction

With the developments of computer graphics and the progress in multimedia hardware technologies, 3D models are emerging in many application fields such as game, medicine, and molecular biology and are playing more important role in multimedia data types. 3D model has gradually become the fourth multimedia data type after voice, image, and video. Consequently, achieving effective and efficient content-based 3D model retrieval has now become a hotspot in the research of multimedia information retrieval [1]. How to search useful and the same theme of the models usefully and effectively has become the main goal of 3D model retrieval. In order to provide users with a convenient way of search, a mature retrieval system should have a good interactive performance.

Compared with images and other 2D media, 3D models contain more rich information. Therefore, 3D model of content-based retrieval system generally has a variety of query approaches [2, 3]. Some ways to retrieve of a desired model from a large selection of 3D shape models have already been proposed, for example, the text-based search method, 2D hand-drawn draft search method, 3D hand-drawn draft search method, the example of 3D models search method, and interactive retrieval of 3D shape models using physical objects [4–9].

Currently, the text-based search method is popular. It only needs to enter the keyword, but a text description is often too limited and incorrect. The most important thing is that we must artificially mark models firstly, and it is not realistic to the mass of the models data. 2D hand-drawn draft search method is to produce an effective 2D visual draft for retrieval, and Princeton University presents a draft manual mapping 2D retrieval interface [5]. Pu and Ramani [10] retrieve by drawing images. Cao et al. [11] and Fonseca et al. [12] retrieve by drawing 2D sketch according to the query model provided by the users. Although users can express their intentions by drawing 2D sketch, it improves the users' requirements. It is very difficult to transform 3D models into 2D images especially for the users who are beginners of studying computer graphics and computer-aided design. In order to input 3D models, Igarashi et al. [13] design a 3D sketching tool Teddy. Hou and Ramani [14] draw 3D part models and retrieve by cluster rules. Compared to the draft drawn 2D sketch, it has more difficulty and more restrictions for drawing 3D sketch. Moreover, the 3D sketching tool is, arguably, hard to use especially for a person who does not have a talent for painting or drawing. Examples of 3D models require users to retrieve the first sample to provide a 3D model. Users can easily operate, but without a suitable 3D model as the retrieval example, it is very difficult to carry on.



(a) (b)

Figure 1: Appearance of the ActiveCube system.

Siegl et al. [15] employ an interactive approach to teaching and retrieving by using mobile AR-kits. Ichida et al. [16] implement a query interface for retrieval of 3D shape models with physical objects by using the ActiveCube system. The appearance of the ActiveCube system is shown in Figure 1.

The query model of interactive retrieval of 3D shape models using physical objects is constructed by the users themselves, without converting into images or 3D drawing sketch. The retrieval interface is very simple, and users may participate in the query process. But the retrieval method must use physical objects as the media; otherwise, it is unable to carry on. Furthermore, it is restricted by only six surfaces of the connection and the number of physical objects. The expression of models is very weak. Is there a better retrieval way? This paper proposes a new 3D model retrieval method with building blocks.

The rest of the paper is organized as follows: 3D model retrieval method with building blocks is introduced in Section 2. Section 3 gives experimental results to show the effectiveness with the proposed search method. Section 4 presents the method of retrieval optimization. Finally, conclusions are given in Section 5.

2. 3D Model Retrieval Method with Building Blocks

Building blocks is a common Children's toy. By a few simple building blocks, it combines various characters, animals, bridges, houses, towers, and so on. This paper is inspired by the building blocks in the game and structures a virtual environment in the computer. It constructs 3D query models by building blocks, expresses the users' expression, and realizes 3D models retrieval.

2.1. Retrieval Method Introduction. As shown in Figure 2, the retrieval system sample points to 3D polygon models and builds voxel model database. Then users construct the query online model. Finally the system makes similarity computations and gets the query result by contrasting the query model with the models of voxel model database.

2.2. Generating Voxel Model. We assume that the entire 3D model is composed by the triangle mesh $T = (T_1, T_2, \ldots, T_n)$. A, B, C are the vertices. The vertex density or mesh density varies greatly. In order to effectively describe the characteristics of 3D models, this paper samples points by subdividing triangle mesh.

As shown in Figure 3, the triangle mesh (A, B, C) is to be subdivided, (B, C) is the longest side, and O_3 is the midpoint of BC. So the triangle mesh may be subdivided into two triangle meshes, and they are the triangle mesh (A, B, O_3) and the triangle mesh (A, O_3, C). If the area of the triangle meshes (A, B, O_3) and (A, O_3, C) is more than a threshold, we will continue subdividing the triangle meshes (A, B, O_3) and (A, O_3, C) as that of the triangle mesh (A, B, C). Iteration will not stop until the area of the subdivided triangle mesh is less than the threshold T. We call the center of the subdivided triangle mesh as the sampling points of the 3D model. In Figure 3, P_1, P_2, P_3, and $P_4$are sampling points. And all the sampling points of triangle meshes constitute voxel model of polygon model. Figure 4(a) is a chair model, and Figure 4(b) is a voxel model of chair model.

2.3. Construction of Query Model. This paper uses a cube as the baseblock of building blocks. We construct query model by building blocks with many baseblocks. Construction interface is shown in Figure 5. In the construction interface, each baseblock is expressed by a red cube. The red cube has 6 surfaces, 8 angles, 22 sides, and 26 kinds of connections direction. We may build blocks in each direction, and the yellow translucent cube represents the next baseblock which will be built.

The construction process of query model is as follows.

(1) Click on any red block in the interface, then we will see 26 semitransparent connected yellow baseblocks.

(2) When clicked, the semitransparent baseblock becomes red baseblock.

(3) According to the need of the constructing model, repeat (1), (2) and continue building blocks until we are satisfied.

Through the above process, we can get a satisfactory query model. Figures 6 and 7 are, respectively, the constructing process of a plane model and a stool model.

2.4. Similarity Computations. We suppose that the query model is constructed by n baseblocks according to the prior process, and we describe the details of this similar calculation procedure in the following set of steps.

(1) Bound the query model constructed by n cube-blocks with bounding box, obtain the minimum size of its bounding box, and denote with M.

(2) Pick one of polyhedral models, sample it with the method of 2.2, and obtain its voxel model and the minimum size of its bounding box, denoting with N.

(3) Assume that the smallest box M is segmented into $a \times b \times c$, and the smallest box N is also segmented into $a \times b \times c$. The segmenting progress is shown in Figure 8.

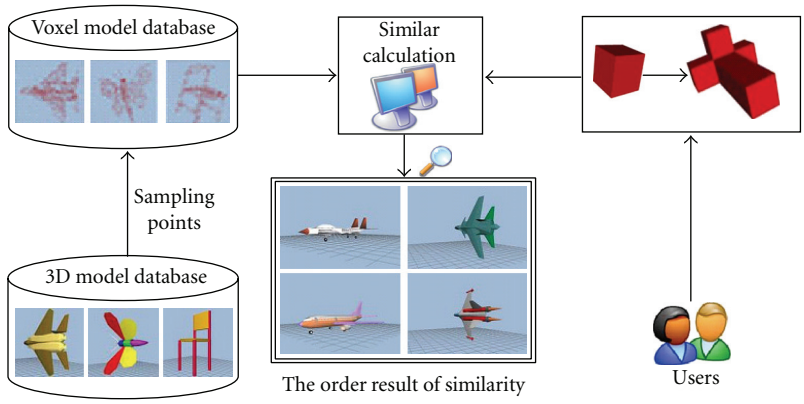


Figure 2: System flow chart.

