

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

EDITOR – IN – CHIEF

Václav Skala

Journal of WSCG

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia, Univerzitni 8, Box 314
306 14 Plzen
Czech Republic
skala@kiv.zcu.cz

Managing Editor: Vaclav Skala

Author Service Department & Distribution:
Vaclav Skala UNION Agency
Na Mazinach 9
322 00 Plzen
Czech Republic

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

Journal of WSCG 2009

Editorial Advisory Board

Adzhiev, V. (United Kingdom)
Baranoski, G. (Canada)
Bekaert, P. (Belgium)
Benes, B. (United States)
Bilbao, J. (Spain)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bouatouch, K. (France)
Brodie, K. (United Kingdom)
Buehler, K. (Austria)
Chen, M. (United Kingdom)
Chover, M. (Spain)
Coquillart, S. (France)
Crosnier, A. (France)
Csebfalvi, B. (Hungary)
Cunningham, S. (United States)
Daniel, M. (France)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Feito, F. (Spain)
Ferguson, S. (United Kingdom)
Flaquer, J. (Spain)
Flusser, J. (Czech Republic)
Gallo, G. (Italy)
Groeller, E. (Austria)
Gudukbay, U. (Turkey)
Gutierrez, D. (Spain)
Havran, V. (Czech Republic)
Jansen, F. (Netherlands)
Kavan, L. (Ireland)
Klein, R. (Germany)
Klosowski, J. (United States)
Lee, T. (Taiwan)
Magnor, M. (Germany)
Mollá Vayá, R. (Spain)
Muller, H. (Germany)
Murtagh, F. (Ireland)
Myszkowski, K. (Germany)
Pasko, A. (United Kingdom)
Pedrini, H. (Brazil)
Peroche, B. (France)
Platis, N. (Greece)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Rojas-Sola, J. (Spain)
Rokita, P. (Poland)
Rosenhahn, B. (Germany)
Rossignac, J. (United States)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Schumann, H. (Germany)
Segura, R. (Spain)
Selim, B. (Turkey)
Skala, V. (Czech Republic)
Slavik, P. (Czech Republic)
Slusallek, P. (Germany)
Sramek, M. (Austria)
Stroud, I. (Switzerland)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Triantafyllidis, G. (Greece)
Vergeest, J. (Netherlands)
Wu, E. (China)
Wuethrich, C. (Germany)
Zara, J. (Czech Republic)
Zemcik, P. (Czech Republic)

WSCG 2009

Board of Reviewers

Abas,M. (Malaysia)
Adzhiev,V. (United Kingdom)
Akleman,E. (United States)
Andreadis,I. (Greece)
Aspragathos,N. (Greece)
Aveneau,L. (France)
Baranoski,G. (Canada)
Battiato,S. (Italy)
Bekaert,P. (Belgium)
Bellon,O. (Brazil)
Benes,B. (United States)
Bilbao,J. (Spain)
Biri,V. (France)
Bittner,J. (Czech Republic)
Borchani,M. (France)
Bouatouch,K. (France)
Bouville,C. (France)
Brodlie,K. (United Kingdom)
Bruckner,S. (Austria)
Bruni,V. (Italy)
Brunnet,G. (Germany)
Buehler,K. (Austria)
CarmenJuan-Lizandra,M. (Spain)
Chaudhuri,D. (India)
Chen,M. (United Kingdom)
Chover,M. (Spain)
Chum,O. (Czech Republic)
Coquillart,S. (France)
Crosnier,A. (France)
Csebfalvi,B. (Hungary)
Cunningham,S. (United States)
Daniel,M. (France)
de Geus,K. (Brazil)
Debelov,V. (Russia)
Dingliana,J. (Ireland)
Drbohlav,O. (Czech Republic)
Duce,D. (United Kingdom)
Durikovic,R. (Slovakia)
Egges,A. (Netherlands)
Eisemann,M. (Germany)
Erbacher,R. (United States)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Ferko,A. (Slovakia)
Fernandes,A. (Portugal)
Flaquer,J. (Spain)
Flusser,J. (Czech Republic)
Foufou,S. (France)
Gallo,G. (Italy)
Galo,M. (Brazil)
Ganovelli,F. (Italy)
Garcia-Alonso,A. (Spain)
Giannini,F. (Italy)
Gonzalez,P. (Spain)
Grammalidis,N. (Greece)
Groeller,E. (Austria)
Gudukbay,U. (Turkey)
Gumbau,J. (Spain)
Gupta,A. (United States)
Gutierrez,D. (Spain)
Hanak,I. (Czech Republic)
Haro,A. (United States)
Hasler,N. (Germany)
Havran,V. (Czech Republic)
Hernández,B. (Mexico)
Herout,A. (Czech Republic)
Horain,P. (France)
House,D. (United States)
Iwanowski,M. (Poland)
Janda,M. (Czech Republic)
Jansen,F. (Netherlands)
Joan-Arinyo,R. (Spain)
Jones,M. (United Kingdom)
Karabassi,E. (Greece)
Kavan,L. (Ireland)
Kimmel,B. (Canada)
Klein,R. (Germany)
Klosowski,J. (United States)

Knight,M. (United Kingdom)
Kohout,J. (Czech Republic)
Kolcun,A. (Czech Republic)
Krishnaswamy,A. (United States)
Lanquetin,S. (France)
Lee,T. (Taiwan)
Lewis,J. (New Zealand)
Lin,W. (Taiwan)
Liu,D. (Taiwan)
Maciel,A. (Brazil)
Magnor,M. (Germany)
Maierhofer,S. (Austria)
Mandl,T. (Germany)
Matey,L. (Spain)
Matkovic,K. (Austria)
Mattausch,O. (Austria)
Michoud,B. (France)
Mokhtari,M. (Canada)
Mollá Vayá,R. (Spain)
Montrucchio,B. (Italy)
Mudur,S. (Canada)
Muller,H. (Germany)
Murtagh,F. (Ireland)
Myszkowski,K. (Germany)
OliveiraJunior,P. (Brazil)
Papaioannou,G. (Greece)
Pasko,A. (United Kingdom)
Pasko,G. (Cyprus)
Paulin,M. (France)
Pedrini,H. (Brazil)
Peroche,B. (France)
Pettifer,S. (United Kingdom)
Platis,N. (Greece)
Přikryl,J. (Czech Republic)
Puig,A. (Spain)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Renaud,C. (France)
Ripolles,O. (Spain)
Ritschel,T. (Germany)
Rodeiro,J. (Spain)
Rojas-Sola,J. (Spain)
Rokita,P. (Poland)
Rosenhahn,B. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sanna,A. (Italy)
Schumann,H. (Germany)
Segura,R. (Spain)
Selim,B. (Turkey)
Sellent,A. (Germany)
Sirakov,N. (United States)
Skala,V. (Czech Republic)
Slavik,P. (Czech Republic)
Slusallek,P. (Germany)
Solis,A. (Mexico)
Sousa,A. (Portugal)
Sramek,M. (Austria)
Stroud,I. (Switzerland)
SuarezRivero,J. (Spain)
Svoboda,T. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Theußl,T. (Austria)
Torrens,F. (Spain)
Triantafyllidis,G. (Greece)
Tytkowski,K. (Poland)
Vanecek,P. (Czech Republic)
Vasa,L. (Czech Republic)
Veiga,L. (Portugal)
Vergeest,J. (Netherlands)
Viola,I. (Norway)
Vitulano,D. (Italy)
Wan,T. (United Kingdom)
Wu,E. (China)
Wuethrich,C. (Germany)
Yencharis,L. (United States)
You,S. (United States)
Zach,C. (United States)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhu,Y. (United States)

Journal of WSCG

Vol. 17, No.1-3

Contents

Eissele,M., Sanftmann,H., Ertl, T.: Interactively Refining Object-Recognition System	1
Radziszewski,M., Boryczko,K., Alda,W.: An Improved Technique for Full Spectral Rendering	9
Malik,M.M., Heinzl,Ch., Gröller,M.E.: Computation and Visualization of Fabrication Artifacts	17
Taibo,J., Seoane,A., Hernández,A.: Dynamic Virtual Textures	25
Wenger,S., Fernandez,A., Morisset,J.C., Magnor,M.: Algebraic 3D Reconstruction of Planetary Nebulae	33
Hermann,M., Greß,A., Klein,R.: Interactive Exploration of Large Event Datasets in High Energy Physics	41
Csébfalvi,B., Domonkos,B.: Prefiltered Gradient Reconstruction for Volume Rendering	49
Latapie,S.: Hybrid sort-first/sort-last rendering for dense material particle systems	57
Orthmann,J., Salama,Ch.R., Kolb,A.: Responsive Grass	65
Parys,R., Knittel,G.: Giga-Voxel Rendering from Compressed Data on a Display Wall	73
Bauer,F., Stamminger,M., Meister,M.: Reconstructing Indoor Scenes with Omni-Cameras	81
Elmezain,M., Al-Hamadi,A., Michaelis,B.: A Novel System for Automatic Hand Gesture Spotting and Recognition in Stereo Color Image	89
Stanek,S.: Simple emphatic user interface	97

Interactively Refining Object-Recognition System

Mike Eissele¹

Harald Sanftmann²

Thomas Ertl¹

¹Visualization and Interactive Systems Group ²Visualization Research Center
Universität Stuttgart, 70569 Stuttgart, Germany
{eissele|sanftmann|ertl}@vis.uni-stuttgart.de

ABSTRACT

Existing techniques for object recognition often make use of a combination of multiple algorithms and sensors to achieve adequate results. In this paper we propose a real-time system to efficiently combine multiple object-recognition techniques, appropriate for mobile Augmented Reality applications. We focus on the challenge to differentiate objects with only marginal distinguishing features that can often only be identified from specific points of view, and solve this problem by interactively guiding the user during the recognition process. The system is based on a hierarchy to organize model data and control the corresponding feature-detection techniques as shown in a prototypical implementation. Furthermore, recognition techniques are chosen based on context information, e.g. feature type, reliability of sensor data, etc.

Keywords: Multi-Technique Object Recognition, Mobile Augmented Reality.

1 INTRODUCTION

The problem of object recognition is a common issue in many real-world applications. Marker-based systems are efficient for object identification and pose estimation, but require a deployment of markers to target objects. In contrast, there are numerous scenarios where markers cannot be used, either because of esthetic reasons or technical problems. Therefore, other methods have evolved to provide a marker-less object recognition. For interactive applications it is further required that the recognition processed is performed in realtime. Examples are most Augmented Reality (AR) applications where mobile users interactively control the camera via direct manipulation.

Most available recognition techniques have specific advantages and disadvantages in certain situations. Therefore, many setups exist which make use of multiple sensors—optical, inertial, etc.—with different algorithms to achieve improved results. However, most of these systems are not applicable to mobile scenarios and are designed for very specific problems and do not focus on an easy extensibility with additional sensors or algorithms. Therefore, we proposed a general concept to combine arbitrary object-recognition techniques to build a robust, reliable, and efficient real-time object-recognition system. In contrast to existing sensor-fusion methods the proposed system detects and

selects the most appropriate algorithm to differentiate objects based on various context information.

A specific goal of the proposed system is to efficiently differentiate object classes with a huge number of only marginal different entities. The differentiation can be so fine granulated that even individual object can be uniquely identified. Thereby, distinguishing features might not be captured by any of the available sensors from some locations. The system is targeted for interactive mobile AR applications where virtual geometry is accurately aligned with a real-world image as depicted in Figure 1b,c. Therefore, a primary challenge—when supporting multiple, possibly alternative, algorithms for object recognition—is to prevent a degradation of performance due to numerous object-algorithm combinations that have to be evaluated.

The proposed system prevents such a performance degradation by utilizing a hierarchical structure to organize model data. During the recognition process, the hierarchy is traversed and the number of possible object matches is continuously reduced. The hierarchy also allows to present intermediate recognition results, e.g. object classes, and trigger actions that are needed to further descent the hierarchy, if the system cannot differentiate an object. This triggers can effectively be used to initiate user-operated sensor adjustments on handheld devices that commonly lack of mechanical installation to, e.g., change the view direction of a camera.

2 CONTEXT AWARENESS

The usage of context information to support object recognition is already motivated by Oliva and Torralba in [13]. They show the importance of context for the human visual system and propose to use context data also in Computer Vision systems. They focus on semantic context information of captured scenes, in contrast, we

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



Figure 1: Steps of an interactive refining object recognition: a) recognition of the object class *Auto Data Switch* and presentation of a hint to capture further distinguishing features, as seen in b) or c). For illustration, different models of the *Auto Data Switch* shown in b) and c) are augmented with enhancements.

propose to widen this concept and further include arbitrary information related to the current situation.

Acquisition, (pre-)processing, and storage of arbitrary context data can easily be provided by Nexus, a framework for mobile context-aware applications [4, 5]. The open system offers the possibility to query basic context information or even an estimation of a high-level situation description. The concept allows to connect any data provider and any data consumer. The support of quality metrics further helps to weight the received context data. Using Nexus as an underlying service enables access to a variety of data—e.g. user position, lighting conditions, available hardware, etc.—to control and enhance the process of object recognition.

An overview of the proposed system is given in Section 4.1 followed by a detailed description of the hierarchical structure of model data. The execution of an object-recognition operation is detailed in Section 4.3. Prototype implementation and results are discussed in Section 4.5.

3 MULTI-TECHNIQUE OBJECT-RECOGNITION METHODS

For the survey of previous work, we primarily concentrate on methods which utilize multiple different techniques/sensors for the recognition of objects.

In general, research on object recognition can be separated in two categories: marker-based and marker-less techniques; both are supported by the proposed framework. Object identification and pose estimation based on synthetic markers is, amongst others, shown by Ababsa and Mallem in [1].

The advantage of using hierarchies in terms of decision trees is proposed, e.g., by Mehrotra et al. [12]. They group distinctive features of objects to generate the tree and traverse it during the object recognition phase. Our work is based on this general concept, however we extend several aspects: Support for multiple techniques to evaluate the decisions, even on a per-node basis, the possibility to return intermediate results, and therewith the triggering of actions to allow an unambiguous object recognition or even identification. Also,

Viola and Jones propose to use a degenerated decision tree to achieve fast recognition results [16]. They concatenate multiple weak continue/reject classifiers to build stronger classifiers, however near-equal objects that cannot be differentiated in arbitrary captured views are not handled adequately. Grabner et al. use SIFT-like features in [8] to distinguish objects. The system groups similar objects in a hierarchical structure, however only a single recognition technique is utilized.

Dhome et al. propose a method to find an analytical solution for the attitude of a 3D object in space [7]. Simplifications for the special cases of coplanar lines and three-line junctions are given which reduce the problem to four-degree equations. Beier et al. present an application of Dhome's method on mobile devices [2]. In addition, a simple image-based 2D filter is used to differentiate similar objects. Lowe presents an algorithm that iteratively refines an initially guessed view point via Newton's method [11]. Kang et al. propose a technique to efficiently extract topology information, i.e. line junctions, within an image [9] and use it to perform object recognition and pose estimation [10]. Vacchetti et al. present in [15] a marker-less registration method based on the combination of image feature points and edge tracking. The authors compare different setups for sensor fusion and show that with multiple hypotheses the initial result can be improved. A system which uses vision and inertial sensing for tracking is proposed by You et al. [17]. An inertial sensor provides changes in orientation since the previous frame to estimate new camera orientations used as input for a Computer Vision approach.

If objects that have to be recognized are very similar often a single view is insufficient to distinguish the objects, independent of the applied method. Therefore, a number of systems have been presented that evaluate the best view of an object to achieve a reliable recognition. After a first recognition phase, these *active vision* systems build rules to, e.g., move the camera to a view, which allows further refining the object recognition. However, most active-vision approaches assume that a automated camera movement is available, which is practically impossible for handheld devices.

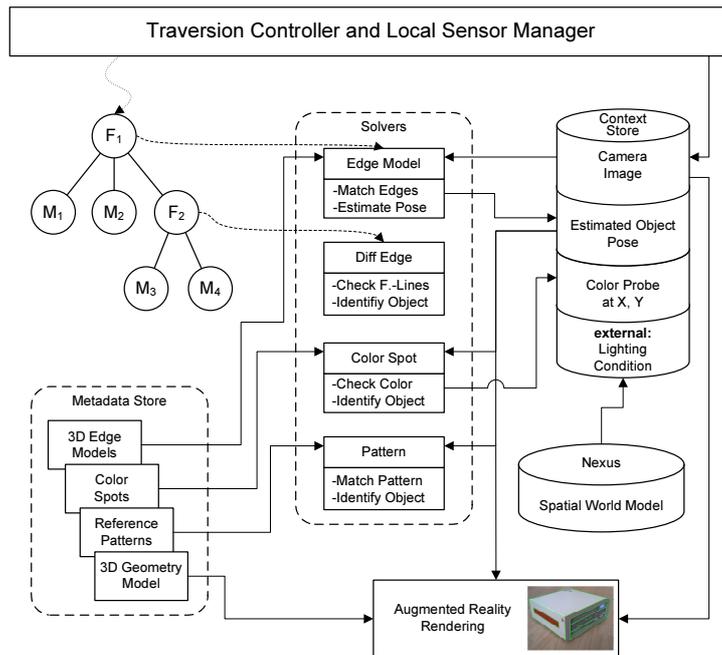


Figure 2: Overview of the object-recognition framework.

Borotschnig et al. present a comparison of three approaches for these so-called active object-recognition systems [3]. The examined techniques are based on different uncertainty calculi: probability theory, possibility theory, and Dempster-Shafer theory of evidence. Reinhold et al. present a statistical appearance-based object-recognition approach combined with an active view point selection [14]. Deinzer et al. presents a non-realtime method which uses a training set to learn *good* next views unsupervised [6].

4 HIERARCHICAL RECOGNITION

There are already existing methods (e.g. [9]) which provide promising results for recognition of dissimilar objects. Most are not well suited for the task of recognizing many similar objects. In contrast, similar objects motivate a grouping of objects with partially equal features, e.g. same overall shape. By iteratively repeating this division within the resulting subgroups an object hierarchy is generated, thereby each distinguishing feature may be detected using a different algorithm.

4.1 System Components

The architecture of our presented object-recognition system is illustrated in Figure 2. All algorithms for object recognition need some information about the models that have to be recognized. This can be reference images, texture information, 3D models, or any other information. Such information—in the following referred to as metadata—is typically generated in an off-line preprocessing task for each node and stored in the *Metadata Store* as shown in Figure 2.

The data contained in the metadata store is primarily accessed by integrated solvers. The framework is based on the concept that multiple different solvers—seen in the center of Figure 2—are utilized subsequently during the object recognition. This way, arbitrary object-recognition techniques using various metadata can be integrated and combined to calculate the final result of the object recognition. In addition, solvers have access to a further data source the so-called *Context Store*.

The context store’s content can be considered as data that describes the current conditions of the application, environment, or any other attribute which might dynamically influence the recognition. We use this store for locally acquired sensor data, intermediate recognition results, and *external* context data from Nexus [4].

The previously mentioned hierarchy with different algorithms (F_x) to detect features is shown in the upper left corner of Figure 2. The traversing of this hierarchy is controlled by a simple controller which executes the referenced solvers (Fig. 2, dashed arrows).

4.2 Organization of Recognition Nodes

The hierarchical structure used for the recognition (recognition tree) utilizes several node types that define different behaviors that are triggered during the traversal of the graph.

A basic node type is the *Search Node F*. It references multiple alternative solvers S_x that are adequate to iterate through all of the Search Node’s children and search for the best matching. A behavior similar to simple traditional recognition systems that iterate through all integrated object models can be simulated this way, where all models M_x are checked for a match in se-

quence. A corresponding example recognition tree is shown in Fig. 3. During traversing, solver S_A and S_B access corresponding metadata, referenced by the current node, e.g. relevant line features of real-world objects.

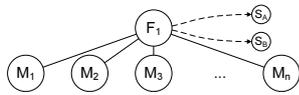


Figure 3: A simple recognition tree to simulate a traditional recognition system which linearly checks each model to search for the best match.

An advantage of the hierarchical structure is that it allows to group similar objects based on common features. These intermediate group nodes reference common metadata of all their child objects that is stored in the *Metadata Store*. Meta information that is only adequate for intermediate nodes may also be stored in the metadata store as it is required by solvers to match intermediate nodes, like a model of common feature lines of a group of objects. Furthermore, the framework is able to present intermediate results of the object recognition even if it cannot entirely differentiate all objects, referenced by a *Search Node*. Applications may already benefit from such intermediate results—as presented in our prototype in Section 4.5—to, e.g., show a coarse 3D proxy model presenting the common appearance of the entire model group. An example configuration is depicted in Figure 4. The search node F_1 will execute solver S_A during object-recognition traversal to differentiate models $M_{1..3}$ and the group of models subsumed under F_2 . Models $M_{4..6}$ are further distinguished based on the referenced metadata of F_2 .

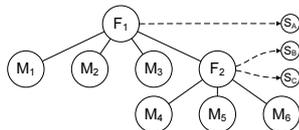


Figure 4: Hierarchical organization of the models. F_1 differentiates models $M_{1..3}$ and the model group of F_2 . Models $M_{4..6}$ are distinguished via node F_2 .

Context-Switch Node C is a node type that is able to route the traversing of the tree dependent on context attributes. Context-switch nodes can also have a number of child nodes but do not reference any solvers, they only evaluate context data to decide how the traversing continues. This way, alternative sequences for the recognition can be integrated in the mostly static hierarchy. Figure 5 shows a configuration where the context-switch node C_1 decides into which child the traversing descends, based on context information like the current pose of the to-be-identified object. As can be seen in Figure 5 the two subgraphs are simply swapped versions of each other. This way, sequences where the system benefits most—e.g. does not have to request user interaction (Section 4.3)—can dynamically be selected.

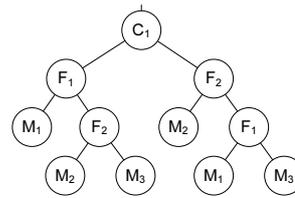


Figure 5: Alternative sequences for the recognition process via a Context Switch C_1 .

4.3 Object Recognition by Traversing the Recognition Tree

The object recognition is performed by traversing the recognition tree and, dependent on the node type, processing the results after executing the referenced solvers to select the child node in which to descend. Exemplary setups of recognition trees are illustrated in Figure 6. We integrated multiple solvers that identify objects based on completely different features: 3D geometry, color, patterns, or line features. A more detailed description of the utilized techniques is given in Subsection 4.5. An important aspect of the proposed system is that the knowledge gained about the current and previous iterations of the object recognition, e.g. the pose of the to-be-identified object or the estimated camera position in previous iterations, is gathered and stored as context data. During traversing of the recognition tree, subsequent solvers have access to this information and may consume, correct, or extend it.

If a leaf of the hierarchy is reached during the traversing then a model has unambiguously been identified and the recognition task is finished, returning the identified object and the gathered context information (see Figure 6a, following steps $I_1:1$, $I_1:2$, $I_2:1$). An important advantage of using a model hierarchy is that our system is able to descend the hierarchy as soon as an adequate match is found in the referenced models, as similar models will be summarized using a group in the hierarchy. In contrast, simple approaches would have to linearly check *each* referenced model if it matches to find the best matching which could potentially be the last reference.

Whenever a node can no further distinguish its children and therefore cannot further descend the recognition tree, intermediate recognition results are returned. For that purpose, our system supports a novel interactive refinement of intermediate recognition results by triggering actions that help the system to further differentiate the objects as can be seen in Figures 7, 1a, and 6a, following steps $I_1:1$, $I_1:2$, and $I_1:3$. This feature can be used, e.g., to instruct the user to perform appropriate camera movements and optimize the point of view to capture additional features (see Fig. 6a step $I_2:1$). User performed adjustments are therefore utilized to compensate for the lack of automatic (e.g. mechanical) installations, which are

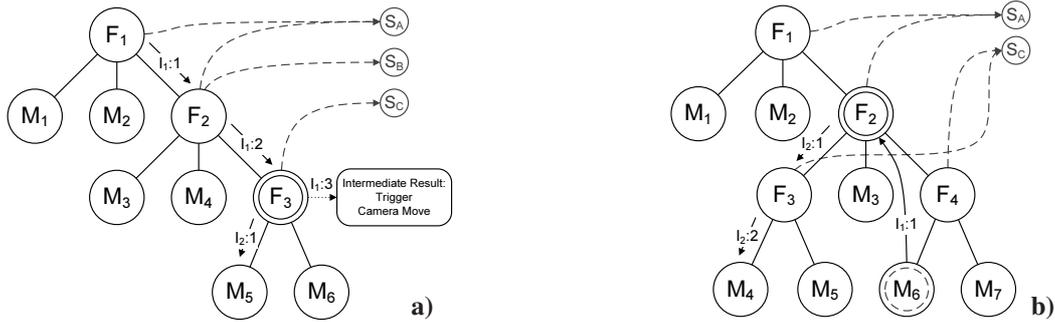


Figure 6: Hierarchical organization of models via multiple search nodes F_x . The recognition performed in a) stops at the intermediate model referenced by F_3 . Afterwards a trigger is executed to request a camera movement ($I_1:3$) to capture further distinguishing features. In b) the previously identified model has changed, therefore the system performs a backtracking ($I_1:1$) and initiates the next iteration at F_2 .



Figure 7: During object recognition the system triggered a request to adjust the lighting condition.

unavailable on handheld devices. Within our prototype we limited these actions to manual camera-movement (Fig. 1a) and light-adjustment (Fig. 7) commands, but other actions—automatic or requests for manual adjustments—like adjusting camera shutter/focus might be triggered.

For achieving a high performance in the recognition phase, the hierarchy helps in two ways: First, the number of models that have to be searched within each search nodes are reduced due to the tree-like structure and second, the temporal coherence—i.e. in most subsequent frames the same object is captured—can effectively be used to skip large parts of the recognition tree. This is achieved by starting the traversing at the node returned in the previous iteration, symbolized as double-framed nodes in Figure 6. If the starting node is not the recognition-tree root the system has to check if the to-be-recognized object is still the same. Furthermore, some context information might get invalid since the previous frame and has to be updated, e.g. due to slight camera movements. Therefore, the system has to ensure that the information required by subsequent nodes are up-to-date by executing the corresponding solvers. This dependency can be determined and stored in a pre-processing step. For the prototypical implementation, we optimized the restart by using a combined solver that checks for a change of the object and estimates its new pose. The system simply checks if the object is

still the same by trying to match line features that are referenced by the node where the recognition process continued. The matching is executed quite fast since line features are already known and a good approximation of the camera position is provided as a result of the previous iteration. For the prototype (Section 4.5) the estimation of the camera movement between subsequent iterations was improved using an inertial sensor to measure the acceleration and approximate the new camera position.

If the matching returns a positive result, the newly corrected pose estimation is stored as context information and the traversing continues, thereby keeping all recognition results of previous iterations. This is illustrated by step $I_2:1$ in Figure 6a. In contrast, if the verification fails the system assumes that the object which has been recognized in the previous iteration has changed and therefore has to check if other objects are present in the captured scene.

Therefore, the system can simply *reset* and start the recognition from the root of the recognition tree. This, however, will result in an inefficient behavior whenever an object cannot be recognized continuously in subsequent iterations: The system will have to descend the entire hierarchy. To overcome this limitation a simple backtracking mechanism is integrated to ensure that the system remains efficient, i.e. restarting the recognition from a previous node on the node path back to the root. This recursive process continues until the recognition is able to descend the hierarchy again or—in the worst case—a restart of the recognition is initiated at the root node of the tree.

During construction time of the recognition tree a link can be stored per node that is followed during backtracking to skip in-between nodes to increase the efficiency as shown in Figure 6b step $I_1:1$. Afterwards, a following iteration ($I_2:1, I_2:2$) traverses again to a leaf node.

In the proposed interactive refining, special cases occur that are annoying for users: The system might re-

quest a camera movement to the right to continue the traversing. However, a following search node might request a movement back to the left whereby it possibly would have been satisfied with the camera position at the beginning. In worst cases, users are required to adjust the, e.g., camera view multiple times whereas one adjustment would have been sufficient. Our system overcomes such scenarios by integrating context-switch nodes C to select alternative sequences for the recognition. The subgraphs of C nodes are simply permuted in their order of execution as illustrated in Figure 5 and selected based on the current context.

4.4 Using external Context Information for Recognition-Technique Selection

During traversing of the hierarchy the system gathers context information required by subsequent solver nodes to perform their task. In addition to this internally generated data, *external* context information also helps to control and improve the recognition process.

Each search node of the recognition tree that summarizes multiple nodes, references at least one technique to search through its children. The selection of the solver to differentiate features of referenced models is done in a preprocessing step during the recognition-tree construction. Therefore, an algorithm is chosen which is assumed to deliver best results on average in terms of reliability, robustness, or performance. However, selecting the most adequate technique to distinguish groups of similar objects during the setup of the recognition tree in a pre-processing step is not always possible: Changes in the environmental context, application states, sensor quality, etc. might occur during runtime and therewith invalidate the selection of recognition algorithms based on these attributes.

In order to overcome this limitation, the proposed system allows to assign multiple alternative recognition techniques per node that are dynamically selected, based on *internal* or *external* context information. The external context information is provided via Nexus [4, 5], as mentioned in Section 2.

4.5 Prototype System and Results

The presented recognition system has been specifically designed to share the context and metadata store with other application parts. Therefore, Augmented Reality visualizations can easily access the position and orientation information using the context store. The prototypical implementation of a mobile interactive assistant system to help users to identify and augment objects makes use of this concept. A key concept of the system is that very similar objects are differentiated using the proposed refining technique for controlled user intervention. An exemplary target application for the prototype is an information system for customers and consultants who are interested in HIFI appliances. These

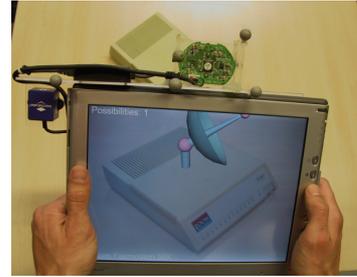


Figure 8: The prototype hardware in use. A standard TabletPC was equipped with a webcam (top middle) and an inertial sensor (top left).

items have many similar aspects which cannot easily be differentiated. Augmented Reality provides an intuitive way to present different instances of an object—probably not yet available—like extra attachments, different colors, or even custom case modifications. For evaluation of the proposed concept, a prototype as seen in Figure 8 was built.

4.5.1 Implementation

Multiple solvers have been implemented to support the recognition of various feature types that might be useful to differentiate similar objects. The most advanced solver is—in addition to identification—also able to estimate the pose of objects. It is implemented using Computer Vision methods: First, feature lines and three-junctions are searched in the captured image. These are linked to the model’s geometry description—stored in the metadata store—to generate hypotheses of possible models and their orientation [7]. The second part of the solver is based on a technique proposed by Lowe [11] and is used to check generated hypotheses and further improve the pose-estimation accuracy by minimizing the matching error. We refer to this solver as *edge-model solver* S_E .

A cut-down version of the *edge-model solver* is integrated to differentiate similar models, where one model has additional feature lines. A prerequisite for this solver is a pose estimation, calculated by any previous *search node* in the recognition tree. The solver can then project the additional line features, stored for one specific object of a group, and search for matching edges in the captured image, therefore the solver is termed as *diff-edge solver* S_D . Distinguishing features, e.g. additional lines, might be visible only from a specific point(s) of view therefore a differentiation of the objects based on a captured image from an arbitrary view is not always possible and an intermediate result might be returned. As our approach is especially targeted for mobile clients, where often only a single camera with a fixed viewing direction is available, we cannot expect that other sensors might capture a distinguishing feature. The novel approach of our proposed interactive refining technique to solve such un-



Figure 9: Rendering of augmented camera images at different steps of the recognition process. Detected and vectorized edges of the camera images are shown in a). A hypothesis (blue) generated by the approach of Dhome [7] and model's feature lines are seen in b). A finally checked and improved solution—based on Lowe [11]—is displayed in c). The red line segment could not be matched in the camera image.

determined cases is that solvers can report intermediate results with hints/triggers that state which changes have to be made in order to continue the recognition. This includes change requests for, e.g., camera movements or lighting conditions that are displayed to the users as can be seen in Figure 1a and Figure 7.

In contrast to line features, the *color-spot solver* S_C is able to check the color value at pre-defined locations on the object surface. It simply projects pre-defined color-probe locations using the previously estimated object pose to the image space and examines the pixel color in the captured image. Therefore, objects with (partially) different colors can efficiently be identified even if colored features are only visible at specific points.

Similar real-world objects are often labeled or marked in order to express their differences. Good examples are electronic appliances like HIFI components which might only be different in their insides and probably their serial numbers. Our prototype utilizes a recognition approach to compare previously acquired reference images to the captured image information to support an identification based on patterns. This *pattern solver* S_P benefits of a previously calculated pose estimation in two ways: It is able to determine if the pattern is entirely visible in the camera image, i.e. it is not hidden or occluded, and the perspective distortion is a priori known due to the previously calculated object orientation and the pattern location in model coordinates, stored as metadata.

Many additional techniques can easily be integrated to detect more complicated features like, e.g., a solver to recognize curved surfaces. But also identification components like optical character recognition or barcode scanners can be applied.

The proposed recognition system is applicable for various applications, for the prototype we chose to implement an Augmented Reality rendering module to present real-world aligned virtual information. It is used to show information about the traversal of the recognition tree and object information. If the system is unable to totally identify an object, only its category is displayed. We further make use of AR rendering to

track and evaluate the recognition process and its accuracy. The precision of the object's pose estimation can easily be seen via an augmentation of the camera image with superimposed 3D geometry model (see Fig. 9).

For the proposed scenario, presentations based on AR further benefit from the possibility to show different virtual instances of objects, similar to the illustrations in Figures 1b,c. These renderings depict two different augmentations, which might represent future configurations or not-in-stock items.

As mentioned in Section 4.3, intermediate nodes may trigger actions to improve the recognition. We implemented triggers to initiate camera-movement requests which display messages to guide users interactively how the camera should be positioned in order to achieve better recognition results, as seen in Fig. 1a, and Fig. 7.

4.5.2 Results

For the evaluation of the prototype implementation a data set with five computer-appliance objects were used. Two objects are identical except for a red stripe on one object. Therefore, these objects can only be distinguished if the part where the red stripe is located on is within the camera view. A direct comparison to existing systems cannot be given, since the proposed setup is rarely examined, often a specialized algorithm is utilized where the objects used to evaluate the system fit to the proposed algorithm. The theoretical complexity in terms of executions of *solver-model* pairs in the hierarchical implementation is optimally $O(\log(n))$. For a simple linear search method the complexity is $O(n)$ which corresponds to the worst case of our approach. In practical setups the system therefore achieves a performance in-between both extremes. However, these numbers strongly depend on the number, type, and quality of the objects and the structure used for the recognition tree. The implemented recognition algorithms and the selected techniques also have a great influence to the overall performance.

Measurements in Table 1 present the performance for an Intel Core2 Quad Q6600@2.4GHz CPU (only a sin-

gle core was utilized). The video stream of the camera was simulated with a static 320×240 resolution image in order to achieve comparable results.

	<i>non-coherent</i>	<i>coherent</i>
<i>Preparation</i>	21.3	21.3
<i>hypOther</i>	163.1	-
<i>hypValid</i>	5.8	-
<i>hypCheck</i>	0.8	0.8
<i>Rendering</i>	1.4	1.4
Overall	192.4 ms	23.5 ms

Table 1: (Re)start performance of an object-recognition process utilizing temporal coherence.

The first phase of our approach is equal for both cases: Undistortion of the camera image, generation of a monochromatic image, execution of a Sobel/Canny edge detection, and the merging of collinear line fragments which is summarized as *Preparation*. In the *non-coherent* case, where no object registration and pose estimation is available from previous frames, a large number of hypotheses have to be evaluated. The timing values of *hypOther* refer to hypotheses that are evaluated with 3D object models which do not correspond to the captured camera image. The *hypValid* measurements refer to hypotheses that are evaluated with a matching 3D object model. In the *coherent* case we already have a valid object pose from a previous iteration which is already accurate, as the camera is fixed during the evaluation. Timings of *hypCheck* refer to the Lowe based approach for checking and improving the pose-estimation hypothesis. Timings for display of the captured camera image and optional augmentations are summarized in *Rendering*.

The measurements show the benefit of utilizing temporal coherency as the most time-consuming part of the algorithm is efficiently skipped. With the previously mentioned performance improvement due to the utilized hierarchy the system is especially suited for mobile, interactive applications.

5 CONCLUSION

We have presented an approach to combine arbitrary recognition techniques within a single object-recognition and pose-estimation system. A hierarchical structure is utilized to achieve highly efficient and robust object recognition. The recognition process is performed by traversing the hierarchy whereby referenced solvers are executed. The returned result is either the unambiguous object identification or an intermediate result. To improve intermediate results the system can trigger actions—e.g. relocation of the camera—in order to capture additional important features for the recognition process. These triggers are used to implement an interactive process to refine recognition results by providing hints to guide users

how to improve the recognition, thereby providing the possibility to adjust an otherwise static setup to sensors.

In future we will concentrate on automatic construction of a balanced model hierarchy.

REFERENCES

- [1] F. Ababsa and M. Malle. Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems. In *VRCAI '04: Proceedings of the ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 431–435. ACM, 2004.
- [2] D. Beier, R. Billert, B. Brüderlin, D. Stichling, and B. Kleinjohann. Marker-less vision based tracking for mobile augmented reality. In *2nd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, page 258, 2003.
- [3] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. A comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition. In *Computing*, volume 62, pages 293–319, 1999.
- [4] Collaborative Research Centre (SFB627). Nexus: Spatial world models for mobile context-aware applications. <http://www.nexus.uni-stuttgart.de>, 2008.
- [5] P. Coschurba, U. Kubach, and A. Leonhardi. Research issues in developing a platform for spatial-aware applications. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 153–158. ACM Press, 2000.
- [6] F. Deinzer, J. Denzler, and H. Niemann. Viewpoint selection - a classifier independent learning approach. In *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 209–213, 2000.
- [7] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determination of the attitude of 3-d object from a single perspective view. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 11, pages 1265–1278, 1989.
- [8] M. Grabner, H. Grabner, and H. Bischof. Fast visual object identification and categorization. In *Proceedings of NIPS Workshop in Interclass Transfer 2005*, pages 1–8, 2005.
- [9] D. J. Kang, J.-E. Ha, and I.-S. Kweon. Fast object recognition using dynamic programming from combination of salient line groups. *Pattern Recognition*, 36(1):79–90, 2003.
- [10] D. J. Kang, J.-E. Ha, and I.-S. Kweon. 3-D Pose Estimation Algorithm for Model Based Vision. In *6th Asian Conference on Computer Vision (ACCV)*, pages 115–119, 2004.
- [11] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. In *Artificial Intelligence*, volume 31, pages 355–395. Elsevier, 1987.
- [12] R. Mehrotra, W.I. Grosky, and F.K. Kung. Decision-tree based two-dimensional object recognition. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.*, pages 1380–1383, 1988.
- [13] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520–527, 2007.
- [14] M. Reinhold, F. Deinzer, J. Denzler, D. Paulus, and J. Pösl. Active appearance-based object recognition using viewpoint selection. In *VMV*, pages 105–112, 2000.
- [15] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR04)*, pages 48–57, 2004.
- [16] P. Viola and M. J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [17] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proceedings of IEEE Virtual Reality*, pages 260–267, 1999.

An Improved Technique for Full Spectral Rendering

Michal Radziszewski
AGH, Krakow, Poland
mradzisz@student.agh.edu.pl

Krzysztof Boryczko
AGH, Krakow, Poland
boryczko@agh.edu.pl

Witold Alda
AGH, Krakow, Poland
alda@agh.edu.pl

ABSTRACT

In this paper we present an improved approach to full spectral rendering. The technique is optimized for quasi-Monte Carlo ray tracing, however the underlying physical theory can be applied to any global illumination scheme. We start with explanation of the necessity of full spectral rendering in any correct global illumination system. Then we present, step by step, a rendering scheme using full spectrum simulation. First, we give details on a random point sampling as a method of representing spectra, then we introduce improved spectral sampling technique, designed to reduce variance of image of wavelength dependent phenomena, and finally we show how to integrate the novel sampling technique with selected ray tracing algorithms.

Keywords: Full spectrum, quasi-Monte Carlo, ray tracing, rendering.

1 INTRODUCTION

The color phenomenon is caused by a spectral mixture of light, perceived by the human visual system. However, the human visual system cannot distinguish between arbitrary spectral light distributions. Different spectra, which are indistinguishable by human observers, are called metamers. The space of colors recognizable by human observers contains only three independent values, hence the popularity of three component color models.

There are many color models in computer graphics, however most are designed for a specific purpose only. The most common are: RGB designed for displaying images, CMYK for printing and HSV for easy color selection by user. All of these models are to some degree hardware dependent. There is, however, a standard model based on XYZ color space, which is independent of any hardware and can represent all the colors recognizable by a human observer. It was defined by the CIE (Comission Internationale de l'Eclairage) as three weighting functions to obtain x , y and z components from arbitrary spectra. Nevertheless, neither of these models is well suited for rendering, where direct calculations on spectra are the only way to produce correct results [4, 9].

2 NECESSITY OF FULL SPECTRUM

The RGB model is often used for rendering color images. However, this is an abuse of it, since RGB based rendering does not have any physical justification. The

model was designed for storage and effective display of images on a monitor screen, but not for physically accurate rendering. The light reflection computation under the assumption of elastic photon scattering is performed by a multiplication of a spectrum that represents an illumination and a spectrum describing a surface reflectance. This multiplication actually must be performed on spectral distribution functions, not on RGB triplets, in order to get proper results.

The RGB based reflection of white light, or light with smoothly varying spectrum, from a surface with smoothly varying reflectance, typically does not produce substantial inaccuracies. However, when at least one of spectra has large variation, the simulation using RGB model becomes visibly incorrect (see Figure 1, for example). Moreover in global illumination, due to multiple light scattering, even white light becomes colorful, causing scattering inaccuracies to accumulate. This makes RGB based global illumination results unable to accurately capture the physical phenomena.

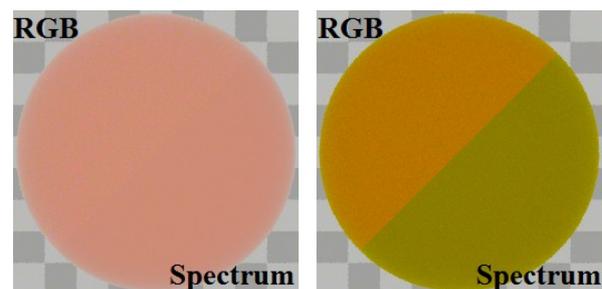


Figure 1: Left image: copper sphere illuminated by a D65 white light. Right image: copper sphere illuminated by a triangular spectral distribution stretched from 535nm to 595nm. Top left half: an RGB model with 645nm, 526nm and 444nm wavelengths. Right bottom half: our full spectral model. For clarity, only diffuse reflection is calculated.

In addition, the most visually distracting error from using an RGB model appears in simulation of phenom-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ena like dispersion. Whenever RGB based light, from a light source with almost parallel output rays, hits a prism, it is scattered into three bands instead of continuous full spectrum, and the rest of the image remains dark (see Figure 2), which looks unrealistic. Using a full spectrum representation gives a continuous rainbow of colors. However, good-looking results may be obtained by an RGB representation if the light source angular distribution is conical and divergent enough. A similar trick is a basis of a simple Nvidia shader demo [5]. An address of the texture on a surface, which is seen through glass, is offsetted independently for each channel. If texture data is blurred enough, the resulting 'spectrum' is smooth. Nevertheless, both of these methods do not have any physical significance, and obviously are incorrect, but, in some conditions, can look convincing.

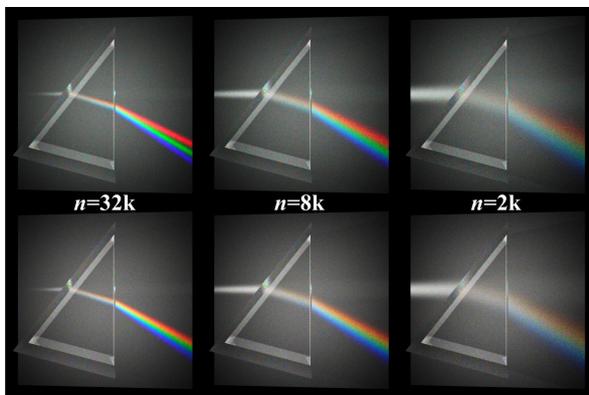


Figure 2: Dispersion on a prism. Top row: RGB model with 645nm, 526nm and 444nm wavelengths. Bottom row: physically correct full spectrum. The light collimation is controlled by a Phong-like function $I \cos^n(\phi)$, with exponent n decreased four times in each subsequent column, and intensity I doubled to compensate light scattering.

3 RELATED WORK

A general description of many popular color models can be found in Stone [16]. Devlin et al. [1] provide references related to data structures for full spectral rendering and algorithms for displaying spectral data. There are several works dedicated to simulation of particular spectral based phenomena. Wilkie et al. [22] simulated dispersion by means of classic (deterministic) ray tracing. Rendering of optical effects based on interference attracted a fair amount of attention. Reflection from optical disks is presented in Stam [15] and Sun et al. [17]. Algorithms for accurate light reflection from thin layers can be found in Gondek et al. [7] and Durikovic and Kimura [3]. The latter paper also shows how this algorithm can be run on contemporary GPUs.

Many papers present methods for representing and operating on spectral data. Peercy [12] designed a spec-

tral color representation as a linear combination of basis functions, chosen in a scene dependent manner. Different algorithm using basis functions is described by Rougeron and Peroche [14]. It uses adaptive projection of spectra to hierarchical basis functions. Sun et al. [18] proposed a decomposition of spectra on smooth functions and set of spikes. Evans and McCool [4] used clusters of many randomly selected spectral point samples. Johnson and Fairchild [9] extended OpenGL hardware rasterization to support full spectra.

Dong [2] points that typically only a part of the scene needs a full spectral simulation and using RGB together with full spectrum can accelerate rendering at cost of only slight quality loss. Ward [21], however, designed a three-component model optimized for rendering, which typically produces images with an acceptable yet imperfect quality, but the model is not general enough and cannot simulate wavelength dependent phenomena like dispersion.

4 REPRESENTING FULL SPECTRA

Full spectral rendering requires an efficient method for representing spectral data. The most common techniques are based on linear combinations of carefully selected basis functions [12, 14, 18] and point sampled continuous functions [4]. Effectiveness of the linear combination approach is strongly dependent on the actual functions and their match to scene spectral distribution. However, the natural solution in Monte Carlo based rendering system is a random point sampling.

4.1 Random Point Sampling

Random point sampling produces noise at low sampling rate, but well-designed variants of this technique converge quickly. Point sampling can effectively handle smooth (like tungsten bulbs) light distributions and very narrow spikes (like neon bulbs) in the same scene. The two greatest strengths of this technique are: randomly selected wavelengths and well defined wavelength value for each spectral sample. The first one ensures correctness, since when more samples are computed, the more different wavelengths are explored, and due to the law of large numbers, the rendering result converges to the true value. The second allows simulating wavelength dependent effects like dispersion at the cost of additional color noise.

It is worth to note that wavelength dependent phenomena cannot be simulated correctly with algorithms based on linear combinations of basis functions with non-zero extent in wavelength space. Even if spectra are represented by unique non-zero coefficients, the corresponding basis functions still have some finite extent, which prevents from doing exact computations with explicit wavelength required.

The simplest approach to point sampled spectra is generation of a single spectral sample per light trans-

port path. However, according to Evans and McCool [4], this technique is inefficient, since it causes a lot of color noise. They proposed using a fixed number of several spectral samples (called a cluster of samples) traced simultaneously along a single light path, which substantially reduces variance with minimal computational overhead.

4.2 Basic Operations

The implementation of multiplication, addition, minimum, etc. operators are obvious, since it is enough to perform appropriate calculation per component, as in RGB model. However, when using full spectrum, computing luminance is a bit more difficult. Particularly, luminance of a spectrum which describes reflectivity of a surface, by definition must be in $[0, 1]$ range.

However, computing luminance as a Monte Carlo quadrature of product of reflectance spectrum $r(\lambda)$ and scaled CIE y weighting function, may randomly lead to numerical errors causing luminance to exceed 1.0 threshold. The equation:

$$L \approx \sum_{i=1}^n \frac{r(\lambda_i)y(\lambda_i)}{p(\lambda_i)} / \sum_{i=1}^n \frac{y(\lambda_i)}{p(\lambda_i)}, \quad (1)$$

where $r(\lambda)$ is the reflectance, $y(\lambda)$ is CIE y weight and $p(\lambda_i)$ is a probability of selecting given λ_i , solves the issue. It guarantees that the luminance is in $[0, 1]$ range, provided that $r(\lambda)$ is also in the specified range.

Wavelength dependent effects can be handled as proposed by Evans and McCool [4] for specular dispersion – by dropping all but one spectral sample from a cluster. This is done by randomly selecting a sample to preserve, with uniform probability. All the samples, except the selected one, are then set to zero, and the power of the chosen one is multiplied by the cluster size. Then the wavelength parameter becomes well defined, and further computations are performed with usage of its actual value. However, when simulated phenomena are not optically perfect, like in Phong-based glossy refraction, it may be more efficient to trace the whole cluster, scaling power of each sample independently. We examine this approach in detail in the next section.

5 SAMPLING OF SPECTRA

Evans and McCool [4] simulate wavelength dependent phenomena by tracing only one spectral sample per path. This particular approach is always correct, and is necessary when a phenomenon is optically perfect, such as refraction on idealized glass. However, when the scattering is not ideal, dropping all but one spectral sample from a cluster, while still being correct, might be extremely wasteful and inefficient. In this section we propose a substantially improved technique.

5.1 Single Scattering Model

For testing purpose, a refraction model with an adjustable, wavelength dependent refraction and imperfection introduced by Phong-based scattering [13], with controllable glossiness is used. An extension to Walter et al. microfacet based refraction [20] supporting dispersion gives better results, but their model is much more complicated and therefore would make evaluation of spectral sampling difficult. Nonetheless, since we have never made assumptions about scattering model, our results are general and, as we have tested, applicable to any wavelength dependent phenomena. For clarity, all tests are based on a single scattering simplification (i.e. light is refracted once, when it enters into glass only). The x component in CIE XYZ space in outgoing direction ω_o is then described by the following formula:

$$I_{CIE_x}(\omega_o) = \int_{\Lambda} \int_{\Omega} f_s(\omega_i, \omega_o, \lambda) L_{\lambda}(\omega_i, \lambda) \cdot w_{CIE_x}(\lambda) d\sigma^{\perp}(\omega_i) d\lambda, \quad (2)$$

where Λ is the space of all visible wavelengths, Ω is the space of all direction vectors, $L_{\lambda}(\omega_i, \lambda)$ is the radiance incoming from direction ω_i , w_{CIE_x} is the CIE weight for x component, and $\sigma^{\perp}(\omega_i)$ is the projected solid angle measure. The y and z components can be evaluated in a similar way. In the rest of this section, the Formula (2) is written in a simplified, still not confusing, form:

$$I = \int_{\Lambda} \int_{\Omega} f(\omega, \lambda) L(\omega, \lambda) w(\lambda) d\sigma^{\perp}(\omega) d\lambda. \quad (3)$$

5.2 Basic and Cluster Based Monte Carlo Estimators

The Monte Carlo method (Equation 14) can be applied to evaluate the two integrals from Formula (3), which leads to the following estimator:

$$I \approx \frac{1}{N} \sum_{i=1}^N \frac{f(\omega_i, \lambda_i)}{p_{\Omega}(\omega_i, \lambda_i)} \frac{w(\lambda_i)}{p_{\Lambda}(\lambda_i)} L(\omega_i, \lambda_i), \quad (4)$$

where p_{Ω} is the probability of selection of a given ω_i evaluated with the $\sigma^{\perp}(\omega)$ measure on Ω and p_{Λ} is the probability of selection of a given λ_i . Quality of this estimator, and all the further estimators in this section, relies on the assumption that scattering model offers proper importance sampling (Equation 15), i.e. $f(\omega, \lambda) \propto p_{\Omega}(\omega, \lambda)$ is roughly satisfied. However, this basic estimator is inefficient, because it forces the numbers of spectral and directional samples to be equal. Each directional sample requires additional rays to be traced, which is computationally expensive, while spectral samples are almost for free. This explains the advantage of clusters of spectral samples over a single spectral sample approach.

The main improvement over Evans and McCool method is tracing a full cluster of spectral samples,

even when wavelength dependent phenomenon is encountered. Wavelength dependence can be defined precisely as the dependence of p_Ω on λ . If scattering is not wavelength dependent, directional sampling is not wavelength dependent as well, i.e. $p_\Omega(\omega, \lambda) \equiv p_\Omega(\omega)$. In our method, a particular spectral sample λ_i^s is selected at random from a cluster, and its value is used for sampling ω_i^s . This leads to the color estimator in the form:

$$\begin{aligned} I &\approx \frac{1}{NC} \sum_{i=1}^N \sum_{j=1}^C \frac{f(\omega_i^s, \lambda_i^j)}{p_\Omega(\omega_i^s, \lambda_i^s)} \frac{w(\lambda_i^j)}{p_\Lambda(\lambda_i^j)} L(\omega_i^s, \lambda_i^j) = \\ &= \frac{1}{NC} \sum_{i=1}^N \frac{1}{p_\Omega(\omega_i^s, \lambda_i^s)} \cdot \\ &\quad \cdot \sum_{j=1}^C f(\omega_i^s, \lambda_i^j) \frac{w(\lambda_i^j)}{p_\Lambda(\lambda_i^j)} L(\omega_i^s, \lambda_i^j), \end{aligned} \quad (5)$$

where N is the number of traced clusters, C is the number of samples in each cluster, and p_Ω is the probability of selecting scattering direction, calculated for the selected wavelength λ_i^s . The estimator (5) can be more efficient than estimator (4), since it traces C spectral samples at the minimal additional cost. On the other hand, it may deteriorate the importance sampling quality significantly. This happens because all samples with potentially wildly different $f(\omega_i^s, \lambda_i^j)$ values are traced, and just one probability $p_\Omega(\omega_i^s, \lambda_i^s)$ which matches the shape of $f(\omega_i^s, \lambda_i^s)$ only, is used. Whenever a direction ω_i^s with low probability $p_\Omega(\omega_i^s, \lambda_i^s)$ is chosen at random, and at least one of the $f(\omega_i^s, \lambda_i^j)$ has a relatively large value in that direction, the value is no longer cancelled by the probability, leading to the excessively high variance in the rendered image. Moreover, the estimator (5) is incorrect whenever $\exists \lambda_i^s, \omega_i^s : p_\Omega(\omega_i^s, \lambda_i^s) = 0$ and $\exists \lambda_i^j : f(\omega_i^s, \lambda_i^j) > 0$, particularly when a wavelength dependent phenomenon is optically perfect, i.e. its f is described by a δ distribution. Thus, the initial version of our new approach is not always better than the traditional technique of tracing only one spectral sample. The question is when the new technique exhibits lower variance and when it does not.

5.3 Multiple Importance Sampling Estimator

Fortunately, the variance issue can be solved automatically. Simple modification of the estimator (5), which incorporates Multiple Importance Sampling [19] (see Appendix A), gives a better estimator with variance as low as possible in a variety of conditions. The new improved estimator is constructed from the estimator (5) multiplying each cluster by C and a weight W_i^s equal to:

$$W_i^s = \frac{p_\Omega(\omega_i^s, \lambda_i^s)}{\sum_{j=1}^C p_\Omega(\omega_i^s, \lambda_i^j)}, \quad (6)$$

where $p_\Omega(\omega_i^s, \lambda_i^s)$ is the probability with which the scattering direction is selected, and the values $p_\Omega(\omega_i^s, \lambda_i^j)$ are hypothetical probabilities of selecting the sampled direction if using λ_i^j value instead. This leads to the final estimator:

$$\begin{aligned} I &\approx \frac{1}{NC} \sum_{i=1}^N \frac{CW_i^s}{p_\Omega(\omega_i^s, \lambda_i^s)} \cdot \\ &\quad \cdot \sum_{j=1}^C f(\omega_i^s, \lambda_i^j) \frac{w(\lambda_i^j)}{p_\Lambda(\lambda_i^j)} L(\omega_i^s, \lambda_i^j) = \\ &= \frac{1}{N} \sum_{i=1}^N \frac{1}{\sum_{j=1}^C p_\Omega(\omega_i, \lambda_i^j)} \cdot \\ &\quad \cdot \sum_{j=1}^C f(\omega_i^s, \lambda_i^j) \frac{w(\lambda_i^j)}{p_\Lambda(\lambda_i^j)} L(\omega_i^s, \lambda_i^j). \end{aligned} \quad (7)$$

Assuming that a scattering model provides proper importance sampling, the estimator (7) leads to a low variance result. Moreover, the estimator (7) is correct whenever scattering model is correct, i.e. whenever $\forall \omega, \lambda : f(\omega, \lambda) > 0 \implies p_\Omega(\omega, \lambda) > 0$, so it is applicable even to optically perfect wavelength dependent phenomena. However, in this case it does not provide any benefit over estimator (4). The comparison between the new estimators (5) and (7) and the previous single sample estimator (4) is presented in Figure 3. The glass sphere has linearly varying refraction from 1.35 for 360nm to 1.2 for 830nm and uses Phong based scattering with $n = 1000$. Images are created using only two 16-sample clusters, to show error more clearly.

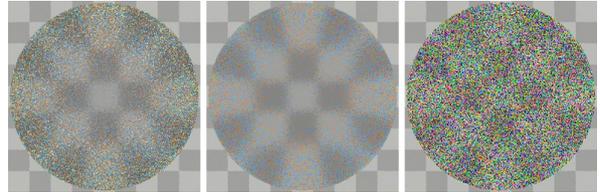


Figure 3: Comparison between new initial estimator (left), new improved estimator (middle) and previous method (right). The new initial estimator exhibits more variance due to lack of proper importance sampling. The color noise from single sample approach makes the rightmost image barely legible.

5.4 Generation of Clusters

In order to generate clusters efficiently, two issues have to be solved, namely: how many samples should a single cluster contain, and how to generate them. The number of spectral samples in a cluster is an important decision for achieving best possible performance. Unfortunately, optimal number of such samples is highly scene dependent. The more variation emission and reflectance spectra have, the more spectral samples a single cluster should contain. Assuming that a scene contains rather smoothly varying spectra (this assumption



Figure 4: Selection of optimal number of spectral samples for a single cluster: 4 samples (left), 8 samples (middle), 12 samples (right). All images were rendered in 640x480, with 200k image samples (i.e. spectral clusters).

typically is satisfied), it is possible to balance excessive color noise and computational overhead. After a few tests we have found that eight spectral samples are optimal¹. Four samples cause significant noise and twelve give barely visible improvement (see Figure 4). Rendering time differences between these images have been less than 1%, which confirms the efficiency of a cluster approach.

The efficient generation of spectral samples proves to be more difficult. Spectra should be importance sampled, but there are at least three factors, which should affect choice of p_Λ , namely: sensor (camera, human eye, etc.) sensitivity, light source spectral distribution and reflectance properties of materials. However, often only sensor is taken into account, and it is assumed that its sensitivity is well described by CIE y weighting function. Unfortunately, despite producing good quality grayscale images, importance sampling wavelength space with respect to the y function causes excessive color noise, and, contrary to common knowledge, is suboptimal. Ideally, a sampling probability should take into account all three x , y , and z components. After some experiments, we found that following probability gives good results:

$$p_\Lambda(\lambda) = N^{-1} f_\Lambda(\lambda), f_\Lambda(\lambda) = \frac{1}{\cosh^2(A(\lambda - B))}, \quad (8)$$

where $A = 0.0072nm^{-1}$ and $B = 538.0nm$ are empirically evaluated constants and $N = \int_{\lambda_{min}}^{\lambda_{max}} f_\Lambda(\lambda) d\lambda$ is a normalization factor. Results of this improved technique are presented in Figure 5.

Moreover, since spectra are typically smooth, sampling them with quasi-Monte Carlo (QMC) low discrepancy sequences instead of random numbers improves results. However, care must be taken when QMC sampling is applied to cluster based spectra. When a wavelength dependent effect is to be simulated, a single sample from the cluster has to be chosen. This choice is tricky due to peculiarities of QMC sampling. In case of true random numbers, selection of first sam-

ple from a cluster always works correctly. On the other hand, it is a serious error to select an every n th sample from a low discrepancy sequence. In the latter case, we assign a separate (pseudo)random sequence for a such selection of a spectral sample, in addition to sequence used for randomizing cluster samples. Results of QMC sampling are presented in Figure 5.

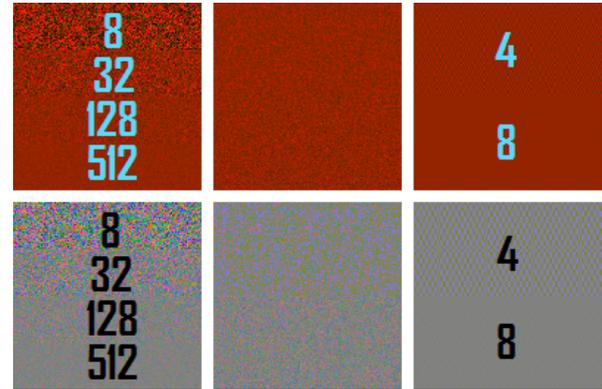


Figure 5: Various methods of sampling spectra. Top row: 2000K blackbody radiation. Bottom row: D65 spectrum. Left column: spectra sampled using random numbers and our importance sampling, with various numbers of samples. Middle column: comparison of luminance based importance sampling (top halves) with our p_Λ (bottom halves) using 128 spectral samples. Right column: spectra sampled using Sobol low discrepancy sequence and our p_Λ , using 4 and 8 spectral samples.

5.5 Results and Discussion

Some more comparison between single spectral sample approach and improved technique is presented in Figure 6. Images in top row use previous settings (refraction coefficient from 1.35 for 360nm to 1.2 for 830nm and glossiness coefficient $n = 1000$). Next, images in bottom row use much sharper settings (refraction coefficient from 1.5 for 360nm to 1.2 for 830nm and glossiness coefficient $n = 4000$). Images from first and second column are rendered to have approximately the same quality, and images from second and third column are rendered with the same number of samples (i.e.

¹ Due to Intel SSE instruction set optimization, our implementation requires the number of samples to be divisible by four.

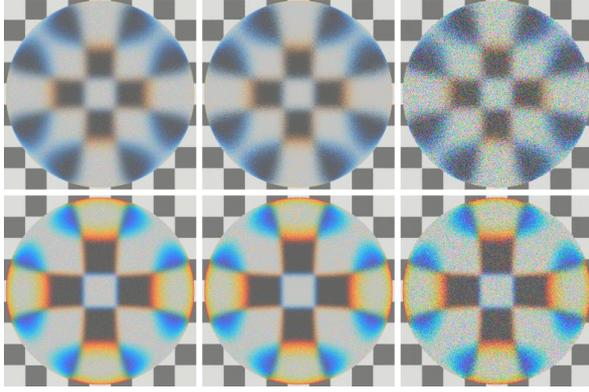


Figure 6: Imperfect refraction with dispersion. Top left image uses previous approach with a massive number of 900 samples per pixel. Top middle image uses new technique with just 50 samples per pixel, yet it has similar quality. Top right image again uses previous approach, but with 50 samples per pixel. However, gains from using the new technique are less spectacular when glossiness or dispersion is increased. Bottom row images use 900, 100, and 100 samples, respectively.

Settings	C	MIS	SSS
$n = 1000$ $\eta = [1.35, 1.20]$	1	$1.26 \cdot 10^{-1}$	$2.47 \cdot 10^{-1}$
	4	$6.67 \cdot 10^{-2}$	$2.02 \cdot 10^{-1}$
	16	$2.63 \cdot 10^{-2}$	$1.34 \cdot 10^{-1}$
	64	$1.22 \cdot 10^{-2}$	$7.56 \cdot 10^{-2}$
	256	$5.32 \cdot 10^{-3}$	$3.83 \cdot 10^{-2}$
$n = 4000$ $\eta = [1.50, 1.20]$	1	$2.07 \cdot 10^{-1}$	$2.46 \cdot 10^{-1}$
	4	$1.33 \cdot 10^{-1}$	$1.96 \cdot 10^{-1}$
	16	$7.39 \cdot 10^{-2}$	$1.29 \cdot 10^{-1}$
	64	$3.84 \cdot 10^{-2}$	$7.37 \cdot 10^{-2}$
	256	$1.74 \cdot 10^{-2}$	$3.72 \cdot 10^{-2}$

Table 1: Comparison of error of our method (MIS) and a single spectral sample approach (SSS), for C 8-sample spectral clusters per pixel. The error is evaluated as a difference between the tested image and the reference image, averaged over all pixels and color components. The pixel values are normalized to $[0, 1]$ range.

traced rays). The average numerical error for various numbers of rays for scene from Figure 6 is summarized in Table 1.

Analysis of two limit cases could give more insight into how this new technique works, and when it is most effective. The analysis is based on the assumption that $f(\omega, \lambda) \propto p_{\Omega}(\omega, \lambda)$ is roughly satisfied. Otherwise, the multiple importance cannot help much in reducing variance. First, when wavelength dependence is negligible, all the scattering probabilities become more and more independent on λ : $p_{\Omega}(\omega_i^s, \lambda_i^j) \approx p_{\Omega}(\omega_i^s)$. The weight W_i^s then becomes:

$$W_i^s = \frac{p_{\Omega}(\omega_i^s, \lambda_i^s)}{\sum_{j=1}^C p_{\Omega}(\omega_i^s, \lambda_i^j)} \approx \frac{p_{\Omega}(\omega_i^s)}{\sum_{j=1}^C p_{\Omega}(\omega_i^s)} \rightarrow \frac{1}{C}, \quad (9)$$

and the estimator:

$$\begin{aligned} I &\approx \frac{1}{N} \sum_{i=1}^N \frac{W_i}{p_{\Omega}(\omega_i^s, \lambda_i^s)} \cdot \\ &\quad \cdot \sum_{j=1}^C f(\omega_i^s, \lambda_i^j) \frac{w(\lambda_i^j)}{p_{\Lambda}(\lambda_i^j)} L(\omega_i^s, \lambda_i^j) \rightarrow \\ &\rightarrow \frac{1}{NC} \sum_{i=1}^N \sum_{j=1}^C \frac{f(\omega_i^s, \lambda_i^j)}{p_{\Omega}(\omega_i^s)} \frac{w(\lambda_i^j)}{p_{\Lambda}(\lambda_i^j)} L(\omega_i^s, \lambda_i^j) \end{aligned} \quad (10)$$

which is an estimator of a simple, wavelength independent, scattering. On the other hand, when scattering becomes more and more glossy and wavelength dependence is significant, with probability close to one the f becomes close to zero for all directions except ω_i^s . The rare cases, when $f(\omega_i^j, \lambda_i^j)$ is large and $j \neq s$, have low weight W_i^s , and therefore cannot affect the estimator much. Moreover, all the probabilities but the selected one go to zero, and therefore W goes to one, which leads to estimator equal to:

$$\begin{aligned} I &\approx \frac{1}{N} \sum_{i=1}^N \frac{W_i^s}{p_{\Omega}(\omega_i^s, \lambda_i^s)} \cdot \\ &\quad \cdot \sum_{j=1}^C f(\omega_i^s, \lambda_i^j) \frac{w(\lambda_i^j)}{p_{\Lambda}(\lambda_i^j)} L(\omega_i^s, \lambda_i^j) \rightarrow \\ &\rightarrow \frac{1}{N} \sum_{i=1}^N \frac{f(\omega_i^s, \lambda_i^s)}{p_{\Omega}(\omega_i^s, \lambda_i^s)} \frac{w(\lambda_i^j)}{p_{\Lambda}(\lambda_i^j)} L(\omega_i^s, \lambda_i^j), \end{aligned} \quad (11)$$

which is equivalent to the one sample estimator. This behaviour of estimator (7) is presented in Figure 7.

The former approach to spectral rendering separates scattering into two cases: wavelength independent scattering, and costly simulation of wavelength dependent phenomena using single spectral sample estimator. On the other hand, our method does not depend on such classification. Due to automatically computed weights, it adjusts itself to these two limit cases, and to the broad spectrum of intermediate cases, when scattering is wavelength dependent, but imperfect. The computational cost of our method depends on strength of wavelength dependence and optical perfection of material. These factors cause the cost to increase, but it never exceeds the cost of single spectral sample estimator.

6 SAMPLING OF LIGHT TRANSPORT PATHS

In this section we describe an integration of our full spectral sampling with selected light transport algorithms – a case when there is more than one wavelength dependent scattering encountered on the same light path. The extension of single scattering approach for Path Tracing [10] and Bidirectional Path Tracing [19] is, however, obvious. The wavelength λ_i^s is selected once for a whole path, and reused at each scattering. The weight W_i^s is therefore computed for the

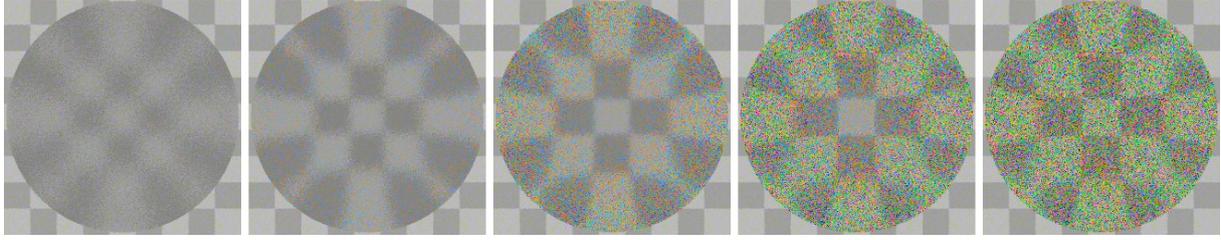


Figure 7: Analysis of behaviour of estimator (7) with increasing glossiness and wavelength dependence of scattering. Wavelength independent scattering (leftmost image). Optically perfect wavelength dependent scattering (rightmost image). Intermediate cases (middle). All the images are rendered with just four clusters.

whole path, using products of probabilities instead of probabilities of single scatterings. Assuming that the sampled path is build by recursively sampling f_s and tracing rays in sampled directions, the W_i^s is given by the following expression:

$$W_i^s = \frac{\prod_{k=1}^m p_{\Omega}(\omega_{ki}^s, \lambda_i^s)}{\sum_{j=1}^C \prod_{k=1}^m p_{\Omega}(\omega_{ki}^j, \lambda_i^j)}, \quad (12)$$

where k is the number of a scattering event and m is the length of the sampled path. Intuitively, a weight W_i^s is a ratio of probability of generating the whole path using selected wavelength λ_i^s to the sum of probabilities of generating such a path using each wavelength from a cluster. If a light transport algorithm generates a path in a different way, or does not use a concept of light transport paths, the weight W_i^s has to be computed in a different manner.

The notable case, where spectral sampling causes difficulties, is Jensen’s Photon Mapping, designed to work with RGB triplets [8]. There are two issues: first, there are no light transport paths, which connect light source and camera, and second, millions of individual photons have to be stored, causing excessive memory consumption if full spectrum is used to describe them. A recent work [11] addresses memory issues. Unfortunately, this algorithm converts photons’ spectra to RGB prior to storing them in a map, and converts RGB to spectra again when searching through photons.

Our approach, on the other hand, is designed to converge *always* to the true result with increased number of photons, and therefore significant compression of spectral data is unsuitable. We trace and store clusters of photons with different wavelengths, instead of describing them by RGB triplets. First, in order to explore wavelength space properly, each emitted photon cluster must have individually chosen wavelengths. The obvious place for optimization is that one emitted photon cluster typically corresponds to several stored photon clusters, and therefore cluster wavelengths are stored once for each emission. Moreover, for storing energy, one can experiment with a non-standard floating point format instead of IEEE single precision. Using 8-sample clusters requires 32B of data for individual stored photon, not to mention an additional 32B for



Figure 8: Full spectral rendering of a non-trivial scene. Dispersion is slightly exaggerated to render spectral sampling quality more prominent.

each emission, which is far more than 12B required by an RGB based implementation. If a compact float format with shared exponent is used, the latter can be compressed even to 4B, however, with potential loss of image quality. We have left this for further research.

When a photon is about to be stored, its energy is multiplied by weight given by Equation (12), which accounts for all encountered wavelength dependent scattering events. In the second pass, rendering of photon map is performed. Camera rays should be weighted similarly prior to photon map lookups. In the classic Photon Mapping, photons are searched in a sphere centered around the intersection point. The sphere radius should be chosen carefully: too small causes noise and too large – blurriness. We extend this approach to wavelength search as well. If a photon cluster is decided to be used in a flux estimate by a sphere test, additional tests are performed on individual photons (with associated wavelengths) using a spectral search distance in a wavelength space. Similarly as with the spatial radius, the spectral distance must be chosen carefully.

7 CONCLUSIONS

We have presented an improved approach to full spectral rendering. Full spectral algorithms realize a model

which is necessary to achieve physically plausible illumination in 3D scenes. The result image rendered with proposed spectral sampling, extended Walter et al. microfacet refraction [20], and Bidirectional Path Tracing is presented in Figure 8. The computational cost of a full spectral simulation in comparison with an RGB model is significant only for simple scenes containing few primitives. The computational complexity of ray tracing typically is logarithmic with respect to the number of primitives, and independent of a color representation. Therefore, when a scene becomes sufficiently complex, the overhead of a physically correct algorithm becomes negligibly small. On the other hand, the memory overhead depends on a particular algorithm. It is negligible for Path Tracing and Bidirectional Path Tracing, but is substantial for Photon Mapping.

A MONTE CARLO ESTIMATORS

This section briefly describes Importance and Multiple Importance Sampling methods. Consult [6] and [19], for more details. Let I be the integral to evaluate:

$$I = \int_{\Psi} f(x) d\mu(x). \quad (13)$$

The basic Monte Carlo estimator of this integral is:

$$\tilde{I} \approx F_N = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}, \quad (14)$$

where $\forall x: f(x) \neq 0 \ p(x) > 0$.

A variance of estimator (14) usually can be decreased if the p is made near proportional to f , or at least to a part of it. This technique is called Importance Sampling. Particularly, when $p \propto f$, the variance is zero. However, to obtain normalization constant, f must be integrated analytically. This is impossible, otherwise Monte Carlo integration would not be necessary.

Suppose that there are i potentially good probability densities p_i for sampling f . If Importance Sampling is used, the p_i used for sampling $f(x)$ has to be chosen at algorithm design time. This can have disastrous consequences, if the p_i poorly matches the actual $f(x)$ shape. In this case, Importance Sampling can actually *increase* variance over sampling with uniform probability. However, Multiple Importance Sampling [19] has been designed to improve the Importance Sampling when the appropriate p_i cannot be chosen at the design time. The algorithm samples from each of these p_i and calculates the final estimator as a weighted sum of these samples:

$$\tilde{I} = \sum_{i=1}^n \frac{1}{m} \sum_{j=1}^m w_i(X_{ij}) \frac{f(X_{ij})}{p_i(X_{ij})}, \quad \forall x \sum_{i=1}^n w_i(x) = 1. \quad (15)$$

The appropriate choice of weights w_i :

$$w_i(x) = \frac{p_i(x)}{\sum_{j=1}^n p_j(x)} \quad (16)$$

is crucial for obtaining low variance results.

REFERENCES

- [1] Kate Devlin, Alan Chalmers, Alexander Wilkie, and Werner Purgathofer. Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics 2002*, pages 101–123, September 2002.
- [2] Weiming Dong. Rendering Optical Effects Based on Spectra Representation in Complex Scenes. In *Computer Graphics International*, pages 719–726, 2006.
- [3] Roman Durikovic and R. Kimura. GPU Rendering of the Thin Film on Paints with Full Spectrum. In *Proceedings of the IEEE Conference on Information Visualization*, pages 751–756, 2006.
- [4] Glenn F. Evans and Michael D. McCool. Stratified wavelength clusters for efficient spectral monte carlo rendering. In *Graphics Interface*, pages 42–49, 1999.
- [5] Randima Fernando and Mark J. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley, Boston, MA, USA, 2003.
- [6] George S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, USA, 1999.
- [7] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. Wavelength dependent reflectance functions. In *SIGGRAPH 1994 Proceedings*, volume 28, pages 213–220, 1994.
- [8] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.
- [9] Garrett M. Johnson and Mark D. Fairchild. Full-spectral color calculations in realistic image synthesis. *IEEE Computer Graphics and Applications*, 19(4):47–53, 1999.
- [10] James T. Kajiya. The rendering equation. In *SIGGRAPH 1986 Proceedings*, pages 143–150, New York, NY, USA, 1986.
- [11] Gorm Lai and Niels Jorgen Christensen. A compression method for spectral photon map rendering. In *WSCG 2007 Proceedings*, pages 95–102, 2007.
- [12] Mark S. Peercy. Linear color representations for full spectral rendering. In *SIGGRAPH 1993 Proceedings*, volume 27, pages 191–198, 1993.
- [13] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.
- [14] Gilles Rougeron and Bernard Peroche. An adaptive representation of spectral data for reflectance computations. In *EGRW 1997 Proceedings*, pages 127–138, 1997.
- [15] Jos Stam. Diffraction shaders. In *SIGGRAPH 1999 Proceedings*, pages 101–110, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [16] Maureen Stone. *A Field Guide to Digital Color*. AK Peters, Natick, MA, USA, 2003.
- [17] Yinlong Sun, David F. Fracchia, Mark S. Drew, and Thomas W. Calvert. Rendering iridescent colors of optical disks. In *EGRW 2000 Proceedings*, pages 341–352, 2000.
- [18] Yinlong Sun, David F. Fracchia, Mark S. Drew, and Thomas W. Calvert. A spectrally based framework for realistic image synthesis. *The Visual Computer*, 17(7):429–444, 2001.
- [19] Eric Veach and L. J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH 1995 Proceedings*, pages 419–428, 1995.
- [20] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet Models for Refraction through Rough Surfaces. In *Eurographics Symposium on Rendering*, pages 195–206, Grenoble, France, 2007.
- [21] Gregory J. Ward and Elena Eydelberg-Vileshin. Picture Perfect RGB Rendering Using Spectral Prefiltering and Sharp Color Primaries. In *EGRW 2002 Proceedings*, pages 117–124, 2002.
- [22] A. Wilkie, R. Tobler, and W. Purgathofer. Raytracing of Dispersion Effects in Transparent Materials. In *WSCG 2000 Conference Proceedings*, 2000.

Computation and Visualization of Fabrication Artifacts

Muhammad Muddassir Malik
Vienna University of Technology,
Austria
mmm@cg.tuwien.ac.at

Christoph Heinzl
Upper Austria University of Applied
Sciences, Austria
christoph.heinzl@fh-wels.at

M. Eduard Gröller
Vienna University of Technology,
Austria
groeller@cg.tuwien.ac.at

ABSTRACT

This paper proposes a novel technique to measure fabrication artifacts through direct comparison of a reference surface model with the corresponding industrial CT volume. Our technique uses the information from the surface model to locate corresponding points in the CT dataset. We then compute various comparison metrics to measure differences (fabrication artifacts) between the two datasets. The differences are presented to the user both visually as well as quantitatively. Our comparison techniques are divided into two groups, namely geometry-driven comparison techniques and visual-driven comparison techniques. The geometry-driven techniques provide an overview, while the visual-driven techniques can be used for a localized examination.

Keywords: Difference measurement; Surface model; Volume rendering.

1 INTRODUCTION

Comparison of two almost identical datasets is very important for the continuously rising demands of quality control in industrial engineering. Recently much work has been done in the area of mesh comparison. A high number of vertices and edges are hard to process in real time due to the limited processing power available in hardware. This initiated research to simplify mesh datasets in such a way that the rendering speed is increased while the mesh distortion is limited. Distortions introduced through mesh simplification led to research on mesh comparison.

In the manufacturing industry, it is necessary to produce industrial components as close as possible to the computer aided design model (CAD) of the part. Engineers use CAD tools like AutoCAD, Pro Engineering etc. for designing. The CAD model is considered to be the ground truth during the manufacturing process. To verify the accuracy of the production process, manufactured components are scanned with an industrial computed tomography (CT) machine. The volumetric dataset obtained from the CT scan is then compared to the CAD model of the part (called surface model henceforth). The comparison between the two datasets is supposed to clearly identify erroneous regions.

The comparison process uses various methods to measure differences between the two datasets. The differences present between the surface model and the volume data are the result of fabrication, measurement, and surface reconstruction artifacts. We are primarily

interested in detecting the fabrication artifacts as these are introduced in an industrial part during the production phase. The goal of the comparison process is to minimize all post-production artifacts so that the differences measured between the datasets mainly correspond to fabrication artifacts.

Datasets of industrial components, unlike medical datasets, mostly consist of materials with distinctive density values. There is a high signal to noise ratio and the interfaces in the volume data are easy to detect. For this reason the most common method for first part inspection is to generate an iso-surface mesh from the CT scan and to compare it with the surface model. In various cases this is not the ideal approach: First, the generation of a mesh from the CT dataset requires a surface extraction algorithm. Industrial components have sharp edges and corners and therefore a lot of surface reconstruction artifacts are introduced [6]. Second, mesh generation for a given iso-value is not interactively possible during the comparison process. Therefore, the need to do a comparison with a higher or lower resolution mesh will lead to a delay in the examination process. Third, a CT dataset goes beyond a surface model and has information about the interior of the mechanical part as well. Losing this information limits the examination possibilities of the CT dataset.

Figure 1 shows a CAD model in (a), direct volume rendering (DVR) of the industrial CT scan in (b) and an iso-surface mesh extracted from the CT scan in (c). In figure 1(c), all the internal information of the volumetric dataset is lost. Areas marked with black rectangles in figure 1(b) and 1(c) are shown as zoom-ins. We observe surface reconstruction artifacts in figure 1(c).

In this paper we present a novel approach to perform a comparison directly between the surface model (which is the ground truth) and the volumetric dataset obtained from the industrial CT scan. We calculate the difference between the surface model

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright UNION Agency – Science Press,
Plzen, Czech Republic

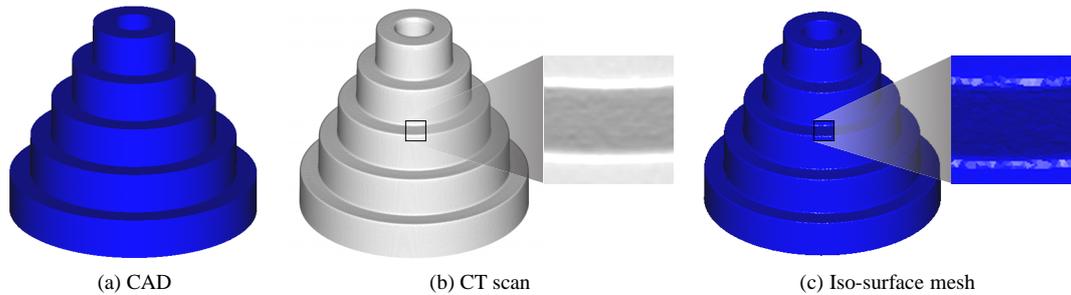


Figure 1: (a) CAD model of test-part-1 (surface model: 200,000 triangles, volumetric dataset: 561x559x436 voxels). (b) Direct volume rendering of the scan of test-part-1. (c) Iso-surface mesh extracted from the volumetric dataset in (b).

and an interface of the volume data and also compare the relative surface smoothness. We ensure that the differences we measure represent fabrication artifacts (section 4). The uncertainty of the measurement process is also evaluated and presented to the user.

Color coding, glyphs, ray profiles, and 3D box plots are provided for visualization and the results are also displayed quantitatively. The proposed method is implemented on the Graphics Processing Unit (GPU). It provides interactive comparison and visualization. We successfully avoid reconstruction artifacts by comparing the surface model directly with the volume data. Delays in the examination process are also avoided by embedding the complete comparison and visualization pipeline in a single system.

2 PREVIOUS WORK

Large numbers of triangles are inefficient to render and also hard to stream over a network. Subsequently, algorithms are proposed to simplify meshes [2]. Mesh simplification distorts the original shape, especially on sharp edges and therefore techniques are proposed to measure the differences between the two meshes.

Many public domain mesh comparison tools have been released in recent years [3, 9]. Metro [3] scan converts one of the surfaces into a set of points and then measures the Hausdorff distance between each point and the other surface. Aspert et al. [1] propose to use an approximation of the Hausdorff distance for measuring differences which is computationally and memory wise efficient. Pichon et al. [7] propose to use the gradient of the Laplacian equation to measure distances between the surfaces.

Weigle and Taylor [11] investigate visualization methods for distance and local shape comparison. Their study shows that glyphs are better in conveying deviation information between surfaces than color coding alone. They use intersecting surfaces with known alignment for their study.

There has been some recent work on the comparison between a surface model and an industrial CT dataset. These methods however introduce a pre-processing step to the comparison process, where an iso-surface mesh is

generated from the CT dataset. Heinzl et al. [5] propose a method for generating a feature preserving mesh from a CT dataset. They use filtering to suppress noise and a watershed segmentation to create a binary dataset. In the final step a surface model is created using elastic surface nets. The creation of a surface model is a time consuming and an error-prone process.

Geomagic Qualify [8] is a well-known software product, used for quality control in industrial engineering. A surface model and an iso-surface mesh of the volumetric dataset are inputs to this tool and it performs distance analysis between the two datasets. Methods for extracting an iso-surface mesh from a volumetric dataset [4, 5, 6, 10] have to be used in a pre-processing step for performing comparison using Geomagic Qualify. Geomagic Qualify works independently from the surface extraction process and therefore does not take into account surface reconstruction artifacts during the comparison process. However such errors are introduced in the pre-processing step.

3 COMPUTATION & VISUALIZATION

Our comparison system is divided into geometry-driven and visual-driven analysis techniques. Geometry-driven techniques provide an overall visualization of the differences between the surface model and the volumetric dataset. Visual-driven techniques are used on top of the geometry-driven comparison techniques for a user guided analysis and for obtaining precise quantitative information.

An overview of the system is shown in figure 2. The Iterated Closest Point (ICP) algorithm performs rigid registration and produces a transformation matrix as output. The output matrix transforms the surface model (moving dataset) through translation, and rotation to closely orient it to the CT dataset (fixed dataset). Registration is not the major scope of our work. We performed it with high accuracy (see section 4.1) using a well known algorithm in a semi-automatic way. Fully automatic registration techniques have not been investigated but might be applied.

Both types of comparison, i.e., geometry-driven and visual-driven comparison techniques (figure 2), query

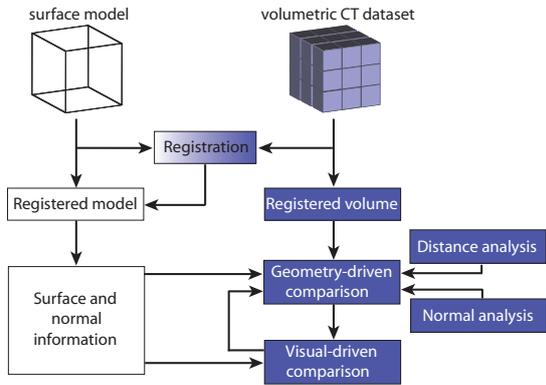


Figure 2: System overview. Geometry-driven comparison techniques color code the datasets and render glyphs. Visual-driven comparison techniques provide localized information about differences.

the registered surface model for the necessary information but work completely independent from each other. The results of the query, i.e., meta data from the CT dataset and the chosen visualization technique, are used to compute quantitative data and to produce images.

Geometry-driven comparison techniques consist of a distance analysis and a normal analysis. The distance analysis calculates the differences between the surface model and an interface in the volumetric dataset as Euclidean distances. It also measures the uncertainty of the measurement process. The normal analysis precisely locates differences in curvature and compares the surface smoothness of the two datasets.

We provide a ray-profile analysis and a magic lens as building blocks of the visual-driven comparison. The ray-profile analysis visually presents the data and differences at a user specified location and also displays the information quantitatively. The magic lens extracts the differences between datasets at a user specified neighborhood and displays them using glyphs, i.e., box plots.

3.1 Geometry-Driven Comparison

Distance and normal analysis methods require the specification of a corresponding point in the CT dataset for each surface point on the surface model. Starting from a surface point we have to locate the corresponding point in the volumetric data. The search direction is approximately along the surface normal. In high curvature areas the search should be extended to nearby directions as well to ensure robustness.

Consider the blue rectangle and the gray object in figure 3(a) to be a surface model and a volumetric dataset respectively. Black spheres represent surface points. A pair of red and green lines originating from each surface point indicates the conical space in which we search for a corresponding point in the volume data. The space is larger for surface points in high curvature regions (see the surface point at the corner in figure 3(a)).

For each triangle of the surface model we evaluate the facet normal and the three vertex normals. The angle between the facet normal and each of the vertex normals is computed and the maximum of the three angles (called search-angle henceforth) is stored. The search-angle indicates the local curvature of the surface model. In areas of high curvature, a large search-angle will be calculated whereas the search-angle will approach zero in planar areas of the surface model.

In figure 3(b) we indicate the search-angle as a red arc between the facet normal (black arrow) and one of the vertex normals (green arrow) of the blue triangle. Using the search-angle we can construct a double cone with the opening angle set to twice the search-angle. The double cone is depicted in figure 3(b) with the apex placed on the surface of the triangle. We then extract the spatial locations and the normal vectors for a set of uniformly distributed surface points on the triangles of the surface model. At each surface point the apex of a double cone is placed and the cone axis is oriented along the surface normal. A triangle therefore bisects the double cone at its apex (figure 3(b)). We call the nappe of the double cone that lies on the front face of the triangle as outside nappe, while the nappe on the back face of the triangle is called inside nappe. The double cone defines a region in which we can search for an interface point in the volumetric dataset. An appropriate interface point found inside the double cone will be associated with the surface point of the triangle for further computations.

In order to search for an interface point in the volume data, we start from the surface point and traverse the volume data along several rays distributed inside the double cone. The rays originate from the surface

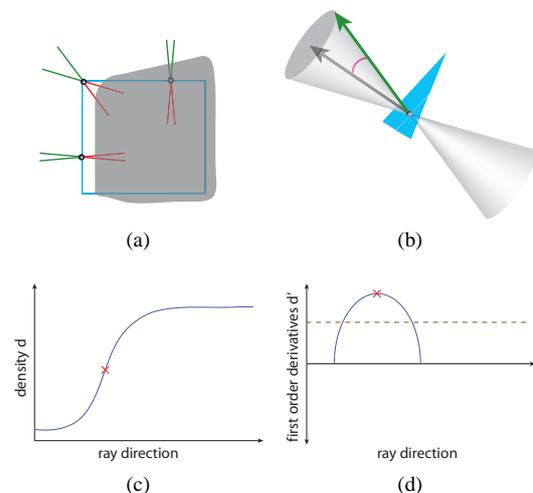


Figure 3: (a) Pairs of red and green lines depict the space in which we search for a corresponding point in the volume data (gray object) for each surface point (black sphere) on the surface model (blue rectangle). (b) Double cone representing the search space in 3D. A density profile and the first order derivative of a density profile are illustrated in (c) and (d) respectively.

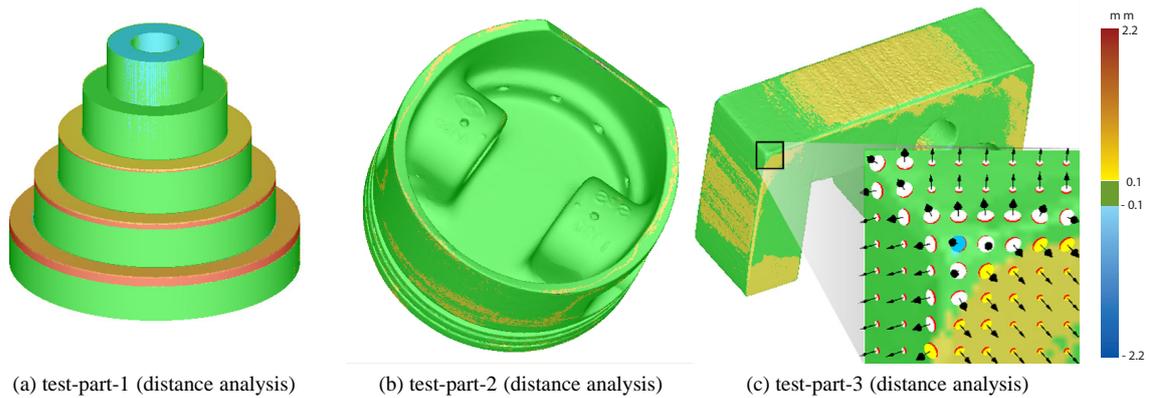


Figure 4: (a) test-part-1, (b) test-part-2 (surface model: 152,054 triangles, volumetric dataset: 408x351x355 voxels), and (c) test-part-3 (surface model: 25,880 triangles, volumetric dataset: 329x527x181 voxels) rendered using distance analysis. The image resolution is 512x512.

point and are directed towards the two bases of the double cone. The density profile of each ray is used to identify the interface point as the position with highest/lowest gradient magnitude (first order derivative is a maximum/minimum and the second order derivative is zero). The gradient magnitude at a spatial location must be greater than a user specified threshold for that location to be considered an interface point. Thresholding is necessary to filter out small changes in gradient magnitude which do not represent an interface. We also apply a median filter to the density values to reduce noise. Among all the considered rays the interface point with minimum distance to the surface point is stored for further processing. The rays are distributed in concentric circles inside the double cone. The density of the rays is kept almost constant by taking more rays on the outer circles compared to inner circles.

A density profile of a ray is illustrated in figure 3(c). The graph of the first order derivative of such a density profile is drawn as the blue curve in figure 3(d). The dashed brown line shows a threshold for the first order derivative. The first peak or valley with absolute derivative above the threshold is considered an interface point in the volumetric dataset. The interface point is indicated by a red cross in figures 3(c) and (d).

As we find an interface point in the volumetric dataset, we store its spatial location, the nappe (inside or outside) in which the interface point was found, and the gradient. The information extracted from the surface model and the CT dataset provides all the required parameters to evaluate the metrics for distance analysis and normal analysis.

Distance Analysis: The computationally intensive step of finding for each point on the surface model a corresponding interface point in the volume data has already been done. The distance analysis shows the difference between the datasets as Euclidean distances. We compute the differences between the spatial locations on the surface model and their corresponding interface points

in the CT dataset. We also have information about the nappe of the double cone in which the interface point was found. Using this information we color code the dataset for distance analysis.

Figures 4(a) and (b) show test-part-1 and test-part-2 respectively. The test-parts are rendered using the distance analysis with distances measured in millimeters. The color scale used for color coding is shown on the right of figure 4. The distance has positive sign if the interface point is found in the inside nappe of the double cone.

Figure 4(c) shows test-part-3 rendered using our distance analysis technique. We render distance glyphs on the zoom-in of the user specified area (black rectangle). The arrow of the distance glyph is aligned with the normal vector of the surface and the diameter of the disc is proportional to the diameter of the base of the double cone. The color of the disc indicates if the difference was found in the inside nappe (yellow), outside nappe (blue) or no difference was recorded (white).

So far we only consider the minimum distance between the surface model and the interface of the volumetric dataset for distance analysis. The technique does not take the interface shape into consideration. The results have uncertainty in high curvature regions which needs to be highlighted. For a double cone the difference between the minimum and maximum distance from the surface model to the volume data will be larger in high curvature regions compared to planar areas. Therefore the difference between the minimum and the maximum distance serves as the uncertainty value of the measurement process.

To determine uncertainty we look for the maximum distance from the surface point to the interface in the volume data. The search for the maximum distance is conducted in the neighborhood of the ray along which the minimum distance was found. The neighborhood for searching the maximum distance has a radius of one voxel. We choose this radius, as the search space should

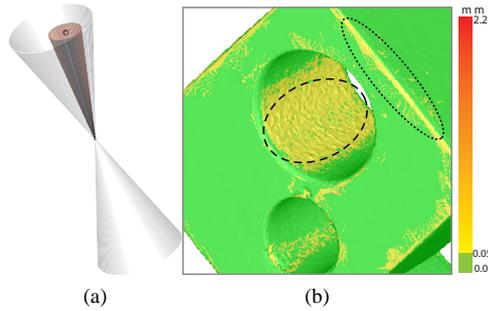


Figure 5: The maximum distance to the interface point is searched in the neighborhood (brown cone) of the ray (cone vertex to black sphere) along which the interface point was recorded. (b) Uncertainty rendering for a zoom-in of test-part-3. A dotted and a dashed oval highlight areas of high curvature and rough surface respectively.

be smaller than the smallest feature in the dataset. Any feature less than the size of a voxel is not detectable in the volumetric dataset anyway.

Figure 5(a) illustrates the uncertainty measurement process. Let us assume that the closest interface point was found along the ray which starts from the surface point (cone vertex) and extends towards the black sphere depicted on the base of the cone. In the neighborhood around that ray (brown cone), we search for an interface point with maximum distance to the surface point. The difference between the minimum and the maximum distance from the surface point to the interface in the volume data is considered the uncertainty of the measurement process.

The uncertainty in the case of test-part-3 is shown in figure 5(b). It becomes apparent that areas of high curvature or high surface roughness, which are highlighted using a dotted and a dashed oval respectively, have higher uncertainty.

Normal Analysis: Normal analysis is proposed as an efficient method to compare surface smoothness. Normal analysis compares the orientation of the normal vectors extracted from the surface model with the gradients obtained from the CT dataset. The angle between the normal vector and the gradient indicates the difference in the curvature of the surface model and the interface of the CT dataset. Normal analysis is easy and efficient to compute given that the surface points and the corresponding interface points are already evaluated.

The type of difference shown by normal analysis may pass undetected by distance analysis. Consider the black plane in figure 6(a) to be part of the surface model with the normal vector indicated by a black arrow. The interface of the volume data (blue plane) overlaps the surface model in the area marked with a red oval. The distance analysis will report no difference in such a case. However, there is a difference in the orientation of the two datasets as the normal vector and the gradient do not point in the same direction. Such differences can be emphasized using normal analysis.

Normal analysis will report a constant difference along the entire surface in this example.

Figure 6(b) shows test-part-1 rendered using normal analysis. Normal analysis detects differences at edges and rough surfaces. As the volumetric dataset is generated from an industrial process, it does not match the smoothness and exactness of the surface model, especially at the edges. The zoom-in in figure 6(b) shows that the top of test-part-1 is quite rough. The color scale can be changed dynamically by the user.

3.2 Visual-Driven Comparison

Visual-driven comparison techniques are grouped into ray profile analysis and magic lens displays. Ray profile analysis displays the differences between the datasets both as 2D plots and as quantitative numbers. A magic lens is used to zoom-in/out of the dataset and to view the differences graphically.

A ray profile display (figure 7) is generated by plotting the first derivative of the density values encountered along the ray in the volume data. The peaks and valleys in the graph show the interface points. The location of the surface point is marked on top of the graph. The horizontal difference between the interface point and the surface point in the plot shows the local difference. This provides precise information about the differences in the datasets.

Figure 7 shows two ray profiles generated for positions on test-part-3 marked with black crosses. The vertical red lines depict the points on the surface model. The blue graph shows the first derivatives of the density values encountered by the ray, along which the interface point in the volume data was found. The peaks in the blue graphs are the edges detected in the volume data.

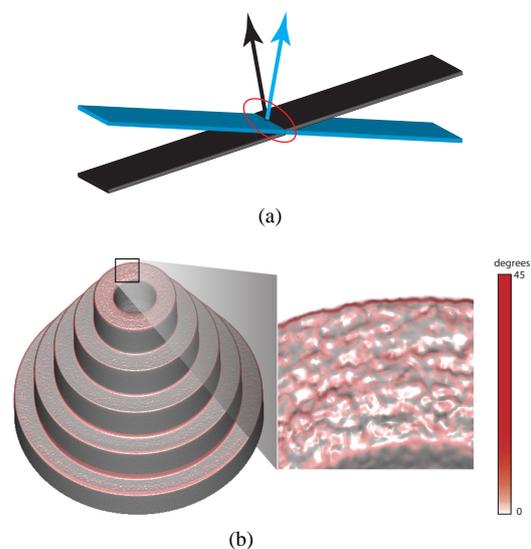


Figure 6: (a) Normal analysis emphasizes differences in orientation. (b) test-part-1 rendered using normal analysis. The zoom-in shows roughness at the top of test-part-1.

The horizontal distance between a peak and a red line indicates the local difference between the datasets.

In the ray profile on the left in figure 7, we observe that the surface point (red line) and the interface point (peak) overlap and thus there is very little difference between the two datasets. The ray profile on the right in figure 7 however shows a difference between the surface model and the volumetric dataset as there is a horizontal difference between the red line and the nearest peak. Our system reported a difference of 0.2 mm .

A ray profile shows the distance at one specific position. The next approach shows differences in a small local neighborhood. A magic lens can provide a precise graphical view of the differences by means of 3D or 2D box plots (figure 8). The box plots are rendered in a user specified area. Each box plot shows the minimum, the maximum, the mean, and the standard deviation of the differences between the two datasets at each local neighborhood. Additionally, 3D box plots are oriented along the normal vectors of the surface model. The diameter of a 3D box plot is directly proportional to the base of the double cone in which the interface point was searched. 3D box plots therefore encode distance values, uncertainty, and the dependent variables (normal vector and the base of the double cone) whereas 2D box plots only encode the distance values and the uncertainty of the measurement process (figure 8(a) and (b)).

Figure 8(c) shows 3D box plots over a user specified area (black rectangle) on test-part-3. 3D box plots similar to white planar discs indicate no difference in the minimum, maximum, and the mean distances recorded between the two datasets. The measurement is most certain in areas where “flat” 3D box plots are rendered.

3D box plots are less suited for a relative comparison as they are differently oriented along surface normals. 2D box plots make a relative comparison in a user specified area easier (figure 8(d)).

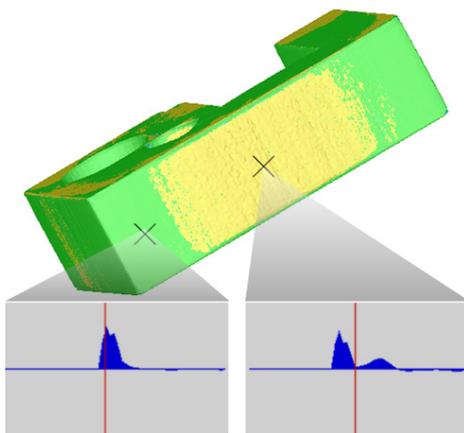


Figure 7: Two ray profiles are extracted for the locations marked with crosses on test-part-3. The horizontal differences between the red line (surface point) and the peaks in the graph (interface points) depict the dataset differences.

4 RESULTS

We implemented a prototype on a Pentium 4, 3.4 GHz CPU and an NVidia GeForce 8800 graphics board. We used C++ and OpenGL/GLSL as programming language. The system renders the volumetric data and the surface model side by side in a volume view and a surface view. We maintain a central queue for the events performed in all views. An operation initiated in one view also pushes an event into the central event queue and releases a signal. The other view pops the event from the queue and executes it. We implement first-come, first-serve scheduling for the central queue.

4.1 Artifacts and Errors

An industrial computed tomography includes fabrication artifacts and measurement errors. Fabrication artifacts are introduced during each step of the manufacturing process whereas measurement errors are caused by the CT machine. Additionally, two kinds of errors are generated by the software that is used to process the CT scan. Surface reconstruction artifacts are introduced while extracting an iso-surface mesh from the volumetric dataset. Registration errors are caused by the registration algorithm.

Quality assurance engineers are primarily interested in measuring the fabrication artifacts. To accurately compute the fabrication artifacts, errors caused by software should be minimized. We perform registration with high accuracy and unlike other contemporary techniques avoid surface reconstruction artifacts.

We evaluate the ICP registration algorithm by performing registration 20 times between test-part-3 and a feature preserving mesh [5] of test-part-3. We use a feature preserving mesh for testing purposes so that the fabrication artifacts and the measurement errors are minimized and we can monitor just the registration error. We measure the mean square error between the mesh and the test-part-3 (see figure 9) and record an average registration error of 0.0152 mm . The registration algorithm converged in 3.5 iterations on average.

Experiment number 18 produced a high error compared to the rest of the experiments. The ICP registration algorithm requires user interaction and the large error in experiment 18 is due to a bad specification of control points. The maximum fabrication and measurement artifact found in test-part-3 is 1.93 mm and the mean difference recorded is 0.27 mm . Thus the average registration error introduced by the ICP algorithm is considerably lower than the mean and the maximum fabrication artifacts in the dataset.

Reconstruction artifacts are introduced while extracting a mesh from a volumetric dataset. We use a synthetic dataset with known fabrication artifacts to evaluate our technique. Measurement and registration errors are not present in a synthetic dataset. This provides a good opportunity to analyze just the effect of

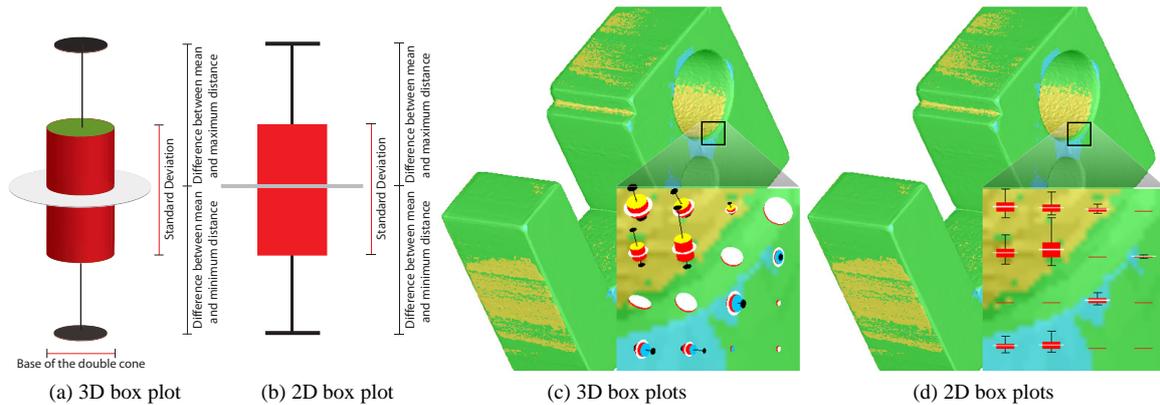


Figure 8: (a) 3D box plot. (b) 2D box plot. (c) and (d) show box plots over a user selected area. The user can interactively switch between 3D box plots and 2D box plots.

surface reconstruction artifacts. Figure 10(a) shows a surface model of a cube dataset and 10(b) shows a volume dataset with known fabrication artifacts. Fabrication artifacts are marked with an oval. The surface model consists of 12,288 triangles and the volume data has a resolution of $256 \times 256 \times 256$.

We generated a feature preserving mesh [5] of the volumetric dataset and compared it with the surface model using Geomagic Qualify (figure 10(c)). Surface reconstruction artifacts are visible in the differences shown by both zoom-ins. A difference is also reported at the vertical edge of the mesh (lower zoom-in) even though there should be no difference. The differences at vertical edges is purely caused by surface reconstruction artifacts and is not present in the dataset (see figure 10(b)). Figure 10(d) shows the comparison using our system. Our system correctly calculates no difference on vertical edges (lower zoom-in). The fabrication artifacts in the volumetric dataset are also reported correctly (upper zoom-in). The color coding is smooth and we do not observe any reconstruction artifacts.

The comparison of the maximum and average difference evaluated by Geomagic Qualify and our technique is given in table 1. Our method calculates the difference very close to the ground truth. Geomagic Qualify reports the maximum difference close to the ground truth but the average difference has a large error. Reconstructing a mesh from the volume data introduces artifacts distributed over the entire mesh. This is why the average error reported by Geomagic Qualify is very

small compared to the ground truth. As we avoid reconstruction artifacts, our calculations are more accurate.

Table 1: Maximum and average voxel difference reported by Geomagic Qualify and our system.

	Ground truth	Geomagic	Our technique
Maximum	8.485	7.95	8.91
Average	3.42	0.195	3.51

4.2 Performance and Evaluation

The earlier solutions proposed for comparison are divided into two major steps. For instance, Heinzl et al. [5] propose a robust surface detection pipeline for effective comparison. First, they extract a feature preserving mesh from the volume dataset. The mesh extraction part consists of a four step pipeline. In the first three steps, an anisotropic diffusion filter, a gradient filter and, a watershed segmentation are in turn applied to the volume dataset. In the final step constrained elastic nets are used. The mesh is then compared to the surface model using some existing tool like Geomagic. We combine the entire comparison and visualization process into a single, interactive system. Table 2 shows the runtime performance of our system, in comparison to the robust surface detection pipeline [5] and Geomagic.

The bottle neck in earlier methods has been the surface extraction process. Due to parameter tweaking the surface extraction took very long as opposed to the actual comparison process. Our method is more automated and requires much less user interaction.

Distance glyphs and the 3D box plots are additional visualization techniques for showing differences and

Table 2: Comparison of the performance of our system.

	Test-part-1	Test-part-3
Distance analysis (our method)	0.051 sec	0.033 sec
Robust surface detection pipeline	10.23 min	4.58 min
Distance analysis (Geomagic)	9.31 sec	8.51 sec

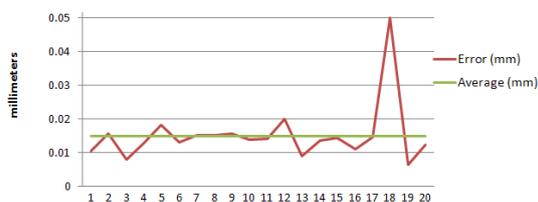


Figure 9: Mean square error produced by point-set to point-set registration on test-part-3 ($60 \times 100 \times 30$ mm).

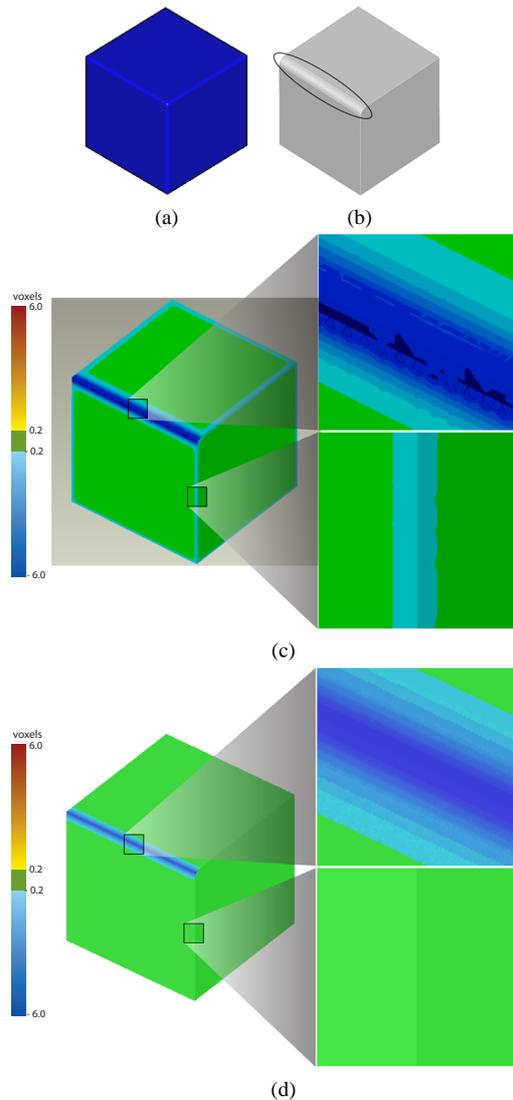


Figure 10: (a) Surface model. (b) Volume data with known fabrication artifacts. Artifacts are highlighted with an oval. (c) Comparison between the surface model and a feature preserving mesh of (b). (d) Direct comparison between a surface model and the volume data (our approach).

uncertainties. Two domain experts who have used various mesh comparison systems in their professional capacity tested the usefulness of our visualization techniques. They were both quite interested in using distance glyphs and 3D box plots to visualize differences as compared to color coding alone.