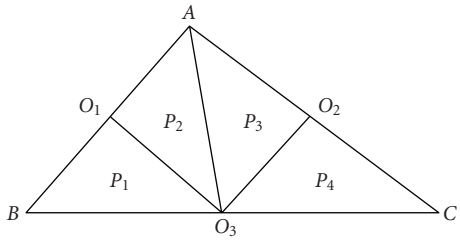
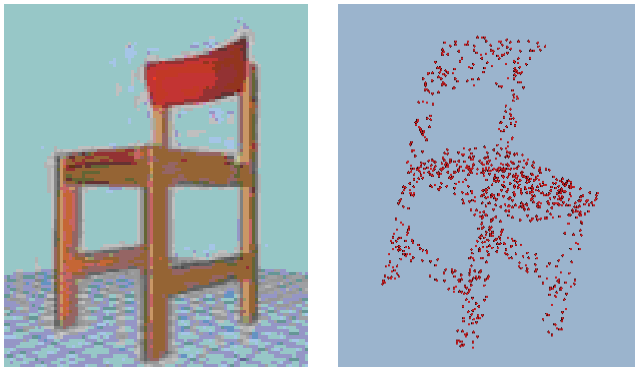


Figure 3: Triangle mesh and sampling points.



(a) (b)

Figure 4: Model chair and its voxel model.

(4) Assume the number of points of the voxel model is, respectively, $P_1, P_2, \ldots, P_n$ which fall into the baseblock. We compute the total number of points which fall into the query model in accordance with formula (1):

$$P' = \sum_{i=1}^{n} P_i, \tag{1}$$

$$\text{Sim} = \frac{P'}{P} \times 100(\%). \tag{2}$$

(5) Flip the smallest bounding box and repeat (3)–(5) until the six flip manners are all processed. Receive maximum P' and make similarity computations in accordance with formula (2).

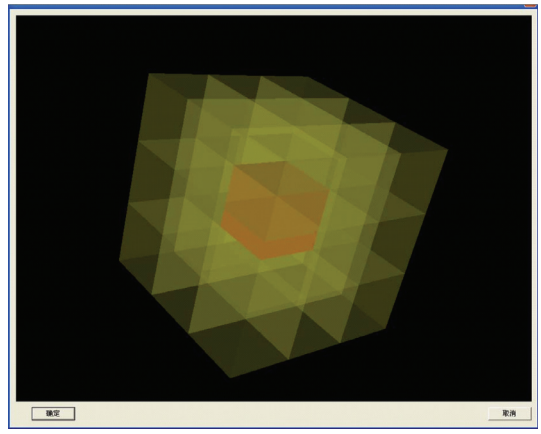


Figure 5: Query model construction interface.

(6) Repeat (2)–(5) operation until the models of the database are all processed.

(7) According to high to low arrangement order, we will be able to get retrieval results.

3. Experimental Results and Performance Evaluation

In this paper, the experimental data model is the Princeton University 3D retrieval database PSB, which contains a total of 1814 models.

3.1. Experimental Results. We construct the plane model and stool as a query model with the method as described above. Their experimental results are shown in Figures 9 and 10.

3.2. Performance Evaluation and Analysis. From Figures 9 and 10 of the retrieval results, we can see that the plane model retrieval result is better than that of the stool model. In order to further test the accuracy of retrieval results, we use recall

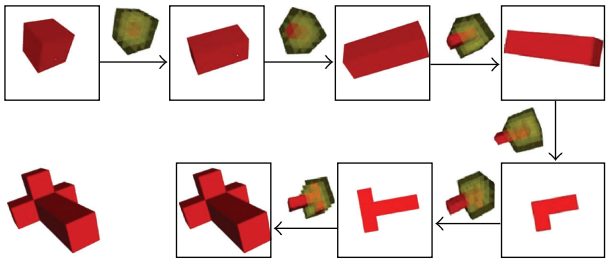


Figure 6: The constructing process of a plane model.

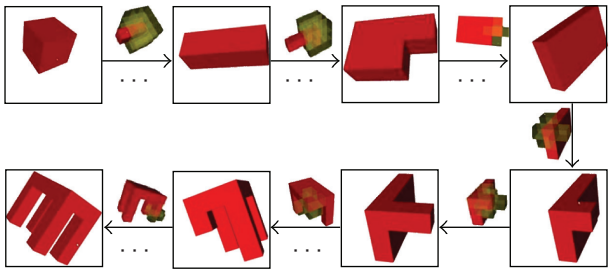


Figure 7: The constructing process of a stool model.

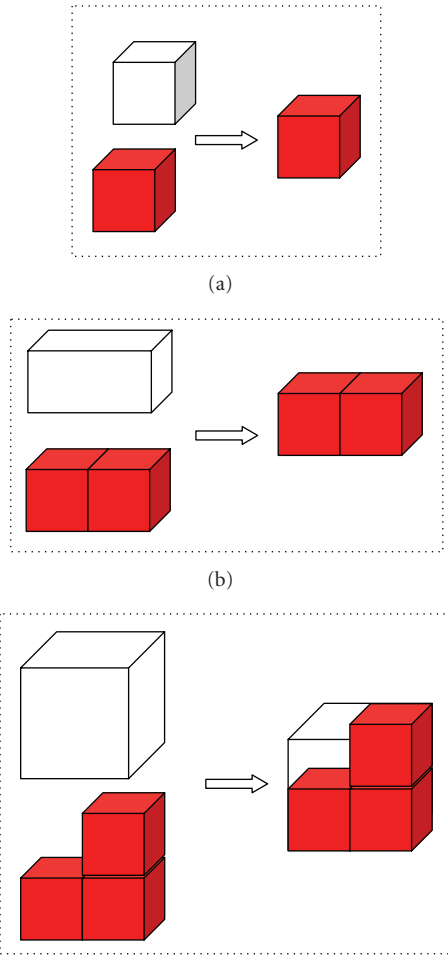


Figure 8: Model of the split-sample map.

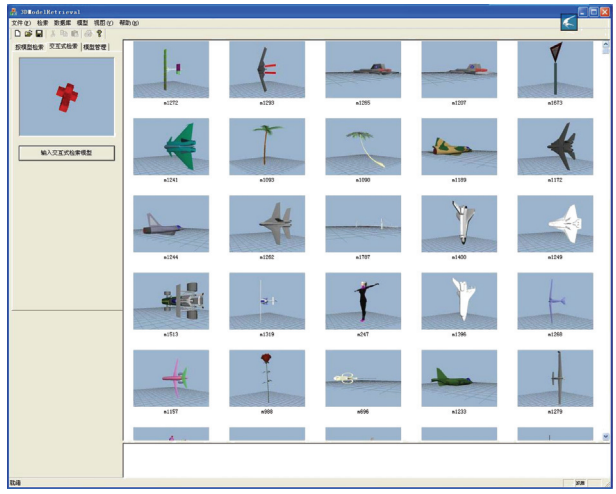


Figure 9: The plane model experimental result.

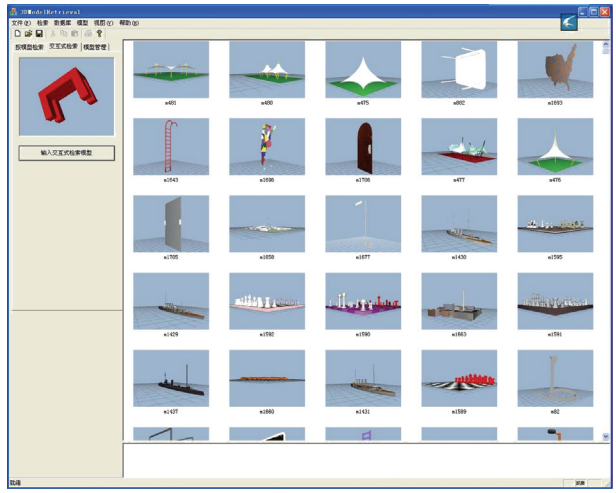


Figure 10: The stool model experimental result.

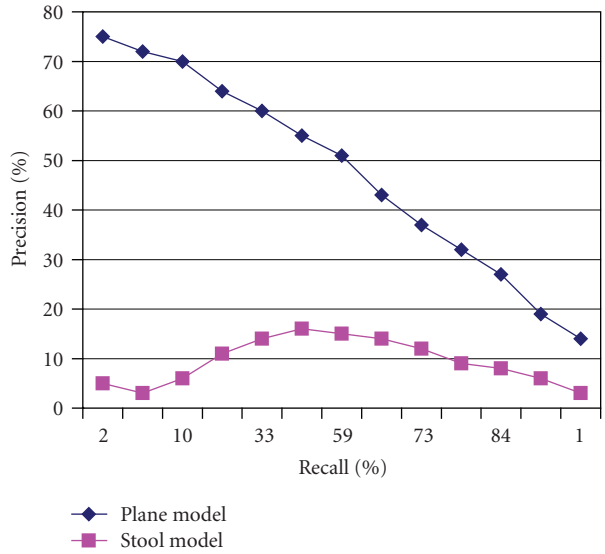


Figure 11: Precision recall of the plane and the stool models.

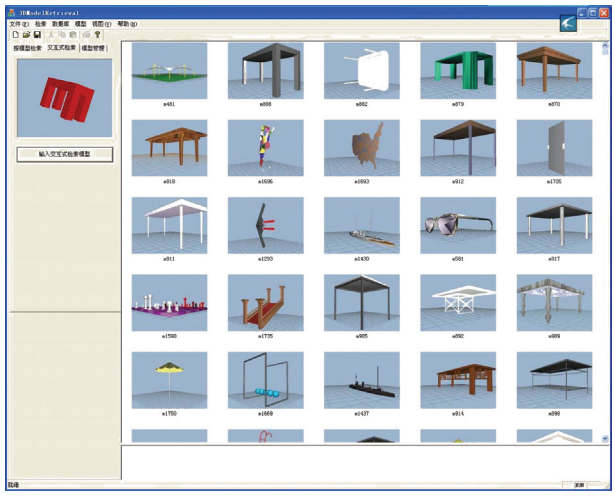


Figure 12: The new experimental result.

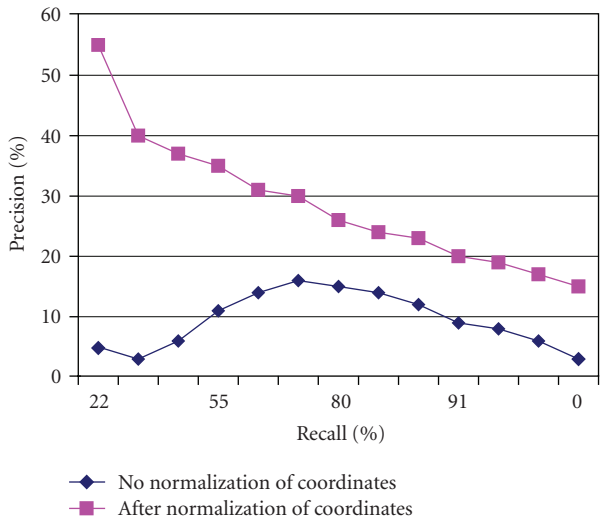


Figure 13: Precision recall of stool model.

rate (recall) and check rate (precision) to evaluate [17, 18]. The recall rate and the precision rate are as follows:

$$\text{Recall} = \frac{\text{relevant correctly retrieved}}{\text{All relevant}}, \qquad \text{Precise} = \frac{\text{relevant correctly retrieved}}{\text{All retrieved}}. \tag{3}$$

According to the retrieval results of plane model and stool model, we first calculate their recall rate and the precision rate and then draw precision-recall curve, as shown in Figure 11.

As can be seen from Figure 11, the plane model precision rate is far higher than that of the stool model. The precision rate of the stool is very low. How can we optimize the results?

4. Retrieval Optimization

In order to make retrieval results better, first of all, we use CPCA method [19, 20] to coordinate normalization of the pretreatment for the model. Retrieve again and the new result is shown in Figure 12.

Contrasting Figure 12 to Figure 10, we can see the retrieval result clearly improved.

To further contrast, we draw the precision-recall curve of the stool model before and after using the coordinate normalization, as shown in Figure 13.

As can be seen from Figure 13, the retrieval accuracy has greatly improved after coordinate normalization.

5. Conclusion

In the analysis and comparison of 3D model retrieval methods, we propose a new 3D model retrieval method with building blocks. Without hardware supporting, we construct query models by building blocks in the virtual environment.

Compared with the previous retrieval methods, it is very easy to construct query models. Users can easily express their intentions to query. The retrieval method we provided opens up a 3D model of the new ideas retrieval method, and it has a good retrieval performance at the same time.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 60736008). As the authors have acknowledged throughout the paper, the experiments conducted in their work rely heavily on the Princeton University 3D retrieval database PSB. The authors would also like to thank the anonymous reviewers.

References

[1] J. Wang, Y. He, H. Tian, and H. Cai, "Retrieving 3D CAD model by freehand sketches for design reuse," *Advanced Engineering Informatics*, vol. 22, no. 3, pp. 385–392, 2008.

[2] P. Min, J. A. Halderman, M. Kazhdan, and T. Funkhouser, "Early experiences with a 3D model search engine," in *Proceedings of the 8th International Conference on 3D Web Technology*, pp. 7–18, Saint Malo, France, March 2003.

[3] Y.-B. Yang, H. Lin, and Q. Zhu, "Content-based 3D model retrieval: a survey," *Chinese Journal of Computers*, vol. 27, no. 10, pp. 1297–1310, 2004.

[4] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.

[5] T. Funkhouser, P. Min, M. Kazhdan, et al., "A search engine for 3D models," *ACM Transactions on Graphics*, vol. 22, no. 1, pp. 83–105, 2003.

[6] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton based shape matching and retrieval," in *Proceedings of the Shape Modeling International (SMI '03)*, pp. 130–139, Seoul, Korea, May 2003.

[7] C. Zhang and T. Chen, "Indexing and retrieval of 3D models aided by active learning," in *Proceedings of the 9th ACM International Conference on Multimedia*, vol. 9, pp. 615–616, Ottawa, Canada, September 2001.

[8] W. Liu, Y. Uehara, Y. Liu, et al., "3DMIRACLES: 3D model retrieval and visualization engine," in *Visualization and Data Analysis 2005*, vol. 5669 of *Proceedings of SPIE*, pp. 250–261, San Jose, Calif, USA, January 2005.

[9] M. T. Suzuki, "A web-based retrieval system for 3D polygonal models," in *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference (NAFIPS '01)*, vol. 4, pp. 2271–2276, Vancouver, Canada, July 2001.

[10] J. Pu and K. Ramani, "On visual similarity based 2D drawing retrieval," *Computer Aided Design*, vol. 38, no. 3, pp. 249–259, 2006.

[11] L. Cao, J. Liu, and X. Tang, "3D object retrieval using 2D line drawing and graph based relevance reedback," in *Proceedings of the 14th Annual ACM International Conference on Multimedia (MM '06)*, pp. 105–108, Santa Barbara, Calif, USA, October 2006.

[12] M. J. Fonseca, A. Ferreira, and J. A. Jorge, "Towards 3D modeling using sketches and retrieval," in *Proceedings of the EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, pp. 127–136, Grenoble, France, August 2004.

[13] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3D freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 409–416, Los Angeles, Calif, USA, August 1999.

[14] S. Hou and K. Ramani, "Classifier combination for sketch-based 3D part retrieval," *Computers and Graphics*, vol. 31, no. 4, pp. 598–609, 2007.

[15] H. Siegl, M. Hanheide, S. Wrede, and A. Pinz, "An augmented reality human-computer interface for object localization in a cognitive vision system," *Image and Vision Computing*, vol. 25, no. 12, pp. 1895–1903, 2007.

[16] H. Ichida, Y. Itoh, Y. Kitamura, and F. Kishino, "Interactive retrieval of 3D shape models using physical objects," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pp. 692–699, New York, NY, USA, October 2004.

[17] K. Michael, F. Thomas, and R. Szymon, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 156–164, Aachen, Germany, June 2003.