They acknowledged that they acquired more valuable information about the surface (surface normal), the measurement process (base of the double cone), and the differences using distance glyphs and 3D box plots. The idea of showing glyphs in a user specified area was one of the issues which the users are missing in conventional tools. The experts also appreciated the idea of showing the uncertainty of the measurement process

along with the distance analysis. The robustness of the registration algorithm was satisfactory for them.

5 CONCLUSION

We have presented techniques that compare a reference surface model directly to the industrial CT scan of specimens, especially in the preproduction phase of the industrial products. We avoid intermediate steps of data enhancement and surface extraction. Two sets of tools, namely geometry-driven and visual-driven techniques, provide comprehensive comparison opportunities.

ACKNOWLEDGMENTS

This work is partly funded by the SimCT project (FFG Bridge initiative). We are thankful to Torsen Möller from the Simon Fraser University for fruitful discussions and help in designing this method.

REFERENCES

- [1] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring errors between surfaces using the Hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume I, pages 705–708, 2002.
- [2] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1998.
- [3] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.
- [4] C. Heinzl, J. Kastner, and E. Gröller. Surface extraction from multi-material components for metrology using dual energy CT. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1520–1527, 2007.
- [5] C. Heinzl, R. Klingsberger, J. Kastner, and M. E. Gröller. Robust surface detection for variance comparison and dimensional measurement. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization*, pages 75–82, 2006.
- [6] L. P. Kobbelt and M. Botsch. Feature sensitive mesh processing. In *Proceedings of the 19th spring conference on Computer graphics*, pages 17–22, 2003.
- [7] E. Pichon, D. Nain, and M. Niethammer. A Laplace equation approach for shape comparison. In *Proceedings of SPIE Medical Imaging*, pages 373–382, 2006.
- [8] Raindrop. Raindrop: The magic of making it simple. March 2007. <http://www.geomagic.com>.
- [9] S. Silva, J. Madeira, and B. S. Santos. Polymeco - a polygonal mesh comparison tool. In *Proceedings of the 9th International Conference on Information Visualization (IV05)*, pages 842–847, 2005.
- [10] Volume-Graphics. *VGStudio Max 1.2 User Manual*. 2004. <http://www.volumegraphics.com>.
- [11] C. Weigle and R. M. Taylor. Visualizing intersecting surfaces with nested-surface techniques. *IEEE Visualization, 2005. (VIS'05)*, pages 503–510, 2005.

Dynamic Virtual Textures

Javier Taibo

University of A Coruña, Spain
jtaibo@udc.es

Antonio Seoane

University of A Coruña, Spain
aseoane@udc.es

Luis Hernández

University of A Coruña, Spain
lhernandez@udc.es

Abstract

The real-time rendering of arbitrarily large textures is a problem that has long been studied in terrain visualization. For years, different approaches have been published that have either expensive hardware requirements or other severe limitations in quality, performance or versatility. The biggest problem is usually a strong coupling between geometry and texture, both regarding database structure, as well as LOD management.

This paper presents a new approach to high resolution, real-time texturing of dynamic data that avoids the drawbacks of previous techniques and offers additional possibilities. The most important benefits are: out-of-core texture visualization from dynamic data, efficient per-fragment texture LOD computation, total independence from the geometry engine, high quality filtering and easiness of integration with user custom shaders and multitexturing. Because of its versatility and independence from geometry, the proposed technique can be easily and efficiently applied to any existing terrain geometry engine in a transparent way.

Keywords: Terrain rendering, virtual texture, clipmap, dynamic data, GIS, GPU.

1 INTRODUCTION

Terrain visualization typically involves both high resolution geometry and texture management. There are numerous works on these two areas, whether on one or other or on both. Focusing on the texture management, existing techniques require expensive hardware or establish a strong coupling between geometry and texture, both in databases and LOD management systems. We propose a different technique based on the programming capabilities of new GPU generations, that efficiently solves these limitations as well as provides new features for emerging applications to interactively visualize geographic information. The technique enables real-time rendering of high resolution textures composed of dynamic data that is periodically updated. A working subset of the whole dynamic database is cached in TRAM, in a clipmap-like structure [22], allowing arbitrarily large textures to be mapped over any geometry.

This virtual texturing engine can be applied to terrain visualization as well as to other applications that require a high detail 2D texture focusing on a center of interest. The textured dynamic data can be raster or vectorial in its source.

We start with a state of the art review, mentioning the weaknesses and limitations of previous related work, which have constituted our starting point for working on and improving. We then present a new approach,

describing its features, architecture and design throughout section three. Finally, we present some quantitative results from the current test implementation, in order to analyze the performance of the system, followed by the conclusions and future lines of research we are currently working on.

2 RELATED WORK

The use of wide area detailed textures for real-time terrain rendering was first studied by Michael Cosman in 1994 [7]. Two approaches were described: to use a mosaic of small textures of a size supported by the hardware or to use a unique virtual texture managed by specific graphics hardware.

The ideas given by Cosman were later retaken by Tanner et al. when they described the clipmap architecture [22]. This approach was similar as it was based on specific graphics hardware to support arbitrarily large virtual textures with a very limited amount of texture memory. The clipmap technique became quite popular and nowadays it is one of the most important references in the field of real-time terrain texturing.

Later approaches to the large textures problem were published, that avoided the strong hardware requirements of the previous ones. They were mainly based on the texture mosaic approach, and so they suffered most of the limitations described by Cosman. Some of the most important techniques were described by Hütner (MP-Grid) [13], Rabinovich [19], Cline [6], Döllner [9], Klein [14], Cignoni (BDAM/PBDAM) [4, 5], Brodersen [3] and DachsBacher [8].

The biggest drawback of these techniques is the strong coupling between terrain geometry and texture regarding database structure and run-time LOD selection. Geometry must be tessellated and tile boundaries must exactly match those of the tiles of the texture mosaic.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Lefebvre [15] et al. described a generic texture management system for arbitrary meshes but that does not comply with all the requirements we state, detailed in next section. They use a tile pool texture that does not keep continuity, what difficults high quality filtering.

The clipmap approach has also been adapted to geometry rendering, as in the works of Losasso [16], Holkner [12] and Asirvatham [1]. The use of texturing is proposed in some of these works, but texture LODs are fully coupled with geometry LODs and so they are not chosen based on the texture space projection of each fragment. This is an important drawback not only to LOD selection, but also to texture filtering. Some techniques have been proposed recently, as in Ephanov [10] and Seoane [21], based on the idea of clipmapping and trying to provide its advantages while not requiring expensive specific graphics hardware.

Seoane's technique provides a cached window of the virtual texture, for each texture level, as big as desired up to the texture size limit of the hardware. As it is a roaming window instead of a mosaic of textures, geometry tessellation is not so constrained as in other techniques. Geometry boundaries can be anywhere and as soon as the texture cache window contains a geometric primitive, it can be textured with this level of detail.

This decoupling between geometry and texture management is one of the main advantages of this technique. It can be used with different tessellations, even non-rectangular ones, i.e. TINs. The one important aspect is about geometry batch size, not shape. The problem of this technique is that, in order to achieve the highest texture LODs, geometry must be highly fragmented, which may result in performance loss.

Ephanov et al. propose two alternatives for their Virtual Textures: using the OpenGL fixed function pipeline or using pixel shaders. The first choice is similar to Seoane's approach in its relation between texture LODs and geometry tessellation and in that it does not require programmable hardware. But Ephanov does not use the toroidal update like Tanner and Seoane. Instead of the continuous update of a center of detail, the paging centers hop to positions matching the geometry. This means an undesired coupling between texture and geometry management systems: the texturing engine must know the geometry structure.

The second approach proposed by Ephanov extends the previous scheme by using the programmability and multi-texturing capabilities of new graphics hardware. This way, they can map a geometry primitive using several textures to achieve a higher texture detail for this geometry than was possible with the fixed pipeline. The drawback is that it implicates the use of several texture stages, making it difficult to combine several Virtual Textures or to use them in other advanced rendering techniques without expensive multiple render passes.

Also, the use of mipmaps for level tiles imposes a memory usage overhead for redundant data. Moreover, the use of double buffering duplicates the graphics memory assigned to cache the Virtual Texture.

Ben Garney [11] published a technique that emulates clipmaps with programmable hardware. The main limitation of this technique is that it requires one texture stage for each clipmap stack level used. This drastically limits the number of stack levels that are simultaneously usable and, specially, it difficults the combination of several textures with programmed effects.

Later works on the topic are Mittring's Advanced Virtual Textures [17] and Barret's [2] Sparse Virtual Textures. Both of them need at least an additional texture for cache addressing and do not keep the texture space continuity, what difficults high quality filtering. Moreover, none of these techniques address the management of dynamic textures.

3 VIRTUAL DYNAMIC TEXTURING SYSTEM

3.1 Objectives

The goals established in this work were focused on solving the main drawbacks and limitations described in previously mentioned works, supporting some features required for new applications. The technique complies with the following characteristics:

- It is based on a standard, such as OpenGL 2.0, avoiding the requirement of any vendor specific hardware.
- Texture LOD is computed and applied per fragment, resulting in an efficient cache usage and no drops of detail in geometry boundaries.
- Efficient memory usage. Unlike in some of the previous techniques, no redundant data is to be needed for hardware mipmaps or double buffering.
- Allows for several texture filtering types, including dynamically configurable anisotropic filtering.
- High performance, real-time update and texturing.
- Compatibility of virtual texture with any custom vertex or fragment shader.
- Use of only one hardware texture stage, allowing its combination with custom shaders and high performance multitexturing. This minimizes the number of texture binds (only one) without need to sort the geometry batches.
- Total independence from geometry. Possibility to apply an arbitrarily large texture to a single quad and reach its maximum LOD when the camera approaches the textured surface. This way, it allows the geometry engine to dynamically modify the meshes in size, shape or topology.

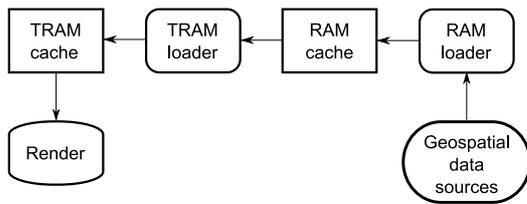


Figure 1: Overall architecture.

- As texture contents may be dynamic, cache is updated not only in space but also in time.

In summary, the proposed texturing system provides the features present in systems such as Cosman [7] and Tanner [22], also allowing for new capabilities such as dynamic update, multitexturing and its combination with custom-made shaders. Moreover, the only hardware requirement is a consumer graphics card.

This system can be applied to static or dynamic data and, in both cases, data sources can be raster as well as vectorial rendered to a procedural texture.

3.2 System architecture

The texturing system proposed is based on a two-level cache hierarchy, following the clipmap[22] structure, adapting it to currently available consumer hardware and improving it to support dynamic textures with contents that are continuously updated as a function of time.

The first cache level is the texture subset stored in TRAM, while second level is stored in RAM. Each cache level has an associated component whose task is to load or generate the contents stored in this cache. The overall architecture is illustrated in Figure 1.

Following the clipmap structure, the *pyramid* (complete levels) as well as the *stack* (incomplete levels) are cached on TRAM.

Because the use of a single texture stage is a requirement of the system, as previously mentioned in the objectives, only a single texture could be used to store the cache contents. We chose to use an OpenGL 3D texture to store all cache contents, because it offers several advantages to both quality and performance of the system. Storing each stack level on its own texture slice, continuity is kept so bilinear filtering in the level can be automatically done by hardware without noticeable overhead. Moreover, this continuity simplifies texture addressing. Both advantages assume the toroidal updating and addressing of the stack levels, as described by Tanner. The clipmap pyramid can be stored completely inside two slices in case of square virtual textures or one slice in rectangular ones. The storage of the clipmap structure in a 3D texture is illustrated in Figure 2.

This storage in a single texture is a big advantage over the mosaic of textures approach followed by many other

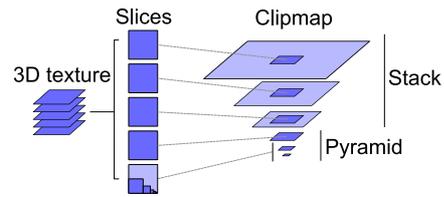


Figure 2: Clipmap structure on a 3D texture.

techniques, critical for the independence between texture management and geometry management, regarding structure as well as LOD management. It also frees precious texture units that can be used for other purposes in a custom shader that use the dynamic virtual texture proposed combined with other textures of any kind. This way, a user can access several virtual textures and use them for any purpose beyond just applying them as fragment color. Some examples of interesting applications for virtual textures beyond color are described by Ephanov [10].

The NVIDIA SDK white paper [18] proposes a structure using texture arrays, that can be an alternative to the use of 3D textures, and the algorithms described in our paper can be applied to this structure as well.

3.3 Updating

The cached region of the virtual texture that will be used in the render is computed from the center of detail supplied by the user. As the center of detail is moved over the virtual texture space, the contents of the caches must be replaced to keep the right information.

The update process implies several decisions affecting cache efficiency and so the final visual quality, though they must never affect the real-time performance of the system. To guarantee this premise, update process is divided in both a synchronous and an asynchronous update. The synchronous update uploads texture data to TRAM, competing for time with the render tasks, and so it has time restrictions to avoid frame drops. These time restrictions are dynamically changed in function of the render load.

The synchronous update, that feeds the first level cache, involves several important decisions. First of all, which levels to load, which regions inside each level and, in both cases, the loading order. Apart from that, there are also some other aspects we must take into consideration, such as temporal updating of dynamic data and asynchronous, predictive RAM cache updating to minimize second level cache misses.

Level loading order Regarding the order of loading, the classic bottom-up approach has the advantage of making the larger areas available first and then refining detail as soon as possible in progressively smaller areas around the center of detail. This is the strategy followed by the great majority of systems, including Cosman [7] and Tanner [22]. Although it is the best solution for static data, in a dynamic data texturing system such as

the one proposed, it is not. This scheduling scheme can lead to situations in which the camera is focused on a small detail of the texture that will never be available because coarser levels covering bigger areas are continuously reloaded even though they are not needed at all. This problem is more severe as the updating frequency of the data increases.

Frequently updated dynamic textures require a different approach for the update scheduling. We need to know in the update stage what texture LODs will be required (not in the fragment shader, as before), and that introduces an extra computation for the CPU. The scene must be examined to determine what range of texture LODs will be needed.

The computation of the exact LOD range needed in the render can be complex and costly, and so affect the system performance. We compute a conservative estimation of the LODs needed in function of the distance of the viewer to the textured geometry, the field of view of the camera and the screen resolution. The distance is estimated by casting some rays through the corners and the center of the viewport (and optionally some other samples, depending on the available computing power) to intersect the terrain. This approximation results quite adequate and needs reduced CPU time.

Inter-level loading order Concerning the update inside a texture level, there are two possible approaches for updating the TRAM texture cache of one texture level: updating variably sized regions of invalidated data according to center of detail displacement, as described by Tanner [22], or tessellate the virtual texture space in square tiles of fixed size, as described by Seoane [21].

Although the first approach can gain a bit of performance in some circumstances, while introducing more complexity to the update and specially to the load time control, the second approach guarantees that loaded texture blocks have an adequate size to maximize transfer rate, avoiding inefficient small block transfers and leaving this time available for other tasks in this frame.

For this reason we use the second approach, loading fixed-size square tiles of texture. Tile size is either specified in the configuration or automatically computed at startup time. The adjustment of this parameter is very important for tile loading, as the transfer rate from RAM to TRAM is strongly affected by it.

In case of procedural textures (tile render on-demand), the tile size is also critical because it affects the render efficiency. The usual behavior is that the bigger the block to upload or render, the higher the transfer rate or the rendering efficiency. However, for an adequate load time control, the load/render quantum must not be too big. In the special case of very dynamic textures, modified every frame, the best option is to render each whole level in one pass (tile size = clip size) as long as the render time of a level does not exceed the available update time.

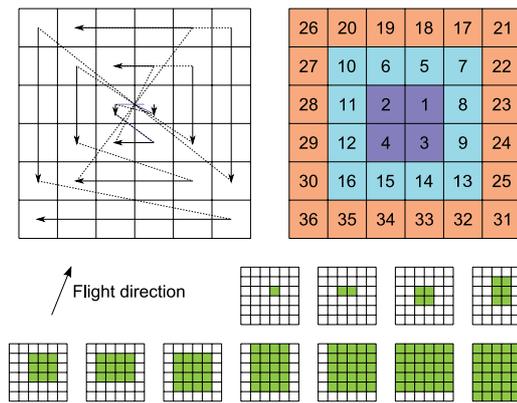


Figure 3: TRAM tiles loading order. Example for a given flight direction.

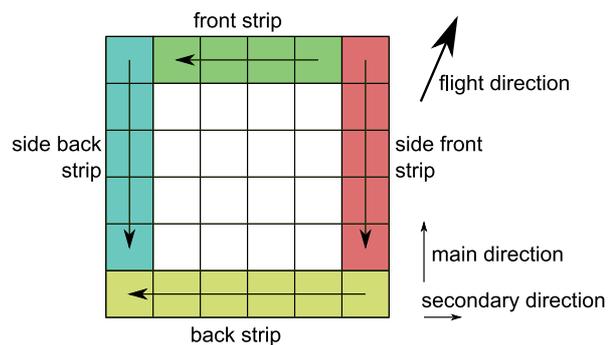


Figure 4: TRAM tiles loading order for a single quad around center of detail. Example for a given flight direction.

Following with the case of loaded tiles (instead of rendered ones), our tests showed that the optimum tile size for loading texture is dependent on the graphics hardware, but it is usually around 128 square texels. It is the smallest size before a severe drop of transfer rate.

Whatever the source of data, raster loaded or vectorial rendered to texture, the tiles loading order inside a texture level is defined with the following goals:

- To prioritize the loading of tiles close to the center of detail, that contain the information located in the region that will capture the user's main attention.
- To progressively load rectangular areas around the center of detail, i.e. to construct, as soon and as big as possible, areas directly usable in the render.
- To prioritize those tiles that, for the current flight direction, will be kept valid for more time, i.e. load first the tiles in the direction of movement.

Following these goals, the sorted list of tiles to load is generated in concentric squares around the center of detail (innermost to outermost), as showed in the example in figure 3. The order in each of these concentric squares is illustrated in figure 4.

Temporal update In case of dynamic textures, cache tiles must be updated, not only when the center of detail is moved, but also in function of time. The time of life of a tile can be determined in two ways. An asynchronous mechanism allows the client of the texturing engine to invalidate any single tile, level or the whole cache. Nevertheless, the usual temporal update is managed through a synchronous mechanism based on an expiry timestamp assigned to each tile. When this expiry time is reached, the data is no longer valid and must be requested again to the next cache level.

This expiry scheme has a problem that produces visible flickering on the render because, when some texture tiles reach their expiration time and so become obsolete, those tiles that could not be updated in this frame will cause a drop of quality in the render.

We solved this problem with a two-level expiration scheme, with hard and soft expiration times. Soft expiration refers to that there is new data available and its loading should be scheduled, but old data can be used meanwhile. Hard expiration implicates that this data is obsolete and while new data is not available, this tile or buffer cannot be used. With this technique, dynamic data will be updated smoothly, keeping the highest level of detail.

The expiration time is established by the data source as part of some metadata attached to each tile, so this information is propagated and taken into consideration in every cache level.

One of the other parameters in this metadata is the “absent” flag that avoids continuous request to the cache pipeline of elements that are not available in the data source. This allows to efficiently manage texture with incomplete levels or heterogeneous detail in the virtual texture. This concept was introduced by Tanner as “high resolution insets”.

The dynamic virtual texture can vary the available information through time, so the absent state also has a lifespan assigned and, once expired, this data can be requested again. Apart from supporting dynamic data that change, not only in contents, but also in spatial and detail availability, it allows the use of existing information in a texture database while it is still being generated.

Asynchronous predictive updating First level cache (TRAM) is updated by its loader from the information available in the second level cache (RAM), that structures the virtual texture space in square buffers with a size that is a multiple of TRAM tile size.

As the transfer rate from secondary storage or even network to RAM is much slower than from RAM to TRAM, this will be the bottleneck of the updating. This is why a prediction system, instead of an on-demand approach, must be designed to minimize cache misses and so avoid loss of detail or a slow refinement.

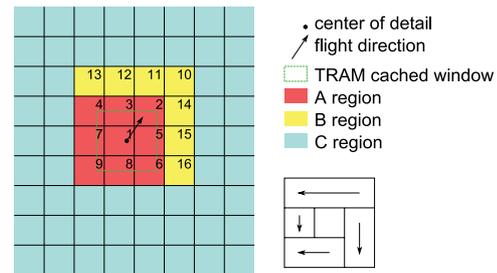


Figure 5: Loading order of RAM cache buffers.

Buffer loading priority is critical. In our experience, the best results have been achieved with the following algorithm (illustrated in Figure 5).

For each texture pyramid level (from coarser to finer), the set of buffers conforming the region that contains the current clipmap stack is computed. This set is called “region A”. The L-shaped set of buffers surrounding the region A, following the movement of the center of detail is called “region B”.

The buffers in each region are sorted and loaded by priority. Region A buffers are loaded innermost to outermost in concentric rings and each ring is loaded beginning with the buffers in the position towards which the center of detail is moving, in a similar way as described for TRAM tiles. Buffers in region B are loaded from the center to the edge of each arm, beginning with the arm in the main direction of movement. This way, buffers closest to the center of detail are loaded first to have valid data as soon as possible. Also, higher loading priority is assigned to those buffers that presumably have a longer life expectation, because they are in the direction of movement.

Region A contains the buffers immediately needed by TRAMCache, so they have the highest priority. But, there is an important design decision about how to do the loading along the clipmap levels.

Loading region A first for each clipmap level from coarser to finer and then beginning with region B in the same order gives the user the highest detail as soon as possible, but the camera movement can make the clipmap levels invalid very soon.

The other alternative is to load region A and then region B for each level in the clipmap, from coarser to finer. It takes a little more time to reach the highest levels of the clipmap, but once they are loaded they will be much more stable and suffer little or no drops of detail.

The first strategy is more adequate when the camera movement is very slow or when the application needs to show the information very quickly in some points, instead of showing a smooth visualization of a flight. As the behavior is application dependent, we decided to make it configurable and support both strategies. Moreover, this behaviour can be dynamically changed in run time, so RAM cache loading schedule can be set as a

function of the speed and/or kind of movement of the camera.

After completion of regions A and B, as described, the surrounding buffers are prefetched (region C), sorted by proximity and clipmap level, until the memory allocated to the cache has been filled.

Once all the available buffer storage space is filled, an LRU policy is applied to discard old buffers and load new ones. Only region C buffers are in the LRU queue; regions A and B buffers are always kept in RAM.

It is important to state that the RAM cache can be shared between several TRAM caches in applications with several views of the same scene with different centers of interest. Regions A and B are dependent on the client (TRAM cache) and they can be overlapped. To keep these buffers locked in cache, a reference count mechanism is used, so no buffer is sent back to the LRU until it is out of every client region A and B.

3.4 Rendering

The kernel of the texturing system is in the fragment shader code. A GLSL function (VTfetch) is provided to access the dynamic virtual texture inside any user shaders for whatever purpose.

The GLSL source code for this function is generated at run-time, during the initialization phase of the TRAM cache, depending on its configuration. This way, this performance critical code is highly optimized for the exact intended use.

The fragment shader uses several parameters, being the most important the texture coordinate set, received from the vertex shader.

Some of the parameters used are the texture sampler used by TRAM cache, the virtual texture size in texels, the number of levels in the clipmap pyramid and stack, the clip size, the tile size, the wrapping mode of the virtual texture, the LOD offset and the limit of anisotropic samples allowed.

The function, as well as these uniform parameters are prefixed with a texture stage identifier in order to allow several instances of virtual textures to coexist in different texture stages and so be combined in a user custom shader.

The main tasks of VTfetch are to compute the texture levels of detail needed for the fragment, check the availability of the needed texels in the cache and select the most adequate level for the fragment, address the real 3D texture to fetch the needed samples and combine them performing the desired filtering scheme.

The computation of texture LOD is done using the partial derivatives of texture coordinates in screen space. In case of isotropic filtering, it is done in a similar way to the one described in the OpenGL specification [20].

After calculating the LOD, it is clamped down to the immediate equal or lower level present in cache for the

texel. The fragment shader needs to know the status of the TRAM cache, i.e. the availability of each texel in the virtual texture full pyramid. This information is communicated to the fragment shader through a set of parameters containing the rectangular valid area for each level, computed by the update process described before.

How many texels and in what coordinates they are fetched depends on the filter used for the virtual texture. To avoid interpolation errors because of hardware bilinear filtering on the 3D texture, a border of half a texel around the valid area is considered unavailable, which means that immediate coarser level texels will be used instead. Supported filters include nearest neighbor, bilinear, trilinear and anisotropic.

As texture fetches are the most time consuming operation in the fragment shader, the number of anisotropic samples is limited by a uniform parameter that can be dynamically updated each frame, depending on the current system stress.

3.5 Stress management

A stress management system was developed with the objective of sustaining the frame rate of the visualization. It has, among others, the responsibility of distributing the available update time for each frame between the virtual textures. This way, the update time adapts to the time available until the next screen buffers swap and textures can be prioritized in function of their relevance or update status.

It can also dynamically adjust some of the quality parameters, such as the number of anisotropic samples, in function of the system stress to avoid frame drops. As the limit of anisotropic samples is a uniform parameter to the fragment shader, it can be changed every frame with no cost.

Monitoring the camera movement enables the rendering engine to increase quality (e.g. by improving the filtering) when the camera stops, in applications that are not frame rate critical and can allow frame drops in this situation, or to change the update scheduling strategy as described before.

3.6 Scalability issues

When virtual texture size is big (for example texturing the whole planet with submetric detail), several problems arise, affecting quality as well as performance and required resources:

- Memory requirements increase due to the amount of stack levels in the cache.
- Update time increases for the same reason.
- 32-bit numeric precision of graphics hardware is insufficient to accurately address the virtual space, producing visible deviations in the texture coordinates.

Filter	Render time
Nearest neighbour	1.6 ms
Bilinear	1.6 ms
Trilinear	1.6 ms
Anisotropic (1 sample)	1.7 ms
Anisotropic (2 samples)	3 ms
Anisotropic (4 samples)	4.4 ms
Anisotropic (8 samples)	5.4 ms
Anisotropic (16 samples)	7.3 ms
Anisotropic (32 samples)	8.5 ms
Anisotropic (64 samples)	9.9 ms
Anisotropic (128 samples)	11 ms

Table 1: Rendering performance.

In such situations, we solve these problems by not caching all the stack levels, but only a subset (floating stack) that is located depending on the proximity between the camera and the textured objects. This placement of this window of levels is computed in a similar way to the one mentioned for the level loading order in very dynamic textures. Apart from the reduction of levels stored, the texture area managed by the texturing system is reduced to an extension between the addressing limits of 32-bit arithmetic precision of the hardware.

Therefore, the application updates the location of this floating stack, as well as the center of detail position. The active texture area can be supplied by the geometry engine (that will suffer the same precision problems for the vertex coordinates and so it must work in local coordinate systems) or it can be automatically computed by the texturing engine following the position of the center of detail. The update of the floating stack as the cached window is moved, is performed in a circular way to minimize memory transfers and optimize performance.

4 RESULTS

The current implementation of the described technique has been integrated in our real-time terrain visualization system in order to test it in a real production environment. For debugging, testing and verification purposes, a standalone texture visualization tool was also developed, that provides detailed graphic information about the internal state of the texturing engine. All the tests presented in this section have been performed on an NVIDIA GeForce 8800 Ultra with driver version 1.4.0.90 for Linux 32-bit.

For the rendering performance and quality analysis, we used a 1048576×1048576 texels (20 levels) texture with its cache fully updated, so no texture loads could affect performance. The cached window for each level (clip size) was 2048×2048 texels. This virtual texture was mapped to a frame filling plane viewed from a shallow angle to force a highly anisotropic situation. Screen resolution was 1280×1024 pixels. In table 1 we can see performance measurements for the different supported filter types and, in the case of anisotropic filtering, different number of samples.

For the testing of update performance, a dynamic texture with high update ratio was used. Texture virtual size was 131072×65536 (17 levels), with a clip size of 1024×1024 and a TRAM cache tile size of 128×128 .

The data source was taken from NASA Blue Marble data set, showing the Earth appearance along a year. Texture update rate was set to one second, with a grace period of another second to reload the information before invalidating it.

This dynamic texture was applied to a model of the Earth globe and moved around, zooming in and out to examine different places as in a typical usage.

The update time assigned for each frame was 3 ms, data was accessed through a network, and cache contents were completely replaced every second. The highest detail was available for the most time, especially with still camera or slow movements. Drops of detail matched the fast movements of the camera (and so the center of detail) but they were barely noticeable because in these situations the user is far away and higher levels are not applied.

The size of the cached window or clip size is an important decision. With static data it is beneficial to have a higher size, because it allows to make fast moves, keeping the maximum detail, but with dynamic data it can be counterproductive, because update time is wasted with information that will expire before being used. In highly dynamic information, it is very critical to adjust the clip size to the minimum required for the screen resolution. We usually chose 1024 for high update ratios (near to a second) and 2048 for low update ratios or static data.

The scalability of the system was successfully tested with a static virtual texture of $2^{27} \times 2^{26}$ texels covering the whole planet, reaching a resolution of 0.25 m/texel in the area of highest detail.

5 CONCLUSION AND FUTURE WORK

We have developed a geospecific, dynamic, virtual texturing engine that fulfills the objectives proposed in section 3.1 and we are beginning to successfully test the system in real environments. We have focused on terrain texture or similar applications, where detail is located around a unique area, and not on other applications that need sparse detail textures. Planetary sized dynamic textures with submetric resolution are supported through the virtualization mechanisms described.

One of the most important achievements of the proposed system is to offer all the previously mentioned features while keeping full geometry independence. This allows homogeneous, high quality aerial image to be mapped over irregularly tessellated terrain, enabling us to use this virtual texturing engine in applications like the one shown in Fig. 6, where terrain geometry

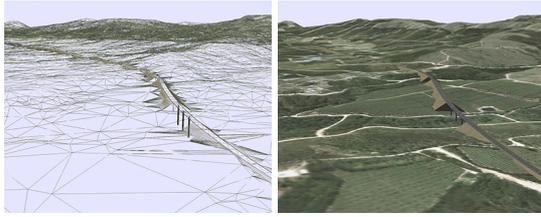


Figure 6: Texturing a TIN-based terrain with embedded 3D models.



Figure 7: Examples of use of the texturing engine.

cannot have a regular tessellation because it must be accurately adapted to a 3D model of a highway.

Multitexture capabilities have been tested blending raster virtual textures together and blending them with vectorial data rendered to another virtual texture, such as technical drawings or GIS layers. Figure 7 shows these examples, as well as the use of the texturing engine within a shader that desaturates some regions depending on their color.

Concerning dynamic update of textures, we have found that quantitative results have outperformed the needs of real applications managing dynamic geographic information, where usual update cycles do not fall below one minute.

We are exploring the benefits of the described technique in some fields of application, combining visualization of high resolution aerial image with dynamic raster and vectorial data over 3D terrain models. This includes projects in real-time traffic management, urban planning, infrastructure project analysis and fire extinction, among others.

REFERENCES

[1] Arul Prakash Asirvatham and Hugues Hoppe. Terrain rendering using gpu-based geometry clipmaps, 2005.

[2] Sean Barrett. Sparse virtual textures. <http://silverspaceship.com/src/svt/>, 2008.

[3] Anders Brodersen. Real-time visualization of large textured terrains. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 439–442, New York, NY, USA, 2005. ACM Press.

[4] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Bdam – batched dynamic adaptive meshes for high performance terrain visualization, 2003.

[5] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet-sized batched dynamic adaptive meshes (p-bdam). In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 20, Washington, DC, USA, 2003. IEEE Computer Society.

[6] David Cline and Parris K. Egbert. Interactive display of very large textures. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 343–350, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[7] Michael A. Cosman. Global terrain texture: Lowering the cost. In Eric G. Monroe, editor, *Proceedings of 1994 IMAGE VII Conference*, pages 53–64. The IMAGE Society, 1994.

[8] Carsten Dachsbacher. *Interactive Terrain Rendering: Towards Realism with Procedural Models and Graphics Hardware*. Universität Erlangen, 2006.

[9] Jürgen Döllner, Konstantin Baumman, and Klaus Hinrichs. Texturing techniques for terrain visualization. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 227–234, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[10] Anton Ephanov and Chris Coleman. Virtual texture: A large area raster resource for the gpu. In *The Interservice/Industry Training, Simulation and Education Conference (IITSEC)*. IITSEC, 2006.

[11] Ben Garney. *Game Programming Gems 7*, chapter Clipmapping on SM1.1 and Higher, pages 413–422. Charles River Media, 2008.

[12] Alex Holkner. Hardware based terrain clipmapping, 2004.

[13] Tobias Hüttner. High resolution textures. In *Visualization '98 - Late Breaking Hot Topics Papers*, pages 13–17, November 1998.

[14] Reinhard Klein and Andreas Schilling. Efficient multiresolution models for progressive terrain rendering. *it - Information Technology*, 44(6):314–321, 2002.

[15] Sylvain Lefebvre, Jerome Darbon, and Fabrice Neyret. Unified texture management for arbitrary meshes. Technical Report RR5210-, INRIA, may 2004.

[16] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 769–776, New York, NY, USA, 2004. ACM Press.

[17] Martin Mittring and Crytek GmbH. Advanced virtual texture topics. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 23–51, New York, NY, USA, 2008. ACM.

[18] NVIDIA. Clipmaps - white paper (wp-03017-001_v01), february 2007.

[19] Boris Rabinovich and Craig Gotsman. Visualization of large terrains in resource-limited computing environments. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 95–102, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[20] Mark Segal and Kurt Akeley. The opengl graphics system: A specification (versión 2.1). Technical report, 2006.

[21] Antonio Seoane, Javier Taibo, Luis Hernández, Rubén López, and Alberto Jaspe. Hardware independent clipmapping. In *WSCG '2007: The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007 - Full Papers Proceedings II*, pages 177–183. Eurographics Association, 2007.

[22] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: a virtual mipmap. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158, New York, NY, USA, 1998. ACM Press.

Algebraic 3D Reconstruction of Planetary Nebulae

Stephan Wenger
TU Braunschweig
wenger@cg.tu-bs.de

Juan Aja
Fernández
Universidad Nacional
Autónoma de México
juan.aja@gmail.com

Christophe
Morisset
Universidad Nacional
Autónoma de México
chris.morisset@gmail.com

Marcus Magnor
TU Braunschweig
magnor@cg.tu-bs.de

ABSTRACT

Distant astrophysical objects like planetary nebulae can normally only be observed from a single point of view. Assuming a cylindrically symmetric geometry, one can nevertheless create 3D models of those objects using tomographic methods. We solve the resulting algebraic equations efficiently on graphics hardware. Small deviations from axial symmetry are then corrected using heuristic methods, because the arising 3D models are, in general, no longer unambiguously defined. We visualize the models using real-time volume rendering. Models for actual planetary nebulae created by this approach match the observational data acquired from the earth's viewpoint, while also looking plausible from other viewpoints for which no experimental data is available.

Keywords: algebraic reconstruction, volumetric modelling, volume reconstruction, 3D modelling

1 INTRODUCTION

When stars not larger than a few sun masses die, they often eject part of their matter until only a small glowing nucleus is left in the center of a gaseous shell. These objects are called *planetary nebulae*. Their shape is often spherical or bipolar (i.e. cylindrically symmetric), but irregular shapes exist as well. Due to the radiation of the central star, the atoms in the shell get ionized and begin to emit light of characteristic wavelengths when the electrons recombine. Usually, there is not much absorption in the shell, so that only emissive effects have to be taken into account for reconstruction and visualization. A more comprehensive introduction to planetary nebulae can be found in [OF06].

In astrophysical research on planetary nebulae, being able to determine plausible models of their three-dimensional shape is an important precondition for a better understanding of the physical processes underlying their structure. For example, simulations of photoionization processes rely on a model of gas volume densities as input data and can in turn validate those models with respect to their physical realism. Interactive 3D visualizations of planetary nebulae can also be useful for scientific and educational purposes, such as digital planetariums.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

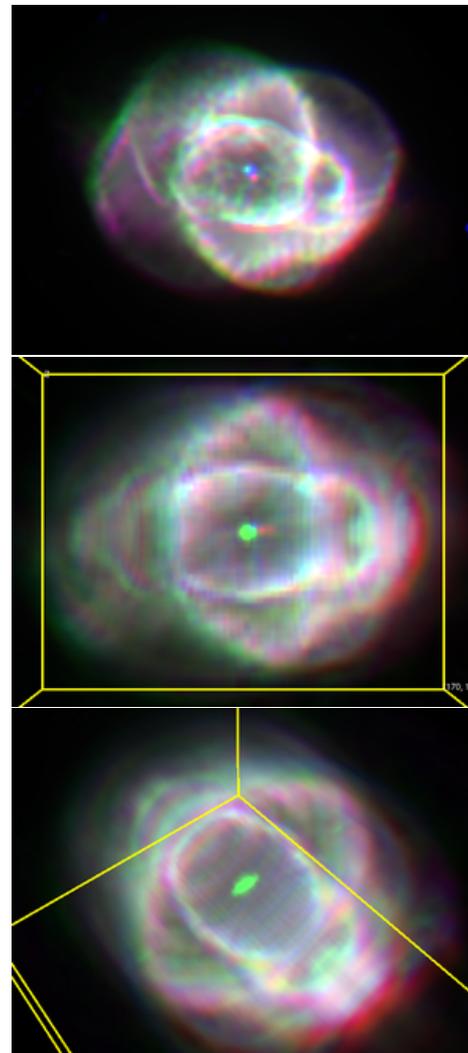


Figure 1: The Cat's Eye Nebula. Top: images taken using filters for 487nm, 502nm and 656nm that are assigned to the red, green and blue color channels, respectively. Middle and bottom: asymmetry-corrected reconstruction, view from earth and from outer space.

While some algorithms already exist to reconstruct the 3D shapes of planetary nebulae using a single input image, we present a fast, GPU-based approach that not only outperforms existing solutions in terms of computing time, but also includes asymmetric features without requiring further input data.

2 RELATED WORK

While planetary nebulae are a common research object in astrophysics and astronomy, their three-dimensional visualization has been purely artistic work for a long time. One notable exception is the rendering of the Orion nebula by Nadeau et al. [NGN⁺01], who created a scientifically accurate fly-through animation of that nebula, but the 3D model they use had to be worked out by hand by astronomers. A large amount of astronomical research work is done on the subject of classifying [Cur18, KK68] and explaining [KPF78, CP83] the three-dimensional structure of planetary nebulae and to simulate the physical processes inside planetary nebulae for the proposed 3D geometries in order to confirm these observational findings [MF89, AKR00, EMB⁺03]. However, the processes leading to the observed structures are still not well understood, and reliable determination of their 3D shape is an open problem.

Sabbadin et al. [Sab84, SCB⁺00] as well as Saurer [Sau97] take a semi-automatic physics-based approach for the reconstruction of the 3D geometry of planetary nebulae. They assume that the velocity of a certain region of gas around the nebula is strongly correlated to its distance from the central star. Calculating the Doppler shift of some well-known emission lines allows to get the velocity component towards the observer. Combining these, depth information can be reconstructed. However, the relation between velocity and distance from the central star is generally unknown, and exact Doppler shift measurement requires elaborate experimental setups, while our reconstruction approach relies on easily available photographic images only.

Methods for tomographic reconstruction of axisymmetric objects based on a single image have been proposed by Hanson [Han93], who applies this approach to man-made objects with known and theoretically perfect axisymmetry.

Magnor et al. [MKHD04, MKHD05] present a hardware accelerated reconstruction method for planetary nebulae that works with a single photograph as input. Their analysis-by-synthesis approach is based on the assumption of axial symmetry and *Constrained Inverse Volume Rendering* (CIVR). While they also propose corrections for small deviations from axial symmetry, these corrections are not realized in the provided examples. Furthermore, the reconstruction using their approach is computationally very expensive.

Lințu et al. have proposed a variant of the above algorithm that estimates absorption and scattering using an infrared image of the same object, allowing to reconstruct the dust density as well and thereby extending the range of reconstructible objects [LLM⁺07b, LLM⁺07a].

Another piece of work by Lințu et al. [LHM⁺07] describes a method to reconstruct the volume density distribution of dust in reflection nebulae using an analysis-by-synthesis approach. The algorithm does not rely on symmetry assumptions but exploits the special properties of light transport in an environment dominated by scattering and absorption to produce non-exact but plausible 3D volumes. However, these properties are not present in planetary nebulae which are usually dominated by emission, and the method can therefore not be applied.

3 ALGEBRAIC RECONSTRUCTION

A common approach for getting three-dimensional volume models from two-dimensional images is *tomographic reconstruction* [KS88]. This method is used, for example, in *computed tomography* (CT) to get volume densities out of multiple x-ray images of an object. While in the case of CT images the density of the object causes *absorption*, the contrary is the case for planetary nebulae, which emit light proportionally to the density of ionized gas. The intensity I_i of a certain pixel i in a discrete two-dimensional image of a planetary nebula is just the integral over all the emission densities along the incident light ray $\text{Ray}(i)$ of this pixel. Using a discrete volume model, this can be written as a sum over all volume elements v_j , where each summand is the length of the ray that lies within the volume element (denoted by $|\text{Ray}(i) \cap v_j|$), multiplied by its emission density ρ_j :

$$I_i = \sum_j |\text{Ray}(i) \cap v_j| \cdot \rho_j, \quad (1)$$

This system of linear equations, usually written as $A\mathbf{x} = \mathbf{b}$ with $x_j = \rho_j$, $b_i = I_i$ and $A_{ij} = |\text{Ray}(i) \cap v_j|$, can now be solved for the ρ_j . Under certain preconditions, this gives a unique solution for the volume emission densities.

For the solution to be uniquely defined, the rank of the matrix A must be at least equal to the number of volume elements. This is usually achieved by using images from multiple viewpoints. In practice, the system will almost always be overdetermined, as one would rather use more pixels than necessary to get more stable results. An approximate solution can then be computed using iterative algorithms that minimize the 2-norm $\|A\mathbf{x} - \mathbf{b}\|_2$ of the residual error.

For most astrophysical objects, getting images from multiple viewpoints is impossible due to the large distance to those objects. This means that if a regular grid

of volume elements (or *voxels*) is used, the linear system is not uniquely defined and would in fact not yield any information about the three-dimensional structure of the object.

This problem can be solved by making additional assumptions about the geometry of the object, e.g. an axial symmetry that is common in planetary nebulae. Such symmetries can easily be reflected in the choice of voxels by combining all regions of space that are assumed to have the same emission density into one voxel (Fig. 2). This can reduce the three-dimensional complexity of the voxels (x, y, z) to a two-dimensional one in the case of cylindrical voxels (r, z) so that the solution of the linear system is unique.

The appearance of axisymmetric models obviously depends strongly on the choice of the symmetry axis. Several possibilities exist for determining a plausible symmetry axis, most of which involve physical reasoning and further observational data. One automatic way to find a symmetry axis would be to determine its angle within the image plane by principal components analysis and then look for elliptical features in the image that are likely to be circles in the real object. The axis angle with respect to the image plane can then be calculated from this projection, although a natural ambivalence between backward and forward inclined axes remains. For our test cases, the symmetry axes were usually determined by hand.

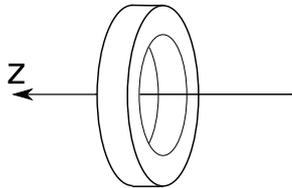


Figure 2: A voxel representing the axial symmetry of the geometry. Since all locations that lie in the same voxel share the same intensity, using voxels of this form guarantees axial symmetry of the result.

While symmetries are common in planetary nebulae, their symmetry is never perfect. To get more realistic results, the residual pixel intensities $\mathbf{b} - \mathbf{A}\mathbf{x}_{\text{approx}}$ for some approximate (and perfectly symmetric) solution $\mathbf{x}_{\text{approx}}$ can be distributed among the voxels that contribute to the intensity of the corresponding pixel. Because no depth information is available for these unmatched emissivity densities, the distribution among the voxels is not uniquely defined and can in fact only be chosen using heuristic methods.

4 A FAST RECONSTRUCTION ALGORITHM

4.1 Specifying the Linear System

In order to specify the system of linear equations (eq. 1), we need to calculate the matrix elements A_{ij} and the right-hand vector \mathbf{b} . While the b_i are already given by the intensities of the pixels, setting up the A_{ij} requires further calculations.

We recall that $A_{ij} = |\text{Ray}(i) \cap v_j|$ is the length of the ray through pixel i that lies within the volume element j . This means we have to calculate the intersection points of lines with voxels that have the form of a hollow cylinder (cf. Fig. 2). Since the viewing angle is usually very small due to the large distance to the object, we can assume that all the incident light rays are orthogonal to the image plane, so that they are defined as

$$\mathbf{r}(t) = \begin{pmatrix} p_x \\ p_y \\ 0 \end{pmatrix} + t \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \quad (2)$$

where p_x and p_y are the pixel x and y coordinates, respectively. This means we are using a three-dimensionally extended version of the image coordinate system for our calculations.

4.2 Solving the Linear System

The linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ must now be solved for the voxel intensities \mathbf{x} . Since the system is usually overdetermined, in general only an approximate solution minimizing the residual norm $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ is possible. This solution can be determined by iterative algorithms such as the *Conjugate Gradient Least Squares* (CGLS) method [Han96].

The fundamental idea of the Conjugate Gradient algorithm is that solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ with A symmetric and positive definite is equivalent to minimizing $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$. Starting from $\mathbf{x} = 0$, each iteration step k modifies the intermediate solution vector \mathbf{x}_k by descending in the direction of the gradient of $f(\mathbf{x})$, so that $\mathbf{x}_{k+1} = \mathbf{x}_k + \varepsilon_k \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$ (where ∇ is the gradient operator). For fast convergence, it is important that ε_k is computed such that \mathbf{x}_{k+1} ends up close to the one-dimensional minimum in direction $\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$, and that the directions of descent are *conjugate* to each other, that means that for all directions $\mathbf{d}_i = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}$, $\mathbf{d}_j = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_j}$: $\mathbf{d}_i^T \mathbf{A} \mathbf{d}_j = 0$.

In our case, however, the matrix A does not fulfill the above conditions. But since for any matrix A the matrix product $A^T A$ is symmetric and positive definite, we can multiply our system $\mathbf{A}\mathbf{x} = \mathbf{b}$ by A^T and solve $A^T \mathbf{A}\mathbf{x} = A^T \mathbf{b}$ instead. The CGLS implementation does this multiplication implicitly, preserving sparsity of the matrix A , which allows for more efficient (memory saving) algorithms.

Since the norm $\|\mathbf{x}\|_2$ of the intermediate solutions increases monotonically during the iteration, it is necessary to start the iteration with $\mathbf{x} = 0$ in order to not exclude any possible solution. The residual norm, on the other hand, is guaranteed to decrease monotonically, so convergence is guaranteed if numerical errors can be neglected. In practice, the iteration can be stopped as soon as the convergence speed falls below a chosen minimal value.

In the original algorithm, the value range for the vector \mathbf{x} is not restricted in any way. Particularly, the entries of the solution vector can be negative. Since negative emission intensities are impossible (they cannot even be regarded as a physically valid model for absorption), the intermediate solutions have to be projected onto the subspace of positive solutions after each step so that the positivity of the solution is guaranteed [IM04].

After this step, we have obtained a *radial map* of the model, that is a 2D grid of densities whose axes are the r and z cylinder coordinates of the corresponding voxel. Rotating this map around the z axis gives the full axisymmetric 3D model.

Implementations of iterative least-squares solvers for linear systems are widely available (Matlab's $\mathbf{x}=\mathbf{A}\backslash\mathbf{b}$ operation, for example), but normally do not allow for additional restrictions to the solution vector \mathbf{x} . Since we need to guarantee $x_j \geq 0$ for all vector components, we adapt an existing implementation [han] to suit our needs.

For the matrix manipulations that are used in the algorithm we make use of the GPU's parallel computing power using the nvidia CUBLAS¹ library. This approach speeds up the reconstruction process by two orders of magnitude (Fig. 3) with respect to a purely CPU-based implementation of the same library². However, the matrix size in the GPU accelerated version is limited by the graphics card memory and the maximum texture size³, but the resulting limit on the model size is usually not much smaller than the limit imposed by the quality of available input images.

4.3 Correcting for Asymmetries

While from a macroscopic point of view many planetary nebulae show axial symmetry, on a smaller scale there is always some deviation from perfect symmetry. This can be seen in the residual image that is left when the projection of the reconstructed model onto the image plane is subtracted from the real image. We dis-

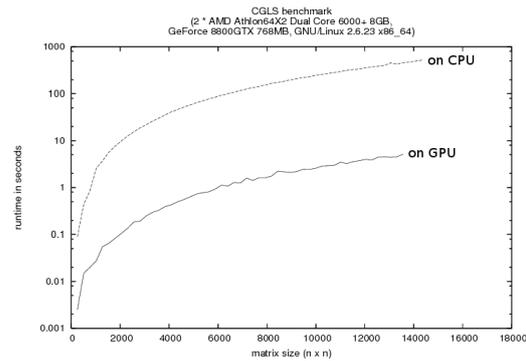


Figure 3: Comparison of computing time for CPU and GPU based implementations, in logarithmic scale, for matrices of size 1x1 up to size 14000x14000.

tribute this residual intensity among the voxels of our model so that the projection equals the original image (cf. Fig. 5).

Since there is no depth information available for the asymmetric part of the intensity, this distribution will always be ambiguous, so we have to choose a “distribution function” that gives subjectively good results when viewed from different angles.

To break up the axial symmetry, the model of cylindrical voxels is first converted into a model of cubic voxels. For efficiency, the model is aligned to the image plane, so that its x and y resolution equal the image x and y resolution. The z resolution is chosen such that the whole model fits into the voxel volume. Since by that choice of the cubic voxel model every voxel only contributes to a single pixel, we do not need to take interdependencies between voxels into account. So the only decision that is left is which voxels that share common x and y coordinates will get how much of the residual intensity of the pixel (x,y) .

A convenient distribution function turns out to be the following: Each voxel gets an amount of residual intensity that is proportional to the amount of intensity it already contributes to the pixel intensity⁴. So if the reconstructed pixel intensity is $I_p = \sum_{v \in V} I_v$ where V is the set of all voxels that contribute to the current pixel, the residual intensity is I_r , and the original pixel intensity is $I_o = I_p + I_r$, the new voxel intensities are

$$I'_v = I_v + \frac{I_v}{I_p} I_r = I_v \left(1 + \frac{I_r}{I_p} \right) = I_v \left(\frac{I_o}{I_p} \right). \quad (3)$$

As this is just a multiplication of all voxels that are projected onto the same pixel with a common value

¹ http://www.nvidia.com/object/cuda_develop.html

² The UBLAS library from http://www.boost.org/doc/libs/1_35_0.

³ On our setup with 768 MB of video RAM, the model is restricted to about 7000 entries in the radial map, or about 120 slices and 60 rings.

⁴ Alternative approaches could use more assumptions about where asymmetries are likely to occur. For example, one might assume that the exploding star itself is perfectly symmetric and only when the expanding gas cloud hits small space debris, it is asymmetrically deformed. This would imply that the residual should preferably be applied to the outer regions of the shell.

$\frac{I_o}{I_p}$, this is efficient to calculate knowing the projected intensity I_p and the residual I_r . The function is also optimal in the sense that it guarantees non-negative voxel intensities whenever possible, which in our case is always the case because the pixel intensities are always non-negative. This means that the rendered model will exactly reproduce the observed image when rendered from the original point of view, provided that the voxel resolution is large enough. It also preserves visual coherence between neighboring voxels which usually have similar intensities.

4.4 Visualizing the Results

The output of the reconstruction algorithm is a three-dimensional grid of cubic voxels, each of which has a certain emission density. To visualize this grid from an arbitrary viewpoint, *volume ray casting* can be used. Since the model is purely emissive, this means that from each screen pixel a ray is cast through the volume and intensities along the ray are summed up. This volume ray casting process is well suited for implementation on graphics hardware. The voxel model, for example, can easily be represented as a three-dimensional texture, and integrating the intensities along a given ray can be approximated by summing up the texture values at a number of fixed-distance sampling points along the ray using a fragment shader.

In order to get an impression of the chemical composition of a planetary nebula, reconstructed voxel models for different wavelengths can be shown simultaneously, assigning a color to each of them. Since the interesting spectral lines are often too close to each other (like the hydrogen and nitrogen lines at 656nm and 658nm, respectively) or even invisible to the human eye, false-color display is used. In the simplest case, when three different source images are to be displayed, these are naturally assigned to the red, green and blue color channels.

5 RESULTS

5.1 An Artificial Test Case

In order to verify the accurateness of our reconstruction, we first reconstruct an artificial model with perfect axisymmetry. For this, a radial map of intensities is drawn and projected into the image plane at different inclination angles. The resulting images are then reconstructed and the reconstructed intensity maps are compared to the original. We can show that for inclinations at least up to 45 degrees, the reconstructed radial map very closely resembles the original (see Fig. 4).

5.2 NGC7009 (Saturn Nebula)

The Saturn Nebula (Fig. 5), discovered by William Herschel in 1782, shows a bright, slightly S-shaped

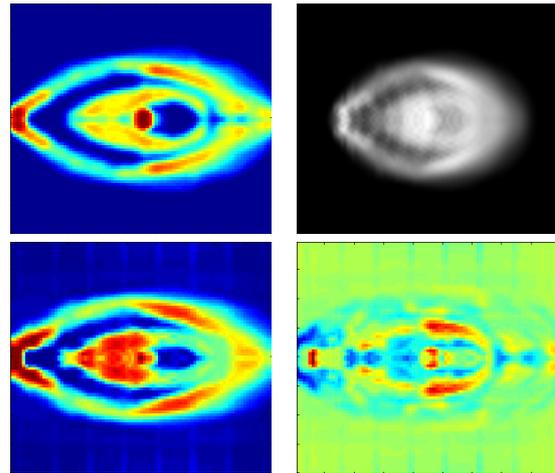


Figure 4: An artificial “nebula” used for testing the reconstruction (from top to bottom and left to right): radial map, rendered view with 35 degrees inclination, reconstructed radial map (reconstructed with somewhat lower resolution and scaled to the size of the original radial map) and difference image of the radial maps

structure in the center, surrounded by a darker, barrel-shaped one. The S-shaped structure has noticeable reddish glowing tips. The original images were moderately disturbed by stars, so slight preprocessing had to be done.

5.3 Mz3 (Ant Nebula)

The Ant Nebula, discovered in 1922 by Donald Menzel, has a number of different gaseous outflows from its bright center. The most visible outflow consists of two approximately spherical lobes, but more subtle “rays” can also be observed outside these lobes. Interestingly, all these features share a common axis of symmetry, so in principle the simultaneous reconstruction of all important features would be possible (Fig. 6). However, due to the large difference in intensity and the linear scale chosen in the visualization, these structures cannot be observed together in one output image. Anyway, the interesting fine-grained asymmetries of the spherical lobes are visible in the asymmetry-corrected output (Fig. 7).

Due to the presence of many bright stars in the original images, heavy preprocessing⁵ needed to be done. This may have caused loss of fine-grained structures in some areas.

Magnor et. al also reconstructed the Ant Nebula using their CIVR approach. The algebraic reconstruction algorithm that we implement outperforms CIVR – which leads to results that are optically very simi-

⁵ including removing the background by thresholding, and removing stars by masking and filling the masked regions by diffusion

lar to the purely symmetric part of our reconstruction – by far: while using CIVR, “the reconstruction of a 128x32-pixel density map takes approximately one day on a 2.4 GHz PC in conjunction with an nVidia GeForce FX 3000 graphics card” [MKHD04], our algebraic approach can reconstruct the nebula with comparable resolution in a matter of seconds, including asymmetry correction.

5.4 NGC6543 (Cat’s Eye Nebula)

The Cat’s Eye Nebula (Fig. 1), discovered by William Herschel in 1786, has a very complex and not quite axisymmetric structure. Its reconstruction is nevertheless quite accurate due to the asymmetry correction. Due to its relatively large brightness compared to the surrounding stars, no preprocessing was needed to get clean results.

6 CONCLUSIONS

We have presented an efficient algebraic reconstruction approach to derive 3D information from single images of axisymmetric and purely emissive objects like planetary nebulae. Axial symmetry is broken in a controlled way in order to achieve closer resemblance between the model and observational data. The calculations are carried out efficiently by making use of the GPU’s parallel computing power, and the resulting models closely resemble the actual photographs.

There are, however, some limitations inherent to our approach. The asymmetry correction is not based on any physical measurement and can only be heuristically and ambiguously determined, as long as no additional data is provided. The reconstruction is also only possible for nebulae whose axis of symmetry is not too far inclined with respect to the image plane because no reliable 3D information can be derived if the axis is close to parallel to the viewing direction. For the same reason, objects without symmetry cannot be reconstructed at all.

However, for axisymmetric objects the results closely resemble the original. The CGLS algorithm is guaranteed to converge, and small asymmetric features can be included in such a way that the original image is exactly reconstructed. For objects with spherical symmetry, the algorithm is also applicable, though it could be optimized further to benefit from the stricter constraints.

The resulting three-dimensional models can be rendered from arbitrary perspectives. This allows for a wide field of applications, in scientific as well as artistic contexts. They may help understanding the complex structure of planetary nebulae and the physical mechanisms that underlie their formation.

In ongoing research we evaluate the possibility of allowing a wider variety of symmetry constraints. A user-supplied set of model parts such as cylinders,

spheres, and other geometric primitives that may represent a physical explanation of the shape of the object could be used instead of just a single cylinder. This would allow to model more complex symmetries such as the helix that is present in the Saturn Nebula.

In order to assure physical realism, depth information from Doppler shift measurements could easily be incorporated into the error function, which would allow to loosen the symmetry constraints where better depth information is available.

ACKNOWLEDGEMENTS

This project is partially funded by the German science foundation DFG under grant 444 MEX-113/25/0-1 and the Mexican science foundation CONACyT under a bilateral cooperation grant.

All astronomical images taken from the Hubble Space Telescope [HST], partly acquired through the MAST search engine [MAS].

REFERENCES

- [AKR00] B. Armsdorfer, S. Kimeswenger, and T. Rauch. Effects of CSPN models on PNe shell modeling. In *Ionized Gaseous Nebulae*, November 2000.
- [CP83] N. Calvet and M. Peimbert. Bipolar nebulae and type I planetary nebulae. *Revista Mexicana de Astronomía y Astrofísica*, 5:319, 1983.
- [Cur18] H. Curtis. The planetary nebulae. *Publ. Lick Observatory*, Part III(13):57–74, 1918.
- [EMB⁺03] C. Ercolano, C. Morisset, M. Barlow, P. Storey, and X.-W. Liu. Three-dimensional photoionization modelling of the planetary nebula NGC3918. *Monthly Notices of the Royal Astronomical Society*, 340:1153–1172, 2003.
- [han] `netlib.org` CGLS algorithm. file `REGU/cgls.m` from <http://netlib.org/numeralgo/na4-matlab7.tgz>.
- [Han93] K. M. Hanson. Special topics in test methodology: Tomographic reconstruction of axially symmetric objects from a single dynamic radiograph. *Prog. in Astronautics and Aeronautics*, 155:687–698, 1993.
- [Han96] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*, chapter 6. Polyteknisk Forlag, 1996.
- [HST] Hubble space telescope. <http://hubblesite.org/>.
- [IM04] Ivo Ihrke and Marcus Magnor. Image-based tomographic reconstruction of flames. In *ACM Siggraph / Eurographics Symposium Proceedings and Symposium on Computer Animation*, pages 367–375, June 2004.
- [KK68] G. Khromov and L. Kohoutek. Morphological study of planetary nebulae. In D. Osterbrock and C. O’Dell, editors, *Planetary Nebulae*, pages 227–235. IAU Symposium 34, 1968.
- [KPF78] S. Kwok, C. Purton, and P. Fitzgerald. On the origin of planetary nebulae. *Astrophysical Journal*, 219:L125–L127, 1978.
- [KS88] A. C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988. Especially chapter 7: “Algebraic Reconstruction Algorithms”.

- [LHM⁺07] Andrei Lințu, Lars Hoffmann, Marcus Magnor, Hendrik P. A. Lensch, and Hans-Peter Seidel. 3D Reconstruction of Reflection Nebulae from a Single Image. In *VMV 2007*, November 2007.
- [LLM⁺07a] Andrei Lințu, Hendrik P. A. Lensch, Marcus Magnor, Sascha El-Abed, and Hans-Peter Seidel. 3D Reconstruction of Emission and Absorption in Planetary Nebulae. In Hege, Hans-Christian, Machiraju, and Raghu, editors, *IEEE/EG International Symposium on Volume Graphics*, pages 9–16, September 2007.
- [LLM⁺07b] Andrei Lințu, Hendrik P. A. Lensch, Marcus Magnor, Ting-Hui Lee, Sascha El-Abed, and Hans-Peter Seidel. A Multi-wavelength-based Method to de-project Gas and Dust Distributions of several Planetary Nebulae. In *Asymmetrical Planetary Nebulae IV*, June 2007.
- [MAS] Multimission archive at STScI. <http://archive.stsci.edu/>.
- [MF89] C. Mellema and A. Frank. Radiation and gasdynamics of planetary nebulae – V. Hot bubble and slow wind dynamics. *Monthly Notices of the Royal Astronomical Society*, 273:401–410, 1989.
- [MKHD04] M. Magnor, G. Kindlmann, C. Hansen, and N. Duric. Constrained inverse volume rendering for planetary nebulae. *Proc. IEEE Visualization 2004 and Austin and USA*, pages 83–90, October 2004.
- [MKHD05] M. Magnor, G. Kindlmann, C. Hansen, and N. Duric. Reconstruction and visualization of planetary nebulae. *IEEE Trans. Visualization and Computer Graphics*, 11(5):485–496, September 2005.
- [NGN⁺01] D. R. Nadeau, J. D. Genetti, S. Napear, B. Pailthorpe, C. Emmart, E. Wesselak, and D. Davidson. Visualizing stars and emission nebulae. *Computer Graphics Forum*, 20(1):27–33, March 2001.
- [OF06] Donald E. Osterbrock and Gary J. Ferland. *Astrophysics of Gaseous Nebulae and Active Galactic Nuclei*. University Science Books, 2006.
- [Sab84] F. Sabbadin. Spatiokinematic models of five planetary nebulae. *Monthly Notices of the Royal Astronomical Society*, 210:341–358, 1984.
- [Sau97] W. Saurer. Morphology and expansion characteristics of the planetary nebula M1–79. *Astronomy & Astrophysics*, 326:1187–1194, October 1997.
- [SCB⁺00] F. Sabbadin, E. Cappellaro, S. Benetti, M. Turatto, and C. Zanin. Tomography of the low excitation planetary nebula NGC 40. *Astronomy & Astrophysics*, 355, 2000.

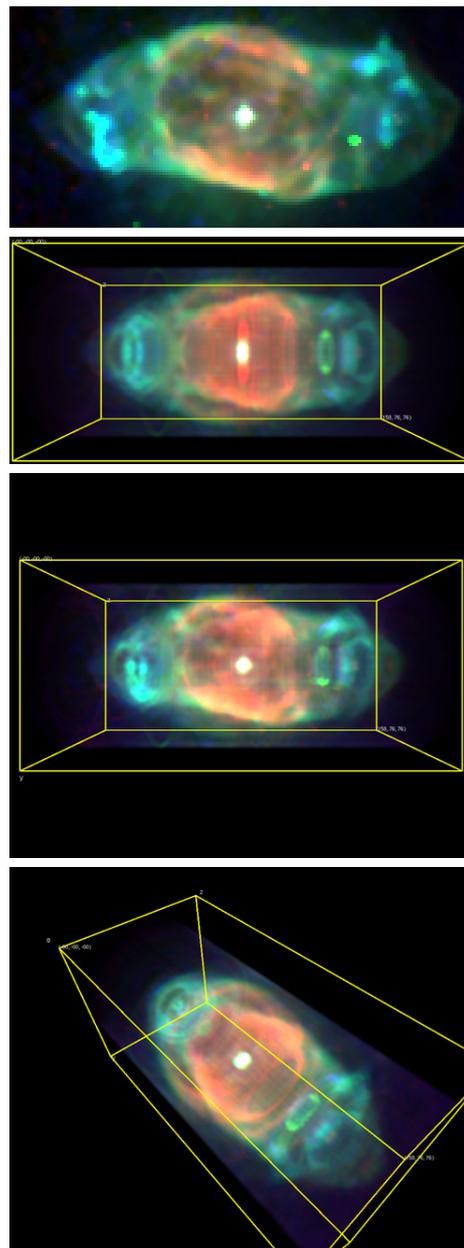


Figure 5: The Saturn Nebula. From top to bottom: images taken using filters for 502nm, 555nm and 658nm that are assigned to the red, green and blue color channels, respectively; reconstruction without asymmetry correction; reconstruction with asymmetry correction, from earth; reconstruction with asymmetry correction, from outer space. Note how the S-shape of the original image is lost during reconstruction because it violates the axisymmetry constraint, and how the asymmetry correction restores this important feature.

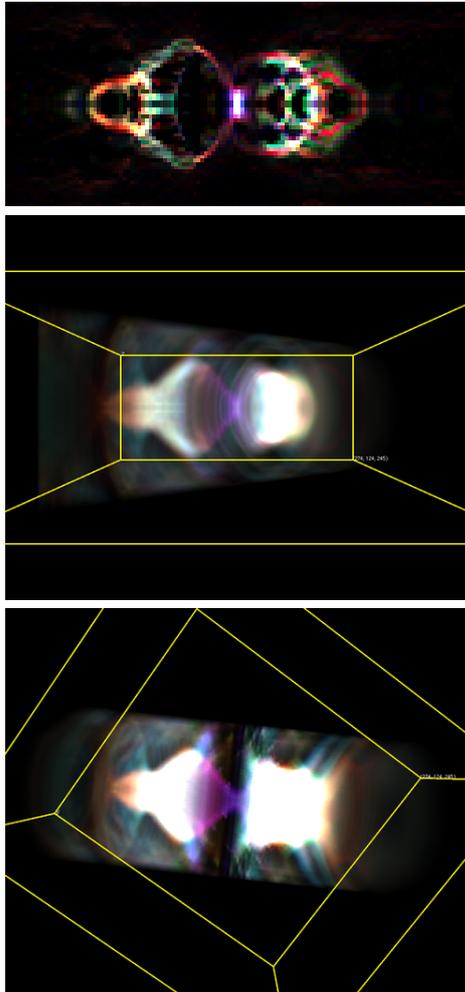


Figure 6: Reconstruction of the Ant Nebula without asymmetry correction: radial map (top) and reconstructed view from earth (middle) and from outer space (bottom). The red, green and blue color channels are assigned to 673nm, 658nm and 487nm, respectively.

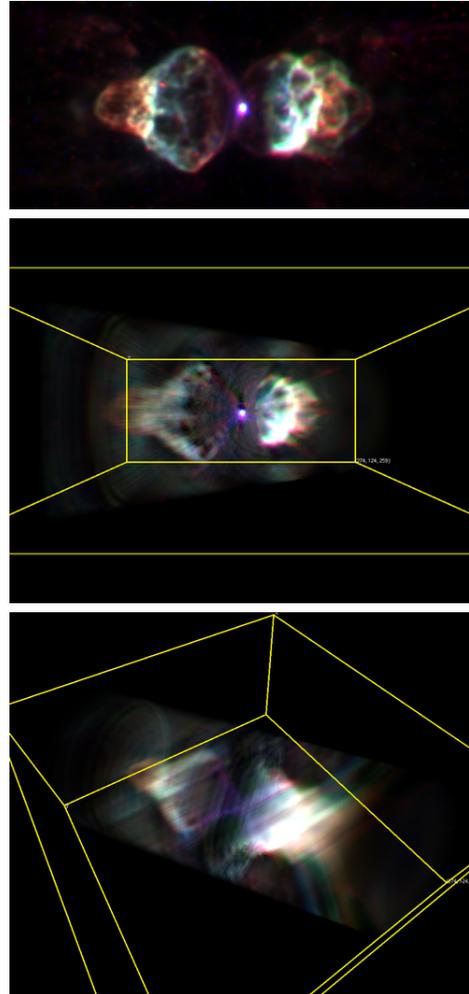


Figure 7: The Ant Nebula. Top: images taken using filters for 487nm, 658nm and 673nm that are assigned to the red, green and blue color channels, respectively. Middle and bottom: asymmetry-corrected reconstruction, view from earth and from outer space. In the bottom image, artifacts of the asymmetry correction due to overly bright regions in the input image can be observed as lines parallel to the viewing direction of the original image.

Interactive Exploration of Large Event Datasets in High Energy Physics

Max Hermann Alexander Greß Reinhard Klein

Universität Bonn
Institut für Informatik II - Computergraphik
D-53117 Bonn, Germany
{hermann, gress, rk}@cs.uni-bonn.de

ABSTRACT

In high energy physics the structure of matter is investigated through particle accelerator experiments where particle collisions (events) occur at such high energies that new particles are produced. Providing tools for interactive visual inspection of billions of such events occurring in an experiment in an intuitive way is a challenging task. In order to solve this problem we built on previous approaches for visual browsing through image databases and extend them in several ways in order to allow efficient navigation through the collision event datasets. The key features of our novel browsing technique are its applicability to the very large event datasets, a more intuitive selection method for specifying a region of interest, and finally a clustering-based technique that further simplifies and improves the navigation process. We demonstrate the potential of our novel visual inspection system by integrating it into an event display application for the COMPASS experiment at CERN.

Keywords: Interactive browsing, similarity-based visualization, Multidimensional Scaling, Earth Mover's Distance.

1 INTRODUCTION

High energy physics (HEP) investigates the inner structure of matter by performing experiments where highly accelerated particles collide with each other or with a fixed target. Each such collision results in the birth of multiple new particles with individual characteristics (charge, momentum, etc.), which is called an *event*. A particle accelerator experiment utilizes a setup of different detectors to identify events and to be able to reconstruct the trajectories of the new particles produced in an event (called *tracks*) and thus their respective physical characteristics. A number of applications, typically called *event displays*, have been developed for the purpose of visualizing the reconstruction of an event and its tracks. However, current state-of-the-art event displays (e.g. [18, 24, 16, 2, 7, 12, 17]) focus primarily on visualizing single events in various ways [6].

With the ever growing size and energy of modern particle accelerators, the number of events produced in an experiment and used in later analysis constantly increased over the last years. A typical event dataset encountered in analysis consists of millions of events, and hundreds of such datasets are produced in the course of a year. In the COMPASS experiment at CERN [1],

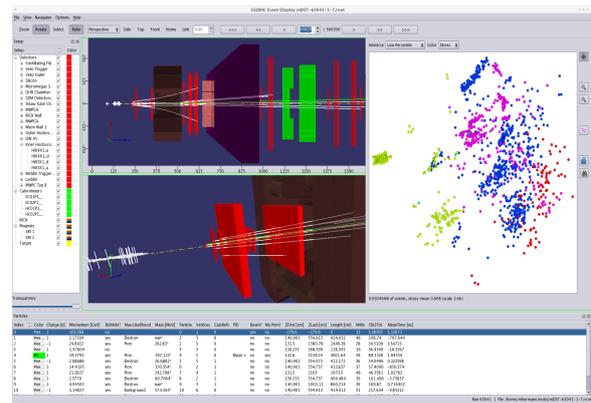


Figure 1: The interactive browsing system (visible to the right) in use in the COMPASS event display.

about 350 TB raw data per year are produced. Through preprocessing and filtering this raw data is reduced by factor 100, and the results are stored in several 2 GB files containing roughly half a million events each.

In event displays, the visualization typically starts by specifying an event dataset, from which events to be visualized can be chosen using very simple techniques, for instance by specifying the identification number of an event or by moving the focus to the preceding or following event in the time line. The CMS event display further has an option to automatically display a random event from the data source every 3 seconds [4].

These present tools for event navigation are neither suited nor designed for purposeful navigation. We therefore believe that a more sophisticated event navigation tool, which permits interactive browsing of the event dataset in an intuitive way will greatly simplify the interactive analysis of physical event data. The in-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright UNION Agency – Science Press

teractive event browser described in this paper exactly fulfills this purpose.

Its workflow was inspired by Rubner et al. [22], who proposed a similar navigation for image databases. The basic idea is to represent events on a two-dimensional map where similar events are located close to each other. To produce such two-dimensional maps, we follow Rubner's approach in that we use multidimensional scaling and define a similarity measure for the events based on the so-called Earth Mover's Distance.

To make the approach scalable to very large numbers of events, in terms of usability as well as computation time, only subsets of events are shown on the map at a time, but the map can iteratively be refined by the user through selecting regions of interest.

While the use of two-dimensional maps has a great potential for navigation through a complex dataset, a major problem is the distortion of distances introduced by the dimensionality reduction that is needed for creating the map. This is especially relevant when the user selects a certain region of interest in the map to interactively browse through similar events or to refine the map to a certain subset of similar events. For this purpose, the selection of similar events solely based on a 2D neighborhood would certainly not be adequate, since because of the generally unavoidable distortion, some pairs of events shown close to each other may exhibit significant dissimilarity (even though the distance of points in the map is in general proportional to the dissimilarity of the respective events). Such outliers should not be included in the refined map to avoid unnecessarily high distortions, which would hinder an efficient navigation.

In this paper, we tackle this problem by defining a new criterion for transferring a selected region on the map to a selection of events in the non-Euclidean space (Section 5). This technique accounts for local distortions in the map projection and is robust to the aforementioned kind of outliers. Another important aspect of the new technique is that it can also be applied when subsampling strategies have been applied during the calculation of the maps, on which the user selects the regions of interest, i.e. when only a partial Euclidean embedding of the respective event set has been determined. This is of relevance because such subsampling strategies are inevitable to make the approach applicable also to very large datasets, as we will explain in Section 3.2.

This improved strategy for selecting a region of interest in the dataset allows for a better user control of the navigation process since it avoids refining into regions corresponding to unwanted outliers contained in a selected map region. Additionally, the control over the navigation process can be further improved by integrating a cluster selection technique which allows not only to inspect certain clusters of interest more easily,

but also helps to produce less distorted maps during iterative refinement.

The paper is organized as follows: Section 2 sketches the previous work on navigation approaches and also briefly describes the basics of the fundamental algorithms for dimensionality reduction and clustering used in this paper. Section 3 describes how we define a similarity measure for HEP event datasets, which is a prerequisite for being able to determine map representations of events. Additionally, it discusses how we make this approach applicable to large-scale datasets by the use of sampling. Section 4 describes the process of interactive navigation through event datasets including cluster selection and iterative refinement. Furthermore, the applicability and limitations of recent approaches to select a region of interest for the refinement are discussed. Section 5 describes the proposed new technique for specifying the region of interest. Finally, Section 6 demonstrates the usefulness of the proposed navigation technique on examples of real event datasets, and Section 7 concludes with a summary.

2 PREVIOUS WORK

If a similarity measure in form of a metric can be supplied for a specific type of data we speak of *metric data*. We call the distance in the corresponding metric space *dissimilarity* to emphasize the fact that the metric space is in general not an Euclidean space.

2.1 Map Navigation

The general idea of a map representation of metric data is to embed it into the Euclidean \mathbb{R}^2 where the distance between two points in the Euclidean space approximates the dissimilarity between the corresponding objects according to the given metric.

In the context of image databases, Rubner et al. [22] describe a navigation technique based on map representations of images. To compute such map representations, a metric for images is proposed based on color distribution. Which images are shown on a map is specified by queries where a query itself is stated in terms of a color distribution. In the navigation process described in [22] the user can create a new map by selecting a point in the current map. For the selected point a query is generated and the k images, which are the most similar to the queried point according to its color distribution, are shown on the new map. The number k of visible images is decreased after each navigation step.

The semantic image browser by Yang et al. [27] also makes use of map representations of images. To select which images are shown on such a map, the user must specify a sample image and a dissimilarity threshold which can be interactively chosen through a scaling bar. Exactly those images are selected whose dissimilarities to the specified sample image are not greater than the specified threshold.

2.2 Earth Mover's Distance

For a dataset whose data items are described by distributions, a metric can be defined by a solution of the so-called mass transportation problem [9, 21]. This solution corresponds to a metric called *Monge-Kantorovich Distance*, or in context of statistics, to the *Mallows Distance* [14]. Also in context of image databases, this similarity measure has been applied in several works [19, 10, 22, 23, 27]. Here, it was given the name *Earth Mover's Distance* (EMD) [22].

The original mass transportation problem was stated by Monge 1781, where he asks how a piece of soil can be moved from fillings to excavations with the least amount of work. Formally Monge's problem can be stated as a linear program in the following way [21]: Let position and masses of the fillings and excavations be given by the discrete distributions $a = \{(x_1, p_1), \dots, (x_m, p_m)\}$ and $b = \{(y_1, q_1), \dots, (y_n, q_n)\}$ where the sites x_i and y_j are typically in \mathbb{R}^d and the masses are normalized, i.e. $\sum p_i = \sum q_j = 1$. The work to move a unit amount of mass from site x_i to site y_j is quantified by the real-valued function $c(x_i, y_j) \equiv c_{ij}$, the so-called *ground distance*. A solution is given by an assignment $\mu : \{1, \dots, m\} \times \{1, \dots, n\} \rightarrow \mathbb{R}$ of how much mass is transported from a filling site to an excavation site. The optimal assignment then is found by solving the following linear program:

$$\min_{\mu} \sum_{i=1}^m \sum_{j=1}^n \mu(i, j) c_{ij} \quad (1)$$

$$\begin{aligned} \text{subject to} \quad & \mu(i, j) \geq 0 & i=1, \dots, m \wedge j=1, \dots, n \\ & \sum_{j=1}^n \mu(i, j) = p_i & i=1, \dots, m \\ & \sum_{i=1}^m \mu(i, j) = q_j & j=1, \dots, n \end{aligned}$$

As shown in [22], the solution to this linear program is truly a metric.

In summary, solving the mass transportation problem lets us define a metric between discrete distributions by the means of an adequate ground distance c_{ij} .

2.3 Multidimensional Scaling

The first *Multidimensional Scaling* (MDS) method, the so-called classical MDS, was introduced by Torgerson in 1952 [25]. For a detailed overview of this and more recent MDS methods, including its metric and non-metric variant, we refer to [3].

In general, MDS methods determine a coordinate representation of dissimilarity data in low dimensional Euclidean space such that the pairwise coordinate distances approximate the dissimilarity data. Because no specific coordinate can be deferred from the pairwise distances only, a reference coordinate system is chosen with the barycenter of the data as origin and an

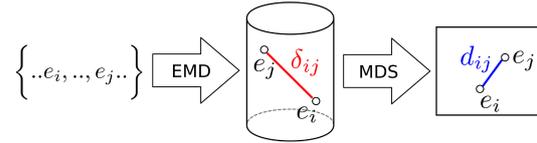


Figure 2: Creation of an event map: First the set of events to be visualized is embedded into the metric event space (depicted as cylinder) and subsequently MDS-projected onto an event map (depicted as rectangle). Two events e_i and e_j are shown with their dissimilarity δ_{ij} and distance d_{ij} .

arbitrary rotation. Thus, the resulting coordinates are unique up to rotation and reflection.

To estimate the quality of approximation, one can relate the resulting coordinate distances d_{ij} to the original dissimilarities δ_{ij} by the use of a stress function such as *Kruskal stress* [13]:

$$s_{\text{Kruskal}} = \sqrt{\frac{\sum_{i < j} (d_{ij} - \delta_{ij})^2}{\sum_{i < j} \delta_{ij}^2}} \quad (2)$$

The Kruskal stress is zero for a perfect reconstruction, while non-zero stress indicates a distortion of the data.

2.4 Clustering

An overview of the numerous work on cluster analysis of high-dimensional data is outside the scope of this paper. We refer to [26] for a recent survey.

Clustering algorithms that operate on metric data, where no representation in an Euclidean space is known, are usually called relational clustering algorithms. An established relational clustering algorithm that partitions the given metric data into a fixed number of clusters is the *Partitioning Around Medoids* (PAM) approach by Kaufman and Rousseeuw [11]. It is similar to the non-relational k -means approach, but tries to find k *representative objects* from the dataset, called *medoids*, that minimize the sum of intra-cluster dissimilarities. In our system, we will use a variation of this approach, called *Clustering Large Application* (CLARA) [11], designed to handle large datasets. It first draws a random sample of the dataset, then uses PAM to find representative objects from this sample, and finally assigns all objects from the dataset to the determined clusters. This is repeated for multiple samples of the dataset, and the best solution is returned.

3 CREATING A MAP OF EVENTS

The similarity measure that we explain in Section 3.1 gives us an embedding of the event dataset into a metric space which we call *event space*. The metric is defined by the pairwise dissimilarity between two events. For arbitrary subsets of this event space a two-dimensional representation of the events can be produced via dimensionality reduction by MDS. The structure of the events can be visualized by showing this representation on a

map, which is what we call an *event map*. The map creation process is summarized in Fig. 2.

In the following we will show how the metric embedding is solved and discuss the computational complexity of the map creation process and the thereby resulting need for subsampling of large datasets.

3.1 Embedding into metric space

An event is fully described by describing the therein produced particles with their trajectories (tracks). The number of tracks in an event is variable and the tracks have no specific order. All tracks can be characterized by the same fixed number p of real-valued physical parameters. Thus a track can be represented by a vector $t \in \mathbb{R}^p$, while an event in turn cannot be considered as a vector of tracks.

In general, it is difficult to define a metric on events because there exists no precise notion of similarity for events in physics. But in contrast to that, defining a metric on tracks is easier because we can exploit the rich set of metrics in \mathbb{R}^p . A suitable metric on tracks must take into account (a) the inhomogeneous ranges of the different physical parameters and (b) the correlation between different parameters. Both requirements are met by the statistical Mahalanobis distance

$$d_{\text{Mahalanobis}}(f, g) = \sqrt{(f - g)^T \Sigma^{-1} (f - g)} \quad (3)$$

where Σ is the covariance matrix for all parameters.

Events are sets of track and, since we can consider a set as a special kind of discrete distribution by assigning each item the same weight, we can use the EMD as described in Section 2.2 to define a metric on events. For this, we formalize an event e as an equally weighted discrete distribution of its n tracks (represented as vectors t_i)

$$e = \{(t_1, 1/n), (t_2, 1/n), \dots, (t_n, 1/n)\} \quad (4)$$

and use the metric (3) as ground distance between tracks.

3.2 Making the Approach Applicable to Large-Scale Datasets

As stated in the introduction, our aim is to provide interactive navigation for event datasets, which consist, even when restricted to relatively short time-frames, of millions of events. Similarly to the known navigation approaches for image databases (see Section 2.1), also our navigation approach, which will be described in Section 4, requires frequent recalculation of Euclidean embeddings for different subsets of the dataset. Unfortunately, it is practically infeasible to calculate two-dimensional Euclidean embeddings for millions of events in a way suitable for such an interactive application. This is detailed in the following section. To circumvent this issue we use the strategy of subsampling as described further below.

Feasibility of the map creation

Computing a map representation of a set of events involves first the computation of all pairwise dissimilarities based on the EMD, and second the calculation of the Euclidean embedding via MDS.

According to [19], calculating the EMD is, in general, in $\mathcal{O}(M^3)$, where in our case M corresponds to the average number of tracks per event. For certain specific ground distances such as the L_1 -metric, there exist faster algorithms for calculating the EMD [10, 15], which utilize the special structure of the ground distance to solve the linear program more efficiently. However, for the Mahalanobis ground distance (3) used in our approach these improvements are not applicable.

Therefore, in our case the time needed for calculating the full $N \times N$ dissimilarity matrix for a set of N events is in $\mathcal{O}(N^2 \cdot M^3)$. Since also the storage space required for the full dissimilarity matrix is quadratic with respect to the number of events, computing and storing the full matrix quickly becomes infeasible with an increasing number of events. Therefore, instead of calculating the full dissimilarity matrix before constructing the Euclidean embedding via MDS, we calculate the dissimilarities on demand, i.e. at the time when they are needed for the MDS calculation. This however implies that new EMD evaluations may have to be performed whenever new Euclidean embeddings are calculated during the interactive navigation process.

In addition to the time required for calculating dissimilarities, also the time required for calculating the Euclidean embedding via MDS is of relevance in this context. In case of the classical MDS, which has in general a lower time-complexity than the metric or non-metric MDS variant, this calculation requires $\mathcal{O}(N^3)$ time for a set of N events if singular value decomposition (SVD) is used for determining the basis of the Euclidean space [20]. In our case where a dimensionality reduction to only two dimensions is desired, the practical runtime of the MDS can be largely improved by using a Lanczos iteration [5] instead of the SVD to compute only the first two eigenvalues and eigenvectors. In addition, we use the fast approximation technique for evaluating the MDS proposed by [28]. With these improvements we observe in our practical application that the time for computing the MDS projection is rather insignificant in comparison to the time required for the associated EMD evaluations. Nevertheless practical timings show the quadratic dependence of the overall runtime on the number of events.

Subsampling

To make the approach applicable to large-scale datasets despite the computational complexity discussed above, we use sampling strategies as follows. Instead of calculating the Euclidean embedding of the whole event

dataset or of the whole set of events that the viewer is currently interested in, the Euclidean embedding is constructed only for a subset of this set of events, whose size allows for a rapid evaluation and thus for a prompt feedback in the navigation process. We found that for the most real datasets, the selection of a representative subset is possible, that exhibits the same, or at least similar, characteristics as the original complete set of events. For a meaningful map representation of a set of events, first of all its overall structure, which is characterized by the formation of clusters and their positioning in relation to each other, is important for the viewer. Dominant clusters that exhibit a large number of events are retained in the map representation with high probability irrespective of the sampling strategy used. Therefore, the use of random sampling is usually sufficient.

4 INTERACTIVE NAVIGATION

In this section we describe our approach for interactive navigation through huge event datasets by the use of event maps, which we call *event browsing*. There are two motivations for employing such a browsing through several event maps representing smaller and smaller subsets of the dataset, corresponding to successively narrower regions in event space:

- **Stress.** Due to the distortion of the MDS-scaling, an event map of all events cannot convey the fine structures and sub-structures of the dataset. In contrast, a map representation corresponding to a smaller region in event space exhibits less distortion and can thereby convey finer structures.
- **Subsampling.** In most practical cases subsampling of a huge event set is needed to create the event map. On such an event map not all events are accessible. But the subset of a small enough event space region can be shown on an event map without subsampling.

The main navigation technique in this approach is a technique we call *refinement* where the user selects a region of interest on an event map and subsequently a new map is computed based on the selection. Repeated application of this refinement yields maps of smaller and smaller subsets of the dataset until a sufficiently narrow region of interest in event space is reached. We call this interactive process *iterative refinement*.

The second technique used in our system is based on clustering the event dataset. Besides improving the visualization of the dataset structure, clustering can support the navigation process by providing an alternative selection method we call *cluster selection*.

We first describe the cluster selection in Section 4.1, while the discussion of iterative refinement is postponed to Section 4.2.

4.1 Cluster Selection

The integration of clustering techniques (cf. Section 2.4) into the interactive event browser was

motivated by the following observation: When analyzing an event dataset using the proposed similarity measure, the contained events typically fall into several clusters and sub-clusters of similar events. Therefore, the detection and labeling of clusters in the event map is an important component of our system to support intelligent user navigation through the dataset.

To visualize the cluster membership of events, the events on an event map can be colored according to their cluster membership. This cluster visualization lets the user recognize regions on the map where due to the MDS projection separate clusters have been mapped on top of each other. Furthermore, the user can select certain clusters and restrict the event map to show only events from this clusters, i.e. in further iterative refinement only events originating from the selected clusters are considered. We call such a restriction of the event set to events from selected clusters *cluster selection*.

Selecting and exploring clusters provides a strategy to solve the problem of overlapping clusters in a map representation. This can be seen as clutter reduction technique [8] but additionally the cluster selection has the advantage, that for the restricted set of events a new map layout is calculated, which is usually less distorted.

4.2 Iterative Refinement

During iterative refinement, the user selects a region of interest on the map from which a new event map is computed which only consists of events inside the region of interest. The underlying idea is to enable the user to examine a subset of the event dataset (which corresponds to a narrower region in event space) in more detail. But due to distortion introduced by MDS projection, a region on the map may contain events which are highly dissimilar to most of the other events inside that specific region. Providing a selection method which is robust to such outliers is non-trivial.

Another requirement for a selection method emerges from the fact that the event set has been subsampled for the calculation of the map. By a selection, we want to determine not only events from the subsample, but rather a region in the event space. Directly mapping the selected region (containing only visible events) into the event space (containing the whole event dataset) is not possible in general since the event space is a pure metric space.

Applicability of recent selection methods

Classical 2D selection techniques like rectangular, elliptic, or freehand selection tools select only a subset of the events visible on the map. Thus they do not meet the requirements for a selection technique on subsampled event maps in the context of iterative refinement.

A selection method similar to the navigation through image databases as proposed by Rubner et al. [22]

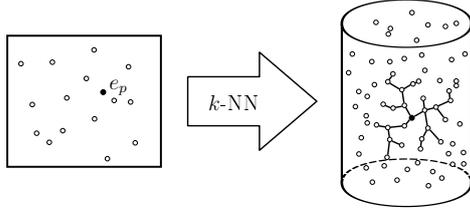


Figure 3: Scheme of previous selection methods.

(cf. Section 2.1) can be stated in our context of event datasets as follows: Starting from a single selected event e_p , the *pivot event*, the set of k nearest neighbors in the event space is determined, from which the new event map can be computed (see Fig. 3). To allow an *iterative* refinement by means of this method the parameter k should be decreased after every refinement step. The main drawback of this selection method is the missing visual feedback about the selected region on the map that would give the user a spatial impression about the region that will be shown on the refined map. Since this is especially important in the context of large datasets where several subsequent refinements are performed, Yang et al. [27] propose to mark the selected k nearest neighbors on the map. But due to distortion, the k neighbors in metric space will probably not be direct neighbors on the map. Furthermore, in the presence of subsampling, most of them may not lie on the map at all. Thus, because this method does not take into account the described problems due to distortion and subsampling, it is not suited for the iterative refinement.

5 NOVEL SELECTION TECHNIQUE FOR ITERATIVE REFINEMENT

Our novel selection technique incorporates the advantage of classical 2D selection techniques to provide the user a direct visual feedback about the selected region, but further fulfills the requirements described in the context of iterative refinement (see Section 4.2) by considering the distortion as well as the subsampling.

The distortion is taken into account by not considering outlying events in the selected region of interest. As an indicator for outliers we use the distortion that a respective event experiences. This is measured by the so-called *local stress* which will be defined in Section 5.1. Based on this, we present our novel selection technique in Section 5.2.

5.1 Local Stress as Indicator for Outliers

Even though MDS generally tries to choose the distances of the points in the map proportional to the dissimilarities of the respective events, it is not unlikely, because of the dimensionality reduction to only two dimensions, that some distances in the map differ largely from the respective dissimilarities. The stress of a map

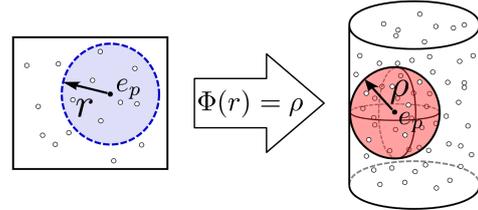


Figure 4: Iterative refinement with the proposed technique. Dependent on a radial selection of radius r around a pivot event e_p , a corresponding set of events in the event space with a maximum dissimilarity ρ to e_p is determined.

(see Section 2.3) describes the overall distortion of dissimilarities. In analogy, to measure the local distortion inside a selected region at a certain point, we define the *local stress* of an event e_i relative to a set E of neighboring events on the map as

$$s_E(e_i) = \frac{\sum_{e_j \in E} d_{ij}}{\sum_{e_j \in E} \delta_{ij}} \quad (5)$$

where d_{ij} is the distance after the projection between e_i and any event e_j in the selection and δ_{ij} the dissimilarity of the corresponding events. $s_E(e_i)$ measures the average distortion of the dissimilarities between event e_i and all other events inside E . A value of 1 denotes that these dissimilarities directly map to the respective map distances on average, and $s_E < 1$ indicates the amount of distortion with respect to set E .

If certain events have a significantly high distortion with respect to E , these events could be fundamentally farther away in metric space than the distance on the map suggests and can thus be considered as outliers. Therefore, the local stress can be used to characterize outliers concerning the current selection that should not be considered in the further refinement.

5.2 Proposed Technique

The idea is to define the selection in the event space as the set of all events which are similar to events contained in the radial selection on the map which are not considered as outliers.

Starting from a pivot event e_p , the user specifies interactively a radial selection region of radius r on the map, see Fig. 4. The thereby selected region defines a set $K = K(e_p, r)$ of events on the map.

To transfer the radial selection on the map to a selection in event space we estimate a dissimilarity ρ in the event space such that all events which exhibit a dissimilarity smaller than ρ towards e_p can be considered to be the region of interest in event space from which the refined map is computed. Choosing a subset in this way from the set of all events has the advantage that events in the subset are similar to most of the events in the selected radial region. Thus they are a good estimation to the region of interest the user wants to explore by the selection. We denote the dissimilarity estimation by $\rho = \Phi(e_p, r)$.

As discussed in the previous section, the distortion leads to outliers. Thus, to take the distortion in the calculation of ρ into account, we identify outliers in the selected region as follows: If the local stress of the respective event deviates more than α times the standard deviation σ_K from the local stress mean \bar{s}_K of all events inside the radial region, then it is considered as outlier. Thereby we can define the set of non-outliers as

$$E_\alpha = \{e_i \in K(e_p, r) \mid s_K(e_i) < (\bar{s}_K - \alpha \cdot \sigma_K)\} \quad (6)$$

where the parameter α specifies the “strictness” of the outlier classification. Under the assumption of normal distribution, a choice of $\alpha \geq 1$ is reasonable.

Finally, ρ is estimated by the greatest dissimilarity a non-outlying event inside the radial selection exhibits towards the pivot event:

$$\Phi_\alpha(e_p, r) = \max\{\delta_{ip} \mid e_i \in K \setminus E_\alpha\} \quad (7)$$

However, in the case of very high stress or several overlapping clusters inside the radial selection region, the maximum dissimilarity ρ may be still overestimated. Therefore optionally a λ -quantile can be applied, which means that only the most similar λ percent of the non-outlying events in the radial selection region for estimating ρ are considered.

The presented technique indeed gives the user feedback about the selected region, but further is also robust against outliers. Navigation is greatly improved by the fact that in response to a selected region, a new map of events representing that region is shown. Parts of the map can now be examined on different scales by specifying repeatedly regions of differing radii. This is a real improvement in contrast to previous selection methods discussed in Section 4.2.

6 APPLICATION

This section demonstrates the potential of the event browser within a real event display application. We tightly integrated the event browser in an event display, which has been developed for the COMPASS experiment at CERN. Fig. 1 shows a screenshot.

Integration into an Event Display

We connected the event browser to the event display in such a way that every event selected inside the currently visible map is passed to the event display for visualization and further analysis. This allows for a rapid examination of the events on the map. Because nearby events on the map are in fact similar to each other (as approved by physicists), the user can restrict the examination to several events from each visible cluster to get an impression of the type of events represented by that cluster or region on the map. In case of overlapping clusters the user can perform a cluster selection and restrict the investigation to a single cluster.

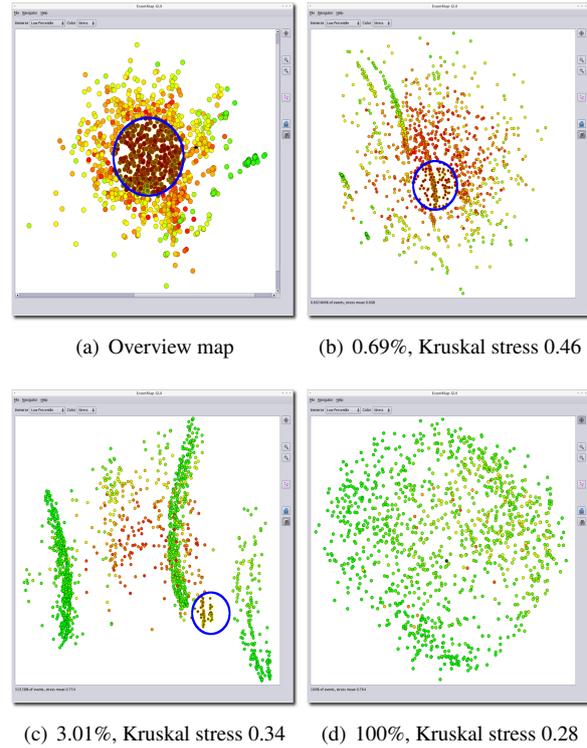


Figure 5: Refinement example. (a)-(d) show successive refinements of the selected regions. Stress is color coded from red (high) to green (low). The percentage gives the fraction of the visible subsample to all events inside region of interest.

Browsing Example

An example browsing workflow is given in Fig. 5, where three successive iterative refinements for a dataset containing about half a million events are shown. Starting with an overview map, which shows a subsample of 1500 events of the complete dataset, in each step a radial region (indicated by the blue circle) is refined. The computation time of a refined map is about 30 seconds for 1500 events on our test system, a 3GHz Pentium-4, and less than 10 seconds for 1000 events. As expected, the overview map exhibits a high stress (visible from the color coding) and no recognizable structure despite the green cluster to the right. But in the course of the following refinements, substructures are revealed in successively less distorted maps (as the Kruskal stress approves). Additionally, since the investigated region in event space gets narrower and narrower, the fraction of visible events increases until potentially all events of the respective region are shown after the last refinement step.

7 CONCLUSION

Based on navigation techniques known from the context of image databases, we have developed an approach for the interactive exploration of large HEP event datasets. A central contribution of this approach is a new criterion for transferring a selected region on the map to a

selection of events in the non-Euclidean, metric space. The proposed technique takes the local stress in the map projection into account and is robust to outliers. This makes the iterative refinement process more intuitive and better controllable for the user compared to previous navigation approaches.

To make the interactive navigation feasible for large datasets, we subsample the event set corresponding to the region of interest in order to obtain a representative subset consisting of not more than a certain fixed number of events before calculating its Euclidean embedding. This is also taken into account when calculating a refined map in such a way that all events in the event space are considered and not just the subsample represented on the map where the region was selected by the user. In addition, we integrated a second navigation technique, namely cluster selection, into our approach to further improve the navigation process.

The practical usability of the proposed approach was verified by applying it to real large-scale datasets from the COMPASS experiment.

It seems also important to note that our improvements of the navigation process in comparison to the recent work are independent of the transferring of these techniques to the domain of HEP event datasets. Therefore, as future work we would like to evaluate these improvements also in context of other large-scale datasets.

REFERENCES

- [1] P. Abbon et al. The COMPASS experiment at CERN. *Nuclear Instruments and Methods*, A577:455–518, 2007.
- [2] G. Barrand. Panoramix. In *Proc. of Computing in High Energy Physics (CHEP'04)*, 2004.
- [3] I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 1997.
- [4] CMS Collaboration. *The CMS Offline WorkBook, Chapter 4.8 The CMS Event Display*. Available online at <https://twiki.cern.ch/twiki/bin/view/CMS/WorkBook>.
- [5] P. Deuffhard and A. Hohmann. *Numerical Analysis in Modern Scientific Computing: An Introduction*. Texts in Applied Mathematics 43. Springer, 2003.
- [6] H. Drevermann, D. Kuhn, and B. Nilsson. Event display: Can we see what we want to see? In *CERN School of Computing*, 1995.
- [7] J. Drohan et al. The ATLANTIS visualisation program for the ATLAS experiment. In *Proc. of Computing in High Energy Physics (CHEP'04)*, pages 361–364, 2004.
- [8] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1216–1223, 2007.
- [9] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *J. Math. Phys.*, 20:224–230, 1941.
- [10] T. Kaijser. Computing the Kantorovich distance for images. *J. Math. Imaging and Vision*, 9(2):173–191, 1998.
- [11] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.
- [12] O. Kind, J. Rautenberg, et al. A ROOT-based client-server event display for the ZEUS experiment. In *Proc. of Computing in High Energy Physics (CHEP'03)*, 2003.
- [13] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29:115–129, 1964.
- [14] E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: Some insights from statistics. In *Proc. of the IEEE International Conference on Computer Vision (ICCV'01)*, 2001.
- [15] H. Ling and K. Okada. EMD- L_1 : An efficient and robust algorithm for comparing histogram-based descriptors. In *Proc. of the 9th European Conference on Computer Vision (ECCV'06)*, pages 330–343, 2006.
- [16] Z. Maxa et al. Event visualization for the ATLAS experiment - the technologies involved. In *Proc. of Computing in High Energy Physics (CHEP'06)*, 2006.
- [17] D. McNally. Event visualization tools at LEP. In *Proc. of the HEPVis 96 Workshop*, CERN, 1996.
- [18] I. Osborne et al. CMS event display and data quality monitoring at LHC start-up. In *Proc. of Computing in High Energy Physics (CHEP'07)*, 2007.
- [19] S. Peleg, M. Werman, and H. Rom. A unified approach to the change of resolution: Space and grey-level. *IEEE Trans. Pattern Anal. and Machine Intelligence*, 11(7):739–742, 1989.
- [20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1997.
- [21] S. T. Rachev and L. Rüschendorf. *Mass Transportation Problems*, volume 1: Theory. Springer, 1998.
- [22] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proc. of the 6th International Conference on Computer Vision (ICCV'98)*, pages 59–66, 1998.
- [23] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Intern. J. Comp. Vision*, 40(2):99–121, 2000.
- [24] M. Tadel and A. Mrak-Tadel. AliEVE - ALICE event visualization environment. In *Proc. of Computing in High Energy Physics (CHEP'06)*, 2006.
- [25] W. S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17:401–419, 1952.
- [26] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3):645–678, 2005.
- [27] J. Yang, J. Fan, D. Hubball, Y. Gao, H. Luo, W. Ribarsky, and M. Ward. Semantic image browser: Bridging information visualization with automated intelligent image analysis. In *Proc. of IEEE Symposium on Visual Analytics Science and Technology*, pages 191–198, 2006.
- [28] T. Yang, J. Liu, L. McMillan, and W. Wang. A fast approximation to multidimensional scaling. In *Proc. of the IEEE Workshop on Computation Intensive Methods for Computer Vision*, 2006.

Prefiltered Gradient Reconstruction for Volume Rendering

Balázs Csébfalvi, Balázs Domonkos

Department of Control Engineering and Information Technology

Budapest University of Technology and Economics

Magyar tudósok krt. 2, Budapest, Hungary, H-1117

E-mail: cseb@iit.bme.hu

ABSTRACT

The quality of images generated by volume rendering strongly depends on the applied continuous reconstruction method. Recently, it has been shown that the reconstruction of the underlying function can be improved by a discrete prefiltering. In volume rendering, however, an accurate gradient reconstruction also plays an important role as it provides the surface normals for the shading computations. Therefore, in this paper, we propose prefiltering schemes in order to increase the accuracy of the estimated gradients yielding higher image quality. We search for discrete prefilters of minimal support which can be efficiently used in a preprocessing as well as on the fly.

Keywords: Volume Rendering, Filtering, Reconstruction.

1 INTRODUCTION

An accurate reconstruction of a continuous function from its evenly located discrete samples is an important issue in many computer graphics applications. Although the reconstruction is usually performed as a convolution-based filtering, it is often not obvious which filter to use for a specific data or re-sampling task. Generally, an appropriate filter is chosen by making a compromise between quality and efficiency. The efficiency directly depends on the support of the given filter, whereas the quality can be analyzed from different aspects.

According to the signal-processing theory, the *sinc* filter is considered to be ideal as it can perfectly reconstruct a band-limited signal sampled above the Nyquist limit [19]. Nevertheless, the *sinc* filter is impractical since it has an infinite support. Although there exist frequency-domain techniques for ideal reconstruction [6, 8, 24, 25, 1, 10], all of them are global methods mainly used for resampling the original discrete representation on a transformed grid. On the other hand, they do not support efficient local resampling. Therefore, in practical applications requiring fast local resampling, the *sinc* filter is either approximated by a filter of finite support or truncated by an appropriate windowing function [14, 22] and the convolution is performed in the spatial domain.

The quality of the reconstruction is mainly influenced by the global frequency-domain behavior of the applied filter, especially if the original signal is sampled near the Nyquist limit. Therefore a filter is usually characterized by a frequency plot and its quality is quantitatively evaluated as the deviation from the ideal pass-band and stop-band behavior [14, 2]. The main drawback of this approach is that practical

signals can hardly be considered band-limited. Thus even the *sinc* filter produces ringing artifacts due to the drastic cut-off in the frequency domain [1].

A reconstruction filter can also be analyzed based on the approximation theory. Here the major aspect is how fast the approximate signal converges to the original one by decreasing the sampling distance. This mainly depends on the approximation power of the reconstruction filter. The order of approximation is L if the frequency response equals to zero at the centers of all the aliasing spectra with a multiplicity of L [21]. However, in order to fully exploit the approximation power of a given filter, usually an appropriate discrete prefiltering is necessary (see Figure 1). Such a prefiltering can ensure that a polynomial of $L - 1$ or lower degree is perfectly reconstructed. If this condition is satisfied, the reconstruction scheme is quasi-interpolating of order L [7].

The prefiltering can improve the reconstruction from other aspects as well. For example, depending on the applied discrete prefilter, it can optimize the pass-band behavior of the reconstruction [13], make a non-interpolating continuous postfilter interpolating [3, 23, 4], or increase the accuracy of the reconstruction in a sense of minimal approximation error [7]. All these prefiltering techniques are of infinite impulse response (IIR) and proven to yield k -EF (error function of order k) reconstruction if the approximation order of the continuous postfilter is $L = k$ [9]. This implies that the result is a quasi-interpolation of order k . Practically, the order of accuracy becomes important if the original signal is at least relatively oversampled, that is, most of its energy is concentrated around the origin in the frequency domain and the overlapping between the replicas is minimal. This assumption is valid for medical volume rendering as the resolution of CT and MRI scanners has been significantly increased in the last two decades.

In this paper, we demonstrate that prefiltering can be optimized also for a gradient estimation of higher accuracy. In volume-rendering applications the following gradient-estimation scenarios can be distinguished:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

1. **Precalculated gradients for semitransparent volume rendering:** The gradients are precalculated for each voxel position. At an arbitrary sample location along a viewing ray the gradient is interpolated from the precalculated ones. Although this approach requires additional memory for storing the gradient components, the SIMD instructions of either the CPU or the GPU can be well exploited for interpolating the components in one step.
2. **On-the-fly gradient estimation for semitransparent volume rendering:** The gradients are calculated on the fly for each sample location along a viewing ray by applying a derivative filter. Although this approach does not require additional memory, the gradient estimation is more expensive computationally as the components are calculated separately.
3. **First-hit ray casting:** Rays are cast into the volume and the first intersection points between the rays and a predefined isosurface are determined. At each intersection point a gradient calculation is performed. The cost of the gradient calculation, which is proportional to the number of pixels, is negligible compared to that of the ray casting.

To support not just the first scenario, but the second and the third ones as well, we search for discrete prefilters of finite impulse response (FIR). Furthermore, we try to find filters of minimal support which maximize the order of accuracy. Such filters can be efficiently evaluated either in a preprocessing or on the fly, and unlike the IIR filters, do not lead to unexpected boundary effects.

Möller et al. classified the normal estimation schemes [16] as follows (\mathcal{F} denotes the original discrete function, whereas symbols \mathcal{D} and \mathcal{H} denote the derivative and interpolation operators respectively):

1. $(\mathcal{F}\mathcal{D})\mathcal{H}$ **Derivative first:** The derivatives are first calculated for the discrete samples, and interpolated afterwards. This scheme fits to the first scenario.
2. $(\mathcal{F}\mathcal{H})\mathcal{D}$ **Interpolation first:** The derivative operator is applied on the reconstructed function. This scheme fits onto the second and third scenarios.
3. $\mathcal{F}(\mathcal{D}\mathcal{H})$ **Continuous derivative:** A continuous derivative filter is constructed by applying the derivative operator on the interpolation operator. This scheme is rather theoretical and equivalent to the first two schemes.
4. $\mathcal{F}\mathcal{H}'$ **Analytical derivative:** The analytical derivative of the interpolation operator is used for calculating the gradient components. This scheme fits onto the second and third scenarios.

In practical volume-rendering applications, usually the trilinear interpolation and the central differences are used as the interpolation and derivative operators respectively, since these operators can be efficiently evaluated on the GPU. Sigg and Hadwiger [20], however, demonstrated that current GPUs can provide interactive frame rates even if tricubic filtering is applied for resampling. They efficiently implemented the tricubic B-spline filtering in the fragment shader using the analytical derivative filter for the gradient estimation. Nevertheless, as it will be shown in this paper, nor the central differences neither the analytical derivative filter can fully utilize the higher approximation power of the tricubic B-spline.

Therefore, we propose to use either a slightly more expensive discrete derivative filter instead of the central differences or to use the analytical derivative filter on prefiltered data rather than on the original data.

2 SPATIAL-DOMAIN FILTER DESIGN

In order to increase the accuracy of gradient estimation, we slightly modify the framework of Möller et al. [17], which is briefly summarized here.

The reconstruction of a continuous function $f(t)$ from its known samples f_k is formulated as a convolution with the impulse response $w(t)$ of the applied filter:

$$f(t) \approx \tilde{f}(t) = \sum_{k=-\infty}^{\infty} f_k \cdot w\left(\frac{t}{T} - k\right), \quad (1)$$

where T is the sampling distance. By the Taylor series expansion of $f_k = f(kT)$ about t , we obtain:

$$f_k = \sum_{n=0}^N \frac{f^{(n)}(t)}{n!} (kT - t)^n + \frac{f^{(N+1)}(\xi_k)}{(N+1)!} (kT - t)^{(N+1)}, \quad (2)$$

where $f^{(n)}(t)$ is the n th derivative of $f(t)$ and $\xi_k \in [t, kT]$. Substituting the Taylor series expansion into the convolution sum in Equation 1, leads to an alternative representation for the reconstructed value at point t :

$$\tilde{f}(t) = \sum_{n=0}^N a_n^w(\tau) f^{(n)}(t) + r_N^w(\tau), \quad (3)$$

$$a_n^w(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k - \tau)^n w(\tau - k),$$

$$r_N^w(\tau) \leq \left(\max_{\xi \in \mathbf{R}} (f^{(N+1)}(\xi)) \right) |a_{N+1}^w(\tau)|,$$

$$\text{or } r_N^w(\tau) \approx a_{N+1}^w(\tau) f^{(N+1)}(t),$$

where τ is chosen such that $t = (i + \tau)T$, with $0 \leq \tau < 1$ and $i \in \mathbf{Z}$. The error coefficients a only depend on the offset τ to the nearest sampling point, that is, they are periodic in the sampling distance T . Additionally, they characterize the asymptotic error behavior of the given filter for decreasing sampling distance T . Assume that N is the largest number such that $a_n = 0$ for $0 < n \leq N$. In this case, the error function is of order $O(T^{N+1})$, and the reconstruction filter is classified as k -EF, where $k = N + 1$. Such a filter can perfectly reconstruct a polynomial of N th or lower degree, or in other words, it is quasi-interpolating of order k .

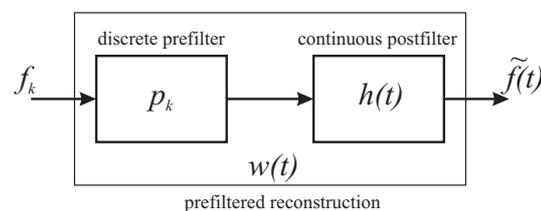


Figure 1: Prefiltered reconstruction: Input samples f_k are first convolved with a discrete prefilter p_k and afterwards with a continuous postfilter $h(t)$. The resultant impulse response is denoted by $w(t)$.