[18] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3D model retrieval," *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, 2003.

[19] D. V. Vranić, D. Saupe, and J. Richter, "Tools for 3D-object retrieval: Karhunen-Loeve transform and spherical harmonics," in *Proceedings of the 4th IEEE Workshop on Multimedia Signal Processing (MMSP '01)*, pp. 293–298, Cannes, France, October 2001.

[20] D. V. Vranic, *3D model retrieval*, Ph.D. dissertation, Department of Computer Science, University of Leipzig, Leipzig, Germany, 2004.

Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 323095, 9 pages
doi:10.1155/2009/323095

Research Article

Player Profile Management on NFC Smart Card for Multiplayer Ubiquitous Games

Romain Pellerin,[1,2] Chen Yan,[1] Julien Cordry,[1] and Eric Gressier-Soudan[1]

[1] *CNAM-CEDRIC, 292 rue St Martin, 75141 Paris Cedex 03, France*
[2] *GET-INT, 9 rue Charles Fourier, 91011 Evry Cedex, France*

Correspondence should be addressed to Julien Cordry, julien.cordry@cnam.fr

Received 30 January 2009; Accepted 14 July 2009

Recommended by Zhongke Wu

One of the goals of mixed reality and ubiquitous computing technologies is to provide an adaptable and personal content at any moment, anywhere, and in any context. In Multiplayer Ubiquitous Games (MUGs), players have to interact in the real world at both physical and virtual levels. Player profiles in MUGs offer an opportunity to provide personalized services to gamers. This paper presents a way to manage MUG player profiles on an NFC Smart Card, and proposes a Java API to integrate Smart Cards in the development of MUGs. This user centric approach brings new forms of gameplay, allowing the player to interact with the game or with other players any time and anywhere. Smart Cards should also help improve the security, ubiquity, and the user mobility in traditional MUGs.

1. Introduction

We deeply believe that the next step for the gaming industry will be Multiplayer Ubiquitous Games (MUGs). In this type of game, users play simultaneously in the real world and in the virtual world [1]. To manage an MUG system which supports social interactions among interconnected users in both worlds, the system has to manage the equipments that are deployed in the real world, and to compute the state of the virtual world.

Our purpose here is to enhance the mobility and the ubiquity in MUGs by using a user-centric approach. This might give rise to new kinds of user interactions.

Various technologies, such as RFID tags, networked objects or environmental sensors, can be used to help the user interact with his/her physical environment. Moreover, the players can have access to hand-held device, biomedical sensors, interaction devices, virtual reality glasses, and so forth. Then, various network connectivities are used to link all these devices: Wi-Fi, Bluetooth, ZigBee, or cellular phone networks. Finally, an MUG server could run the global game logic, centralize the game data, and bring the players together. A proper way to support this technological heterogeneity is to use a middleware, like uGASP [2, 3], which is an OSGi-based [4] open-source middleware dedicated to MUGs.

On the gameplay level, MUG systems introduce the concept of Real world Gaming system Interaction (RGI). It is based on the following properties. Firstly, the gameplay relies on the player's physical mobility and often requires a context and a user adaptation. Secondly, the game interacts with the player in an ubiquitous way (at nondedicated locations through nondedicated objects), and proactively (at uncontrolled times, e.g., through email or phone). Finally, the game leads to social interactions which can be effective in the real world or in the virtual world.

So, MUG systems have to be flexible and adaptable enough to be able to respond to these complex and uncertain relations between the real world and the game world, and between the player and the real world. Furthermore, the player should be able to interact with the game despite a network disconnection, for example, to interact with a smart toy in a nonnetworked area. On the design level, like all games, and, more generally, like all entertainment applications, an MUG system should include a user model. An MUG system can be seen as an information system

requiring some user personal data in order to integrate the user's real life into the game, for example, his/her phone number or his/her real life social relations. Natkin and Yan [5] propose a player profile model to provide a personalized gaming experience to the player.

One of the ways to store a player profile in an MUG system is to let the player carry the profile along with him/her on an embedded computing device, such as NFC Smart Card. The Near Field Communication (NFC, [6]) Smart Cards are a fast growing member of the large Smart Card family. Today, Smart Cards are widespread devices with cryptographic and storage capabilities, and tamper-resistant properties. This makes those devices ideal for many application contexts like in identification, transport, telecommunication, or banking domains. Their non-self-powered essence implies the use of a reader/Card Acceptance Device (CAD) that can power up the card and interact with it. The NFC technology enables them to interact with their environment in a contactless manner, most primarily with mobile phones.

No public attempt to manage an MUG player profile on a Smart Card has been provided so far. Besides, it appears that many game systems tend to understate the security and confidentiality issues that should be addressed in any networking environment while some personal data are involved. The work undertaken here is part of the PLUG research project [7]. PLUG is led by the CNAM-CEDRIC computer science research laboratory in collaboration with Musée des Arts et Métiers, Orange Labs, Institut Sud Telecom, L3i lab from University La Rochelle, and a game studio: TetraEdge. It aims at creating an MUG inside the CNAM museum that takes into account the player characteristics. This MUG is built on top of the uGASP middleware.

This paper introduces a user centric approach dedicated to MUG systems. Our approach consists in using an NFC Smart Card to store the MUG player profile, providing mobility, and guaranteeing user privacy and confidentiality. The player holds some game information in order to interact with the surrounding NFC devices. In addition, the Smart Card provides the player with a secure way to store confidential data. In this work, we present an open-source service to manage MUG player profile (MUGPP) for Java-based devices (card, reader, and server levels): the MUGPPM API (the API for our MUG Player Profile Management). Section 2 describes the MUGPP. Section 3 presents the technologies used for user profile management on Smart Cards. Section 4 discusses the benefits of handling the player profile on an NFC Smart Card for MUGs and the kinds of new interactions it could bring to the user and the MUG system. In Section 5, the general architecture of the system is presented and the security issues related to the protection of the data in the card are discussed. Section 6 describes the new kinds of interactions using our API. The last section concludes and gives our perspectives for future work.

2. Player Profile Definition for MUG (MUGPP)

The essence of gameplay is designing a game with regards to the user point of view. This point of view is implicitly or explicitly coded in the game system: all games and all entertainment applications include a user model. In single player games, it starts from a rough classification of the target players and a limited memory of player actions in the game, but it can also be a complex cognitive model. In multiplayer games, the model contains social attributes and behaviors. In multiplayer ubiquitous games, the model has to be cognitive, social, and related to the history and to the current situation of the player in both the virtual and the real world.

Considering that the user's space of activity embeds computing devices and that information systems become more and more ubiquitous and pervasive, there is a need to consider the interaction between the real and the virtual world in a mixed reality mode, and the possible actions of the user in both universes. So the user model will not only take into account the state and behavior of the user as in classical online gaming situations but also in augmented outdoor or mobile gaming environments.

Our method is to use an explicit user model, the MUG Player Profile (MUGPP), to gather and classify distinctive information about the player. This information will be the deductive basis for the game decision mechanism.

The MUGPP guides the game decision engine to offer diverse game experiences to players. The game quests adapt the game scenario to the personal context of the player, which leads to an action that is executed both in the game and in the real world. The main goal in the use of the MUGPP along with the automatic generation of the narration is to decide which type of quest can best relate to the player profile and to the global narration needs, so as to promote social relations between players. In this way, the playability of the game is augmented: the game is persistent and adaptable. Each player can have a unique experience.

The MUGPP depends on a set of parameters that can be either statically defined by the game designer or dynamically adjusted according to the real time changes in the user's physical states or even in the user's social features. It implies a personalized level of parameters in the user model [5]. Since the player is represented in both the real world and the virtual system, we have to consider his/her knowledge of the gameplay from several different points of view. It is very useful to distinguish the user's general information from his/her in-game data, as his/her general profile could be re-exploited by different game mechanisms. The following three groups describe the kinds of user information that are collected and identified.