Using the filter design approach of Möller et al. [18], the parameters of piecewise polynomial filters are determined by

solving a linear equation system such that the required accuracy and smoothness criteria are satisfied. In this original framework the option of prefiltering has not been considered. However, it can be exploited that the order of the error function belonging to a prefiltered reconstruction can potentially be higher than that of the non-prefiltered one. Using prefiltered reconstruction, the original data is first convolved with a discrete prefilter p_k and afterwards with a continuous postfilter $h(t)$ (see Figure 1). Therefore, the resultant impulse response $w(t)$ is the convolution of p with $h(t)$:

$$w(t) = \sum_{k=-\infty}^{\infty} p_k \cdot h(t-k). \quad (4)$$

The error coefficients for the prefiltered reconstruction can be derived as follows:

$$\begin{aligned} a_n^{ph}(\tau) &= \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n \left(\sum_{l=-\infty}^{\infty} p_l \cdot h(\tau-k-l) \right) \\ &= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} p_l \cdot \left(\sum_{k=-\infty}^{\infty} (k-\tau)^n h(\tau-k-l) \right). \end{aligned} \quad (5)$$

Substituting m for $k+l$ in the inner sum, we get (note that the sums are just formally infinite, as the filters p and h are assumed to be FIR filters):

$$\begin{aligned} a_n^{ph}(\tau) &= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} p_l \cdot \left(\sum_{m=-\infty}^{\infty} (m-\tau-l)^n h(\tau-m) \right) \\ &= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} p_l \left[\sum_{m=-\infty}^{\infty} \left(\sum_{i=0}^n \binom{n}{i} (m-\tau)^i (-l)^{n-i} \right) h(\tau-m) \right], \end{aligned}$$

which resolves to

$$\begin{aligned} a_n^{ph}(\tau) &= \frac{T^n}{n!} \sum_{i=0}^n \binom{n}{i} \left(\sum_{l=-\infty}^{\infty} (-l)^{n-i} p_l \right) \left(\sum_{m=-\infty}^{\infty} (m-\tau)^i h(\tau-m) \right) \\ &= \sum_{i=0}^n a_{n-i}^p(0) a_i^h(\tau). \end{aligned} \quad (6)$$

Thus an error coefficient of the prefiltered reconstruction is simply the convolution of the error coefficients belonging to the discrete prefilter p and the continuous postfilter h . This derivation was originally published by Möller et al. [16] but in a different context, analyzing the numerical accuracy of the normal estimation scheme $\mathcal{F}(\mathcal{D}\mathcal{H})$. In this scheme, the discrete prefilter p and the continuous filter h play the roles of the derivative operator \mathcal{D} and the interpolation operator \mathcal{H} respectively.

In contrast, we use the discrete prefiltering in a more general manner. To improve the accuracy of both function and derivative reconstruction, we apply different prefilters combined with either the continuous postfilter h or its analytical derivative h' . In the following sections we illustrate our filter design approach with a concrete example, where h is the cubic B-spline defined as follows:

$$\beta^3(t) = \begin{cases} \frac{1}{2}|t|^3 - |t|^2 + \frac{2}{3} & \text{if } |t| \leq 1 \\ -\frac{1}{6}|t|^3 + |t|^2 - 2|t| + \frac{4}{3} & \text{if } 1 < |t| \leq 2 \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

The cubic B-spline has several advantageous properties. For example, it provides an approximation order $L = 4$ with a

minimal support, and yields a C^2 continuous reconstruction. Furthermore, its 3D tensor-product extension can be efficiently evaluated on the recent graphics cards by only eight trilinear texture fetches per sample [20].

3 PREFILTERED FUNCTION RECONSTRUCTION

It is easy to verify that the error coefficients of the cubic B-spline are the following ¹:

$$\begin{aligned} a_0^{\beta^3}(\tau) &= 1, \quad a_1^{\beta^3}(\tau) = 0, \\ a_2^{\beta^3}(\tau) &= \frac{T^2}{6}, \quad a_3^{\beta^3}(\tau) = 0. \end{aligned} \quad (8)$$

Since $a_2^{\beta^3}$ is non-zero, the cubic B-spline is a 2-EF filter. Nevertheless, its approximation order is four, which can be exploited by a discrete prefiltering.

Let us assume that the discrete prefilter p has only three non-zero weights, which are p_{-1}, p_0, p_1 at grid points $-T, 0,$ and T respectively. Additionally, we search for a symmetric filter, that is, $p_{-1} = p_1$. Thus there are just two free parameters to be determined. The error coefficients for the prefilter p are the following:

$$\begin{aligned} a_0^p(0) &= p_0 + 2p_1, \quad a_1^p(0) = 0, \\ a_2^p(0) &= T^2 p_1, \quad a_3^p(0) = 0. \end{aligned} \quad (9)$$

The error coefficients for the prefiltered reconstruction can be evaluated according to Equation 6:

$$\begin{aligned} a_0^{p\beta^3}(\tau) &= p_0 + 2p_1, \quad a_1^{p\beta^3}(\tau) = 0, \\ a_2^{p\beta^3}(\tau) &= T^2 \frac{p_0 + 8p_1}{6}, \quad a_3^{p\beta^3}(\tau) = 0. \end{aligned} \quad (10)$$

To guarantee a 4-EF reconstruction, $a_0^{p\beta^3}$ has to be equal to one, while coefficient $a_2^{p\beta^3}$ has to be equal to zero. Solving Equation 10 with these constraints, the following filter weights are obtained: $p_{-1} = p_1 = -\frac{1}{6}, p_0 = \frac{8}{6}$ (see Figure 2). Note that, the resultant impulse response $w = p * h$ is exactly the same as that of the C^2 4-EF reconstruction filter designed in [18]. However, there is a significant difference in the computational costs. In our case, the discrete prefiltering with p is performed in a preprocessing, while the continuous postfiltering with h is evaluated on the fly from the nearest 64 voxels. In contrast, a direct convolution with w would take the nearest 216 voxels into account. The discrete prefilter p can also be obtained by a different derivation proposed by Blu and Unser [5], therefore we do not consider it as a new result.

4 PREFILTERED DERIVATIVE RECONSTRUCTION

In this section we show that using a simple discrete prefilter not just the accuracy of function reconstruction can be improved, but the accuracy of normal estimation as well. In previous volume-rendering applications, when the cubic B-spline is used for function reconstruction, the derivatives are

¹ The cubic B-spline is the special case of the BC-splines [15]. The asymptotic error behavior of this general family of cubic filters has been analyzed in detail by Möller et al. [17].

computed by either the central differences or taking the analytical derivative of the cubic B-spline as a continuous derivative filter [16, 20, 12, 11]. These techniques, however, do not exploit the approximation power of the cubic B-spline.

The calculation of central differences on the reconstructed function is equivalent to a filtering by a discrete derivative filter c , where the non-zero weights are $c_{-1} = \frac{1}{2T}$ and $c_1 = -\frac{1}{2T}$ at positions $-T$ and T respectively. The error coefficients for this discrete derivative filter are the following:

$$\begin{aligned} a_0^c(0) &= 0, a_1^c(0) = 1, \\ a_2^c(0) &= 0, a_3^c(0) = \frac{T^2}{6}. \end{aligned} \quad (11)$$

If the central differences are calculated on a function reconstructed by the cubic B-spline, it is equivalent by a filtering with a continuous derivative filter $c * \beta^3$. According to Equation 6, the corresponding error coefficients are obtained as:

$$\begin{aligned} a_0^{c\beta^3}(\tau) &= 0, a_1^{c\beta^3}(\tau) = 1, \\ a_2^{c\beta^3}(\tau) &= 0, a_3^{c\beta^3}(\tau) = \frac{T^2}{3}. \end{aligned} \quad (12)$$

Thus the central differences combined with the cubic B-spline yield just a 2-EF derivative filtering. This order of accuracy is not improved even if the analytical derivative of the cubic B-spline is used, which also leads to a 2-EF derivative filtering [18].

One possibility for increasing the accuracy of the derivative filtering is to apply the analytical derivative of the cubic B-spline on data prefiltered by the discrete filter p . The error coefficients corresponding to the analytical derivative filter β^{3r} are as follows [17]:

$$\begin{aligned} a_0^{\beta^{3r}}(\tau) &= 0, a_1^{\beta^{3r}}(\tau) = T, \\ a_2^{\beta^{3r}}(\tau) &= 0, a_3^{\beta^{3r}}(\tau) = \frac{T^3}{6}, \\ a_4^{\beta^{3r}}(\tau) &= \frac{T^4}{12} \tau(1-\tau)(2\tau-1). \end{aligned} \quad (13)$$

If β^{3r} is combined with the discrete filter p , the error coefficients are obtained from Equation 6:

$$\begin{aligned} a_0^{p\beta^{3r}}(\tau) &= 0, a_1^{p\beta^{3r}}(\tau) = T, \\ a_2^{p\beta^{3r}}(\tau) &= 0, a_3^{p\beta^{3r}}(\tau) = 0, \\ a_4^{p\beta^{3r}}(\tau) &= \frac{T^4}{12} \tau(1-\tau)(2\tau-1). \end{aligned} \quad (14)$$

Thus, after the normalization by T , the combination of β^{3r} and p results in a 3-EF derivative filtering.

In order to fully exploit the approximation power of the cubic B-spline, we search for a discrete prefilter d with a support of 2, where the non-zero weights are $d_{-2} = -d_2$ and $d_{-1} = -d_1$. The error coefficients for this discrete derivative filter are the following:

$$\begin{aligned} a_0^d(0) &= 0, a_1^d(0) = -2T(d_1 + 2d_2), \\ a_2^d(0) &= 0, a_3^d(0) = -T^3 \frac{d_1 + 8d_2}{3}. \end{aligned} \quad (15)$$

If the discrete derivative filter d is combined with the cubic B-spline then the error coefficients of the equivalent continuous filtering are obtained as (see Equation 6):

$$\begin{aligned} a_0^{d\beta^3}(\tau) &= 0, a_1^{d\beta^3}(\tau) = -2T(d_1 + 2d_2), \\ a_2^{d\beta^3}(\tau) &= 0, a_3^{d\beta^3}(\tau) = -2T^3 \frac{d_1 + 5d_2}{3}. \end{aligned} \quad (16)$$

To reconstruct the first derivative instead of some multiple of it, the error coefficient $a_1^{d\beta^3}(\tau)$ has to be equal to one. Additionally, to maximize the order of accuracy, the error coefficient $a_3^{d\beta^3}(\tau)$ has to be equal to zero. By solving Equation 16 for these constraints we obtain: $d_1 = -\frac{5}{6T}$ and $d_2 = \frac{1}{6T}$ (see Figure 2). It is easy to see that the error coefficient $a_4^{d\beta^3}(\tau)$ is also zero for the combined filter $d * \beta^3$. The error coefficient $a_5^{d\beta^3}(\tau)$, however, is clearly non-zero. Therefore the error function contains at least fourth-degree powers of T due to the normalization. As a consequence, the combined filter is a 4-EF derivative filter.

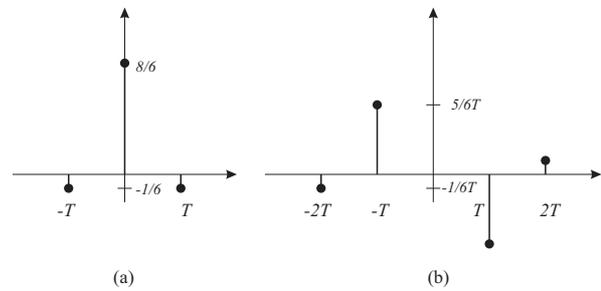


Figure 2: Discrete prefilters for a cubic B-spline reconstruction. (a): Prefilter p for 4-EF function reconstruction. (b): Prefilter d for 4-EF derivative reconstruction.

5 FREQUENCY-DOMAIN ANALYSIS

The cubic B-spline can be obtained by successively convolving a symmetric box filter (the B-spline of order zero) three times with itself. Since the Fourier transform of the symmetric box filter is $\text{sinc}(\omega/2) = \sin(\omega/2)/(\omega/2)$ and the consecutive convolutions in the spatial domain correspond to consecutive multiplications in the frequency domain, the Fourier transform of the cubic B-spline is $\text{sinc}^4(\omega/2)$. When the cubic B-spline is combined with the discrete prefilter p , the frequency response of the equivalent filter $w(t)$ is $W(\omega) = \text{sinc}^4(\omega/2) \cdot P(\omega)$, where the Fourier transform of the prefilter p is $P(\omega) = (4 - \cos(\omega))/3$. Figure 5 shows that the combined filter represents a kind of compromise, as its pass-band behavior is better than that of the non-prefiltered cubic B-spline but worse than that of the Catmull-Rom spline. On the other hand, the Catmull-Rom spline improves the pass-band behavior at the cost of a much higher postaliasing.

The Fourier transform of our discrete derivative filter d is $D(\omega) = i(5\sin(\omega) - \sin(2\omega))/3$. Combining it with the cubic B-spline, the frequency response of the equivalent continuous derivative filter is $W(\omega) = \text{sinc}^4(\omega/2) \cdot D(\omega)$, which is shown in Figure 6. The derivative filter d ensures much better pass-band behavior than the central differences. Although the analytical derivative of the cubic B-spline performs even better in the pass-band, its postaliasing effect is significantly

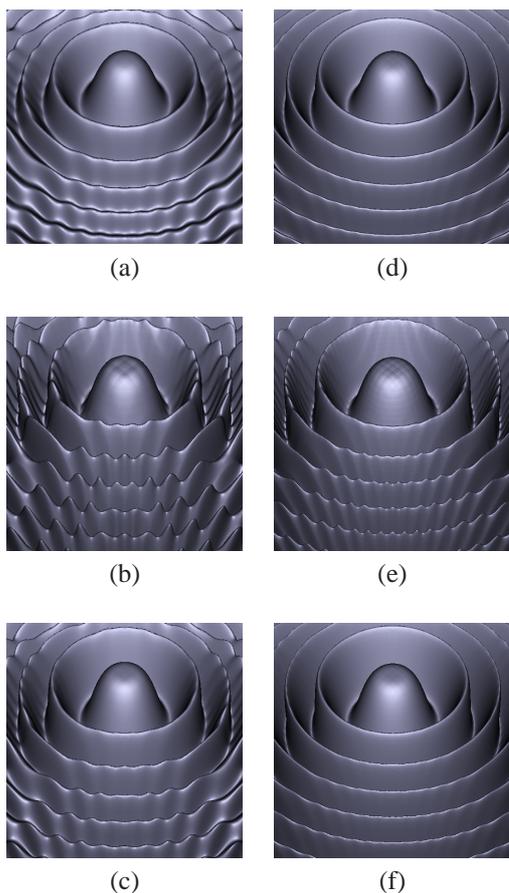


Figure 3: Reconstruction of the Marschner-Lobb signal from $40 \times 40 \times 40$ (a-c) and from $60 \times 60 \times 60$ (d-f) samples. (a, d): Cubic B-spline. (b, e): Catmull-Rom spline. (c, f): Cubic B-spline prefiltered by the discrete filter p . The isosurface is shaded based on the analytical gradient of the reconstructed function.

higher. The best pass-band behavior is achieved if the analytical derivative of the cubic B-spline is applied on data prefiltered by p (the frequency response of the equivalent continuous derivative filter is $W(\omega) = i\omega \text{sinc}^4(\omega/2) \cdot P(\omega)$), but this technique causes also the highest postaliasing.

6 EXPERIMENTAL EVALUATION

In order to empirically analyze our discrete prefilters, we implemented a high-quality ray caster and rendered artificial and real-world data sets. We used the classical Marschner-Lobb signal to separately evaluate the effects of the prefilter p and the discrete derivative filter d . Figure 3 shows the shaded isosurface of the test signal reconstructed by the cubic B-spline (a, d), the Catmull-Rom spline (b, e), and the cubic B-spline prefiltered by the discrete filter p (c, f). Here the gradients used for the shading computation are the exact analytical gradients of the reconstructed function. Note that the highest quality is ensured by the prefiltered cubic B-spline reconstruction even for the low-resolution volume representation, but its superiority is much more apparent if the resolution is increased by a factor of 1.5. Theoretically, the C^2 4-EF prefiltered cubic-B-spline is superior over the C^1 3-EF Catmull-

Rom spline considering the order of both continuity and accuracy. This is completely confirmed by our test results.

To fairly test our prefiltered derivative reconstruction schemes independently from the effect of the prefiltered function reconstruction, we calculated the exact intersection points between the rays and the original test signal, but at these hit points we evaluated the gradients using different derivative filters. Figure 4 shows the angular errors of the gradients reconstructed by the cubic B-spline combined with central differences (first column) and our discrete derivative filter d (second column). The third and fourth columns show the angular error for the analytical derivative of the cubic B-spline applied on non-prefiltered data, and data prefiltered by the discrete filter p respectively.

The worst results are obtained by using the central differences combined with the cubic B-spline. The angular error is significantly reduced if our discrete derivative filter d is applied instead of the central differences. It is interesting to note, that the analytical derivative filter performs even better for the lower-resolution volumes, although it provides slower convergence (2-EF) to the original signal if the resolution is increased. The best results, however, are achieved by the analytical derivative filter applied on data prefiltered by the discrete filter p .

Reconstruction and derivative filters that perform well for synthetic data might not provide good results for real-world measured data sets, which are usually corrupted by measurement and quantization noise. Therefore, we tested the prefiltered derivative filtering schemes also on CT and MRI data. The results are shown in Figure 7. The fine details are best captured if the underlying signal is reconstructed from data prefiltered by the discrete filter p . The benefit of prefiltering in terms of gradient accuracy, however, is not so obvious. For example, the analytical derivative of the cubic B-spline applied on prefiltered data even emphasizes the quantization noise, which leads to severe staircase artifacts. In contrast, our discrete derivative filter represents a good compromise. On one hand, it does not blur the gradients as much as the central differences, thus it preserves the contrast and sharpness of the contours. On the other hand, it does not introduce so strong staircase aliasing as the analytical derivative of the cubic B-spline applied on either prefiltered or non-prefiltered data.

7 EFFICIENCY CONSIDERATIONS

The evaluation of our discrete prefilter d is twice as expensive computationally as that of central differences. Therefore we propose using it mainly for the first and the third volume-rendering scenarios. In first-hit ray casting the cost of the gradient estimation is negligible compared to that of the ray casting, while in case of precalculated gradients the more expensive preprocessing is acceptable as it has to be performed just once. Nevertheless, using the derivative filter d for on-the-fly gradient computation significantly reduces the rendering performance.

Due to its good pass-band behavior, we propose to use the analytical derivative of the cubic B-spline combined with the discrete prefilter p especially for rendering synthetic data which is not corrupted by prealiasing or quantization noise. This gradient computation scheme can be efficiently applied also for the second volume-rendering scenario, where only the

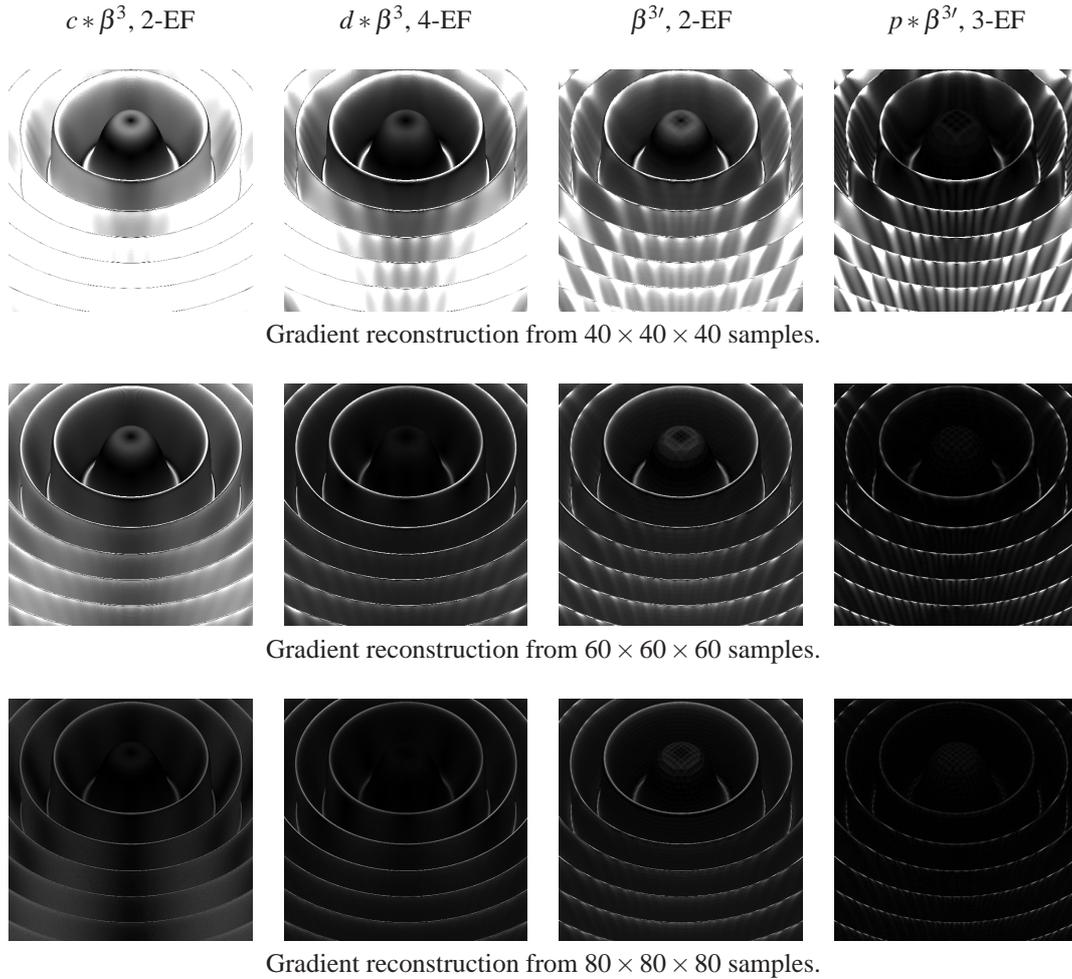


Figure 4: Angular errors of the gradients reconstructed from the Marschner-Lobb test data sets of different resolutions. Angular error of zero degree is mapped to black, whereas angular error of 30 degrees is mapped to white. First column: central differences combined with the cubic B-spline. Second column: cubic B-spline combined with our discrete derivative filter d . Third column: analytical derivative of the cubic B-spline. Fourth column: analytical derivative of the cubic B-spline combined with the discrete prefilter p .

prefiltered function values have to be stored without precalculated gradient components. The drawback of this approach is that the prefiltered data still requires at least a 16-bit floating-point number per voxel to store. Note that, the computational cost of the on-the-fly gradient computation is exactly the same as if the analytical derivative of the cubic B-spline was used on non-prefiltered data.

8 CONCLUSION

In this paper, different prefiltered gradient reconstruction schemes have been evaluated both in the spatial domain and in the frequency domain. We have shown that, applying a tricubic B-spline reconstruction filter, the accuracy of the gradients can be significantly increased if either our discrete derivative filter d is used instead of the central differences or the analytical derivative of the tricubic B-spline is used on data prefiltered by the discrete filter p . According to our experiments, the former approach is more appropriate for rendering real-world measured data sets, whereas the latter

approach performs better for synthetic data. It is interesting to note that filters which are theoretically more accurate do not necessarily provide the expected higher reconstruction quality in practice. The well-known spatial-domain and frequency-domain filter design criteria assume that the voxels represent accurate samples of the underlying signal and the sampling frequency is sufficiently high. These assumptions, however, are usually not valid for practical data. Therefore, in our future work, we plan to extend the classical filter design approach by practical criteria like sensitivity to the quantization noise.

ACKNOWLEDGEMENTS

This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, OTKA (F68945), the Hungarian National Office for Research and Technology, and Hewlett-Packard. The first author of this paper is a grantee of the János Bolyai Scholarship.

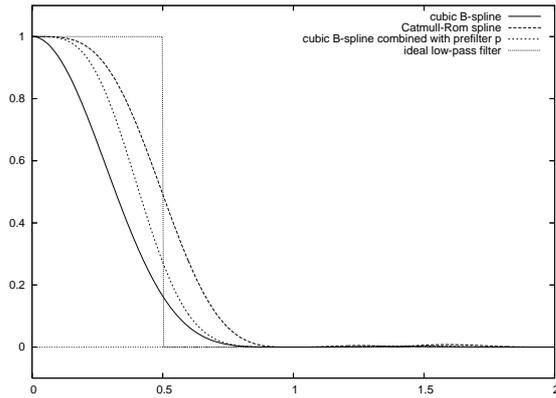


Figure 5: Comparison of the frequency response of the pre-filtered cubic B-spline reconstruction scheme to that of the non-pre-filtered Catmull-Rom and cubic B-spline reconstructions (the horizontal axis represents the ordinary frequency $v = \frac{\omega}{2\pi}$).

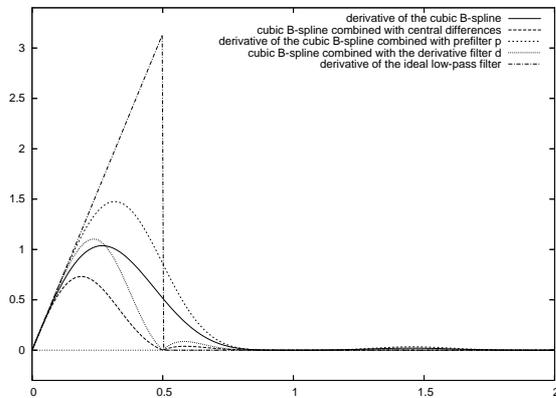


Figure 6: Comparison of the frequency response of our pre-filtered derivative reconstruction scheme to that of the analytical derivative of the cubic B-spline and the central differences combined with the cubic B-spline (the horizontal axis represents the ordinary frequency $v = \frac{\omega}{2\pi}$).

REFERENCES

- [1] M. Artner, T. Möller, I. Viola, and M. E. Gröller. High-quality volume rendering with resampling in the frequency domain. In *Proceedings of Joint EUROGRAPHICS-IEEE VGTC Symposium on Visualization (EuroVis)*, pages 85–92, 2005.
- [2] Mark J. Bentum, Barthold B. A. Lichtenbelt, and Tom Malzbender. Frequency analysis of gradient estimators in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, 1996.
- [3] T. Blu, P. Thévenaz, and M. Unser. Generalized interpolation: Higher quality at no additional cost. In *Proceedings of IEEE International Conference on Image Processing*, pages 667–671, 1999.
- [4] T. Blu, P. Thévenaz, and M. Unser. Linear interpolation revitalized. *IEEE Transactions on Image Processing*, 13(5):710–719, 2004.
- [5] T. Blu and M. Unser. Quantitative Fourier analysis of approximation techniques. *IEEE Transactions on Signal Processing*, 47(10):2783–2806, 1999.
- [6] Q. Chen, R. Crownover, and M. Weinhaus. Subunity coordinate translation with Fourier transform to achieve efficient and quality three-dimensional medical image interpolation. *Med. Phys.*, 26(9):1776–1782, 1999.
- [7] L. Condat, T. Blu, and M. Unser. Beyond interpolation: Optimal reconstruction by quasi-interpolation. In *Proceedings of International Conference on Image Processing (ICIP)*, pages 33–36, 2005.
- [8] R. Cox and R. Tong. Two- and three-dimensional image rotation using the FFT. *IEEE Transactions on Image Processing*, 8(9):1297–1299, 1999.
- [9] B. Csébfalvi. An evaluation of prefiltered reconstruction schemes for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):289–301, 2008.
- [10] A. Li, K. Mueller, and T. Ernst. Methods for efficient, high quality volume resampling in the frequency domain. In *Proceedings of IEEE Visualization*, pages 3–10, 2004.
- [11] S. Li and K. Mueller. Accelerated, high-quality refraction computations for volume graphics. In *Proceedings of Volume Graphics*, pages 73–81, 2005.
- [12] S. Li and K. Mueller. Spline-based gradient filters for high-quality refraction computations in discrete datasets. In *Proceedings of Joint EUROGRAPHICS-IEEE VGTC Symposium on Visualization (EuroVis)*, pages 217–222, 2005.
- [13] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, 1993.
- [14] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization*, pages 100–107, 1994.
- [15] D. Mitchell and A. Netravali. Reconstruction filters in computer graphics. In *Proceedings of SIGGRAPH*, pages 221–228, 1988.
- [16] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In *Proceedings of IEEE Visualization*, pages 19–26, 1997.
- [17] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a Taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997.
- [18] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 143–151, 1998.
- [19] A. V. Oppenheim and R. W. Schaffer. *Discrete-Time Signal Processing*. Prentice Hall Inc., Englewood Cliffs, 2nd edition, 1989.
- [20] C. Sigg and M. Hadwiger. Fast third-order texture filtering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pages 313–329. Matt Pharr (ed.), Addison-Wesley, 2005.
- [21] G. Strang and G. Fix. A Fourier analysis of the finite element variational method. In *Constructive Aspects of Functional Analysis*, pages 796–830, 1971.
- [22] T. Theußl, H. Hauser, and M. E. Gröller. Mastering windows: Improving reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 101–108, 2000.
- [23] P. Thévenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000.
- [24] R. Tong and R. Cox. Rotation of NMR images using the 2D chirp-z transform. *Magnetic Resonance in Medicine*, 41(2):253–256, 1999.
- [25] M. Unser, P. Thévenaz, and L. Yaroslavsky. Convolution-based interpolation for fast, high-quality rotation of images. *IEEE Transactions on Image Processing*, 4(10):1371–1381, 1995.

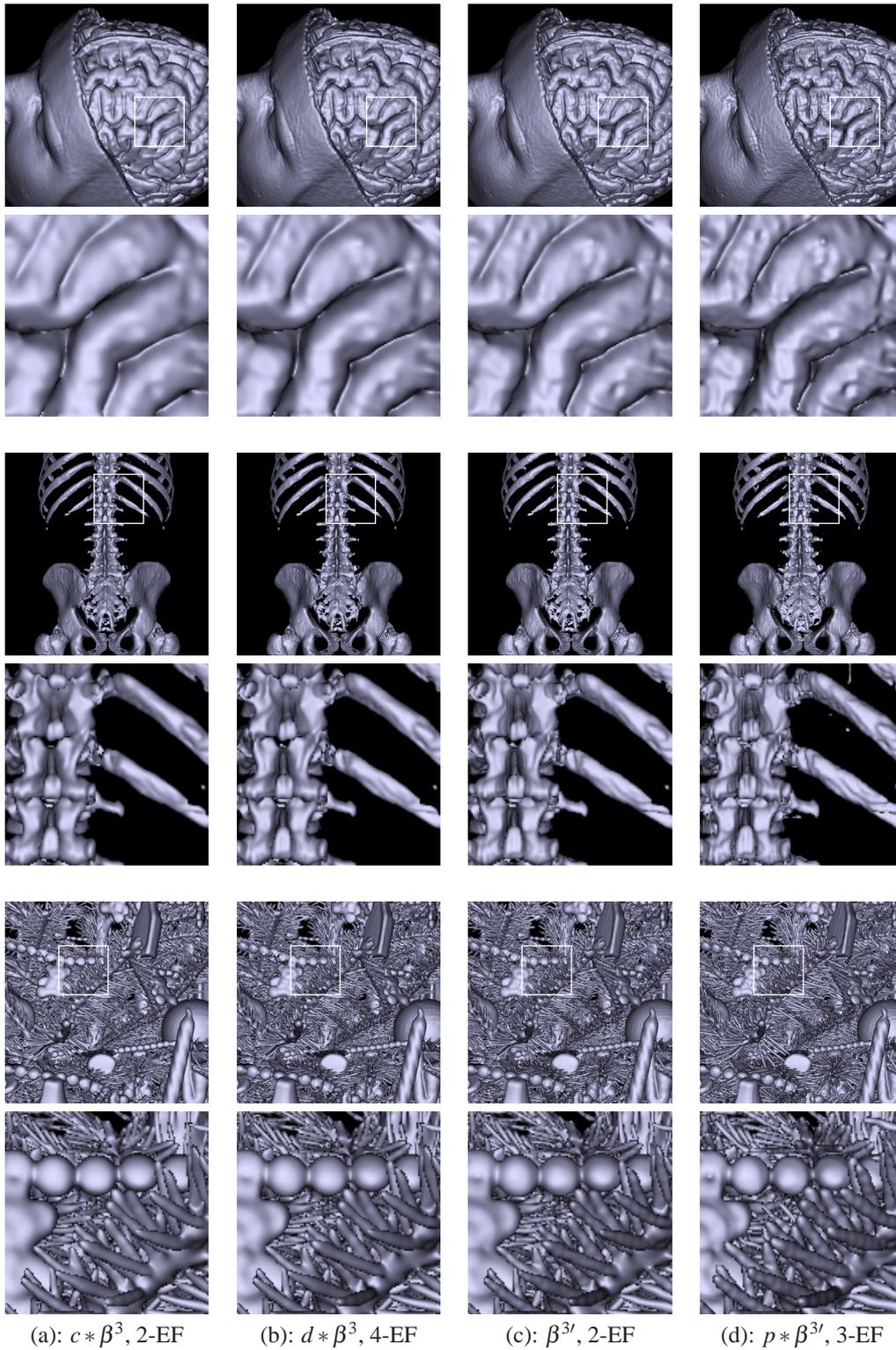


Figure 7: Real-world test data sets rendered by first-hit ray casting. The data is resampled along the rays by using the cubic B-spline filter to find the first intersection points ((a-c): The data is not prefiltered. (d): The data is prefiltered by p). The derivatives at these hit points are calculated by different gradient estimation schemes: (a): Central differences. (b): Our discrete derivative filter d . (c): Analytical derivative of the cubic B-spline. (d): Analytical derivative of the cubic B-spline applied on data prefiltered by p .

Hybrid sort-first/sort-last rendering for dense material particle systems

Simon Latapie

CEA
DAM, DIF

F-91297 Arpajon, France

Laboratoire MAS
Ecole Centrale Paris
Grande Voie des Vignes

92290, Châtenay-Malabry, France

simon.latapie@gmail.com

ABSTRACT

This paper describes a solution designed for efficient visualization of large and dense sets of particles, typically generated by molecular dynamics simulations in materials science. This solution is based on a hybrid distributed sort-first/sort-last architecture, and meant to work on a generic commodity cluster feeding a tiled display. The package relies on VTK framework with various extensions to achieve statistical occlusion culling, smart data partitioning and GPU-accelerated rendering.

Keywords

Hybrid sort-first/sort-last rendering, Statistical culling, VTK, Ice-T, Particle rendering, Dense particle system

1. INTRODUCTION

Materials science increasingly uses numerical simulations at different scales of space and time to better understand and predict the properties of matter. Molecular dynamics is one of the most widely used approaches in computational materials science. Thanks to the joint advances in parallel computing and in physics modeling, molecular dynamics can now be used to simulate systems with millions to hundreds of millions of particles[Stre 05]. We focus here on such simulations at nanoscopic to microscopic scales, which describe matter in dense states by large sets of particles.

Suitable and efficient visualization tools must be provided besides simulation codes in order to benefit from these very detailed computations, and especially interactive tools that help to explore complex 3D features such as blast waves, solidifications, dislocations, etc. We are going to describe how we built a distributed visualization tool to exploit a small graphics cluster and tiled display for almost interactive exploration of such datasets. The solution is quite standard since it relies on widely used software components, such as VTK[Schr 06], with various optimizations.

After reviewing related work which partly inspired us, we are going to describe the overall organization of the system, then give more details on some technical aspects of the algorithms we combined: culling by space partitioning and statistical occlusion, particle rendering and parallelization.

2. RELATED WORK

Only few existing solutions are specifically designed for visualization of global phenomena inside large particle sets on a tiled display.

Many systems are especially designed for biological molecular dynamics, like VMD[Hump 96] or Molekel[Fluk 00]. Some exhibit very advanced rendering techniques, via GPU programming, accelerating complex shape rendering like TexMol[Baja 04], or global illumination like QuteMol[Tari 06]. Such tools are generally optimized to represent domain specific features or sub-structures with non-spherical shapes, like ribbons, tubes, or molecular surfaces. These representations cannot be used in materials physics where there are no such apparent structures as proteins parts, or identified zones like in Terascale Particle Visualization[Ells 04].

Our application domain requires to focus on efficient raw rendering of particles, basically represented as colored opaque spheres. Opaque sphere representation is very important because on dense particles systems from materials science it can preserve graphical aspects of several structure properties, such as some surfaces granularities, which are lost with non-opaque, point-only

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

represented particles, or volume rendering solutions, such as in Liang et al.[Lian 05] solution.

Moreover, most of the aforementioned tools cannot be easily integrated in a distributed rendering architecture. Another example in another application domain is Kruger et al.[Krug 05] solution, a very efficient particle system rendering to visualize 3D flow fields. Such a solution takes advantage of new GPUs rendering capabilities, but as all data is stored in graphic card memory, scalability and possible extensions to a distributed architecture are compromised.

Very few solutions are scalable and designed to display raw real sphere representation of large sets of particles on large definition displays such as tiled displays. Atomviewer[Shar 03] is one of them: it uses efficient optimizations for sort-first rendering of large sets of particles: Z-order data organization, octree space partitioning, probabilistic culling method. However, Atomviewer has been adjusted to a specific hardware and display configuration (ImmersaDeskTM)[Shar 02b].

All these observations have lead us to work on the integration of culling methods and hardware-accelerated rendering in a generic distributed architecture.

3. GENERAL OVERVIEW

The main objective of our architecture is to provide new optimizations while re-using VTK/Ice-T[More 01]. Ice-T is a sort-last rendering solution for tiled displays, which has been proven[More 03] to be more efficient than generic solutions for tiled displays such as Chromium[Hump 02]. Sort-last rendering is scalable with respect to the size of the data, but the known bottleneck of such an approach is the network bandwidth. Ice-T brings improvements to usual sort-last rendering, such as an efficient distribution of images to be composed, or a floating viewport technique. Nethertheless, network bandwidth remains the bottleneck of such a method. Our strategy is to achieve distributed sort-first operations to increase spatial coherence of per process data to reduce network bandwidth usage, and lower the number of spheres to be displayed, to reduce the actual sphere rendering stage. This can be considered to be an attempt to transpose Samanta et al.[Sama 00] hybrid architecture to non-structured particle systems visualization.

Before data is to be displayed, we generate a space partitioning Kd-Tree, and we reorganize the dataset to gather all data attached to a same leaf of the tree. Each node of the tree contains its bounds, its data storage location, a link to its children if it has any, and a density factor, which is described in section 4.

The global architecture of our system, described in Figure 1, is meant to run on a cluster of N nodes, with P of them actually connected to a display with P physical or logical tiles. Each node runs a MPI process. Each process has a complete copy of the tree, but only part

of the data, in memory. During a frame rendering, each process computes frustum and occlusion culling for its own part of the tree, then all processes merge their results to have a complete up-to-date tree. At that time all processes know which data is really to be displayed. Then they compute a balanced per process redistribution of the data, load missing data and unload useless data if necessary, and render them. For the sort-last part of the architecture, Ice-T library performs a distributed rendering, and displays the frame.

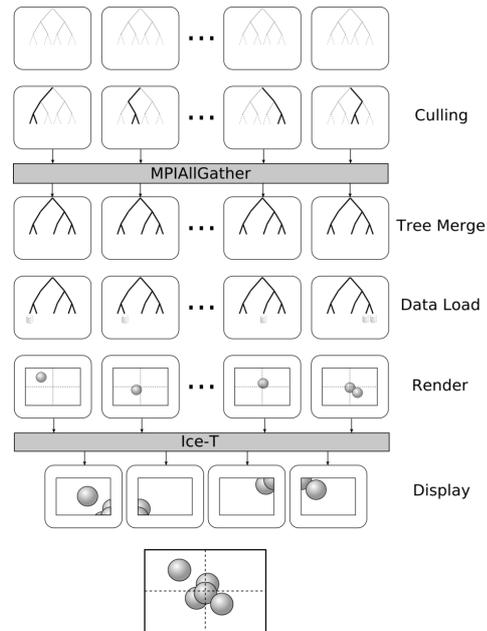


Figure 1: Global Architecture.

In the following sections, we are going to describe the tree for space decomposition and the culling algorithms we designed to organize datasets and optimize Ice-T rendering process. Then we will mention the GPU optimizations for the sphere rendering stage, the parallelization strategies, and finally we will highlight the implementation and a few results.

4. CULLING IN DENSE MATERIAL PARTICLE SYSTEMS

Although Atomviewer is based on a specific architecture, some of the pre-rendering techniques it uses are very effective, such as the idea of probabilistic occlusion culling[Shar 02a]. This approach introduces a notion of density of particles in the cells of an octree, which is used to randomly drop part of the particles displayed.

Dense material systems are often composed of large groups of particles which are very close to each other. The probabilistic occlusion can be pre-computed, as our datasets are non-interactively generated. Atomviewer's density computation, which is the volume

of particles divided by the volume of the cell, is not a very good factor to use for culling, because it assumes the distribution of particles is uniform in all cells.

In this section, we propose a complete pre-processing solution adapted to dense material systems visualization, which consists in a space partitioning algorithm, and a cell density computation. We also propose a culling algorithm which takes the pre-processing specificities into account.

Space partitioning

We use a Kd-Tree structure for space partitioning, which aims at separating dense space and empty space, in a very quick and simple way. We do not use the Kd-Tree VTK implementation, because of specific tree parsing methods and data order tests needs.

4.1.1 Empty space detection Kd-Tree

The algorithm described below is a simple way of generating a Kd-Tree for empty space detection.

As seen in Figure 2, for each tree node, we check particles positions for a given axis. If there is a left or right space between particles and node bounds, an empty space isolating split is done. Otherwise, a middle split is performed. Then the particles are sorted by comparing their axis position with the split position. Axis is alternatively x , y , then z .

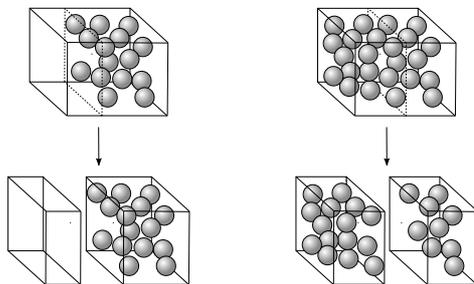


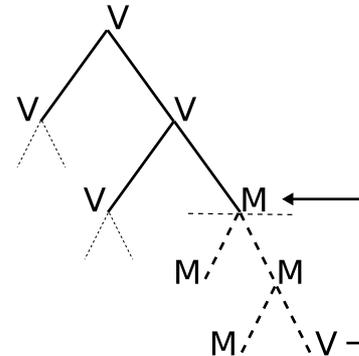
Figure 2: Kd-Tree generation: empty space partition

The complexity of the generation is $O(n \ln(n))$, where n is the number of particles, as the most consuming part is the particle sort, which is very much like a global quicksort, the split position being a pivot value. Optimum depth of the tree is very dependent on particles repartition in the scene, but empirical tests shows a 15 depth is a good overall value.

4.1.2 Kd-Tree optimization

The main goal of this space partition is to find dense cells which can occlude other cells. So we want the dense cells to be as large as possible. This is why we can optimize the Kd-Tree generation by moving up the effective splits, which are the empty space ones. Density computation is explained in section 4.2.

Figure 3 explains such an optimization: when a middle split is performed, a temporary tree branch is computed, until an empty space split is found, or maximum depth is reached. On the first case, the empty space split is applied on base node and the branch is computed again, otherwise the branch is kept as it is.



V node is an empty space splitting node (Void split)

M node is a middle splitting node

Figure 3: Kd-Tree optimization

Statistical occlusion culling

The following step in pre-processing is tree cells density computation. Density must describe the culling effect of a cell in relation to another one. We use a Monte-Carlo method to compute the probability for a ray to go through a space-partitioning cell containing spheres.

We launch N rays with random position and direction through the cell, and we check if the ray goes through any of the spheres or not (see Figure 4). Sphere radii are fixed attributes of the particles. It is a five degrees of freedom problem, and Monte-Carlo basic method provides a $1/\sqrt{N}$ error convergence. A typical number of casted rays for a 1% error on density for one cell is 100,000 casts.

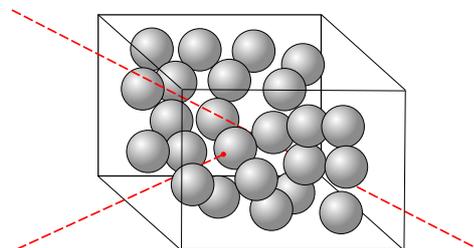


Figure 4: Monte-Carlo method for computing cell density.

This provides a 0 to 1 density factor, which represents the probability a ray has to go through the cell, and can be used as a trust percentage of this cell to occlude another. Each node of the tree has such a density factor.

Culling algorithms

We had to make a choice between the two most frequently used strategies for occlusion and frustum culling. The first one is silhouette comparison in viewport coordinates, like in [Coor 97]. The other one uses occlusion maps[Zhan 97]. We chose a strategy similar to the first one, because occlusion maps require a lot of GPU memory, which we would like to keep for the VBO cache system as described in section 5.2.

4.3.1 Silhouette computation

Occlusion culling is achieved by computing a silhouette of the occluding cell, and checking if a potentially occluded cell is inside the silhouette in viewport coordinates (cf Figure 5). Graham scan algorithm[Grah 72] gives us a y-sorted couple of point list which represents left side and right side of the silhouette.

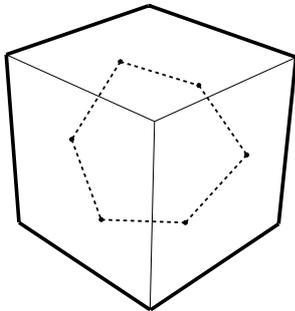


Figure 5: Block silhouette occluding another block

For each potentially occluded cell vertex, we find the left and right segments of the silhouette that share the same y-position as the top. Then we compare the x position of the top and the segments (see Figure6).

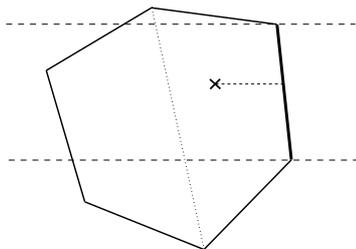


Figure 6: Silhouette occlusion check: X position compared to Y including segment

Depth check is done by comparing the z position of a vertex in relation to one or two planes. Planes are defined by triangles made by silhouette points close to the checked point, as seen in Figure 7. It is worth noting that this depth check is possible because we compare only disjoint cells.

4.3.2 Application to Kd-Tree

As seen in section 4.2, the Monte-Carlo pre-processing gives us a trust factor for cell opacity. If we set a threshold on this factor, we can consider some of the cells

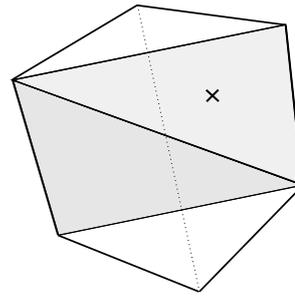


Figure 7: Depth check: triangles for planes definitions

are completely opaque, and then make a global culling comparison between tree nodes.

Each node has three possible states: *not occluded* (visible), *partially occluded*, and *occluded*. By default, all nodes are *not occluded* (visible). Leaves, as elementary undividable nodes, can only be tagged as *not occluded* (visible) or *occluded*.

As the tree is pre-computed, each node has a density attribute, and a maximum density of all the nodes beneath it. Then we recursively browse the tree from root to leaves, stopping as soon as a node with enough density is found.

For each occluding node, we perform the occlusion test, as seen in Figure 8: if the potentially occluded node (*PON*) contains no particle, or we already know it is occluded by another node, it is obviously occluded. If it is a child of the occluding node (*ON*), a test between the *ON*'s direct children and the *PON*. If the *ON* and the *PON* are the same node, and have children, they can potentially occlude a part of themselves, so we achieve a test between one of them and their children. If the *PON* is already partially occluded, we know that part of his children are already occluded, so we test his children. Otherwise, the silhouette of the *ON* is computed, and the occlusion is effective. Note that occlusion test between a node and one of its children can be meaningless because they can share bounds.

4.3.3 Frustum culling

Frustum culling is rather simple: the whole silhouette algorithm works in viewport coordinates. We only have to check if all cell coordinates are out of bounds per axis, i.e. if for any axis, cell coordinates are all either lower than -1 or higher than $+1$ in viewport coordinates.

5. PARTICLE RENDERING

In this section we describe some of the techniques we used to manage GPU memory and rendering.

Sphere rendering

We use GPU OpenGL shaders[Kess 06] programming to render particles as spheres. The big advantage of this technique is the per pixel precision of the rendering: thanks to the fragment shader program, all

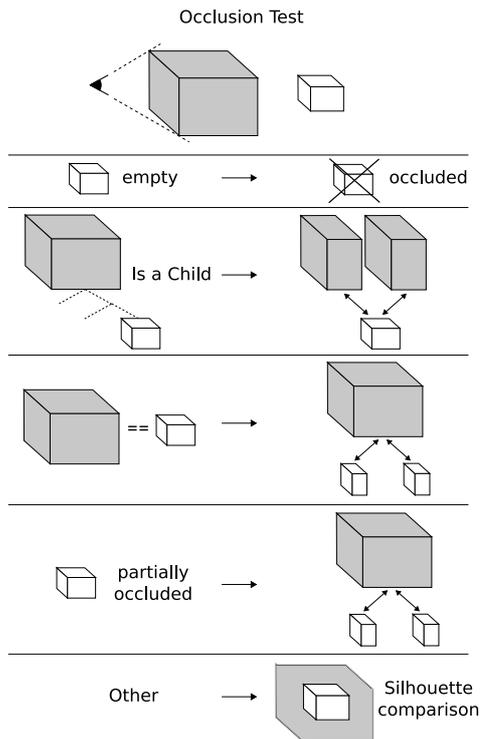


Figure 8: Occlusion Test

spheres are rendered pixel per pixel, and not with a group of vertices. This provides a direct level of detail feature, because rendering precision is only dependent on the number of pixels the sphere needs to be displayed. This also allows non standard lighting effects, like Phong[Phon 75] illumination, an example of which is shown in Figure 9.

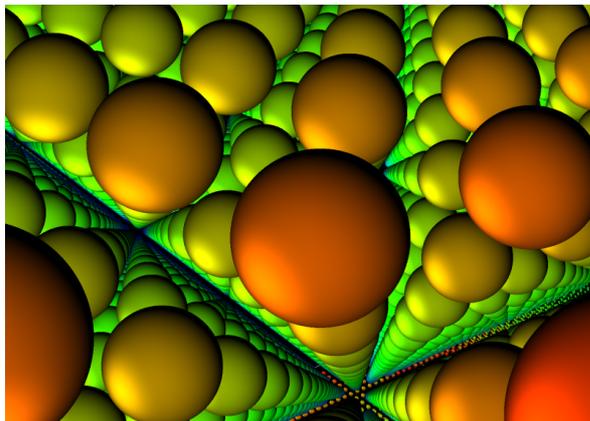


Figure 9: Phong illuminated spheres

GPU memory management

Vertex Buffer Objects (VBO)[Nvid 03] are a powerful way of managing GPU memory and RAM-to-GPU data transfers. We use them to create a cache system on GPU memory: for each tree leaf which is to be displayed, we create a VBO containing actual particles data used in

rendering stage (positions, colors, radii). We unload it only if GPU memory is full and the leaf not displayed.

6. PARALLEL ALGORITHMS

This section presents the distributed strategies applied to previously described occluding and rendering algorithms.

Sort-last stage

As said before, we chose to use Ice-T, with *Reduce to Single Tile*[More 01] method: each process is assigned to one of the display tiles. First it renders the part of the scene which consists of the data loaded by this process. Then each process splits the rendered image, and for each part of the image, which is to be displayed by a tile, sends the part to one of the processes of the tile group. Each process receives a balanced number of partial images to be displayed by the tile. Then the processes of a same tile group compose their images by binary-swap before display. Moreland[More 03] tests on a generic architecture with a cluster and a tiled display were conclusive.

Since the Ice-T algorithm is very dependent on network bandwidth, it includes some optimizations, like floating viewport, which reduces the size of images transferred for compositing by detecting not rendered zones on the global viewport. This feature plays an important part in the sort-first algorithms efficiency and we used it as is.

Sort-first stage

6.2.1 Partitioning of rendered tree parts

We try to share parts of the tree to balance per cluster node rendering time and network transfers.

All processes have the complete tree structure loaded. Tree nodes numbering is in Z-order, like in [Shar 02a], which provides a good spatial coherence between nodes with close numbers.

For each frame, we assign the first not occluded leaves with an average number of particles to the first process, the following leaves to the second process, and so on, until all leaves are assigned, as seen in Figure 10. This average number or particles is the nearest integer to $(Visible\ particles) / (Number\ of\ cluster\ nodes)$, modulo the cardinality of the last leaf assigned to the current process.

The per node data is then spatially coherent, which is good for Ice-T floating viewport technique. Moreover, as two successive frames have relatively similar trees, cluster nodes have to render almost the same leaves, and the cache system is efficiently used.

6.2.2 Overlapping culling algorithm

The occlusion culling algorithm was easily modified to become distributed.

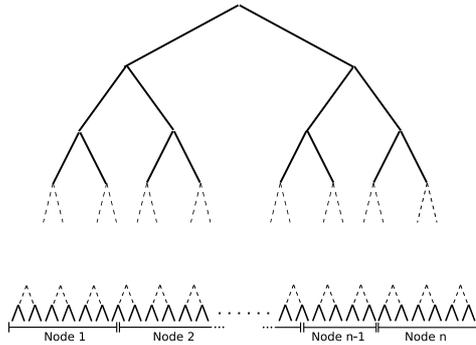


Figure 10: Displayable leaves assignment

Each frame, each cluster node performs an occlusion test, with only his previously assigned tree parts as potentially occluded cells. Then all nodes broadcast a signature of the tree, and each node merge them to have the entire tree configuration. The tree signature is a $(Number\ of\ leaves)/4$ bytes buffer: each leaf is coded with 2 bits, because it has three states. All untested nodes are *not occluded*.

Although signature broadcast is not a very good scalable method, the really small size of the signature implies that this part does not slow down the overall process. As an example, with a typical depth of 15 for the tree for the 32 million particles dataset described in next section, the signature has a size of 8192 bytes, which is no more than Ethernet Gigabit maximum MTU (9000 bytes).

7. IMPLEMENTATION AND FIRST RESULTS

Base framework

Ice-T is integrated in Paraview[Ahre 05], based on the generic framework VTK. For our current implementation we use only core VTK and Ice-T, with a number of additional C++ classes compliant with VTK. We also created a *vtkMapper* family of classes to integrate VBO usage in VTK rendering process.

Protocol

We use a 32 millions particles dataset, describing a typical atomic system used in molecular dynamics detonation wave simulations. The graphics and display hardware is a four-tiled, 2048x1536 display, with an 8-node Gigabit Ethernet commodity cluster. Each node is a Bi-Xeon 3.4Ghz with a NVidia Quadro FX 4500 graphics board.

We compared three sort-first methods: a non hierarchical non ordered method, i.e. only a big VBO per cluster node; a Z-ordered repartition without occlusion; and the full occlusion method. In the first method, network is only used to send synchronization signals: data is distributed among nodes, and loaded once in GPU memory.

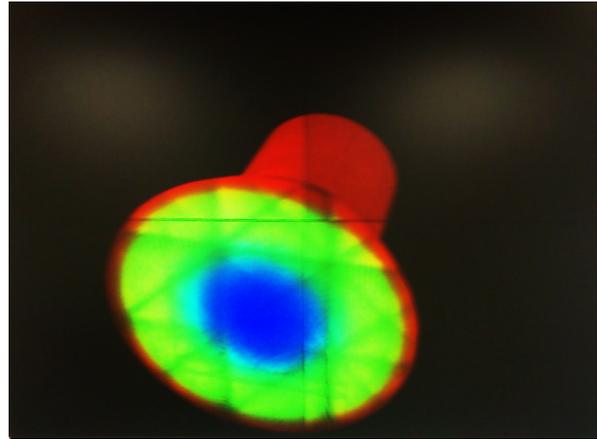


Figure 11: 32 million particle cylinder displayed on the 4-tile, 2m wide screen (around 3.3 M pixels)

As the full solution was designed to globally explore datasets, we tested two different camera movements: a rotation with constant long or short distance from dataset center; and a zoom towards a point of interest.

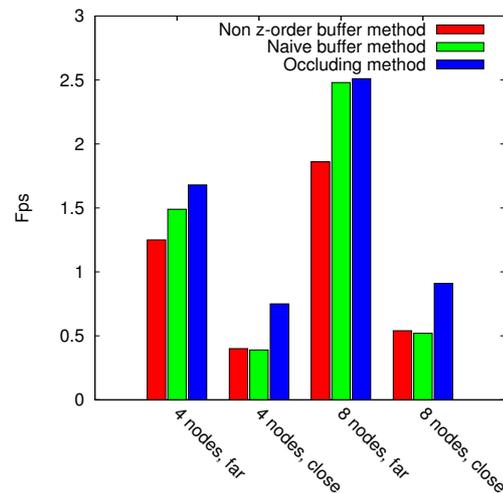


Figure 12: Camera rotating around dataset

On Figure 12 we can see the importance of Z-order in cluster nodes repartition.

We can notice that with eight nodes on the far view tests, culling (in blue) does not bring much better results than the naive (in green) method. The reason is that good Z-order data rendering repartition strongly reduces the floating viewport surfaces, which lower the network bandwidth usage. For example, in the 8 nodes far view test, bandwidth usage drops from 750Mb/s (effective maximum bandwidth) with non Z-order method to an average 250Mb/s in both native and occluding method. On the close tests, Figure 12 shows the obvious efficiency of culling in such a situation. Z-order repartition and culling each reveal their efficiency in one of the different rotation scenarios.

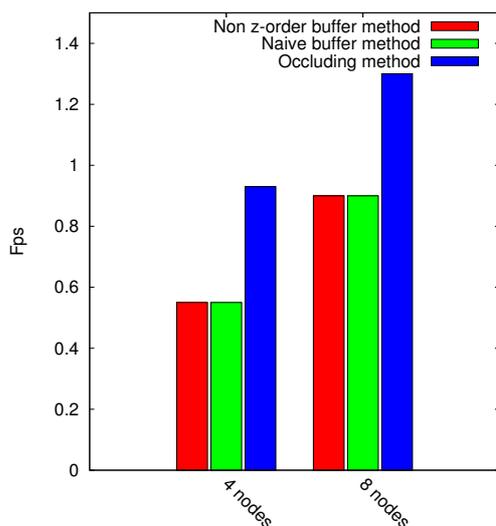


Figure 13: Camera zooming in dataset

Zooming results confirm this (Figure 13). As the camera gets closer to the dataset, the scene takes more and more space in the viewport. The network is saturated (like rotation tests, from an average 250Mb/s to 750Mb/s), and actual rendering becomes more and more time-consuming. Z-order is not very important, because the floating viewport is not activated here.

The rendering quality is a step function of the density threshold. If the density threshold is high enough, there is almost no difference between occluding and non occluding rendering. The threshold used for above results is 95%. For the camera rotation movement, 30% to 60% of the particles were culled. In the zoom test, culling got up to 94%. Lower threshold values obviously provide more interactive rendering, but artefacts do appear, like holes in the displayed dataset (cf Figure 14).

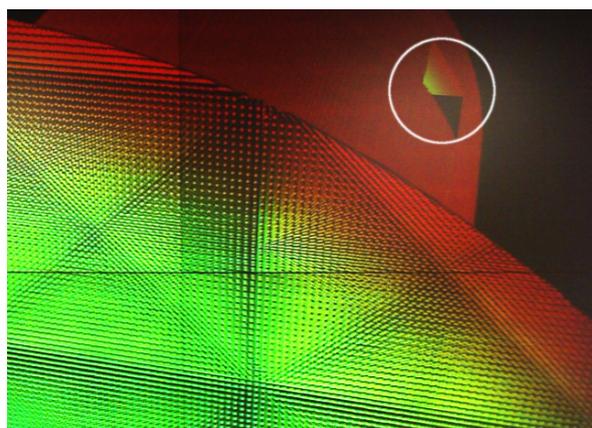


Figure 14: Artefacts with low density threshold

As priority for such a solution is global exploration and zone of interest detection, the most important fac-

tor in results is close rotation framerate. Our solution provides twice the framerate as the GPU only solution.

It is important to note that density is used the same way in far and in close cases. It should be interesting to lower threshold as camera's distance from datae increases. This could greatly improve results in far cases.

8. CONCLUSION AND FUTURE WORK

Our system provides a good framework for the exploration of large particle datasets representing dense material simulations. Sort-first rendering based on a specific Kd-Tree partitioning and statistical culling, with Monte-Carlo density computation, improves Ice-T sort-last strategies to reduce network bandwidth usage. Spheres rendering is also accelerated by GPU shaders. Future scale up tests will be achieved on a 12 tiled display fed by a 12 nodes cluster. We could further improve rendering by using better illumination methods, like ambient occlusion[Tari 06], to exploit the maximum potential of modern GPU power. Tests on a lower-latency and higher bandwidth network such as Infiniband should also be interesting.

Our next solution improvement will be to take camera's distance to the dataset into account to choose density threshold. We expect far camera tests framerate to be greatly improved.

Interactive tree computation is already fast, but with some optimizations, it could be done in real time. The main problem is density computation. One of the solutions could be to use General-Purpose computation on GPUs (GPGPU), like NVidia CUDA technology to accelerate the Monte-Carlo method.

9. ACKNOWLEDGEMENTS

Very special thanks to people from CEA/DIF, and especially Jean-Philippe Nominé for his support and his help, Thierry Carrard for his advices on GPU and parallel programming, Laurent Soulard for his teaching on materials physics. Also thanks to John Biddiscombe and Jean Favre from CSCS for respectively initial work on vtk sphere mapper classes and VTK advices, and Patrick Callet from Ecole Centrale Paris for his tips on sphere illumination, and reviewers for their thoughtful and accurate comments.

10. REFERENCES

- [Ahre 05] J. Ahrens, B. Geveci, and C. Law. "ParaView: An End User Tool for Large Data Visualization". In: *Visualization Handbook*, Chap. 7, Academic Press, 2005.
- [Baja 04] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. "TexMol: Interactive Visual Exploration of Large Flexible Multi-Component Molecular Complexes". In:

- VIS '04: Proceedings of the conference on Visualization '04*, pp. 243–250, IEEE Computer Society, Washington, DC, USA, 2004.
- [Coor 97] S. Coorg and S. Teller. “Real-time occlusion culling for models with large occluders”. In: *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pp. 83–ff., ACM, New York, NY, USA, 1997.
- [Ells 04] D. Ellsworth, B. Green, and P. Moran. “Interactive Terascale Particle Visualization”. In: *VIS '04: Proceedings of the conference on Visualization '04*, pp. 353–360, IEEE Computer Society, Washington, DC, USA, 2004.
- [Fluk 00] P. Flukiger, H. Luthi, S. Portmann, and J. Weber. “MOLEKEL 4.0”. 2000.
- [Grah 72] R. L. Graham. “An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set”. *Inf. Process. Lett.*, Vol. 1, No. 4, pp. 132–133, 1972.
- [Hump 02] G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner, and J. Klosowski. “Chromium: A Stream Processing Framework for Interactive Graphics on Clusters”. 2002. presented at SIGGRAPH, San Antonio, Texas.
- [Hump 96] W. Humphrey, A. Dalke, and K. Schulten. “VMD - Visual Molecular Dynamics”. *Journal of Molecular Graphics*, Vol. 14, pp. 33–38, 1996.
- [Kess 06] J. Kessenich. “The OpenGL Shading Language (1.20)”. Tech. Rep., opengl.org, 2006.
- [Krug 05] J. Kruger, P. Kipfer, P. Kondratieva, and R. Westermann. “A Particle System for Interactive Visualization of 3D Flows”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, No. 6, pp. 744–756, 2005.
- [Lian 05] K. Liang, P. Monger, and H. Couchman. “Interactive parallel visualization of large particle datasets”. *Parallel Comput.*, Vol. 31, No. 2, pp. 243–260, 2005.
- [More 01] K. Moreland, B. Wylie, and C. Pavlakos. “Sort-Last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays”. In: *PVG '01: Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pp. 85–92, IEEE Press, 2001.
- [More 03] K. Moreland and D. Thompson. “From cluster to wall with VTK”. *IEEE symposium on Parallel and Large-Data Visualization and Graphics*, pp. 25–31, 2003.
- [Nvid 03] Nvidia. “Using vertex buffer objects”. Tech. Rep., NVIDIA Corporation, 2003. http://developer.nvidia.com/object/using_VBOs.html.
- [Phon 75] B. T. Phong. “Illumination for Computer Generated Pictures”. *Commun. ACM*, Vol. 18, No. 6, pp. 311–317, 1975.
- [Sama 00] R. Samanta, T. Funkhouser, K. Li, and J. Singh. “Hybrid sort-first and sort-last parallel rendering with a cluster of pcs”. In: *Eurographics/SIGGRAPH workshop on Graphics hardware*, pp. 99–108, 2000.
- [Schr 06] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th edition*. Kitware, Inc. publishers, 2006.
- [Shar 02a] A. Sharma. *Techniques and algorithms for immersive and interactive visualization of large datasets*. PhD thesis, Louisiana State University, 2002.
- [Shar 02b] A. Sharma, X. Liu, P. Miller, A. Nakano, R. K. Kalia, P. Vashishta, W. Zhao, T. J. Campbell, and A. Haas. “Immersive and Interactive Exploration of Billion-Atom Systems”. In: *VR '02: Proceedings of the IEEE Virtual Reality Conference 2002*, p. 217, IEEE Computer Society, Washington, DC, USA, 2002.
- [Shar 03] A. Sharma, R. K. Kalia, A. Nakano, and P. Vashishta. “Large Multidimensional Data Visualization for Materials Science”. *Computing in Science and Engg.*, Vol. 5, No. 2, pp. 26–33, 2003.
- [Stre 05] F. H. Streitz, J. N. Glosli, M. V. Patel, B. Chan, R. K. Yates, B. R. de Supinski, J. Sexton, and J. A. Gunnels. “100+ TFlop solidification simulations on BlueGene/L”. In: *Gordon Bell Prize at IEEE/ACM SC05*, 2005.
- [Tari 06] M. Tarini, P. Cignoni, and C. Montani. “Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 5, pp. 1237–1244, 2006.
- [Zhan 97] H. Zhang, D. Manocha, T. Hudson, and K. E. Hoff III. “Visibility Culling Using Hierarchical Occlusion Maps”. *Computer Graphics*, Vol. 31, No. Annual Conference Series, pp. 77–88, 1997.

GPU-based Responsive Grass

Orthmann Jens
Computer Graphics Group
University of Siegen
56076, Siegen, Germany
orthmann@fb12.uni-siegen.de

Christof Rezk Salama
Computer Graphics Group
University of Siegen
56076, Siegen, Germany
rezk@fb12.uni-siegen.de

Andreas Kolb
Computer Graphics Group
University of Siegen
56076, Siegen, Germany
kolb@fb12.uni-siegen.de

ABSTRACT

Large natural environments are often essential for today's computer games. Interaction with the environment is widely implemented in order to satisfy the player's expectations of a living scenery and to help increasing the immersion of the player. Within this context our work describes an efficient way to simulate a responsive grass layer with today's graphics cards in real-time. Clumps of grass are approximated by two billboard representations. GPU-based distance maps of scene objects are employed to test for penetrations and for resolving them. Adaptive refinement is necessary to preserve the shape of deformed billboards. A recovering process is applied after the deformation which restores the original that is to say the undeformed and efficient shape. The primitives of each billboard are assembled during the rendering process. Their vertices are dynamically lit within an ambient occlusion based irradiance volume. Alpha-to-Coverage completes the illusion as it is used to simulate the semitransparent nature of grass.

Keywords: Grass Simulation, Interactive Environments, GPU-based Collision Handling

1 INTRODUCTION

State-of-the-art 3D games and realtime simulations demonstrate the power of currently available graphics hardware for rendering exciting natural sceneries in real-time. As nature scenes often include a lot of plants (blades of grass, shrubs, trees etc.) the rendering of a large number of them is still challenging. Furthermore, they cannot be displayed with complex geometry in real time. Many of the approaches make use of billboard representations to preserve the real-time constraint while leaving out user interaction.

In general, static level design is more and more replaced by dynamic environments that can be modified in real-time throughout the gaming process. Due to the fact that natural phenomena are better approximated in the game, the player feels a higher immersion while playing [McM03]. Consequently, the dynamic environment is becoming a part of the game logic: Trees are chopped to clear the path and objects need to be moved in order to fulfill quests. The more the realism of the scene is enhanced the more of the player's expectations are satisfied.

Following this trend, our paper takes dynamic environments one step further by integrating responsive real-time simulation of ground vegetation. We propose a highly efficient technique for GPU-based simulation

of responsive grass billboards. Our implementation targets Shader Model 4 graphics boards, including geometry shaders and stream output. The collision detection with dynamic scene objects, the response and the recovering are directly simulated on the GPU. An adaptive geometrical representation of the grass guarantees a pleasing visual rendering in conjunction with a high performance. Thus, the responsive grass approach has the potential to significantly improve the challenges in game play of modern games and may lead to a better perception of interactive environments.

The structure of this paper is as follows: in Section 2, an overview of the related work on grass simulation is given, followed by an overview of the responsive grass system in Section 3. Section 4 proposes the procedural generation process of the grass layer. In Section 5, the realization of the collision system is described. The rendering of the grass layer is presented in Section 6 and the results and performance of our technique is discussed in Section 7. Finally, Section 8 concludes the presented responsive grass approach.

2 RELATED WORK

In recent years, most research applied to natural sceneries focuses on the rendering and animation of a great number of plants. For volumetric representations, as proposed in [BCF⁺05, BPB06], collision detection and reaction is awkward to handle. Guerraz et al. [GPR⁺03], however, presented an approach which allows an object to tramp on the grass layer. A primitive is moved along the character's trajectory while affecting the procedural animation process of the grass. Nevertheless there still is no possibility to react to collision, based upon the object's geometry. The reuse of grass tiles amplifies the problem of collision

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

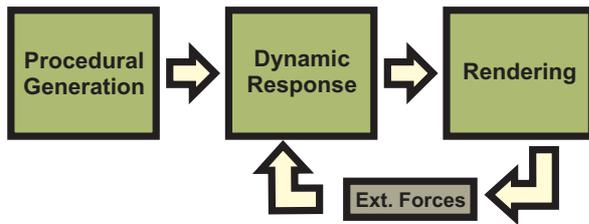


Figure 1: The system for responsive grass.

response. Billboards which represent a number of grass blades as a semi-transparent 2D texture are more suitable in that case. The billboard representations stored in a single vertex buffer [Wha05] are efficiently animated [Pel04, Bot06, Sou07] and rendered [BCF⁺05] on the GPU.

As the collision detection for grass is less explored, related algorithms on a wider range are examined. In case large dynamic geometry is stored and processed completely on the graphics memory, image-based techniques [VSC01, KP03, HTG04, KLRS04, GLM05, Sat06] are proven to solve the collision tests very fast. Kolb et al. [KLRS04] offered an approach to collision detection using distance maps which are fully generated and accessed on the GPU. A lookup into each distance map is used to decide whether a vertex lies inside or outside of a object. Using the normal information the vertex can be translated in the direction of the shortest way out of the object. Their approach fits best in case all computations, including the collision reaction, are done on the GPU.

Cloth models [Pro95, FGL03, Zel07] are applied in order to overcome the problems in the context of the collision reaction. Fuhrmann et al. [FGL03] replace the cloth forces [Pro95] by several length constraints along the connection of two particles in order to avoid problems which are caused by large time steps. Zellner [Zel07] entirely offloads the model to the GPU and handles the recursion via the stream output stage.

Regarding high quality rendering of massive material scenery a precomputed irradiance volume is employed [Oat06, CL07]. The volume stores the irradiance information of the whole static scene. Interpolation within the volume allows us to dynamically lit the grass billboards at runtime similar to the two-sided lighting proposed by Kharlamov et al. [KCS07]. The Alpha-To-Coverage feature of today's graphic cards [Mye06] avoids expensive depth-sorting of the semi-transparent billboards while maintaining a consistent visual appearance similar to David Whatleys procedure [Wha05].

3 SYSTEM OVERVIEW

The pipeline for responsive grass comprises the following components as shown in Figure 1:

- **Procedural Generation:**

For a given terrain mesh, a geometry shader automatically generates billboards for grass blades. This geometry shader is executed once for each tile of terrain, and the results are stored in local video memory using the stream-out capabilities. We describe the process in detail in Section 4.

- **Dynamic Response:**

A CPU-based broad phase working on the spatial organized grass tiles and a GPU-based narrow phase working on the generated grass billboards constitute the responsive component. During this stage the grass layer will be adapted whenever external forces like colliding scene objects make it necessary. This process which is implemented within the collision system is outlined in Section 5.

- **Rendering:**

Deformed or undeformed billboards are rendered based on the output of the collision system. Pre-computed occlusion volumes respectively irradiance volumes may be employed to integrate ground vegetation into a dynamic global lighting environment. We adapt such techniques for realistic rendering of dynamic ground vegetation as described in Section 6.

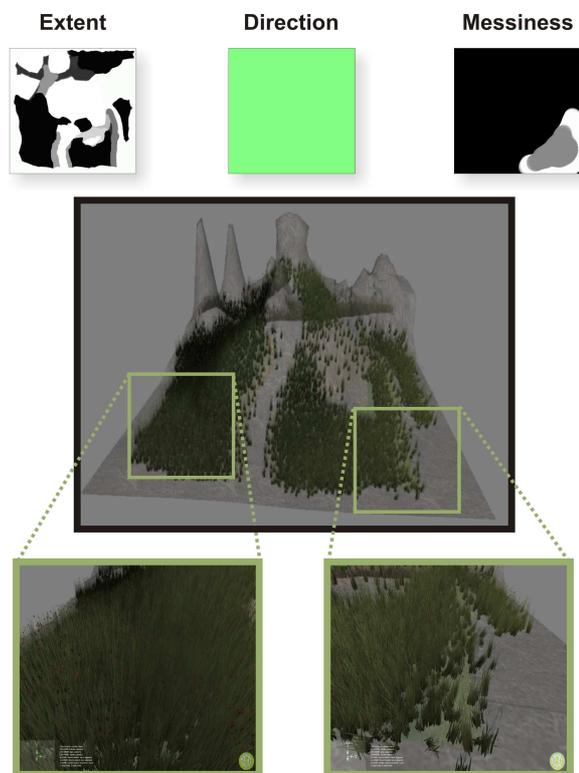


Figure 2: The top row shows the texture images for the extent, direction and messiness and underneath the resulting plant cover is displayed.

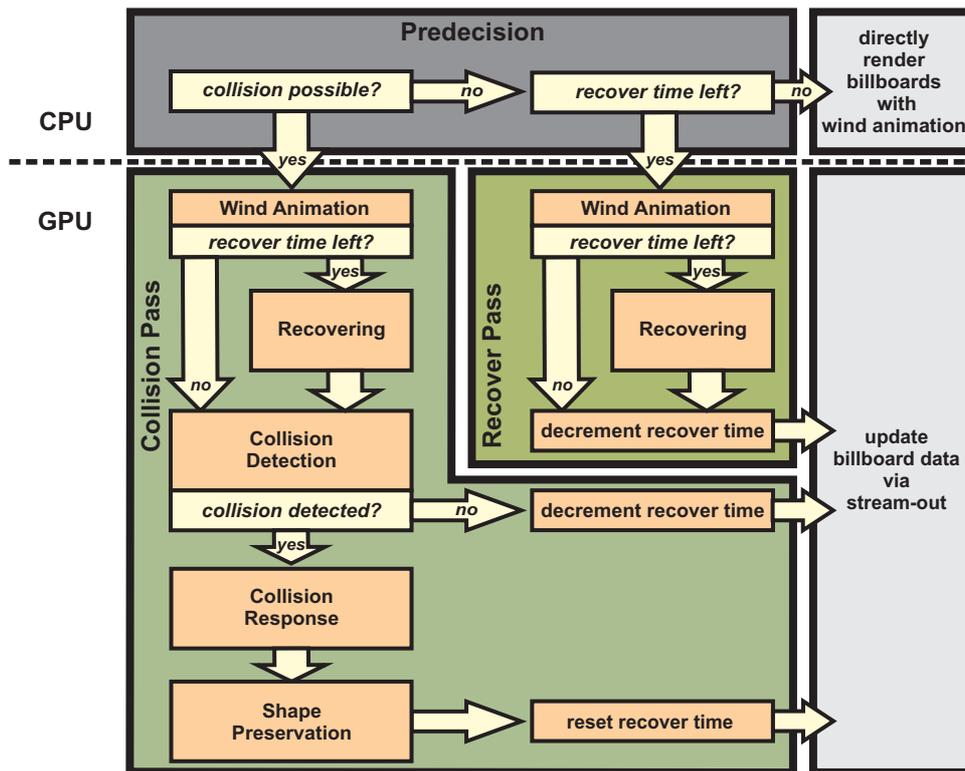


Figure 3: Flow diagram for collision detection, reaction and recovering.

4 PROCEDURAL GENERATION OF THE BILLBOARDS

A clump of grass is represented by a semi-transparent textured quad. The individual billboards are created in a pre-processing step performed by the GPU. A set of texture images provides the information of the global layout of the grass layer as shown in Figure 2. These textures are in detail:

- a grayscale texture map which defines the regions of the plant cover (extent),
- a RGB texture which defines the direction to which the grass blades grow (for simplicity the direction is chosen to be the same for all grass blades),
- a grayscale *messiness* texture which defines the amount of randomness for the blades.

The geometry shader creates a randomized set of billboards representing the grass blades. Each billboard stores an orientation, a position, a collision state, and a texture index, addressing a 2D texture array, which stores different semi-transparent images of grass clumps. Each billboard is passed through the pipeline as a point primitive, which allows the different geometry shaders to handle its information en bloc during the collision handling and rendering. When the billboards are generated, they are streamed to one large vertex buffer [Wha05] to minimize subsequent render

calls. For a coarse collision detection on the CPU, the terrain mesh is used to divide the set of billboards into an octree hierarchy. Each leaf node of the octree stores a range of indices into the vertex buffer of the billboards and state information described throughout the next section.

5 COLLISION SYSTEM

The pipeline of the collision system is outlined in Figure 3. Without collision, the billboard quads can directly be rendered. The upper two vertices of each billboard quad are transformed with a procedural wind animation based on a weighted sum of trigonometric functions with different frequencies [Pei04, Wha05, Bot06, Sou07]. The collision system is split into a CPU-based coarse handling and two GPU-based procedures, one for executing the collision test and response and one for performing the recovering. The different steps of the GPU-based collision handling are outlined in Figure 4.

5.1 Coarse Handling on the CPU

At each frame, the bounding volumes of all dynamic collision meshes are tested for collision with the axis aligned bounding boxes (AABB) of the octree containing the grass blades. According to the current state information and the results of the collision test, each node is marked as either *possibly colliding*, *non-colliding* or *recovering*. The geometry assigned to each octree node

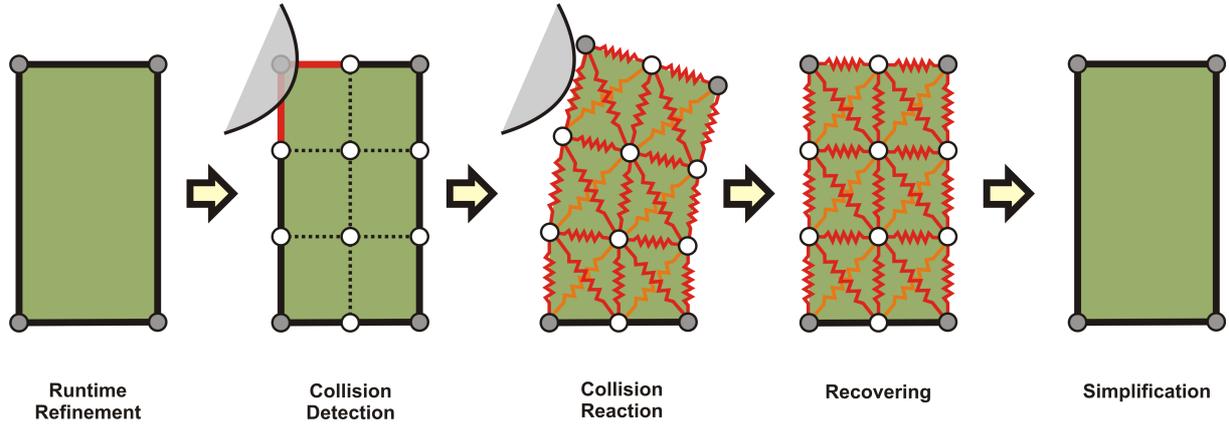


Figure 4: Collision detection, reaction and recovering for a single billboard.

marked as *possibly colliding* is streamed through a collision pass on the GPU. A tile marked as *recovering* will stay active for a fixed amount of time after the collision when the object has left the AABB of the octree node. During that time a separate geometry shader recovers the original shape of the grass blades.

5.2 Collision Detection

In the collision pass, a geometry shader receives all the vertex information of a potentially colliding billboard at once. The geometry shader computes the bounding sphere of the animated billboard and performs a collision test against the bounding spheres of the dynamic objects. These bounding spheres are managed in a dynamic texture resource, which is updated every frame. If the collision test is passed on the bounding sphere level, a second and more exact collision test is performed on a subdivided mesh of the billboard. The geometry shader determines for each vertex whether it lies inside or outside the dynamic collision object by performing lookups into the depth cube map [KLRS04] of the object. Dynamic objects capable of colliding with the plant cover are represented by depth cube maps for efficiency. These cube maps are computed by projecting the object's mesh onto the faces of a bounding cube and store the distance to the cube plane and the respective object normal in the four texture components. They are updated for each frame to account for animated objects. Thus, to perform the test each vertex $\mathbf{v} = (v_x, v_y, v_z, 1)^T$ is transformed to each of the six projection spaces:

$$\mathbf{v}^i = \mathbf{T}_{OC \rightarrow DM}^i \mathbf{v}, i = 1, \dots, 6, \quad (1)$$

where $\mathbf{v}^i = (v_x^i, v_y^i, v_z^i, 1)^T$ is the transformed vertex of the billboard. $\mathbf{T}_{OC \rightarrow DM}^i$ is a transformation from the object coordinate space to the i -th projection space from where the current distance map was computed. Along the projection direction the vertex lies within the object if

$$d^i(\mathbf{v}) = dm^i(v_x^i, v_y^i) - v_z^i < 0, \quad (2)$$

where $dm^i(x, y)$ is the distance looked up within the distance map dm^i at pixel position (x, y) . If the distances for all the six faces i of the cube do not yield a distance value $d^i(\mathbf{v})$ less than zero a collision with the billboard has been detected. Identifying the distance $d(\mathbf{v})$ between the closest surface point in the corresponding depth cube face for a given point \mathbf{v} , the following formula is used (for details see [KLRS04]):

$$d(\mathbf{v}) = \begin{cases} \max\{d^i(\mathbf{v})\} & \text{if } d^i(\mathbf{v}) < 0 \forall i \\ \min\{d^i(\mathbf{v}) : d^i(\mathbf{v}) > 0\} & \text{else} \end{cases} \quad (3)$$

5.3 Collision Response

If a collision has been detected, the vertex is moved out of the object's shape. Its position is translated along a normal vector \mathbf{n} obtained from the depth cube map:

$$\mathbf{v} \leftarrow \mathbf{v} + s \mathbf{n}, \quad (4)$$

where \mathbf{n} is taken from the depth cube map face dm^i providing the smallest distance. s is the reaction strength that is to say the surface normal \mathbf{n} multiplied with the smallest distances to the surface $d(\mathbf{v})$:

$$s = d(\mathbf{v}) \frac{\mathbf{n}}{\|\mathbf{n}\|}. \quad (5)$$

In order to remember the collision, the data-structure of the billboard is expanded by an additional value storing its recover time. In case of a collision the recover time is reset.

5.4 Shape Preservation

As the separate processing of individual vertices may lead to visually unpleasant distortions, a cloth model based on spring constraints [Pro95, FGL03, Zel07], is applied to preserve the overall shape of the grass clump. A network of structural and shear springs takes care of the billboard mesh. Whenever such a spring is compressed or stretched, which means the connected vertices diverge or converge, the resulting spring force translates the connected vertices.

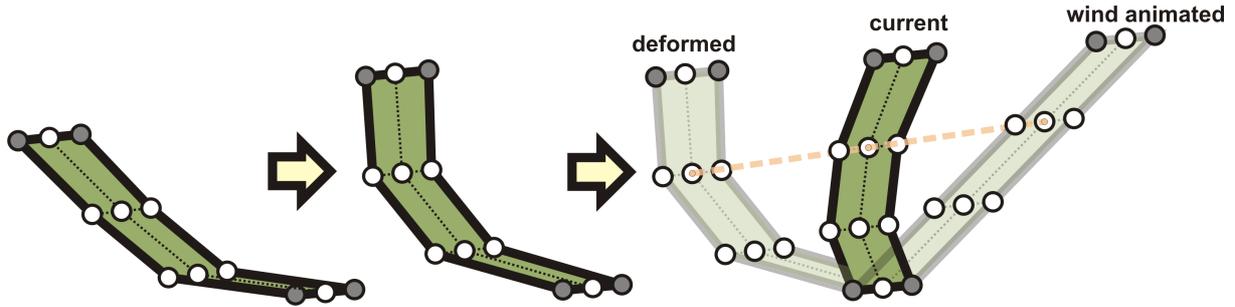


Figure 5: The interpolation between the vertices of the current mesh (deformed mesh) and the vertices solely affected by the wind animation results in a smooth recovering over time.

Referring to Provat et al. [Pro95], a spring force $\mathbf{f} \in \mathbb{R}^3$ between two billboard vertices \mathbf{v}_1 and \mathbf{v}_2 is defined as:

$$\mathbf{f} = k(\|\mathbf{l}\| - l_0) \frac{\mathbf{l}}{\|\mathbf{l}\|}, \quad (6)$$

where $\mathbf{l} = \mathbf{v}_1 - \mathbf{v}_2$ is the direction of the connection between both vertices. l_0 is the initial length of the spring and $k \in [0, 1]$ is the stiffness of the spring. A stiffness of 1 results in a conservative spring in contrast to a value of 0 which has no effect. Each spring force directly affects the two connected vertices [FGL03, Zel07]:

$$\begin{aligned} \mathbf{v}_1 &\leftarrow \mathbf{v}_1 - r_1 \Delta t \mathbf{f} \\ \mathbf{v}_2 &\leftarrow \mathbf{v}_2 + r_2 \Delta t \mathbf{f}, \end{aligned} \quad (7)$$

where r_1 is the responsiveness for vertex \mathbf{v}_1 and r_2 is the responsiveness for vertex \mathbf{v}_2 with $r_1 + r_2 = 1$. We added the responsiveness in order to distinguish between fixed ground vertices and movable vertices. As a fixed vertex should not be moved, the responsiveness is set to zero whereas the other vertex then is completely responsive. If both vertices are not fixed they are equal responsive and thus $r_1 = r_2 = 0.5$.

As the relaxation of one spring affects the neighbouring springs as well, in general more iterations over all springs have to be applied to get a good result. In our case two iterations yield visually pleasant results due to the small number of vertices.

5.5 Recovering

The recovering is processed on each billboard that has some recover time left. Since the animation is a stateless process, solely based upon the position of the fixed ground vertices and the current time [Sou07], it is possible to compute the original shape defined by the wind without considering the current collision state. The linear interpolation between the deformed vertex and its original position, with respect to the recover time left, results in the current shape of the grass clump as shown in Figure 5:

$$\mathbf{v} \leftarrow (1 - t^3)\mathbf{w} + t^3\mathbf{v}, \quad (8)$$

where $t \in [0, 1]$ is the recover time left, \mathbf{w} is the vertex position obtained by the wind function and \mathbf{v} is the current respectively last recovered vertex position.

Collision tests are required in case that there are still collision objects inside the AABB of the respective oc-tree node. At every time without any collision, the recover time will be decreased. After the recover time has elapsed, the billboards will be handled again as simple quads. However, the recovering does not preserve the length of the billboards.

6 RENDERING

On the CPU level, grass tiles which previously have been streamed and others that have not been affected by neither collisions nor wind exist. The tiles run through separate render passes: Collided billboards are rendered using their current refined mesh whereas the unaffected ones are animated and rendered using their simple quad-representation. Furthermore, to overcome problems caused by too much render calls, only batches of visible tiles, which have not been culled by view or occlusion queries, are rendered.

6.1 Global Illumination

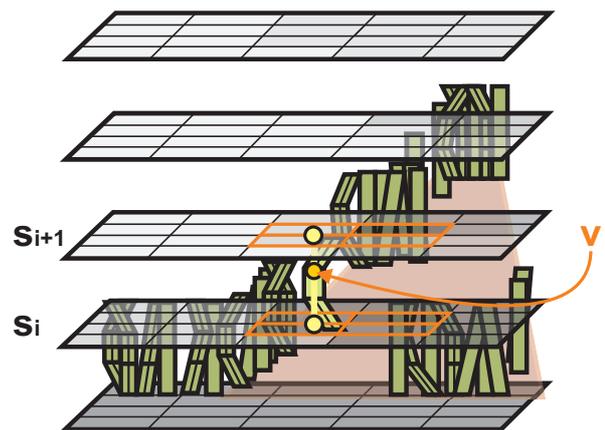


Figure 6: The irradiance for each vertex \mathbf{v} of the billboard is interpolated within the two closest texture slices s_i and s_{i+1} .

Dynamic global illumination is achieved by pre-computing a volume, with each voxel storing ambient occlusion information for its location in the scene [CL07]. The whole volume is then stored



Figure 7: The result of the collision handling in a dense field of grass.

as an 2D texture array to allow linear interpolation based on mip-mapping. In addition, a second volume which covers the same space provides pre-computed irradiance information for each point. The irradiance is determined by sampling an environment map by using the previously computed ambient occlusion information [PG04].

The texture coordinate for the volume texture can easily be obtained from the billboard's vertex positions in the geometry shader. Ambient occlusion and irradiance information is trilinear interpolated between adjacent texture slices, and the incident light is evaluated per vertex during the geometry shader process as illustrated in Figure 6. Finally, the pixel shader uses the texture index into the semi-transparent texture array to receive the decal color and transparency of the grass clump. Multiplying this decal value with the incident two-sided light [KCS07] results in the final semi-transparent pixel color.

6.2 Alpha-To-Coverage

Since grass has a semi-transparent nature a feature of modern cards, so-called Alpha-to-Coverage, is used to blend the billboards without the necessity to perform expensive depth-sorting. The alpha value is used to determine the number of subpixels, that will be filled with the current pixel color. Then, blending between the subpixels is performed while resolving the multisample resolution to the final image resolution [Mye06].

7 RESULTS AND PERFORMANCE

Achieving a high performance is one of the major aims to real-time applications. All components concerning the grass layer are designed to reduce the workload of the CPU as much as possible. Thus, the simulation is almost completely shifted to the GPU. All the tests are performed on an AMD Athlon 64 3500+ 2.2 GHz processor including a GeForce 8800 GTX graphics card with 768 MB DDR3 memory. Figure 7 shows the response of the grass after the scene object has moved through the meadow. The scene, presented in Figure 2, is running at 40-80 frames per second by using

DirectX 10 and fourfold multi sampling anti aliasing (4xMSAA). The grass layer contains 60000 grass billboards requiring 12 MByte of graphics memory. All invisible grass tiles are culled. The grass is pushed to the side or is stamped down on the line of movement. The object has left a clearly noticeable imprint on the grass. We analyzed the performance of the scene with the aid of the NVidia PerfHUD tool. In Figure 8 the number of colliding grass tiles (red boxes) respectively recovering grass tiles (green boxes) increases from top down. The lower left overlay displayed in each image shows the workload balancing of the programmable render pipeline stages: The unified streaming processors are utilized to work on pixels with about 50 to 60 percent (the blue bar) whereas the geometry shader unit of the pipeline is active by approximately ten percent (the green bar). The remaining workload is caused by frame buffer operations. Approximately 16 million pixels are processed within the fragment shader resulting in many read as well as write accesses to the frame buffer. Those are amplified by the Alpha-to-Coverage feature which in that case requires a multisample resolution that is four times higher than the image resolution. The diagrams located at the right hand side of each image in Figure 8 present the amount of time which is consumed within each GPU pass: Please note that the time spent within the recover process (R) and the collision pass (C) varies only by small amounts. In contrast, the more grass billboards are deformed the more time is spent rendering the collided and recovering grass tiles (RA). This performance loss is caused by the primitive generation as well as the rendering of the high number of primitives. Referring to the utilization graph and the time measurements the performance of the system depends on the number of assembled primitives which are passed through the rasterizer back-end. Thus, both the memory operations as well as the workload shifted to the fragment shader stage, are influenced by the number of colliding grass billboards. Consequently, it is necessary to set up a low recover time and to provide a low multi-sampling rate for the Alpha-to-Coverage process to preserve the overall performance. In con-

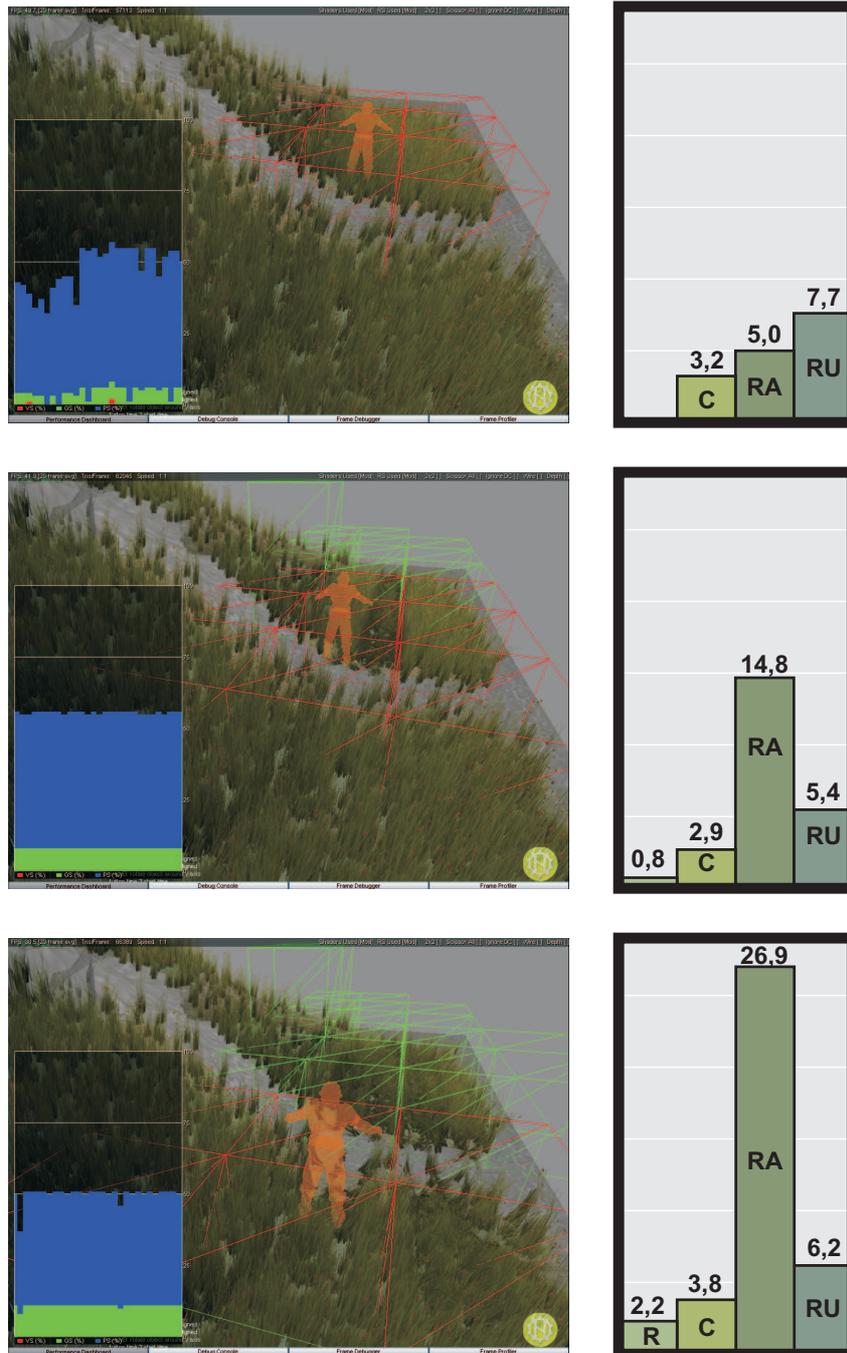


Figure 8: A scene which contains more and more deformed grass billboards. The time spent within each GPU pass is displayed in milliseconds on the right-hand side of the corresponding image. The measured passes are: The recovering pass (R), the pass performing the collision handling (C), the rendering of the possibly affected grass tiles (RA), and the rendering of the unaffected grass tiles (RU).

trast, the time spent within the collision handling depends mainly on the number of scene objects moving through the grass layer.

8 CONCLUSION AND FUTURE WORK

In the past thousands of billboards were successfully used to create an illusion of dense grass vegetation. In

combination with wind animation nice visual results were achieved. But the visual perception was often compromised by lack of interactivity: Objects are moving through the grass without leaving a trace. Due to prior hardware constraints a visually pleasing collision reaction for a large area of grass was unachievable. The visual quality of dense vegetation and the good performance give a proof of the great suitability of our imple-

mentation strategies for large responsive grass layers in today's real-time applications.

The results are demonstrating that collision response works fine for regions where the flat structure of the grass billboards is hardly recognized. However, in areas where grass is planted sparsely, for example at the borders of the grass layer, due to the coarse mesh of the billboards the visual impression could be improved. Two different approaches might be promising when trying to solve this problem: On the one hand the collision handling for each billboard could be distributed over several streaming passes which allows the spring constraints to work on a higher subdivided mesh. On the other hand the displayed primitives could be assembled by a higher order interpolation during rendering.

REFERENCES

- [BCF⁺05] S. Behrendt, C. Colditz, O. Franzke, J. Kopf, and O. Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Comput. Graph. Forum*, 24(3):507–516, 2005.
- [Bot06] A. Botorabi. *Shader X5*, chapter Animating Vegetation Using GPU Programs, pages 141 – 175. Charles River Media, 2006.
- [BPB06] K. Boulanger, S. Pattanaik, and K. Bouatouch. Rendering grass in real-time with dynamic light sources and shadows, 2006.
- [CL07] G. Cadet and B. Lécussan. Fast approximate ambient occlusion. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*, page 191. ACM, 2007.
- [FGL03] A. Fuhrmann, C. Groß, and V. Luckas. Interactive animation of cloth including self collision detection. In *WSCG*, 2003.
- [GLM05] N. K. Govindaraju, M. C. Lin, and D. Manocha. Quick-cullide: Fast inter- and intra-object collision culling using graphics hardware. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 59–66, 319, Washington, DC, USA, 2005. IEEE Computer Society.
- [GPR⁺03] S. Guerraz, F. Perbet, D. Raulo, F. Faure, and M.-P. Cani, editors. *A Procedural Approach to Animate Interactive Natural Sceneries*. IEEE Computer Society, 2003.
- [HTG04] B. Heidelberger, M. Teschner, and M. H. Gross. Detection of collisions and self-collisions using image-space techniques. In *WSCG*, pages 145–152, 2004.
- [KCS07] A. Kharlamov, I. Cantlay, and Y. Stepanenko. *GPU Gems 3*, chapter Next-Generation SpeedTree Rendering, pages 69–92. Addison-Wesley, 2007.
- [KLRS04] A. Kolb, L. Latta, and C. Rezk-Salama. Hardware-based simulation and collision detection for large particle systems. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 123–131, New York, NY, USA, 2004. ACM.
- [KP03] D. Knott and D. K. Pai. CInDeR: Collision and interference detection in real-time using graphics hardware. In *Graphics Interface*, pages 73–80, 2003.
- [McM03] A. McMahan. *Immersion, Engagement, and Presence - A Method for Analysing 3-D Video Games*, chapter 3, pages 67–85. Routledge Taylor & Francis Group, 2003.
- [Mye06] K. Myers. *Shader X5*, chapter Alpha-to-Coverage in Depth, pages 69 – 74. Charles River Media, 2006.
- [Oat06] C. Oat. *Shader X5*, chapter Irradiance Volumes for Real-time Rendering, pages 333 – 357. Charles River Media, 2006.
- [Pel04] K. Pelzer. *GPU Gems*, chapter Rendering Countless Blades of Waving Grass, pages 107 – 121. Addison-Wesley, 2004.
- [PG04] M. Pharr and S. Green. *GPU Gems*, chapter Ambient Occlusion, pages 279 – 292. Addison-Wesley, 2004.
- [Pro95] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, May 1995.
- [Sat06] R. Sathe. *Shader X5*, chapter Collision Detection Shader Using Cube-Maps, pages 533 – 542. Charles River Media, 2006.
- [Sou07] T. Sousa. *GPU Gems 3*, chapter Vegetation Procedural Animation and Shading in Crysis, pages 373 – 407. Addison-Wesley, 2007.
- [VSC01] T. I. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. *Comput. Graph. Forum*, 20(3), 2001.
- [Wha05] D. Whatley. *GPU Gems 2*, chapter Toward Photorealism in Virtual Botany, pages 7–25. Addison-Wesley, 2005.
- [Zel07] C. Zeller. Cloth simulation. White paper, NVidia, 2007.

Giga-Voxel Rendering from Compressed Data on a Display Wall

R. Parys, G. Knittel
WSI/GRIS, Tuebingen University
72076 Tuebingen, Germany
[parys | knittel]@gris.uni-tuebingen.de

ABSTRACT

We present a parallel system capable of rendering multi-gigabyte data sets on a multi-megapixel display wall at interactive rates. The system is based on Residual Vector Quantization which allows us to render extremely large data sets out of the graphics memory. At 0.75 bits per voxel, such large data sets can even be kept on a consumer-level graphics card. As an example we compress the whole full color “Visible Human Female” data set, approximately 21GByte in size, down to 700MByte. Taking advantage of the fixed code length and the extremely simple decompression scheme of RVQ, all decompression is done on the GPU at very high rates. For each frame the data set is decompressed into small subvolumes which are rendered front to back. Classification and shading can be moved into the decompression step, speeding up the rendering pass.

Keywords

Distributed and Parallel Graphics, Volume Rendering, GPU Programming

1. INTRODUCTION

Since quite some time volume rendering has made it from a purely academic research area into a well-established computer graphics application with high economic relevance. Still, comprehensive platform support as in the case of computer games, as an example, is largely missing. Nevertheless, visualization requirements steadily increase: data sets are getting larger, rendering algorithms are getting more complex, and display resolutions are increasing as well. In this work we present a fairly extreme example: volume rendering of a very large data set (about 7.5G voxels) on a high-resolution display wall (65,536,000 pixels). This display system (see Fig. 3 at the end of the paper) consists of 16 LCDs with a resolution of 2560×1600 pixels each. Each display is driven by a PC, being equipped with a Dual-Core 2.4GHz Intel CPU, 4GByte of main memory and a 8800GT graphics card from NVidia. The latter in turn has a video memory of 1GByte capacity. The PCs are connected via GBit Ethernet. As a side note, each display is connected to its PC via a dual-link DVI cable no shorter than 20 meters.

While the PC-cluster would otherwise represent a decent computing platform, given the task at hand it is somewhat underpowered. Thus, we need to apply efficient optimizations. Also, the workload should be placed where the strongest computing resources are, and notorious bottlenecks such as the network should be avoided as much as possible, even if this causes some amount of redundant computation. With respect to computational power and memory bandwidth there is an easy choice: GPUs are approaching and surpassing the teraflops-mark, and peak memory bandwidth on consumer-level cards approaches 150GByte/s. These numbers are unavailable anywhere else in a typical workstation. From these consideration we derived the design choices of our rendering system, which will be described in detail in the following sections.

2. RELATED WORK

Other work related to our project falls into the areas of parallel volume rendering, data set compression, and GPU-based volume rendering.

2.1 Data Set Compression

A study on lossless compression for volume data is presented in [6]. The authors report a maximum of about 50% reduction in size for selected data sets, however, the work was targeted at reducing storage space rather than increasing rendering speed. The use of vector quantization for volume rendering was first proposed in [18]. The presented rendering system operates directly on the compressed data. An improved version can be found in [22], however, the method provides only nearest-neighbor interpolation and is thus limited in rendering functionality. In other early work the authors used Block Truncation Coding in a space-filling way to reduce memory bandwidth

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

[12]. Wavelet-based encoding, however, has received the most attention in recent years [9],[17],[20],[21]. A hierarchical wavelet representation of large data sets is used in [7]. The authors claim to achieve a compression rate of 30:1 without noticeable artifacts in the image. A quality measure is derived from the wavelet representation during rendering to minimize the number of voxels to process. Interactive rendering speeds for large data sets can be achieved on standard PCs. We didn't follow this approach since decompression is most practically done on the CPU, and sending pixel data over the bus to the graphics card can severely limit performance (see Table 1).

2.2 GPU-based Volume Rendering

The use of graphics hardware for volumetric data processing dates back to [1], [3] and [4]. Originally, 3D texturing hardware was used for volume rendering. Screen-aligned slices were swept through the 3D texture and blended in back-to-front order. Thus, a volume data set was treated as a light-emitting, translucent material. Later improvements included gradient shading [15], multi-dimensional transfer functions [11], pre-integrated transfer functions [5], and the processing of pre-segmented data sets [8]. Where it is possible and useful we try to integrate these techniques into our framework. Some features have lower priority, though. For example, data sets of the size considered here are rarely pre-segmented due to the large effort it takes. Thus, support for this feature is postponed to later versions.

2.3 Parallel Volume Rendering

Basically, parallel rendering can be done in two ways: object-space partitioning, and screen-space partitioning (see Section 3.2.6). To a large degree, the performance of systems using object-space partitioning is limited by the alpha blending of the intermediate images. Solutions are proposed in [14], [23] and [24]. In [24], it is described that for alpha-blending the CPU is used instead of the much better suited GPU. This highlights the difficulties of handling such large data streams in the network.

Rendering to a display wall of about 63M pixels is described in [16]. Here, isosurfaces are rendered from a total of 470M triangles. An example is shown with a rendering time of 15 seconds, demonstrating the challenges presented by these display systems.

3. THE GIGA-VOXEL SYSTEM

As previously mentioned, we prioritize the graphics card for all computations, even if this means a certain amount of redundant processing. A few benchmark figures might further motivate this choice. On a Dell XPS700 workstation, equipped with an Intel Core 2 Duo CPU at 2.13GHz and an NVidia GTX280 (optionally an 8800GT), we obtained the following results (measured with the SiSoft Sandra benchmark suite and "bandwidthTest" from the NVidia CUDA SDK [19]):

Test	Bandwidth
CPU ↔ Cache	98,520
CPU ↔ Memory, 16MB Blocks	2,100
CPU ↔ Graphics Card (PCIe 1.x)	1,500
GPU ↔ Video Memory GTX280	110,028
GPU ↔ Video Memory 8800GT	43,357

Table 1: Actual Bandwidth Measurements [MB/s]

Interestingly, the tools report the internal CPU cache bandwidth to be lower than the bandwidth to the external video memory on the GTX280. Thus the design target was set to keep all necessary data locally in video memory, and to use the GPU for all compute-intensive tasks. As a side effect, this frees the CPU to do supporting activities such as tissue simulations. In a cluster environment this means that the data must be replicated, and that pixel traffic must be kept to a minimum. Clearly, given the size of typical volume data sets, the former can only be achieved using data set compression. The compression scheme, however, has to fulfil contradicting requirements: it must provide a high compression rate at still high image quality, and the decompression must be simple and extremely fast. We found *Residual Vector Quantization* (RVQ) to be an interesting candidate for this purpose.

Thus, rendering a data set using the Giga-Voxel System is a two-stage process: first the data set needs to be compressed in an offline step, and then it can be loaded on the graphics cards and rendered. We'll start the description of the process chain with the compression step.

3.1 Residual Vector Quantization

Residual Vector Quantization has first been described in [10]. An excellent survey of RVQ and related techniques can be found in [2]. RVQ is an extension to standard vector quantization (VQ). In VQ, a (large) set of vectors is replaced by a (small) set of representative vectors (here called *codevectors*), while trying to minimize overall error. Often, clustering methods are used to find a proper set of codevectors (collectively called a *codebook*). A frequently used method is *k-means* [13]. Starting from an initial set of random codevectors (*seeds*), each vector is assigned to its nearest codevector, thereby forming clusters. Once finished, the codevectors are moved to the center of their respective cluster, and then clustering is started anew. This process is repeated until the system reaches a stable state. Each vector will now be replaced by the index of its codevector in the codebook. Decompression merely consists of a table lookup.

If the set of vectors is too large, or unknown at the time of codebook construction, a subset of vectors (*training set*) can be used to build the codebook. Any further vector is then replaced by the index of the best-matching codevector within the existing codebook.

For RVQ, the set of difference vectors is constructed, i.e., vector - codevector. This set of vectors, equal in number and dimension to the original vectors, is now

subjected to yet another VQ, giving a second set of indices and codevectors. Each original vector is now replaced by two indices, and the corresponding codevectors are simply added to give the decompressed version of the vector. This process can be repeated for the desired number of levels.

There are a number of quantities which affect the performance of RVQ. The number of codevectors per codebook and the number of codebooks define the length of the index set (the *codelength*) in bits. The number of dimensions of the original vectors and the width of each component define the compression rate relative to a given codelength.

From experiments with a large number of images we have found that larger codebooks should be favored over a high number of levels, since the image quality in terms of PSNR is higher for a given codelength. Clearly, however, there is a practical limit in codebook size, both with respect to memory consumption and compression time. Compared to an on-chip decompressor, we can take advantage of the much larger but still fast video memory. A number of tests have shown that a high image quality can be achieved using four levels, with a codebook of 4k codevectors on each level. This gives a codelength of 48 bits.

Since our test data set is the “Visible Human Female” in the full-color version, a pixel is a 24-bit RGB quantity. A vector is formed by a $4 \times 4 \times 4$ pixel cube in the image stack, and so the vector dimension is 192. The 64 pixels in a cube are compressed into 48 bits, giving a compression rate of 32:1, or 0.75bpp.

For the components of a codevector we use a higher precision to account for rounding errors. The RGB-fields of one pixel are packed into one 32-bit word in 11-11-10 format. Thus, a codevector consists of $64 \times 4 = 256$ Bytes. A codebook occupies 1MByte accordingly, for a total of 4MBytes for all levels.

3.1.1 Compressing the Visible Human Female

Originally, these images have a resolution of 2048×1216 pixels [25]. There are a total of 5189 images. The cadaver was submerged in a blue gel, which we have set to a black “empty space”. However, there is too much empty space around the data, so we have cropped the images to a final resolution of 1608×896 pixels. This gives an input data set size of 20.9GByte.

Compression time for a data set of this size would be too long, so we have selected a training set equivalent in size to 300 images. Construction of the four code-

books took roughly 21 hours. However, the code was running on CPUs (actually on an eight-core machine). Since this is not the focus of this work, we haven’t yet implemented a parallel cluster version, nor a GPU version. Since codebook construction mainly consists of nearest-neighbor searches which can easily be parallelized, there is reason to assume that compression speed can be improved significantly.

Compressing a slice of $1608 \times 896 \times 4$ using the existing set of codebooks takes roughly 30 seconds. Thus, this second step adds about 10 hours to the overall compression time.

We give the image quality in terms of PSNR. All $4 \times 4 \times 4$ cubes which are completely background have not been included into the PSNR computation. The overall PSNR is about 27dB. An example of an original image versus the decompressed image is shown in Fig. 1b and c. Both images form a stack of four 64×64 pixel cut-outs of the same image portion. Fig. 1a shows a codebook on level 0.