The first group includes some data about the user "by himself, " that is, unrelated to his/her game practice: civil status, preferences, and so forth. Most of this data can only be provided directly by the player during the creation of the game account. Since this data changes infrequently, it has to be accessible by any MUG on the game platform, so the player does not need to register his/her civil status every time he/she plans to play a new game.

The second group collects the knowledge about the user defined "as a player." It includes some exact information corresponding to the basic choices of the player: the type of account, distribution of the duration of play in each location, and so forth. It includes also statistical data or some real-time

data gathered during the play: his/her physical location, his/her interaction with the various interactive devices in the real environment, and so forth.

The third group defines the status of the player's avatar in the game from both a statistical and a real-time point of view, such as the standard information of his/her avatar, his/her equipment and inventory, or his/her social relations in the game. This data could be used by the game server to propose some special customized game events to the players, such as a specific common quest requiring a particular object from two players' inventories.

This user model has been experimented in the prototype MugNSRC [8]. The original game, NSRC, is based on cartoon type wheelchair races in the office of a virtual Japanese Company. MugNSRC uses this context and integrates a user model with the player's motivation profile in the game engine as a mean to manage and develop a community through cooperative and competitive goals assigned to the players.

The question of which device hosts the user profile in the system relies on the global architecture of the game. Generally, multiplayer games follow a client/server architecture. The user profile is managed by the server, as in MugNSRC. Initial values of each class of data in the MUGPP are computed following the principle of a questionnaire. The player is invited to fill a form used to set the initial values of MUGPP parameters before the creation of his/her game account. These values could be changed according to a feedback loop related to the player choices and actions in the game. The user can log in to access his/her account, and retrieve his/her profile. On the other hand, P2P multiplayer games manage the player profile on the client side. The disadvantages of such an architecture are that the user has to manage himself/herself his/her profile when he/she changes to a new terminal, and that the players can cheat easily.

3. Smart Cards in the Management of User Profiles

Our work focuses on finding a way to manage efficiently player profiles in MUGs in order to provide a more personalized game to the gamers. Since network coverage and network connections are potentially unreliable, an interesting approach to carry out the game in a continuous manner would be to let the player carry his/her player profile along with him, so that the user is still able to play despite disconnection. This assessment leads to build a distributed and persistent information system for game data, and especially what we called MUGPP information. To manage this information, wearable devices are appropriate. The list of such devices includes mobile phones, PDA, Smart Cards, game consoles, memory cards, and so forth. Among those, Smart Cards are a good compromise in terms of wearability, security mechanisms, and costs.

Smart Cards are the most secure and widespread portable computing device today. They have been used successfully around the world in various applications involving money, proprietary data, and personal data (such as banking, pay-TV or GSM subscriber identification, loyalty, health-care, insurance, etc.). The Java Card [9] and the Microsoft.Net framework for Smart Cards are platforms that support a multiapplication environment, and in their modern versions, tend to go multithread. One of the key elements of Smart Cards is to improve on basic magnetic stripe cards with dynamically programmable microcontrollers, cryptographic Coprocessors, and means to protect the embedded data. Furthermore, Java Card platforms usually embed some code verifier, making those devices safer. Aside from their small size (to fit on a flexible plastic card and to increase hardware security) and from their low cost (to be sold in large volumes), this makes them ideal for any ubiquitous security-sensitive environment. Today Smart Cards are small computers, providing 8, 16, or 32 bits CPU with clock speeds ranging from 5 up to 40 MHz, ROM memory between 32 and 128 KB, EEPROM memory (writable, persistent) between 16 and 64 KB and RAM memory (writable, nonpersistent) between 3 and 5 KB. Smart Cards communicate with the rest of the world through Application Protocol Data Units (APDUs, ISO 7816-4 standard). The communication is done in client-server mode, the Smart Card playing the role of the server. It is always the terminal application that initiates the communication by sending a command APDU to the card and then the card replies by sending back a response APDU (possibly with an empty content).

Smart Cards can be accessed through a reader. The access has traditionally meant inserting the Smart Card in the reader. However, the trend is to interact in a contactless manner, to improve the Human Computer Interface (HCI) aspects. The Near Field Communication (NFC) technology provides devices with the ability to interact within a short range (less than 10 centimeters) by radio signal. This technology stems from the recent RFID market development. It works at a 13.56 MHz frequency, provides a 424 kbit/s bandwidth, and supports a half-duplex communication between devices. NFC Smart Cards combine the two previous technologies, so they are easily accessible in a contactless manner. Since these cards are non-self-powered, the radio signal from a reader is used to power the Smart Card-integrated circuit, in the same manner as RFID tags.

In the context of ubiquitous systems, the user can either carry an NFC Smart Card, which is readable within a short range by an NFC reader, or carry a reader, which is able to interact with the NFC devices disseminated over an area. As far as we know, there is no MUG that makes use of a Smart Card. However, there are some similarities between using a Smart Card for an MUG and using a Smart Card that is dedicated to commercial applications like public transportation systems and banking applications. Today, numerous cities in the world use contactless Smart Card-based systems to manage their public transportation system. For instance, the Paris commuters can use their contactless Smart Card (*Navigo*) as a mean to access transportation facilities (trains, buses, etc.) as well as the public bicycles network (*Velib*). The later involves a network of bicycle stations, which are equipped with an NFC readers, and a central authority, to help regulate the traffic. The Smart Card is used to store some user-related data, for example, the log of the stations he/she went through.

The core of this type of distributed information system is the management of user data on Smart Cards. There has been some effort to manage a health profile with PicoDBMS [10]. PicoDBMS is a Database management system dedicated to Smart Cards. PicoDBMS has also been used in some work undertaken by Lahlou and Urien [11] to filter some Internet data through a Smart Card-based user profile. They manage the profile dynamically (the user can specify his/her preferences). The security approach is that of the P3P (Platform for Privacy Preferences) [12] normalization group. The framework offers two security levels, the less secure being the less Smart Card intensive. The approach leaves out any gaming/ubiquitous aspect, and there is no mention of any authentication/confidentiality of the information. Ubiquitous systems should introduce a middleware to support this distributed information system. There are three essential components for these systems which are the users and their Smart Cards, the readers, and a central authority server.

4. Playing MUGs with NFC Smart Cards

The game system of some existing MUGs, such as [13–15], relies on the capability to control all the physical objects, which are integrated in the game, their impacts on the player, and all the various real-world embedded sensors, which take part in a hierarchy of networks. The participants of MUGs often experience the heavy load of physical wearable devices, or they have to deal with network disconnection problems [16]. Our proposal consists in using a Smart Card as an add-on interface for the interactions between the player, the virtual world, and the real world.

On the player's side, the MUGPP can be specified on a Smart Card, which enables the player to have access to some of his/her game-related information. The player can monitor his/her game process, manage his/her game objects, and even visualize or being informed with the game progress by either using one of the fixed terminals that are spread over the game area or by using a mobile terminal. In the context of an NFC Smart Card-based player profile, this would mean that the player could interact by using a Smart Card with a fixed NFC reader or with his/her mobile phone integrated reader.

The update of the MUGPP is executed automatically by the system and manually by the player. Firstly the MUGPP could be renewed by the player's physical interaction, that is to say, the player's physical movement and behavior in the real environment (outdoor and indoor). As the real environment is embedded with tangible objects, the player's physical location could be "tracked" as he/she walks through the game zones. The interaction between a Smart Card and a smart object using NFC readers can be performed without any connection to the game server. Every time the player comes close to a Smart Card reader, some of the MUGPP information can be updated and used in any way by dealing with the "as a player" data. Secondly, the MUGPP is updated following the communication or social interaction among players in the real world. The players should be able to sell and buy the game items they own to other players even while they are offline. The third group of information, that is to say, the "as his/her avatar" data can be updated dynamically. The social dimension of the gameplay is extended to the spatial and temporal dimension of the game. Therefore, the game system could trigger and control some game events in real time and real space for a group of players in the same game zone. Thus, the MUGPP can be updated during the real time interactions between the players, the game, and the physical space.

Playing MUGs with a Smart Card is a relatively new experience for the user, which will bring new forms of interaction to the players, new contents, and new security features.

Using a Smart Card gives the players new ways to interact with the game, potentially without any display device. This means that an automatic tangible interaction between the NFC Smart Card and the NFC reader can take place by bringing them close to one another. For the user, the most accessible and affordable mobile terminal is the mobile phone. Also, some are able to integrate the NFC technology, like the *Nokia 6131 NFC* and the *Sagem My700x*. Therefore, we suggest to use an NFC mobile phone to run a client application in our MUG system.

An ideal MUG is a digital environment with smart objects surrounding the user. This would allow him/her to interact with the game anywhere. Therefore, we can embed NFC readers in smart objects, such as Nabaztag [17], which could interact with each user's Smart Card. To enrich the user experience, a television decoder may also integrate an NFC reader so that the player could gain access to the multimedia content related to the game.

On the Smart Card, we aim at defining and formalizing an MUGPP which might help maintain decentralized user data from the game server. This MUGPP allows the user's personal information to be reused by several game mechanisms and to be completed by several applications. The interest of having an MUGPP on a Smart Card is not only that users have a more "wearable" computing device but also that the game designers can provide each individual with a personalized gaming experience. In the mechanism of a MUG, the MUGPP can take a central role rather than being a peripheral or real context to influence the game server in making the decision for a customized service to the end user.

Considering security aspects, the specification of player profiles as separated from the server will guarantee the confidentiality of each individual's private information and the related service. For example, it could be possible to register the information of the player's bank account on the Smart Card which allows the player to have access to a paying service. In "World of Warcraft" (Blizzard Entertainment, 2004), the user can register his/her bank account on the game server, which can be unsafe despite the login/password protection, in order to obtain some special services from the game editor. From a perspective point of view, this will enlarge the possibility of license management such as biological or vocal based identity.

As a consequence, there is a need to support Smart Card, NFC reader in the MUG system architecture. We will describe an API which provides this service in more details in the following sections.

5. Architecture to Manage MUGPP on an NFC Smart Card

The NFC interactions in MUGs (see Section 4) and of the MUG player profile (see Section 2) are key issues of our proposal. The main component of this architecture is the service that manages the MUG player profile on the external NFC Smart Card. We have implemented a library which enables *Java 2 Micro Edition* [18] (J2ME) *Mobile Information Device Profile-* (MIDP-) based mobile phones to exchange data with Smart Cards and game server logic. The server is itself implemented in J2SE and the Smart Card part of the application is a Java Card cardlet. Finally, we use the security mechanisms to ensure the privacy of the player profile data. Figure 1 presents an overview of the MUGPPM architecture.

5.1. Oncard Service. Our card-side implementation aims the card applications based on Java Card platform which complies with the ISO 14443 [19] standard part 1, 2, and 3 type A. An oncard Java applet is dedicated to the MUGPPM. It implements a set of instructions to handle communication with an NFC reader. These instructions are built using the Application Protocol Data Unit (APDU) protocol defined in the ISO/IEC 7816 standard. Besides, it maintains the player profile data model with some added security features.

5.1.1. APDU Instructions. The APDU instruction set used in MUGPPM allows the following:

(i) to manage the player profile,

(ii) to manage the default entries, for example, static entries of our MUG player profile definition, for example, the *username*, *age,* or *playtime* fields,

(iii) to manage object entries, for example, entries corresponding to game data objects, like inventory items,

(iv) to manage the private and public key entries.

With this set of instructions, the reader can access a profile stored on the Smart Card, save/load each profile fields independently, and store/retrieve the game objects. Objects can be defined as exchangeable between players. Nevertheless, it is the MUG game designer who has to decide if a game object is sharable or not. Table 1 shows the instructions used by the MUGPPM. It details the parameters of each instruction and the corresponding response of the Smart Card.

5.1.2. Data Model. The field lengths have been bounded due to the memory limitation that characterizes the Smart Card platforms. We tested our implementation on an *Oberthur Cosmo Dual* card which offers only 72 KB of memory space (EEPROM).

Nevertheless the profile itself is not really heavy, since the CAP file containing our oncard application uses around 6 KB. We also use a 4 KB memory buffer to deal with large I/Os. The fields of the GameProfile class themselves include a number of byte arrays (264 bytes), and a couple of OwnerPIN objects to manage the user password and the game provider password (the object size depends on the Java Card Virtual Machine (JCVM) implementation, but the password itself is limited to 8 bytes). Furthermore, the profile is associated with three 2048 bits RSA keys (768 bytes). So the application itself requires 8 KB and each instance of a profile should require less than 2 KB (depending on the JCVM implementation). Therefore, we could theoreticly manage about 30 different game profiles with this Smart Card.

5.2. NFC Reader Side Service. The main functionalities of the NFC reader API are to access the MUG player profile stored on the Smart Card and to communicate with the profile-based services that are hosted on the MUG server.

The APDUDataManager class is used to establish the NFC communication toward the card and to some send APDU formatted messages. The GameProfile class is used to manage the player profile fields during profile manipulation on the reader. Finally, the NetworkCom class handles object-oriented HTTP communications with the server based on the MooDS protocol [20].

We have prototyped a J2ME version of our MUGPPM service to have a Java mobile phone access to the oncard MUGPPM service. This choice is obvious considering the mobile phone is the most widespread mobile terminal for end-users. Moreover, in 2007, some J2ME mobile phones embedding NFC readers, such as the *Nokia 6131 NFC* or the *Sagem my700X*, were placed on the market. An API to help establish a contactless communication between a J2ME mobile phone and an NFC Smart Card has been released the same year: the *JSR257* [19].

A specific API is traditionally used to handle an APDU-based communication on J2ME mobile phones: the *JSR177* [21]. However, the use of this API is not mandatory in the case of an external NFC Smart Card. In fact, it offers essential mechanisms enabling the mobile phone to communicate with its embedded SIM card. Thus, our prototype uses the JSR257 functionalities to initiate a communication between the mobile phone and the Smart Card.

In order to use the MUGPPM functionalities, the first step for the player is creating his/her MUGPP on the Smart Card. So, he/she has to enter a login and a password which will be used to access his/her profile. In this first step, he/she has to enter his/her personal information, for example, the user "by himself" is part of the MUGPP. Thus, the API can load the player profile fields from the card onto the mobile phone, to store them in the profile object representation. Afterward, the MUG client game engine can start using the player profile as it is defined in the MUG game design. Figure 2 summarizes the architecture used to provide the MUG client game engine with an access to the oncard MUG Player Profile Management service.

It is important to notice that the interaction between the mobile phone and the NFC Smart Card depends on the player since, he/she has to draw the card near the mobile phone during all the process, for example, during a game save. The MUG game designer must take into account this specific Human Computer Interaction (HCI).

Besides, our prototype can communicate with an MUG HTTP server. It uses the MooDS protocol to communicate

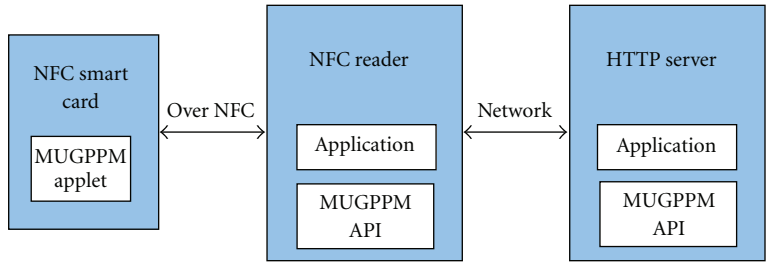


Figure 1: MUGPPM architecture overview.

Table 1: APDU instructions used in MUGPPM.