The result of the compression step is an array of $402 \times 224 \times 1297 = 116,792,256$ index sets of 48 bits each, for total size of 700,753,536 Bytes. Thus, the entire compressed data set along with the codebooks fits on a graphics card with 1GByte of video memory. In this work we only consider the case that the compressed dataset fits entirely into the video memory. Otherwise swapping from main memory or even hard disk would be required, which, however, would also benefit from the high compression rate.

3.2 Rendering

In general, rendering is done by repeatedly decompressing subvolumes of the compressed data set into an intermediate 3D-texture in video memory, rendering this 3D-texture using a raycaster, and blending the resulting images. The raycaster we use is supplied with the SDK from NVidia. Classification using a 3D lookup-table, and optionally gradient extraction and shading, are integrated into the decompression step in order to not slow down the raycaster. Early-ray-termination is included on a per-ray basis in this raycaster, we added early-exit on a per-subvolume basis using occlusion culling. Empty-space-skipping is applied to subvolumes after classification, i.e., whenever the visible contribution of a subvolume according to the actual transfer function is below a user-supplied threshold. Multi-resolution rendering can also be integrated in an elegant way.

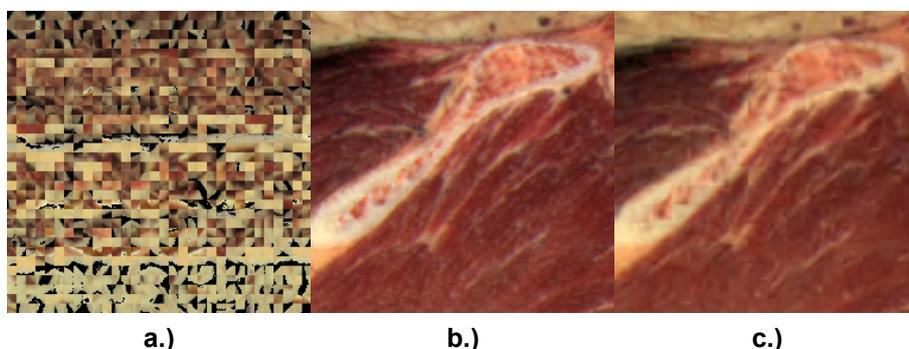


Fig. 1: a.) Codebook example on Level 0. b.) Original image. c.) Decompressed image.

We will now discuss the individual steps in more detail. A diagram depicting the overall flow is shown in Fig. 2.

3.2.1 Decompression

We have implemented the decompressor as a CUDA kernel, taking advantage of advanced features of the NVidia GPUs. Most notably, we make heavy use of the on-chip shared memory buffer. Processing is as follows.

Decompression is done in units of 256 index sets (worth 16k voxels), which are loaded into the shared memory. Each index set consists of 6 Bytes, which are unpacked into four 16-bit indices again into shared memory. For each voxel to be generated, there is one thread in the kernel. Each thread reads the unpacked index set, and fetches from memory those parts of the codevectors which it needs for its voxel. After unpacking the codevector components (from 11-11-10-format, see Section 3.1), and accumulation, the RGB-components of the voxel are written into shared memory.

These quantities are also used to access a 3D lookup-table which contains opacity (α) values. The α -value is again written into the shared memory, which completes the voxel generation. The system keeps track of the visible contribution of all voxels in a subvolume (color components multiplied by alpha), if the contribution of a subvolume to the final image is too low after classification, the subvolume is excluded from rendering (empty-space-skipping on the subvolume level).

When a certain number of threads have finished their voxel, the contents of the shared memory are written to video memory, that is, to the intermediate 3D texture.

By means of this process order we can make sure that memory transfers are mostly large bursts, and so bandwidth is high. Decompression performance is 1.86G voxels/s on the GTX280, and 0.60G voxels/s on the 8800GT.

Partitioning the volume into subvolumes always causes problems at the subvolume boundaries. During raycasting, the reconstruction filter (tri-linear interpolation) is missing voxels from the neighboring subvolume, during gradient extraction (see Section 3.2.2), the kernel hits the same problem. Most often, this problem is solved by using overlapping subvolumes, and we adopt this method. Each subvolume is extended in x-, y- and z-direction by two layers of $4 \times 4 \times 4$ voxels. Net subvolume size was chosen to be 128^3 , and so final subvolume size is $136 \times 136 \times 136$. The added overhead is about 20%. Decompression performance for different levels of detail is summarized in Table 2.

3.2.2 Gradient Extraction and Shading

Once a given subvolume has been decompressed, the system can optionally perform gradient shading. In this work, we derive the gradient from the opacity, since steep changes in opacity represent the surfaces of regions of interest.

GPU	Level	Voxels/s	Subvolumes/s
8800GT	0	0.60G	254
GTX280	0	1.86G	776
8800GT	1	0.20G	716
GTX280	1	0.72G	2463
8800GT	2	0.03G	841
GTX280	2	0.06G	1667

Table 2: Decompression Performance

For smooth surfaces, we use a variant of a $3 \times 3 \times 3$ Sobel filter (see Fig. 2). Two problems need to be addressed, however:

- high computation costs due to the large kernel,
- a certain amount of noise still in the image.

We solve both problems by using downsampled versions of the subvolume for gradient estimation (see also section 3.2.5, Multi-Resolution Rendering). The system generates two additional levels of detail: a 68^3 , and a 34^3 subvolume. The gradients are computed only on the lowest-resolution grid, again using a CUDA-kernel. Performance is given in Table 3.

GPU	Gradients/s	Subvolumes/s
8800GT	17.8M	570
GTX280	63.9M	2045

Table 3: Gradient Estimation Performance

Gradient extraction and shading are done at the voxel positions, the contributions from specular reflection are added to the just decompressed RGB-quantities. Thus, the operation of the raycaster is not at all affected by the shading operation, and is therefore not slowed down. On the other hand, decompression speed does not suffer too much because of the still regular memory access pattern.

For gradient extraction the system can use a Central Difference (CD) operator, or a Gradient shading is again implemented as a CUDA-kernel. As before, each thread processes one voxel. Each thread reads a certain subset of the required voxel neighborhood, so that by the end of this step a large block of voxels resides in shared memory.

To speed up shading we assume light sources at infinity, and a constant viewing direction throughout the volume (only for the shading, not for the raycasting). Thus, the halfway vectors are all constant, and don't need to be computed for each voxel. It is true that the placement of the highlights will be incorrect, however, such artifacts are rarely disturbing. Exponentiation is done by a look-up in a precomputed table.

Gradient shading speed for CD and one light source is summarized in Table 4.

GPU	Level	Voxels/s	Subvolumes/s
8800GT	0	0.227G	111
GTX280	0	0.411G	201
8800GT	1	0.309G	966
GTX280	1	0.546G	1712
8800GT	2	0.271G	7426
GTX280	2	0.482G	13158

Table 4: Gradient Shading Performance (CD)

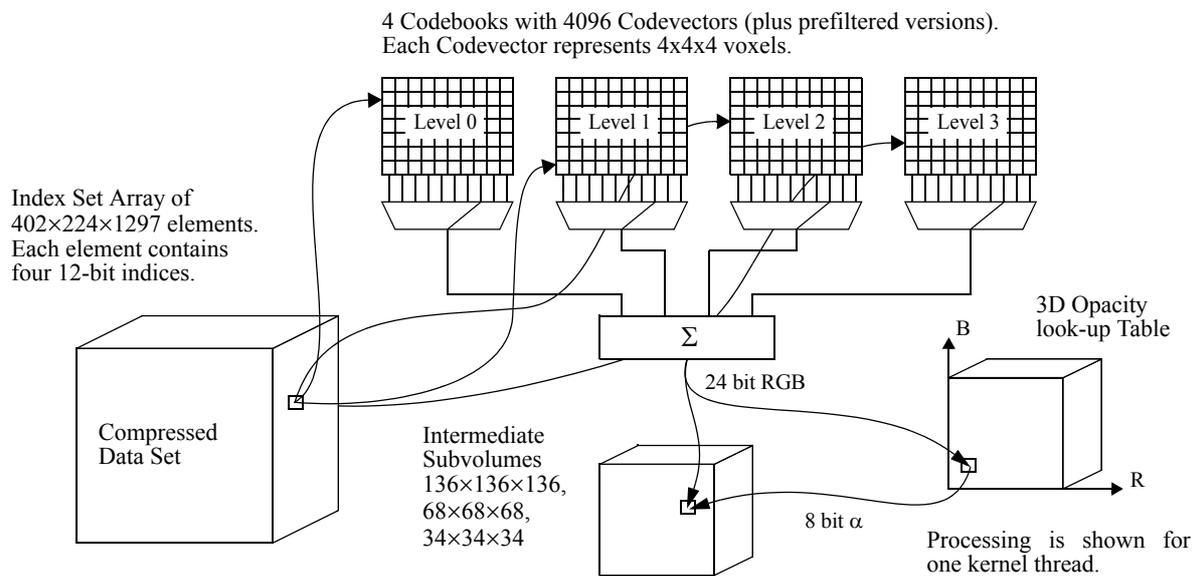
3.2.3 The Raycaster

As previously mentioned we use the raycaster in the SDK from NVidia. Since it is not the focus of this work, no attempt was made to optimize this code.

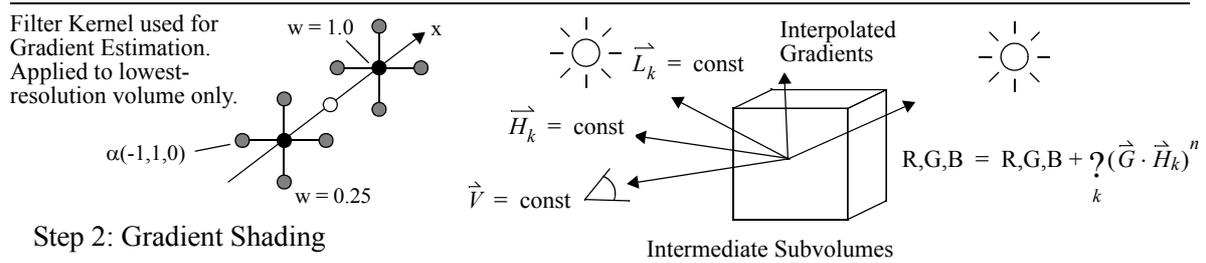
On the GTX280 graphics card, the rendering time of this raycaster was measured to be an average of 3.97ms per subvolume (early-ray-termination disabled), and so to be about 4 times slower than the pure decompression. Thus, rendering time is largely dominated by raycasting; the time spent in the recurring decompression can be tolerated fairly well.

3.2.4 Blending and Occlusion Culling

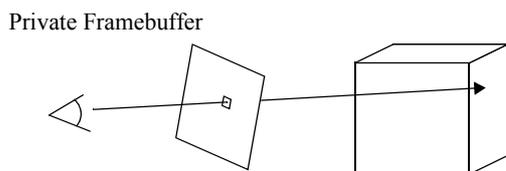
The subvolumes are rendered in front-to-back order, according to their Manhattan Distance to the viewer. The result of the rendering of one subvolume is a private frame buffer of RGB α -values. This buffer is α -blended with the compound frame buffer, which in the end contains the final image.



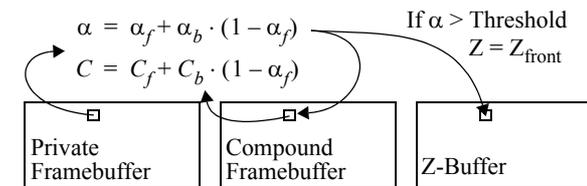
Step 1: Decompression & Classification



Step 2: Gradient Shading



Step 3: Raycasting



Step 4: Alpha-Blending & Z-Buffer Update

Fig. 2: Process Flow Diagram.

During blending, the system also updates a Z-buffer. Whenever the α -value of a pixel in the compound frame buffer exceeds a threshold, the corresponding entry in the Z-buffer is set to Z-front. This is then used to exclude subvolumes which are occluded by opaque structures in the data set from decompression and rendering. To this end, an occlusion query is submitted with the bounding box of the subvolume, which returns the number of visible pixels. Depending on a user-defined threshold, the subvolume is rendered or rejected.

Occlusion queries can be accelerated by submitting a batch of bounding boxes. In our system, all subvolumes with the same Manhattan Distance could be rendered in parallel and in any order, so they are queried in one batch. Subvolumes which are located at any of the visible faces of the entire volume will be rendered in any case and are excluded from occlusion query.

3.2.5 Multi-Resolution Rendering

To avoid subsampling of the data during raycasting, and to speed up rendering of distant subvolumes, the system can generate decompressed subvolumes at different resolutions. Here we can take advantage of the fact that a downsampled version of the subvolume can be generated from a *downsampled version of the codebooks*. Thus there is no need to keep a separate index set array for each level of detail in video memory, all we need is a small amount of extra memory for the downsampled codebooks.

The system supports decompressed subvolumes with 136, 68, and 34 voxels along each axis. By the use of normalized texture coordinates, the raycaster automatically performs proper voxel access and filtering. Only the opacity must be adjusted, we accomplish this by using a separate 3D look-up table for each resolution.

During rendering, the proper resolution of each subvolume is selected according to the raypoint spacing on neighboring rays, or, of course, according to user input.

3.2.6 Parallel Rendering on the Cluster

In general, the work can be partitioned in two ways: object-space partitioning (OSP), and screen-space partitioning (SSP). In OSP, a workpackage consists of a subvolume. This is rendered to a private frame buffer including the alpha-channel. Since a subvolume can project to any part of the global frame (spanning all displays), at least a subset of pixels need to be sent over the network for blending into the local compound frame buffer at the receiving node. Depending on how many subvolumes contribute to a given screen pixel, each final pixel may have caused multiple transfers over the network. Although sophisticated schemes have been developed to optimize this operation [14],[23], this pixel traffic still represents a severe bottleneck, especially over relatively slow GBit Ethernet. The upside is, though, that any subvolume is processed at most once.

In SSP, a workpackage consists of a rectangular region on the screen (a *tile*). A machine having been assigned a certain rectangle renders this tile to completion, and sends the final pixels to the destination

screen. Since the viewpoint can be at an arbitrary location, each node needs a complete copy of the data. Most obviously we have selected this method, and use the RVQ-compression to fulfil this requirement.

In SSP, the view frustum of a given tile can intersect a number of subvolumes, which contribute only partially to the tile pixels. Such subvolumes need to be processed again for neighboring tiles, which introduces a certain overhead. Since the raycaster will not process rays redundantly, but decompression will only generate complete subvolumes, the overhead mainly consists of redundant decompression (plus redundant occlusion queries). Since the decompression is very fast, we opted for sacrificing GPU cycles in favor of reduced network traffic.

Tiles are assigned dynamically on demand. Thus, there is a scheduling thread in the system which hands out tiles to requesting nodes. As a further optimization, each requesting node first gets tiles from its own display.

4. PERFORMANCE

A photo of the display wall showing a rendering of the Visible Human Female is shown in Fig. 3. The alpha-threshold for occlusion culling was set to 0.95. Renderings like these rotating around the z-axis take an average of 3.9 seconds per frame. If using only downfiltered subvolumes, average rendering time decreases to 2.8 or 2.3 seconds per frame, for 68^3 and 34^3 subvolumes, respectively. Tests have shown that image completion time is reduced by roughly 50% if empty pixel packets are transferred, i.e., only synchronization messages are sent. This confirms our choice of rendering mode, since the network is already saturated with this minimal amount of pixel data. We have included lossless image compression before sending tile pixels (a LZW-variant), but coincidentally the reduced transmission time was exactly offset by compression and decompression times. Thus, reducing network overhead remains a research topic in this project.

5. CONCLUSIONS

We have presented a parallel volume graphics system for rendering very large data sets on a high-resolution display wall. It provides the following features:

- Compression of the data set down to 0.75bpp, thereby enabling data set replication on all nodes, as well as placement of large data sets entirely in fast video memory,
- fast and simple decompression, entirely handled by the GPU,
- on-the-fly classification and gradient shading,
- empty-space-skipping on a per-subvolume basis,
- occlusion culling on a per-subvolume basis,
- multi-resolution rendering,
- and parallel rendering with screen-space partitioning.

The system can render our compressed version of the Visible Human Female at interactive speed. Still fine details of the anatomy, such as thin blood vessels, are preserved.

Future work will be directed at improving the image quality, and at increasing the rendering speed. Improving the image quality is not a matter of better rendering in the first place, but of better compression. Thus we will try to increase the PSNR and alleviate the block artifacts in the decompressed image. As a first step the use of even larger codebooks will be investigated, which will most likely not affect rendering speed.

The latter can still be improved by further optimizing the CUDA kernels. It is not always intuitive which codes lead to a speed-up and why. Thus, kernel optimization often means time-consuming try-and-error. We are confident, however, that significant speed gains can still be achieved.

6. REFERENCES

- [1] K. Akeley, "RealityEngine Graphics", Proc. ACM Siggraph 93 Conference, pp. 109-116
- [2] C. F. Barnes, S. A. Rizvi, "Advances in Residual Vector Quantization: A Review", IEEE Trans. on Image Processing, Vol. 5, No. 2, 1996
- [3] B. Cabral, N. Cam, J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware", Proc. ACM Symposium on Volume Visualization 1994, pp. 91-97
- [4] U. Cullip, U. Neumann, "Accelerating Volume Reconstruction with 3D Texture Hardware", UNC Tech Report TR93-0027, 1993
- [5] K. Engel, M. Kraus, T. Ertl, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading", Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2001
- [6] J. E. Fowler, R. Yagel, "Lossless Compression of Volume Data", Proc. ACM Symposium on Volume Visualization 1994, pp. 43-50
- [7] S. Guthe, M. Wand, J. Gonser, W. Strasser, "Interactive Rendering of Large Volume Data Sets", Proc. IEEE Visualization Conference 2002, Boston, MA, pp. 53-60
- [8] M. Hadwiger, C. Berger, H. Hauser, "High-quality two-level volume rendering of segmented data sets on consumer graphics hardware", Proc. IEEE Visualization Conference 2003, pp. 301-308
- [9] I. Ihm, S. Park, "Wavelet-based 3D compression scheme for very large volume data", Proc. Graphics Interface 1998, pp. 107-116
- [10] B. H. Juang, A. H. Gray, "Multiple Stage Vector Quantization for Speech Coding", Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, Vol. 1, Apr. 1982, pp. 597-600
- [11] J. Kniss, G. Kindelmann, C. Hansen, "Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets", Proc. IEEE Visualization Conference 2001, pp. 255-262
- [12] G. Knittel, "High-Speed Volume Rendering Using Redundant Block Compression", Proc. IEEE Visualization '95 Conference, Atlanta, GA, October 29 - November 3, 1995, pages 176-183
- [13] S. P. Lloyd, "Least squares quantization in PCM", IEEE Trans. on Information Theory, Vol. 28, 1982, pages 129-137
- [14] K.-L. Ma, J. S. Painter, C. D. Hansen, M. F. Krogh, "Parallel volume rendering using binary-swap compositing", IEEE Computer Graphics and Applications, Vol. 14, No. 4, 1994, pp. 59-68
- [15] M. Meissner, U. Hoffmann, W. Strasser, "Enabling Classification and Shading for 3D Texture Mapping Based Volume Rendering", Proc. 10th IEEE Visualization Conference 1999 (VIS '99), p. 32
- [16] K. Moreland, D. Thompson, "From Cluster to Wall with VTK", IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003, October 20-21, 2003, Seattle, Washington, USA, pp. 25-31
- [17] K. Nguyen, D. Saupe, "Rapid high quality compression of volume data for visualization", Computer Graphics Forum 20, 13, 2001
- [18] P. Ning, L. Hesselink, "Vector Quantization for Volume Rendering", Proc. ACM Workshop on Volume Visualization 1992, Boston, MA, pp. 69-74
- [19] NVIDIA Corporation, "CUDA Zone", http://www.nvidia.com/object/cuda_home.html#
- [20] F. Rodler, "Wavelet based 3D compression with fast random access for very large volume data", Proc. Pacific Graphics 1999, pp. 108-117
- [21] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, W. Strasser, "Smart Hardware-Accelerated Volume Rendering", Proc. EG/IEEE TCVG Symposium on Visualization 2003, pp. 231-301
- [22] J. Schneider, R. Westermann, "Compression Domain Volume Rendering", Proc. 14th IEEE Visualization Conference 2003 (VIS'03), p. 39
- [23] A. Stoppel, K.-L. Ma, E. B. Lum, J. Ahrens, J. Patchett, "SLIC: Scheduled Linear Image Compositing for Parallel Volume Rendering", IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003, October 20-21, 2003, Seattle, Washington, USA, pp. 33-40
- [24] M. Strengert, M. Magallon, D. Weiskopf, S. Guthe, T. Ertl, "Hierarchical Visualization and Compression of Large Volume Datasets Using GPU Clusters", Proc. EG Symposium on Parallel Graphics and Visualization 2004, pp. 41-48
- [25] United States National Library of Medicine, "The Visible Human Project", http://www.nlm.nih.gov/research/visible/getting_data.html

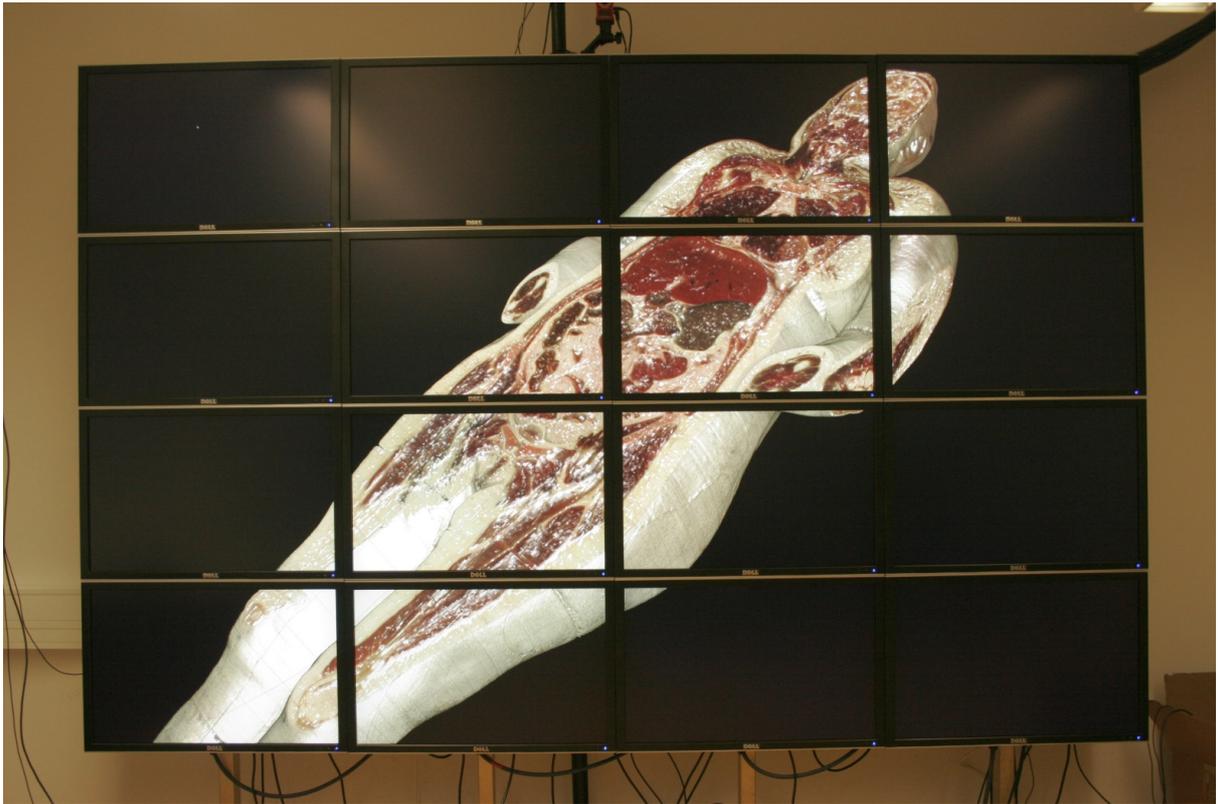


Fig. 3: The Display Wall.

Reconstructing Indoor Scenes with Omni-Cameras

Frank Bauer

Martin Meister

Marc Stamminger

University of Erlangen-Nuremberg
Am Wolfsmantel 33
91058 Erlangen, Germany

{frank.bauer, martin.meister, marc.stamminger}@informatik.uni-erlangen.de

ABSTRACT

We present a system similar to Debevec's *Facade* [DTM96] that improves the reconstruction of indoor scenes from photographs. With confined spaces it is often impractical to use regular photos as the base of the reconstruction. Combining pinhole cameras with fisheye shoots or photographs of any kind of reflective, parametrisable body such as light probes eases this problem. We call the later camera setup an omni-camera, because it enables us to acquire as much information as possible from a given viewpoint. Omni-cameras make it possible to reconstruct the geometry of an entire room from just one view. Removing the pinhole camera constraint invalidates some key assumptions made in *Facade*. This paper shows how to work around the problems arising from this approach by adding scene specific knowledge as well as a genetic component to the solver. When using omni-cameras we can no longer take advantage of a simple texture projection to obtain the materials for the scene. Instead we propose a method for texture generation that is transparent to the camera setup used.

Keywords

Image-Based, Modeling, Reconstruction, Architecture, Omni-Camera

1. INTRODUCTION

The reconstruction of shape from photographs is one of the fundamental problems of computer vision and computer graphics. It is used either to model important present-day landmark architectural scenes and famous buildings as well as in cultural heritage projects with scientific background, e.g. aiming at digitally preserving a present state of conservation. In recent development, textures and models produced by systems as the one detailed in this paper can be used as the base to retrieving grammars for procedural modelling approaches [MZWG07].

Image-based scene reconstruction under general circumstances with no a priori knowledge of the position of cameras or any constraints on the geometry of the real scene is an ill-posed problem. Several approaches have been proposed, dealing with a subset of unknown and known factors. For very densely sampled scenes traditional light field renderers can give a very good approximation of the model and the reflectance. Global proxy geometries are extractable by shape-from-silhouette methods from the visual hull [GGSC96]. In most cases however, these methods are

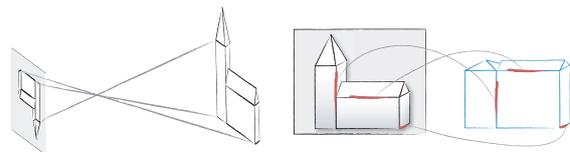


Figure 1: *left* Pinhole camera model; *right* Setting up correspondences

only applicable to turntable setups of objects rather small in size.

Another approach to the same problem is illustrated in [KS00]. The positions of the cameras are known and the underlying unknown scene is reconstructed by a volumetric approach that discards all voxels that are not mapped photo-consistently in all images. The algorithm works well for lambertian scenes with an extension to more complex colour models possible. Global effects such as shadowing, transparency and inter-reflections must be ignored and cannot be modelled.

Modelling from a sparse set of photographs requires additional constraints on the reconstruction algorithm. One feasible way was presented in 1996 [DTM96] where a user has to define a crude box-based geometry (the **base model**) and manually find correspondences between features in the images and features in that base model. The reconstruction features used are edges in the source camera images. When dealing with situations where some parts of the model cannot be seen, symmetries of the model are exploited to re-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

trieve information about the hidden surfaces or blocks. Symmetries are also useful to reduce the dimension of the reconstruction problem.

2. FACADE

Facade was restricted to a pinhole-camera model (see Figure 1, left). Constraints imposed by this choice were closely interwoven into the algorithms presented. For example, setting up the feature correspondences (see Figure 1, right) usually works like this:

1. Mark two points on an edge in a source image (**projected edge**)
2. Construct a ray through the cameras focal point and each of the previously marked points (point rays)
3. The focal point and the two point rays construct a plane (**reconstruction plane E**)
4. The user chooses an edge (**source edge**) in the base model and links it to the projected edge

The rays spanning the reconstruction plane are called **reconstruction rays \vec{r}** in this paper. We need to find the translation T_K and rotation R_K for each camera in order to determine the position of the reconstruction plane in world-coordinates. Using the fact that this plane should contain the source edge we can derive some simple formulas to find the camera rotation matrix R_K (see Figure 2).

Most edges in an architectural scene are axis aligned, so we know that the reconstruction plane of those edges should be parallel to a given axis d_{BK} . In other words the plane normal vector \vec{n}_E has to be perpendicular to that axis. This gives the equation

$$\vec{n}_E * R_K * d_{BK} = 0$$

Having multiple edges that are parallel to different axes it is possible to build an objective function we can use to obtain an initial estimate for the camera rotation (for details see Section 5.1).

When the camera rotation is known the translation T_K is simultaneously reconstructed with all other parameters of the scene (size, location and rotation of the blocks in the base model) in [DTM96]. When no rotated block is present, the resulting functions are linear and a result can be computed.

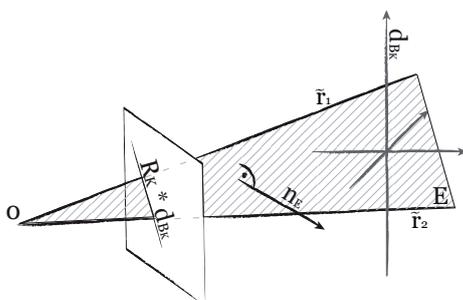


Figure 2: Camera rotation estimation

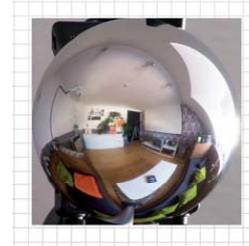


Figure 3: Photo of a Light Probe

With rotations the task is not as easy. In *Facade* the user needs to give an initial estimate for them. Since the resulting functions are no longer linear, an objective function is generated once again and solved with a Newton-Raphson algorithm (for details see Section 5.2).

At this stage we know the location and rotation of each camera in world-space as well as every parameter that defines our reconstructed model. The cameras in world-space are used to project their corresponding source images onto the blocks in the scene, allowing the software to impose view dependent texturing onto the scene in a very simple fashion, all possible because of the pinhole camera constraint.

3. OUR CONTRIBUTION

When shooting scenes inside we have to deal with confined spaces, where it is not feasible to take an overview picture capturing more than a part of a wall without using wide-angle lenses. To resolve this restriction we tried to use omni-camera setups like the photograph of a light probe (Figure 3).

We introduced new problems when building the reconstruction planes by removing the pinhole constraint *Facade* relies on (see Section 2). Non-skew rays reflected on the sphere for example are skew in general. Approximating a plane with those rays has the effect, that the resulting reconstruction plane does not contain the source edge. This renders the Newton-Raphson optimizers used by *Facade* less stable. We propose some additional enhancements to circumvent the loss in robustness in Section 5.3.

Marking a projected edge in the camera images is also no longer straight forward, as the source edge projects onto a curve in some setups.

Since the projective texturing relied on the pinhole camera model we can no longer use it for our omni-camera setups. Instead we propose a simple ray casting approach, as detailed in Section 6. This enables a texturing process independent of the camera setup used. We employ the textures to export a complete scene for use in other modelling or rendering applications or as a block replacement (Section 7) in a more complex scene.

When reconstructing an indoor scene we need rotated Blocks more frequently than with regular architec-

tural outdoor settings. That circumstance forces us to use non-linear optimization for almost every scene we reconstruct, slowing down the process. We propose some enhancements to the classic *Facade* approach (Section 5.1) resulting in lower reconstruction time and improved robustness that compensates for the error introduced by approximating the reconstruction plane.

The advantage of an omni-camera is obvious. With a light probe setup we can gather almost a 360-degree view of our room with only one shoot, often allowing us to reconstruct the geometry of the scene from one photograph.

In order to give the scenes some more depth and allow easy incremental modelling, we also introduce the use of block replacements to our system as discussed in Section 7.

4. SCENE PREPARATION

Preparing a scene for reconstruction is predominantly independent of the camera system used. We will highlight everything that is different for sundries setups or whenever the classic *Facade* setting is not applicable to our omni-cameras.

4.1. Photographs

With a standard camera setup we have to consider some constraints that arise from the pinhole assumption. We need to set long focal lengths and a big aperture value to gather results that match a photo taken by an ideal pinhole camera as much as possible.

When shooting a light probe setup we direct the camera towards the mirror ball in such a way that the centre of the mirror ball lies on the optical axis of that camera. The diameter of the light probe should be as small as feasible compared to the focal length. We generally work with focal lengths of $450mm$ and sphere diameters of 80 to $150mm$. With this setup the camera and its supporting tripod obscure as little space on the image as possible.

In a pre-process we mask unwanted geometry in the obtained photographs. Omitting this step would result in the camera and the tripod to get projected on the textures in the final step.

4.2. Base Model

Every object in the scene has to be represented by a crude approximating block (like a cube or a ramp) defined by a type and a set of numeric parameters (like width, height...).

By using constraints on the blocks (like symmetries) we can reduce the number of parameters that have to be determined during the reconstruction.

We would like to point out, that in contrast to *Facade* our system does not rely on a crude approximation for

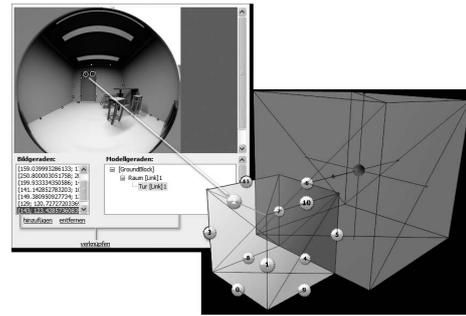


Figure 4: Link a curve in the source image to an edge in the model. The crude model shows the cube for the room itself and a door.

the block parameters given by the user. Our optimization to the reconstruction process makes it more robust than the original approach.

4.3. Adding Cameras

We have to create a camera for every taken image. Our software allows us to mix cameras of different types. We found that it simplifies the reconstruction process if we use the omni-setups to reconstruct the geometry of the room and its objects as well as a first and very crude texture. Regular photographs are then used to refine the visual quality of the result by adding additional images in a later iteration.

For each camera our system needs to know the following intrinsic parameters: camera type (regular pinhole, light probe setup...), film size and focal length. In case of a light probe setup we additionally need the diameter of the mirror ball and the distance of the focal point to the balls centre.

If the images were taken digitally we can use the EXIF information stored to automatically determine the film size (by camera model) and the focal length. If the user specifies the radius of the sphere we can also automatically compute the distance between sphere and camera (assuming the sphere completely fills the photograph).

5. SCENE RECONSTRUCTION

We need to set up correspondences before we can reconstruct the camera transformation matrices R_K and T_K or any other parameter. Section 2 already explained how this is done for the pinhole camera. It also detailed that we need to construct a reconstruction plane for the *Facade*-algorithms to work properly.

When using a light probe setup, the user has to perform the same basic steps. In this case however a line in world space projects to a curve in image space. Marking two points on that curve is still enough to identify a straight edge in world-space (see Figure 4).

For every point marked in the image we construct point rays $\vec{d}_c = \overline{B_K F}$ (see Figure 5). When using light probes the point rays are not equal to the reconstruction rays \vec{r} , as they do not intersect with the source

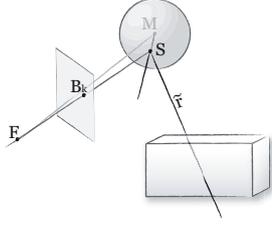


Figure 5: Constructing the reflected ray from a selected point B_K in the image plane.

edge in global space. To obtain the reconstruction rays we have to reflect them on the surface of the mirror ball using:

$$\begin{aligned}\vec{n}_s &= \frac{M - S}{\|M - S\|} \\ \tilde{r} &= \vec{d}_c - (2 * (\vec{d}_c \circ \vec{n}_s) * \vec{n}_s)\end{aligned}$$

As the rays \tilde{r} are not guaranteed to be non-skew we cannot create the plane E from them as easy as it is done at this point in the process for the pinhole camera. With the pinhole model we were able to use the focal point (where the reconstruction rays intersect) and the direction of the two reconstruction rays to construct the reconstruction plane E .

We still use the directions of our two reconstruction rays, but since they are skew, they do not intersect in one point. We decided to use the average of the two starting points of our reconstruction rays as an approximation of an intersection point.

In contrast to the reconstruction planes obtained by the pinhole model the orientation of this plane varies depending on the points we select on the projection of the edge in the source image. This obviously will be a problem for a robust reconstruction. By adapting the optimization strategies introduced in *Facade* we can still obtain very good results, as we will describe in detail in 5.1 and 5.2.

5.1. Camera Rotation

We proceed similar to the way suggested by Debevec in [DTM96] by finding an appropriate objective function $O = \sum(\text{Err}_i)^2$ using the pseudo reconstruction plane E instead of the ones described in the original work. We use the square of this sum to better fit the Newton-Raphson method that is used throughout the paper.

An error or disparity function Err_i can be set up to calculate the rotation of each camera separately. The camera rotation R_K is processed in an upstream task, to reduce the number of parameters that have to be estimated, and to separate the error prone optimization of the Euler rotations from the rest of the reconstruction. For each correspondence we have one pseudo reconstruction plane E . Together with the Euler camera rotation matrix R_K^{-1} we can use that plane to formulate

the disparity function Err_i . We choose R_K^{-1} such that the normal \vec{n}_E of the reconstruction plane is perpendicular to the desired direction \vec{d}_{B_K} of the source edge in the model. This corresponds to a rotation of the camera around its pivot using R_K^{-1} (see Figure 2).

$$\text{Err}_i = (\vec{n}_E \circ (R_K * \vec{d}_{B_K}))^2 \quad (1)$$

As we explained in the previous chapter, the pseudo reconstruction plane E is only an approximation of a plane that really contains the source edge. This renders the Newton-Raphson optimization less robust due to the additional error. In order to compensate, we propose the use of a genetic algorithm to automatically find crude initial values for the rotation matrix. The one we used is a genetic algorithm based on an elite selection strategy without crossovers [Mit98].

Every generation contains 1000 individuals. Each is composed of the three Euler angles that determine the rotation of one camera. The initial population samples the cameras rotational space around its coordinate axes equidistantly (10 degrees). Each individual is assigned a quality, which corresponds to the square of the evaluation of the error function Err_i using the angles specified by that individual.

With each iteration a new generation is created containing the best 60% of parent individuals, and 40% newly created ones. The new ones are built based on the values of a chosen parent (an individual with high quality is more likely to be chosen). Those values are changed using a Gaussian distributed mutation. The distribution is adapted by decreasing the variance of the Gaussian in approximately every 20th generation to achieve a very dense sampling around the individuals of later generations. The iteration is stopped if the best individual of a generation meets a predefined criterion, or the 500th generation was spawned.

We find that the final result (the camera rotation) of this genetic procedure is only marginally improved by the following Newton-Raphson optimization.

5.2. Translation and Model Parameters

To obtain the global model parameters and the camera translations, we have to build another objective function, that represents the distance of each edge from the model to their corresponding reconstruction plane E (see Figure 6) in world-space.

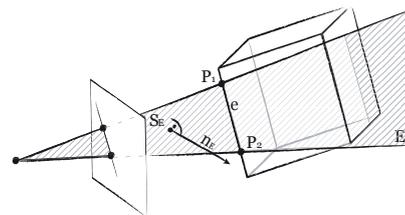


Figure 6: Camera Translation and block parameters

For two points P on a source edge e we can calculate that error by

$$d(e) = \vec{n}_E \circ (((R * (P - M)) - S_E) \quad (2)$$

where S_E is an arbitrary point on the reconstruction plane. This is a slight difference compared to the implementation in [DTM96]. We also have to take into account, that our reconstruction plane E is not as stable as the one used in a pure pinhole setup as we explained in Section 5.1.

Using Equation 2, we can calculate the objective function for all images I and all line correspondences e_I in that image by:

$$O = \sum_I \sum_{e_I} d(e_I)^2 \quad (3)$$

Minimizing this function yields the reconstruction of the scene.

5.3. Improving Reconstruction Results

If all rotation matrices R_B for all blocks in the scene are constant, we can compute the solution by solving a linear equation system, which is simple and often applicable when working with outdoor architecture. However, we found that indoor scenes tend to have a higher quantity of rotated blocks.

Since the evaluation of a Newton-Raphson algorithm can become slow, and might get caught in local minima, we decided to split this process into two separate tasks.

First we automatically select all base geometry blocks that are rotated around a known angle (this, of course, includes blocks not rotated) along with all camera translations. We can solve the resulting linear equation system comprised of the square of the distances $d(e)^2$ define in equation 2 for all correspondences relating to blocks not rotated.

As a result the camera translation and the fixedly rotated blocks are now consistently set up. Only the parameters of blocks with an unknown rotation remain unset. We build the same objective function as described in equation 3 for all unset edges. Since we do not include edges already computed in the linear step, the dimension and complexity of the objective function O is reduced.

At first we used a standard Newton-Raphson implementation that had to re-evaluate the hessian symbolically in each step. This proved to be a very slow solution, as the terms that had to be optimized were rather complex, and we had to tackle with big hessian matrices comprised of the second derivate of those terms.

We changed that implementation to a quasi Newton-Raphson algorithm, as described in [BNS94]. This method has the advantage that we never have to compute the hessian matrix, but can calculate an estimate for it through the gradients of the objective function.

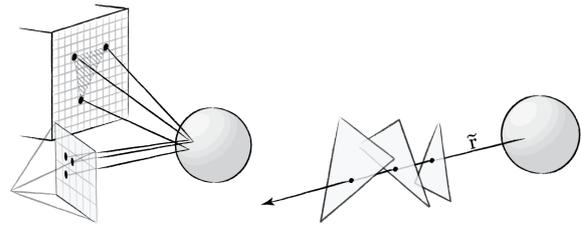


Figure 7: *left* Projecting source pixels into the reconstructed scene; *right* Only the first surface hit by a ray gets textured.

As a side note we would like to point to the fact that the optimization over the $SO(3)$ group [Kue03] that is done for all camera and block rotation matrices is often not enough. Special care has to be taken since ambiguities can occur (for example positive or negative view directions) that can either be resolved by user-interaction, reparameterization of the rotational domain or specially adapted tests [ML03].

Our software automatically fixes the camera view direction by checking if the intersection of any given reconstruction ray with the model object is located in the expected octant of the cameras coordinate system.

In case of a light probe, the expected octant is determined by the location of the point that corresponds to the reconstruction ray on the light probe. For a regular pinhole setup we simply check if the intersection is located in front of or behind the camera.

In addition we added an extra phase into the Newton-Raphson optimization that is evaluating the objective function for different angles after each iteration.

After the result for one iteration is calculated, we change the variables that represent angles in 45-degree steps and calculate the error value achieved with the altered angles. If the result is smaller than the one obtained through the optimization step, we will use the new angles in the following Newton-iteration.

This alleviates the user from the need to set an approximate rotation for any block before starting the reconstruction, as it was necessary with *Facade*. All in all our changes made the reconstruction of rotated blocks and the use of pseudo reconstruction planes more reliable.

6. TEXTURE GENERATION

With our omni-cameras we can no longer use the simple projective texturing approach, as it is used in *Facade*. We propose to use a ray-casting algorithm to render the textures for a scene. This creates a layer of abstraction between the camera model used and the texturing process.

We build a reconstruction ray for each pixel in each source image, constructing it the exact same way as it is done for the points marked by the user during the scene reconstruction. Reutilizing the code generating

the reconstruction rays makes this process transparent to the underlying camera type.

By intersecting those rays with our scene geometry we obtain a list of points in world space that are visible through a given pixel. Only the closest intersection in front of the camera is coloured with the colour of the source pixel (see Figure 7, right). We choose to use forward ray tracing, because it is not always possible (depending on the camera type used) to find a unique ray from world space into the source images. Think about the contour of a sphere. At this singularity, one point in world-space is mapped to all points on that contour.

To write colour values to the textures we apply a linear interpolation of three projected neighbouring camera rays to fill larger texture regions by rasterizing the resulting triangle in object texture space (Figure 7 left). The alpha mask provided for each image in the pre process can be used to blend out regions that are defective. Furthermore, we calculate the scalar product of the ray with the surface normal and weight the incoming texels accordingly (similar to the blend field methods in [BBM⁺01]). This ensures that rays with grazing angles contribute little to the resulting textures.

Another way for the user to interact with texture generation is to select a global weight for each texture. Especially in the presence of regular pinhole source images it may be advisable to discard any texture information from the reflective sources where more detailed source pictures are available. The pinhole source images have a far better local resolution and produce an increased local texture quality. In Figure 11 such detail images can be seen for the white radio on top of the shelf and the cupboard on the floor.

Of course there are problems with regions that are not seen from any of the source images. As they are never hit by any ray, we can fill the missing texture regions with a blank colour (see the greyish colour in Figure 8 on the floor projecting away from the chairs), fill it by interpolation techniques from neighbouring texels or by utilizing a texture synthesis approach [WL00].

Using this approach to generate textures allows our system to export the result to a file (for example to VRML), making the reconstructed scene independent from a specialized viewer.

7. MODEL EXCHANGE

To alleviate the hassle for the user to work out seemingly unnecessary details, we provide for an easy model block exchange.

Instead of reproducing an indoor scene with every detail, we use a bounding box as base geometry for an object we want to replace with a more detailed version. After scene reconstruction is finished any table model from a 3D model library can be fitted into the scene by applying the affine transformation that is available

from the reconstruction process for each block. In Figure 8 the bounding boxes are shown over the replaced geometry.

Since we also allow the export of our scenes into arbitrary formats (like VRML or XSI), the model exchange can be used to build a complex scene iteratively. With this system it is convenient to first model some details of the scene using standard photographs for higher resolution. When finished the results are exported to a file and (if necessary) refined in an external editor.

In the next iteration we could start to model the room itself, representing the previous reconstructed detailed model with a simple block. When the reconstruction of the entire room is finished, we simply fit the model stored in the file into the bounding box of the block we created as a placeholder.

8. RESULTS

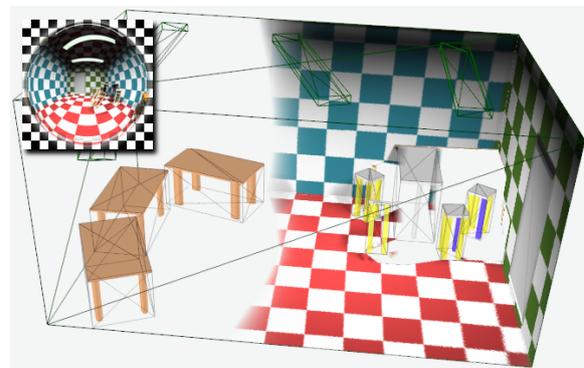


Figure 8: Reconstruction of a synthetic example from one light probe image.

We tested our implementation with four different scenes. The first one was a synthetic scene (Figure 8), to show the general usability of the algorithm. The scene was generated with Blender, using a near optimal light probe. Only half of the reconstructed room is textured, as the image resolution of the light probe is not high enough for the room behind the probe.

The second scene contained two offices (Figures 9 and 10). The third one was a home office (Figure 11) and the fourth a living room (Figure 12).

While a calibrated camera was used in [DTM96] to acquire the photos for the reconstruction, all our results were obtained with an uncalibrated one.

We compare the dimensions of the found parameters to the ones in the real scene, which gives us a measure for the quality of the reconstructed geometry. Since the reconstructed width of the scene is always set to 1.0, we have to multiply all our values with the actual width. However, the camera position and rotation in



Figure 9: Reconstruction of a small office from two light probes with 800x800 pixels each. This example shows the projection of an unmodelled chair onto the desktop.

the real scene was not recorded, so we had to compare those results visually.

The most challenging scene was the home office scene (Figure 11), because the space in that room is quite confined. The floor has a footprint of $12.5 m^2$. Just considering the area of the room, that is more than $1m$ high, we get a size of about $9 m^2$. It was reconstructed using a simple cube and an additional ramp for the room model. In a room this small it would not be practical to use regular photographs for the reconstruction of the geometry.

Parameter	Measurement	Reconst.	Result
Width	2.70m	1.000	2.70m
Height	2.30m	0.857	2.31m
Length	4.65m	1.718	4.64m

Table 1: Comparing reconstructed parameters to real world measurements in the home office scene.

The results in Table 1 show, that the reconstructed dimensions are in good correspondence with the mea-



Figure 10: Another small office scene using a quite bumpy light probe.



Figure 11: Confined space ($9 m^2$) home office scene: 3D view of extracted textures from light probe and detail perspective images.

sured values. The size of the reconstructed scene is only 1 cm off the real values.

We would like to point out, that we used the foot of a lamp to reconstruct, which was a slightly flattened sphere. This demonstrates that our method can generate robust results for the geometry with suboptimal light probes. We gain this robustness through our extensions to the calculation of the reconstruction plane (see Section 5) where we calculate the average of the starting points of the skew reconstruction rays.

Using the deformed light probe the texture correspondence was not always given. This became most obvious, if the surface projects to the outer regions of the mirror ball (see Figure 11).

The reconstruction of the camera position was very precise. We put the hemisphere we used as the light probe on the door and the walls of the room. This position was reconstructed correctly.

In order to determine the influence of a deformed sphere on the reconstruction results, we used a non deformed light probe for the living room scene in Figure 12. The results for the geometry was only marginally better than the one in the home office scene, but the textures were in better correspondence (except in the outer regions of the mirror ball).



Figure 12: A living room reconstructed using 2 light probe images and several pinhole shoots

9. CONCLUSION

We showed how to reconstruct an indoor scene using only one or very few images by applying a well-known method to various omni-cameras. Texture generation can be largely automated and yields atlas maps for single objects or the whole scene exportable to any 3D graphics format. Crude base blocks can be substituted with complex 3D geometry reducing the amount of detail work for the user while enhancing quality.

Allowing the usage of non-pinhole camera setups revealed a series of difficulties with the existing approach we had to tackle. Most of them due to the additional error introduced by the pseudo reconstruction planes we have to use in omni-setups.

The use of a genetic algorithm to get a good approximation for the initial values of the camera rotation used in the quasi Newton-Raphson minimization proved to be very precise, and made the estimation more robust compared to the original approach using just a Newton-Raphson solver.

By splitting the following estimate of the camera translation and the model parameters in a step where all non-rotated blocks are computed and a step that minimizes the parameters for all rotated blocks, we gained a big advantage in terms of speed and precision. It also contributed to a more reliable reconstruction.

The most surprising result was, that our method could robustly reconstruct a scene using non-optimal spheres as light probes.

In the future, we would like to extend the presented system by automatic edge detection in the source images that would speed up the scene preparations. Reconstructing the illumination of the scene as detailed in [SHR⁺99] from the generated textured objects is another possible enhancement that would be very useful when importing external geometry, as it allows us to match the lighting of the loaded textures to the scene.

REFERENCES

- [BBM⁺01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 425–432. ACM Press / ACM SIGGRAPH, 2001.
- [BNS94] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.
- [DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 11–20, New York, NY, USA, 1996. ACM Press.
- [GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 43–54, New York, NY, USA, 1996. ACM Press.
- [KS00] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.
- [Kue03] Frank O. Kuehnel. On the minimization over $so(3)$ manifolds. Technical report, RIACS NASA Ames Research Center, 2003.
- [Mit98] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press, 1998.
- [ML03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. Technical report, Society for Industrial and Applied Mathematics, 2003.
- [MZWG07] Pascal Mueller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. 26(3), 2007.
- [SHR⁺99] H. Schirmacher, W. Heidrich, M. Rubick, D. Schiron, and H. Seidel. Image-based brdf reconstruction, 1999.
- [WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. pages 479–488, 2000.

A Novel System for Automatic Hand Gesture Spotting and Recognition in Stereo Color Image Sequences

Mahmoud Elmezain, Ayoub Al-Hamadi, Bernd Michaelis
Institute for Electronics, Signal Processing and Communications (IESK)
Otto-von-Guericke-University Magdeburg
Germany
{Mahmoud.Elmezain, Ayoub.Al-Hamadi}@ovgu.de

ABSTRACT

Automatic gesture spotting and recognition is a challenging task for locating the start and end points that correspond to a gesture of interest in Human-Computer Interaction. This paper proposes a novel gesture spotting system that is suitable for real-time implementation. The system executes gesture segmentation and recognition simultaneously without any time delay based on Hidden Markov Models. In the segmentation module, the hand of the user is tracked using mean-shift algorithm, which is a non-parametric density estimator that optimizes the smooth similarity function to find the direction of hand gesture path. In order to spot key gesture accurately, a sophisticated method for designing a non-gesture model is proposed, which is constructed by collecting the states of all gesture models in the system. The non-gesture model is a weak model compared to all trained gesture models. Therefore, it provides a good confirmation for rejecting the non-gesture pattern. To reduce the states of the non-gesture model, similar probability distributions states are merged based on relative entropy measure. Experimental results show that the proposed system can automatically recognize isolated gestures with 97.78% and key gestures with 93.31% reliability for Arabic numbers from 0 to 9.

Keywords

Gesture spotting, Gesture recognition, Pattern recognition, Computer vision, Application.

1. INTRODUCTION

The hand gesture recognition is an active area of research in the vision community, mainly Human-Computer Interaction (HCI). A gesture is spatio-temporal pattern which may be static, dynamic or both. The goal of gesture interpretation is to push the advanced human-computer communication to bring the performance of HCI close to human-human interaction. In the last decade, several methods of potential applications [Dey06a, Elm08a, Kim07a, Mit07a, Yan07a] in the advanced gesture interfaces for HCI have been suggested but these differ from one another in their models. Some of these models are Neural Network (NN) [Dey06a], Hidden Markov Models (HMM) [Elm08a, Elm08b] and Dynamic Time Warping (DTW) [Tak92a]. One main concern

of gesture recognition is how to segment some key gestures from a continuous sequence of motions. The gesture segmentation is also called gesture spotting. This is considered as a highly difficult process for two major problems, which arise in real-time gesture recognition system for continuous gesture to extract key gestures. The first problem is segmentation that means how to determine when a gesture starts and when it ends from hand motion trajectory. The second problem is caused by the fact that the same gesture varies in shape, trajectory and duration, even for the same person.

To overcome these problems, HMM is used in our system because it is capable of modeling spatio-temporal time series of gestures effectively and can handle non-gesture patterns. On the other hand, NN and DTW hardly represent the non-gesture patterns. Lee *et al.* [Lee99a] proposed an ergodic model based on adaptive threshold to spot the start and the end points of input patterns, and also classify the meaningful gestures by combining all states from all trained gesture models using HMM. Kang *et al.* [Kan04a] developed a method to spot and recognize

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright UNION Agency – Science Press, Plzen, Czech Republic.

the meaningful movements where this method concurrently separates unintentional movements from a given image sequences. Alon *et al.* [Alo05a] proposed a new gesture spotting and recognition algorithm using a pruning method that allows the system to evaluate a relatively small number of hypotheses compared to Continuous Dynamic Programming (CDP). Yang *et al.* [Yan07a] presented a method for recognition of whole-body key gestures in Human-Robot Interaction (HRI) by HMM and garbage model for non-gesture patterns. Mostly, previous approaches use the backward spotting technique that first detects the end point of gesture by comparing the probability of gesture models and non-gesture model. Secondly, they track back to discover the start point of the gesture through the optimal path using Viterbi algorithm [Law89a] and then the segmented gesture is sent to HMM for recognition. So, there is an inevitable time delay between the key gesture segmentation and recognition, where this time delay is not well for on-line gesture recognition.

To treat this problem, we propose a forward gesture spotting system that executes gesture segmentation and recognition simultaneously. The system recognize the isolated and key gestures for Arabic numbers (0-9) in real-time from stereo color image sequences by the motion trajectory of a single hand using HMM. To spot key gesture accurately, a sophisticated method of designing a non-gesture model is proposed, which is constructed by collecting the states of all gesture models in the system. The non-gesture model is a weak model for all trained gesture models where its likelihood is smaller than that the dedicated model for a given gesture.