Instruction	P1	P2	Data	Returns
CREATE_PROFILE			login+pwd	status
LOGIN_PROFILE			login+pwd	status
REINIT_PROFILE			login+pwd	status
DELETE_PROFILE			login+pwd	status
LOAD_DEFAULT_ENTRY	key			data
UPDATE_DEFAULT_ENTRY	key		data	status
LOAD_OBJECT_ENTRIES				data
LOAD_OBJECT_ENTRY	key			data
ADD_OBJECT_ENTRY	key	isSharable	data	status
DELETE_OBJECT_ENTRY				status

with the MUG server in an object-oriented manner. The developer can creates objects which represent the messages used during the client-server communication. Thus, if the MUG client needs a profile-based service from the server, it has to instantiate the corresponding message object and send it through the MooDS encoder. We have created a message to invoke a server side service, the ProfileBasedServiceRequest message class. It can also decode the server response using the MooDS decoder and handle the decoded message objects. If the service requires data from the Smart Card, the client receives a CardDataRequest message from the server which contains a list of required field keys. Then, the MUGPPM API can retrieve the associated fields data from the Smart Card and send it within a DataCardResponse object to the server. Finally, it receives the service response, for example, a player list from a lobby service.

5.3. Server Side Service. The MUGPPM server API offers a Java based MUG server the ability to create a profile based service. It helps create personalized services, for example, profile based lobby or profile based quest provider. The server API and the client API have a similar class to handle MUGPP contents: theGameProfile class. For example, if the server requires the player nationality, it has to request the corresponding field key from the Smart Card and to handle the card response. The communication part of the API is also based on the MooDS protocol.

To request a service, the client has to send a ProfileBasedServiceRequest message with the name of the service needed. Then, if the service requires personal data stored on the Smart Card, it sends back a CardDataRequest message to the client containing a list of required field keys. Afterwards, it receives from the client a DataCardResponse message which contains the required data. Finally, the service computes the response based on the received player personal data and returns a specific response message to the client.

5.4. MUGPP and Security. Some of the MUGPP data deal with the user private life. Furthermore, the lack of a sound and secure authentication procedure typically makes cheating in MUGs an easy feat [22, 23]. There is a need to use improved security mechanisms to act against those threats.

The players and the terminal the players use (in our case a mobile phone) are by definition untrusted, but the oncard application can be securely and reliably developed using Java Card.

In order to insure the security of the player's private data, the card requires an authentication from the reader. This authentication process is based on a personal login/password chosen by the player during his/her account creation. We use the OwnerPIN class on the card to safely store the user password. The login procedure needs to be performed to authorize the access to the smart card cryptographic functionalities. When the user is not playing any more, the user is logged out from the Smart Card. The application provider uses another PIN code to block/unblock the user from modifying certain fields.

We chose to use a public key infrastructure to help the MUG system designers ensure the security of the application. Yet, the management of the keys on a Smart Card is a non trivial issue. The Smart Card requires a personalization phase during which a key pair is created and stored on the

Figure 2: MUGPPM architecture for J2ME devices.

card static memory. The server side also requires a key pair, and an X.509 infrastructure is used to certificate the use of public keys. This public key infrastructure guarantees the privacy of communications between the server and the Smart Card. Thus, when user logs in, he/she does so, not with a Smart Card, and not with a server. He/she can then have access to a higher security level than just a password-based protocol.

When the application needs to interact with the server, the server sends its public key as well as a certificate. The Smart Card can then verify the validity of the key. If the key happens to be valid, the Smart Card can keep the public key. The Smart Card can send its public key to the server. All subsequent interactions between the server and the client can then use an encryption/decryption using one's private key and the other's public key.

The overall mechanism guarantees a stronger identification scheme than just a login/password and might help thwart some common online games cheats. One advantage here is that no critical data is transmitted in plain text format over the network.

A common cheat is the replacement of code or data concerning the game. The simple fact of using a Smart Card to manage the MUGPP makes it considerably difficult to tamper with the game profile, the cheater being unable to directly hack into the profile/oncard game infrastructure. The game designer might want to check an additional server signature for any operation that modifies some elements in the profile.

An other cheat consists in abusing the game procedures. For instance, a player can log out before he/she loses a game. Making the signing of some game procedures by the server necessary can be used as a countermeasure against such cheats.

The mobile aspect of our framework implies that some interactions between two players can occur out of a connection with the game server. For instance, in a role-playing game, the players might want to exchange an item. This operation could take place without a server while still guarantying the nonrepudiation property.

6. MUG PPM Use Cases

Our library can be used for different types of interactions, *connected* or *disconnected* interactions from an MUG server point of view. For example, the secure architecture of the MUGPPM can only be used safely with a network connection in order to validate public keys with a signing authority through a registered MUG server. So, secure interactions have to be carried out in a connected way. However, disconnected interactions are possible without strong security mechanisms, particularly for local interactions. Thus, an MUG can introduce NFC checkpoints or local object exchange mechanisms between players using this API.

6.1. Connected MUG Interaction Examples. Our framework can be used to provide various profile-based connected services in a secure way, like providing players with personalized quests or locating players who speak a common language in a game area.

Via mineralia [24] is a pervasive search and quizz game in the museum of Terra Mineralia in Freiberg. The goal of the game is to realize quests in the context of the mineral exposition. Each point of interest is represented by an RFID tag on the mineral. The MUGPPM can be used in this application to check the visitor card at the museum entry (with an NFC reader) to adapt game content to his/her player profile. For example, different levels of mineral knowledge could be set to fit the category of the visitor (novice, expert, etc.) and to propose personalized quests. Moreover, regular visitors could resume a quest undertaken previously.

As an another use case, we have implemented a profile-based lobby service on top of the MUGPPM secure architecture. This service uses the player's age and the languages he/she knows. The server asks for the user's required personal data while using the security part of MUGPPM. Finally, the profile based lobby service computes the list of connected players matching the required age and spoken languages and returns it to the client. That type of service could have been used in games like the item hunt game "Mogi Mogi" [15]. In this game, some users have been using a lobby-like

application to spy on other younger players. Bypassing the game rules this way can be controlled using our API. Indeed, as the private data is stored on a secure decentralized device (unlike a game server), fraudulent use of personal data is rendered more difficult, while statistics can still help detect that type of behavior.

6.2. Disconnected MUG Interaction Examples. MUG game designers can integrate disconnected interactions in their game by using the MUGPPM API.

Paranoia Syndrome [25] is a classic strategic game that integrates some location based interactions, and RFID tangible objects. One of the perspectives of the game, is that multimedia content and basic AI will be added to the tangible objects to serve different content by regarding the player type (doctor, scientist, alien, etc.). With MUGPPM, the interactive objects (with an embedded NFC reader) could adapt their content and interaction to the player with regard to the player profile in a disconnected way.

Furthermore, a MUG can integrate difficulty levels corresponding to the player's age in order to assign a course to the player in the game area. This interaction can be made between the player and a NFC checkpoint and does not necessarily require a server side resolution.

In addition, MUGs can implement game object exchange mechanisms between players. Such a service should give two players in the same *real world* area, the ability to exchange some game items from their inventories. This interaction can be made by peering the mobile phones of the players over a local communication link. The NFCIPConnection class from the com.nokia.nfc.p2p package (available in the Nokia JSR257 implementation) allows to establish a NFC link between two phones. We have implemented a game object exchange service, on top of our API, that offers to a player to send one sharable item from his/her game inventory to another player. We consider here that each player has previously loaded his/her player profile from the Smart Card. This list can be retrieved from the object representation in the player profile (see Section 5.2 for more details about the profile loading mechanism). So, a player who wants to send an object to his/her friend has to select the item from his/her list and the sender mode, whereas the other player has to select the receiver mode. The players must approach their mobile phones in order to set up the P2P link. As soon as the connection is established, the object is sent as a byte array onto the network. Then, the receiver handles the binary data corresponding to the item and can add it to his/her inventory. Finally, the new inventories of both players will be updated in the Smart Card during their next game save.

These examples emphasize a major benefit provided by our API in the MUG domain: it does not require the players to be connected with the central MUG server in order to interact in the game. Thus, our library enables new interactions for MUG in a totally decentralized manner.

To evaluate the performance of our application, we used the Mesure project [26], which is dedicated to measuring the performance of smart cards. The Mesure project provides detailed time performance of individual bytecodes and API calls. Given the use cases described earlier, we monitored the use of each bytecode and each API call for a regular use of our application. We then matched the list of used bytecodes and API calls with the individual performance of each feature measured on our smart card. The results show that the time necessary to perform a RSA encryption with the smart card is close to half a second, and it is by far the costliest of the operations described earlier. Login into the smart card, as a title of comparison lasts less than 20 milliseconds.

7. Conclusions and Perspectives

This paper presents an NFC Smart Card based approach to handle the player profile in the context of MUGs. This NFC card centric architecture allows new kinds of interactions in both centralized and decentralized ways. The main advantage of our method is to allow the players to play at any time, and anywhere, hence the ubiquitous aspect of the game. We have presented the MUGPPM API which is dedicated to the Java Card/J2ME/J2SE platforms. This enables MUG developers to implement a Smart Card based architecture to provide profile-based services. Thus, players can have a personalized game experience. Besides, this API provides the player with a secure way to ensure a certain level of data confidentiality. We will release the MUGPPM server API as an open source OSGi bundle to be integrated in the uGASP [2, 3] middleware. Thereafter, game developers could implement MUGs based on this framework, therefore offering personalized services.

On the basis of our framework, it is possible to specialize and realize an authoring tool for the development of MUGs. It would be interesting to consider using the NFC Smart Card from a more conceptual point of view during the design of the game. Using Smart Cards in MUGs may also give rise to the future direction of game design by developing new forms of interaction and narration based on new technology of mobility and ubiquity.

The question of "who personalizes the Smart Card" remains open. In traditional banking, telecom or transport applications, this is carried out by the card emitting company. However, the ongrowing multiapplication aspect of Smart Card makes it more and more questionable. For the purpose of testing our API, we let the user fill out the form which might be questionable for a secure application. Still, the application provider has some control over the fields through its own PIN code.

Future works include a generalization of the security architecture in terms of key sizes and algorithms, depending on the functionalities of a given Smart Card.

In addition, we will generalize the API to facilitate the description of services and to manipulate the player profile data structure. On the server side, this should help the describe connected the player profile-based services. On the card side, we will investigate PicoDBMS database to handle the player profile data structure.

We await the results of an other project: T2TIT [27] (Things to Things in the Internet of things). This project proposes to interact with contactless object, going as far as to give them a network identity, while keeping some strong security properties. The eventual conclusion of T2TIT can be helpful to us, for instance, we can expect to use some

encrypted channels. We intend to use the T2TIT security mechanisms in our work. The newly published Java Card 3.0 specification [9] introduces multithreading mechanisms in Smart Cards. This suggests other interactions between different profiles, which were not considered in this paper.

In terms of oncard code verifiers, works like embedded data flow analysis (see [28]) might also provide us with some strong on card inter-application protection features. We could reliably share some data from one profile to an other, and deny the access to such data from other profiles.

We have not explored here the issues of biometric identification. It is clearly complementary to the traditional cryptographic schemes, and as the Smart Card industry is integrating more and more of those, so should we.

References

[1] S. Björk, M. Börjesson, P. Ljungstrand, et al., "Designing ubiquitous computing games—a report from a workshop exploring ubiquitous computing entertainment," *Personal and Ubiquitous Computing*, vol. 6, no. 5-6, pp. 443–458, 2002.

[2] R. Pellerin, E. Gressier-Soudan, and M. Simatic, "uGASP: an OSGi based middleware enabling multiplayer ubiquitous gaming," in *Proceedings of the International Conference on Pervasive Services (ICPS '08)*, Sorento, Italy, July 2008, Demonstration Workshop.

[3] GASP/uGASP project, http://gasp.ow2.org.

[4] OSGi alliance, http://www.osgi.org/Main/HomePage.

[5] S. Natkin and C. Yan, "User model in multiplayer mixed reality entertainment applications," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '06)*, Hollywood, Calif, USA, June 2006.

[6] NFC Forum, http://www.nfc-forum.org/home.

[7] The PLUG project, http://www.capdigital.com/plug/.

[8] C. Yan, *Adaptive multiplayer ubiquitous games: design principles and an implementation framework*, Ph.D. thesis, Cotutelle Research Program with Orange Labs and CNAM, Paris, France, 2007, Supervisor: Stephane Natkin.

[9] Java Card platform, http://java.sun.com/javacard.

[10] P. Pucheral, L. Bouganim, P. Valduriez, and C. Bobineau, "PicoDBMS: scaling down database techniques for the smartcard," *Very Large Data Bases Journal*, vol. 10, no. 2-3, pp. 120–132, 2001.

[11] A. Lahlou and P. Urien, "SIM-Filter: user profile based smart information filtering and personalization in smartcard," in *Proceedings of the Ubiquitous Mobile Information and Collaboration Systems (UMICS '03)*, Klagenfurt/Velden, Austria, June 2003.

[12] Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P.

[13] S. Jonsson, A. Waern, M. Montola, and J. Stenros, "Game mastering a pervasive larp. Experiences from momentum," in *Proceedings of the 4th International Symposium on Pervasive Gaming Applications (PerGames '07)*, Magerkurth, Carsten, et al., Eds., pp. 31–39, Salzburg, Austria, June 2007.

[14] O. Sotamaa, "All the world's a botfighter stage: notes on location-based multi-user gaming," in *Proceedings of the Computer Games and Digital Cultures Conference (CDGC '02)*, F. Mäyrä, Ed., Tampere, Finland, June 2002.

[15] Mogi Mogi,http://www.mogimogi.com.

[16] D. Cheok, et al., "Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 8, no. 2, pp. 71–81, 2004.

[17] Friedrich von Borries, Steffen P. Walz, and Matthias Böttger, "Mogi: Location-Based Services—A Community Game in Japan ," in *Space Time Play*, vol. 2007, pp. 224–225, Birkhäuser Basel, Switzerland, 2008, http://www.springerlink.com/content/j0277056ult42551.

[18] J2ME MIDP, http://java.sun.com/javame/index.jsp.

[19] JSR257, http://jcp.org/en/jsr/detail?id=257.

[20] R. Pellerin, "The MooDS protocol: a J2ME object-oriented communication protocol," in *Proceedings of the 4th Mobility Conference*, Singapore, September 2007.

[21] JSR177, http://jcp.org/en/jsr/detail?id=177.

[22] J. Yan and B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '05)*, New York, NY, USA, October 2005.

[23] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and peer-to-peer gaming," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 1–13, 2007.

[24] G. Heumer, F. Gommlich, B. Jung, and A. Müller, "Via Mineralia: a pervasive museum exploration game," in *Proceedings of the 4th International Symposium on Pervasive Gaming Applications (PerGames '07)*, pp. 157–158, June 2007.

[25] G. Heumer, D. Carlson, S. H. Kaligiri, et al., "Paranoia Syndrome: a pervasive multiplayer game using PDAs, RFID, and tangible objects," in *Proceedings of the 3rd International Symposium on Pervasive Gaming Applications (PerGames '06)*, pp. 157–158, June 2007.

[26] The Mesure project, http://mesure.gforge.inria.fr.

[27] P. Urien, et al., "The T2TIT research project. Introducing HIP RFIDs for the IoT," in *Proceedings of the 1st International Workshop on System Support for the Internet of Things (WoSSIoT '07)*, Lisbon, Portugal, March 2007.

[28] D. Ghindici, G. Grimaud, and I. Simplot-Ryl, "An information flow verifier for small embedded systems," in *Proceedings of the International Workshop on Information Security Theory and Practices (WISTP '07)*, vol. 4462 of *Lecture Notes in Computer Science*, pp. 189–201, May 2007.