The start and end points of gestures are based on the competitive differential observation probability value, which is determined by the difference of observation probability value of maximal gesture models and non-gesture model. The key gesture starts (ends) when the value of competitive differential observation probability changes from negative to positive (positive to negative). To reduce the states of the non-gesture model, model reduction which merges similar probability distributions states based on relative entropy is used [Cov91a]. Moreover, each isolated gesture number is based on 60 video sequences (42 for training and 18 for testing) and the continuous gestures are based on 280 video sequences for spotting key gestures and testing it. The achievement recognition rates on isolated and key gestures are 97.78% and 93.31% respectively. The organization of this paper is as follows; in section 2, hand tracking technique is introduced. Section 3 demonstrates the key gesture spotting system in three subsections. The experimental results are described in Section 4. Finally, Section 5 gives a few concluding remarks and refers to our future aims.

2. REAL-TIME HAND TRACKING

The hand is tracked in our system by mean-shift algorithm, which is a non-parametric (i.e. kernel) density estimator that optimizes a smooth similarity function to find the direction of the hand target's movement. We decide to use m -bin histograms as the representation of the object's color probabilities density function (pdf's), as they can satisfy the low-cost requirement of real-time tracking. $YCbCr$ color space is used, where Y channel represents brightness and (Cb, Cr) channels refer to chrominance. The segmentation of skin colored regions becomes robust if only the chrominance is used in analysis. Therefore, we ignore Y channel to reduce the effect of brightness variation and use only the chrominance channels, which fully represent the color information. The segmentation of the hand with complex background takes place using 3D depth map and color information, which is more robust to the disadvantageous lighting and partial occlusion. This is done using Gaussian Mixture Models (GMM). For more details, the reader refers to [Elm08a, Nie07a].

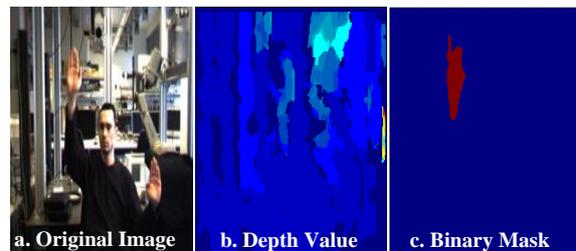


Figure 1. (a) First frame of video stream. (b) The Depth value of original image from a Bumblebee stereo camera. (c) Binary masked for left hand.

The segmentation module detects and localizes our object of interest (left hand) in the first video frame and we know exactly its position, as well as its shape and dimension (Fig. 1) [Elm08a]. Therefore, before starting with tracking, we used a binary mask to extract our hand target from the initial frame and find its color histogram with Epanechnikov kernel (monotonic decreasing kernel profile $k(x)$) [Com00a, Com03a, Sco92a] (Fig. 2(b)). Epanechnikov kernel assigns smaller weights to pixels farther from the center. Using these weights increases the robustness of the density estimation since the peripheral pixels are the least reliable, being often affected by occlusions. Let $\{x_i^*\}, i=1..n$ be the normalized pixel locations in the region defined as the hand target model. The probability of the feature $u=1..m$ in the hand target model histogram is computed as;

$$q_u = F \sum_{i=1}^n k(\|x_i^*\|^2) \delta[b(x_i^*) - u] \quad (1)$$

where δ is the *Kronecker delta* function, equal to 1 only at u and 0 otherwise. The normalization constant

F is derived by imposing the condition $\sum_{u=1}^m q_u = 1$, where

$$F = \frac{1}{\sum_{i=1}^n k(\|x_i^*\|^2)} \quad (2)$$

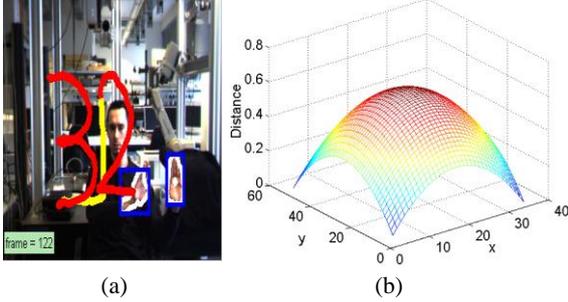


Figure 2. (a) Gesture path for number 32 that is generated from connecting hand centroid points. (b) The Epanechnikov monotonically decreasing kernel for the hand target model of first image.

For the hand target candidate in the next frame, Let $\{x_i\}$, $i=1 \dots n_h$ be the normalized pixel locations of the hand target candidate, centered at y in the current frame. Using the same kernel profile $k(x)$, but with bandwidth h . The probability of the feature $u=1 \dots m$ in hand target candidate histogram is computed as;

$$p_u(y) = F_h \sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right) \delta[b(x_i) - u] \quad (3)$$

where

$$F_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)} \quad (4)$$

Moreover, the Bhattacharyya coefficient [Kha06a] is more suitable to evaluate the similarity between the hand target model and the chosen candidate rather than many more commonly technique, such as histogram intersection. The maximization of the Bhattacharyya coefficient between the unit vectors \sqrt{q} and $\sqrt{p(y)}$ that representing the hand target model histogram and candidate model histogram respectively takes the following form;

$$\rho[p(y_0), q] = \sum_{u=1}^m \sqrt{p_u(y_0)q_u} \quad (5)$$

To find the best match of our hand target in the sequential frames, the Bhattacharyya coefficient is maximized, which means that we need to maximize the term;

$$\sum_{i=1}^n w_i k\left(\left\|\frac{y-x_i}{h}\right\|^2\right) \quad (6)$$

where h is the kernel's smoothing parameter or bandwidth and the weights w_i is given by;

$$w_i = \sum_{i=1}^n \sqrt{\frac{q_u}{p_u(y_0)}} \delta[b(x_i) - u] \quad (7)$$

Mean shift iteration uses the gradient of this similarity function as an indicator of the direction of hand's movement where the centroid point of hand candidate is shifted by Eq. 8;

$$y = \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} \quad (8)$$

Thereby, the hand motion trajectory so-called gesture path is generated from connecting the centroid points of hand regions (Fig. 2(a)).

3. KEY GESTURE SPOTTING

The task of locating key patterns from a stream of input signal is to find the start and end points of a meaningful gesture while ignoring the rest. Here, we discuss how to model gesture patterns discriminately and how to model non-gesture patterns effectively. Each reference pattern for Arabic numbers from 0 to 9 is modeled by gesture HMM and all other patterns are modeled by a single HMM called a non-gesture model (garbage model) [Yan07a, Lee99a], however, it is not easy to obtain the set of non-gesture patterns because there are infinite varieties of meaningless motion. Fig. 3 represents a simplified gesture spotting structure where the hand gesture path is projected into 3D-plane.

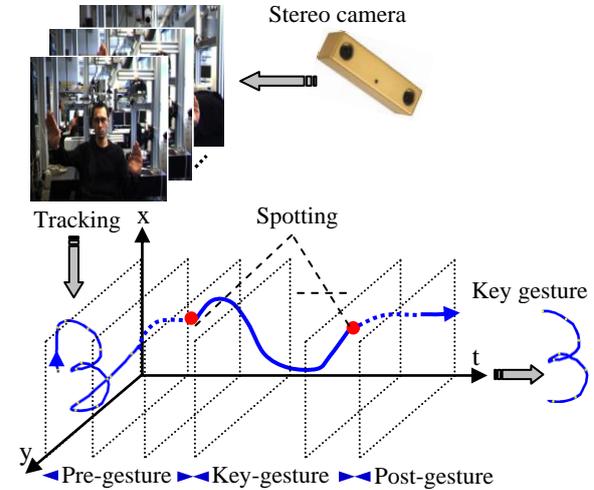


Figure 3. Gesture spotting structure, the dotted curve (Pre-and Post-) refer to non-gesture pattern and dark curve (Key-) represents gesture pattern.

3.1 Gesture Model

For each reference gesture model, each HMM state represents the local segmental part of it, while the states transition represent the sequential order structure in a gesture trajectory. The number of HMM states is an important parameter for each reference pattern because the excessive number of states can generate the over-fitting problem if the

number of training samples is insufficient compared to the model parameters. When there are insufficient number of states, the discrimination power of the HMM is reduced, since more than one segmented part should be modeled on one state. Moreover, the number of states in our gesture spotting system is based on the complexity of each gesture number and is determined by mapping each straight-line segment into a single HMM state (Fig. 4). In practice, we considered the Left-Right Banded topology (LRB) [Law89a] for the following reasons. Since each state in Ergodic topology has many transitions than Left-Right (LR) and LRB topologies, the structure data can be lost easily. On the other hand, LRB topology has no backward transition where the state index either increases or stays the same as time increases. In addition, LRB topology is more restricted rather than LR topology and simple for training data that will be able to match the data to the model. Orientation dynamic features are obtained from spatio-temporal trajectories and then quantized to generate its codewords (1-18). The quantized vectors are trained by Baum-Welch (BW) re-estimation algorithm [Law89a] for the initialized HMM parameters $\lambda = (II, A, B)$. For more details, the reader can refer to [Elm07a, Elm08a, Elm08b].

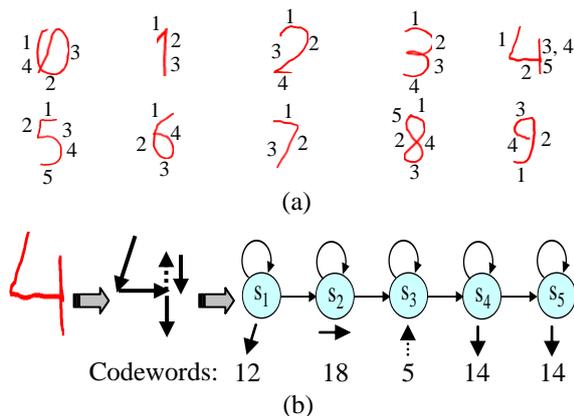


Figure 4. The hand gesture paths and straight-line segmentation. (a) The Gesture paths from hand motion trajectory for Arabic numbers (0-9) with its segmented parts. (b) The LRB topology with segmented line codewords for a gesture path 4.

3.2 Non-gesture Model

A non-gesture model represents any motion trajectory or any part of it other than gesture model. For correcting gesture spotting, the likelihood of a gesture model for a given pattern that is mentioned previously should be distinct enough. Unfortunately, the HMM recognizer selects a model with the best likelihood; we cannot ensure that the pattern is really similar to the reference gesture model unless the likelihood value is high enough. Thus, the non-gesture model is proposed where it provides a good confirmation for rejecting the non-gesture pattern.

The property of HMM internal segmentation denotes that each state with its self-transition represents a segmental pattern of a target gesture and the outgoing transitions represent a sequential progression of the segments in a gesture. With this property, we construct ergodic model with the states copied from all gesture models in our system, in addition two dummy states (Start state ST and End state ET), and then fully connect the states (Fig. 5). The dummy states are called null states, which observe no symbol and are passed without time delay [Yan07a, Pre98a]. We construct our non-gesture model as follows:

1. Duplicate all states from all gesture models, each with output observation probabilities. Then, we re-estimate that probabilities with Gaussian distribution smoothing filter to make the states represent any pattern.
2. Self-transition probabilities are kept as in the gesture models.
3. All outgoing transitions are equally assigned as;

$$\hat{a}_{ij} = \frac{1 - a_{ij}}{N - 1} \quad ; \text{for all } j, i \neq j \quad (9)$$

where \hat{a}_{ij} represents the transition probabilities of non-gesture model from state s_i to state s_j , a_{ij} is the transition probabilities of gesture models from state s_i to state s_j and N is the number of states in all gesture models. The non-gesture model is a weak model for all trained gesture models and represents every possible pattern where its likelihood is smaller than the dedicated reference model for a given gesture because of the reduced forward transition probabilities. Also, the likelihood of the non-gesture model provides a confidence limit for the calculated likelihood by other gesture models. Thereby, we can use confidence measures as an adaptive threshold for selecting the proper gesture model or gesture spotting. The number of states for non-gesture model increases as the number of gesture model increases. Furthermore, an increase in the number of states is nothing but dues to a waste time and space. To treat this problem, relative entropy [Cov91a] is used to reduce the non-gesture model states because there are many states with similar probability distribution.

3.3 Key Gesture Spotting & Recognition

In continuous hand motion, key gestures appear intermittently with transition connecting motion. To spot these key gestures in our system, we construct gesture spotting network as shown in Fig. 6. The gesture spotting network can be easily expanded the vocabularies by adding a new key gesture HMM model and then rebuilding a non-gesture model. This network contains ten gesture models for Arabic numbers from 0 to 9. These ten models are designed using LRB model with number of states ranging from 3 to 5 based on its complexity.

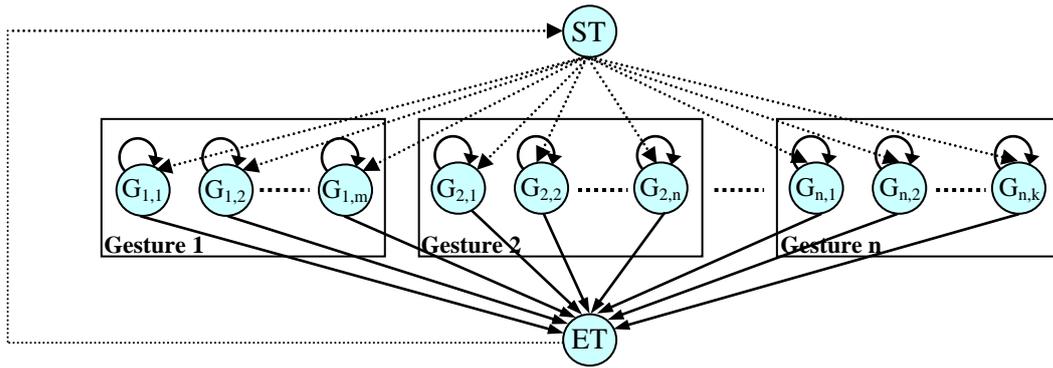


Figure 5. The general non-gesture model, where the dotted arrows represent null transitions, $G_{i,j}$ refers to the state j in gesture number i , ST and ET are the two dummy states for starting and ending, receptively.

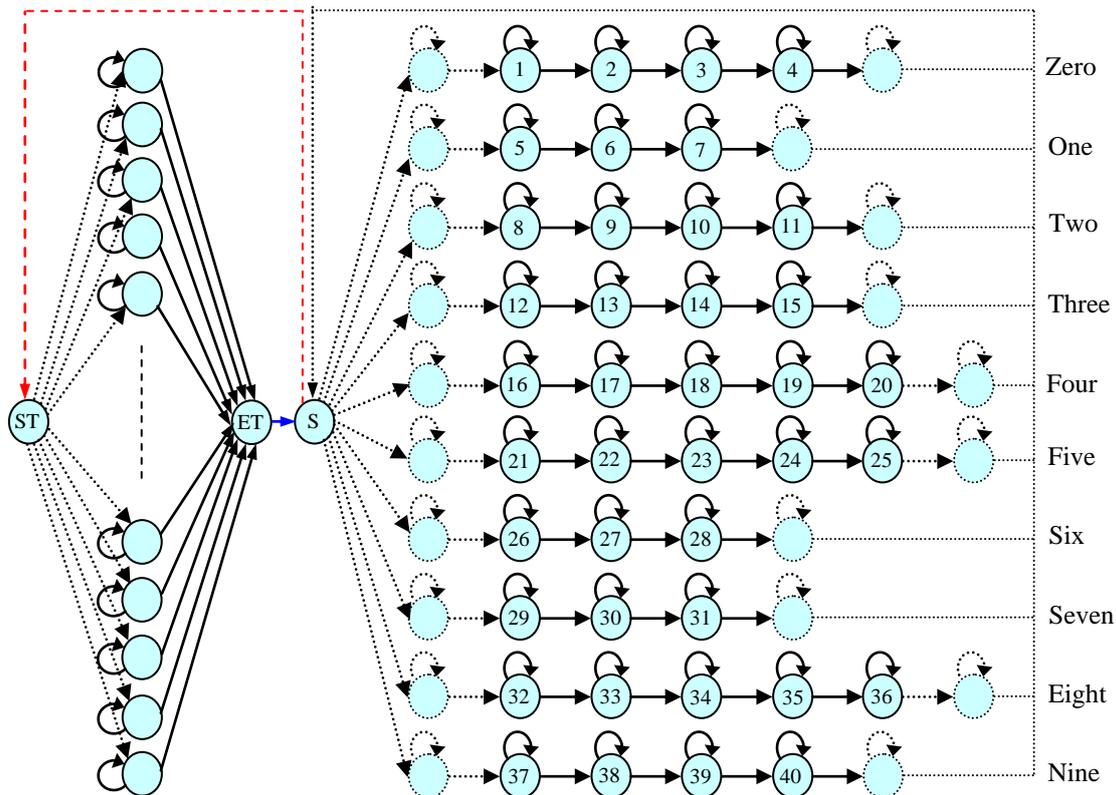


Figure 6. The gesture spotting network, where it contains ten Arabic number gesture models from 0 to 9 that are designed by using LRB model with varying states from 3 to 5 and also contains the non-gesture model after states reduction by relative entropy.

Also, it contains non-gesture model after states reduction by relative entropy function and the dummy start state S. The gesture spotting network finds the start and end points of key gestures that is embedded in the input stream and performs the segmentation and recognition tasks simultaneously. For forward spotting, we have defined a competitive differential observation probability value, which is determined by the difference observation probability value of maximal gesture models and non-gesture model (Fig. 7). The maximal gesture model is the gesture whose observation probability is the largest

among all ten gesture $p(O|\lambda_g)$. The transition from non-gesture to gesture occurs when the competitive differential observation probability value changes from negative to positive (Eq.10, then O can possibly as gesture g). Similarly, the transition from gesture to non-gesture occurs around the time that this value changes from positive to negative (Eq.11, then O cannot be a gesture). These observations can be used as a rule for detecting start and end point of gestures.

$$\exists g : p(o|\lambda_g) > p(o|\lambda_{non-gesture}) \quad (10)$$

$$\forall g : p(o|\lambda_g) < p(o|\lambda_{non-gesture}) \quad (11)$$

The proposed gesture spotting system contains two main modules (segmentation module and recognition module). In the gesture segmentation module, we use a sliding window technique, which calculates the observation probability of all gesture models and non-gesture model for observed segmented parts to spot the start point by competitive differential observation probability value. The optimal size of sliding window is determined empirically (equal 5 in our system) where the system is the best in term of results (Fig. 8(b)).

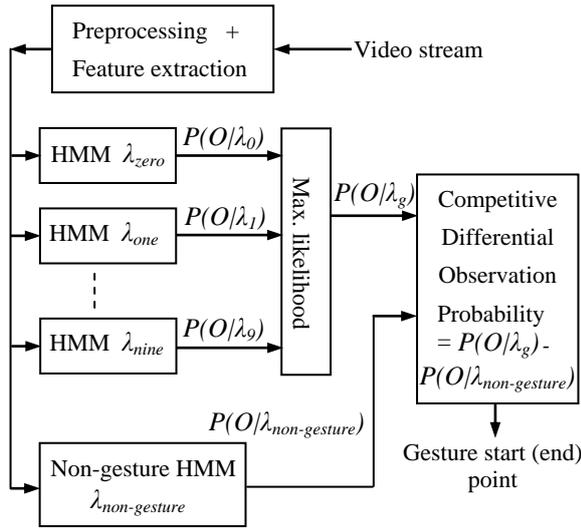


Figure 7. A block diagram shows how to calculate a competitive differential observation probability value between maximal gesture models for Arabic numbers from 0 to 9 and non-gesture model.

After spotting a start point in a continuous image sequences, then it activates gesture recognition module, which performs the recognition task for the segmented part accumulatively until it receives the end signal of a gesture. At this point, the gesture recognition module decides the type of observed gesture segmentation ($\text{argmax } P(O/\lambda_g)$) by Viterbi algorithm [Law89a]. This procedure is repeated until no input images exist. The following steps show how Viterbi algorithm works on gesture model λ_g ;

1. Initialization:

$$\delta_1^g(i) = \Pi_i b_i^g(o_1); \quad \text{for } 1 \leq i \leq N \quad (12)$$

2. Recursion (accumulative observation probability computation):

$$\text{for } 2 \leq t \leq T, 1 \leq j \leq N$$

$$\delta_t^g(j) = \max_i [\delta_{t-1}^g(i) a_{ij}^g] \cdot b_j^g(o_t); \quad (13)$$

3. Termination:

$$p(o|\lambda_g) = \max_i [\delta_T^g(i)] \quad (14)$$

where a_{ij}^g is the transition probability from state i to state j , $b_j^g(o_t)$ refers to the probability of emitting o

at time t in state j , and $\delta_t^g(j)$ represents the maximum likelihood value in state j at time t .

4. EXPERIMENTAL RESULTS

Our proposed system showed good results to recognize Arabic numbers in real-time from stereo color image sequences via the motion trajectory of a single hand using HMM. The system was implemented in Matlab language and the input images were captured by Bumblebee stereo camera system that has 6 mm focal length for about 2 to 5 second at 15 frames per second with 240×320 pixels image resolution. Our experiment is carried out an isolated gesture recognition test and key gesture spotting test.

4.1 Isolated Gesture Recognition

In our experimental results, each isolated gesture number from 0 to 9 was based on 60 video sequences, which 42 video samples for training by BW algorithm and 18 video samples for testing (Totally, our database contains 420 video sample for training and 180 video sample for testing). The gesture recognition module match the segmented gesture against database of reference gestures, to classify which class it belongs to. The higher priority was computed by Viterbi algorithm to recognize the numbers in real-time frame by frame over LRB topology with different number of states ranging from 3 to 5 based on its complexity. From table 1, the recognition ratio of isolated gestures achieved best results with 97.78%. The recognition ratio is the number of correctly recognized gestures over the number of input gestures (Eq.15). Fig. 8(a) shows the output of our system for isolated gesture number 8.

$$\text{Recognition ratio} = \frac{\# \text{ of correctly recognized gestures}}{\# \text{ of test gestures}} \times 100\% \quad (15)$$

4.2 Key Gesture Spotting Test

Our database includes on 280 video samples for continuous hand motion. Each video sample either contains one key gesture or more than one key gesture. Fig. 8(b) shows that the gesture spotting performance based on the size of sliding windows. So, we measure the gesture spotting accuracy according to different window size from 1 to 8. We noted that, the gesture spotting accuracy is improved initially as the sliding window size increase, but degrades as sliding window size increase further. Therefore, the optimal size of sliding window is 5 empirically, where the reliability of automatic gesture spotting system is 93.31% (Table 1). In automatic gesture spotting task, there are three types of errors, namely, insertion, substitution and deletion. The insertion error occurs when the spotter detects a nonexistent gesture. A substitution error occurs when

Gesture path	Train Data	Isolated gestures results			Key gestures spotting Results					
		Test	Correct	Rec.(%)	Test	Insert	Delete	Substitute	Correct	Rel.(%)
0	42	18	17	94.44	28	2	1	2	25	83.33
1	42	18	18	100.00	28	0	1	1	26	92.86
2	42	18	17	94.44	28	0	0	2	26	92.86
3	42	18	18	100.00	28	0	0	0	28	100.00
4	42	18	18	100.00	28	0	0	1	27	96.43
5	42	18	18	100.00	28	0	1	1	26	92.86
6	42	18	17	94.44	28	1	1	1	26	89.66
7	42	18	18	100.00	28	0	0	0	28	100.00
8	42	18	17	94.44	28	1	0	2	26	89.66
9	42	18	18	100.00	28	0	1	0	27	96.43
Total	420	180	176	97.78	280	4	5	10	265	93.31

Table 1. Isolated gesture recognition and key spotting gesture results for Arabic numbers from 0 to 9.

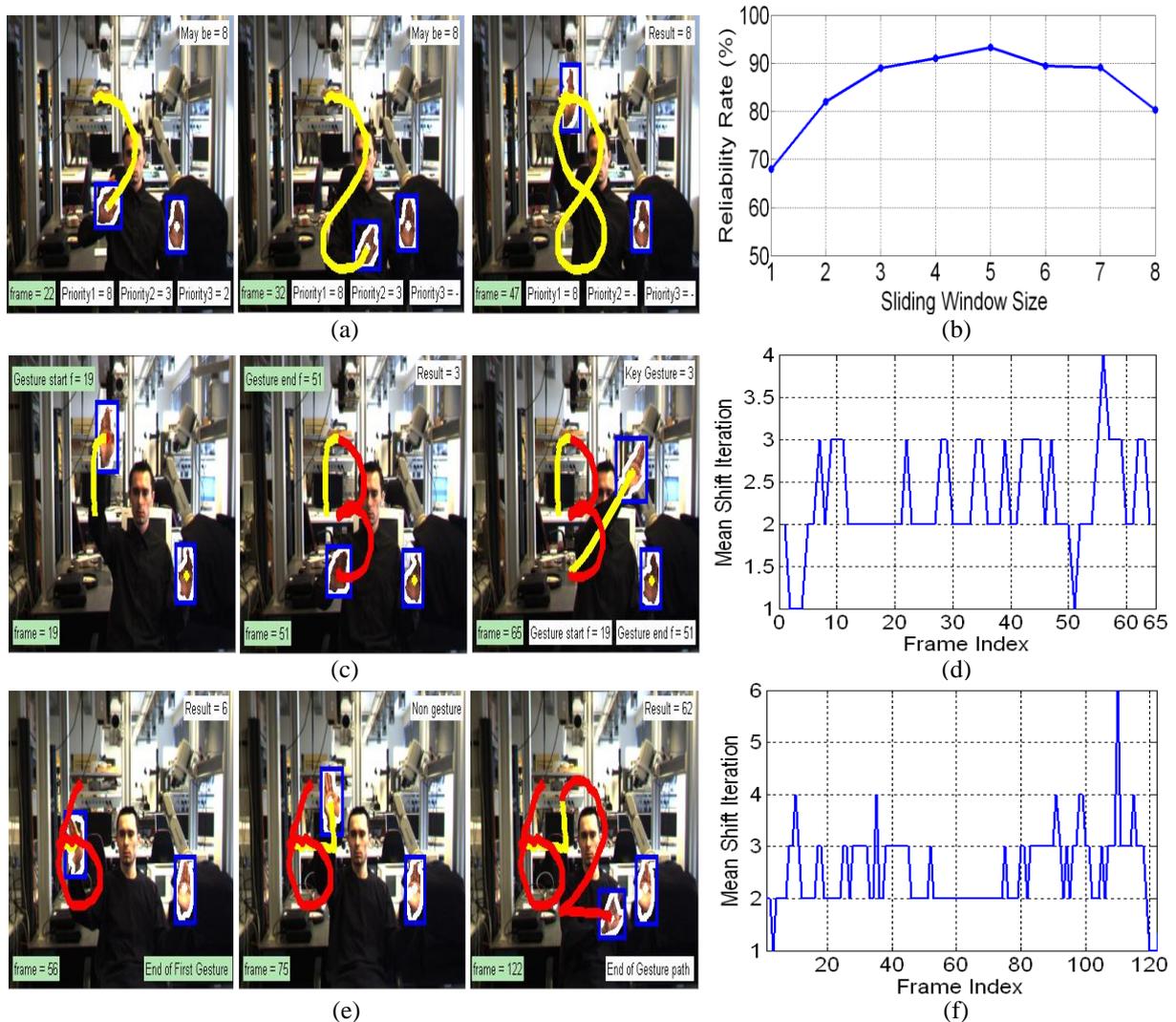


Figure 8. The system outputs. (a) Isolated gesture 8. (b) Spotting accuracy for different sliding window size from 1 to 8. (c) One key gesture spotting 3, where the start point at frame=19 and the end point at frame=51. (d) The number of mean shift iterations function (to connect hand centroid points) of the frame index for gesture path 3 where the mean number of iteration is a 2.29 per frame. (e) Gesture spotting 62. (f) The number of mean shift iterations function is 2.38 per frame for gesture path 62.

the key gesture is classified falsely. The deletion error occurs when the spotter fails to detect a key gesture. The reliability of gesture spotting system in terms of these errors is measured by Eq. 16.

$$\text{Reliability} = \frac{\# \text{ of correctly recognized gestures}}{\# \text{ of test gestures} + \# \text{ of insertion errors}} \quad (16)$$

Here, we note that some insertion errors cause the substitution errors or deletion errors where the insertion errors affect on the gesture spotting ratio directly. Fig. 8(c) and Fig. 8(e) show the results of key gesture spotting 3 and 62 respectively.

5. SUMMARY AND CONCLUSION

This paper proposes an automatic system that recognizes isolated gesture, in addition to key gesture spotting from continuous hand motion for Arabic numbers from 0 to 9 based on HMM. The proposed system describes the gesture spotting network, which finds the start and end points of key gestures that is embedded in the input stream by the difference observation probability value of maximal gesture models and non-gesture model. Our system uses forward spotting technique that performs the segmentation and recognition tasks simultaneously. This technique is suitable for real-time applications and solves the issues of time delay between segmentation and recognition tasks. Our results show that; an average recognition rate is 97.78% and 93.31% for isolated and key gestures spotting, respectively. In future, our research focuses on the motion trajectory will carried out by a fingertip using multi-camera system over combined features.

6. ACKNOWLEDGMENTS

This work was supported by Bernstein-Group (BMBF: 01GQ0702), LSA grants (C4-NIMITEK 2, FKZ: XN3621E/1005M) and DFG grants (SFB/TR 62, TP C1).

REFERENCES

- [Alo05a] Alon, J., Athitsos, V., Scharoff, S.: Accurate and Efficient Gesture Spotting via Pruning and Sungesture Reasoning. 3766 LNCS, pp. 189--198, 2005.
- [Com00a] Comaniciu, D., Ramesh, S., Meer, P.: Real-Time Tracking of Non-Rigid Objects Using Mean Shift. Conf. on Computer Vision and Pattern Recognition (CVPR), Vol. 2, pp. 1--8, 2000.
- [Com03a] Comaniciu, D., Ramesh, S., Meer, P.: kernel-Based Object Tracking. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 25(5), pp. 564--577, 2003.
- [Cov91a] Cover, T.M., Thomas, J.A.: Entropy, Relative Entropy and Mutual Information. Elements of Information Theory, pp. 12--49, 1991.
- [Dey06a] Deyou, X.: A Network Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG. ICPR 06, pp. 519--522, 2006.
- [Elm07a] Elmezain, M., Al-Hamadi, A., Michaelis, B.: Gesture Recognition for Alphabets from Hand Motion Trajectory Using Hidden Markov Models. IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pp. 1209--1214, 2007.
- [Elm08a] Elmezain, M., Al-Hamadi, A., Michaelis, B.: Real-Time Capable System for Hand Gesture Recognition Using Hidden Markov Models in Stereo Color Image Sequences. The Journal of WSCG'08, Vol. 16(1), pp. 65--72, 2008.
- [Elm08b] Elmezain, M., Al-Hamadi, A., Appenrodt, J., Michaelis, B.: A Hidden Markov Model-Based Isolated and Meaningful Hand Gesture Recognition. The International Conference on Computer Vision, Image and Signal Processing (CVISP2008), PWASET, Vol. 31, ISSN 1307-6884, pp. 394--401, 2008.
- [Kan04a] Kang, H., Lee, C., Jung, K.: Recognition-based Gesture Spotting in Video Games. Pattern Recognition Letters, Vol. 25(15), pp. 1701--1714, 2004.
- [Kha06a] Khalid, S., Ilyas, U., Sarfaraz, S., Ajaz, A.: Bhattacharyya Coefficient in Correlation of Gary-Scale Objects. Journal of Multimedia, Vol. 1(1), pp. 56--61, 2006.
- [Kim07a] Kim, D., Song, J., Kim, D.: Simultaneous Gesture Segmentation and Recognition Based on Forward Spotting Accumulative HMMs. The Journal of the Pattern Recognition Society, Vol. 40(2007), pp. 3012--3026, 2007.
- [Law89a] Lawrence, R.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceeding of the IEEE, Vol. 77(2), pp. 257--286, 1989.
- [Lee99a] Lee, H., Kim, J.: An HMM-Based Threshold Model Approach for Gesture Recognition. IEEE Trans. PAMI, Vol. 21(10), pp. 961--973, 1999.
- [Mit07a] Mitra, S., Acharya, T.: Gesture Recognition: A Survey. IEEE Transactions on Systems, MAN, and Cybernetics, pp. 311--324, 2007.
- [Nie07b] Niese, R., Al-Hamadi, A., Michaelis, B.: A Novel Method for 3D Face Detection and Normalization, Journal of Multimedia, ISSN,1796-2048, 2007.
- [Pre98a] Du Preez, J.A.: Efficient Training of High-Order Hidden Markov Models Using First-Order Representations. Comput. Speech Lang., Vol. 12(1), pp. 23--39, 1998
- [Sco92a] Scott, D., W.: Multivariate Density Estimation. Wiley, 1992.
- [Tak92a] Takahashi, K., Sexi, S., Oka, R.: Spotting Recognition of Human Gestures From Motion Images. Technical Report IE92-134, pp. 9--16, 1992.
- [Yan07a] Yang, H., Park, A., Lee, S.: Gesture Spotting and Recognition for Human-Robot Interaction. IEEE Transaction on Robotics, Vol. 23(2), pp. 256--270, 2007.

Simple emphatic user interface for virtual heritage

Stanek Stanislav
Comenius University
Mlynská dolina
84248, Bratislava, Slovakia
stanek@sccg.sk

ABSTRACT

We present a simple user interface combining h-Anim with Perlin's face. The main application is for exploring virtual environments especially those representing real environments with places, buildings or objects that belong to cultural heritage or those with historical past, famous story or something interesting. Therefore, information about them should be delivered to user. This kind of information is usually full of emotions and that is why the most suitable way (from user interface point of view) is to deliver it with emphatic storytelling. We are introducing our simple emphatic system (implementation uses ActiveX objects, VRML, ECMA Script, Java Script) that uses simple hardware configuration with web cams used for capturing user's presence and his/her head movements and if possible capturing position of some facial features, defined in MPEG-4 standard, and used to recognize user's simple emotions. User presence, head movements, and simple emotions are used to create simple emphatic user interface. In this paper we present our results already used in some application projects for virtual museums.

Keywords

Virtual environments, user interface, storytelling, emphatic, autonomous agents, capture, interaction

1. INTRODUCTION

We present a simple user interface combining h-Anim with Perlin's face. The main application is for exploring virtual environments (VEs) especially those representing real environments with places, buildings or objects that belong to cultural heritage or those with history, famous story or something interesting. This information is also usually full of emotions and that is why the most suitable way (from user interface point of view) is to deliver it with emphatic storytelling to user or visitor.

In the real world it would be a real person that is telling a story, answering questions, expressing emotions that come with story and also creating emotions that are arising from a simple conversation with a visitor. Emphatic communication between a storyteller and a visitor is in real world set very naturally this way.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

However, VEs [QVO01,QVO02] it is not so easy to set up this kind of emphatic communication. We are introducing our simple emphatic system that uses simple hardware configuration with web cams used for capturing user's presence and his/her head movements (position and orientation) and if possible capturing position of some of his/her facial features, defined in MPEG-4 standard, and used to recognize user's simple emotions. According captured information like user presence, head movements and simple emotions, our simple emphatic interface is creating actions and reactions (eg. starts/stops presentation) and introducing simple empathy this way. To achieve this emphatic feeling we are also using human-like autonomous agents as a part of this user interface. At this time we are working on middle precision autonomous agent based on minimal Perlin's face [PER00] structure and structure of its body is H-Anim 200x [HAnim] compatible with few extensions.

The rest of the paper is organized as follows. In background section some parts of recent project are described in more detail. We briefly describe previous work. Then we present our recent work and project in which recent results are applied. Finally, we outline future work.

2. BACKGROUND

Emphatic autonomous agents

Trying to construct empathic autonomous agents, we have to consider the immersion of a guest in VE and the immersion of the autonomous agent into the feelings of real visitors of VEs. Our idea is to limit the empathy of an avatar with respect to the well-known contexts. If the guest is very distant, no empathy makes sense. Strictly speaking, there are two sources of avatar's "emotionality". It is the message itself (sad story, ballad, funny paradox, irony...) and the guest's reactions. The first layer of emotions is ready in advance, say, off-line. We will use it for message presentation, e.g. while looking and pointing at the Crowning Tower of Bratislav Castle [VHCE04].

Another communication layer - guest's reactions - are available on-line only. They can be derived from the distance, head movements and facial expressions. This can be subdivided to data from user profile (ID, gender, etc. – data record) and guest behavior (navigation, exploration, interaction, cooperation, expressions, emotions ...).

The two layers of emotionality are combined using the algebra of facial expressions. We will weight the importance of the on-line layer. To clarify this we give two examples. Both start when tourist guide explains a story of a castle.

1. When a guest rotates his/her head, then the guiding autonomous agent will nonverbally respond to this. She/he will add to her/his head movement a half-way angle rotation. For this we have a metaphor of a delayed and lazy mirror.

2. When a guest quits his/her tour, then the guiding autonomous agent has to cancel the original role by saying "Goodbye and See you soon here in our castle".

The difference between the two examples is in a different use of the signal from guest. In the first case the response is added, whereas in the second case the importance of a signal is absolute and there is nothing to combine.

We will limit the movement of the guest and guiding avatar. The inspiration arose from Virtual Old Prague project [ŽÁRA02], where the city sectors are separated by invisible walls. Their touching works as sensors for fetching of another part of VRML/MySQL database. This improves the real-time rendering budget. Our idea is to surround visible objects (autonomous agents included) by multiple glass walls. The closest of them will prevent unreal close-ups and thus limit the needs for texture storage and processing. The area between two glass walls surrounding the autonomous agent can be

characterized in terms of rendering speed. The information tells to avatar which precision of empathy is practically needed. E.g. for a very distant user no facial expressions are needed. The middle distance signals the need for voice and simplified (Perlin) face. The nearest level of detail will start the complete use of MPEG-4 complaint full emphatic functionality.

The feasible combination of levels of details is under study (stories, photorealism, and autonomous agents empathy).

Modeling autonomous agents

The whole model of autonomous agent is hierarchical, segmented structure defined by joints and segments as specified in H-Anim standard representation for humanoids [HAnim]. So, the position and possible transformations per every segment using its joint are defined. For every segment there are defined also deformations that are used to create expressions of the segment.

Using Perlin's face we are able to create all basic facial expressions defined in MPEG-4 standard (anger, fear, sadness, surprise, joy and disgust) ([ABR99], Figure 1). Perlin's face uses minimal structure needed to create readable basic facial expressions. In the current work we are more concerned to create facial expressions and head movements in real-time for VE and that is why Perlin's minimal face is useful for us. It can be said that this face is defined, so to say, from approximate subset of feature points defined in MPEG-4 standard and that is why we refer to it as medium precision. We are planning for the future to extend this face to high precision face that will be compatible with MPEG-4 definitions and will contain all feature points defined in MPEG-4. Perlin's face structure in our application is transformed to satisfy H-Anim standard (consists of segments, joints, ...) Facial expression defined in Perlin's face model is transformed to deformations of face segments in H-Anim representation. And body structure also satisfies H-Anim standard. So our model is created as H-Anim humanoid model.

Using H-Anim standard for body representation allows us to use also body language to extend facial expressions with whole body expressions and also allows us this way to extend facial emphatic communication with emphatic communication created with body movements (for example head movements or hand gestures). The structure and functionality of body parts can be also referred to as a medium precision, because in high precision also their deformations and more complex structure will be considered. Using this standard it is also helpful for creating simple motions and emotions. Also the

application of pre-captured or real-time captured motion data is not complex. At this moment we are using pre-captured motions and expressions stored in a motions and expressions library.

VRML prototypes

We are concerned to VRML virtual environments and our autonomous agent is also VRML model and it uses H-Anim prototypes. We implemented in VRML its functionality (scripting with ECMA Script) using structure expressions that are created using other structure expressions or deform expressions or rotation expressions. Because model of autonomous agent is a hierarchical VRML structure consisting of segments that are substructures of the structure and any segment can have defined any expressions, we created functional prototypes for any type of expression. The prototypes are defined for reusability. We use these prototypes for export of created model to VRML file supporting defined functionality of the autonomous agent.

For our purposes we also extended H-Anim standard prototypes with some VRML nodes to achieve needed functionality. We introduced some extensions to Joint and Displacer prototypes that are parts of Humanoid prototype defined in H-Anim

As for the Segment node in H-Anim is defined Displacer node we introduced Rotator node for Joint node in H-Anim.

```
PROTO Joint [ ...
# Extended with:
  exposedField MFNode rotators [ ]
  exposedField SFNode rotationControler NULL
  exposedField MFNode StructureExpressions [ ]
  exposedField MFNode Sounds [ ]
]
```

```
PROTO Displacer [ ...
# Extended with:
  field SFNode Segment Segment {}
  field MFNode DeformExpressions [ ]
  eventIn SFNode set_DeformExpressionState
]
```

```
PROTO Rotator [
  exposedField SFString name ""
  exposedField SFInt32 ID -1
  exposedField SFRotation rotation 0 0 1 0
```

```
  exposedField SFFloat weight 0
  exposedField MFFloat weight_ranges [-1 0 1]
  exposedField MFRotation orientations [ ]
  field MFNode RotationExpressions [ ]
  eventIn SFNode set_RotationExpressionState
...
]
```

Prototypes RotationExpressions are used to create body and head motions and emotions (body language) applying rotational transformations to joints of avatar's body structure. Prototypes DeformExpressions are used to create facial expressions applying translation transformations to points of segments (segment deformations) of avatar's body structure. For example, DeformExpressions are also used to create mouth movements corresponding to actual words of an autonomous agent. At least to each phoneme defined in MPEG-4 there is corresponding DeformExpression.

Sound node in Joint node is used to correctly localize voice of an autonomous agent (it has to come from its mouth and if the avatar is far away, you are not able to hear it, but if you get closer to it you will hear the words it is saying. This functionality comes from specification of VRML files and using audio in VRML environments.).

Because of reusability with setting only some parameters we also created new prototypes StoryTeller, Timeline, TimelineAction and ActionEvent. Using these prototypes we are defining functionality that is needed to tell a story to a user. That means that these prototypes are used to store what to say, how to say it and when to say it. They are also taking care about all synchronizations and especially they are taking care about synchronizations of voice with lips, facial and body expressions.

This way, having textual representation of the audio with story, we have information about the mouth deformation sequence in time. Additionally we have to synchronize this deformation sequence with storytelling audio. So storytelling timeline has defined marks for this synchronization. Definition of these marks is at this moment done manually with only a simple automatic processing, but this will become automatic with integration with system for facial feature capturing.

All prototypes are defined so that we are able to do simple combinations (or interpolations) of any expressions.

We created this “language” also because we need to combine different motions that can be divided into at least two layers of motions respectively expressions or emotions. The first layer of motions and emotions is defined with the content of the story. The second one is defined by the environment in which the storytelling avatar actually is. This second layer also consists of motions and expressions that are created according to given information about position in environment and for example also information about position of an object that it is actually speaking about. To the second layer belong also expressions and movements created as reactions to captured and recognized expressions of user if capturing system is working.

All defined prototypes are using advantages of TimeSensors and ROUTEs that are defined in VRML specification.

3. PREVIOUS WORK

Our work is still in progress but some achieved partial results are already used in some projects. Final results will be used in many other running projects presenting real environments using its virtual approximation and each of these projects needs storytelling in VE.

Older results are already used in international project [VHCE04] that is dealing with cultural heritage in different countries. Our autonomous agents are in this project used as emphatic virtual storytellers or emphatic tour guides, telling stories with emotions achieved with body and head motions, facial expressions and with speech synchronized lips movements.

We created simple user interface for VEs that offers to see predefined viewpoints and predefined guided tour to user. Our autonomous agent is a key object in this interface. On the figure 1 you can see an example of created user interface used in VHCE project and description of it. Using this interface user can switch between predefined viewpoints, call storyteller, and turn on and off its emotions (humanlike behavior) and also to view predefined guided tour. Predefined guided tour is made of motion trajectory used as guide for flying around, close to or inside of presented objects and during this fly the storyteller is telling interesting story. This guided tour can be stopped, paused or played again at any time.

Emotional layer of autonomous agent is created in two sub layers. First one is defined by the story and its emotions and that is why these emotions are the same each time the storyteller is telling the story. We use predefined body and head expressions defined by rotations and facial expressions with moving lips

synchronized to speech defined by deformations and time stamps.

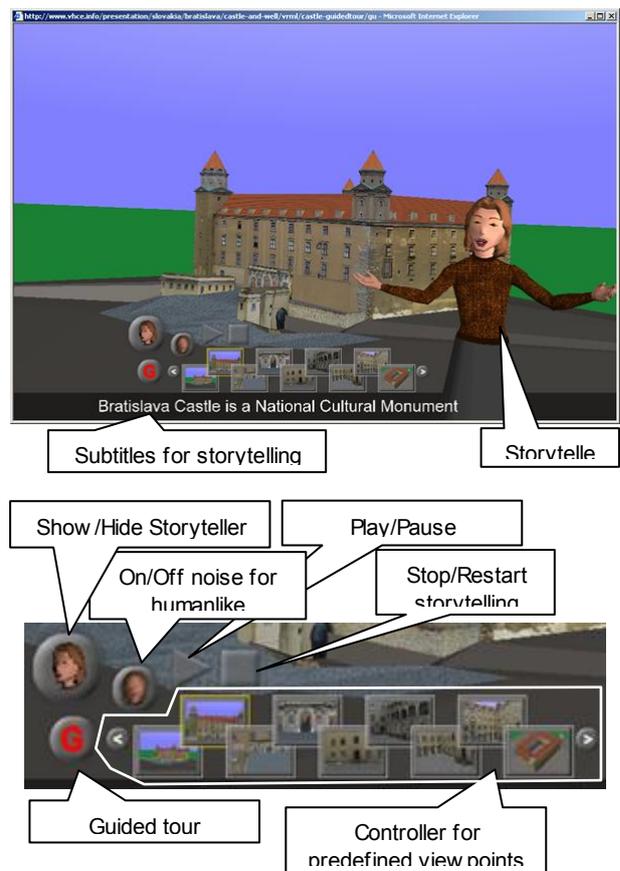


Figure 1. UI for VRML worlds with avatar (functionality description), Screenshot from project VHCE [VHCE].

Second layer is independent of the storytelling. It represents emotions according to situation in which the storyteller actually is (eg. position in VE or in the future also movements and emotions of visitor in VE). These emotions are created by applying facial expressions (deform expressions) and head and body expressions (rotational expressions) in time with weight that is processed with Perlin’s noise and that is why it look so random and natural. This second layer helps us to create autonomous agent that acts approximately like a real human.

For user interface in VE we created also graphical and functional prototypes that can be used to create custom interfaces depending on application or type of VE. All these prototypes are also used for creating of user interface for prototype GuidedTour in VHCE project that has integrated our empathic autonomous agent. These prototypes are nowadays used and extended to fulfill functionality needed for virtual museums (see next section).

4. VIRTUAL MUSEUM

The latest results of our research are planned to be used and implemented in many project. We will describe our latest results by describing the project “Považské múzeum 3D” where these results are implemented.

Project description

This project is dealing with creating virtual representation of a real museum called “Považské múzeum“ [PMZ08]. For this project we created special hardware configuration or “hot spot” or as we are saying “well of knowledge” where people can virtually visit this museum and get a lot of information about this museum (see Figure 2).

This design for the “hot spot” in this project has triple metaphorisation - a drop of water falling up combined with the view through a locked door, using just the keyhole into the 13th chamber, where the source of knowledge is hidden.



Figure 2. Gateway to virtual museum or “hot spot” or “well of knowledge”. (photos by Ela Šikudová)

The third metaphor is inside - the source of living water. The virtual water has sound rendering, as well, and after double-click it metamorphoses into a book, symbolic source of knowledge. The drop falling up symbolizes the visit of a museum. Go back in the time



Figure 3. Design metaphors and screenshots from presentation layer.

and causality and refresh your memory. The UNESCO page mission is credited here “Heritage is our legacy from the past, what we live with today, and what we pass on to future generations.” (see Figure 3)

Functionality

This “hot spot” is a gateway to virtual museum and is created for any kind of user and that is why the user interface should be as simple as possible. And also to be able to deliver information about this museum in a natural way we are introducing emphatic user interface achieved with built-in web cams that are capturing and monitoring situation close to display part. (see Figure 2.) In this hardware configuration according to captured space in front of the hot spot web cams have these three main functionalities:

1. Identify presence of a possible user
2. Identify head of a user and capture its movements
3. If possible identify facial features and try to identify users emotions

First two main functions are already working. First one is used to send presentation into a sleeping mode (like screensaver with black screen, in this project there is just a simulation of a well with water and with pure lighting) when there is no user detected (no motion is detected) or it will “wake up” presentation (starts displaying presentation on the screen, in this project it means that a simulation of a well with water and lighting is displayed) when there is a possible user detected.

Second one is used to identify that a person is present and that a book of knowledge should be rendered inside of the well under the water waiting for next interaction through touch display.

Touching the display is just simulating touching of water. This part is created with flash technology and ActionScript. After double-click on the top of the water it brings the book of knowledge in front of the water and user can choose from topics that he/she is interested in. Available topics are information about project partners and presentations about museum with simple user interface for choosing a presentation, to play and stopping, moving forward or back slides in presentation. Presentation and its slides are predefined and can consist of images and videos.

One of those presentations is created as a VE with 3D model of a castle where the museum is placed and our emphatic autonomous agent with predefined guided tour is ready to tell a simple story about this museum (Figure 4.). In this part second and third web cam functionalities are mainly used. Because we are using

web cams with low resolution, nowadays we are not successful with automatic facial features extraction and that is why we have also problem with identification of any simple emotion of present user.



Figure 4. Guided tour. Virtual environment without textures.

System functionality

In this section we will simply describe functionality of our system.

There is a simple block diagram of our system on the figure 5. We are using a web browser (Internet Explorer or Firefox) for which ActiveX objects are created and used. It is because in the future this system should be placed and work on the Internet.

For VEs created in VRML [VRML97] we use viewer *Cortona* (www.parallelgraphics.com) as ActiveX component in browser. It is used as one of many viewers for VEs created in VRML. We use it because it is also viewer that is commonly used. We created VRML prototypes for our user interface between user and VE (see VRML prototypes). Guided tour that consists of predefined viewpoints and autonomous agent is part of this interface.

There is another ActiveX object in browser created by *VideoForge* system [KUB06] that is responsible for capturing and analyzing captured data from cameras. VideoForge is working in real time and it should have actual information about presence of a user, his/her head movements and in the future also movements of facial features of the user. It is using simple and complex filter sequences for detection and features tracking. We created an *Interface* as a communication bridge between VideoForge and Cortona. This interface is nowadays for our testing

purposes just a simple set of JavaScript functions. VideoForge and Cortona are working separately. Interface is generating questions to Videoforge about user and answers to this questions are analyzed and the results are sent to Cortona where Guided tour respectively autonomous agent is responding to this results. Interface is also asking question to Guided tour and autonomous about actual viewing situation and depending on answer it is needed or not to send results or ask questions to VideoForge.

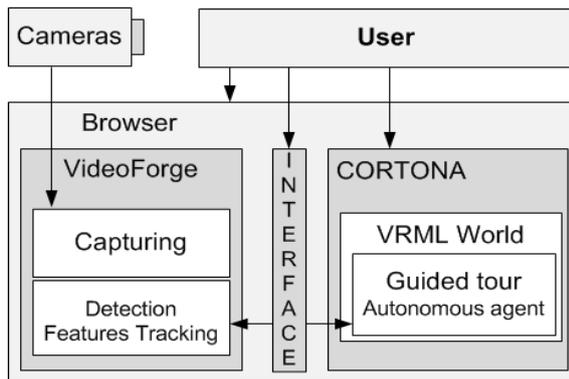


Figure 5. Simple block diagram of the system.

5. CONCLUSIONS

We are introducing our working results used in projects dealing with delivering information to users through virtual environments. Our work is still in progress and some work has to be done to fulfill our final aim to create fully working emphatic user interface. Here we presented only our partial working results, but those already used in practice.

We are planning for the future to extend this face to high precision face that will be compatible with MPEG-4 definitions and will contain all feature points defined in MPEG-4.

Now we already know that to successfully capture main feature points on user face we need to use cameras with higher resolution. This way we will need to have bigger processing power to work in real time. In the future this system should be placed and work on the Internet. We expect sending huge amount of data to users over Internet or establishing distant communication between users.

There is a group of people dealing with similar ideas how to extend user interface for viewing virtual environments with cultural heritage and they are also using capturing system. Their work is presented in [FRAN08]. Hopefully in close future we will exchange useful information and we will try to cooperate with

this group on a project dealing with user interfaces for virtual environments with cultural heritage.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the Scientific Grant Agency VEGA for supporting this work under the contract "Complexity of Geometric Algorithms for Real time rendering in Virtual reality" No. VEGA 1/3083/06.

7. REFERENCES

- [ABR99] ABRANTES, G. - PEREIRA, F. 1999. *MPEG-4 Facial Animation Technology: Survey, Implementation and Results*. IEEE CSVT vol. 9, no. 2, pp. 290-305, 1999.
- [HAnim] H-ANIM. Humanoid ANIMATION WORKING Group. <http://www.h-anim.org/> [online] Accessed October 23, 2008.
- [FER04] Ferko, A. et AL.. 2004. Virtual Heart of Central Europe. *CORP 2004*. www.corp.at. Vienna: TU Wien 2004. [online] Accessed October 23, 2008.
- [GEJ04] GEJGUŠ, P. – KUBÍNI, P. 2004. Face tracking using convolution filters and skincolor model. In *Proceedings of International Conference on Computer Vision and Graphics*. Berlin:Springer, 2006. ISBN 1-4020-4178-0. pp. 721-726. Warsaw, 22.-24.9.2004. PL
- [KUB06] KUBÍNI, P. 2006. VideoForge - An image processing system. In *Proceedings of Spring Conference on Computer Graphics 2006: Conference Materials Posters*. Bratislava: Comenius University, 2006. pp. 69-70 SCCG Častá-Papiernička , 20.-22.4.2006.
- [PAR96] PARKE, F. I. – WATERS, K. 1996. *Computer Facial Animation*, ISBN 1-56881-014-8, A K Peters. Ltd. 1996.
- [PER00] PERLIN, K. 2003. *Face demo applet using noise*. SIGGRAPH 2000. [online] <http://mrl.nyu.edu/~perlin/facedemo>. Accessed October 23, 2008.
- [PMZ08] PAGE OF Považské MÚZEUM ŽILINA. 2008. www.pmza.sk [online] Accessed October 23, 2008.
- [QVO02] QVORTRUP, L. ed. 2002. *Virtual Interaction: Interaction in Virtual Inhabited 3D Worlds*. London Berlin Heidelberg: Springer-Verlag 2002. ISBN 1-85233-516-5.
- [QVO01] QVORTRUP, L. ed. 2001. *Virtual Space: Spatiality in Virtual Inhabited 3D Worlds*. London Berlin Heidelberg: Springer-Verlag 2001. ISBN 1-85233-331-6.
- [STA02] STANEK, S., FERKO, A. 2002. Navigation and Interaction in Cyber Cities: Head Motions and

- Facial Expressions. Pp.75-78 in *Proceedings of Symposium on Computational Geometry SCG 2002*. Slovak University of Technology in Bratislava 2002. ISBN 80-227-1773-8
- [STA03] STANEK, S. - FERKO A. - KUBINI, P. 2003. Real-time Virtual Storytelling for Augmented Cultural Heritage: Message & Empathy. Pp. 45-46 in MAGNENAT-THALMANN, N. ed. *Proceedings for the first Research Workshop on Augmented Virtual Reality*. Organized by MIRALab, University of Geneva, September 18th-19th 2003, Geneva
- [VHCE04] International EC Culture 2000 project: *Virtual Heart of Central Europe*. www.vhce.info. [online] Accessed October 23, 2008.
- [VRML97] The Virtual Reality Modeling Language, VRML97. Functional specification and VRML97 External Authoring Interface (EAI) International Standard. ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2002.
- [ŽÁRA02] ŽÁRA, J. 2002. Concise Tour to the Virtual Old Prague.,. EUROGRAPHICS 2002, Short Presentations, Saarbrücken:. Eurographics Association, 2002, pp. 191-198.
- [FRAN08] FRANGESKIDES, F., LANITIS, A. and PAPANTONIOU, G. 2008. A contact-less interactive tool for exploring archaeological data. pp. 307-310 in *Digital Heritage - Proceedings of the 14th International Conference on Virtual Systems and Multimedia*. October 20th-25th 2008, Limassol, Cyprus